

Cours de Cryptographie

(version préliminaire 2005/2006)

Daniel Barsky

février 2006

Résumé

Le but de ce cours est une introduction à la cryptographie moderne utilisée dans la transmission et le stockage sécurisé de données. L'accent mis sur les principes et les outils mathématiques utilisés (arithmétique, algèbre, algorithmique, complexité, probabilité, théorie de l'information,..), ainsi que sur les protocoles.

Les problèmes informatiques, les produits et les normes sont décrits dans des cours plus appliqués (réseaux, sécurité réseaux,...)

Table des Matières

1	Introduction	5
1.1	Qu'est ce que la cryptographie	6
1.2	Qualités d'un cryptosystème	7
2	Historique	9
2.1	Codes à répertoire	9
2.2	Codes de permutation ou de transposition	10
2.3	Codes de substitution	12
2.3.1	Cryptanalyse des codes de César	13
2.4	Le code de Vigenère	13
2.5	Chiffrement de Hill	15
2.6	Commentaires historiques	15
3	Quelques méthodes de codage	17
3.1	Chiffrement en chaîne	17
3.2	Codes à confidentialité parfaite	17
3.3	Registres à décalage	18
3.3.1	Régistres à décalages	19
3.4	Lutte contre le brouillage	20
4	Sécurité d'un cryptosystème	22
4.1	Différentes notions de sécurité	23
5	Les codes modernes	24
5.1	Objectifs des codes actuels	25
5.2	Les familles de codes modernes	26
5.3	Codes symétriques	26
5.4	Codes asymétriques	27
5.5	Les échanges de clefs	28
5.5.1	Protocole d'échange de clefs	29

6 Applications de la cryptographie	30
6.1 Quel cryptosystème choisir	31
6.2 Quelques utilisations de la cryptographie	31
6.3 Quelles mathématiques pour la cryptographie	32
7 Codes à clefs secrètes	34
7.1 Description de DES	34
7.1.1 Quelques aspects techniques de DES	35
7.2 Description d'AES	38
7.2.1 Quelques aspects techniques d'AES	40
7.3 Infrastructure des systèmes à clef secrète	44
8 Codes à clefs publiques	46
8.1 Principe des codes à clef publique	46
8.1.1 Fonctions à sens unique	46
8.2 Le système RSA	47
8.2.1 Description du cryptosystème RSA	47
8.2.2 Protocole d'envoi d'un message en RSA	48
8.2.3 Protocole de signature RSA	49
8.2.4 Exemple académique de codes RSA	50
8.2.5 Exemple de code RSA	52
8.2.6 Sécurité du système RSA	54
8.3 Le cryptosystème El Gamal	55
8.3.1 Description du cryptosystème El Gamal	55
8.3.2 Signature El Gamal	56
8.3.3 Sécurité du système EL Gamal	58
8.3.4 Exemple académique de code El Gamal	58
8.4 Infrastructure des systèmes à clef publique	60
9 Fonctions de Hachage	63
9.1 Construction des fonctions de hachage	64
9.1.1 Attaques des anniversaires	65
9.1.2 Exemple académique de fonction de hachage	66
9.1.3 Fonction de hachage standard	66
10 Quelques protocoles cryptographiques	68
10.1 Protocoles de signature	68
10.1.1 Protocole de signature à clef privée	68
10.1.2 Protocole de signature à clef publique	69
10.2 Protocoles de datation	70

10.2.1	Protocole de datation	71
10.3	Signature avec fonction de hachage	71
10.4	Fonction de hachage et mot de passe	72
10.5	Preuve sans transfert de connaissance	73
10.5.1	Preuve sans transfert de connaissances	74
10.5.2	Transfert inconscient	75
11	Rappels Mathématiques	77
11.1	Théorie de l'information	77
11.1.1	Rappels de probabilités discrètes	77
11.1.2	Confidentialité parfaite	78
11.1.3	Entropie	80
11.2	Théorie de la complexité	81
11.2.1	Décidabilité	82
11.2.2	Complexité algorithmique	83
11.2.3	Algorithmes polynomiaux	84
11.3	Rappels d'arithmétique	86
11.3.1	La division euclidienne	86
11.3.2	Plus Grand Commun Diviseur	88
11.3.3	Algorithme du PGCD	89
11.3.4	Les Congruences	92
11.4	Tests de primalité	100
11.5	Méthode de factorisation	102
11.6	Rappels d'algèbre	104
11.6.1	Anneau des polynômes	104
11.7	Courbes elliptiques	113
11.7.1	Courbes Elliptiques sur un corps fini	117
	Bibliographie	120
	Index	121

Chapitre 1

Introduction.

Le but du cours est de donner une introduction à la cryptographie moderne utilisée dans la transmission et le stockage sécurisé de données. Après un rapide historique de la cryptographie on examinera les principaux systèmes cryptographiques actuellement utilisés

- Les codes par flots
 - les codes basés sur les registres à décalages
- les codes par blocs
 - Les systèmes à clef secrètes ou symétriques: AES avec un rappel sur DES et les systèmes dérivés
 - Les systèmes à clefs publiques ou asymétriques: RSA et El Gamal

L'accent sera mis sur les principes et les outils mathématiques utilisés (arithmétique, algèbre, algorithmique, complexité, probabilité, théorie de l'information,...). Néanmoins des protocoles seront décrits et on a ajouté une sensibilisation aux Infrastructures pour les Systèmes à Clef Publique (Public Key Infrastructures) et au systèmes de Management des Clefs Secrètes (Symmetric Keys Management).

La mise en oeuvre d'un cryptosystème à clef publique (public key infrastructure ou PKI) ainsi que celle d'un cryptosystème à clef secrète seront juste évoquées. Les problèmes de mise en oeuvre informatique, les produits et les normes sont décrits dans des cours plus appliqués (réseaux, sécurité réseaux,...).

On emploiera indifféremment les mots cryptographie, chiffrement et même parfois codage.

Ce cours s'est beaucoup inspiré des cours de François Arnaux, [2], Jean-Louis Pons, [14], et Guy Robin, [17], ainsi que des livres de Douglas Stinson, [23], Neal Koblitz, [11], John Daemen et Vincent Rijmen, [6], Benne de Weger, [25] et Gilles Zémor, [27], ainsi que de celui de Simon Singh, [21], pour la partie historique.

1.1 Qu'est ce que la cryptographie.

La cryptographie ou science du secret est un art très ancien, c'est finalement

l'art de remplacer un secret encombrant par un secret miniature

La *cryptographie* est l'art de rendre inintelligible, de crypter, de coder, un message à ceux qui ne sont pas habilités à en prendre connaissance. Le chiffre, le code est le procédé, l'algorithme, qui permet de crypter un message.

La *cryptanalyse* est l'art pour une personne non habilitée, de décrypter, de décoder, de déchiffrer, un message.

La *cryptologie* est l'ensemble formé de la cryptographie et de la cryptanalyse.

La cryptologie fait partie d'un ensemble de théories et de techniques liées à la transmission de l'information (théorie des ondes électro-magnétiques, théorie du signal, théorie des codes correcteur d'erreurs, théorie de l'information, théorie de la complexité,...).

Un expéditeur *Alice* veut envoyer un message à un destinataire *Bob* en évitant les oreilles indiscrete d'*Eve*, et les attaques malveillantes de *Martin*.

Pour cela Alice se met d'accord avec Bob sur le cryptosystème qu'ils vont utiliser. Ce choix n'a pas besoin d'être secret en vertu du principe de Kerckhoff. Par exemple ils peuvent décider d'utiliser un des systèmes suivants

- un code par blocs (block-cipher) à clef publique ou code asymétrique.
- un code par blocs à clef secrète ou code symétrique.
- un code par flots ou en continu (stream-cipher).

Chacun de ces systèmes dépend d'un ou deux paramètres de taille assez réduite (128 à 2048 bits) appelés la clef de chiffrement et la clé de déchiffrement. Les clefs de chiffrement et de déchiffrement n'ont aucune

raison d'être identiques. Seule la clef de déchiffrement doit impérativement être secrète.

Pour *crypter le message* \mathcal{M} Alice le découpe en blocs $(x_i)_{1 \leq i \leq n}$, de taille 1 pour des codes par flots, de taille 128 à 256 bits pour des codes par blocs comme AES.

Alice transforme chaque bloc x_i de texte en clair à l'aide de la *fonction de chiffrement*, e_{K_C} , dépendant d'une *clef de chiffrement*, K_C , en un bloc y_i , le texte chiffré.

À L'autre extrémité Bob *déchiffre, décrypte* le message codé à l'aide la *fonction de déchiffrement*, d_{K_D} , dépendant d'une *clef de décryptage ou de déchiffrement*, K_D .

1.2 Qualités d'un cryptosystème.

Alice veut être certaine

- qu'une personne non-autorisée (Ève) ne peut pas prendre connaissance de ses messages
- que ses messages ne sont pas falsifiés par un attaquant malveillant (Martin)
- que le destinataire (Bob) a bien pris connaissance de ses messages et ne pourra pas nier l'avoir reçu (*non-répudiation*)
- que son message n'est pas brouillé par les imperfections du canal de transmission (cette exigence ne relève pas du cryptage mais de la correction d'erreur)

Bob veut être certain

- que le message reçu est *authentique* c'est à dire
 - que le message n'a pas été falsifié par un attaquant malveillant (Martin).
 - que le messages vient bien d'Alice (autrement dit qu'un attaquant (Oscar) ne se fait pas passer pour Alice, *mascarade*)
- que l'expéditeur (Alice) ne pourra pas nier avoir envoyé le message (non-répudiation)

- que le message n'est pas brouillé (par les imperfections du canal de transmission ou par un brouillage intentionnel), autrement dit qu'il est identique à l'original que lui a envoyé Alice.

Les qualités demandées à un système cryptographique sont résumées par les mots clefs suivants:

- *Intégrité des données*: le message ne peut pas être falsifié sans qu'on s'en aperçoive
- Identité des interlocuteurs du message:
 - l'émetteur est sûr de l'identité du destinataire c'est à dire que seul le destinataire pourra prendre connaissance du message car il est le seul à disposer de la clef de déchiffrement.
 - *Authentication*, le receveur est sûr de l'identité de l'émetteur grâce à une *signature*
- *Non-répudiation* se décompose en 3
 - *non-répudiation d'origine* l'émetteur ne peut nier avoir écrit le message
 - *non-répudiation de reception* le receveur ne peut nier avoir reçu le message.
 - *non-répudiation de transmission* l'émetteur du message ne peut nier avoir envoyé le message.

Chapitre 2

Historique.

Il y avait historiquement deux grandes familles de codes classiques avec des hybrides

- Les codes à répertoire
- Les codes à clefs secrètes qui se subdivisent en deux familles
 - les codes de transposition ou de permutation qui sont des codes par blocs.
 - les codes de substitution qui peuvent être des codes par blocs ou par flots

On trouve des utilisations attestées par des documents historiques comme par exemple

- Scytale à Sparte vers -450, (principe des codes de permutation).
- Code de Jules César vers -50, (principe des codes de substitution).
- Les codes à répertoires, très anciens ont été utilisés intensivement jusqu'au début du 20-ième siècle.

2.1 Codes à répertoire.

Ces codes manquent de souplesse ils ne permettent pas de coder des mots nouveaux sans des échanges de codes entre l'expéditeur et le destinataire ce qui accroît le risque de voir le code intercepté. Ils ne sont plus utilisés pour les usages publics.

On peut par exemple créer le dictionnaire suivant:

rendez-vous ↔ 175	demain ↔ oiseaux
midi ↔ à vendre	Villetaneuse ↔ au marché

La phrase en clair:

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

devient avec ce code

175 OISEAUX À VENDRE AU MARCHÉ

Il faut donc disposer de dictionnaires qui prévoient toutes les possibilités. Donc sauf si on se restreint à transmettre des informations très limitées, la taille du dictionnaire s'accroît démesurément. Au 19^e siècle on avait ainsi pour des usages commerciaux ou militaires des dictionnaires de plusieurs milliers de mots de codes. Tout changement du code nécessitait l'envoi de documents volumineux avec un risque d'interception non négligeable.

2.2 Codes de permutation ou de transposition.

On partage le texte en blocs, on garde le même alphabet mais on change la place des lettres à l'intérieur d'un bloc (on les permute).

Un exemple historique dont le principe est encore utilisé dans les codes à clef secrète (DES, AES) est la *méthode de la grille* (principe de la *scytale* utilisé par les spartiates).

On veut envoyer le message suivant:

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

L'expéditeur et le destinataire du message se mettent d'accord sur une grille de largeur fixée à l'avance (ici une grille de 6 cases de large).

L'expéditeur écrit le message dans la grille en remplaçant les espaces entre les mots par le symbole □. Il obtient:

R	E	N	D	E	Z
□	V	O	U	S	□
D	E	M	A	I	N
□	M	I	D	I	□
V	I	L	L	E	T
A	N	E	U	S	E

Il lit le texte en colonne et obtient ainsi le message crypté:

R□D□VAEVEMINNOMILEDUADLUESIIESZ□N□TEC

Pour augmenter la sécurité les deux interlocuteurs peuvent décider l'ajout d'une clef.

Le but est de pouvoir changer facilement le cryptage d'un message tout en gardant le même algorithme de codage. Pour cela on rajoute une *clef secrète* constituée par l'ordre de lecture des colonnes.

Exemples 1. On choisit la clé: **CAPTER**

On numérote les colonne en fonction du rang des lettres du mot CAPTER dans l'alphabet c'est à dire

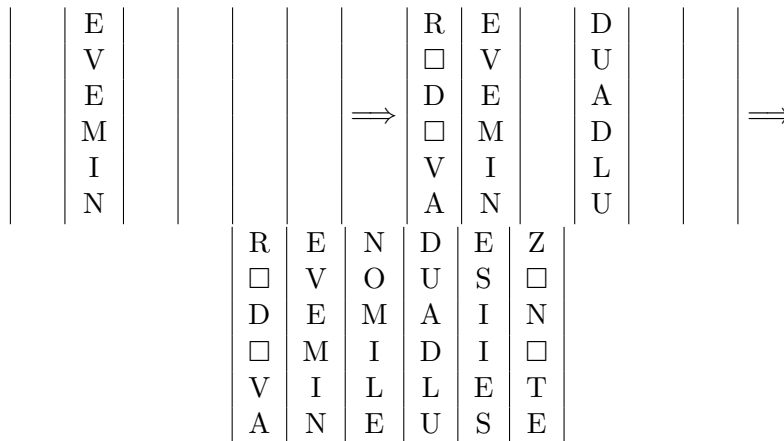
2, 1, 4, 6, 3, 5

et on lit les colonnes dans l'ordre indiqué.

EVEMINR□D□DADUADLUZ□N□TENOMILEESIIES

On a 6! codes différents.

Pour décoder le message précédent on range en colonne sur la grille en suivant l'ordre des colonnes donné par le mot de code



On a affaire à un code à clef secrète ou code symétrique car la clef de décodage est la même que la clef de codage.

2.3 Codes de substitution.

Dans les codes de substitution par flots ou par blocs l'ordre des lettres est conservé mais on les remplace par des symboles d'un nouvel alphabet suivant un algorithme précis.

Exemples 2. *Code de César:*

Pour coder on remplace chaque lettre par son rang dans l'alphabet.

$$A=1, B=2, C=3, \dots, M=13, N=14, \dots, S=20, \dots, X=24, Y=25, Z=26$$

D'après Suetone, Vie des douze Césars, Jules César pendant la guerre des Gaules avait utilisé le code de substitution par flot suivant

$$\text{lettre codée} = \text{lettre claire} + 3 \text{ modulo } 26$$

Le message en clair

RENDEZ VOUS DEMAIN MIDI VILLETANEUSE

devient

UHQGHC YRXV GHPDLQ PLGL YLOOHWDQHXXVH

Avantage: on peut considérer toute la famille des codes

$$\text{lettre codée} = \text{lettre claire} + n \text{ modulo } 26$$

où n est un entier entre 0 et 25 appelé la clef du code.

Avec la clef $n = 7$ le texte codé du message précédent devient:

YLUKLG CVBZ KLTHPU TPKP CPSSLAHULBZLBZL

Le *décodage* se fait en utilisant la relation

$$\text{lettre claire} = \text{lettre codée} - n \text{ mod } 26$$

On a encore affaire à un *code en continu ou par flot* symétrique ou à clef secrète.

2.3.1 Cryptanalyse des codes de César.

On considère un message codé avec une substitution monoalphabétique:

JTVMNKKTVLDEVVTLWTWITKTXUTLWJERUTVTWTHDXATLIUNEWV.
 JTIVIEWELOWENLNVNOEDJJTVLTPTXYTLWTWUT
 SNLITTVQXTVXUJXWEJEWTONKKXLT.

Décodage par *analyse de fréquence*.

Analyse de fréquence

Lettre	% français	% texte	Lettre	% français	% texte
A	9,4	1	N	7,2	5
B	1,0	0	O	5,1	2,5
C	2,6	0	P	2,9	1
D	3,4	2,5	Q	1,1	1
E	15,9	8	R	6,5	1
F	1	0	S	7,9	1
G	1	0	T	7,3	20
H	0,8	1	U	6,2	4,5
I	8,4	3,8	V	2,1	12
J	0,9	5,1	W	0	9,9
K	0	4,7	X	0,3	6
L	5,3	9	Y	0,2	1
M	3,2	1	Z	0,3	0

On peut donc faire l'hypothèse que T=E puis que V=S (à cause des lettres doublées) puis que les voyelles A, I, O, U correspondent à D,E, N, X et finalement on obtient la correspondance

A	B	C	D	E	F	G	H	I	J	K	L	M
D	R	O	I	T	S	H	M	E	F	G	J	K
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
L	N	P	Q	U	V	W	X	Y	Z	A	B	C

2.4 Le code de Vigenère.

La faiblesse des codes de César et des systèmes analogues est que la fréquence des lettres est conservée ce qui permet une cryptanalyse aisée par analyse de fréquences.

Pour améliorer la sécurité on peut faire un code de César par blocs dans lequel on change de substitution pour chaque lettre d'un bloc. On obtient

ainsi le *code de Vigenère*, mis au point par Leon Batista Alberti au 15-ième siècle et développé par Blaise de Vigenère.

Très sûrs pendant 4 siècles. Ils ont été cryptanalysés officiellement par Charles Babbage et Friedrich Wilhelm Kasiski au 19-ième siècle.

Le code de César et le code de substitution sont des *codes monoalphabétiques*. Le code de Vigenère est un code *polyalphabétique* ou *par blocs*.

- On se fixe une longueur de bloc m .
- On découpe le message en blocs de m -lettres.
- On chiffre par blocs de m lettres. On décide par exemple que la première lettre d'un bloc de m est codée avec un code de César de clef n_1 , la deuxième avec un code de César de clef n_2 et la m -ième par un code de César de clef n_m .

Exemples 3. $m = 5$, $n_1 = 3$, $n_2 = 14$, $n_3 = 7$, $n_4 = 22$, $n_5 = 19$, le message en clair est:

$M = \left\{ \text{Ce système de codage n'est pas sûr, mais plus que le code de César si la clé est longue} \right\}$.

on partage en blocs de taille 5 en partant de la gauche

CESYS TEMED ECODA GENES TPASS URMAI SPLUS QUELE
CODED ECESA RSILA CLEES TLONG UEXXX

Les XXX ont été ajoutés pour compléter le dernier bloc. La manière de compléter le dernier bloc peut être une faiblesse du code. Le message codé devient:

$e(M) =$ FSZUL WST**AW** HQTZT JSU**AL** WDHOL XFTWB VDSQL
TILHX FCK**AW** HQL**OT** UGPHT FZL**AL** WZVJZ XSETQ

La cryptanalyse du cryptosystème de Vigenère peut se faire, si le message est assez long, à texte chiffré connu, cf. le chapitre 4, en remarquant que des répétitions de lettres assez longues (3 au moins) doivent correspondre dans le texte clair à des répétitions de lettres aussi. Ceci permet de majorer la taille de la clé. On se ramène alors à la cryptanalyse d'un chiffrement de César.

Ici compte tenu de la longueur du texte on trouve les doublets **AW** et **AL** éloignés de 40 lettres et 45 lettres dont le PGCD est 5 ce qui suggère une clef de longueur 5.

Une fois que l'on a déterminé la longueur de la clef, le décodage est le même que celui de 5 codes de César.

2.5 Chiffrement de Hill.

Ce cryptosystème généralise celui de Vigenère. Il a été publié par L. S. Hill en 1929.

- On choisit un alphabet de n lettres (on prendra dans nos exemples $n = 26$) et une taille m pour les blocs, par exemple $m = 2$. Alors $\mathcal{P} = \mathcal{E} = (\mathbb{Z}/26\mathbb{Z})^2$, (en général $\mathbb{Z}/n\mathbb{Z}$).
- La clef de codage est une matrice inversible $K \in \text{GL}_m(\mathbb{Z}/26\mathbb{Z})$, si $m = 2$

$$K = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \text{GL}_2(\mathbb{Z}/26\mathbb{Z})$$

Si $(x_1, x_2) \in (\mathbb{Z}/26\mathbb{Z})^2$ est le message clair alors le message codé sera:

$$(y_1, y_2) = e_K((x_1, x_2)) = (x_1, x_2) \begin{pmatrix} a & b \\ c & d \end{pmatrix} = (ax_1 + cx_2, bx_1 + dx_2)$$

La clé de déchiffrement est la matrice inverse de K dans $\text{GL}_m(\mathbb{Z}/26\mathbb{Z})$.

Par exemple avec $m = 2$ et $K = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}$ alors $K^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$.

Le cryptosystème de Hill succombe facilement aux attaques à texte clair choisi.

2.6 Commentaires historiques.

Les méthodes historiques pour authentifier un message se retrouvent dans les cryptosystèmes modernes

On utilisait des messagers qui devaient échanger des signes de reconnaissance convenus (phrase de reconnaissance, lettres d'introduction, etc..) avec le destinataire pour authentifier le messenger, l'expéditeur ou le destinataire, pour éviter les faux messages.

Pour éviter la falsification du message on le mettait dans une enveloppe scellée revêtue de cachets, le message lui-même était signé, l'écriture permettait elle aussi l'authentification du message, etc...

Le destinataire pouvait parfois accuser réception en guise de protocole de non répudiation.

La cryptanalyse a été pratiquée depuis la plus haute antiquité. Le code de César a été cryptanalysé par les lettrés arabes du 9-ième siècle; Al Kindi a décrit la méthode de l'analyse statistique.

Tous les rois avaient leurs cryptographes-cryptanalystes (les Rossignol pour Louis XIV) et leur cabinet noir.

Les rapports entre cryptographie et cryptanalyse sont analogues aux rapports entre blindage et artillerie. Le code de César a tenu 9 siècle, celui de Vigenère a tenu 4 siècles, le standard DES a tenu 20 ans. le standard RSA est en passe d'être supplanté par les codes elliptiques.

Chapitre 3

Quelques méthodes de codage.

3.1 Chiffrement en chaîne.

Dans les systèmes cryptographiques que nous avons décrits les éléments de texte clair sont chiffrés de la même manière à partir de la clef K

$$y = y_1 y_2 \cdots = e_K(x_1) e_K(x_2) \cdots$$

On peut procéder différemment. On se donne une clé K et pour chaque entier i une fonction $f_i(K, x_1, \dots, x_{i-1})$ qui dépend de K et des blocs précédents x_i .

La valeur $K_i = f_i(K, x_1, \dots, x_{i-1})$ est la clé utilisée pour coder le i -ième bloc x_i

$$\underbrace{y_1 y_2 \cdots}_{\text{texte codé}} = e_{K_1}(x_1) e_{K_2}(x_2) \cdots$$

Exemples 4. On prend

$$\mathcal{P} = \mathcal{E} = \mathbb{Z}/26\mathbb{Z}, \quad K = 8 \text{ et } f_1(K) = K \\ K_i = f_i(K, x_1, \dots, x_{i-1}) = x_{i-1}, \quad e_{K_i}(x_i) = x_i + K_i$$

Le chiffrement est dit *synchrone* si f ne dépend pas des x_i , il est dit *asynchrone* si f dépend effectivement des x_i .

3.2 Codes à confidentialité parfaite.

La théorie de l'information due à Claude Shannon permet de définir un *code à confidentialité parfaite*, voir la définition 11.1.6 page 79. Grossièrement il s'agit de codes tels que la connaissance du texte crypté ne permette d'avoir aucune information sur le texte en clair.

De tels codes existent, ce sont les *codes de Vernam* ou *codes à masques jetables*, (1917). Ils reposent sur la fabrication de clefs ‘au hasard’ aussi longues que le texte à coder et qui ne servent qu’une fois.

La mise en oeuvre de ces codes est lourde et délicate. La fabrication de clefs ‘au hasard’ est un problème très difficile et mal résolu. Le stockage et la transmission de ces clés posent un problème de sécurité.

Ils sont rarement utilisés.

Le principe de ces codes est le suivant. On transforme le texte en une suite de chiffres en base b (souvent $b = 2$). On fabrique ensuite une suite aléatoire de chiffres de même longueur et l’on ajoute les deux suites ainsi obtenues sans retenue, c’est à dire que l’on fait le calcul chiffre à chiffre modulo b .

Exemples 5. On transforme les lettres de l’alphabet en nombres de 1 à 26 donc ici $b = 26$

$$\text{lettre codée} \equiv \text{lettre claire} + \text{lettre de la clé} \pmod{26}$$

On code le message *chiens* qui comporte 6 lettres avec la clef *KZUTEG*

$$(\text{message}=\text{CHIENS}) + (\text{clef}=\text{KZUTEG}) = \text{NHDYSZ}$$

On constate que si l’on avait envoyé le message *cerise* avec la clé *KCNPZC* on aurait obtenu

$$(\text{message}=\text{CERISE}) + (\text{clé}=\text{KCNPZC}) = \text{NHDYSZ}$$

Il est donc impossible de remonter au texte clair si on change de clé aléatoirement à chaque fois.

Cet exemple montre que la confidentialité est parfaite. Ces codes sont très sûrs. Mais leur mise en oeuvre est très délicate et ils sont très lents.

3.3 Registres à décalage.

On a construit des codes par flots qui cherchent à imiter les codes de Vernam mais qui sont très rapides et très faciles à mettre en oeuvre afin de pouvoir coder et décoder à la volée en temps réel (Télévision,...).

Une famille de tels codes est basée sur les *registres à décalages* ou *suites récurrentes linéaires sur un corps fini* ou encore dans la terminologie anglo-saxonne les *linear feedback register* ou en abrégé *LFBR*.

On fabrique avec les registres à décalage des *suites pseudo-aléatoires* pour générer la clef du code. Ces codes sont donc beaucoup moins sûrs que les codes de Vernam mais ne nécessitent pas la fabrication et la transmission d'une clef aléatoire de la longueur du message à transmettre.

3.3.1 Régistres à décalage ou suites récurrentes linéaires sur un corps fini.

Pour des raisons techniques le corps fini considéré sera le corps à 2 éléments $\mathbb{F}_2 \simeq \mathbb{Z}/2\mathbb{Z}$, c'est à dire que $\mathbb{F}_2 = \{0, 1\}$, muni des deux lois $+$ et \times

$$\begin{aligned} 0 + 0 = 0, & \quad 1 + 0 = 0 + 1 = 1, & \quad 1 + 1 = 0 \\ 0 \times 0 = 0 & \quad 1 \times 0 = 0 \times 1 = 0 & \quad 1 \times 1 = 1 \end{aligned}$$

ces deux lois sont associatives et la loi $+$ distributive par rapport à la loi \times , c'est à dire:

$$(a + b) + c = a + (b + c), \quad (a \times b) \times c = a \times (b \times c), \quad (a + b) \times c = (a \times c) + (b \times c).$$

On se donne un entier m , la longueur de la récurrence, et des éléments

$$\begin{aligned} k_1, \dots, k_m &\in \mathbb{Z}/2\mathbb{Z}, & \text{les conditions initiales} \\ c_0, \dots, c_{m-1} &\in \mathbb{Z}/2\mathbb{Z} & \text{les coefficients de la récurrence} \end{aligned}$$

On construit une suite d'éléments $(z_i)_{i \in \mathbb{N}}$ de $\mathbb{Z}/2\mathbb{Z}$:

$$\begin{aligned} z_1 = k_1, \dots, z_m = k_m, & \quad z_{m+1} = c_0 z_{m_1} + \dots + c_{m-1} z_m \\ z_{m+i} = c_0 z_i + \dots + c_{m-1} z_{m-1+i} &= \sum_{j=0}^{m-1} c_j z_{j+i} \end{aligned}$$

Il est facile de montrer que la clef est périodique (il existe T tel que $z_{i+T} = z_i$ pour i assez grand); mais si on choisit bien les k_i et les c_j la période est grande devant m .

Exemples 6. Exemple: $m=4$, $(k_1, k_2, k_3, k_4) = (1, 0, 0, 0)$ et

$$z_{i+4} = (z_i + z_{i+1}) \pmod{2}$$

On vérifie que la période est 15.

On considère alors le chiffrement en chaîne (synchrone ou asynchrone) dont la clef est $K = z_1 z_2 \dots z_n \dots$. On code alors en effectuant un *XOR* sur le message avec la clé K . C'est à dire qu'on fait une somme modulo 2 (sans retenue) d'un bit du message et du bit de même rang de la clef.

Exemples 7. Chiffrement en chaîne synchrone par XORisation:

message : 10010111010011000
 clef : 10100111010001011
 message codé : 00110000000010011

Ce type de codage se prête très bien au codage en temps réel d'images, de sons. La clef peut être construite par des registres à décalage qui peuvent être facilement câblés. Sous la forme indiquée ce système de codage est peu sûr à cause de son caractère linéaire..

Pour le rendre plus sûr on utilise n registres à décalages que l'on combine entre eux à l'aide d'une fonction booléenne non-linéaire c'est à dire une fonction non linéaire, f , du corps fini à 2^n -éléments \mathbb{F}_{2^n} que l'on identifie avec un \mathbb{F}_2 -espace vectoriel de dimension n et à valeurs dans le corps à deux éléments $\mathbb{F}_2 \simeq \mathbb{Z}/2\mathbb{Z}$. Cette fonction est appelée *fonction de combinaison* ou *fonction de filtrage*. On lui demande de satisfaire certains critères cryptographiques.

Remarquons que le cardinal de l'ensemble, \mathcal{B}_n , des fonctions booléennes de \mathbb{F}_{2^n} dans \mathbb{F}_2 est 2^{2^n} et qu'il croit extrêmement vite

n	4	5	6	7	8
\mathcal{B}_n	2^{16}	2^{32}	2^{64}	2^{128}	2^{256}

donc le choix de bonnes fonctions cryptographiques nécessite des preuves mathématiques.

3.4 Lutte contre le brouillage.

Le canal de transmission d'un message est souvent brouillé de manière naturelle (parasites dus aux activités humaines ou aux phénomènes naturels dans les transmissions électromagnétiques, hertziennes ou filaires).

Le stockage des données se dégrade naturellement avec le temps (action des particules très énergiques, poussières, rayures sur les CD et DVD, etc.).

Pour lutter contre tous ces brouillages on dispose de la théorie des codes correcteurs d'erreurs. elle est basée

- sur la théorie des espaces vectoriels sur un corps fini.
- sur la théorie des polynômes sur un corps fini.
- plus généralement sur la géométrie algébrique sur un corps fini.

Exemples 8. Exemples de codes correcteurs d'erreurs

L'exemple le plus simple de code correcteur d'erreurs est la répétition du message un certain nombre de fois en espérant que les parasites (supposés aléatoires) n'affecteront pas toujours la même partie du message. Mais cette méthode est lourde pour des messages longs et ralentit fortement la transmission des données.

Autre exemple de code correcteur d'erreurs: on suppose que le message est une suite de 0 et de 1 (des bits). On les groupe par paquets de 7 et on ajoute dans chaque paquet de 7 bits un huitième bit (0 ou 1) de telle sorte que la somme de ces 8 chiffres soit paire (code de Hamming). Ce code repère au plus une erreur dans chaque groupe de 8 bits.

Chapitre 4

Sécurité d'un cryptosystème.

Le principe de Kerckhoff stipule que

La sécurité d'un cryptosystème ne repose pas sur le secret du cryptosystème mais seulement sur la clef du cryptosystème qui est un paramètre facile à changer, de taille réduite (actuellement de 64 à 2048 bits suivant le type de code et la sécurité demandée) et donc assez facile à transmettre secrètement.

On suppose donc pour toutes les évaluations de sécurité d'un cryptosystème que l'attaquant connaît le système cryptographique utilisé, la seule partie secrète du cryptosystème est la clef.

Par exemple dans un cryptosystème basé sur des registres à décalage on suppose que l'attaquant connaît la forme des récurrences linéaires ainsi que la fonction de combinaison mais pas les conditions initiales des récurrences qui constituent la clef du code.

Les principaux types d'attaques:

- *attaque à texte chiffré connu*: l'opposant ne connaît que le message chiffré y .
- *attaque à texte clair connu*: l'opposant dispose d'un texte clair x et du message chiffré correspondant y
- *attaque à texte clair choisi*: l'opposant a accès à une machine chiffrente. Il peut choisir un texte clair et obtenir le texte chiffré correspondant y .
- *attaque à texte chiffré choisi*: l'opposant a accès à une machine déchiffrante. Il peut choisir un texte chiffré, y et obtenir le texte clair correspondant x .

4.1 Différentes notions de sécurité d'un cryptosystème.

- La *sécurité inconditionnelle* qui ne préjuge pas de la puissance de calcul du cryptanalyste qui peut être illimitée.
- La *sécurité calculatoire* qui repose sur l'impossibilité de faire en un temps raisonnable les calculs nécessaires pour décrypter un message.
- La *sécurité prouvée* qui réduit la sécurité du cryptosystème à un problème bien connu réputé difficile, par exemple on pourrait prouver un théorème disant qu'un système cryptographique donné est sûr si un entier donné n ne peut être factorisé.
- La *confidentialité parfaite* qualité des codes pour lesquels un couple (message clair, message chiffré) ne donne aucune information sur la clef.

Toutes ces notions de sûreté reposent sur la *théorie de l'information* de Claude Shannon. Il a pu donner un sens précis basé sur les probabilités à la première notion si l'on précise le type d'attaques permises. Il a aussi donné un sens précis à la notion de confidentialité parfaite.

La deuxième notion repose sur la *théorie de la complexité*. Dans la pratique il faut préciser le type d'attaque. C'est celle que l'on utilise dans la plupart des applications.

Chapitre 5

Les codes modernes.

Tout d'abord nous allons donner une description un peu formelle d'un cryptosystème

Définition 5.0.1. *Un cryptosystème est un dictionnaire entre les messages en clair et les messages chiffrés.*

Afin de pouvoir travailler efficacement sur les codes et définir de manière quantitative leur sécurité on est amené à les modéliser et donc à définir de manière axiomatique un cryptosystème. Définition formelle d'un cryptosystème

Définition 5.0.2. *Formellement un cryptosystème est un quintuplet*

$$(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$$

où

- \mathcal{P} est un ensemble fini de blocs : les mots en clair
- \mathcal{C} est un ensemble fini de blocs : les mots codés
- \mathcal{K} est un ensemble : l'espace des clefs
- Pour tout $K \in \mathcal{K}$ une règle de chiffrement $e_K \in \mathcal{E}$ et une règle de déchiffrement correspondante $d_K \in \mathcal{D}$. Chaque $e_K : \mathcal{P} \rightarrow \mathcal{C}$ et $d_K : \mathcal{C} \rightarrow \mathcal{P}$ sont des fonctions telles que $d_K(e_K(x)) = x$ pour tout $x \in \mathcal{P}$

Pour chaque $K \in \mathcal{K}$ (la clef du code) la transformation (bijective) e_K est appelée le procédé de chiffrement, ou l'algorithme de codage, entre les textes en clair \mathcal{P} et les textes codés \mathcal{C} .

La fonction inverse d_K de e_K est appelée la fonction de décodage.

On exige que tout message codé puisse être décodé.

Un code moderne est donc constitué

- D'un algorithme de chiffrement $f = f_{K_C}$, supposé connu de tous, dépendant d'un paramètre K_C , la clé de chiffrement. L'algorithme est fixé et public, seule la clé change, principe de Kerckhoff.
- De la valeur de la clé de chiffrement, K_C qui est secrète ou non suivant que l'on a affaire à un code à clef secrète ou à un code à clef publique.
- D'un algorithme de déchiffrement $g = g_{K_D} = f^{-1}$ (supposé connu de tous) dépendant d'une clé de déchiffrement, K_D , différente ou non de K_C .
- De la valeur de la clé de déchiffrement, K_D , qui est toujours secrète.

5.1 Objectifs des codes actuels.

- Les messages ne peuvent être déchiffrés que par le destinataire (confidentialité des données).
- Ils ne peuvent être modifiés par un tiers non autorisé (intégrité des données).
- L'identité des différents participants peut être vérifiée (authentification).
- L'expéditeur ne peut nier avoir émis le message et le destinataire ne peut nier l'avoir reçu (non-répudiation des données).
- Il doit permettre de coder et décoder rapidement (en temps réel pour certaines applications).
- Il doit au moins résister aux attaques connues et si possible avoir une sécurité prouvée.

Il n'existe pas de codes qui réunissent toutes ces qualités simultanément. Il faut donc un compromis adapté à chaque situation.

5.2 Les familles de codes modernes.

Les principales familles de codes modernes sont

- Les codes par flot (registres à décalage)
- les codes par blocs $\left\{ \begin{array}{l} \text{les codes symétriques (ou à clé secrète).} \\ \text{les codes asymétriques (ou à clé publique)} \end{array} \right.$

Les codes à clé publique sont basés sur la notion de fonction à sens unique.

5.3 Codes symétriques.

Les principes des codes symétriques commerciaux modernes du type DES (Data Encryption Standard) ont été mis au point dans les années 1970 par IBM avec l'aide de la NSA (National Security Agency), ce sont des hybrides de codes de substitutions et de codes de transpositions.

Ils restent très sûrs avec des clés assez courtes de 128 bits à 256 bits. Leur sûreté est non prouvée. Leur cryptanalyse a fait des progrès (cryptanalyse linéaire et différentielle), ce qui permet de mieux cerner leur sûreté.

Les codes à clé secrète sont les plus employés actuellement car ils sont éprouvés, résistants, rapides et assez facile à mettre en oeuvre. Actuellement leur sécurité est garantie (mais non prouvée) avec des clés assez courtes de 128 à 256 bits pour le standard actuel AES.

Mais il faut échanger la clé secrète avec le destinataire d'où un risque. Ils nécessitent un grand nombre de clés. Il faut

$$\frac{n(n-1)}{2}$$

clés pour que n interlocuteurs puissent échanger des informations d'où un problème de fabrication et d'échange de clefs fiables.

Ils ont du mal à réaliser sans *tiers de confiance* le partage des clefs, la signature, l'authentification, et la non répudiation des messages transmis. Ils s'adjoignent parfois un code asymétrique pour cela, voir le cryptosystème *PGP*.

La génération actuelle de codes symétriques est le cryptosystème AES développé dans les années 2000 à la suite d'un appel d'offre international par John Daemon et Vincent Rijmen, [6]. Il utilise dans sa conception nettement plus de mathématiques que son prédécesseur le DES.

5.4 Codes asymétriques.

Les principes des codes asymétriques ont été mis au point par Diffie et Hellman en 1976. Ils ont dégagé la notion de *fonction à sens unique*. Ce sont des fonctions qui sont faciles à calculer c'est à dire *calculable en temps polynomial* en fonction de la taille des données mais tel que l'image réciproque (les antécédents) d'un élément est très difficile à calculer explicitement (au moins calculatoirement). Autrement dit on demande que la fonction inverse ne soit pas calculable en temps polynomial en fonction de la taille des données.

La première solution celle de Diffie, Hellmann et Merkle était basée sur le problème du sac à dos dont il a été démontré que c'est un problème de type *problème de type NP* (NP pour Non Polynomial).

C'est donc un problème dont la solution exige un calcul en *temps non polynomial* en fonction de la taille des données, sous réserve de la validité de la conjecture $P \neq NP$. On peut donc espérer qu'il permette de fabriquer un cryptosystème offrant une sécurité calculatoire élevée.

Malheureusement toutes les instances (réalisations) de ce problème ne sont pas de type NP. En particulier les réalisations pratiques de Diffie, Hellmann et Merkle utilisaient des *sacs à dos* qui n'étaient pas dans la classe NP. Leur cryptosystème a succombé à l'attaque des cryptanalystes (en particulier Shamir).

Dans la solution suivante, 1978, le *cryptosystème RSA* (*Rivest-Shamir et Adleman*), la fonction à sens unique sous-jacente est la multiplication des entiers qui appartient à la classe P (P pour Polynomial) des *problèmes polynomiaux en temps*. Sa fonction réciproque la *factorisation des entiers* est actuellement dans la classe NP des *problèmes non polynomiaux en temps*, c'est un **fait d'expérience pas un théorème**.

D'autres solutions sont apparues peu après: le système El Gamal, les codes basés sur les courbes elliptiques.

Les cryptosystèmes asymétriques reposent sur des structures mathématiques élaborées. Leur sûreté est bonne mais non prouvée. Leur cryptanalyse dépend beaucoup des progrès des mathématiques correspondantes.

Sauf pour les codes elliptiques, ils nécessitent des clés longues (1024 à 2048 bits) pour avoir une sûreté équivalentes à celle d'un cryptosystème à clef secrète de 128 à 256 bits. Ils sont 1000 à 1500 fois plus lents. Mais ils permettent de réaliser facilement les fonctionnalités suivantes

- partage des clefs,
- authentification
- intégrité
- non répudiation

et bien d'autres encore grâce au fait que la clef est en fait constituée de deux clefs la clef de codage et la clef de décodage et que cette dernière est attachée à une personne et la caractérise.

Ils ne nécessitent que n paires de clés (une clé secrète et une clé publique) pour n interlocuteurs.

On les utilise souvent pour la transmission des clés secrètes des codes symétriques.

5.5 Les échanges de clefs.

Un problème important rencontré dans l'utilisation d'un cryptosystème est l'échange des clefs entre des interlocuteurs. C'est un des moments où la sécurité du cryptosystème est la plus vulnérable.

Rappelons que dans un cryptosystème à clef secrète lorsque deux interlocuteur veulent converser il leur faut échanger une clef. Il faut donc autant de clefs qu'il y a de paires non ordonnées d'interlocuteurs c'est à dire pour n interlocuteurs $\binom{n}{2} = \frac{n(n-1)}{2}$. Le nombre de clefs à échanger entre n interlocuteurs est donné par le tableau suivant

nombre d'interlocuteurs	nombre de clés secrètes	nombre de clés publiques
2	1	2
3	3	3
4	6	4
5	10	5
⋮	⋮	⋮
10	45	10
⋮	⋮	⋮
100	4950	100
⋮	⋮	⋮
1000	499 500	1 000
⋮	⋮	⋮
1 000 000	499 999 500 000	1 000 000

Par contre dans un système à clef publique il faut autant de paires de clefs qu'il y a d'interlocuteurs. Chaque interlocuteur dispose en effet d'une clef publique pour le cryptage des messages qui lui sont adressés et d'une clef privée pour le décryptage des messages qu'il reçoit.

5.5.1 Protocole d'échange de clefs.

Diffie, Hellmann et Merkle vers 1974 résolvaient le problème de partage d'une clé secrète sans envoi physique de la clef.

Leur protocole est le suivant:

1. Alice et Bob choisissent d'un commun accord sur une ligne non protégée un grand nombre premier p . et α un générateur de $\mathbb{Z}/p\mathbb{Z}$
2. Alice choisit a et calcule $\alpha^a = \alpha_a \pmod{p}$
3. Bob choisit b et calcule $\alpha^b = \alpha_b \pmod{p}$
4. Alice et Bob s'échangent α_a et α_b
5. Alice calcule $\alpha_b^a \pmod{p}$ et Bob calcule $\alpha_a^b \pmod{p}$

On a

$$\alpha_b^a \equiv \alpha^{ab} \equiv \alpha_a^b \pmod{p}$$

c'est la clef commune d'Alice et Bob.

Pourquoi ce protocole est-il sûr? Si Ève intercepte la communication entre Alice et Bob, elle peut lire les nombres α_a et α_b .

Pour calculer la clef commune d'Alice et de Bob, il faut qu'Eve puisse en déduire a et b c'est à dire qu'elle puisse calculer le *logarithme discret* dans \mathbb{F}_p .

Pour l'instant il n'existe pas d'algorithme rapide, c'est à dire polynomial en temps en fonction de la taille des données, pour calculer le logarithme discret dans \mathbb{F}_p . Les meilleurs algorithmes connus sont *sous-exponentiels*, cf. la sous-section 11.4. Pour un premier p de l'ordre de 10^{500} il faut des mois, voire des années, pour calculer un logarithme discret.

Par contre cet échange de clef peut-être soumis à d'autres types d'attaques.

Chapitre 6

Applications de la cryptographie.

Jusqu'au 20-ième siècle la cryptographie (légale) était essentiellement réservée aux militaires et aux diplomates et accessoirement aux industriels et banquiers.

Le développement des moyens de communications électromagnétiques facile à intercepter, des stockages de données confidentielles dans des sites assez faciles à pénétrer et sur des supports (bandes ou disques, magnétiques optiques) faciles à dupliquer ont généré une demande de cryptosystèmes sûrs faciles d'emploi et rapides pour des usages civils et commerciaux.

Les codes symétriques ou à clefs secrètes, type DES (Data Encryption Standard), IDEA (International Digital Encryption Algorithm), AES (Advanced Encryption Standard) ont été créés pour couvrir ces besoins. L'application première de la cryptographie est et reste encore la transmission sécurisée de données sensibles.

Le commerce en ligne, la consultation des données bancaires sur internet, le paiement par carte bancaire ont entraîné une explosion de l'utilisation de la cryptographie mais ont aussi une extension de ses applications. Cette extension a été facilitée par la mise au point des procédés de transferts de clefs de Diffie et Hellman et des codes asymétriques ou à clefs publiques de type RSA ou El Gamal.

Les codes à clef publique permettent de résoudre avec des protocoles légers les questions de transfert de clefs, d'identification, d'authentification, de signature entre correspondants.

Aujourd'hui les plus gros utilisateurs de cryptosystèmes sont les institutions financières (banques, bourses,...), les télécommunications (téléphonie mobile, WIFI, Internet,...), les sites d'achats en ligne,...

Il est probable que le *tatouge* (*watermarking*) des données numériques ainsi que la *gestion des droits numériques* (*digital right management* ou *DRM*) vont entraîner une demande accrue de cryptographie.

Le mariage de la cryptographie et de la théorie de la complexité aboutit à des extensions spectaculaires du champ de la cryptographie classique. Ces nouvelles applications sont développées au chapitre 10 page 68.

6.1 Quel cryptosystème choisir.

Chacun des cryptosystèmes existant codes par flots, codes par blocs à clé secrète, codes à clés publiques ont leurs avantages et leurs inconvénients. Ils ont chacun des applications privilégiées et sont souvent utilisés en association.

Les codes par flots basés sur les registres à décalage sont très utilisés en télécommunications (télévision à péage), à cause de leur rapidité. ils permettent de faire du codage et du décodage à la volée en temps réel.

Les codes par blocs à clé secrète sont les plus employés car ils réalisent un excellent compromis entre rapidité et sécurité. Ils ne nécessitent que des clefs assez courtes pour un bon niveau de sécurité (128/256 bits). Par contre ils nécessitent d'échanger de nombreuses clés secrètes avec un risque pour la sécurité. On peut les associer à un cryptosystème à clef publique pour échanger les clefs qui doivent être changées très régulièrement pour garder une bonne sécurité et aussi pour réaliser les signatures et authentifications.

Les cryptosystèmes asymétriques sont plus lents. Ils nécessitent des clés longues (1024/2048 bits) sauf les codes elliptiques (128/256 bits). Ils ne nécessitent pas d'échange de clés secrètes et permettent de réaliser facilement des fonctionnalités très utiles (signature, authentification, non répudiation,...).

6.2 Quelques utilisations de la cryptographie.

La nécessité de pouvoir identifier de manière sûre:

- le titulaire d'un compte en banque qui veut retirer de l'argent, ou qui paye par carte bancaire,
- un avion au moment de la procédure d'atterrissage, un véhicule suivi par GPS,

- sur un champ de bataille ses propres troupes et celles de l'ennemi,...

a entraîné le développement des *protocoles d'identification* largement basés sur les principes des codes asymétriques.

La recherche d'une sécurité accrue pour les sites sécurisés de paiement en ligne profite du développement des

- Les protocoles de preuves sans apport de connaissance.
- La signature aveugle.

basés sur la théorie de la *preuve sans apport de connaissance* (*zero-knowledge proof*).

La nécessité de conserver l'anonymat dans certaines situations (secret médical par exemple, secret de la vie privée, etc..) peut être assuré grâce au

- Le transfert inconscient.

Il y a encore bien d'autres applications.

6.3 Quelles mathématiques pour la cryptographie.

Actuellement tous les cryptosystèmes utilisent des mathématiques. Elles sont utilisées aussi pour définir et tester la sécurité des cryptosystèmes.

- Logique: théorie de la complexité.
- Théorie de l'information
- Probabilités: pour la théorie de l'information.
- Combinatoire (théorie de graphes): construction de cryptosystèmes asymétriques, preuves sans apport d'information, partage de secret à seuil.
- Algèbre, corps finis, polynômes sur un corps fini,...
- Théorie des nombres (arithmétique modulaire, théorie algébrique des nombres) : construction de cryptosystèmes asymétriques (RSA, El-Gamal), générateurs de nombres aléatoires.
- Géométrie algébrique sur un corps fini: construction de cryptosystèmes basés sur les courbes elliptiques sur un corps finis, cryptosystèmes basés sur les codes correcteurs d'erreurs.

- Algorithmique: mesure de la complexité algorithmique, réalisation pratique d'algorithmes performants.

La réalisation pratique des cryptosystèmes repose sur une implantation informatique, leur **degré de sécurité**, leurs performances en dépendent grandement.

Chapitre 7

Codes à clefs secrètes.

Devant l'explosion des besoins de cryptage pour des données non-classifiées (c'est à dire non militaires et non diplomatiques) le *National Bureau of Standards* des Etats-Unis a lancé un appel d'offre avec un cahier des charges en 1973.

Cet appel d'offre a donné naissance au cryptosystème *DES* (*Data Encryption Standard*). Il a été publié en 1975 et adopté par le NBS en 1977 comme standard de cryptage pour les applications non classifiées. Il reprend les principes et une partie du système de cryptage d'IBM denommé LUCIFER.

Il est à la base d'autres cryptosystèmes plus récents comme IDEA, FEAL, CAST, RC5, BLOW-FISH.

Il est remplacé aujourd'hui par *AES* (*Advanced Encryption Standard*) de J. Daeme et V. Rijmen, basé sur le même principe avec une clé plus longue (128 à 256 bits), plus structuré et avec des fonctionnalités plus étendues. AES a été retenu en 2000 après un appel d'offre international.

7.1 Description de DES.

C'est un *système cryptographique produit*, basé sur un *schéma de Feistel*. Il compose des opérations de cryptage, autrement dit il effectue un produit d'opérations de cryptage. Il répète 16 fois un algorithme la *fonction d'étage* qui dépend d'un paramètre la *clef d'étage*. La répétition de cet algorithme mélange les bits du message en clair en respectant les principes de C. Shannon: *confusion et diffusion*.

- La confusion gomme les relations entre le texte clair et le texte chiffré pour éviter les attaques par analyse statistique. Autrement dit un

changement d'un seul bit dans le texte clair doit affecter un grand nombre de bits (tous) du texte chiffré.

- La diffusion disperse la redondance du texte clair dans le texte chiffré, par exemple deux lettres doublées ne doivent pas rester côte à côte après cryptage.

7.1.1 Quelques aspects techniques de DES.

DES utilise une clé K de 56 bits utiles, en fait une clé de 64 bits dont les bits 8, 16, 24, 32, 40, 48, 56 ne sont pas utilisés pour la clé mais participent à un code correcteur qui permet de vérifier que la clé n'a pas été altérée.

DES est un cryptosystème par blocs qui travaille sur des blocs de 64 bits.

La clé de DES est trop courte pour les puissances de calcul actuelles. La taille de la clé secrète 56 bits le rend aujourd'hui vulnérable aux attaques par force brute. En 1997 la clé a été cassée en 3 semaines par une fédération de machines sur internet. Elle a été cassée en 56 heures en 1998.

En 1999 la clé a été cassée en moins d'une journée, 22 heures, grâce à un ordinateur dédié couplé avec un réseau de plusieurs milliers d'ordinateurs.

Afin de prolonger la durée de vie de DES en attendant AES on a introduit le triple DES. Il consiste à appliquer successivement au message DES muni de la clef K , à le décoder avec DES muni de la clef K' et de le recoder avec DES muni de la clef K . Au total tout se passe comme si on avait codé le message avec une clé nettement plus longue.

C'est un système de chiffrement itéré à 16 étages

1. La clé K donne naissance à 16 sous-clés, les clefs d'étage, de 48 bits K_1, K_2, \dots, K_{16}
2. Etant donné un bloc de texte clair de 64 bits, X , on construit, grâce à la fonction d'étage g_0 , un nouveau texte, X_0 , par permutation de l'ordre des bits grâce à la *permutation initiale* IP .

$$g_0(X) = X_0 = IP(X)$$

3. on sépare les 32 bits de gauche, L_0 , et les 32 bits de droites, R_0 , de X_0 , donc $X_0 = L_0R_0$

4. On effectue 16 itérations (ou tours ou étages). On calcule la valeur de la fonction d'étage $g(X_{i-1}, K^i) = L_i R_i$, $1 \leq i \leq 16$ suivant la règle

$$\begin{aligned}L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i)\end{aligned}$$

où \oplus est la somme dans $\mathbb{Z}/2\mathbb{Z}$ de L_{i-1} et $f(R_{i-1}, K_i)$ et où f une fonction non linéaire qui participe à la diffusion et qui est décrite par les *S-boîtes*

5. Pour finir on applique la permutation inverse de la permutation initiale, IP^{-1} , à $R_{16}L_{16}$

le schéma suivant résume ce qui précède, il est extrait du cours de Jean-Louis Pons, [14].

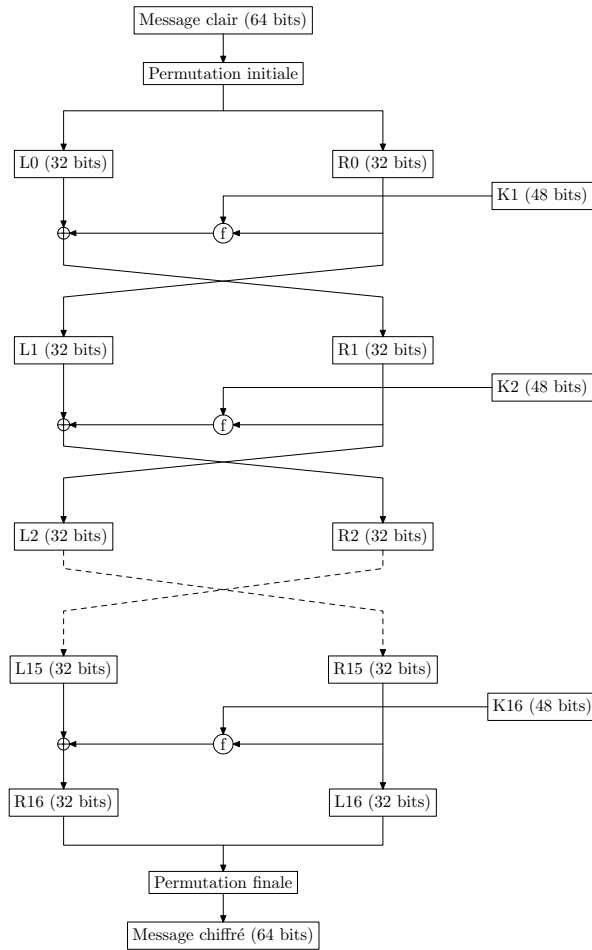


FIG. 1 – Architecture du DES

7.2 Description d'AES.

Le principe de l'AES, très proche du DES. C'est aussi un système cryptographique produit constitué d'une suite d'opérations de permutation et de substitution.

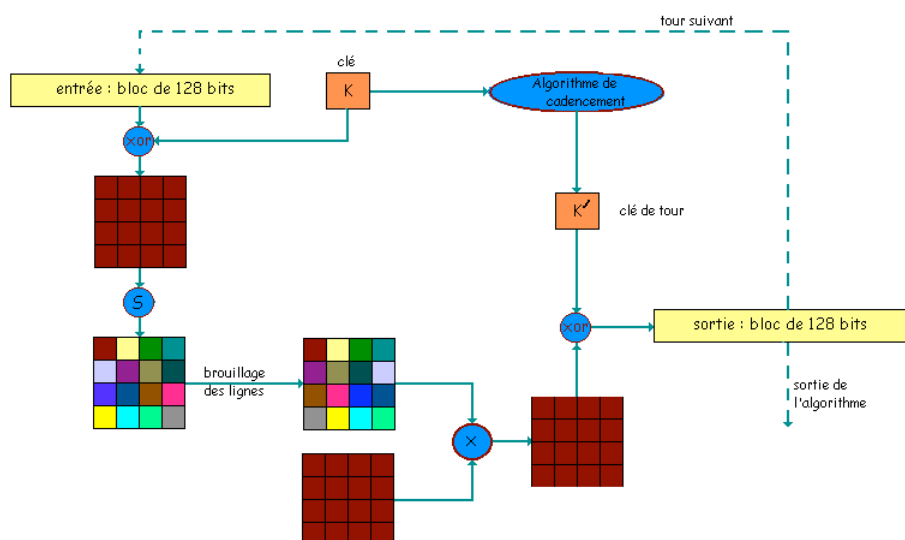
AES travaille sur des blocs de 128 bits avec des clefs de longueur 128, 256 ou 384 bits. Le passage à une clé de 128 bits minimum rend impossible dans le futur prévisible les recherches exhaustives de clefs. Si on suppose que l'on a un algorithme capable de comparer en une seconde 2^{56} clefs (i.e de casser DES en une seconde) il lui faudra 149 mille milliards d'années pour casser AES.

AES résiste à toutes les attaques connues (**attention ce n'est qu'un fait d'expérience pas une preuve**).

Les deux schémas qui suivent sont extraits respectivement des sites

<http://www.bibmath.net/crypto/moderne/aes.php3>

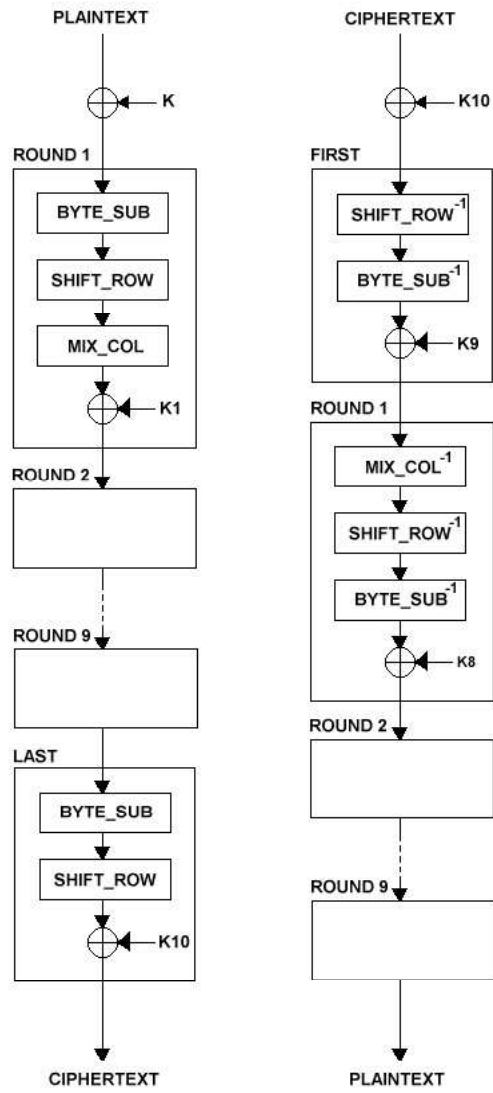
<http://www.securiteinfo.com/crypto/aes.shtml>



09/09/2005

file:/home1/barsky/enseignement/cours-...ours-2005/cours-2005/aes_files/aes.gif

#1



7.2.1 Quelques aspects techniques d'AES.

Pour toute la partie mathématique on pourra consulter dans les compléments mathématiques la section sur les extensions de corps et les anneaux de polynômes sur un corps fini, section 11.6 page 104.

AES est un *système de chiffrement itéré* constitué d'un algorithme de chiffrement, la *fonction d'étage*, répété de N_r fois, avec N_r allant de 10 à 14, et d'une clef d'une longueur de 128, 192 ou 256 bits pour chaque étage, la clef d'étage.

Un *algorithme de diversification de clef*, **ExpandKey** $[i]$, permet de créer une clef pour chacun des étages, i , à partir de la clef secrète K .

On notera N_e le nombre d'étages. La fonction d'étage, g , est l'ensemble des opérations que l'on fait subir au texte qui arrive à l'étage i . Elle est la même pour tous les étages sauf le dernier dans un soucis de simplicité.

La fonction g consiste en l'application successive de 4 opérations **SubBytes**, **ShiftRows**, **MixColumns**, **AddRoundKey**.

À partir de la clef secrète, K , on génère des sous clefs d'étage par l'algorithme de diversification de la clef, **ExpandKey** $[i]$. On obtient les clefs d'étage: K^1, \dots, K^{N_e} .

On note x le texte initial (texte en clair), w^i le texte utilisé en entrée à l'étage i et y sera le texte crypté final.

Tous les textes sont supposés transformés en une suite de zéros et de uns. Si K et L sont deux telles suites on note

$$K \oplus L$$

L'opération *XOR* entre la suite K et la suite L , c'est à dire l'addition sans retenue modulo 2 entre K et L , cf. page 20.

Avec ces notations la description schématique d'AES est la suivante:

1. On initialise w^0 à x et on effectue l'opération **AddRoundKey**

$$W^0 \oplus K$$

2. Pour chacun des $1 < i \leq N_e - 1$ étages suivants on effectue sur w^i les opérations suivantes
 - (a) l'opération de substitution **SubBytes**

- (b) sur le résultat de **SubBytes** une permutation notée **ShiftRows** (permutation circulaire sur les éléments des lignes de la matrice des octets $s_{i,j}$)
- (c) sur le résultat de **ShiftRows** une opération notée **MixColumns** (application linéaire sur l'espace des matrices sur \mathbb{F}_{2^8})
- (d) Sur le résultat de **MixColumns** l'opération **AddRoundKey**[i] avec la sous-clé de l'étage i

3. Pour le dernier étage N_e on supprime l'opération **MixColumns**

On va décrire maintenant chacune des opérations utilisées.

L'opération **ExpandKey**

L'opération **ExpandKey** dépend de la taille de blocs choisie, $N_b \times 32$, qui peut être égale à 128, 160, 192 ou 224, 256 bits, et de la taille de clé choisie, $N_k \times 32$, qui peut être égale à 128, 160, 192, 224 ou 256 bits. On supposera dans la suite que la longueur de la clé est de 192 bits avec donc $N_k = 6$.

Extension de la clé secrète K . On écrit la clé K sous forme d'une matrice de N_k colonne de 4 bytes chacune (donc 4 lignes l'élément d'une ligne étant un mot de 8 bits) dénotés k_0, \dots, k_5

k_0	k_1	k_2	k_3	k_4	k_5
-------	-------	-------	-------	-------	-------

Ensuite cette matrice est étendue en une matrice de taille $N_b(N_r + 1)$ où N_r est le nombre d'étages par l'algorithme suivant

- Si i n'est pas un multiple de N_k (la longueur de la clé) alors la colonne d'indice i , k_i , est la XORisation de la colonne d'indice $i - N_k$, k_{i-N_k} , et de la colonne d'indice $i - 1$, k_{i-1} . C'est donc l'addition bit à bit sans retenue de la colonne k_{i-N_k} et de la colonne k_{i-1} , $k_i = k_{i-N_k} \oplus k_{i-1}$.
- Si i est un multiple de N_k on applique à chacun des bytes de la colonne k_{i-1} la permutation π_S décrite au paragraphe suivant suivie d'une rotation dans les bytes de la nouvelle colonne k_{i-1} et de l'addition d'une constante d'étage définie par **Fieldto Binary**(x^{j-1}) pour l'étage j . On XOR le résultat obtenu avec la colonne k_{i-N_k} .

On obtient ainsi

k_0	k_1	k_2	k_3	k_4	k_5	...	$k_{i \cdot N_b}$...	$k_{(i+1)N_b-1}$...
clé d'étage 0						...	clé d'étage i			...

L'opération SubBytes

L'opération **SubBytes** est la seule opération non linéaire. On suppose que le message w^i est un nombre de 128 bits=16 octets, $(s_{i,j})_{0 \leq i,j \leq 3}$, écrit en base 2. On le réécrit sous la forme d'une matrice S_i , où les $s_{i,j}$ sont des octets.

$$S_i = \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix}$$

L'opération **SubBytes** consiste en une permutations π_S sur $\{0, 1\}^8$, agissant indépendamment sur chacun des octets $s_{i,j}$ de l'étage i .

Pour décrire π_S on travaille dans un surcorps du corps à 2 éléments, \mathbb{F}_2 , le corps fini à 256 éléments, \mathbb{F}_{2^8} , que l'on identifie avec les polynômes de degré ≤ 8 , munis de l'addition et de la multiplication modulo le polynôme irréductible de $\mathbb{F}[x]$, $x^8 + x^4 + x^3 + x + 1$, cf. la paragraphe 11.6.1, page 104, de rappel sur les anneaux de polynômes,

$$\mathbb{F}_{2^8} \simeq \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)\mathbb{F}_2[x]$$

On associe à l'octet $a_7a_6 \dots a_1a_0$ l'élément

$$\mathbf{BinarytoField}(a_7a_6 \dots a_1a_0) = \sum_{i=0}^7 a_i x^i$$

du corps \mathbb{F}_{2^8} dont l'application inverse est **FieldtoBinary**. On a dans le corps \mathbb{F}_{2^8} l'opération **FieldInv** qui à un élément $z \neq 0$ du corps associe z^{-1} et à $z = 0$ associe 0.

Exemples 9. Calcul dans \mathbb{F}_{2^8} . Calculons **FieldInv**(00000011) on a:

- **FieldtoBinary**(00000011)= $x + 1$
- **FieldInv**($x + 1$)= $x^7 + x^6 + x^5 + x^4 + x^2 + x$

car on a l'identité de Bézout

$$(x+1)(x^7+x^6+x^5+x^4+x^2+x) + (x^8+x^4+x^3+x+1) = 1$$

obtenue par l'algorithme d'Euclide ou directement,

donc dans $\mathbb{F}_{2^8} \simeq \mathbb{F}[x]/(x^8+x^4+x^3+x+1)$

$$(x+1)^{-1} = (x^7+x^6+x^5+x^4+x^2+x)$$

et donc

- **BinarytoField**($x^7+x^6+x^5+x^4+x^2+x$)=(11110110)

Revenons à la description de l'opération **SubBytes** c'est à dire à la description de la permutation π_S pour le mot de 8 bits $(a_7a_6 \dots a_1a_0)$ consiste en

- application de **BinarytoField** à $(a_7a_6 \dots a_1a_0) = (b_7b_6 \dots b_1b_0)$ qui donne $a \in \mathbb{F}_{2^8}$
- application **FieldInv** à a qui donne a^{-1} si $a \neq 0$ ou 0 si $a = 0$
- ajout de l'élément $c = (c_7c_6 \dots c_1c_0) = \mathbf{BinarytoField}(01100011)$ à a^{-1} pour obtenir $b = (b_7b_6 \dots b_1b_0)$
- application de **Fieldto Binary** à b

c est une constante d'AES.

L'opération ShiftRows

C'est la permutation cyclique dans les lignes de la matrice S_i

$$\begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline e & f & g & h \\ \hline i & j & k & l \\ \hline m & n & o & p \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline f & g & h & e \\ \hline k & l & i & j \\ \hline p & m & n & o \\ \hline \end{array}$$

L'opération MixColumns

C'est une transformation linéaire dans les colonnes; la colonne j est transformée de la manière suivante (les calculs sont faits dans \mathbb{F}_{2^8})

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

L'opération **AddRoundKey**

AddRoundKey $[i]$ est l'addition de la clef obtenue à l'étage i par l'algorithme de diversification des clés, **ExpandKey** $[i]$, au texte obtenu à l'issue de l'étape **MixColumns**.

On écrit la clef **ExpandKey** $[i]$, supposée ici de 128 bits, sous la forme d'une matrice de 4×4 bytes et on l'ajoute au résultat de l'étape précédente par un XOR

$$\begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|} \hline k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ \hline k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ \hline k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ \hline k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

Si la clef avait une taille $32N_k$ bites on l'écrirait sous la forme d'une matrice de N_k colonnes de 4 bytes et on procéderait de même.

7.3 Infrastructure des systèmes à clef secrète.

Les *infrastructures pour les systèmes à clef secrète* consistent en toutes les dispositions techniques et l'organisation nécessaires pour gérer un système cryptographique à clef secrète.

Autant il est facile pour un ordinateur de retenir une clé secrète de 128 bits, autant c'est difficile pour une personne. On utilise alors le système des mots de passe. Mais pour ne pas affaiblir la sécurité du cryptosystème le mot de passe doit obéir à certaines règles

- Avoir une entropie, cf. le paragraphe 11.1.3 page 80, au moins égal à 128 bits. Pour avoir une entropie de 128 bits il suffit de choisir 22 caractères au hasard dans un alphabet de 64 lettres (minuscules, majuscules, chiffres, point et espace).
- Il doit résister aux attaques-dictionnaires.
- Un mot de passe ne doit pas être réutilisé

On peut prendre les mesures suivantes pour améliorer la sécurité d'un système de mots de passe

- On utilise le mot de passe pour calculer une clé de session qui change à chaque fois.

- On rajoute des ingrédients pour éviter les attaques dictionnaires (sel).
- On itère la fonction de hachage, par exemple 1000 fois. Pour l'utilisateur le temps de calcul est à peine affecté. Pour l'attaquant qui procède par attaque-dictionnaire, c'est prohibitif.

On utilise aussi un *système de gestion des clefs* (*Symmetric Keys Management*) qui

1. émet les clefs
2. les authentifie
3. les archive
4. les garde en dépôt

Un des systèmes les plus utilisés est *Kerberos*, cf. [25].

Chapitre 8

Codes à clefs publiques.

Diffie et Hellman ont dégagé vers 1976 la notion de *code à clef publique ou code asymétrique* fondée sur la notion de fonction à sens unique.

Les codes asymétriques les plus utilisés sont

- **RSA** basé sur la difficulté calculatoire de la factorisation des grands entiers.
- **El Gamal** basé sur la difficulté calculatoire de calculer le logarithme discret dans un corps fini.
- **McEliece** basé sur la théorie algébrique des codes correcteurs d'erreurs. Il repose sur la difficulté calculatoire du décodage d'un code linéaire (problème NP complet). Considéré comme sûr il nécessite des clefs extrêmement longues 10^{19} bits.
- **Chor-Rivest** basé une instance du problème du **sac à dos** considéré comme sûr.
- Les cryptosystèmes basés sur les courbes elliptiques qui sont des modifications des autres cryptosystèmes, comme El Gamal. Ils utilisent le groupe des points d'une courbe elliptique sur un corps fini au lieu du groupe multiplicatif du corps fini, par exemple Menezes-Vanstone.

8.1 Principe des codes à clef publique.

8.1.1 Fonctions à sens unique.

On cherche une fonction f entre des ensembles \mathcal{P} et \mathcal{E} telle que

- le calcul de $f(x)$ pour $x \in \mathcal{P}$ se fait en temps polynomial en fonction de la taille des données.
- le calcul de $f^{-1}(y)$ pour $y \in \mathcal{E}$ ne se fait pas en temps polynomial en fonction de la taille des données. Le plus souvent la fonction f possède une *porte dérobée* (trapdoor) qui permet de calculer facilement $f^{-1}(y)$ lorsqu'on a une information supplémentaire comme la clef secrète.

Autrement dit on cherche des fonction $f : \mathcal{P} \rightarrow \mathcal{E}$ appartenant à la classe des problèmes P (calculables en temps polynomial en fonction de la taille des données) et telle que la fonction réciproque (ou l'image réciproque si ne suppose pas f bijective) f^{-1} appartienne à la classe des problèmes NP , (non calculables en temps polynomial en fonction de la taille des données, mais vérifiables en temps polynomial).

La mise au point de telles fonctions a révolutionné la cryptographie et élargi son champ d'application.

8.2 Le système RSA.

Le système RSA est nommé d'après le nom de ses inventeurs: Rivest, Shamir, Adleman.

Il est basé sur la fonction à sens unique *produit de deux entiers*

$$f : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$$

$$(n, m) \longmapsto f(n, m) = n \times m$$

sa fonction réciproque étant la décomposition d'un nombre entier en un produit de facteur premier, cf la section arithmétique dans les compléments de mathématiques, section 11.3 page 86.

Le calcul du produit de deux nombres entiers est une opération "facile", "rapide", "calculatoirement facile" (c'est à dire faisable en temps polynomial en fonction de la taille des données) alors que la décomposition en facteurs premiers n'est pas "facile", est "lente", "calculatoirement difficile" (c'est à dire qu'elle n'est pas faisable en temps polynomial en fonction de la taille des données)

8.2.1 Description du cryptosystème RSA.

On a un ensemble $(A_i)_{i \in I}$ de correspondants (personnes physiques ou ordinateurs). Le principe du cryptosystème RSA est le suivant

- Chacun des correspondants A_i choisit deux nombres premiers p_i et q_i distincts. On note $n_i = p_i q_i$, le *module du cryptosystème* de A_i et $\varphi(n_i) = (p_i - 1)(q_i - 1)$, l'indicatrice d'Euler de n_i .
- A_i choisit ensuite un nombre e_i l'*exposant de la clé publique* premier avec $\varphi(n_i)$ (autrement dit le Plus Grand Commun Diviseur (PGCD ou GCD en anglais) de e_i et $\varphi(n_i)$ est 1. On note ce PGCD $e_i \wedge \varphi(n_i) = 1$) en pratique on impose de plus $1 \leq e_i \leq \varphi(n_i) - 1$.
- Puis A_i calcule l'inverse d_i de e_i modulo $\varphi(n_i)$, d_i est appelé *exposant de la clé secrète* c'est à dire en pratique, on cherche $d_i \in \mathbb{N}$ tel qu'il existe $k_i \in \mathbb{N}$ avec

$$d_i \times e_i = 1 + k_i \varphi(n_i) \text{ et } 1 \leq d_i \leq \varphi(n_i) - 1$$

- Chacun des correspondants A_j publie dans un annuaire public les nombres n_j et e_j qui forment sa *clé publique de chiffrement* et garde secret p_j , q_j et d_j qui forment sa *clé secrète de déchiffrement*.

En pratique on choisit les nombres premiers p_i et q_i de taille comparable de telle sorte que leur produit $n_i = p_i q_i$ soit un nombre d'au moins 300 chiffres en base 10 et plutôt 500 pour une protection longue.

8.2.2 Protocole d'envoi d'un message en RSA.

Alice veut envoyer un message \mathcal{M} à Bob. La clé publique d'Alice (resp Bob) est (n_a, e_a) , (resp. (n_b, e_b)), sa clé secrète est (p_a, q_a, d_a) (resp. (p_b, q_b, d_b)). Elle suit le protocole suivant

1. Alice commence par transformer le message en une suite de chiffres, par exemple en remplaçant les lettres et les différents symboles utilisés par des chiffres (de 0 à 255 dans le cas du code ASCII).
2. Alice regarde dans l'annuaire public la clé de chiffrement n_b, e_b de Bob. Elle découpe le message \mathcal{M} en blocs M de taille approximativement n_b .
3. Elle calcule

$$f_b(M) = M' \equiv M^{e_b} \pmod{n_b}$$

et envoie $M' = f_b(M)$ à Bob.

4. Pour récupérer le texte en clair Bob calcule

$$f_b^{-1}(M') = (M')^{d_b} = f_b(M)^{d_b} \equiv M^{e_b d_b} = M^{1+k_b \varphi(n_b)} \pmod{n_b}$$

D'après le théorème d'Euler (cf. théorème 11.3.19 page 97) on a

$$M^{\varphi(n_b)} \equiv 1 \pmod{n_b}$$

et par conséquent

$$f_b^{-1}(M') \equiv M \pmod{n_b}$$

Si la taille de M est adaptée à celle de n_b , il n'y a pas d'ambiguïté Bob récupère donc bien le message d'Alice.

8.2.3 Protocole de signature RSA.

Le cryptosystème RSA permet aussi facilement de réaliser l'authentification de l'expéditeur (en admettant que sa clé privée n'est utilisée que par lui).

1. Alice commence par découper le message \mathcal{M} en blocs M de taille bien choisie par rapport à n_a et n_b . Elle crypte chaque bloc du message avec sa clé secrète, d_a , elle obtient à partir du texte en clair M un premier texte crypté, M''

$$M'' = f_a^{-1}(M) \equiv M^{d_a} \pmod{p_a q_a}$$

2. Avec la clé publique de Bob, elle crypte M'' et obtient un deuxième texte crypté, M'

$$M' = f_b(M'') = f_b(f_a^{-1}(M))$$

qui constitue sa signature du message M .

3. Elle crypte alors le message M avec la clé publique de Bob en suivant le protocole précédent et obtient le texte crypté du message $f_b(M)$

$$f_b(M) = M^{e_b} \pmod{n_b}$$

4. Elle envoie à Bob le couple

$$(f_b(M), M')$$

qui constitue le message chiffré et signé d'Alice à Bob.

Pour décoder et vérifier la signature qui authentifie l'expéditeur du message, d'Alice, Bob effectue les opérations suivantes

1. Il calcule d'une part

$$\begin{aligned} f_b^{-1}(M') &\equiv M'' \pmod{n_b} \\ &\equiv f_a\left(f_b^{-1}\left(f_b(f_a^{-1}(M) \pmod{n_b})\right)\right) \pmod{n_a} \\ &\equiv M \pmod{n_a} \end{aligned}$$

(Si le choix de la taille des blocs M est bien choisie par rapport à n_a et n_b , il n'y a pas d'ambiguïté au décodage)

2. Il calcule d'autre part, en décodant comme au paragraphe 8.2.2,

$$f_b^{-1}f_b(M) \equiv M \pmod{n_b}$$

3. Il compare les deux résultats obtenus. S'il constate que $M = M$, c'est à dire que les deux résultats sont les mêmes, il est sûr alors que le message vient d'Alice car seule Alice connaît

$$f_a^{-1} \iff d_a$$

c'est à dire sa clé secrète. De plus il est sûr que le message n'a pas été altéré.

8.2.4 Exemple académique de codes RSA.

Alice publie sa clé publique $(n_A, e_A) = (65, 5)$, sa clé secrète est d_A .

Bob publie sa clé publique $(n_B, e_B) = (77, 7)$, sa clé secrète est d_B .

Pour trouver les clés secrètes on décompose en un produit de facteurs premiers n_A et n_B

$$n_A = 13 \cdot 5 \text{ et } n_B = 7 \cdot 11$$

On calcule ensuite les indicatrices d'Euler correspondantes

$$\varphi(n_A) = 12 \times 4 = 48 \text{ et } \varphi(n_B) = 6 \times 10 = 60$$

En utilisant l'algorithme de PGCD décrit au paragraphe 11.3.3 page 89 (ou directement) on calcule d_A

$$5 \times 29 = 145 = 1 + 3 \times 48 \implies d_A = 29$$

respectivement d_B

$$7 \times 43 = 301 = 5 \times 60 + 1 \implies d_B = 43$$

Alice veut transmettre le message 3 à Bob. Elle calcule donc

$$3^7 \equiv 31 \pmod{77} = 7 \times 11$$

et elle envoie 31 à Bob. Bob décode en calculant

$$31^{43} \equiv 31^{1+2+8+32} \equiv 31 \cdot 37 \cdot 58 \cdot 37 \cdot 31 \equiv 3 \pmod{77}$$

(remarquer la manière de calculer une puissance).

Si Alice veut signer le message elle n'envoie pas seulement 31, mais elle calcule d'abord

$$f_a^{-1}(3) = (3^{29} \pmod{65}) = 48$$

puis elle calcule

$$f_b(f_a^{-1}(3)) = f_b(3^{29} \pmod{65}) = (48^7 \pmod{77}) = 27$$

puis elle envoie à Bob le couple

$$(31, 27)$$

constitué de la signature et du message.

Pour décoder Bob calcule

$$f_a(f_b^{-1}(27)) = f_a(27^{43} \pmod{77}) = (48^5 \pmod{65}) = 3 \pmod{65}$$

et

$$31^{43} \equiv 3 \pmod{77}$$

et il constate que les deux résultats coïncident. L'expéditeur du message est bien Alice.

Remarque 8.2.1. Pour signer le message Alice aurait pu aussi bien mener les calculs dans l'ordre inverse, c'est à dire calculer d'abord avec la clé publique de Bob

$$\widetilde{M}'' = f_b^{e_b}(M) \pmod{n_b}$$

puis avec sa clé secrète d_a elle crypte \widetilde{M}'' et obtient un deuxième texte crypté, \widetilde{M}'

$$\widetilde{M}' = f_a^{-1}(\widetilde{M}'') = f_a^{-1}(f_b^{e_b}(M)) \equiv (\widetilde{M}'')^{d_a} \pmod{n_a}$$

qui constitue sa signature du message M .

Elle crypte alors le message M avec la clé publique de Bob en suivant le protocole précédent et obtient le texte crypté du message $f_b(M)$

$$f_b(M) = M^{e_b} \pmod{n_b}$$

Elle envoie à Bob le couple

$$(f_b(M), \widetilde{M}')$$

Bob décode en calculant d'une part

$$\begin{aligned} f_a(\widetilde{M}') &\equiv \widetilde{M}'' \pmod{n_a} \\ f_b^{-1}(f_a(\widetilde{M}')) &= f_b^{-1}(f_a(f_a^{-1}(f_b(M)))) \equiv M \pmod{n_b} \end{aligned}$$

Les deux protocoles ont leurs avantages et inconvénients. On préfère en général le premier protocole car il suit de plus près la manière traditionnelle de signer une lettre: on signe d'abord avant de la mettre dans l'enveloppe. Le second protocole consiste à mettre la lettre dans l'enveloppe et à signer l'enveloppe.

8.2.5 Exemple de code RSA.

On choisit un alphabet à 40 lettres

$$\begin{array}{cccccccc} A= 0, & B= 1, & C= 2, & D= 3, & E= 4, & F= 5, & \dots, & \\ \dots, & X= 23, & Y= 24, & Z= 25, & \square=26, & .= 27, & ?= 28, & \\ euros= 29, & 0= 30, & 1= 31, & \dots, & 8= 38, & 9= 39, & & \end{array}$$

On choisit comme clé publique le couple

$$n = 2047, \quad e = 179$$

on a

$$40^2 \leq 2047 \leq 40^3$$

on choisit donc de découper les messages en clair en blocs de deux lettres et les messages chiffrés en blocs de trois lettres.

On écrit chaque bloc de deux symboles clairs X_1X_2 sous la forme $40x_1 + x_2$ où x_i est le nombre entre 1 et 39 correspondant à X_i .

On écrit chaque bloc de trois symboles codés $Y_1Y_2Y_3$ sous la forme $40^2y_1 + 40y_2 + y_3$ où y_i est le nombre entre 0 et 39 correspondant à Y_i .

Le message clair est

ENVOYEZ euros2500.

Pour le coder on considère les blocs

$$\begin{aligned} \text{EN} &= 4 \times 40 + 13 = 173, & \text{VO} &= 21 \times 40 + 14 = 854, \\ \text{YE} &= 24 \times 40 + 4 = 964, & \text{Z}\square &= 1026, \\ \text{euros2} &= 1192, & 50 &= 1430, \\ 0. &= 1110 \end{aligned}$$

Codage du message

$$\begin{aligned} \text{EN} &\mapsto (173)^{179} \pmod{2047} = 854 = 0 \times 40^2 + 21 \times 40 + 14 = \text{AVO} \\ \text{VO} &\mapsto (854)^{179} \pmod{2047} = 1315 = 0 \times 40^2 + 32 \times 40 + 35 = \text{A25} \\ \text{YE} &\mapsto (964)^{179} \pmod{2047} = 452 = 0 \times 40^2 + 11 \times 40 + 12 = \text{ALM} \\ \text{Z}\square &\mapsto (1026)^{179} \pmod{2047} = 1295 = 0 \times 40^2 + 32 \times 40 + 15 = \text{A2P} \\ \text{euros2} &\mapsto (1192)^{179} \pmod{2047} = 511 = 0 \times 40^2 + 12 \times 40 + 31 = \text{AM1} \\ 50 &\mapsto (1430)^{179} \pmod{2047} = 1996 = 1 \times 40^2 + 9 \times 40 + 36 = \text{BJ6} \\ 0. &\mapsto (1110)^{179} \pmod{2047} = 1642 = 1 \times 40^2 + 1 \times 40 + 2 = \text{BBC} \end{aligned}$$

Le message codé est donc

AVOA25ALMA2PAM1BJ6BBC

Pour décoder on factorise

$$2047 = 23 \times 89 \implies \varphi(2047) = 22 \times 88 = 1936$$

et on déduit par l'algorithme d'Euclide que

$$d = 411.$$

Remarque 8.2.2. En fait les paramètres de ce code sont mal choisis car 22 divise 88 et donc on peut prendre pour d n'importe quel inverse de 179 modulo 88 comme par exemple 59. On constate que $411 = 59 + 4 \times 88$

8.2.6 Sécurité du système RSA.

La sécurité de RSA repose sur les faits suivants:

- on ne sait pas décoder sans connaître l'exposant de la clé secrète d
- trouver d équivaut à factoriser $n = pq$
- on ne connaît pas d'algorithme polynomial en temps en fonction de la taille des données (c'est à dire de la longueur des nombres n et e) pour calculer d c'est à dire pour factoriser n , les meilleurs sont en

$$O\left(e^{C\sqrt{\log_2 n \log_2 \log_2 n}}\right) \text{ opérations.}$$

on dit que l'algorithme est *sous-exponentiel*

Ce sont des faits d'expérience pas des théorèmes.

Depuis sa création le système RSA a résisté à toutes les attaques de manière satisfaisante si l'on choisit bien p , q et e .

Pour construire et utiliser un code RSA on a besoin de savoir faire rapidement (c'est à dire en temps polynomial au plus) les opérations suivantes:

- construire des grands nombres premiers
- calculer des PGCD
- faire de l'arithmétique modulaire (addition, multiplication, exponentiation)

Pour évaluer et tester la sécurité du code RSA, il faut en particulier

- évaluer la rapidité des algorithmes de factorisations de grands nombres entiers
- Démontrer que la clef secrète de déchiffrement d ne peut pas être obtenue sans factoriser $n = pq$ et plus généralement qu'elle ne peut pas être calculer en temps polynomiale en fonction de n .
- montrer que l'on ne peut pas décoder un message sans la clef secrète

8.3 Le cryptosystème El Gamal.

Le cryptosystème El Gamal est basée sur la fonction à sens unique *logarithme discret*. On considère un groupe cyclique fini, G , de cardinal n engendré par un générateur, g . Donc

$$G = \{g^i \mid 0 \leq i \leq n - 1\}$$

On suppose que le calcul de g^i est “facile”, “rapide” “calculatoirement facile” (c’est à dire faisable en temps polynomial en fonction de la taille des données) mais que le calcul de i connaissant g^i n’est “pas facile”, est “lent”, “calculatoirement difficile” (c’est à dire: n’est pas faisable en temps polynomial en fonction de la taille des données). Si $a = g^i$, i est appelé le *logarithme discret* de a .

On utilise les deux réalisations suivantes de G . On considère un nombre premier p et on choisit

$$G = (\mathbb{Z}/p\mathbb{Z})^* \simeq \mathbb{F}_p^*$$

c’est à dire que le G est le groupe des éléments inversible pour la multiplication de $\mathbb{Z}/p\mathbb{Z}$ les entiers modulo p , pour la définition voir dans les compléments mathématiques, le paragraphe sur les congruences 11.3.4 page 92.

D’après un théorème de Gauss, théorème 11.3.13 page 94, le groupe multiplicatif de $\mathbb{Z}/p\mathbb{Z}$ est cyclique et le calcul de $g^i \pmod p$ est rapide alors qu’on ne connaît pas d’algorithme en temps polynomial pour calculer i connaissant $a = g^i$ pour de bons choix de p . Plus généralement on peut considérer le groupe multiplicatif des éléments inversibles d’un corps fini à $q = p^s$ éléments, \mathbb{F}_q .

Une autre réalisation est comme groupe G le groupe des points d’une courbe elliptique sur un corps fini, $E(\mathbb{F}_q)$ qui donne lieu aux cryptosystèmes elliptiques, cf le paragraphe 11.7.1 page 117. Ils se développent actuellement car ils possèdent des avantages en terme de taille de clé et de sécurité prouvée.

8.3.1 Description du cryptosystème El Gamal.

On considère le cryptosystème suivant: Soit p un nombre premier tel que le problème du logarithme discret dans $\mathbb{Z}/p\mathbb{Z}$ soit difficile.

Soit g une racine primitive modulo p , cf. le théorème 11.3.13 page 94, c’est à dire que g est un générateur du groupe cyclique $(\mathbb{Z}/p\mathbb{Z})^*$.

Soit $\mathcal{P} = (\mathbb{Z}/p\mathbb{Z})^*$, les messages en clair, et soit $\mathcal{C} = (\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/p\mathbb{Z})^*$ les messages cryptés et enfin soit l'espace des clefs

$$\mathcal{K}_b = \{(p, g, \alpha, \beta); \beta \equiv g^\alpha \pmod{p}\} \quad \text{où} \quad \begin{cases} (p, g, \beta) & \text{est la clef publique} \\ \alpha & \text{est la clef secrète} \end{cases}$$

Alice veut transmettre un message, \mathcal{M} , à Bob

1. Bob choisit un grand nombre premier p_b , un générateur g_b du groupe cyclique multiplicatif $(\mathbb{Z}/p_b\mathbb{Z})$ et un entier α_b inférieurs à p
2. Bob calcule $\beta_b = g_b^{\alpha_b}$; alors le triplet (β_b, g_b, p_b) constitue sa clef publique, α_b est sa clef secrète.
3. Alice découpe le message \mathcal{M} en blocs M de taille inférieure à p_b , elle choisit un entier k_a (inférieur à $p_b - 1$) pour chacun des blocs M et calcule

$$y_1 \equiv g_b^{k_a} \pmod{p_b} \quad \text{et} \quad y_2 = \beta_b^{k_a} M$$

La paire $e_{K_b}(M, k_a) = (y_1, y_2)$ est le message codé qu'Alice envoie à Bob.

Pour décoder

1. Bob calcule

$$M \equiv y_2 (y_1^{\alpha_b})^{-1} \pmod{p_b}$$

Comme $y_1^{\alpha_b} = g_b^{k_a \alpha_b} \pmod{p}$ on a:

$$\begin{aligned} d_K(y_1, y_2) &= y_2 (y_1^{\alpha_b})^{-1} \equiv \beta_b^{k_a} M (g_b^{k_a \alpha_b})^{-1} \equiv \\ &\equiv g_b^{\alpha_b k_a} M g_b^{-\alpha_b k_a} \equiv M \pmod{p} \end{aligned}$$

et comme M est inférieur à p_b il n'y a pas d'ambiguïté dans le décodage.

Pour une bonne sécurité, Alice doit changer souvent k_a .

8.3.2 Signature El Gamal.

On reprend les notations précédentes. Soit M un bloc d'un message \mathcal{M} qu'Alice veut transmettre en le signant à Bob.

1. Alice choisit elle-aussi un grand nombre premier p_a tel que le problème du logarithme discret dans $(\mathbb{Z}/p_a\mathbb{Z})^*$ soit difficile.

2. Elle choisit aussi un générateur g_a de $(\mathbb{Z}/p_a\mathbb{Z})^*$ et un entier $0 \leq \alpha_a \leq p_a - 2$ et elle calcule $\beta_a = g_a^{\alpha_a}$.

Elle dispose ainsi elle aussi d'une clef publique (p_a, g_a, β_a) et d'une clef privée (secrète) α_a . Pour signer le message M , Alice réalise les opérations suivantes

1. elle choisit $k'_a \in (\mathbb{Z}/p\mathbb{Z})^*$ secret et premier à $p_a - 1$.
2. elle calcule

$$\begin{aligned}\gamma_a &\equiv g_a^{k'_a} \pmod{p_a} \\ \delta_a &\equiv (M - \alpha_a \gamma_a) k'^{-1}_a \pmod{p_a - 1}\end{aligned}$$

3. la signature du bloc M est alors $\text{sign}_{K_a}(M, k'_a)$ avec

$$\text{sign}_{K_a}(M, k'_a) = (\gamma_a, \delta_a)$$

4. Puis elle envoie le couple

$$(e_{K_b}(M, k_a), \text{sign}_{K_a}(M, k'_a))$$

c'est à dire le quadruplet $((y_1, y_2), (\gamma_a, \delta_a))$.

On constate que

$$\beta_a^{\gamma_a} \gamma_a^{\delta_a} \equiv g_a^M$$

Remarque 8.3.1. La condition k'_a premier à $p_a - 1$ est nécessaire car Alice utilise pour calculer δ_a la quantité $k'^{-1}_a \pmod{p_a - 1}$.

Cette procédure réalise bien une signature du bloc M . Pour vérifier la signature et décoder Bob suit le protocole suivant:

1. Bob reçoit le couple constitué par le message crypté par Alice de la signature de ce message c'est à dire le quadruplet $((y_1, y_2), (\gamma_a, \delta_a))$.
2. Il décode le message à l'aide de sa clé secrète et récupère M .
3. Il calcule

$$\beta_a^{\gamma_a} \gamma_a^{\delta_a} \pmod{p_a}$$

et il vérifie que

$$\beta_a^{\gamma_a} \gamma_a^{\delta_a} \equiv g_a^M$$

S'il en est ainsi il est sûr que l'expéditeur du message est bien Alice.

On constate que cette procédure réalise bien une signature du message M car pour fabriquer le couple (γ_a, δ_a) Alice a utilisé sa clé secrète et si l'on remplace dans la formule donnant δ_a la clé secrète α_a par un autre nombre compris entre 0 et $p_a - 1$ on n'aura plus l'égalité

$$\beta_a^{\gamma_a} \gamma_a^{\delta_a} \equiv g_a^M \pmod{p_a}$$

8.3.3 Sécurité du système EL Gamal.

La sécurité du cryptosystème EL Gamal repose sur les faits suivants:

- on ne sait pas décoder sans connaître la clé secrète α
- trouver α équivaut à trouver le logarithme discret de $\beta = g^\alpha$ dans $(\mathbb{Z}/p\mathbb{Z})^*$
- on ne connaît pas d'algorithme polynomial en temps en fonction de la taille des données (c'est à dire de la longueur du nombre p) pour calculer α . Les meilleurs sont en

$$O\left(e^{C\sqrt{\log_2(p)\log_2\log_2(p)}}\right) \text{ opérations.}$$

Ce sont des faits d'expérience pas des théorèmes.

8.3.4 Exemple académique de code El Gamal.

Alice et Bob veulent correspondre en employant le système El-Gamal. Alice choisit comme paramètres

$$p_a = 263, g_a = 5, \alpha_a = 47$$

Bob choisit comme paramètres

$$p_b = 257, g_b = 5, \alpha_b = 67$$

On vérifie facilement que p_a et p_b sont des nombres premiers, que g_a et g_b sont respectivement des générateurs des groupes cycliques $\mathbb{Z}/p_a\mathbb{Z}$ et $\mathbb{Z}/p_b\mathbb{Z}$. Il suffit pour cela de vérifier (par exemple à l'aide d'une table ou d'un système de calcul faisant de l'arithmétique modulaire) que

$$g_a^{\alpha_a} \not\equiv 1 \pmod{p_a}, \quad g_b^{\alpha_b} \not\equiv 1 \pmod{p_b}$$

pour tout diviseur non trivial d_a de $p_a - 1$ et tout diviseur non trivial d_b de $p_b - 1$. En effet d'après le corollaire 11.3.20 le plus petit entier non nul e tel que $g_a^e \equiv 1 \pmod{p_a}$ est un diviseur de $p_a - 1$.

Alice calcule $\beta_a = p_a^{\alpha_a} \pmod{p_a} = 40$ et sa clef publique est

$$K_a = (p_a = 263, g_a = 5, \beta_a = 40)$$

sa clé secrète est $\alpha_a = 47$.

Bob calcule $\beta_b = p_b^{\alpha_b} \pmod{p_b} = 201$ et sa clef publique est

$$K_b = (p_b = 257, g_b = 5, \beta_b = 201)$$

sa clé secrète est $\alpha_b = 67$.

On suppose qu'Alice et Bob s'envoient des nombres de deux chiffres en base 10. Alice envoie à Bob le message M qu'elle code en $(y_1, y_2) = (76, 251)$ à l'aide la clé publique de Bob et d'un k_a qu'elle garde secret.

Bob décode le message en effectuant l'opération

$$\begin{aligned} y_2(y_1^{\alpha_b})^{-1} \pmod{257} &= 251 \times 76^{-67} \pmod{257} \\ &= 251 \times 76^{256-67} \pmod{257} \\ &= 251 \times 76^{189} \pmod{257} = 251 \times 155 \pmod{257} \\ &= (-6) \times 155 \pmod{257} = 98 \end{aligned}$$

.Le message en clair d'Alice est $M = 98$.

On peut retrouver le paramètre k_a à l'aide d'une table des puissances de $g_b \pmod{257}$. On sait que

$$y_1 = g_b^{k_a} \pmod{257} = 76, \quad y_2 = \beta_b^{k_a} M \pmod{257} = 251$$

On trouve à l'aide d'une table ou d'un logiciel de calcul d'arithmétique modulaire $k_a = 139$.

Si Alice veut envoyer le message $M = 87$ à Bob avec les mêmes paramètres, y compris k_a , elle effectue le calcul

$$\begin{aligned} z_1 &= y_1 = g_b^{k_a} \pmod{257} = 76 \\ z_2 &= \beta_b^{k_a} \times 87 \pmod{257} = 201^{139} \times 87 \pmod{257} = 173 \end{aligned}$$

Pour un bon choix de p le système El-Gamal résiste bien aux attaques. Comme le système RSA il est assez lent et nécessite des clefs longues 1024

à 2048 bits actuellement). Il se transpose sans difficulté à n'importe quel groupe cyclique où le problème du logarithme discret est difficile, en particulier pour le groupe des points d'une courbe elliptique sur un corps fini. Avec ce groupe malgré les difficultés algorithmique on a de bons codes avec des clefs courtes (128 à 256 bits actuellement) dont la sécurité peut être au moins partiellement prouvée.

8.4 Infrastructure des systèmes à clef publique.

Les *infrastructures des systèmes à clef publique* ou *PKI* (*Public Key Infrastructure*) consistent en toutes les dispositions techniques et organisationnelles nécessaires pour gérer un système cryptographique à clef publique.

On a un ensemble d'interlocuteurs en réseau qui se sont mis d'accord sur un cryptosystème à clef publique (par exemple RSA), sur une fonction de hachage (cf. le chapitre 9 page 63) et sur un protocole de signature. On suppose que chacun d'entre eux dispose d'une paire (K_e, K_d) (clé publique, clé secrète), (clé de chiffrement, clé de déchiffrement) et que chacun d'entre eux est capable de chiffrer, déchiffrer et signer.

Du fait que les clés publiques ne sont pas confidentielles, il n'est pas nécessaire de les crypter pour les transmettre. Mais il est très important et même vital pour la sécurité des transmissions de s'assurer de l'authenticité des clés ainsi transmises.

En effet si Alice désire transmettre sa clef publique à Bob, n'importe quel opposant, Martin par exemple, peut intercepter le message la contenant. Martin peut ensuite envoyer un message à Bob en se faisant passer pour Alice et contenant sa propre clé publique et donnant comme adresse de retour sa propre adresse. Ainsi il est capable de lire les messages cryptés que Bob envoie à Alice avec la soit-disant clé publique d'Alice que Bob croit posséder. Une fois qu'il les a décryptés et lu il peut les envoyer à Alice dont il possède la clé publique en les modifiant s'il l'estime nécessaire.

Il faut donc d'une certaine manière faire un lien entre chacun des participants au réseau et sa clé publique. Pour cela on utilise les *certificats*.

Un certificat consiste en une clef publique et une identité digitale (par exemple une suite de symboles contenant le nom du propriétaire de la clef, de la même manière qu'on met une étiquette sur une clé ordinaire), le tout étant cacheté à l'aide de la signature digitale d'une personne ou d'une organisation en laquelle on a confiance et appelée un *Tiers de Confiance* (*Trusted Third Party* ou *TTP*).

Pratiquement on peut par exemple concaténer, mettre bout à bout, la clef publique, le nom de son possesseur et signer le message obtenu à l'aide de la clef privée du Tiers de Confiance (il faut que personne ne puisse usurper l'identité du Tiers de Confiance).

Il existe plusieurs modèles de réseau avec Tiers de Confiance, avec deux modèles extrêmes, le modèle hiérarchique et le modèle distribué.

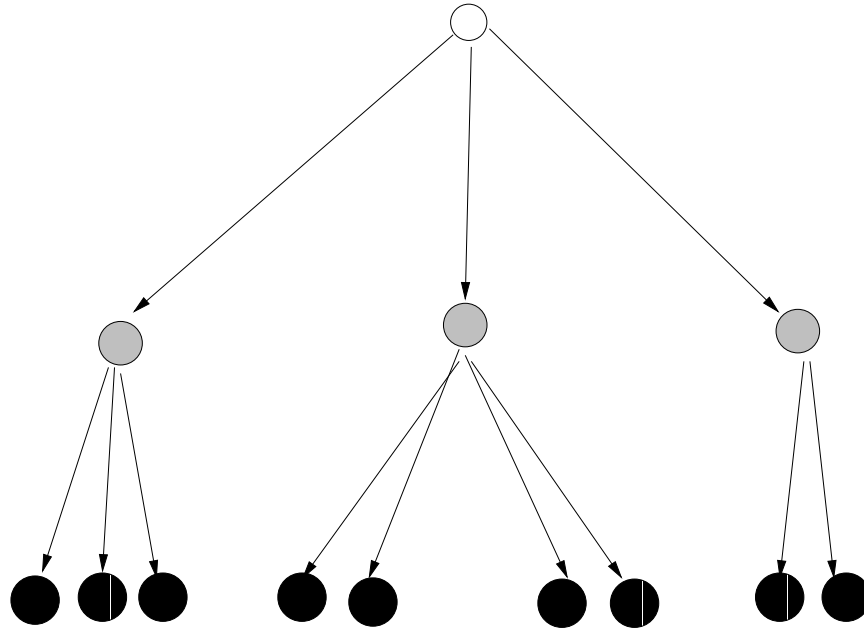
Un système très employé de système hiérarchique de Tiers de Confiance est le *modèle X.509*, [25].

Les fonctions principales d'un PKI sont

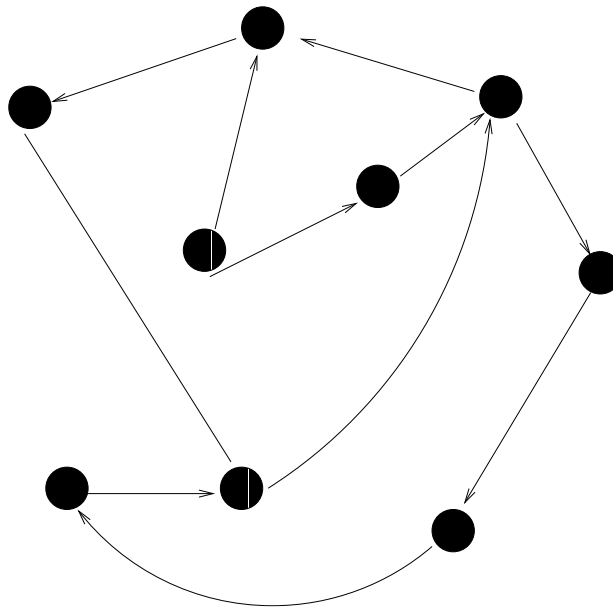
- Émission de paires de clefs (clé publique, clé privée)
- Attribution d'*identifiants uniques* aux participants au réseau
- Conservation des clefs
 1. Dépôt de clefs, utile quand il s'agit de garantir la continuité d'un service en cas de changement de titulaire
 2. Archivage des clefs, utile pour pouvoir relire de vieux messages
 3. Sauvegarde des clefs, utile en cas de perte de clef
 4. Récupération de clef perdues

ces fonctions ne sont pas très distinctes

- Émettre et publier des certificats
- Révoquer les certificats
- fixer leur durée de validité
- Archiver les certificats expirés (important pour les problèmes de non-répudiation)
- Donner une datation sécurisée



Système hiérarchique de Tiers de Confiance



Système non-hiérarchique de Tiers de Confiance

Chapitre 9

Fonctions de Hachage.

Les procédures de signature précédentes ont un coût prohibitif pour signer des longs messages car la signature est aussi longue que le message. On double donc la longueur du texte à crypter.

Pour réduire la longueur de la signature on peut utiliser une *fonction de hachage* cryptographique (*hash function* en anglais), h , ou fonction de condensation. C'est une fonction à sens unique qui sera publique dans les applications.

Les fonctions de hachage ont d'autres applications cryptographiques que la signature, cf le chapitre 10 page 68 sur les protocoles cryptographiques.

Une fonction de hachage est une fonction rapide à calculer mais dont l'image réciproque est dans la classe des problèmes calculatoirement difficiles (la classe NP). Elle transforme un message de longueur arbitraire en une *empreinte numérique* ou *code d'authentification* du message de taille fixée, 160 bits en général actuellement. Pour des raisons de sécurité on tend à augmenter la taille de l'empreinte.

Le schéma de calcul d'une signature avec une fonction de hachage est le suivant:

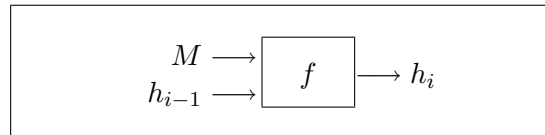
message	x	longueur arbitraire
	\downarrow	
empreinte numérique	$z = h(x)$	160 bits
	\downarrow	
signature	$y = \text{sign}_K(z)$	320 bits

En réalité une fonction de hachage prend un message x de taille inférieure à N fixé qu'elle transforme en une empreinte de taille 160 bits (ou 256 ou 384 ou 512).

Il existe des techniques classiques, cf. [23], pour étendre les fonctions de hachage de domaine de départ fini (*fonction de compression*) $\leq N$ en des fonctions de hachage dont le domaine de départ est constitué de messages de longueur arbitraire. Elles sont appelées alors *fonctions de hachage itérées*. Le principe pour l'itération est le suivant. On suppose que la fonction h peut recevoir deux entrées

- une empreinte h_{i-1}
- un bloc M de taille N

et donner en sortie une empreinte h_i .



On peut supposer que cette extension des fonctions de hachage est déjà faite.

Lorsqu'Alice souhaite envoyer un message signé, x , elle calcule d'abord l'empreinte numérique, $z = h(x)$, elle signe avec $y = \text{sign}_K(z)$ et transmet le couple (x, y) par le canal de communication.

Tout le monde peut vérifier la signature en calculant l'empreinte $z = h(x)$ et en utilisant le procédé de vérification de la signature $\text{ver}_K(z, y)$.

9.1 Construction des fonctions de hachage.

La fonction de hachage doit être construite avec soin pour qu'elle n'affaiblisse pas le protocole de signature. Un opposant comme Martin ne doit pas pouvoir forger la signature d'Alice.

Comme une fonction de hachage n'est évidemment pas injective, il existe des couples de messages x' et x tels que $h(x') = h(x)$. L'attaque la plus évidente consiste pour Martin à partir d'un message signé (x, y) authentique ($y = \text{sign}_K(x)$) précédemment calculé par Alice à calculer $z = h(x)$ et à chercher $x' \neq x$ tel que $h(x) = h(x')$. Si Martin y parvient (x', y) est un message valide.

Donnons tout d'abord quelques définitions de qualités que l'on peut exiger d'une fonction de hachage.

Définition 9.1.1. Une fonction de hachage h est à collisions faibles difficiles si étant donné un message x , il est calculatoirement difficile d'obtenir un message $x' \neq x$ tel que $h(x) = h(x')$.

Définition 9.1.2. Une fonction de hachage h est à collisions fortes difficiles s'il est calculatoirement difficile d'obtenir deux messages différents x et x' tels que $h(x) = h(x')$

Définition 9.1.3. Une fonction de hachage est une fonction de hachage à sens unique si étant donnée une empreinte numérique z , il est calculatoirement difficile de trouver un message x tels $h(x) = z$

Si une fonction de hachage est à collisions fortes difficiles elle est bien sûr à collisions faibles difficiles, mais aussi à sens unique.

9.1.1 Attaques des anniversaires.

Une fonction de hachage doit aussi résister aux *attaques des anniversaires*.

Une méthode simple pour obtenir des collisions, i.e. des messages x et x' tels que $h(x) = h(x')$ est l'attaque des anniversaires décrite ci-dessous.

On a une fonction de hachage $h : X \rightarrow Z$ où $|X| = m < +\infty$ et $|Z| = n$ et $m \geq 2n$. Dans ces conditions il y a au moins n collisions.

On va trouver la borne inférieure de la probabilité de trouver une collision en tirant k messages aléatoires distincts. Cette borne dépend de k et n mais pas de m . On fait l'hypothèse simplificatrice que $|h^{-1}(z)| \approx m/n$.

Il est alors facile de calculer la probabilité pour que k éléments $z_i \in Z$ soient tous distincts si l'on tire au hasard k éléments $x_i \in X$.

On ordonne les z_i en z_1, \dots, z_k . Le premier tirage z_1 est arbitraire; la probabilité pour que $z_2 \neq z_1$ est $1 - \frac{1}{n}$; la probabilité pour que z_3 soit distinct de z_1 et z_2 est $1 - \frac{2}{n}$, etc...

La probabilité de collision est donc

$$\left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) \dots \left(1 - \frac{k-1}{n}\right) = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right)$$

avec un peu d'analyse élémentaire on voit facilement en utilisant l'approximation

$$e^{-x} \approx 1 - x \text{ si } x \text{ petit}$$

que:

$$\prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right) \approx \prod_{i=1}^{k-1} e^{-\frac{i}{n}} = e^{-\frac{k(k-1)}{2n}}$$

La probabilité d'une collision est donc $1 - e^{-\frac{k(k-1)}{2n}}$. Si l'on veut que cette probabilité soit inférieure à ε alors il suffit de prendre

$$k \geq \sqrt{2n \log \frac{1}{1 - \varepsilon}}$$

Par exemple dans une assemblée de 23 personnes la probabilité d'avoir deux dates d'anniversaire égales est supérieure à $1/2$.

Une empreinte de 40 bits est vulnérable à une attaque des anniversaires avec probabilité supérieure à $1/2$ en utilisant seulement 2^{20} messages aléatoires. Pour une empreinte de 128 bits il faut 2^{64} messages aléatoires. Le choix d'une empreinte de 160 bits donne une bonne résistance aux attaques anniversaires. Mais l'augmentation de la puissance de calcul disponible pour un attaquant pousse à augmenter la taille des empreintes, c'est pourquoi AES prévoit des empreintes de 256 à 512 bits.

9.1.2 Exemple académique de fonction de hachage.

Fonction de hachage de Chaum-van Heijst-Pfitzmann. Elle n'est pas utilisée en pratique car trop lente.

Soit $p = 1 + 2q$ un grand nombre premier tel que q soit aussi premier. Soit α et β deux éléments primitifs de $(\mathbb{Z}/p\mathbb{Z})^*$. La valeur de $\log_{\alpha} \beta$ n'est pas publique et l'on suppose qu'elle est calculatoirement difficile à obtenir.

On démontre que la fonction de hachage

$$\begin{aligned} h : \{0, 1, \dots, q-1\} \times \{0, 1, \dots, q-1\} &\longrightarrow (\mathbb{Z}/p\mathbb{Z})^* \\ (x_1, x_2) &\longmapsto h(x_1, x_2) = \alpha^{x_1} \beta^{x_2} \end{aligned}$$

résiste aux collisions si le calcul de $\log_{\alpha} \beta$ est difficile.

9.1.3 Fonction de hachage standard.

Un standard actuel en matière de fonction de hachage est *SHA-1* (*Secure Hash Algorithm*) avec une empreinte de taille 160 bits, faisant suite aux standard MD4 (1990), MD5 (1992), SHA (1995), cf. [23].

Depuis 2001, une nouvelle version de SHA-1, SHA-2, ainsi que les versions SHA-256, SHA-384 et SHA-512 sont en cours de validation (256, 384, 512 est la taille en bits de l’empreinte).

Chapitre 10

Quelques protocoles cryptographiques.

10.1 Protocoles de signature.

Nous avons déjà décrit des protocoles de signature RSA et El Gamal. Nous allons décrire un protocole de signature utilisable avec les cryptosystèmes symétriques et nous allons rappeler le protocole de signature pour un cryptosystème à clef publique déjà décrit.

10.1.1 Protocole de signature à clef privée.

Signature d'un document à l'aide d'un cryptosystème à clef secrète et d'un arbitre ou *tiers de confiance*.

Les utilisateurs d'un cryptosystème à clef publique se choisissent un arbitre, Yvan, c'est à dire une entité (personne, organisation, machine) en qui ils ont toute confiance.

Yvan communique avec Alice et Bob. Il partage une clé secrète K_A avec Alice et une clé secrète K_B avec Bob.

Le protocole de signature est alors le suivant

1. Alice chiffre son message pour Bob avec la clé K_A et envoie le résultat à Yvan.
2. Yvan déchiffre le message avec K_A et garde l'original.
3. Yvan assemble le message avec un avis certifiant que le message vient d'Alice. Il chiffre le tout avec K_B et l'envoie à Bob.
4. Bob déchiffre le message d'Yvan avec K_B . Il lit le message et le certifie d'Yvan.

5. Comme il a confiance en Yvan, il admet que le message reçu est conforme à l'original et vient bien d'Alice.

Ce protocole est-il sûr?

Les 3 qualités que l'on demande à un cryptosystème sont-elles bien présentes:

- Intégrité des données.
- Identités des différents interlocuteurs.
- Non-répudiation.

Bien sûr on suppose qu'Alice garde bien secrète la clef qu'elle partage avec Yvan et qu'Yvan est vraiment un tiers de confiance.

1. La signature est authentique car Yvan est respecté et seul Alice et lui connaissent K_A donc la message vient bien d'Alice.
2. La signature est infalsifiable. Seule Alice (et Yvan qui est insoupçonnable) connaît K_A , donc seule Alice peut envoyer le message codé avec K_A .
3. La signature n'est pas réutilisable pour un autre message. Si Bob essaie de prendre le certificat d'Yvan et de l'associer à un autre message censé provenir d'Alice, cette dernière crierait à l'imposture. L'arbitre demandera alors à Bob de produire le texte en clair. Il le chiffrerait avec K_A et comparerait avec le message original d'Alice. Il verra qu'ils sont différents et en informera qui de droit.
4. Le document est immuable. Si Bob falsifiait le message après l'avoir reçu, Yvan pourrait dévoiler l'imposture comme ci-dessus.
5. La signature ne peut pas être reniée. Si Alice prétend ne pas avoir envoyé le message, le certificat d'Yvan prouverait le contraire (tout le monde a confiance en Yvan).

Bob pourrait nier avoir reçu le message mais alors un système d'accusé de réception permettrait d'éviter ce problème.

10.1.2 Protocole de signature à clef publique.

Le protocole est simple

1. Alice chiffre le document d'une part avec sa clé privée, signant ainsi le document, et d'autre part avec la clé publique de Bob

2. Alice envoie les deux documents à Bob.
3. Bob déchiffre le premier document avec la clé publique d'Alice et le second avec sa clé privée. Si les deux sont identiques il est sûr qu'Alice est l'expéditeur.

Ce protocole est-il sûr?

Les 3 qualités que l'on demande à un cryptosystème sont-elles bien présentes:

- Intégrité des données.
- Identités des différents interlocuteurs.
- Non-répudiation.

Bien sûr on suppose que chacun des correspondants ne communique à personne sa clé secrète

1. La signature est authentique: quand Bob vérifie le message avec la clé publique d'Alice, il est sûr que seule Alice pouvait l'avoir crypté avec sa clé privée.
2. La signature est infalsifiable. Seule Alice connaît sa clé privée.
3. La signature n'est pas réutilisable. La signature est une fonction du document et elle ne peut pas être transférée sur n'importe quel autre document.
4. Le document est immuable. Si Bob falsifiait le message après l'avoir reçu, il ne pourrait pas le signer car la clé privée d'Alice n'est connue que d'elle seule.
5. La signature ne peut pas être reniée. Bob n'a pas besoin de l'aide d'Alice pour vérifier sa signature.

10.2 Protocoles de datation.

Dans certaines circonstances Bob peut duper Alice.

Il peut réutiliser le document (par exemple un chèque signé) et le présenter plusieurs fois à la banque. Pour l'éviter les signatures numériques comportent souvent une datation (date+heure). Cette datation de la signature est attachée au message et signée avec lui.

La banque stocke ces datations dans une base de données.

10.2.1 Protocole de datation.

On peut confier la datation des documents à un service officiel de datation. Bob veut dater une signature du message x . Il dispose d'une fonction de hachage à sens unique, h . Il suit alors le protocole suivant

1. Bob calcule $z = h(x)$ et $y = sig_K(z)$
2. Bob soumet (z, y) au service de datation.
3. Le service de datation ajoute la date D et signe le triplet (z, y, D)

Bob peut aussi dater un document, x , seul. Pour cela il collecte un certain nombre d'informations publiques récentes (qui n'auraient pas pu être prédites auparavant), notée pub . Par exemple les derniers résultats des courses hippiques et les cours actuels de la bourse. Il dispose aussi d'une fonction de hachage publique à sens unique, h . Bob suit alors le protocole suivant:

1. Bob calcule $z = h(x)$
2. Bob calcule $z' = h(z||pub)$
3. Bob calcule $y = sig_K(z')$
4. Bob publie (z, pub, y) dans le prochain quotidien

La date de la signature de Bob est comprise entre la date des informations pub et la date de parution du quotidien.

10.3 Protocole de signature à clef publique et fonction de hachage.

Les algorithmes à clef publique sont trop lents pour signer de longs documents.

Pour gagner du temps les protocoles de signature numérique sont souvent réalisés avec des fonctions de hachage à sens unique.

Au lieu de signer le document Alice signe l'empreinte du document en suivant le protocole suivant:

1. Alice calcule à l'aide de la fonction de hachage à sens unique, l'empreinte du document.

2. Alice chiffre, à l'aide de l'algorithme de signature numérique, cette empreinte avec sa clef privée, signant par la même occasion le document.
3. Alice envoie le document et l'empreinte signée à Bob (à l'aide de la clef publique de Bob).
4. Bob calcule, à l'aide de la fonction de hachage à sens unique, l'empreinte du document qu'Alice lui a envoyé. Ensuite à l'aide de l'algorithme de signature numérique, il déchiffre l'empreinte signée avec la clef publique d'Alice. La signature est valide si l'empreinte de la signature est la même que l'empreinte qu'il a produite.

Avantage de ce procédé:

- rapidité de la transmission et de la comparaison des empreintes car une empreinte ne comporte que 160 bits ou au plus 512.
- confidentialité car la signature peut être gardée à part du message. On peut donc vérifier l'existence du document sans stocker son contenu.

10.4 Fonction de hachage et mot de passe.

Pour accéder à un ordinateur (ou à un distributeur de billet) on utilise souvent un mot de passe qui doit être reconnu par l'ordinateur. S'il est stocké dans l'ordinateur il y a danger pour la sécurité car un ordinateur peut être facilement infiltré (virus). Grâce à la fonction de hachage on peut résoudre à ce problème:

On dispose d'une fonction de hachage à sens unique, h . L'ordinateur ne stocke pas le mot de passe mp d'Alice mais le résultat de la fonction de hachage à sens unique appliquée à mp . Le protocole est donc le suivant:

1. Alice envoie son mot de passe mp à l'ordinateur.
2. L'ordinateur calcule $h(mp)$
3. L'ordinateur compare le résultat de ce calcul à celui qu'il a dans sa base de données.

Ce protocole permet de se défendre contre le vol de la base des données des mots de passe des utilisateurs.

En fait on peut encore améliorer ce protocole avec les protocoles de preuve sans transfert de connaissance.

10.5 Preuve sans transfert de connaissance.

Un jeu de *Pile ou face par Téléphone*. Considérons la situation suivante, cf. [27].

Alice et Bob viennent de divorcer et habitent dans des villes différentes et veulent par téléphone tirer à pile ou face pour savoir à qui va échoir la voiture, la machine à laver, les livres, etc...

Mais Bob hésite à dire à Alice *face* pour s'entendre dire *voilà je jette une pièce,...., elle retombe,...., pas de chance c'est pile*.

Ils veulent un protocole qui évite toute tricherie.

Ils se donnent un ensemble E par exemple $E = \{0, 1, \dots, n\}$ et une partition $E = X_0 \cup X_1$ de E , X_0 sera les entiers pairs de E et X_1 les entiers impairs de E . Puis ils se mettent d'accord sur une fonction à sens unique, f , de E dans un ensemble F .

On considère le protocole suivant

1. Alice choisit un élément $x \in E$ aléatoirement (c'est le jet de la pièce), calcule $y = f(x)$ et communique y à Bob (Bob ne peut pas retrouver x à partir de y car f est à sens unique).
2. Bob choisit son bit aléatoire $b \in \{0, 1\}$ et l'annonce à Alice
3. Alice déclare qui a gagné suivant que $x \in X_b$ ou non: elle prouve sa bonne foi en révélant x
4. Bob se convainc qu'il n'y a pas eu tricherie en vérifiant que $y = f(x)$

Ce protocole est-il sûr:

1. Si la fonction f est bien choisie la donnée de $f(x)$ n'apprend rien à Bob sur x .
2. Pour qu'Alice ne puisse pas tricher, il faut s'assurer qu'elle ne peut pas fabriquer deux entiers $x_0 \in X_0$ et $x_1 \in X_1$ tels que $f(x_0) = f(x_1)$, par exemple on peut prendre f bijective car l'ensemble est de taille réduite.
3. f doit être à sens unique en un sens fort pour que la connaissance de $f(x)$ n'apprenne rien à Bob sur l'appartenance de x à X_0 ou X_1

Ce protocole met en évidence la notion d'*engagement*: Alice s'engage sur x en publiant $f(x)$. Cela ne révèle rien sur x mais force Alice à ne pas modifier son choix.

10.5.1 Protocole de preuve sans transfert de connaissances.

Le protocole précédent peut être utilisé dans beaucoup de situation où l'on veut prouver que l'on connaît un secret sans avoir besoin de le révéler.

Pour accéder à un ordinateur on donne son mot de passe qui est reconnu par l'ordinateur et donc stocké dedans. Si l'ordinateur auquel on se connecte est lointain, le mot de passe circule (crypté) sur des canaux qui risquent d'être espionnés.

Pour la sécurité il faudrait le changer souvent. Si la connexion est coupée accidentellement, il faut le redonner sans avoir la possibilité d'en changer.

Le clavier sur lequel on compose le mot de passe peut être espionné.

Il y a donc un intérêt à avoir un système de reconnaissance sans envoi de mot de passe même crypté.

Pour cela il faut disposer d'un secret personnel s et d'un protocole qui permet de persuader l'ordinateur (ou la personne) auquel on se connecte (s'adresse) qu'on connaît s sans avoir à le révéler et ce à chaque fois à l'aide de messages différents.

De tels protocoles existent ce sont des applications des idées de *complexité calculatoire* et de fonctions à sens unique.

Exemples 10 (Preuve de possession d'un logarithme discret). Cet exemple est tiré de [27]. On se donne p un nombre premier et g une racine primitive modulo p (théorème 11.3.13 page 94) qui sont publics et tels que le problème du logarithme discret soit un problème calculatoirement difficile.

Supposons que P (le prouveur) détienne le nombre secret s et soit identifié par $I = g^s \pmod{p}$. Il s'agit de convaincre V (le vérificateur) que P connaît s .

Considérons le protocole suivant:

1. P choisit un $r \pmod{p-1}$ aléatoire, il calcule $t = g^r \pmod{p}$ et le communique à V .
2. V choisit un bit aléatoire $\varepsilon = 0, 1$ et le communique à P
3. P donne x à V où
$$\begin{cases} x \equiv r \pmod{p-1} & \text{si } \varepsilon = 0 \\ x \equiv r + s & \text{si } \varepsilon = 1 \end{cases}$$

V vérifie alors que
$$\begin{cases} \alpha^x = t \pmod{p} & \text{si } \varepsilon = 0 \\ \alpha^x \equiv It \pmod{p} & \text{si } \varepsilon = 1 \end{cases}.$$

Comme dans le protocole de pile ou face P s'est engagé sur r en communiquant $t = \alpha^r$.

Que peut faire un imposteur P^* qui ne connaît pas s et veut se faire passer pour P auprès de V .

- Il peut suivre le protocole et espérer que $\varepsilon = 0$; dans le cas contraire il ne saura pas quel y communiquer à V .
- Il peut au contraire choisir un r aléatoire et communiquer à V la quantité $t' = \frac{\alpha^r}{I}$. Si V choisit $\varepsilon = 1$ alors P^* donne $x' = r$ et survit à la vérification $\alpha^{x'} = t'I$. Mais si $\varepsilon = 0$ alors P^* ne sait que faire.

Donc quel que soit la manière dont P^* s'y prenne il n'a qu'une chance sur deux de donner la bonne réponse.

1. Au bout de k applications du protocole, V est convaincu avec probabilité de $1 - \frac{1}{2^k}$ que son interlocuteur détient un logarithme de I
2. L'information révélée à V n'est qu'une liste d'entiers modulo p aléatoires et de leurs images par l'exponentielle $x \mapsto \alpha^x$. Le vérificateur V aurait pu aussi bien se la constituer seul en choisissant au hasard des entiers modulo p et en les exponentiant modulo p .
3. P n'a révélé aucune information sur son secret le logarithme de I .

C'est un exemple de *protocole interactif* et sans transfert de connaissance (zero knowledge proof).

Il y a d'autres protocoles qui permettent par exemple de réaliser un *transfert inconscient*.

10.5.2 Transfert inconscient.

Alice dispose d'un ensemble de m secrets $\{s_1, s_2, \dots, s_m\}$. Alice est prête à en donner (vendre) un à Bob.

Bob aimerait obtenir le secret s_i , mais il ne souhaite pas révéler à Alice lequel des secrets l'intéresse. Il existe des protocoles, [27], qui donnent une solution à ce problème:

76 CHAPITRE 10. QUELQUES PROTOCOLES CRYPTOGRAPHIQUES

1. ils permettent à Bob de disposer de s_i
2. ils ne lui donnent aucune information sur les autres secrets s_j , $j \neq i$
3. ils ne permettent pas à Alice de savoir quel secret elle a livré à Bob

Chapitre 11

Rappels Mathématiques.

11.1 Théorie de l'information.

La théorie de l'information qui est due à Claude Shannon dans un article fondateur paru en 1949, [18], utilise quelques notions de probabilité qui sont rappelées ci-dessous.

11.1.1 Rappels de probabilités discrètes.

Définition 11.1.1. Une variable aléatoire discrète, \mathbf{X} , consiste en un ensemble X , une distribution de probabilités discrètes $(p_x)_{x \in X}$ sur X et de la donnée $\Pr[\mathbf{X} = x]$: la probabilité que X se réalise en x . Par définition on a

$$\forall x \in X, 0 \leq \Pr[\mathbf{X} = x] \leq 1, \quad \text{et} \quad \sum_{x \in X} \Pr[\mathbf{X} = x] = 1$$

Exemples 11. Le jet d'une pièce de monnaie est une variable aléatoire, \mathbf{X} , définie sur l'ensemble $X = \{\text{pile}, \text{face}\}$, la probabilité associée étant $\Pr[\mathbf{X} = \text{pile}] = \Pr[\mathbf{X} = \text{face}] = \frac{1}{2}$

Exemples 12. Jet aléatoire d'une paire de dés. On peut le modéliser par la variable aléatoire \mathbf{Z} sur l'ensemble

$$Z = \{1, 2, 3, 4, 5, 6\} \times \{1, 2, 3, 4, 5, 6\}$$

muni de la probabilité $\Pr[\mathbf{Z} = (i, j)] = \Pr[(i, j)] = \frac{1}{36}$. Si on veut calculer la probabilité pour que la somme des deux dés lors d'un lancer soit 4. Cette valeur correspond à l'évènement

$$S_4 = \{(1, 3), (2, 2), (3, 1)\} \implies \Pr[S_4] = \frac{3}{36} = \frac{1}{12}$$

Pour définir la confidentialité parfaite on introduit les définitions suivantes:

Définition 11.1.2. Soient deux variables aléatoires \mathbf{X} et \mathbf{Y} définies sur des ensembles finis X et Y respectivement. La probabilité mutuelle $\Pr[\mathbf{X} = x, \mathbf{Y} = y] = \Pr[x, y]$ est la probabilité pour que \mathbf{X} se réalise en x et \mathbf{Y} se réalise en y .

La probabilité conditionnelle $\Pr[\mathbf{X} = x | \mathbf{Y} = y] = \Pr[x | y]$ est la probabilité que \mathbf{X} se réalise en x sachant que \mathbf{Y} s'est réalisé en y .

Définition 11.1.3. Les variables aléatoires \mathbf{X} et \mathbf{Y} sont dites des *variables aléatoires indépendantes* si $\Pr[x, y] = \Pr[x|y] \Pr[y]$ pour tout $x \in X$ et tout $y \in Y$

On a la proposition

Proposition 11.1.4. On a la relation entre probabilité mutuelle et probabilité conditionnelle

$$\Pr[x, y] = \Pr[x|y] \Pr[y]$$

Démonstration: C'est évident. □

On a le théorème important suivant

Théorème 11.1.5 (Théorème de Bayes). Si $\Pr[y] > 0$, on a

$$\Pr[x|y] = \frac{\Pr[x] \Pr[y|x]}{\Pr[y]}$$

Démonstration: D'après la proposition 11.1.4 on a

$$\Pr[x, y] = \Pr[x|y] \Pr[y]$$

et en échangeant x et y on aussi $\Pr[x, y] = \Pr[y|x] \Pr[x]$. □

11.1.2 Confidentialité parfaite.

On considère un cryptosystème $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ (définition 5.0.2 page 24) et on suppose qu'une clef $K \in \mathcal{K}$ n'est utilisée qu'une fois.

On suppose que l'espace des textes clairs \mathcal{P} est muni d'une distribution de probabilités associée à une variable aléatoire \mathbf{x} sur \mathcal{P} , on définit $\Pr[\mathbf{x} = x]$ comme étant la *probabilité a-priori d'occurrence du texte clair x* .

De même on suppose que l'espace des clefs \mathcal{K} est muni d'une distribution de probabilités associée à la variable aléatoire \mathbf{K} sur l'espace des clefs \mathcal{K} , on

définit $\Pr[\mathbf{K} = K]$ la *probabilité pour que la clé K soit utilisée* (souvent on suppose les clés équiprobables).

Rappelons que la clé est toujours choisie par Alice et Bob avant de savoir quel message Alice va transmettre. On peut donc supposer que les variables aléatoires \mathbf{x} et \mathbf{K} sont indépendantes.

Les deux distributions de probabilités sur \mathcal{P} et \mathcal{C} induisent une distribution de probabilité sur l'espace des messages chiffrés \mathcal{C} associée à la variable aléatoire \mathbf{y} . Un tel système sera dit probabilisé.

On pose pour $K \in \mathcal{K}$ fixé et pour $e_K \in \mathcal{E}$ la fonction de chiffrement associée

$$C(K) = \{e_K(x) : x \in \mathcal{P}\}$$

$C(K)$ est l'ensemble des messages chiffrés avec la clé K . Pour tout $y \in \mathcal{C}$ on a

$$\Pr[\mathbf{y} = y] = \sum_{K \in \mathcal{K}, y \in C(K)} \Pr[\mathbf{K} = K] \Pr[x = d_K(y)]$$

où $d_K \in \mathcal{D}$ est la fonction de décodage associée à la clé K .

On a par application du théorème de Bayes

$$\Pr[\mathbf{x} = x | \mathbf{y} = y] = \frac{\Pr[\mathbf{x} = x] \times \sum_{K \in \mathcal{K}, y \in C(K)} \Pr[\mathbf{K} = K]}{\sum_{K \in \mathcal{K}, y \in C(K)} \Pr[\mathbf{K} = K] \Pr[x = d_K(y)]}$$

On pose la définition

Définition 11.1.6. *Un système cryptographique probabilisé $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ assure une confidentialité parfaite si $\Pr[x|y] = \Pr[x]$ pour tout $x \in \mathcal{P}$ et tout $y \in \mathcal{C}$, c'est à dire si la probabilité a-posteriori que le texte clair soit x sachant que le texte chiffré est y est égale à la probabilité a-priori que le texte clair soit x .*

On démontre les théorèmes suivants

Théorème 11.1.7. *Si les vingt-six clefs d'un chiffrement par décalage (code de Cesar) sont utilisées avec la même probabilité $\frac{1}{26}$ alors pour toute distribution d'un bloc de texte clair on a confidentialité parfaite.*

Théorème 11.1.8. *Un système cryptographique probabilisé $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ tel que $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$ assure une confidentialité parfaite si et seulement si chaque clé est utilisée avec la même probabilité $\frac{1}{|\mathcal{K}|}$ et si pour chaque $x \in \mathcal{P}$ et chaque $y \in \mathcal{C}$, il existe une clé K unique telle que $e_K(x) = y$*

Ce théorème est une mise en forme des remarques faites lors de l'étude du code de Vernam. On voit bien la différence qu'il y a entre confidentialité parfaite et code incassable. Un code de César est à confidentialité parfaite d'après le théorème précédent pourtant il est facilement cassable.

11.1.3 Entropie.

Pour étudier la situation plus réaliste où une clé est utilisée plusieurs fois, Claude Shannon a introduit la notion d'*entropie* qui permet de modéliser l'information révélée à un opposant lors d'utilisations multiples d'une même clé.

Définition 11.1.9. *Soit une variable aléatoire \mathbf{X} définie sur un ensemble fini X . L'entropie de la variable aléatoire \mathbf{X} est définie comme étant la quantité*

$$H(\mathbf{X}) = - \sum_{x \in X} \Pr[x] \log_2 \Pr[x]$$

C'est une notion qui vient de la thermodynamique.

On définit ensuite l'entropie $H(L)$ d'un langage naturel L comme le français ou l'anglais. On essaye de quantifier les informations qu'apporte le fait qu'en français le Q est presque toujours suivi du U et qu'un S est assez souvent suivi d'un autre S etc...

On se donne un cryptosystème $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ en langue naturelle L . On définit sa *redondance*

$$R_L = 1 - \frac{H_L}{\log_2 |\mathcal{P}|}$$

On choisit une clef et un ensemble fini de messages clairs que l'on chiffre avec cette clé. On appelle *clefs parasites* les clefs de \mathcal{K} qui donnent les mêmes messages chiffrés pour le même ensemble de message.

Définition 11.1.10. *La distance d'unicité d'un cryptosystème est la plus petite valeur n , notée n_0 , telle que le nombre moyen de clefs parasites soit nul. C'est la quantité de texte chiffré nécessaire à un opposant disposant de suffisamment de temps de calcul pour déterminer la clef.*

On montre que $n_0 \sim \frac{|\mathcal{K}|}{R_L \log_2 |\mathcal{P}|}$. Par exemple pour le chiffrement par substitution sur les 26 lettres de l'alphabet $|\mathcal{P}| = 26$, $|\mathcal{K}| = 26!$, si l'on prend $R_L = 0,75$ qui est une valeur admise pour l'anglais, il vient $n_0 \sim 25$. Ceci montre que pour un message de 25 lettres il n'y a en principe (en moyenne) qu'un seul déchiffrement possible (cf. les alphabets T9 sur les portables).

11.2 Théorie de la complexité.

Le but de cette théorie est de classifier les problèmes algorithmiques en fonction de leur difficulté.

Il faut donc un modèle d'ordinateur. Le plus utilisé est celui des *machines de Turing* (Alan Turing, 1912-1954).

Une machine de Turing est la donnée de

1. un alphabet fini Σ
2. une bande infinie dans les deux sens formée de cases. Dans chaque case peut être inscrit au plus un symbole de Σ . Les cases vides sont marquées par un symbole spécial \emptyset . On pose $\Sigma' = \Sigma \cup \emptyset$. la bande modélise la mémoire de l'ordinateur. A chaque instant seul un nombre fini de cases de la bande contient un symbole de Σ .
3. une tête de lecture/écriture qui se déplace d'une case à l'autre.
4. un ensemble fini d'états Q , l'un d'entre eux étant appelé l'état initial.
5. un programme ou fonction de transition, composé d'un tableau, indexé par Q et Σ' .

Pour chaque couple $(q, s) \in Q \times \Sigma'$ le programme possède au plus une instruction qui sera exécutée quand la machine sera dans l'état q et que la tête de lecture lira le symbole s .

Cette instruction est codée

$$(q', s', d) \text{ avec } q' \in Q, s' \in \Sigma'$$

$$d \in \{\text{gauche, droite}\}.$$

Son exécution consiste à écrire le symbole s' à la place de s à déplacer la tête de lecture dans la direction d et à placer la machine dans l'état q' .

Pour faire fonctionner la machine de Turing on inscrit une suite finie de symboles $\in \Sigma$ sur la bande. On place la machine dans l'état initial et on positionne la tête de lecture sur la première case à gauche contenant un symbole de Σ .

Ensuite la machine exécute le programme et s'arrête quand elle ne possède pas d'instructions correspondant au symbole lu et l'état où elle se trouve.

Exemples 13. Testeur de parité. Cette machine teste la parité du nombre de 1 dans un nombre n écrit en base 2. Elle s'arrête dans l'état q_i si n a un nombre impair de 1 et elle s'arrête dans l'état q_p si n possède un nombre pair de 1.

$\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_p, q_i\}$, état initial q_0 , fonction de transition

	0	1	\emptyset
q_0	$(q_0, \emptyset, \text{droite})$	$(q_1, \emptyset, \text{droite})$	$(q_p, \emptyset, \text{droite})$
q_1	$(q_1, \emptyset, \text{droite})$	$(q_1, \emptyset, \text{droite})$	$(q_i, \emptyset, \text{droite})$

Il y a aussi des machines de Turing non déterministes qui peuvent avoir plusieurs instructions possibles pour un couple de (Q, Σ') . La machine choisit "au hasard" quelle instruction exécuter.

11.2.1 Décidabilité.

On introduit les définitions suivantes

Définition 11.2.1. Soit Σ un alphabet et Σ^* l'ensemble des mots sur Σ . Un langage sur Σ est une partie de Σ^*

Définition 11.2.2. Soit M une machine de Turing sur l'alphabet Σ et q_y un état de M . On dit que M accepte la donnée x (par l'état q_y) si M s'arrête dans l'état q_y lorsque la donnée est x .

L'ensemble des mots acceptés par M s'appelle le langage reconnu par M que l'on note

$$L_{q_y}(M) = L(M)$$

Le complémentaire de $L(M)$ dans Σ^* est l'ensemble des mots pour lesquels soit M ne s'arrête pas soit M s'arrête dans un état $\neq q_y$.

Définition 11.2.3. Un problème de décision \mathcal{D} appartient à la classe **P** s'il existe une machine de Turing déterministe qui le résout en un temps polynomial en fonction de la taille des données (i.e. du nombre de symboles de Σ apparaissant dans la donnée). Si $\Sigma = \{0, 1\}$ alors le problème de décision appartient à la classe **P** des problèmes polynomiaux en temps si

$$\forall x \in \mathcal{D} \Rightarrow \mathcal{C}_{\text{temps}}(x) = O(\text{polynôme en } \log_2(x))$$

Un problème de décision \mathcal{D} appartient à la classe NP des problèmes non polynomiaux en temps s'il existe une machine de Turing non-déterministe qui le résoud en un temps polynomial

Conjecture 1. $P \neq NP$.

Autrement dit il existe au moins un problème pour lequel on peut montrer qu'il n'existe pas de machine de Turing déterministe pour le résoudre en temps polynomial et qui peut être résolu en temps polynomial par une machine de Turing non-déterministe.

11.2.2 Complexité algorithmique.

Si on veut exécuter un algorithme sur une donnée x deux coûts sont à envisager.

- le *coût en temps* $\mathcal{C}_{\text{temps}}(x)$, c'est à dire le nombre d'opérations effectuées pour obtenir le résultat final ou plus formellement le nombre de déplacements de la tête de lecture/écriture avant arrêt de la machine de Turing modélisant l'ordinateur.
- le *coût en espace*, $\mathcal{C}_{\text{espace}}(x)$ est la taille de la mémoire utilisée, plus formellement le nombre de case de la bande écrite au moins une fois.

On distingue essentiellement deux notions de complexité algorithmique la *complexité polynomiale* et la *complexité non polynomiale* (sous entendu en fonction de la taille des données).

Quel est le coût acceptable pour un problème donné. Il n'y a ni réponse claire ni réponse absolue. En 2000 on estimait que

- 2^{40} opérations élémentaires est accessible à un particulier s'il est patient.
- 2^{56} opérations élémentaires est accessible avec de gros moyens.
- 2^{80} opérations élémentaires est hors de portée de quiconque.

Soit un algorithme dont le nombre d'opérations élémentaires en fonction de la taille n de l'entrée est décrit par la fonction f . On dispose d'un ordinateur faisant 10^9 opérations élémentaires par secondes. Le temps pris par l'algorithme suivant les instances de f est:

f	$\log_2(n)$	$\log_2^3(n)$	n	$n \log_2(n)$	n^2	2^n
$n = 100$	$6,6 \cdot 10^{-9}$ s	$4,4 \cdot 10^{-7}$ s	7,5 s	$6,7 \cdot 10^{-7}$ s	10^{-5} s	$4 \cdot 10^{13}$ ans
$n = 10^5$	$1,7 \cdot 10^{-8}$ s	$5,2 \cdot 10^{-6}$ s	10^{-4} s	$1,7 \cdot 10^{-3}$ s	10 s	$\geq 10^{30086}$ ans
$n = 10^{10}$ s	$3,3 \cdot 10^{-8}$	$3,5 \cdot 10^{-5}$ s	10 s	330 s	3 siècles	$\geq 10^{3 \cdot 10^9}$ ans
$n = 10^{20}$	$6,6 \cdot 10^{-8}$ s	$2,3 \cdot 10^{-4}$ s	30 siècles	$\geq 10^5$ ans	$\geq 10^{23}$ ans	!!!

11.2.3 Algorithmes polynomiaux en fonction de la taille des données.

On décompose tout algorithme arithmétique en *opérations élémentaires*.

L'opération élémentaire est l'addition de 3 chiffres en base 2 et le report de la retenue

$$\text{retenue} + \text{chiffre 1} + \text{chiffre 2} = \text{résultat} + \text{retenue}$$

Définition 11.2.4. On dit qu'un algorithme est polynomial en fonction de la taille des données s'il peut être décomposé en un nombre d'opérations élémentaires majoré par une fonction polynomiale du nombre de chiffres des données.

Exemples 14. Montrons que l'addition de deux nombres de tailles $\leq k$ est une fonction polynomiale de k

$$\begin{array}{r}
 \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & & & \\ \hline & 1 & 1 & 1 & 0 & 0 & \\ \hline & 0 & 0 & 1 & 1 & 1 & \\ \hline \end{array} & \begin{array}{l} \text{retenue} \\ 0 \text{ ligne 1} \\ 0 \text{ ligne 2} \end{array} \\
 + \\
 \hline
 \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline \end{array} & 0 \text{ résultat}
 \end{array}$$

On suppose que le plus grand des entiers a k chiffres. Décomposons la somme en opérations élémentaires

1. Regarder dans la colonne i ($1 \leq i \leq k+1$) les chiffres des lignes *lignes 1, 2 et retenue*
2. S'il y a 0 dans les *lignes 1, 2 et retenue* mettre 0 dans la ligne *résultat* et aller à la colonne $i+1$

3. S'il y a un 0 dans deux des *lignes 1 et 2 et retenue* on reporte 1 dans la ligne *résultat* et on passe à la colonne $i + 1$
4. S'il y a un 1 dans deux des *lignes 1 et 2 et retenue* on reporte 0 dans la ligne *résultat* on met 1 dans la ligne *retenue* à la colonne $i + 1$ et on passe à la colonne $i + 1$
5. S'il y a un 1 dans les 3 *lignes 1 et 2 et retenue* on reporte 1 dans la ligne *résultat* on met 1 dans la ligne *retenue* à la colonne $i + 1$ et on passe à la colonne $i + 1$

On admet en première approximation que le temps nécessaire pour faire k opérations élémentaires est proportionnel à k avec une constante dépendant de l'ordinateur et de l'implantation de l'algorithme.

Exemples d'algorithmes polynomiaux en fonction de la taille des entrées:

- L'addition de deux entiers $n \leq m$ de longueur $\leq k$ est polynomiale en fonction de la taille des entrées car (en première approximation) il faut

$$C \cdot k \stackrel{\text{déf}}{=} O(k) = O(\log_2(m)) \text{ opérations}$$

Ajouter deux entiers de 100 chiffres en base 10 avec un ordinateur faisant 10^9 opérations par secondes nécessite $\sim 300 \cdot 10^{-9} \text{sec} \sim 3\mu\text{sec}$.

- La multiplication de deux entiers $n \leq m$ de taille k et ℓ nécessite

$$2 \cdot k\ell = O(\log_2^2(m)) \text{ opérations}$$

Multiplier deux entiers de 100 chiffres en base 10 nécessite $\sim 90000 \cdot 10^{-9} \text{sec} \sim 10\mu \text{sec}$.

- Calcul du PGCD de deux entiers, $n \geq m$ de taille $k \geq \ell$ nécessite

$$O(k^3) = O(\log_2^3(n)) \text{ opérations}$$

Trouver le PGCD de deux entiers de 100 chiffres en base 10 nécessite $\sim 27000000 \cdot 10^{-9} \sim 0,027 \text{sec}$.

- Calculer $b^m \bmod n$ nécessite $O(\log_2^3(n))$ opérations.
- Décider si un nombre entier n de taille k est premier ou non nécessite

$$O(k^{6+\varepsilon}) = O(\log_2^{6+\varepsilon}(n)) \text{ opérations}$$

Tester si un entier de 100 chiffres en base 10 est premier nécessite avec l'algorithme polynomial $\sim 10^{26} \mu \text{ sec} \sim 10^9$ années. Il y a des algorithmes pour tester la primalité d'un entier non polynômiaux et/ou non déterministes qui sont plus efficaces pour des nombres pas trop grands.

le dernier exemple montre que si le polynôme est de degré trop élevé (≥ 3), un algorithme polynomial peut avoir un coût prohibitif.

Exemples d'algorithmes non polynômiaux en fonction de la taille des entrées

- Calcul de $n! = 1 \cdot 2 \cdot \dots \cdot (n-1) \cdot n$, si l'on calcule brutalement il faut environ

$$C \cdot n^2 \log_2^2 n = O(n^2 \log_2^2 n) \text{ opérations}$$

Calculer $(10^{100})!$ nécessite $\sim 10^{190} \mu \text{ sec} \sim 10^{180}$ années!!!

- Recherche d'un diviseur d'un entier n , les meilleurs algorithmes nécessitent

$$O\left(e^{C \sqrt{\log_2 n \log_2 \log_2 n}}\right) \text{ opérations, } C \leq 2$$

Trouver un diviseur d'un nombre de 100 chiffres en base 10 nécessite $\sim e^{30} \cdot 10^{-9} \sim 10^5 \text{ sec} \sim 1,1$ jours. Un tel algorithme est dit sous-exponentiel.

11.3 Rappels d'arithmétique.

Ainsi qu'on l'a vu dans la description des cryptosystèmes à clé publique RSA et El Gamal, leur principe est basé sur de l'arithmétique élémentaire.

11.3.1 La division euclidienne.

Les entiers naturels $\mathbb{N} = \{1, 2, 3, \dots\}$ sont munis de deux opérations internes l'addition, notée $+$, et la multiplication, notée \times ou \cdot ou même sans symbole, et d'une relation d'ordre total, notée \leq , compatible avec l'addition et la multiplication c'est à dire

$$(\forall a, b \in \mathbb{N}, a \leq b) \implies \forall c \in \mathbb{N}; a + c \leq b + c$$

$$(\forall a, b \in \mathbb{N}, a \leq b) \implies \forall c \in \mathbb{N}; a \times c \leq b \times c$$

On définit sur \mathbb{N} la *division euclidienne*

Définition 11.3.1. Si a et b sont deux entiers ($b \neq 0$) il existe un unique couple d'entiers q et r tel que

$$a = b \cdot q + r \quad \text{et} \quad \begin{cases} \text{ou bien} & r = 0 \\ \text{ou bien} & 1 \leq r \leq b - 1 \end{cases}$$

on dit que b est un diviseur de a s'il existe $q \in \mathbb{N}$ tel que $a = bq$, on note $a | b$ pour dire a divise b .

Tout entier a possède au moins deux diviseurs triviaux 1 et lui-même car on a toujours

$$a = 1 \cdot a, \quad a = a \cdot 1$$

Mais il peut en avoir d'autres par exemple

$$6 = 2 \cdot 3, \quad 28 = 4 \cdot 7 = 2 \cdot 2 \cdot 7 = 2 \cdot 14$$

donc 1, 2, 3 et 6 sont des diviseurs de 6 et 1, 2, 4, 7, 14 et 28 sont des diviseurs de 28.

La division euclidienne implique que \mathbb{Z} , les entiers relatifs, est un *anneau euclidien*, i.e. qu'il est muni de deux lois de composition interne l'addition et la multiplication qui possèdent certaines propriétés et d'une division euclidienne, cf. [19],

1. La loi $+$ est commutative (pour tout $(a, b) \in \mathbb{Z}^2$ on a $a + b = b + a$), associative (pour tous $(a, b, c) \in \mathbb{Z}^3$ on a $(a + b) + c = a + (b + c)$), il y a un élément neutre, 0, tel que pour tout $a \in \mathbb{Z}$ on a $a + 0 = 0 + a = a$, tout élément $a \in \mathbb{Z}$ possède un opposé $b = -a$ tel que $a + b = b + a = 0$.
2. La loi \times est commutative ($a \times b = b \times a$), associative ($(a \times b) \times c = a \times (b \times c)$), il y a un élément neutre, 1, ($a \times 1 = 1 \times a = a$), la multiplication est distributive par rapport à l'addition ($(a + b) \times c = (a \times c) + (b \times c)$)

Dans les entiers naturels on distingue

- 1 qui est une *unité*, c'est à dire que 1 possède un inverse pour la multiplication (il existe $b \in \mathbb{N}$ tel que $1 \times b = b \times 1 = 1$) qui est 1.
- les *nombre premiers* qui n'ont pas d'autres diviseurs que les diviseurs triviaux i. e. 1 et lui même (e.g. 2,3,5,7,...,37,...).

Le théorème suivant qui découle de l'existence de la division euclidienne est appelé le théorème fondamental de l'arithmétique

Théorème 11.3.2. *Tout nombre entier différent de 1 possède une unique décomposition en facteurs premiers à l'ordre près des facteurs, certains facteurs peuvent être répétés.*

Démonstration: Pour la preuve voir [19] □

Autrement dit si $a \in \mathbb{N}$, $a \leq 2$, alors il existe des nombres premiers p_1, p_2, \dots, p_{n_a} non nécessairement distincts tels que

$$a = p_1 \times p_2 \times \dots \times p_n, \quad p_i \text{ premier pour } 1 \leq i \leq n$$

l'entier n dépend de a . L'unicité à l'ordre près des facteurs impose que si on a une égalité

$$a = q_1 \times q_2 \times \dots \times q_m, \quad q_j \text{ premier pour } 1 \leq j \leq m$$

alors nécessairement $n = m$ et il existe une permutation ψ des entiers de 1 à n_a telle que

$$p_1 = q_{\psi(1)}, p_2 = q_{\psi(2)}, \dots, p_n = q_{\psi(n)}$$

On peut écrire la décomposition en facteurs premiers de $a \in \mathbb{N}$ en regroupant tous les nombres premiers égaux, alors

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_\ell^{\alpha_\ell} \text{ avec } p_1 < p_2 < \dots < p_\ell, \quad p_i \text{ premier pour } 1 \leq i \leq \ell$$

L'unicité à l'ordre près des facteurs impose que si on a une égalité

$$a = q_1^{\beta_1} q_2^{\beta_2} \dots q_k^{\beta_k} \text{ avec } q_1 < q_2 < \dots < q_\ell, \quad q_j \text{ premier pour } 1 \leq j \leq k$$

alors

$$\ell = k \text{ et } \alpha_i = \beta_i, \text{ pour } 1 \leq i \leq \ell$$

Exemples 15. $10780 = 2^2 \cdot 5 \cdot 7^2$, $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11$

11.3.2 Plus Grand Commun Diviseur ou PGCD.

Le *plus grand commun diviseur* abrégé en *PGCD* en français et *GCD* (Great-est Common Divisor) en anglais est défini de la manière suivante

Définition 11.3.3. *Le plus grand commun diviseur de deux nombres entiers a et b est le plus grand des entiers qui divisent à la fois a et b , on le note $PGCD(a, b)$ ou $a \wedge b$ ou encore (a, b) .*

L'ensemble $\{d \in \mathbb{N} \mid d \text{ divise } a \text{ et } d \text{ divise } b\}$ est non vide, car 1 divise toujours a et b , et il est borné par le plus grand des deux entiers a et b , donc le PGCD existe et de plus il est toujours supérieur ou égal à 1.

Proposition 11.3.4. *Le plus grand commun diviseur de a et de b , entiers naturels, s'obtient de la manière suivante. Si*

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_r^{\alpha_r}, \quad b = p_1^{\beta_1} p_2^{\beta_2} \dots p_r^{\beta_r}, \quad \text{avec } \alpha_i, \beta_i \geq 0$$

sont leur décomposition en facteurs premiers, alors le PGCD de a et de b est:

$$a \wedge b = p_1^{\inf\{\alpha_1, \beta_1\}} p_2^{\inf\{\alpha_2, \beta_2\}} \dots p_r^{\inf\{\alpha_r, \beta_r\}}$$

Démonstration: Pour la preuve voir [19] □

Pour calculer le PGCD ni la définition 11.3.3 ni la proposition 11.3.4 ne sont efficaces car elles nécessitent l'une comme l'autre de trouver des diviseurs de a et de b . Or les meilleurs algorithmes connus pour trouver un diviseur non trivial de a ou pour décomposer a en facteurs premiers ce qui revient au même sont en $O(e^{C\sqrt{\log_2(a) \log_2 \log_2(a)}})$. Ces algorithmes ne sont donc pas polynomiaux en temps en fonction de la taille des données, leur coût est prohibitif pour des grands nombres entiers (500 chiffres en base 10 par exemple) même avec des ordinateurs très puissants.

Exemples 16. Le PGCD de $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7$ et de $10780 = 2^2 \cdot 5 \cdot 7^2 \cdot 11$ est

$$\text{PGCD}(4200, 10780) = 4200 \wedge 10780 = 2^2 \cdot 5 \cdot 7 = 140$$

Par contre l'algorithme de la division euclidienne fournit grâce au théorème de Bézout un algorithme très efficace, polynomial en temps en fonction de la taille des données, pour calculer le PGCD de deux entiers naturels.

11.3.3 Algorithme du plus grand commun diviseur ou algorithme du PGCD.

Considérons le problème suivant:

Soit $a \geq b$ deux entiers de taille k au plus. Trouver le PGCD $a \wedge b$ de a et de b et évaluer le nombre d'opérations élémentaires nécessaires.

Proposition 11.3.5. *On écrit les divisions euclidiennes successives*

$$(11.1) \quad \begin{array}{ll} a = bq_0 + r_1 & b = q_1r_1 + r_2 \\ r_1 = q_2r_2 + r_3 & \dots \\ \vdots & \\ r_{j-2} = q_{j-1}r_{j-1} + r_j & \\ r_{j-1} = q_jr_j + r_{j+1} & r_j = q_{j+1}r_{j+1} \end{array}$$

avec $0 \leq r_1 < b$, $0 \leq r_2 < r_1$, $0 \leq r_3 < r_2, \dots$, $0 \leq r_j < r_{j+1}$.

On arrête l'algorithme dès qu'un reste est nul et alors

$$a \wedge b = r_{j+1}$$

Démonstration: En effet la dernière identité de division euclidienne de (11.1) montre que r_{j+1} divise r_j , l'avant dernière identité de division euclidienne de (11.1) montre que r_{j+1} divise r_j et r_{j-1} . Puis par récurrence on montre que r_{j+1} divise a et b .

Donc r_{j+1} est un diviseur commun de a et de b , est-ce le plus grand? Considérons un diviseur commun de a et de b noté d . Par définition d divise a et b donc d'après la première des identités de division euclidienne de (11.1) on a $d \mid r_1$, alors la deuxième des identités de division euclidienne de (11.1) implique que $d \mid r_2$ puis par récurrence on montre que $d \mid r_{j+1}$.

On a donc montré que tout diviseur d commun de a et de b est un diviseur de r_{j+1} et comme r_{j+1} est un diviseur de a et de b c'est le plus grand, autrement dit $r_{j+1} = a \wedge b$. \square

Cet algorithme nécessite $O(\log_2^3(n))$ opérations élémentaires.

Exemples 17. Trouver le PGCD de 4200 et 10780:

$$\begin{aligned} 10780 &= 2 \times 4200 + 2380; & 4200 &= 2380 + 1820; \\ 2380 &= 1 \times 1820 + 560, & 1820 &= 3 \times 560 + 140, & 560 &= 4 \times 140 \\ \text{PGCD}(10780, 4200) &= 140 \end{aligned}$$

L'algorithme de la division euclidienne donne aussi une version effective du théorème de Bézout, autrement dit une version effective du calcul du générateur de l'idéal $\langle a, b \rangle$ engendré par a et b .

Théorème 11.3.6 (Théorème de Bézout). *Soit a et b deux entiers naturels de PGCD: $a \wedge b = d$. Alors il existe deux entiers relatifs u, v tels que*

$$d = au + bv$$

u et v sont appelés les coefficients de Bézout.

Démonstration: on reprend l'algorithme du PGCD à partir de la fin. On écrit avec les notations précédentes

$$d = r_{j+1} = r_{j-1} - q_j r_j = r_{j-1} u_{j-1}(1) - r_j v_j(q_j)$$

$$\begin{aligned}
&= r_{j-1} - q_j(r_{j-2} - q_{j-1}r_{j-1}) \\
&= r_{j-1}(1 + q_jq_{j-1}) - q_jr_{j-2} \\
&= -u_{j-2}(q_j)r_{j-2} + r_{j-1}v_{j-1}(q_{j-1}, q_j) \\
&= (r_{j-3} - q_{j-2}r_{j-2})(1 + q_jq_{j-1}) - q_jr_{j-2} \\
&= r_{j-3}(1 + q_jq_{j-1}) - r_{j-2}(q_{j-2}(1 + q_jq_{j-1}) + q_j) \\
d &= r_{j-3}u_{j-3}(q_j, q_{j-1}) - r_{j-2}v_{j-2}(q_{j-2}, q_{j-1}, q_j) \\
&\quad \vdots \\
&= (-1)^{j+1}au_0(q_1, \dots, q_j) + (-1)^j bv_0(q_0, \dots, q_j)
\end{aligned}$$

On a une relation de récurrence facile sur u_j et v_j .

$$\begin{aligned}
u_{j+1} &= u_{j-1} - q_{j+1}u_j \\
v_{j+1} &= v_{j-1} - q_{j+1}v_j \quad \square
\end{aligned}$$

Exemples 18. Relation de Bézout entre 10780 et 4200:

$$\begin{aligned}
140 &= 1820 - 3 \times 560 = 1820 - 3 \times (2380 - 1820) \\
&= 4 \times 1820 - 3 \times 2380 = 4 \times (4200 - 2380) - 3 \times 2380 \\
&= 4 \times 4200 - 7 \times 2380 = 4 \times 4200 - 7 \times (10780 - 2 \times 4200) \\
&= 18 \times 4200 - 7 \times 10780 = 75600 - 75460
\end{aligned}$$

Remarque 11.3.1. Le couple (u, v) tel que $au + bv = d = a \wedge b$ dont l'existence est garantie par le théorème de Bézout n'est pas unique car si (u_0, v_0) est un tel couple et si (x_1, y_1) est une solution en nombres entiers relatifs de l'équation

$$ax_1 + by_1 = 0$$

alors on a encore

$$a(u + x_1) + b(v + y_1) = d$$

Par contre on a unicité au signe près si on impose

$$1 \leq |u| \leq b, \quad 1 \leq |v| \leq a$$

Algorithme d'Euclide.

On donne une version de type programme de l'algorithme d'Euclide.

- Entrée 2 entiers positifs a et b non tous deux nuls

- Sortie $a \wedge b$
 - Tant que $b \neq 0$ faire

$$a \wedge b := b \wedge (a \bmod b)$$

- si $b = 0$ retourner a et fin

Algorithme d'Euclide étendu.

On donne une version récursive de type programme de l'algorithme d'Euclide étendu qui donne les coefficients de Bézout pour le PGCD de deux entiers.

1. Entrée 2 entiers positifs a et b non tous nuls
2. Sortie 3 entiers u, v, d tels que $au + bv = a \wedge b = d$
 - (a) Variables entières $u_1, v_1, u_2, v_2, u_3, v_3, q, r$
 - (b) Si $a = 0$ retourner $(0, 1, b)$ et fin
 - (c) sinon $Au_1 + Bv_1 = a$ et $Au_2 + Bv_2 = b$ où A et B sont les valeurs initiales de a et b
 - (d) Faire

$$(u_1, v_1) := (1, 0) \quad (u_2, v_2) := (0, 1)$$
 - (e) Tant que $b \neq 0$ faire

$$(q, r) := \text{quotient et reste de la division euclidienne de } a \text{ par } b$$

$$(u_3, v_3) := (u_1 - qu_2, v_1 - qv_2)$$

$$(u_1v_1, a) := (u_2, v_2, b)$$

$$(u_2, v_2, b) := (u_3, v_3, r)$$
3. si $b = 0$ retourner (u_1, v_1, a) .

11.3.4 Les Congruences.

Les cryptosystèmes RSA et El Gamal reposent sur la théorie des congruences ou *arithmétique modulaire*.

Définition 11.3.7. Soit a, b et m trois entiers. On dit que a est congru à b modulo m et on écrit

$$a \equiv b \pmod{m}$$

si la différence $a - b$ est divisible par m

Une autre manière de dire la même chose

Corollaire 11.3.8. *Soit m un entier non nul, alors a et b sont congrus modulo m si et seulement si les restes de la division euclidienne de a par m et de b par m sont les mêmes*

Démonstration: C'est immédiat. \square

Exemples 19. Prenons $m = 2$ alors deux entiers a et b sont congrus modulo 2 s'il sont soit tous les deux pairs soit tous les deux impairs.

Prenons $m = 5$ alors 5 est congru à 0 modulo 5 (car $5 - 0 = 5$ est divisible par 5) ou à -700 modulo 5 (car $5 - (-700) = 705$ est divisible par 5), 3 est congru à -2 modulo 5 (car $3 - (-2) = 5$ est divisible par 5) ou à 38 modulo 5 (car $3 - 38 = -35$ est divisible par 5)

On montre facilement que

Proposition 11.3.9. *La relation de congruence est une relation d'équivalence sur les entiers.*

c'est à dire que

- $a \equiv a \pmod{m}$ (reflexivité)
- $a \equiv b \pmod{m} \implies b \equiv a \pmod{m}$ (symétrie)
- $(a \equiv b \pmod{m} \text{ et } b \equiv c \pmod{m}) \implies a \equiv c \pmod{m}$ (transitivité)

Démonstration: C'est évident. \square

Proposition 11.3.10. *L'addition et la multiplication sont compatibles à la relation de congruence sur les entiers; autrement dit:*

$$\begin{aligned} \left((a \pmod{m}) + (b \pmod{m}) \pmod{m} \right) &= (a + b \pmod{m}) \\ \left((a \pmod{m}) \times (b \pmod{m}) \pmod{m} \right) &= (a \times b \pmod{m}) \end{aligned}$$

Démonstration: Pour la preuve cf. \square

Exemples 20. Si $m = 2$ alors la proposition signifie simplement que la somme de deux nombres pairs est paire, que la somme de deux nombres impairs est paire, que la somme d'un nombre pair et d'un nombre impair est impaire et que le produit de deux nombres pairs est pair, celui de deux

nombres impairs est impair, celui d'un nombre pair et d'un nombre impair est pair.

Si $m = 5$ on constate que

$$5 + 3 \equiv -700 - 2 \equiv 0 + 38 \equiv 3 \pmod{5}, \quad 5 \times 3 \equiv 0 \times 3 \equiv 5 \times 38 \pmod{5}$$

On fixe m et on décide de ne pas distinguer deux éléments congrus modulo m . L'ensemble des éléments de \mathbb{Z} qui sont congrus modulo m à un entier fixé s'appellera une *classe de congruence modulo m* . Un élément d'une classe de congruence s'appelle un *représentant de la classe de congruence*. Parmi les représentants d'une classe de congruence, il y en a toujours un, unique, compris entre 0 et $m - 1$.

On notera $\mathbb{Z}/m\mathbb{Z}$ l'ensemble des classes de congruence de \mathbb{Z} modulo m . D'après la proposition 11.3.10 on peut munir $\mathbb{Z}/m\mathbb{Z}$ des deux lois $+$ et \times puisque que le résultat modulo m de ces deux opérations ne dépend pas des représentants choisis dans une classe de congruences.

Corollaire 11.3.11. *Tout élément de $\mathbb{Z}/m\mathbb{Z}$ possède un opposé pour l'addition. Les éléments de $\mathbb{Z}/m\mathbb{Z}$ qui ont un inverse multiplicatif sont ceux dont les représentants dans \mathbb{Z} sont premiers à m .*

Démonstration: Facile par Bézout □

Théorème 11.3.12. *Si p est premier $\mathbb{Z}/p\mathbb{Z}$ est un corps fini à p éléments, c'est à dire que tout élément différent de la classe de zéro possède un inverse multiplicatif. On le note \mathbb{F}_p*

Démonstration: facile d'après le corollaire précédent. □

On peut montrer que pour chaque entier $r \geq 1$ il existe un seul corps fini à $q = p^r$ éléments à isomorphisme de corps près, noté $\mathbb{F}_{p^r} = \mathbb{F}_q$. Ils s'obtiennent en considérant les quotients

$$\mathbb{F}_q \simeq \mathbb{F}_p[X]/P(x)\mathbb{F}_p[X], \quad P(X) \in \mathbb{F}_p[X], \text{ irréductible, de degré } r$$

cf. la section 11.6. Le théorème suivant est du à C.F. Gauss

Théorème 11.3.13. *Si p est premier, le groupe multiplicatif*

$$(\mathbb{Z}/p\mathbb{Z})^* = \{\text{éléments inversibles de } (\mathbb{Z}/p\mathbb{Z}, \times)\}$$

est cyclique, i.e. il existe g premier à p tel que

$$\{1, \dots, p-1\} \equiv \{g^n, 0 \leq n \leq p-2\}$$

Un générateur g du groupe cyclique $\mathbb{Z}/p\mathbb{Z}$ s'appelle une racine primitive modulo p .

De même si $q = p^r$ avec p premier le groupe multiplicatif, \mathbb{F}_q^* , des éléments inversibles de \mathbb{F}_q est cyclique.

Démonstration: Pour la preuve cf. [19] □

Le théorème suivant joue un rôle extrêmement important en arithmétique modulaire et en particulier pour le décodage du cryptosystème RSA, ainsi que pour la mise en oeuvre du cryptosystème El Gamal

Théorème 11.3.14 (petit théorème de Fermat). *Soit p un nombre premier. Tout entier a vérifie la congruence $a^p \equiv a \pmod{p}$, et si $a \wedge p = 1$ on a aussi $a^{p-1} \equiv 1 \pmod{p}$.*

Exemples 21. $p = 7$, $a = 3$ alors

$$\begin{aligned} \left(3^6 \pmod{7}\right) &= \left(3^{2+4} \pmod{7}\right) \\ &= \left(3^2 \pmod{7}\right) \left((3^2)^2 \pmod{7}\right) \\ &= \left(2 \times 4 \pmod{7}\right) = \left(1 \pmod{7}\right) \end{aligned}$$

Remarquer la manière de calculer une puissance: écrire l'exposant en base 2 et procéder par élévations au carré successives.

Démonstration: Les éléments inversibles de \mathbb{N} modulo p plus petits que p sont $1, 2, \dots, (p-1)$. Considérons a premier à p donc inversible modulo p et considérons l'ensemble

$$\{a, 2a, 3a, \dots, (p-1)a\} \pmod{p}$$

c'est une permutation de l'ensemble $\{1, 2, \dots, (p-1)\}$ car a est inversible modulo p . Donc:

$$\begin{aligned} a \cdot 2a \cdot 3a \dots (p-1)a &\equiv 1 \cdot 2 \cdot 3 \dots (p-1) \pmod{p} \\ &\Updownarrow \\ (a^{p-1} - 1)((p-1)!) &\equiv 0 \pmod{p} \end{aligned}$$

Comme $(p-1)!$ est premier à p donc inversible modulo p on a

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

Le théorème suivant est appelé: Théorème des restes chinois

Théorème 11.3.15. Soient m_1, m_2, \dots, m_r des entiers naturels deux à deux premiers entre eux (i. e. $m_i \wedge m_j = 1$ si $i \neq j$) et soient a_1, a_2, \dots, a_r des entiers relatifs. Le système de congruences

$$(11.2) \quad \begin{cases} x \equiv a_1 \pmod{m_1}, & x \equiv a_2 \pmod{m_2} \\ \vdots & \vdots \\ \dots & x \equiv a_r \pmod{m_r} \end{cases}$$

a toujours une solution et deux solutions quelconques sont congrues modulo $M = m_1 m_2 \dots m_r$

Exemples 22. Prenons $r = 2$, $m_1 = 15$, $m_2 = 14$, $a_1 = 6$, $a_2 = 9$. On veut donc résoudre simultanément

$$(11.3) \quad x \equiv 6 \pmod{15} \text{ et } x \equiv 9 \pmod{14}$$

La première équation impose que x est de la forme $x = 6 + 15k$ avec $k \in \mathbb{Z}$ et la deuxième impose que x est de la forme $x = 9 + 14\ell$ $\ell \in \mathbb{Z}$ et par conséquent on doit avoir

$$6 + 15k = 9 + 14\ell \iff 15k - 14\ell = 3$$

Comme $14 \wedge 15 = 1$ le théorème de Bézout indique qu'il existe $(u, v) = (1, -1)$ tel que $15u + 14v = 1$ et donc en prenant $k = \ell = 3$ on a une solution $x = 51$ du système de congruences (11.3)

Démonstration: Unicité (mod M) de la solution au système de congruences. Si x_1 et x_2 sont deux solutions alors $x = x_1 - x_2 \equiv 0 \pmod{m_i}$ pour $1 \leq i \leq r$ et donc $x \equiv 0 \pmod{M}$.

Soit $M_i = \frac{M}{m_i}$, clairement $M_i \wedge m_i = 1$ donc il existe un entier N_i tel que $M_i N_i \equiv 1 \pmod{m_i}$ et de plus $m_i \mid M_j$ si $i \neq j$. Alors

$$x = \sum_{i=1}^r a_i M_i N_i \implies x \equiv a_i \pmod{m_i}, \quad 1 \leq i \leq r \quad \square$$

Ce théorème permet de remplacer un système de congruences modulo les m_i par une congruence unique modulo $M = m_1 \cdot \dots \cdot m_r$.

Corollaire 11.3.16. *Avec les notations du théorème 11.3.15, le système de congruences (11.2)*

$$\begin{cases} x \equiv a_1 \pmod{m_1}, & x \equiv a_2 \pmod{m_2} \\ \vdots & \vdots \\ \dots & x \equiv a_r \pmod{m_r} \end{cases}$$

équivalent à la congruence unique

$$y \equiv x \pmod{M}$$

Démonstration: C'est évident. □

Remarque 11.3.2. Ce corollaire exprime que l'on a

$$\prod_{i=1}^r \mathbb{Z}/m_i\mathbb{Z} \simeq \mathbb{Z}/M\mathbb{Z}$$

où l'isomorphisme est un isomorphisme d'anneau.

Définition 11.3.17. *Si m est un entier positif on note $\varphi(m)$ le nombre d'entiers $b < m$ et premiers à m , ou de manière équivalente $\varphi(m)$ est le nombre d'entiers $b < m$ inversibles modulo m*

La fonction φ est appelée l'*indicatrice d'Euler*.

Proposition 11.3.18. *Si $m = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_r^{\alpha_r}$ est la décomposition de m en facteurs premiers distincts alors*

$$(11.4) \quad \varphi(m) = p_1^{\alpha_1-1}(p_1-1)p_2^{\alpha_2-1}(p_2-1)\dots p_r^{\alpha_r-1}(p_r-1)$$

Démonstration: Il est clair que $\varphi(1) = 1$, que si p est premier $\varphi(p) = p-1$ et que $\varphi(p^\alpha) = (p-1)p^{\alpha-1}$. Par le théorème des restes chinois si m est premier à n alors

$$\varphi(mn) = \varphi(m) \times \varphi(n) \quad \square$$

Théorème 11.3.19 (Théorème d'Euler). *Soit $m > 1$ un entier et soit a un entier premier à m , alors on a : $a^{\varphi(m)} \equiv 1 \pmod{m}$*

Démonstration: Même preuve que pour le petit théorème de Fermat □

En particulier si p et q sont premiers,

$$(11.5) \quad \varphi(pq) = (p-1)(q-1)$$

et

$$(11.6) \quad a^{(p-1)(q-1)} \equiv 1 \pmod{pq}$$

Il est facile de montrer à partir du théorème d'Euler le corollaire important suivant qui est un cas particulier d'un théorème de Lagrange, [19],

Corollaire 11.3.20. *Soit $m \in \mathbb{N}$ et soit $a \in \mathbb{N}$ tel que $a \wedge m = 1$ alors le plus petit entier $e \geq 1$ tel que $a^e \equiv 1 \pmod{m}$ est un diviseur de $\varphi(m)$.*

Démonstration: Supposons que e ne soit pas un diviseur de $\varphi(m)$ et écrivons l'identité de la division euclidienne de $\varphi(m)$ par e

$$\varphi(m) = e \times q + r \text{ avec } 0 < r \leq e - 1$$

on a $r \neq 0$ car e ne divise pas $\varphi(m)$. Alors

$$a^r = a^{\varphi(m) - eq} = a^{\varphi(m)} \times a^{-eq} = a^{\varphi(m)} \times (a^e)^{-q} \equiv 1 \pmod{m}$$

Or comme $1 \leq r \leq e - 1$ ceci contredit la définition de e . \square

Résidus quadratiques modulo p .

On va étudier les *résidus quadratiques* modulo un nombre premier p c'est à dire les carrés modulo un nombre premier p . Cette étude joue un rôle important dans les tests de primalité probabilistes ou non.

Définition 11.3.21 (Symbole de Legendre). *On pose si p est premier impair*

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{si } a \text{ est premier à } p \text{ et est un carré } \pmod{p} \\ -1 & \text{si } a \text{ est premier à } p \text{ et n'est pas un carré } \pmod{p} \\ 0 & \text{si } a \text{ n'est pas premier à } p \end{cases}$$

Le symbole $\left(\frac{a}{p}\right)$ est appelé le symbole de Legendre.

C'est un caractère du groupe multiplicatif $(\mathbb{Z}/p\mathbb{Z})^*$ à valeur dans $\{\pm 1\} \cup \{0\}$ de période p autrement dit:

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

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

Théorème 11.3.22 (Euler). *Si p est premier impair et a premier à p alors*

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}, \quad \left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}, \quad \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}$$

Démonstration: Pour la démonstration voir [19]. \square

Théorème 11.3.23 (Loi de réciprocité quadratique). *Soit p et q des premiers impairs alors*

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

Démonstration: Pour la démonstration voir [19]. \square

Ce théorème (un des plus célèbres de l'arithmétique) est du à C. F. Gauss, il se généralise au symbole de Jacobi ci-dessous. Sa preuve est délicate.

Définition 11.3.24 (Symbole de Jacobi). *On pose si $m = \prod_{p_i | m} p_i^{\alpha_i}$ est*

impair et $n \in \mathbb{Z}$

$$\left(\frac{n}{m}\right) = \begin{cases} \prod_{p_i | m} \left(\frac{n}{p_i}\right)^{\alpha_i} & \text{si } m \wedge n = 1 \\ 1 & \text{si } m = 1 \\ 0 & \text{si } m \wedge n \neq 1 \end{cases}$$

Le symbole de Jacobi possède les propriétés suivantes. Si m est un entier impair

$$(11.7) \quad n_1 \equiv n_2 \pmod{m} \implies \left(\frac{n_1}{m}\right) = \left(\frac{n_2}{m}\right)$$

$$(11.8) \quad \left(\frac{2}{m}\right) = \begin{cases} +1 & \text{si } m \equiv \pm 1 \pmod{8} \\ -1 & \text{si } m \equiv \pm 3 \pmod{8} \end{cases}$$

$$(11.9) \quad \left(\frac{n_1 n_2}{m}\right) = \left(\frac{n_1}{m}\right) \left(\frac{n_2}{m}\right)$$

Il vérifie aussi une *loi de réciprocité quadratique*: Si m et n sont entiers impairs:

$$(11.10) \quad \left(\frac{n}{m}\right) = \begin{cases} -\left(\frac{m}{n}\right) & \text{si } m \equiv n \equiv 3 \pmod{4} \\ +\left(\frac{m}{n}\right) & \text{sinon} \end{cases}$$

11.4 Tests de primalité.

Pour tester si n est premier on pourrait essayer de le diviser par tous les entiers $< \sqrt{n}$, c'est la méthode du *crible d'Eratosthenes*. C'est impraticable dès que n est grand car \sqrt{n} n'est pas majoré uniformément par un polynôme en $\log n$.

Le temps de calcul devient vite prohibitif. On a vu en effet qu'une division de n par un entier inférieur demande $O(\log^3 n)$ opérations élémentaires donc cette méthode nécessitera

$$O\left(\sum_{i=1}^{\sqrt{n}} \log^3 n\right) = O(\sqrt{n} \log^3 n) \text{ opérations}$$

Ce n'est donc pas un algorithme polynomial en temps.

Parmi les tests de primalité certains parmi les plus utilisés sont basés sur le petit théorème de Fermat, ou sur les propriétés du symbole de Legendre et sur la théorie des résidus quadratiques.

Il existe, depuis le résultat d'Agrawal-Kayal-Saxena en 2002, [1], des tests de primalité, basés sur le petit théorème de Fermat, qui sont polynomiaux en temps (actuellement en $O(\log^{12}(n))$ et même $O(\log^6(n))$). Mais pour l'instant ils sont moins performants que des tests probabilistes comme ceux décrits ci-dessous.

Pour tester si n est premier on procède souvent en pratique de la manière suivante dans les logiciels de calcul formel.

1. Vérifier que n n'est pas divisible par des petits facteurs premiers.
2. Puis pour des a choisis au hasard et tels que $1 \leq a \leq n$ et $a \wedge n = 1$ calculer $a^{n-1} \pmod n$.
3. Si $a^{n-1} \not\equiv 1 \pmod n$ pour au moins un a alors n n'est pas premier.
4. Sinon on ne peut pas conclure (cf. nombres de Carmichael, par exemple 561). On dit que n est pseudo premier.

Pour avoir un résultat sûr on peut utiliser le *test de Lucas*

Théorème 11.4.1 (Lucas). *Si a est premier à n et si pour tout diviseur premier p de $n - 1$ on a $a^{\frac{n-1}{p}} \not\equiv 1 \pmod n$ alors n est premier*

Démonstration: La preuve de ce résultat est immédiate. □

On peut aussi utiliser les propriétés des symboles de Legendre et de Jacobi.

Pour tester si p est premier on utilise les remarques suivantes

1. On choisit b premier à n avec $1 \leq b \leq n-1$ et on calcule $b^{\frac{p-1}{2}} \pmod{p}$.
2. Si p est premier on doit avoir $b^{\frac{p-1}{2}} \equiv \left(\frac{b}{p}\right) \pmod{p}$ pour tout b (théorème 11.3.22 page 99).
3. Si p n'est pas premier alors on a $b^{\frac{p-1}{2}} \not\equiv \left(\frac{b}{p}\right) \pmod{p}$ pour au moins 50% des b premiers à n .

On aboutit ainsi au *test de primalité de Solovay-Strassen*

1. Tirer un entier aléatoire a , $1 \leq a \leq n-1$
2. Si $\left(\frac{a}{n}\right) \begin{cases} \equiv a^{\frac{n-1}{2}} \pmod{n} & n \text{ est probablement premier} \\ \not\equiv a^{\frac{n-1}{2}} \pmod{n} & n \text{ est décomposable} \end{cases}$
3. Si $\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \pmod{n}$ choisir un autre a et recommencer.

La loi de réciprocité quadratique rend ce test probabiliste très efficace. Elle évite en effet pour calculer le symbole de Jacobi d'avoir à factoriser a et n .

Exemples 23. Testons la primalité de 547. On choisit $a = 5$.

On calcule d'une part

$$\left(\frac{5}{547}\right) = \left(\frac{547}{5}\right) = \left(\frac{2}{5}\right) = (-1)^{\frac{5^2-1}{8}} = -1$$

d'autre part

$$5^{\frac{547-1}{2}} \pmod{547} \equiv 5^{1+2^4+2^8} \pmod{547} \equiv 5 \cdot 113 \cdot 395 \pmod{547} \\ \equiv -1 \pmod{547}$$

Donc 547 a une chance d'être premier. Pour se conforter on recommence avec 7 au lieu de 5, etc.

Actuellement on est capable de prouver la primalité d'un nombre sans symétrie particulière de 2000 à 3000 chiffres en base 10 en un mois de CPU sur une station de travail. On prouve aussi la primalité de nombres tests comme les nombres de Mersenne $(2^p - 1)$ ayant plusieurs millions de chiffres.

Pour construire de grands nombres premiers on couple les résultats précédents avec des théorèmes sur la répartition des nombres premiers.

On sait par exemple que le nombre de nombres premiers inférieurs à x , $\pi(x) \sim \frac{x}{\log x}$. On en déduit qu'il y a toujours un nombre premier entre n et $2n$ (en fait on sait plus). On prend un nombre n au hasard et on teste s'il est premier, s'il ne l'est pas on passe à $n + 1$, etc.

11.5 Méthode de factorisation.

Principe d'un algorithme très utilisé pour factoriser de grand nombres entiers, le *crible quadratique*. Il est fondé sur la proposition suivante

Proposition 11.5.1 (Fermat). *Soit n un entier positif impair. Il y a une bijection entre les décompositions de n sous la forme $n = ab$ avec $a \geq b \geq 0$ entiers et les représentations de n sous la forme $n = t^2 - s^2$, où s et t sont des entiers positifs ou nuls. La bijection est donnée par les équations*

$$t = \frac{a+b}{2}, \quad s = \frac{a-b}{2}, \quad a = t+s, \quad b = t-s$$

Exemples 24. Soit à factoriser $n = 23360947609$

$$n = 23360947609 \implies E(\sqrt{n}) = 152843$$

On pose $t = 152843$. On teste s'il existe i pas trop grand tel que

$$(t+i)^2 - n = (152843+i)^2 - 23360947609$$

est un carré.

On trouve que pour $i = 2$ on a

$$152845^2 - 23360947609 = 804^2$$

et donc $n = pq$ avec $\begin{cases} p = 152845 + 804 = 152649 \\ q = 152845 - 804 = 152041 \end{cases}$. On vérifie aisément que p et q sont premiers.

Cette méthode ne marche pas à coup sûr, on l'a raffinée de diverses manières mais la factorisation d'un nombre reste encore un problème difficile.

Dans la proposition précédente on utilise le fait que l'on a

$$t^2 \equiv s^2 \pmod{n} \text{ et } t \not\equiv \pm s \pmod{n}$$

Ces deux relations impliquent que l'on trouve alors un diviseur non trivial de n en calculant $\text{PGCD}(t + s, n)$ et $\text{PGCD}(t - s, n)$. En effet $n \mid t^2 - s^2$ et $n \nmid t + s, n \nmid t - s$, donc $a = \text{PGCD}(t + s, n)$ est un diviseur non trivial de n et $b = \frac{n}{a}$ divise $a = \text{PGCD}(t - s, n)$.

Exemples 25. On veut factoriser 4633. On remarque que

$$118^2 \equiv 25^2 \pmod{4633},$$

$$\text{PGCD}(118 + 25, 4633) = 41, \quad \text{PGCD}(118 - 25, 4633) = 113$$

On constate que $4633 = 41 \times 113$.

Pour mettre cette remarque en oeuvre on cherche une famille finie de nombres premiers $\mathcal{B} = \{p_1, p_2, \dots, p_h\}$, où $p_1 = -1$ et des entiers b_i tels que le plus petit représentant de $(b_i^2 \pmod n)$ noté $\{b_i^2\}$ (i.e le représentant compris entre $-\frac{n}{2}$ et $+\frac{n}{2} - 1$) s'écrive $\{b_i^2\} = \prod_{j=1}^h p_j^{\varepsilon_{j,i}}$ avec pour tout j , $\sum_i \varepsilon_{j,i} \equiv 0 \pmod 2$.

On pose alors

$$\gamma_j = \frac{1}{2} \sum_i \varepsilon_{j,i}$$

et

$$\{b\} = \prod_i b_i \pmod n, \quad \text{et } c = \prod_{p \in \mathcal{B}} p^{\gamma_j}$$

Si $b \not\equiv \pm c \pmod n$ on a gagné et on a un diviseur de n par la remarque précédente. Sinon il faut recommencer avec un autre choix pour \mathcal{B} ou pour les b_i .

La factorisation des entiers reste un problème difficile. En 1994 il a fallu le travail combiné de 600 personnes pendant 8 mois pour factoriser un nombre de 129 chiffres proposé 17 ans plus tôt et pour cela il fallu développer dans l'intervalle de nouvelles méthodes de factorisation.

En 1999 il a fallu 8 mois de travail pour factoriser un nombre de 155 chiffres. Il a fallu en plus de l'augmentation de puissance des ordinateurs développer et perfectionner de nouvelles méthodes de factorisation.

En 2005 on a factorisé des nombres de 200 chiffres en base 10 en utilisant le crible quadratique et 80 ordinateurs travaillant en parallèle pendant plusieurs mois.

Pour des généralisations des codes RSA ainsi que pour des procédés de factorisation et des tests de primalité, on utilise l'arithmétique sur les courbes elliptiques qui est une sorte de généralisation de l'arithmétique modulaire .

11.6 Rappels d'algèbre.

Ainsi qu'on l'a vu à la section 7.2 page 38 la description d'AES nécessite celle des extensions de corps finis comme quotient d'anneaux de polynômes.

11.6.1 Anneau des polynômes sur un corps

Soit \mathbb{K} un corps, par exemple \mathbb{Q} , \mathbb{R} , \mathbb{C} ou un corps fini comme $\mathbb{Z}/2\mathbb{Z}$, $\mathbb{Z}/p\mathbb{Z}$ avec p premier.

Définition 11.6.1. *L'anneau des polynômes à coefficients dans \mathbb{K} , noté $\mathbb{K}[X]$, est l'ensemble des suites finies d'éléments de \mathbb{K} ,*

$$P = P(X) = (a_0, a_1, \dots, a_n)$$

appelées polynômes. Deux polynômes

$$P(X) = (a_0, a_1, \dots, a_n) \in \mathbb{K}[X], \quad Q(X) = (b_0, b_1, \dots, b_m) \in \mathbb{K}[X]$$

(avec $m \geq n$) sont égaux si $a_i = b_i$ pour $0 \leq i \leq n$ et $b_j = 0$ pour $n+1 \leq j \leq m$, en particulier

$$P(X) = (a_0, a_1, \dots, a_n) = \underbrace{(a_0, a_1, \dots, a_n, 0, 0, \dots, 0)}_{n+h \text{ éléments}}$$

Considérons les polynômes

$$P(X) = (a_0, a_1, \dots, a_n) \in \mathbb{K}[X], \quad Q(X) = (b_0, b_1, \dots, b_m) \in \mathbb{K}[X]$$

On définit la somme $P(X) + Q(X)$ par

$$P(X) + Q(X) = R(X) = (c_0, c_1, \dots, c_n) = ((a_0 + b_0), (a_1 + b_1), \dots, (a_n + b_n))$$

et le produit $P(X) \times Q(X) = P \cdot Q = PQ$ par

$$P(X) \times Q(X) = (d_0, d_1, \dots, d_{m+n}) = ((a_0 b_0), (a_0 b_1 + a_1 b_0), \dots, \underbrace{(a_j b_0 + a_{j-1} b_1 + \dots + a_{j-k} b_k + \dots + a_0 b_j)}_{\text{avec } a_\ell = 0 \text{ si } \ell > n, b_\lambda = 0 \text{ si } m > \lambda} + \dots + (a_n b_m))$$

Notations 11.6.1. On adopte les notations classiques suivantes

- Le polynôme $(0, 0, \dots, 0)$ sera noté 0 et on a $P(X) + 0 = 0 + P(X) = P(X)$ et $0 \times P(X) = P(X) \times 0 = 0$. Le polynôme $(1, 0, 0, \dots, 0)$ sera noté 1 et on a $1 \times P(X) = P(X) \times 1 = P(X)$.

- Le polynôme $P(X) = (0, 1, 0, \dots, 0)$ sera noté X et appelé l'*indéterminée* ou la *variable*.
- Le polynôme $P(X) = \underbrace{(0, 1, 0, \dots, 0) \times \dots \times (0, 1, 0, \dots, 0)}_{n \text{ facteurs}}$ sera noté X^n ,
- Si $P(X) \neq 0$ alors il existe un indice n tel que si

$$P(X) = (a_0, a_1, \dots, a_m)$$

on ait $a_\ell = 0$ pour $\ell \geq n + 1$. L'entier n sera appelé le *degré du polynôme* P et sera noté $\deg(P)$. Le degré du polynôme 0 n'est pas défini, on lui donne parfois par convention le degré $-\infty$.

- Avec ces notations on écrira le polynôme $P(X) = (a_0, a_1, \dots, a_n)$ de degré n sous la forme $P(X) = a_0 + a_1X + a_2X^2 + \dots + a_nX^n$, le terme a_iX^i , $0 \leq i \leq n$ sera appelé le *terme de degré i* de $P(X)$. On remarque que l'on a aussi $P(X) = a_0 + a_1X + \dots + a_nX^n + 0x^{n+1} + \dots + 0x^m$
- Un polynôme non nul, $P(X)$, sera dit *unitaire* si le coefficient de son terme de plus haut degré est 1.

Muni de ces deux opérations $\mathbb{K}[X]$ est un anneau commutatif unitaire, cf. [20], muni d'une division euclidienne définie de la manière suivante

Définition 11.6.2 (Division euclidienne). *Si $A(X)$ et $B(X)$ sont deux polynômes de $\mathbb{K}[X]$ ($B \neq 0$) il existe un unique couple de polynômes $Q(X)$ et $R(X)$ tel que*

$$A(X) = B(X)Q(X) + R(X) \text{ et } \begin{cases} \text{ou bien} & R = 0 \\ \text{ou bien} & 0 \leq \deg(R) \leq \deg(B) - 1 \end{cases}$$

On dit que $B(X)$ est un diviseur de $A(X)$ s'il existe $Q(X) \in \mathbb{K}[X]$ tel que $A(X) = B(X)Q(X)$, on note $B(X) \mid A(X)$.

On dit que deux polynômes non nuls $A(X)$ et $B(X)$ sont associés s'il existe $a \in \mathbb{K}^*$ tel que $A(X) = a \times B(X)$.

La notion de divisibilité fournit une relation d'ordre partielle sur l'anneau des polynômes.

Tout polynôme $A(X) = a_0 + a_1X + \dots + a_nX^n$ a au moins comme diviseurs les éléments non-nuls de \mathbb{K} et lui-même car si $a \neq 0 \in \mathbb{K}$ alors a^{-1} existe et on a

$$A(X) = a \left((a_0a^{-1}) + (a_1a^{-1})X + \dots + (a_na^{-1})X^n \right), \quad A(X) = 1 \times A(X)$$

Mais il peut en avoir d'autres par exemple:

$$X^2 - 1 = (X - 1)(X + 1)$$

$X^2 - 1$ a donc comme diviseurs les constantes non nulles ($=\mathbb{K}^*$), les polynômes $X - 1$ et $X + 1$ à une constante multiplicative non nulle près et lui-même à une constante multiplicative non nulle près.

Un polynôme, $P(X)$, est appelé une *unité de* $\mathbb{K}[X]$ s'il est réduit à un élément de $\mathbb{K}^* = \mathbb{K} \setminus \{0\}$, autrement dit si P^{-1} existe dans $\mathbb{K}[X]$

Définition 11.6.3. *Un polynôme $P(X) \in \mathbb{K}[X]$ est appelé un polynôme premier ou si ses seuls diviseurs sont lui-même et les éléments de \mathbb{K}^**

Exemples 26. Dans $\mathbb{Q}[X]$ le polynôme $X^2 + 1$ est irréductible, par contre dans $\mathbb{F}_2[X]$ il ne l'est pas car sur \mathbb{F}_2 on a $X^2 + 1 = (X + 1)^2$.

Proposition 11.6.4. *Tout polynôme non nul $A(X) \in \mathbb{K}[X]$ possède une décomposition en un produit de polynômes premiers*

$$A(X) = a \cdot P_1(X)P_2(X) \dots P_m(X)$$

où $a \in \mathbb{K}^*$, $P_j(X)$ premier pour $1 \leq j \leq m$

Cette décomposition est unique à l'unité a près et à l'ordre près des facteurs P_j . Si l'on regroupe tous les polynômes P_j associés la décomposition en produit de polynômes premiers prend la forme suivante

$$A(X) = a \cdot P_1(X)^{\alpha_1} P_2(X)^{\alpha_2} \dots P_\ell(X)^{\alpha_\ell}, \quad \alpha_i \in \mathbb{N},$$

où $a \in \mathbb{K}^*$, $P_j(X)$ premier pour $1 \leq j \leq \ell$,
 P_i et P_j ne sont pas associés pour $i \neq j$

sous cette forme la décomposition est unique à l'unité a près et à l'ordre des facteurs si on choisit les P_j unitaires et les $\alpha_i \geq 1$ (c'est à dire qu'on s'interdit des exposant nuls).

Démonstration: La preuve repose sur l'algorithme de division euclidienne, cf. [20]. □

On définit alors comme dans le cas de \mathbb{Z} la notion de Plus Grand Commun Diviseur (PGCD) de deux polynômes. Les degrés des diviseurs unitaires communs à A et B forment un ensemble majoré par le maximum des degrés de A et de B .

Définition 11.6.5. *Le PGCD de deux polynômes $A(X)$ et $B(X)$ de $\mathbb{K}[X]$ est le polynôme unitaire de plus grand degré $D(X)$ qui divise à la fois $A(X)$ et $B(X)$. On note $\text{PGCD}(A, B) = A \wedge B$*

Proposition 11.6.6. *Soient $A(X)$ et $B(X)$ des polynômes non nuls. On peut toujours supposer, quitte à modifier les unités a et b et à utiliser des exposants, que leurs décomposition en produit de facteurs premiers unitaires est*

$$\begin{aligned} A(X) &= a \times P_1(X)^{\alpha_1} P_2(X)^{\alpha_2} \dots P_r(X)^{\alpha_r}, \\ B(X) &= b \times P_1(X)^{\beta_1} P_2(X)^{\beta_2} \dots P_r(X)^{\beta_r}, \text{ avec } \alpha_i, \beta_i \geq 0 \end{aligned}$$

alors le PGCD de A et de B est:

$$A(X) \wedge B(X) = P_1(X)^{\inf\{\alpha_1, \beta_1\}} P_2(X)^{\inf\{\alpha_2, \beta_2\}} \dots P_r(X)^{\inf\{\alpha_r, \beta_r\}}$$

Démonstration: Pour la démonstration voir [20]. □

Ni la définition ni la proposition 11.6.6 ne fournissent d'algorithmes très efficaces pour calculer le PGCD de deux polynômes, $A(X)$ et $B(X)$.

Par contre l'algorithme de la division euclidienne en fournit un très efficace.

Proposition 11.6.7. *Le PGCD des polynômes A et B est donné par l'algorithme suivant. On écrit les divisions euclidiennes successives*

$$(11.11) \quad \begin{cases} A(X) = B(X)Q(X)_0 + R_1(X) \\ B(X) = Q_1(X)R_1(X) + R_2(X) \\ R_1(X) = Q_2(X)R_2(X) + R_3(X) \\ \vdots \\ R_{j-2}(X) = Q_{j-1}(X)R_{j-1}(X) + R_j(X) \\ R_{j-1}(X) = Q_j(X)R_j(X) + R_{j+1}(X) \\ R_j(X) = q_{j+1}(X)r_{j+1}(X) \end{cases}$$

avec $(0 \leq \deg(R_1) < \deg(B)$ ou $R_1 = 0)$, $(0 \leq \deg(R_2) < \deg(R_1)$ ou $R_2 = 0)$, $0 \leq \deg(R_3) < \deg(R_2), \dots, 0 \leq \deg(R_j) < \deg(R_{j+1})$.

On arrête l'algorithme dès qu'un reste est nul et alors

$$A(X) \wedge B(X) = R_{j+1}(X)$$

Démonstration: La dernière identité de division euclidienne de (11.11) montre que R_{j+1} divise R_j , l'avant dernière identité de division euclidienne de (11.11) montre que R_{j+1} divise R_j et R_{j-1} . Puis par récurrence on montre que R_{j+1} divise A et B .

Donc R_{j+1} est un diviseur commun de A et de B , est-ce le plus grand? Considérons un diviseur commun de A et de B noté D . Par définition D divise A et B donc d'après la première des identités de division euclidienne de (11.11) on a $D \mid R_1$, alors la deuxième des identités de division euclidienne de (11.11) implique que $D \mid R_2$ puis par récurrence on montre que $D \mid R_{j+1}$.

On a donc montré que tout diviseur D commun de A et de B est un diviseur de R_{j+1} et comme R_{j+1} est un diviseur de A et de B c'est le plus grand, autrement dit $R_{j+1} = A \wedge B$. \square

L'arithmétique de $\mathbb{K}[X]$ est donc tout à fait parallèle à celle de \mathbb{Z} .

- Les deux sont des anneaux munis d'une division euclidienne.
- On définit sur $\mathbb{K}[X]$ une notion de factorisation en facteurs premiers,
- On peut définir le PGCD de deux polynômes,
- Il y a sur $\mathbb{K}[X]$ une relation de Bezout, un algorithme d'Euclide étendu pour calculer le PGCD.
- etc..

On a le théorème de Bézout comme dans le cas des entiers

Théorème 11.6.8 (Théorème de Bézout). *Soient $A(X), B(X) \in \mathbb{K}[X]$, alors il existe des polynômes $U(X), V(X) \in \mathbb{K}[X]$ tels que*

$$(11.12) \quad A(X)U(X) + B(X)V(X) = A(X) \wedge B(X) = D(X)$$

Démonstration: La preuve est la même que sur \mathbb{Z} . On reprend l'algorithme du PGCD (11.11) à partir de la fin. On écrit avec les notations précédentes

$$\begin{aligned} D &= R_{j+1} = R_{j-1} - Q_j R_j = R_{j-1} U_{j-1}(1) - R_j V_j(Q_j) \\ &= R_{j-1} - Q_j (R_{j-2} - Q_{j-1} R_{j-1}) \\ &= R_{j-1} (1 + Q_j Q_{j-1}) - Q_j R_{j-2} \\ &= -U_{j-2}(Q_j) R_{j-2} + R_{j-1} V_{j-1}(Q_{j-1}, Q_j) \\ &= (R_{j-3} - Q_{j-2} R_{j-2}) (1 + Q_j Q_{j-1}) - Q_j R_{j-2} \end{aligned}$$

$$\begin{aligned}
&= R_{j-3}(1 + Q_j Q_{j-1}) - R_{j-2}(Q_{j-2}(1 + Q_j Q_{j-1}) + Q_j) \\
d &= R_{j-3}U_{j-3}(Q_j, Q_{j-1}) - R_{j-2}V_{j-2}(Q_{j-2}, Q_{j-1}, Q_j) \\
&\quad \vdots \\
&= (-1)^{j+1}aU_0(Q_1, \dots, Q_j) + (-1)^j bV_0(Q_0, \dots, Q_j)
\end{aligned}$$

On a une relation de récurrence facile sur U_j et V_j .

$$\begin{aligned}
U_{j+1} &= U_{j-1} - Q_{j+1}U_j \\
V_{j+1} &= V_{j-1} - Q_{j+1}V_j \quad \square
\end{aligned}$$

On définit alors les congruences entre polynômes comme dans le cas des entiers

Définition 11.6.9. Si $P(X) \in \mathbb{K}[X]$ et si $A(X), B(X)$ appartiennent à $\mathbb{K}[X]$ on dit que A et B sont congrus modulo P s'il existe $Q \in \mathbb{K}[X]$ tel que $A - B = PQ$. On écrira $A \equiv B \pmod{P}$.

On vérifie aisément que c'est une relation d'équivalence sur l'anneau des polynômes qui est compatible aux opérations sur les polynômes.

Proposition 11.6.10. L'addition et la multiplication sont compatibles à la relation de congruence sur les polynômes; autrement dit si $P(X)$ est un polynôme fixé et si $A(X), B(X) \in \mathbb{K}[X]$ alors

$$\begin{aligned}
&\left((A(X) \pmod{P(X)}) + (B(X) \pmod{P(X)}) \pmod{P(X)} \right) = \\
&\quad (A(X) + B(X) \pmod{P(X)}) \\
&\left((A(X) \pmod{P(X)}) \times (B(X) \pmod{P(X)}) \pmod{P(X)} \right) = \\
&\quad = (A(X) \times B(X) \pmod{P(X)})
\end{aligned}$$

Démonstration: Pour la preuve cf. [19] □

Proposition 11.6.11. L'anneau des polynômes modulo la relation d'équivalence noté

$$\mathbb{K}[X]/P(X)\mathbb{K}[X] \text{ ou } \mathbb{K}[X]/P(X)$$

est un anneau commutatif unitaire et c'est un corps si P est irréductible

Démonstration: Pour la première partie cf. [20]. Si P est irréductible alors par définition si $A(X) \not\equiv 0 \pmod{P(X)}$ on a $A(X) \wedge B(X) = 1$. Donc d'après le théorème de Bezout, il existe $U(X), V(X) \in \mathbb{K}[X]$ tels que

$$A(X)U(X) + P(X)V(X) = 1$$

et donc

$$A(X)U(X) \equiv 1 \pmod{P(X)}$$

Donc si $P(X)$ est irréductible tout élément non nul de $\mathbb{K}[X]/P(X)$ possède un inverse multiplicatif, autrement dit c'est un corps. \square

Proposition 11.6.12. *Un système complet de représentants de*

$$\mathbb{K}[X]/P(X)\mathbb{K}[X]$$

est l'ensemble des restes de la division euclidienne des polynômes de $\mathbb{K}[X]$ par $P(X)$.

Démonstration: C'est immédiat. \square

Si \mathbb{K} est un corps fini, \mathbb{F}_p , et si P est un polynôme irréductible sur \mathbb{F}_p alors $\mathbb{F}_p[X]/P(X)$ est un corps fini qui contient \mathbb{F}_p . Sur tout corps fini il y a des polynômes irréductibles de degré n pour tout n , [20].

Exemples 27. Sur \mathbb{F}_2 le polynôme $P = X^2 + X + 1$ est irréductible car il n'est divisible par aucun polynôme de degré 1. En effet les seuls polynômes de degré 1 de $\mathbb{F}_2[X]$ sont X et $X + 1 = X - 1$ et il est clair que dans $\mathbb{F}_2[X]$ les équations

$$X^2 + X + 1 = X(X + b), \quad X^2 + X + 1 = (X + 1)(X + b)$$

n'ont pas de solution.

Donc $\mathbb{F}_2[X]/X^2 + X + 1$ est un corps noté \mathbb{F}_2^2 dont on va donner les éléments

$$\mathbb{F}_2^2 = \mathbb{F}_2[X]/(X^2 + X + 1)\mathbb{F}_2[X] = \{0, 1, X, X + 1\}$$

Rappelons que dans \mathbb{F}_2 on a $0 + 1 = 1 + 1 = 0$, $0 + 1 = 1 + 0 = 1$, $0 \times 0 = 0 \times 1 = 0 \times 0 = 0$ et $1 \times 1 = 1$ et donc que dans \mathbb{F}_2 , $1 = -1$ et $1 + 1 = 2 = 0$.

L'addition sur \mathbb{F}_2^2 est donnée par la table suivante

+	0	1	X	$X + 1$
0	0	1	X	$X + 1$
1	1	0	$X + 1$	X
X	X	$X + 1$	0	1
$X + 1$	$X + 1$	X	1	0

Cette table est évidente.

La multiplication sur \mathbb{F}_2^2 est donnée par la table suivante

\times	0	1	X	$X + 1$
0	0	0	0	0
1	0	1	X	$X + 1$
X	0	X	$X + 1$	1
$X + 1$	0	$X + 1$	1	X

Les seules choses à montrer sont

$$\begin{aligned} X \times X &= X^2 \equiv X + 1 \pmod{X^2 + X + 1} \\ X \times X + 1 &= X^2 + X \equiv 1 \pmod{X^2 + X + 1} \\ (X + 1) \times (X + 1) &= (X + 1)^2 \equiv X + 1 \pmod{X^2 + X + 1} \end{aligned}$$

ce qui est vrai car

$$\begin{aligned} X^2 - (X + 1) &= X^2 + X + 1 \equiv 0 \pmod{X^2 + X + 1} \\ X^2 + X - 1 &= X^2 + X + 1 \equiv 0 \pmod{X^2 + X + 1} \\ (X + 1)^2 - (X + 1) &= X^2 + X + 1 \equiv 0 \pmod{X^2 + X + 1} \end{aligned}$$

Donc dans \mathbb{F}_2^2 l'inverse multiplicatif de $X + 1$ est X .

Le théorème des restes chinois se généralise sans difficulté à $\mathbb{K}[X]$ avec la même preuve.

Corollaire 11.6.13. *Si p est un nombre premier et si $P(x)$ est un polynôme irréductible de degré f de $\mathbb{F}_p[X]$ alors $\mathbb{F}_p[X]/P(X)\mathbb{F}_p[X]$ est le corps fini \mathbb{F}_q à $q = p^f$ éléments. En particulier si $p = 2$ et $P(X) = X^8 + X^4 + X^3 + X + 1$ alors $\mathbb{F}_2[X]/P(X)\mathbb{F}_2[X]$ est le corps fini à $2^8 = 256$ éléments.*

Démonstration: Il suffit de remarquer que $\mathbb{F}_p[X]/P(X)\mathbb{F}_p[X]$ est un \mathbb{F}_p espace vectoriel dont une base est constituée des classes de $1, X, \dots, X^i, \dots, X^{f-1}$, et donc le cardinal de $\mathbb{F}_p[X]/P(X)\mathbb{F}_p[X]$ est $p^f = q$. \square

On a vu que \mathbb{F}_q^* est un groupe multiplicatif cyclique à $q - 1$ éléments. On peut donc considérer des cryptosystèmes El Gamal utilisant le groupe \mathbb{F}_q^* , par exemple

Exemples 28. Considérons $p = 3$, on montre facilement que le polynôme $P(X) = X^3 - X + 1$ est un polynôme irréductible de $\mathbb{F}_3[X]$. En effet sinon on aurait nécessairement une décomposition

$$X^3 - X + 1 = (X + a)(X^2 + bX + c), \text{ avec } a, b, c \in \mathbb{F}_3 \simeq \mathbb{Z}/3\mathbb{Z}$$

avec un facteur de degré 1 au moins. Autrement dit le polynôme $P(X)$ aurait au moins une racine dans \mathbb{F}_3 , or on vérifie aisément avec le petit théorème de Fermat que

$$P(0) = P(1) = P(2) = 1$$

Donc $\mathbb{F}_3[X]/(X^3 + X + 1)\mathbb{F}_3[X]$ n'est autre que le corps \mathbb{F}_{3^3} à 27 éléments.

On a vu que son groupe multiplicatif $\mathbb{F}_{3^3}^*$ est cyclique. Trouvons un générateur g sous forme polynomiale. Montrons que que la classe de X modulo $X^3 - X + 1$ engendre $\mathbb{F}_{3^3}^*$.

i	0	1	2	3
X^i	1	X	X^2	$X + 2$
i	4	5	6	7
X^i	$X^2 + 2X$	$2X^2 + X + 2$	$X^2 + X + 1$	$X^2 + 2X + 2$
i	8	9	10	11
X^i	$2X^2 + 2$	$X + 1$	$X^2 + X$	$X^2 + X + 2$
i	12	13	14	15
X^i	$X^2 + 2$	2	$2X$	$2X^2$
i	16	17	18	19
X^i	$2X + 1$	$2X^2 + X$	$X^2 + 2X + 1$	$2X^2 + 2X + 2$
i	20	21	22	23
X^i	$2X^2 + X + 1$	$X^2 + 1$	$2X + 2$	$2X^2 + X$
i	24	25	26	
X^i	$2X^2 + 2X + 2$	$2X^2 + 2$	1	

Il suffit pour cela de donner le tableau (ci-dessus) des puissances successives de $X \pmod{X^3 + X + 1}$. On prend comme système de représentant de \mathbb{F}_3 , $\{0, 1, 2\}$ (en fait d'après le théorème de Lagrange il suffit de vérifier que $X^{13} \neq 1 \pmod{X^3 - X + 1}$).

11.7 Courbes elliptiques.

Les courbes elliptiques définies sur un corps \mathbb{K} sont des courbes décrites par l'ensemble des solutions de certaines équations polynomiales à deux inconnues $f(x, y) \in \mathbb{K}[X, Y]$. Pour plus de précisions on pourra consulter [23] ou [24].

Définition 11.7.1. Soit \mathbb{R} le corps des réels et soit $a, b \in \mathbb{R}$ tels que $4a^3 + 27q^2 \neq 0$. Une courbe elliptique non singulière définie sur \mathbb{R} est l'ensemble E des points P de \mathbb{R}^2 dont les coordonnées (x, y) sont des solutions de l'équation

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

plus un point spécial \mathcal{O} appelé le point à l'infini.

La condition $4a^3 + 27q^2 \neq 0$ assure que la courbe est sans point double. Si $4a^3 + 27q^2 = 0$ on a affaire à une courbe singulière.

Exemples 29. La courbe $y^2 = x^3 - 4x$

Si E est une courbe elliptique non-singulière on définit sur les points de E une addition notée $+$ qui fera de l'ensemble des points de E un groupe abélien.

- $\forall P \in E$, par définition $P + \mathcal{O} = \mathcal{O} + P = P$.
- Si $P, Q \in E$ avec $P = (x_1, y_1)$, $Q = (x_2, y_2)$ on considère les 3 cas
 1. $x_1 \neq x_2$
 2. $x_1 = x_2$ et $y_1 = -y_2$
 3. $x_1 = x_2$ et $y_1 = y_2$
 - Dans le cas 1, on note L la droite passant par P et Q . Elle coupe E en 3 points dont deux, P et Q , sont déjà connus, on note $R' = (x', y')$ le troisième point d'intersection. On prend le symétrique de R' par rapport à l'axe des abscisses. On obtient

un point $R = (x', -y') = (x, y)$ qui est encore sur E . On pose par définition

$$P + Q = R$$

Par un calcul sans mystère on trouve l'équation de L

$$y = \lambda x + \nu = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)x + \left(y_1 - \frac{y_2 - y_1}{x_2 - x_1}x_1\right)$$

on obtient alors les coordonnées de R

$$x = \lambda^2 - x_1 - x_2 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$$

$$y = \lambda(x_1 - x_2) - y_1 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_2) - y_1$$

– Dans le cas 2, on définit si $P = (x, y) \in E$, $-P = (x, -y)$

$$(x, y) + (x, -y) = \mathcal{O}$$

– Le cas 3 se ramène au cas 2 en considérant que la droite L est la tangente en P à E . On trouve que l'équation de L est

$$y = \lambda x + \nu = \left(\frac{3x_1^2 + a}{2y_1}\right)x + \left(y_1 - \left(\frac{3x_1^2 + a}{2y_1}\right) \cdot x_1\right)$$

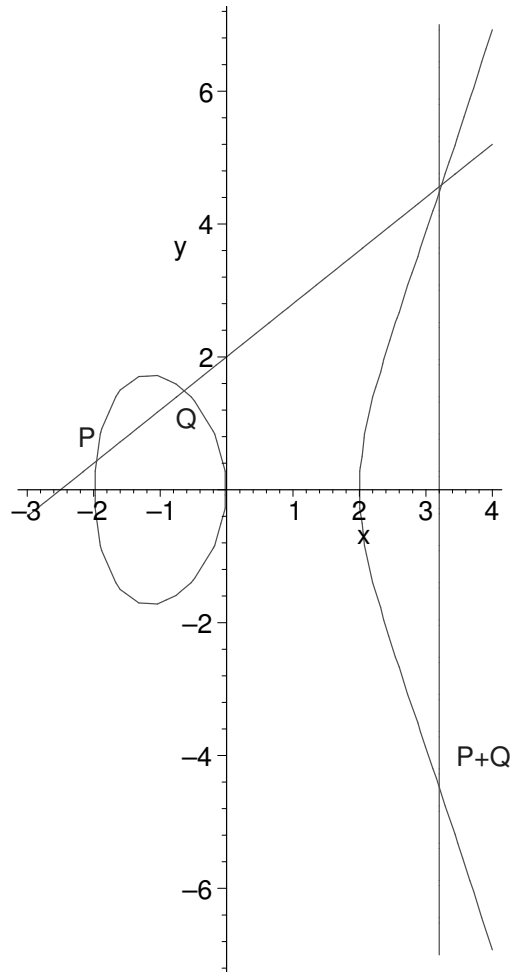
ensuite le calcul de R est identique.

On montre que la loi ainsi définie est

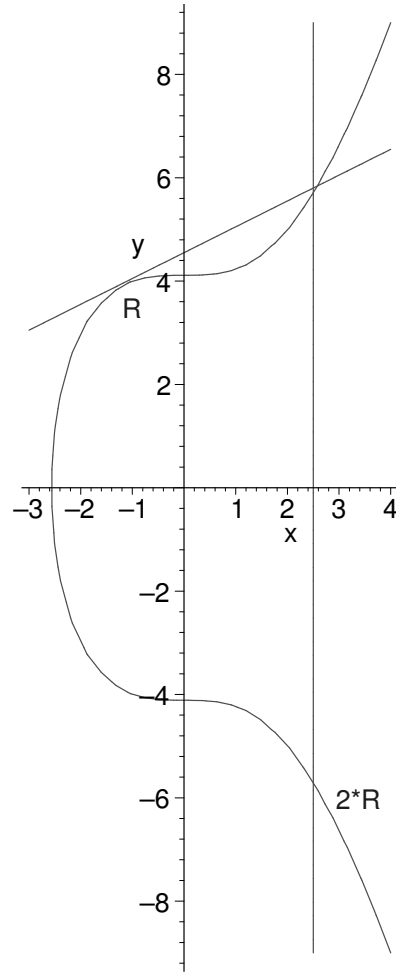
1. stable sur E : si P et Q appartiennent à E , $P + Q \in E$
2. associative: P , Q et R appartiennent à E alors $(P + Q) + R = P + (Q + R)$
3. commutative: $P + Q = Q + P$
4. possède un élément neutre, \mathcal{O} : $P + \mathcal{O} = \mathcal{O} + P = P$
5. Chaque point de E admet un opposé pour cette addition: $\forall P \in E \exists Q \in E$ tel que $P + Q = \mathcal{O}$

Cette loi fait donc des points de E définis sur \mathbb{K} un groupe abélien.

Tout ceci est résumé par les figures suivantes:



courbe elliptique sur \mathbb{R} ,
 $y^2 = x^3 - 4x$



courbe elliptique sur \mathbb{R}
 $y^2 = x^3 + 17$

11.7.1 Courbes Elliptiques sur un corps fini

Le fait que le corps de base soit \mathbb{R} ne joue pas grand rôle dans les calculs précédents (sauf pour pouvoir dessiner les courbes).

Si maintenant on a un corps fini \mathbb{F}_q où $q = p^f$ avec p un nombre premier on peut refaire la théorie en supposant que l'on cherche des solutions dans \mathbb{F}_q^2 (on peut prendre $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$).

Définition 11.7.2. Soit \mathbb{F}_q un corps fini avec $p \geq 5$ et soit $a, b \in \mathbb{F}_q$ tels que $4a^3 + 27b^2 \neq 0$. Une courbe elliptique non singulière définie sur \mathbb{F}_q est l'ensemble E des points P de \mathbb{F}_q^2 dont les coordonnées (x, y) sont des solutions de l'équation

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

plus un point spécial \mathcal{O} appelé point à l'infini.

Exemples 30. Calculons par exemple les points de la courbe elliptique $y^2 = x^3 + x + 6$ dans \mathbb{F}_{11}

x	$x^3 + x + 6 \pmod{11}$	résidu quadratique?	y
0	6	non	
1	8	non	
2	5	oui	4,7
3	3	oui	5,6
4	8	non	
5	4	oui	2,9
6	8	non	
7	4	oui	2,9
8	9	oui	3,8
9	7	non	
10	4	oui	2,9

Cette courbe sur \mathbb{F}_{11} admet 13 points y compris celui à l'infini.

On a le théorème important

Théorème 11.7.3 (Hasse). Soit p un nombre premier supérieur à 3 et soit $q = p^f$. Soit E une courbe elliptique définie sur le corps fini \mathbb{F}_q . On note $\#E$ le nombre de points de E définis sur \mathbb{F}_q . On a

$$q + 1 - 2\sqrt{q} \leq \#E \leq q + 1 + 2\sqrt{q}$$

Le calcul de $\#E$ est difficile mais il existe un algorithme performant pour le faire l'*algorithme de Schoof*.

Pour pouvoir utiliser le groupe des points d'une courbe elliptique sur un corps fini, E , il faut ensuite trouver un sous-groupe cyclique aussi gros que possible afin de pouvoir transposer le schéma de codage El-Gamal. Il y a des algorithmes efficaces pour cela.

Avec des sous-groupes cycliques de E à 2^{160} éléments on a une très bonne sécurité si l'on prend quelques précautions pour éliminer des courbes elliptiques indésirables pour lesquelles le problème du logarithme discret est facile.

L'avantage des cryptosystèmes basés sur les courbes elliptiques est, entre autre, la possibilité d'avoir des clés courtes pour une bonne sécurité.

Sites internet

- cryp1** <http://cryptosec.lautre.net/index.php3>;
donne un résumé et des références sur divers aspects de la cryptographie
- hist1** <http://histoirecrypto.ifrance.com/histoirecrypto/> ;
site historique
- prim1** <http://www.utm.edu/research/primes/> ;
donne des renseignements sur les nombres premiers (records, tests de primalité,...)
- cryp2** <http://www.securite.org/db/crypto/> ;
donne un résumé et des références sur divers aspects de la cryptographie
- aes1** <http://www.securiteinfo.com/crypto/aes.shtml>;
donne une description succincte d'AES et de sa comparaison avec triple DES
- aes2** <http://www.cryptageaes.com/>;
présente des produits commerciaux utilisant AES
- aes3** <http://www.bibmath.net/crypto/moderne/aes.php3>;
donne une description succincte d'AES

Bibliographie

- [1] Manindra AGRAWAL, Neeraj KAYAL & Nitin SAXENA, PRIMES is in P, *Annals of Mathematics* **160**, n°2, (2004), 781-793.
- [2] François ARNAULT, *Théorie des nombres et Cryptographie*, cours DEA Université de Limoges, **2000**.
- [3] Christophe BIDAN, *Cryptographie et Cryptanalyse*, Cours, <http://www.supelec-rennes.fr/ren/perso/cbidan/cours/crypto.pdf>
- [4] I.F BLAKE, G. SEROUSSI, N. P. SMART, *Elliptic curves in cryptography*, London Mathematical Society Lecture Note Series, **265** Cambridge University Press **1999**.
- [5] Johannes A. BUCHMANN, *Introduction to cryptography*, Springer, Undergraduate Texts in Mathematics, **2000**.
- [6] Joan DAEMEN & Vincent RIJMEN, *The design of Rijndael* Springer Verlag, Berlin Heidelberg New-York, **2002**.
- [7] W. DIFFIE, M.E. HELMAN, New Directions in Cryptography, IEEE Transactions on Information Theory, 22 , 644-654, **1976**.
- [8] Gilles DUBERTET, *Initiation à la Cryptographie*, éditions Vuibert.
- [9] Robert HARRIS, *Enigma*, éditions Pocket.
- [10] K. IRELAND, M. ROSEN, *A Classical Introduction to Modern Number Theory*, Springer-Verlag, GTM, **1990**.
- [11] Neal KOBLITZ, *A Course in Number Theory and Cryptography*, 2nd edition, Springer, Graduate Texts in Mathematics n°114, , **1994**.
- [12] Neal KOBLITZ, *Introduction to Elliptic Curves and Modular Forms*, Springer-Verlag, GTM 97, **1987**.

- [13] A. J. MENEZES, P. C. van OORSCHOT, and S. A. VANSTONE, *Handbook of Applied cryptography*, Discrete Mathematics and its applications, CRC Press, **1997**.
- [14] Jean-Louis PONS, *Introduction à la Cryptographie*, cours ENSAM Aix en Provence, **2003**.
- [15] La Recherche
- [16] Pour la Science
- [17] Guy ROBIN, *Algorithmique et cryptographie*, éditions Ellipses.
- [18] Claude E. SHANNON, Communication Theory of secrecy systems, *Bell Systems Technical Journal*, **28** (1949), 656-715.
- [19] Bruce SCHNEIER, *Cryptographie Appliquée*, éditions Thomson Publishing.
- [20] Lionel SCHWARTZ, *Mathématiques pour la Licence, Algèbre*, Dunod, Paris, **1998**.
- [21] Simon SINGH, *Histoire des codes secret*, JC Lattès, **1999**.
- [22] Jacques STERN *La Science du Secret*, Odile Jacob, Paris, **1998**.
- [23] Douglas STINSON, *Cryptographie, théorie et pratique*, éditions Vuibert 2e édition, **2003**.
- [24] Lawrence C. WASHINGTON, *Elliptic Curves, Number Theory and Cryptography*, Discrete Mathematics and its applications, Chapman & Hall/CRC, **2003**.
- [25] Benne de Weger, *Cryptographic Systems*, cours Technische Universiteit Eindhoven **2005**; <http://www.win.tue.nl/bdeweger/>
- [26] V. V. YASCHENKO, *Cryptography: An Introduction*, Moscow Center for Continuous Mathematics Education, Editor AMS, **2002**.
- [27] Gilles ZÉMOR, *Cours de cryptographie*, éditions Cassini, **2002**.

Index

(a, b) ,	p. 88	clé secrète,	p. 11
N_r ,	p. 40	clé secrète	
\mathcal{C} ,	p. 24	de déchiffrement,	p. 48
\mathcal{K} ,	p. 24	classe de congruence	
\mathcal{P} ,	p. 24	modulo m ,	p. 94
$a \wedge b$,	p. 88	clef d'étage,	p. 34
Advanced Encrytion		clefs parasites,	p. 80
Standard,	p. 34	coût en espace,	p. 83
AES,	p. 34	coût en temps,	p. 83
algorithme de codage,	p. 24	code d'authentification,	p. 63
algorithme de		code de Vigènère,	p. 14
diversification de clef, ..	p. 40	codes à masques jetables,	p. 18
analyse de fréquence,	p. 13	Codes à répertoire,	p. 9
arbitre,	p. 68	codes de César,	p. 12
arithmétique modulaire,	p. 92	codes de substitution,	p. 12
asynchrone,	p. 17	codes de Vernam,	p. 18
attaque à texte		codes monoalphabétiques, ...	p. 14
chiffré choisi,	p. 22	collisions faibles difficiles,	p. 65
attaque à texte		collisions fortes difficiles,	p. 65
chiffré connu,	p. 22	complexité calculatoire,	p. 74
attaque à texte		complexité polynomiale,	p. 83
clair choisi,	p. 22	complexité	
attaque à texte		non polynomiale,	p. 83
clair connu,	p. 22	confidentialité	
attaques des anniversaires, ...	p. 65	parfaite, .. p. 23, .. p. 79	
authentification, 8		confusion et diffusion,	p. 34
authentique,	p. 7	congrus modulo P ,	p. 109
certificat,	p. 60	courbe elliptique	
clé publique		non singulière,	p. 113
de chiffrement,	p. 48	courbe singulière,	p. 113
		crible d'Eratosthenes,	p. 100
		crible quadratique,	p. 102

- cryptanalyse, p. 6
 cryptographie, p. 6
 cryptologie, p. 6
 cryptosystème RSA, p. 27
 Data Encryption Standard, ... p. 34
 degré du polynôme, p. 105
 DES, p. 34
 digital right management, ... p. 31
 diviseur, p. 87, p. 105
 diviseurs triviaux, p. 87
 division euclidienne, p. 86
 DRM, p. 31
 empreinte numérique, p. 63
 engagement, p. 73
 entropie, p. 80
 espace des clefs, p. 24
 exposant de
 la clé publique, p. 48
 exposant de
 la clé secrète, p. 48
 factorisation des entiers, p. 27
 fonction à sens unique, p. 27
 fonction d'étage, p. 34
 fonction de combinaison, p. 20
 fonction de compression, p. 64
 fonction de décodage, p. 24
 fonction de filtrage, p. 20
 fonction de hachage, p. 63
 fonction de hachage
 à sens unique, p. 65
 fonction de hachage
 itérée, p. 64
 gestion des
 droits numériques, ... p. 31
 hash function, p. 63
 indéterminée, p. 105
 indicatrice d'Euler, p. 97
 infrastructure des
 systèmes à clé secrète, p. 44
 infrastructures des
 systèmes à clé publique, ... p. 60
 Intégrité des données, p. 8
 Kerberos, p. 45
 les mots codés, p. 24
 les mots en clair, p. 24
 LFBR, p. 18
 linear feedback register, p. 18
 logarithme discret, p. 55
 méthode de la grille, p. 10
 machines de Turing, p. 81
 mascarade, p. 7
 modèle X.509, p. 61
 module du cryptosystème, ... p. 48
 National Bureau of
 Standards, p. 34
 nombres premiers, p. 87
 non-répudiation, p. 8
 non-répudiation
 de transmission, ... p. 8
 non-répudiation
 d'origine, p. 8
 non-répudiation
 de réception, p. 8
 NP, p. 83
 opérations élémentaires, p. 84
 P, p. 82
 par blocs, p. 14
 permutation initiale, p. 35
 PGCD, p. 88
 PGP, p. 26
 PKI, p. 60

- plus grand commun
 diviseur, p. 88
- point à l'infini, p. 113
- polyalphabétique, p. 14
- polynôme irréductible, 106, .. p. 106
- polynôme premier, p. 106
- polynôme unitaire, p. 105
- polynômes associés, p. 105
- porte dérobée, p. 47
- preuve sans apport
 de connaissance, ... p. 32
- procédé de chiffrement, p. 24
- probabilité mutuelle, p. 78
- probabilité conditionnelle, ... p. 78
- problèmes non polynomiaux
 en temps, p. 83
- problèmes polynomiaux
 en temps, p. 82
- produit de polynômes
 premiers, p. 106
- protocole interactif, p. 75
- protocoles d'identification, ... p. 32
- Public Key
 Infrastructure, p. 60
- réciprocité quadratique, p. 99
- résidus quadratiques, p. 98
- règle de chiffrement, p. 24
- règle de déchiffrement, p. 24
- racine primitive modulo p , ... p. 95
- redondance, p. 80
- registres à décalages, p. 18
- représentant de la classe
 de congruence, p. 94
- Rivest-Shamir et Adleman, .. p. 27
- S-boîtes, p. 36
- sécurité calculatoire, p. 23
- sécurité inconditionnelle, p. 23
- sécurité prouvée, p. 23
- schéma de Feistel, p. 34
- scytale, p. 10
- Secure Hash Algorithm, p. 66
- SHA-1, 66
- signature, p. 8
- sous-exponentiel, p. 29
- suites pseudo-aléatoires, p. 19
- suites récurrentes
 linéaires sur un corps fini, . p. 18
- symbole de Jacobi, p. 99
- symbole de Legendre, p. 98
- Symmetric Keys
 Management, p. 45
- synchrone, p. 17
- système cryptographique
 produit, p. 34
- système de chiffrement
 itéré, p. 40
- système de gestion
 des clefs, p. 45
- tatouge, p. 31
- terme de degré i , p. 105
- test de Lucas, p. 100
- test de primalité
 de Solovay-Strassen, p. 101
- théorie de l'information, p. 23
- théorie de la complexité, p. 23
- Tiers de Confiance, p. 60
- tiers de confiance, p. 68
- transfert inconscient, p. 75
- Trusted Third Party, p. 60
- TTP, p. 60
- unité, p. 87
- unité de $\mathbb{K}[X]$, p. 106
- variable, p. 105
- watermarking, p. 31

XOR, p. 20
zero-knowledge proof,p. 32