UNIVERSITÉ PARIS 8
VINCENNES-SAINT-DENIS

2 rue de la Liberté
93526 SAINT-DENIS CEDEX

Licence 3

Mention : Informatique, parcours mathématiques

***

COURS D'INTRODUCTION A LA CRYPTOGRAPHIE

***

ENSEIGNANTE : SIHEM MESNAGER

Année Universitaire 2014-2015

1 – Introduction

1.1 – Cryptologie

La cryptologie consiste en l'étude des techniques destinées à protéger l'information contre une attaque malveillante. En ce sens, elle diffère du codage correcteur d'erreur qui protège l'information contre une altération fortuite : bruit thermique ou galactique, parasite radio, rayure sur un disque. Trois acteurs font partie d'un système cryptologique : l'émetteur et le destinataire du message mais aussi l'adversaire. Ce dernier peut être éventuellement l'un des deux protagonistes, dans le cas où, par exemple, l'émetteur d'un document voudrait répudier sa signature.
La cryptologie est la réunion de deux disciplines :
- la *cryptographie* a pour objet la conception de systèmes résistants ;
- la *cryptanalyse* consiste en l'attaque et la recherche de failles sur des systèmes existants.

Ces deux disciplines s'alimentent l'une l'autre. On ne peut pas évaluer la sécurité d'un mécanisme sans le soumettre à des attaques qui, à leur tour, conduisent à des critères de conception pour rendre les procédés plus sûrs. Ces derniers seront à nouveau passés au crible du cryptanalyste...

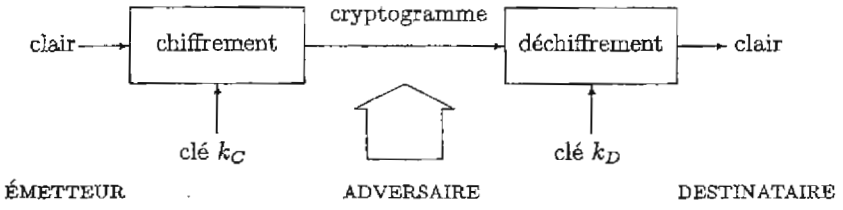
1.2 – Système cryptographique

Un système cryptographique est un ensemble de protocoles et d'algorithmes qui permettent à l'émetteur et au destinataire d'échanger des données de manière sûre. Ceux-ci se mettent préalablement d'accord sur un secret appelé *clé* et sur le procédé qui utilisera ce secret pour transformer les messages.
Le chiffrement est la technique dont l'objectif est de maintenir le message secret. C'est l'utilisation traditionnelle de la cryptographie qui trouve son origine très ancienne dans les applications militaires. Les ordres transmis par les généraux doivent demeurés secrets y compris lorsque le message tombe aux mains de l'ennemi. Actuellement, le chiffrement n'est qu'un service parmi d'autres offert par la cryptographie (voir § 1.7).
L'émetteur chiffre un message clair à l'aide d'un algorithme de chiffrement E qui utilise une clé de chiffrement k_C. Cela le transforme en un *cryptogramme*, incompréhensible pour l'adversaire. Le destinataire reçoit le cryptogramme et le déchiffre à l'aide d'un algorithme de déchiffrement D et d'une clé de déchiffrement k_D. Il reconstitue alors le message clair.

Un système cryptographique de chiffrement est la donnée de
- un ensemble de messages clairs, $\mathcal{M}$, et un ensemble de cryptogrammes, $\mathcal{C}$. Il s'agit souvent de l'ensemble $\{0,1\}^*$ des mots sur l'alphabet $\{0,1\}$. Dans le système RSA, il s'agit de l'ensemble $\mathbb{Z}/N\mathbb{Z}$ des entiers modulo N.
- un ensemble $\mathcal{K}_\mathcal{C}$ de clés de chiffrement k_C ;
- un algorithme de chiffrement E_{k_C} paramétré par une clé de chiffrement k_C. Cet algorithme n'est pas nécessairement déterministe. Il se peut, et c'est d'ailleurs préférable, qu'un même message soit chiffré différemment lors de deux envois.
- un ensemble $\mathcal{K}_\mathcal{D}$ de clés de déchiffrement k_D. La clé de déchiffrement n'est pas nécessairement égale à la clé de chiffrement.
- un algorithme déterministe de déchiffrement D_{k_D} paramétré par une clé de déchiffrement k_D. Cet algorithme doit déchiffrer correctement les clairs, c'est-à-dire

$$\forall m \in \mathcal{M} \quad D_{k_D} \circ E_{k_C}(m) = m$$

Tout ceci peut se résumer par le schéma suivant :



1.3 – Sécurité inconditionnelle

La sécurité est une notion probabiliste. L'adversaire réussit ou échoue avec une certaine probabilité. On dira qu'un système est inconditionnellement sûr si la meilleure stratégie de l'adversaire est de choisir le clair au hasard.

THÉORÈME. *Si un système, dans lequel les ensembles des messages clairs des clés de déchiffrement sont finis, est inconditionnellement sûr alors le cardinal de l'ensemble des messages clairs est inférieur au cardinal de l'ensemble des clés de déchiffrement.*

Preuve. Si tel n'est pas le cas, le tirage d'une clé au hasard conduit plus probablement au message clair que le tirage au hasard du clair lui-même. □

En particulier, l'utilisation d'un algorithme de type AES avec des clés de 128 bits n'est pas inconditionnellement sûr dès lors qu'il s'agit de chiffrer des messages de plus de 128 bits. On peut toutefois utiliser cet algorithme en toute confiance, la puissance de calcul pour retrouver la clé par recherche exhaustive reste hors de portée des moyens actuels.

1.4 – Cryptographie symétrique

On dit qu'un système cryptographique est *symétrique* lorsque les clés de chiffrement et de déchiffrement se déduisent facilement l'une de l'autre. Les deux clés doivent donc être maintenues secrètes pour garantir la sécurité.
On peut toujours supposer que les clés k_C et k_D sont égales, quitte par exemple à intégrer le calcul $k_C \mapsto k_D$ dans l'algorithme de déchiffrement. On supposera désormais que dans un système symétrique, on a $k_C = k_D = k$.
Le transport des clés est la principale difficulté de mise en œuvre de la cryptographie symétrique. Dans un réseau qui comprend n participants, il faut protéger chaque liaison avec des clés toutes distinctes, ce qui nécessite un grand nombre de clés (égal à $n(n-1)/2$).

1.5 – Cryptographie asymétrique

Jusqu'en 1976, date d'invention de la cryptographie à clé publique par DIFFIE et HELLMANN, les seuls systèmes connus étaient symétriques.
On dit qu'un système est *asymétrique* lorsqu'il est facile de déduire la clé de chiffrement k_C à partir de la clé de déchiffrement k_D, mais que l'opération inverse est pratiquement impossible.
On peut alors diffuser la clé de chiffrement (clé publique). Seule la clé de déchiffrement doit être maintenue secrète (clé privée). Ainsi, tout le monde peut chiffrer un message, mais seul le destinataire, détenteur de la clé privée, a les moyens de le déchiffrer.
Les systèmes asymétriques reposent sur des problèmes mathématiques réputés difficiles. Par exemple, le RSA (1978) repose sur la difficulté de factoriser les grands nombres. Il est facile de calculer $n = p \times q$, mais, étant donné n seulement, s'il est assez grand, il est difficile de retrouver p et q.
L'asymétrie des clés autorise d'autres services comme par exemple celui de signature électronique. Il est en quelque sorte le dual du service de chiffrement. En rendant publique la clé de déchiffrement et en maintenant secrète la clé de chiffrement, il permet à tout le monde de vérifier une signature, alors que seul l'émetteur qui dispose de la clé privée est capable de la produire.

Dans un réseau de n participants, lorsqu'un utilisateur veut transmettre un message, il consulte un annuaire pour trouver la clé publique de son correspondant qui, lui seul, peut déchiffrer le message à l'aide de sa clé privée. Il n'y a besoin que d'un couple (clé publique, clé privée) par utilisateur. Les systèmes asymétriques simplifient la gestion des clés, mais les clés doivent être plus longues et les calculs sont plus lents que dans les systèmes symétriques. Dans la pratique, un système asymétrique est utilisé pour échanger une clé secrète qui servira à chiffrer les données à l'aide d'un système symétrique.
La clé publique n'est pas confidentielle, par contre, on doit s'assurer de son origine pour éviter qu'un adversaire ne substitue sa propre clé publique et ainsi déchiffrer des messages qui ne lui sont pas destinés. C'est pourquoi une *autorité de certification* certifie les clés publiques à l'aide d'un mécanisme de signature numérique, ce qui en garantit l'authenticité.

1.6 – Modèles d'attaque

Le principe de KERCKOFFS (*La cryptographie militaire* 1883) est maintenant universellement admis :

La description des mécanismes mis en œuvre, en particulier les algorithmes et les protocoles, peut être rendue publique. La sécurité repose uniquement sur le secret de la clé (secrète ou privée).

On emploie le terme *décrypter* pour désigner la reconstitution du message clair par un adversaire qui ne dispose pas de la clé.

Il existe plusieurs modèles d'attaques selon les informations dont dispose l'adversaire.

1. *attaque à cryptogramme seul* : l'adversaire ne dispose que du cryptogramme. C'est l'attaque la plus difficile. Elle est impossible avec les systèmes modernes.
2. *attaque à couples (clairs, cryptogrammes) connus* : l'adversaire dispose d'un certain nombre de messages et de leurs cryptogrammes. C'est le cas, par exemple, lorsque les messages commencent tous par le même en-tête. C'est ce qui a permis la cryptanalyse de la machine ENIGMA durant la seconde guerre mondiale.
3. *attaque à clairs choisis* : l'adversaire dispose d'un *oracle* à qui il peut demander le chiffrement d'un message de son choix. Cette attaque est le modèle minimal qu'il faut considérer pour évaluer la sécurité d'un système de chiffrement asymétrique. La clé publique permet à l'adversaire de chiffrer tous les messages qu'il désire.
4. *attaque à cryptogrammes choisis* : l'adversaire dispose d'un *oracle* à qui il peut demander le déchiffrement d'un message de son choix à l'exception du message à décrypter, par exemple en se faisant passer temporairement pour le véritable destinataire.

Une attaque à clairs ou cryptogramme choisis est dite *adaptative* lorsque le choix du clair ou du cryptogramme dépend des résultats précédents.

1.7 – Services de sécurité

La cryptographie moderne propose bien d'autres services de sécurité que la seule confidentialité. Chaque service protège l'information contre une menace particulière.

CONFIDENTIALITÉ : protéger le contenu d'une information, c'est-à-dire la garder secrète sauf pour le destinataire.

INTÉGRITÉ : s'assurer que l'information n'a pas été intentionnellement altérée au cours de son transport ou de son stockage ;

AUTHENTIFICATION ou SIGNATURE : garantir la source de l'information.

CONTRÔLE D'ACCÈS : s'assurer qu'une information diffusée n'est accessible qu'aux destinataires autorisés, comme par exemple la TV à péage.

RÉVOCATION : empêcher l'accès à l'information après la résiliation d'un abonnement, ou le dépassement d'une date de validité.

NON RÉPUDIATION : empêcher de se rétracter d'un engagement ou de nier avoir signé un document.

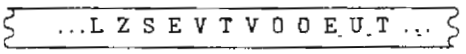
2 – Chiffrements traditionnels

Dans les systèmes traditionnels, l'espace des clairs et des cryptogrammes est souvent l'ensemble des mots sur l'alphabet latin majuscule. $\mathcal{A} = \{\mathtt{A}, \ldots, \mathtt{Z}\}$. L'usage est d'ignorer les espaces et les autres caractères, puis de regrouper les lettres du cryptogramme cinq par cinq. Le destinataire doit reconstituer lui-même le découpage en mots et en phrases. Sont présentés ci-après quelques uns des plus représentatifs.

2.1 – Transposition

Dans un système à transposition, le cryptogramme est obtenu en changeant l'ordre des lettres du message clair.

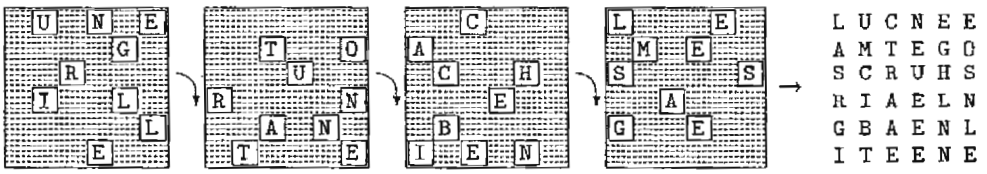
a) La Scytale (Sparte, 400 avant J.C.) Le procédé consiste à écrire le message sur un ruban enroulé autour d'un bâton dont le diamètre est maintenu secret.



Il semblerait que la scytale ait servi comme moyen luxueux d'échange de courrier, plutôt que pour camoufler des messages.

b) La grille tournante. On utilise une grille 6×6 (par exemple) dans laquelle sont percés 9 trous de telle sorte que lorsqu'on fait faire 1/4 de tour à la grille, les trous ne se superposent pas. Les lettres du message clair (de 36 lettres) sont écrites successivement dans les trous de la grille en lui faisant faire quatre fois 1/4 de tour. Ce procédé est utilisé dans le roman de Jules Verne, *Mathias Sandorf*. Ci-dessous, est montré le chiffrement de la phrase

UNE GRILLE TOURNANTE CACHE BIEN LE MESSAGE



Ce procédé est sensible à l'attaque à *mot probable*, si on sait qu'un mot est très probablement présent dans le clair (clair connu).

2.2 – Substitution simple

Dans un système à substitution, on remplace une lettre par une autre. Ces systèmes diffèrent selon le procédé utilisé pour effectuer ce remplacement.

a) Chiffre de César. Il consiste à décaler cycliquement l'alphabet de quelques rangs. Lorsqu'on code chaque lettre par un nombre compris entre 0 et 25 ($\mathtt{A} = 0, \mathtt{B} = 1, \ldots, \mathtt{Z} = 25$), cela consiste à effectuer la transformation $x \mapsto x + k \bmod 26$. Par exemple, avec $k = 3$, CESAR devient FHVDU.
Le secret réside dans la valeur k du décalage. Ce procédé ne résiste pas à l'essai exhaustif de toutes les valeurs possibles.

b) Rot13. Il s'agit d'un chiffre de César avec $k = 13$. Cette valeur a la propriété de rendre la transformation involutive, *i.e.* c'est le même procédé qui est utilisé pour chiffrer et déchiffrer.
Ce procédé est utilisé pour brouiller (plutôt que pour chiffrer, car il n'y a pas de secret) les courriers électroniques sur internet. La fonction est toujours présente sur les navigateurs.

c) **Chiffre de** WOLSELEY. Cette méthode permet de construire une substitution involutive à partir d'une phrase clé. Le procédé pour chiffrer est le même que pour déchiffrer. Deux correspondants conviennent par exemple d'utiliser la phrase clé « SESAME OUVRE TOI ». Les lettres sont rangées dans l'ordre dans un carré 5×5 sans répéter deux fois la même lettre. La grille est ensuite complétée avec le reste de l'alphabet, en omettant la lettre W. La substitution consiste à échanger deux lettres qui sont symétriques par rapport au centre ce ce carré.

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

d) Cryptanalyse. Les substitutions simples sont sensibles à ce qui est appelé l'*analyse des fréquences simple*. Il s'agit d'une attaque statistique sur le cryptogramme seul. Connaissant la fréquence des lettres dans la langue utilisée, il est possible de reconstituer la substitution qui a servie. Si cela s'avère insuffisant pour reconstituer un texte compréhensible, on peut aussi d'utiliser la fréquence des bigrammes, des lettres doubles ou des trigrammes.
Des améliorations ont été proposées pour lisser les fréquences des lettres et ainsi limiter l'efficacité des attaques statistiques. On peut par exemple supprimer l'usage de la lettre W qui est très rare en Français et mettre à profit la lettre rendue disponible pour coder le E aléatoirement de deux façons différentes.

2.3 – Substitutions polygrammiques

Une substitution polygrammique est une substitution sur plusieurs lettres. On présente ci-dessous deux substitutions sur les couples de lettres.

a) Chiffre de PLAYFAIR. Ce procédé a été inventé par Charles WEATSTONE en 1854, puis popularisé par Lord PLAYFAIR à qui il doit son nom.
Pour ce chiffre, on commence par remplir une grille 5×5 à partir d'une phrase clé comme pour le chiffre de WOLSELEY. Ce carré sert à définir une substitution sur des couples de lettres. Trois cas sont possibles.

- Si les deux lettres sont situées sur la même ligne ou la même colonne, elles sont remplacées par les deux lettres suivantes cycliquement. Dans l'exemple ci-dessus, (E, A) → (A, M) ; (J, Q) → (Q, E)
- Si les deux lettres ne sont ni sur la même ligne, ni sur la même colonne, elles définissent la diagonale d'un rectangle. On les remplace alors par les lettres de l'autre diagonale, les deux premières lettres étant sur la même ligne. Ainsi dans l'exemple ci dessus (V, L) → (T, J) ; (N, B) → (H, G)
- S'il s'agit d'une lettre double, on insère entre elles une lettre quelconque (en général un X) pour éliminer ce cas.

Ainsi, PLAYFAIR se chiffre par YHXMDMUT.
Tout comme les substitutions simples, ce procédé est sensible à l'attaque statistique. Si on dispose des fréquences des bigrammes de la langue utilisée et d'un cryptogramme suffisamment long, il est possible de reconstituer le clair.

b) Chiffre de HILL. Comme pour le chiffre de CÉSAR, chaque lettre du message est remplacée par son rang dans l'alphabet (A = 0, B = 1, ...). Puis elles sont regroupées deux par deux afin de former des vecteurs de deux composantes. La clé est k une matrice inversible 2×2 égale à $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, à coefficients dans $\mathbb{Z}/26\mathbb{Z}$. Le cryptogramme est calculé en multipliant chaque vecteur du clair par cette matrice. Pour deux lettres (x, y) du clair, le cryptogramme se calcule par $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \bmod 26 \\ cx + dy \bmod 26 \end{pmatrix}$. On retrouve le clair à partir du cryptogramme en effectuant la même opération avec la matrice inverse k^{-1}, égale à $\dfrac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.

Exemple avec la clé $k = \begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$. La clé pour le déchiffrement est $k^{-1} = \begin{pmatrix} 5 & 12 \\ 15 & 25 \end{pmatrix}$.

Message clair	:	LE	CHIFFRE	DE	HILL			
Codage	:	(11,4)	(2,7)	(8,5)	(5,17)	(4,3)	(4,7)	(8,11)	(11,0)
Multiplication par k	:	(11,5)	(20,7)	(14,23)	(9,14)	(22,15)	(12,17)	(12,13)	(21,3)
Cryptogramme	:	LFUHO	XJOWP	MRMNV	D			

La fonction de chiffrement est *linéaire*. Ce système ne résiste pas à une attaque à clair connu. La connaissance d'un message et de son cryptogramme permet de reconstituer la matrice qui a servi au chiffrement en résolvant un système linéaire de 4 équations à 4 inconnues.

2.4 – Le carré de VIGENÈRE (1586)

C'est de loin le plus sûr des systèmes traditionnels. Il s'avère bien plus sûr que d'autres procédés qui lui sont postérieurs.
Il s'agit d'un chiffre de CÉSAR dont le décalage est variable selon une clé cycliquement répétée. Chaque lettre du cryptogramme est calculé à partir de la lettre du clair m_i et de la lettre de la clé située à cette position k_i par la formule

$$c_i = m_i + k_i \pmod{26}$$

On retrouve le clair par l'opération inverse :

$$m_i = c_i - k_i \pmod{26}$$

Exemple.

Clé	:	CODECODE
Message	:	VIGENERE
Cryptogramme	:	XWJIPSUI

Le système doit son nom à la table carrée contenant tous les décalages de l'alphabet selon le caractère de la clé et qui était destiné à faciliter le calcul du cryptogramme. Il s'agit en fait de la table d'addition dans $\mathbb{Z}/26\mathbb{Z}$ où les éléments sont notés par les lettres de l'alphabet latin.
Ce système résiste à l'analyse des fréquences simples. C'est, encore maintenant, le système qui a résisté le plus longtemps à toute attaque. Sa première cryptanalyse a été effectuée par BABBAGE en 1849. Il s'agissait d'une cryptanalyse à clair connu. En 1922, FRIEDMANN a proposé une cryptanalyse à cryptogramme seul. Les étapes de cette attaque sont :

1. Déterminer la longueur ℓ de la clé par recherche exhaustive. On utilise le calcul d'une quantité appelée *indice de coïncidence* qui permet de décider si une suite de caractères est aléatoire, ou correspond à une substitution pour une langue donnée.
2. Effectuer une analyse des fréquences simple sur le cryptogramme décimé (une lettre sur ℓ).

On appelle *indice de coïncidence* la probabilité de tirer deux lettre identiques dans un texte. Soit p_i la probabilité d'occurrence de la i^e lettre. La valeur de l'indice de coïncidence IC est donnée par

$$IC = \sum_{i=0}^{25} p_i^2.$$

Pour une distribution aléatoire, cette valeur est $\frac{1}{26} \approx 0,038$. Pour la langue française, la valeur est environ $0,074$. On calcule l'indice de coïncidence empirique sur les suites décimées du cryptogramme obtenues en ne considérant qu'une lettre sur ℓ. Lorsque ℓ est égal à la taille de la clé, la valeur de cet indice est notablement différente.

Ce qui permet la cryptanalyse est la répétition cyclique de la clé et l'utilisation plusieurs fois des mêmes caractères. Si la clé est aléatoire ou si elle n'a pas de structure exploitable, le système de VIGENÈRE reste encore sûr aujourd'hui.

La table du carré de VIGENÈRE

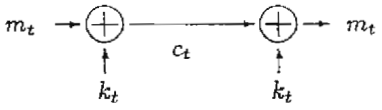
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T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

3 – Chiffrement symétrique moderne

3.1 – Le chiffrement à flot *(stream cipher)*

L'ensemble des messages clairs est l'ensemble $\{0,1\}^*$ des mots sur l'alphabet $\{0,1\}$. Le chiffrement consiste à combiner chaque symbole par une addition modulo 2 (*ou exclusif*) avec un terme d'une suite binaire $k \in \{0,1\}^*$ appelée *masque* ou *suite chiffrante*.

$$c_t = m_t \oplus k_t$$



Exemple.

Message	$: m$	$=$ 00110001010011110011
Suite chiffrante	$: k$	$=$ 11010111001001111010
Cryptogramme	$: c$	$=$ 11100110011010010001

Le destinataire déchiffre en effectuant la même opération, avec la même suite : $m_t = c_t \oplus k_t$.

Ce dispositif peut être *synchrone* ou *autosynchronisant*.

Les différents systèmes de chiffrement à flot sont classés selon la manière dont est produite la suite chiffrante $k = (k_t)_{t\in\mathbb{N}}$.

a) Le masque jetable (Vernam, 1917)

La suite chiffrante est une suite d'aléa pur.

Un aléa pur est produit par une source physique non prédictible : lancé d'une pièce, bruit de fond dans un composant électronique, instants précis d'actionnement des touches sur un clavier d'ordinateur *etc.*

Pour tout cryptogramme, tous les clairs possibles sont équiprobables. Ce système atteint donc la *sécurité inconditionnelle*. C'est le seul pour lequel on puisse le prouver. La preuve repose sur le caractère *aléatoire* de la suite chiffrante. Chaque terme ne doit être utilisé qu'une seule fois.

Les correspondants doivent échanger de façon sûre une très grande quantité d'aléa. Cela rend la mise en œuvre très délicate. Ce système est réservé aux communications stratégique de très haut niveau de confidentialité.

b) Générateur pseudo-aléatoire

Le système à générateur pseudo-aléatoire (GPA) cherche à simuler un système de Vernam. A partir d'une clé de taille réduite (128 bits), le GPA produit une longue séquence, de la taille des données à chiffrer, et utilisée comme suite chiffrante. Cette suite doit :

1. être reproductible à l'aide de la clé pour que le destinataire puisse déchiffrer ;
2. ne pas être distinguable d'un véritable aléa pour quiconque ne dispose pas de la clé. En particulier, elle doit avoir de bonnes propriétés statistiques et ne pas être prédictible.

c) Clé de message

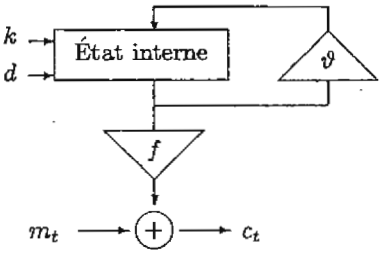
Une même suite chiffrante ne doit pas être utilisée deux fois pour chiffrer deux messages différents, d'où la nécessité de diversifier la suite pseudo-aléatoire chiffrante à l'aide d'un complément de clé appelé *clé de message* ou *diversifiant*.

Ce complément de clé est transmis avec cryptogramme. Le destinataire en a besoin pour reconstituer la suite chiffrante. Cette donnée n'est pas secrète.

d) **Chiffrement synchrone.**

Dans le chiffrement synchrone, la suite chiffrante pseudo-aléatoire est construite à partir d'un automate à nombre d'états fini. Le secret réside dans l'état initial de l'automate.
Par définition, un chiffreur synchrone est la donnée d'un quadruplet $\mathcal{A} = (\mathcal{E}, \vartheta, f, i)$, où :

- $\mathcal{E}$ est un ensemble fini d'états, par exemple $\mathcal{E} = \{0,1\}^n$ ou $\mathcal{E} = \{0, \ldots, N-1\}$, pour $N \in \mathbb{N}^*$;
- ϑ est la fonction de transition d'état $\mathcal{E} \to \mathcal{E}$; l'état de l'automate à l'instant suivant est $e_{t+1} = \vartheta(e_t)$.
- f est la fonction d'extraction de pseudo-aléa $\mathcal{E} \to \{0,1\}$. Chaque terme de la suite pseudo-aléatoire chiffrante est fonction de l'état interne : $k_t = f(e_t)$;
- $i : \mathcal{K} \times \mathcal{D} \to \mathcal{E}$ est la fonction d'initialisation. Elle définit l'état initial de l'automate à partir d'une clé $k \in \mathcal{K}$ et d'une clé de message (un diversifiant) $d \in \mathcal{D}$.



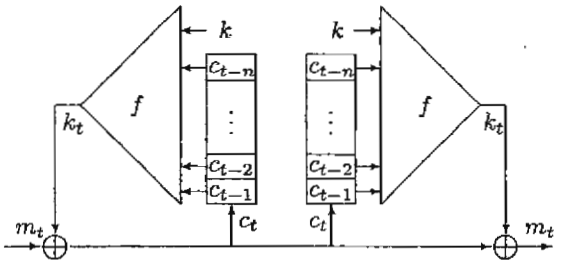
Ce système ne propage pas les erreurs. Si un symbole est mal transmis, l'erreur reste localisée au même endroit sur le clair déchiffré. Il est donc particulièrement adapté pour chiffrer dans un environnement très bruité comme par exemple les communications radio téléphoniques.
Par contre, synchrone. Si un symbole est perdu sans que le destinataire le sache, la suite chiffrante est alors décalée entre les deux correspondants et le déchiffrement est devenu impossible.

e) **Distance d'unicité**

Considérons un chiffreur synchrone qui a 2^n états internes possibles. Si on demande à ce chiffreur de produire des suites binaires de longueur k avec $k < n$, alors il y a moins de suites que d'états initiaux. En conséquence, il existe deux états initiaux qui produisent la même suite, du moins si on se limite aux k premier termes. Lorsque $k \geq n$, ce raisonnement ne s'applique plus.
On appelle *distance d'unicité* d'un chiffreur synchrone le nombre de terme à produire pour être sûr qu'il est le résultat d'une unique initialisation. Nécessairement, cette valeur d vérifie $d \geq n$.

f) **Chiffrement autosynchronisant**

On dit que le chiffrement est autosynchronisant lorsque la suite chiffrante est une fonction secrète des n derniers symboles de cryptogramme $(c_{t-1}, \ldots, c_{t-n})$.
Ceci est réalisé par un registre à décalage non rebouclé de taille n comme indiqué sur la figure ci-après.

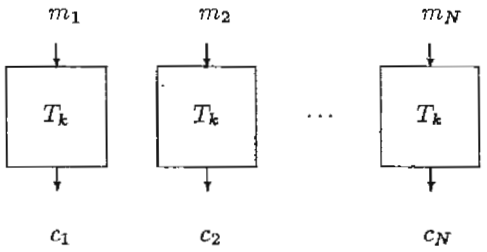


La fonction secrète f_k est une fonction connue f et paramétrée par la clé secrète k.
Si un symbole de cryptogramme est altéré ou perdu, alors n symboles de clair sont perdus, mais le dispositif se resynchronise automatiquement après la lecture correcte de n symboles de cryptogrammes.
Le paramètre n s'appelle la *distance de synchronisation.*

3.2 – Le chiffrement par blocs

a) Définition. Un chiffrement par bloc réalise une substitution simple sur un alphabet de très grande taille: $A = \{0,1\}^n$ avec $n = 64$ ou $n = 128$. L'entier n est la taille du bloc.
La clé secrète k est utilisée pour paramétrer une transformation $T_k : A \mapsto A$. Cette transformation est appelée *primitive de chiffrement par bloc.*
Pour toute clé k, la transformation T_k doit être bijective. La transformation inverse T_k^{-1} est requise pour le déchiffrement.
La taille du bloc doit être suffisante ($n \geq 64$) pour empêcher une attaque statistique et une attaque exhaustive consistant à créer, pour claque clé, la collection des couples (clair, cryptogramme).
La transformation T_k doit présenter un caractère aléatoire. Lorsqu'on choisit aléatoirement une clé k on ne doit pas pouvoir distinguer la transformation T_k d'une bijection obtenue par tirage aléatoire dans l'ensemble de toutes les bijections sur l'alphabet A. Pour tout $x \in A$, la valeur $T_k(x)$ doit être imprédictible lorsqu'on ne connaît pas k.
Les principaux modes d'utilisation d'une primitive de chiffrement par bloc pour chiffrer des messages sont décrits ci-après.

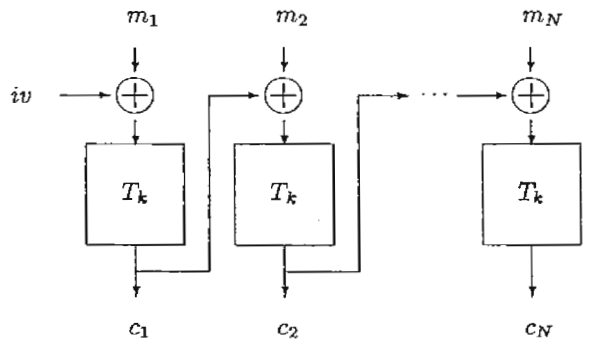
b) Mode ECB. (*Electronic CodeBook* ou mode dictionnaire).
Le message clair m est découpé en blocs de n bits: $m = (m_1, \ldots, m_N)$. Si la taille du message m n'est pas multiple de n, on doit le compléter.
Chaque bloc est traité indépendamment des autres. Le cryptogramme est calculé par $c_i = T_k(m_i)$



Le déchiffrement consiste à appliquer la transformation inverse à chaque bloc de cryptogramme: $m_i = T_k^{-1}(c_i)$.
Une erreur de transmission sur un symbole du cryptogramme crée un bloc entier erroné sur le clair déchiffré.
Si deux blocs de cryptogramme sont égaux, alors les deux blocs de clair correspondants le sont aussi. Le cryptogramme peut donner une information sur le clair. Pour cette raison, le mode ECB est considéré comme **non sûr**. Par exemple, une image chiffrée dans ce mode, lorsqu'elle est constituée de zones uniformes, est souvent parfaitement reconnaissable.

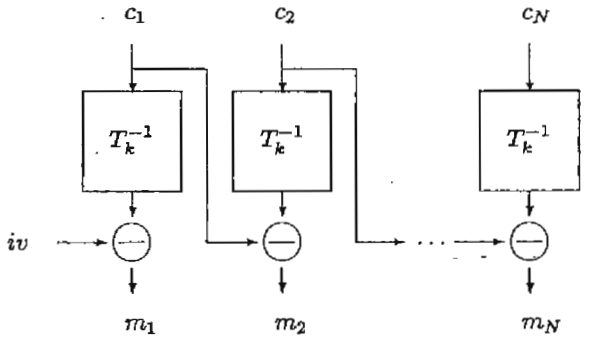
c) Mode CBC. (*Cipher Block Chainig*).
Comme dans le mode ECB, le message clair est découpé en blocs de n bits: $m = (m_1, \ldots, m_N)$.
iv est un vecteur d'initialisation aléatoire de n bits. Le cryptogramme est calculé ainsi:

$$\begin{cases} c_1 = T_k(m_1 \oplus iv) \\ \text{et pour } i \geq 2,\ c_i = T_k(m_i \oplus c_{i-1}) \end{cases}$$



Le cryptogramme est constitué du vecteur d'initialisation iv suivi des blocs c_i : $c = (iv, c_1, \ldots, c_N)$.
L'opération $\oplus$ est en général l'addition modulo 2 bit à bit sur $\{0,1\}^n$. Cela peut être également n'importe quelle autre opération de groupe, comme par exemple l'addition des entiers modulo 2^n.
Si deux messages clairs diffèrent sur le bloc m_i, le cryptogramme c_i ainsi que tous les suivants seront totalement erronés, avec un bit sur deux faux en moyenne.
Le même message chiffré avec deux vecteurs iv différents seront complètement différents. Le mode CBC est **prouvé sûr** dès lors que la taille des blocs est suffisante.
Le chiffrement est séquentiel. On ne peut pas calculer c_{i+1} sans avoir au préalable calculé c_i.
Le message clair est reconstitué à partir du cryptogramme en effectuant l'opération inverse :

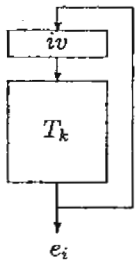
$$\begin{cases} m_1 = T_k(c_1) \ominus iv \\ \text{et pour } i \geq 2,\ m_i = T_k(c_i) \ominus c_{i-1} \end{cases}$$



Si une erreur de transmission apparaît sur le bloc c_i, le bloc déchiffré m_i sera complètement erroné (1 bit sur 2 en moyenne faux) et une erreur sera localisé au même endroit sur le bloc m_{i+1}.

d) Mode OFB. (*Output FeedBack*).
Le mode OFB utilise la transformation T_k pour produire une séquence pseudo-aléatoire qui sert à simuler un masque jetable.
À partir d'un vecteur d'initialisation iv aléatoire, l'émetteur et le destinataire construisent la suite $(e_i)_{i\geq 1}$ par :

$$\begin{cases} e_1 = T_K(iv) \\ \text{et pour } i \geq 2,\ e_i = T_K(e_{i-1}) \end{cases}$$



Dans ce mode, c'est également T_k, et non pas son inverse qui est utilisée pour déchiffrer. Il n'est donc pas nécessaire que T_k soit bijective.
Le cryptogramme est donné par $c_i = m_i \oplus e_i$.
Le clair est reconstitué par $m_i = c_i \ominus e_i$
Le mode OFB peut chiffrer des messages de taille arbitraire qui n'ont pas besoin d'avoir pour longueur un multiple de la taille du bloc.
Ce mode ne propage pas les erreurs. Une erreur de transmission sur le cryptogramme reste localisée au même endroit sur le clair déchiffré.

4 – Registres à décalage

4.1 – Polynômes binaires

Le corps $\mathbb{F}_2$ est par définition le corps à deux éléments $\mathbb{F}_2 = \{0, 1\}$ muni de l'addition et de la multiplication modulo 2 ($1 + 1 = 0$).

L'ensemble des mots binaires de longueur n se note $\mathbb{F}_2^n$. Il est muni d'une structure d'espace vectoriel de dimension n sur $\mathbb{F}_2$. L'addition de deux vecteurs s'effectue en additionnant modulo 2 composante à composante. Par exemple dans $\mathbb{F}_2^3$, on a $(0,1,1) + (1,1,1) = (1,1,0)$.
Une forme linéaire sur $\mathbb{F}_2^n$ est une application :

$$\begin{array}{rcl} \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2 \\ x = (x_0, \ldots, x_{n-1}) & \mapsto & a_0x_0 + \cdots + a_{n-1}x_{n-1} = a \cdot x \end{array}$$

L'ensemble $\mathbb{F}_2[X]$ est l'ensemble des polynômes à coefficients dans $\mathbb{F}_2$. Dans $\mathbb{F}_2[X]$, il existe une *division euclidienne*. Pour tout $A \in \mathbb{F}_2[X]$ et tout $B \in \mathbb{F}_2[X]$ non nul, il existe un unique couple $(Q, R) \in \mathbb{F}_2[X] \times \mathbb{F}_2[X]$ qui vérifie :

$$\begin{cases} A = BQ + R \\ \deg(R) < \deg(B) \end{cases}$$

L'ensemble $(\mathbb{F}_2[X], +, \times)$ est par conséquent un anneau commutatif unitaire intègre dans lequel tout idéal est principal. On peut y appliquer l'algorithme d'Euclide pour calculer le pgcd de deux polynômes.

Théorème de Bézout. *Pour tout polynôme A et B de $\mathbb{F}_2[X]$, non tous deux nuls, il existe deux polynômes U et V dans $\mathbb{F}_2[X]$ tel que*

$$AU + BV = \text{pgcd}(A, B).$$

Preuve. Soit $\mathcal{D}$ l'ensemble des polynômes non nuls qui s'écrivent $AU + BV$. Considérons P un élément de degré minimal de $\mathcal{D}$. P est un diviseur de A, car dans le cas contraire, le reste de la division de A par P serait un élément non nul appartenant également à $\mathcal{D}$ et serait de degré strictement inférieur contrairement à l'hypothèse. De même, P divise B. De plus, tout diviseur de A et B divise également $AU + BV$ donc divise P, ce qui montre que $P = \text{pgcd}(A, B)$. □

Il sera commode de représenter la forme linéaire $x \mapsto a \cdot x = a_0x_0 + \cdots + a_{n-1}x_{n-1}$ par le polynôme $a_0 + a_1X + \cdots + a_{n-1}X^{n-1}$.

Dans $\mathbb{F}_2[X]$, l'élévation au carré est une opération linéaire. En effet, soient P et Q deux éléments de $\mathbb{F}_2[X]$, on a $(P + Q)^2 = P^2 + 2PQ + Q^2 = P^2 + Q^2$ du fait que dans $\mathbb{F}_2$, on a $2 = 0$.
Par contre l'élévation au cube n'est pas linéaire, puisque par exemple $(1+X)^3 = 1+X+X^2+X^3 \neq 1 + X^3$.

LEMME. *Un polynôme P de $\mathbb{F}_2[X]$ n'a pas de facteur carré dans sa factorisation si et seulement si P et son polynôme dérivé sont premiers entre eux.*

Preuve. Si $P = Q^2R$, alors $P' = 2QQ'R + R'Q^2 = R'Q^2$. Le polynôme Q^2 divise P et P' qui ne sont donc pas premiers entre eux.

La réciproque se prouve par récurrence sur le degré de P. Cette propriété est vraie si $\deg(P) = 1$ car dans ce cas, P' est un polynôme constant qui est nécessairement premier avec P. Supposons que pour tout polynôme D de degré $\leq d$, si $\text{pgcd}(D, D') \neq 1$ alors D a un facteur carré. Soit P un polynôme de degré $d+1$ tel que $\text{pgcd}(P, P') = D \neq 1$. On a $P = DQ$ et $P' = DR$.

$$P' = DQ' + QD' = DR$$
$$D(Q' - R) = -QD'$$

Si D et D' sont premiers entre eux, alors $D|Q$. Comme $P = DQ$, il est divisible par D^2 et a donc un facteur carré.
Si D et D' ne sont pas premiers entre eux, alors D a un facteur carré par hypothèse de récurrence, et donc P aussi. □

N.B. Cette propriété est vrai pour un polynôme à coefficient dans un corps commutatif quelconque.

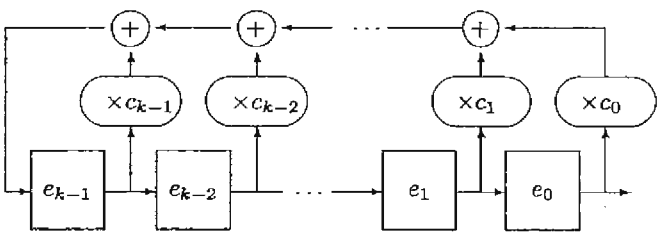
4.2 – LFSR *(Linear Feedback Shift Register)*

Un LFSR, ou registre à décalage rebouclé linéairement, est un dispositif matériel destiné à produire des suites binaires pseudo-aléatoires. Ces suites sont caractérisées par une période élevée et un bonne qualité statistique. Cependant, les suites suites produites sont sont peu complexes et prédictibles.
Pour constituer un générateur pseudo-aléatoire cryptographique destiné à un chiffrement à flots, ils doivent impérativement être associés à une fonction booléenne de filtrage qui assure la non-linéarité indispensable pour que la suite chiffrante ait les propriétés requises.

4.2.1. LFSR

Un registre à décalage de dimension k est constitué de k mémoires, appelées *bascule* ou *flip-flop* pouvant contenir chacune un symbole binaire d'information e_i.
Le vecteur binaire $(e_i)_{i\in\{0,\ldots,k-1\}}$ constitue l'état interne du registre.

Ces cases mémoires sont contrôlées par une horloge. À chaque top d'horloge, le contenu de la mémoire est remplacé par la donnée qui se trouve sur son entrée. Lorsque ces bascules sont chaînées, l'ensemble des données est décalé.
La figure suivante illustre un LFSR.



4.2.2. Matrice de transition. Soit $e_i(t)$ la valeur du contenu de la bascule numéro i à l'instant t. La fonction de transition du LFSR est :

$$\begin{cases} \forall i \in \{0, \ldots, k-2\}, \quad e_i(t+1) = e_{i+1}(t) \\ e_{k-1}(t+1) = \sum_{i=0}^{k-1} c_i e_i(t) \end{cases}$$

Cette fonction de transition est linéaire. Sa matrice, appelée *matrice de transition* du LFSR est :

$$A = \begin{pmatrix} 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \\ c_0 & c_1 & \cdots & c_{k-1} \end{pmatrix}.$$

DÉFINITION.[polynôme caractéristique] *Le polynôme caractéristique d'un LFSR est par définition le polynôme caractéristique de sa matrice de transition.*

Le polynôme caractéristique du LFSR illustré ci-dessus est donc

$$\begin{aligned} P(X) &= \begin{vmatrix} X & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \\ c_0 & c_1 & \cdots & X + c_{k-1} \end{vmatrix} \\ &= c_0 + c_1 X + \cdots + c_{k-2} X^{k-2} + c_{k-1} X^{k-1} + X^k \end{aligned}$$

4.2.3. Suite produite. La suite produite par un LFSR est une suite d'éléments de $\mathbb{F}_2$, dont les termes sont les états successifs de son état interne e_0. Du fait du décalage, les n premiers termes de cette suite sont les n composantes de l'état initial.

Par construction, les suites produites par un LFSR satisfont la relation de récurrence linéaire :

(1) $$\forall t \in \mathbb{N} \quad s_{t+k} + c_{k-1} s_{t+k-1} + c_{k-2} s_{t+k-2} + \cdots + c_1 s_{t+1} + c_0 s_t = 0$$

La représentation polynomiale de cette relation de récurrence linéaire correspond précisément au polynôme caractéristique du LFSR.

PROPOSITION. *Les termes s_t de la suite produite par un LFSR de polynôme caractéristique P sont une fonction linéaire de l'état initial représenté par le polynôme*

$$X^t \bmod P$$

Preuve : Soit $s = (s_t)_{t \in \mathbb{N}}$ la suite produite. Par construction, les k premiers termes correspondent respectivement à l'application des formes linéaires $1, X, X^2, \ldots, X^{k-1}$ à l'état initial $(s_0, s_1, \ldots, s_{k-1})$.
Supposons montré que pour tout entier $i \leq t + k - 1$, le terme s_i est le résultat de l'application à l'état initial de la forme linéaire représentée par le polynôme $X^i \bmod P$. D'après la relation de récurrence (1), le terme s_{t+k} est donné par

$$s_{t+k} = c_{k-1} s_{t+k-1} + c_{k-2} s_{t+k-2} + \cdots + c_1 s_{t+1} + c_0 s_t$$

Par application de l'hypothèse de récurrence, il s'agit de l'application à l'état initial de la forme linéaire représentée par le polynôme

$$c_{k-1} X^{t+k-1} + c_{k-2} X^{t+k-2} + \cdots + c_1 X^{t+1} + c_0 X^t = X^t \big(X^k + P(X)\big) \equiv X^{t+k} \bmod P$$

□

Toute suite périodique de période T peut être produite par un LFSR. Il suffit de considérer un LFSR de dimension T dont le polynôme caractéristique est $X^T + 1$.

Le vecteur nul est un état stable pour tout LFSR. Par conséquent, la période maximale d'une suite produite par un LFSR de dimension n est $2^n - 1$.

4.2.4. Polynôme de connexion.

DÉFINITION.[Polynôme réciproque] *Soit P un élément de $\mathbb{F}_2[X]$ de degré d. On appelle polynôme réciproque de P le polynôme*

$$\widetilde{P}(X) = X^d P\Big(\frac{1}{X}\Big)$$

Exemple. $P(X) = 1 + X + X^3$; $\widetilde{P}(X) = X^3\Big(1 + \frac{1}{X} + \frac{1}{X^3}\Big) = X^3 + X^2 + 1$.

Cette opération n'est pas involutive, par exemple

$$\begin{aligned} P(X) &= X^7 + X^8 \\ \widetilde{P}(X) &= X^8\Big(\frac{1}{X^7} + \frac{1}{X^8}\Big) = X + 1 \\ \widetilde{\widetilde{P}}(X) &= X\Big(\frac{1}{X} + 1\Big) = 1 + X \neq P(X) \end{aligned}$$

Le polynôme $\widetilde{P}$ a toujours un terme constant.
Réciproquement, si un polynôme Q a un terme constant, alors il est le réciproque d'un polynôme P égal à $\widetilde{Q}$, par exemple:

$$\begin{aligned} Q &= 1 + X^3 + X^4 \\ \widetilde{Q} &= X^4 + X + 1 \\ \widetilde{\widetilde{Q}} &= 1 + X^3 + X^4 = Q \end{aligned}$$

DÉFINITION.[polynôme de connexion d'un LFSR] *Le polynôme de connexion d'un LFSR est le réciproque de son polynôme caractéristique.*

4.3 – Série génératrice

DÉFINITION. *Soit $s = (s_t)_{t\in\mathbb{N}}$ une suite d'éléments de $\mathbb{F}_2$. La série génératrice de s est la série formelle*

$$S = s_0 + s_1 X + s_2 X^2 + \cdots + s_n X^n + \cdots = \sum_{t=0}^{\infty} s_t X^t$$

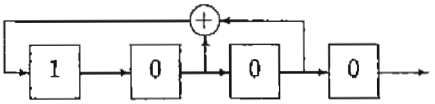
THÉORÈME. *S est la série génératrice d'une suite produite par un LFSR de polynôme caractéristique P de degré d si et seulement si il existe un polynôme Q de degré $< d$ tel que*

$$S\widetilde{P} = Q \qquad \textit{i.e.} \qquad S = \frac{Q}{\widetilde{P}}$$

Preuve : L'expression, pour tout t le coefficient du terme en X^{t+d} dans le produit $S\widetilde{P}$ est précisément égal au premier membre de la relation de récurrence (1) satisfaite par la suite produite. Ces coefficients sont donc nuls. Il ne reste que des termes de degré $< d$. □

Exemple.

On considère le LFSR schématisé ci-dessous :



Son polynôme caractéristique est $P = X + X^2 + X^4$;
Son polynôme de connexion est $\widetilde{P} = 1 + X^2 + X^3$.
Sa matrice de transition est

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

Il produit une suite binaire $s = (s_t)_{\in \mathbb{N}}$, définie par $\begin{cases} & s_0 = 0,\ s_1 = 0,\ s_2 = 0,\ s_3 = 1; \\ \text{et} & \forall t \in \mathbb{N} \quad s_{t+4} = s_{t+2} + s_{t+1} \end{cases}$

La suite s est ultimement périodique sans être périodique : $s = 0, \underbrace{0,0,1,0,1,1,1}, \underbrace{0,0,1,0,1,1,1}, \ldots$.

Sa série génératrice est

$$S = X^3 + X^5 + X^6 + X^7 + X^{10} + X^{12} + X^{13} + X^{14} + \cdots$$

Le calcul du produit $S\widetilde{P}$ donne

$$\begin{aligned} S\widetilde{P} &= X^3(1 + X^2 + X^3) + X^5(1 + X^2 + X^3) + X^6(1 + X^2 + X^3) + \cdots \\ &= X^3 \end{aligned}$$

d'où :

$$S = \frac{X^3}{1 + X^2 + X^3}$$

4.4 – Période des suites produites

La matrice de transition d'un LFSR est inversible si et seulement si son polynôme caractéristique a un terme constant. Dans ce cas, les suites produites par le LFSR sont strictement périodiques. En considérant le ppcm N des périodes des suites produites par ce LFSR, on conclut qu'il existe un entier N tel que $X^N \bmod P = 1$, *i.e.* P divise $X^N + 1$.

DÉFINITION. *On appelle ordre d'un polynôme $P \in \mathbb{F}_2[X]$ qui a un terme constant, le plus petit entier T tel que P divise $X^T + 1$.*

En d'autres termes, il s'agit du plus petit entier T tel que $X^T \equiv 1 \mod P$.

PROPOSITION. *Soit $S = \dfrac{Q}{\widetilde{P}}$ la série génératrice d'une suite s produite par un LFSR de polynôme caractéristique P. Si Q et $\widetilde{P}$ sont premiers entre eux, alors la période de la suite s est l'ordre de $\widetilde{P}$.*

Preuve. Soit T la période de s. La série S s'écrit donc

$$\begin{aligned} S &= R + X^T R + X^{2T} R + \cdots \\ &= R(1 + X^T + X^{2T} + \cdots \\ &= \frac{R}{1 + X^T} = \frac{Q}{\widetilde{P}} \\ \widetilde{P}R &= (1 + X^T)Q \end{aligned}$$

Le polynôme $\widetilde{P}$ divise $(1+X^T)Q$ et est premier avec Q. Il divise donc $1+X^T$.
Réciproquement, si $\widetilde{P}$ divise $1+X^T$, alors $1+X^T = \widetilde{P}P'$

$$S = \frac{Q}{\widetilde{P}} = \frac{QP'}{1+X^T}$$

Ce qui implique que T est une période de la suite dont la série génératrice est S. □

La question abordée maintenant est celle de la période des suites produites par un LFSR de dimension d et dont le polynôme de connexion est $\widetilde{P}$. Une telle suite a une série génératrice qui s'exprime $S = \frac{Q}{\widetilde{P}}$. On peut supposer que Q et $\widetilde{P}$ sont premiers entre eux. Si ce n'est pas le cas, on peut simplifier cette fraction et la même suite est produite par un LFSR plus petit. La période de la suite produite est alors égale à l'ordre de $\widetilde{P}$, ce qui peut se ramener à la période de la suite dont la série génératrice est $S = \frac{1}{\widetilde{P}}$.
Tous les cas se ramènent à une application successive des trois cas suivants :

Cas 1 : $\widetilde{P}$ est un polynôme irréductible de degré d. Soit $r = o(\widetilde{P})$ son ordre. Considérons l'ensemble $\mathbb{F}_2[X]/(P)$ des polynômes binaires modulo P. Tout polynôme R de degré $< d$ est premier avec P, donc, d'après le théorème de BÉZOUT, il existe U et V tels que $UR + VP = 1$, ce qui signifie que U est l'inverse de R modulo P. Tout polynôme non nul est donc inversible dans $\mathbb{F}_2[X]/(P)$. Il s'agit donc d'un corps. L'ordre de P est égal à l'ordre du sous-groupe multiplicatif de $\mathbb{F}_2[X]/(P)$ engendré par X. C'est un donc diviseur de $2^d - 1$.

PROPOSITION. *L'ordre d'un polynôme irréductible de degré d à coefficients dans $\mathbb{F}_2$ est un diviseur de $2^d - 1$.*

Lorsque $2^d - 1$ est premier (par exemple pour $d = 5$, on a $2^5 - 1 = 31$ qui est premier), alors l'ordre est maximal et vaut $2^d - 1$.

DÉFINITION [polynôme primitif]. *On dit qu'un polynôme de degré d à coefficients dans $\mathbb{F}_2$ est primitif lorsqu'il est irréductible et que son ordre vaut exactement $2^d - 1$.*

Lorsque le polynôme de connexion d'un LFSR est primitif, il produit des suites de période maximale.

Cas 2 : *Soit $\widetilde{P} = P_1^k$, une puissance d'un polynôme irréductible P_1. L'ordre r de $\widetilde{P}$ vaut*

$$o(\widetilde{P}) = 2^m r_1$$

où r_1 est l'ordre de P_1 et m est le plus petit exposant d'une puissance de 2 qui dépasse k.

Par exemple, pour $k = 2$, on a $m = 1$, pour $k = 3, 4$, on a $m = 2$, pour $k = 5, 6, 7, 8$, on a $m = 3$, *etc.* m est la partie entière supérieure du logarithme en base 2 de k.
Le polynôme $\widetilde{P} = P_1^k$ divise $(X^{r_1}+1)^{2^m} = X^{2^m r_1} + 1$, ce qui signifie que l'ordre r de $\widetilde{P}$ divise $2^m r_1$. Comme $X^r + 1$ est multiple de P_1^k et donc de P_1, l'entier r est multiple de r_1. Il en résulte que r est de la forme $r = 2^i r_1$ pour un entier $i \leq m$.
Le polynôme dérivé de $X^{r_1} + 1$ est soit 0, soit $X^{r_1 - 1}$ selon la parité de r_1. Il est premier avec $X^{r_1} - 1$. En conséquence, la factorisation de $X^{r_1} + 1$ n'a pas de facteurs carrés et le polynôme P_1 apparaît exactement 2^i fois dans $(X^{r_1}+1)^{2^i}$. Comme P_1^k est multiple de $(X^{r_1}+1)^{2^i}$, on a $k \leq 2^i$, ce qui prouve que $i = m$.
Finalement r vaut exactement $2^m r_1$.

Cas 3 : *Soit $\widetilde{P} = P_1 P_2$ le produit de deux polynômes P_1 et P_2 premiers entre eux. L'ordre de $\widetilde{P}$ est égal au ppcm des ordres de P_1 et P_2 :*

$$o(\widetilde{P}) = \text{ppcm}(o(P_1), o(P_2))$$

D'après le théorème de BÉZOUT, il existe deux polynômes U et V tels que $UP_1 + VP_2 = 1$.

$$S = \frac{1}{P_1 P_2} = \frac{UP_1 + VP_2}{P_1 P_2} = \frac{U}{P_2} + \frac{V}{P_1}$$

La suite s est la somme de deux suites s_1 et s_2, de séries génératrices respectives $\frac{U}{P_2}$ et $\frac{V}{P_1}$. Tout multiple des périodes de s_1 et s_2 est une période de s. La période t de s est donc un diviseur du ppcm des périodes de s_1 et de s_2.
Réciproquement, t étant la période de s, la relation $(X^t + 1)S = 0$ est satisfaite, d'où :

$$(X^t + 1)\left(\frac{U}{P_2} + \frac{V}{P_1}\right) = 0$$
$$S' = (X^t + 1)\frac{U}{P_2} = (X^t + 1)\frac{V}{P_1}$$

Respectivement de ces deux égalités, on déduit :

$$\begin{cases} S'VP_2 &= (X^t + 1)UV \\ S'UP_1 &= (X^t + 1)UV \end{cases}$$

Par addition membre à membre, $(X^t+1)S_1 = 0$ et $(X^t+1)S_2 = 0$, ce qui signifie que t est multiple des périodes de s_1 et de s_2. Finalement il en est le ppcm.

Exemple.

Quelle est la période de la suite dont la série génératrice est $\dfrac{1}{(X^4 + X + 1)^3(X^3 + X^2 + 1)}$?

Les polynômes $X^4 + X + 1$ et $X^3 + X^2 + 1$ sont primitifs. Leur ordre est respectivement 15 et 7. La période des suites produites est $\text{ppcm}(15 \times 2^2, 7) = 420$.

4.5 – M-Suites

Comme le vecteur nul est un état stable pour tout LFSR de dimension n, la période maximale d'une suite qu'il est capable de produire est $2^n - 1$. Dans ce cas, les suites ont des propriétés remarquables intéressantes pour le cryptographe.

DÉFINITION. *On appelle M–suite une suite produite par un LFSR de dimension n de période maximale égale à $2^n - 1$.*

On considère dans la suite un LFSR $\mathcal{R}$ de dimension n qui produit des M–suites et on note P son polynôme caractéristique.

Le polynôme P est irréductible et primitif. On admettra qu'il existe des polynômes primitifs de tout degré.

Le graphe d'état de $\mathcal{R}$ comprend deux cycles : un de taille 1 réduit au seul vecteur nul et l'autre, de taille $2^n - 1$, comprenant tous les vecteurs non nuls de dimension n. En conséquence, dans une M–suite, toutes les figures de n termes non identiquement nuls apparaissent une et une seule fois au cours de la période. Cette propriété confère à la M–suite de bonnes propriétés statistiques.

Pour les même raisons, deux M–suites produites par un même LFSR sont décalées l'une de l'autre.

La somme de deux M–suites différentes produites par $\mathcal{R}$ est une M–suite produite par $\mathcal{R}$. En effet,

$$\frac{Q}{\widetilde{P}} + \frac{Q'}{\widetilde{P}} = \frac{Q + Q'}{\widetilde{P}}.$$

Le polynôme $\widetilde{P}$ étant irréductible, $Q + Q'$ étant non nul et de degré $< n$, il est premier avec $\widetilde{P}$.

4.6 – Complexité linéaire

DÉFINITION. *Soit s une suite (finie ou périodique). La complexité linéaire de s est la taille du plus petit LFSR qui engendre s.*

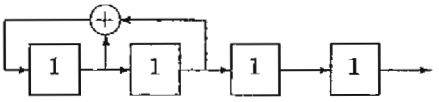
a) Suites ultimement périodiques

Soit $S = \dfrac{Q}{\widetilde{P}}$ la série génératrice d'une suite produite par un LFSR. Si Q et $\widetilde{P}$ sont premiers entre eux, on ne peut pas trouver de polynôme de connexion de degré plus faible pour produire la même suite. La complexité linéaire de la suite est alors égale à

$$L = \max\bigl(\deg(\widetilde{P}), \deg(Q) + 1\bigr)$$

Pour déterminer le registre de dimension minimale, calculer le pgcd de Q et $\widetilde{P}$ de manière à exprimer la série génératrice sous forme de fraction irréductible de polynômes.

Exemple. La suite de série génératrice $S = \dfrac{1 + X^2 + X^3}{1 + X + X^2} = 1 + X + X^2 + X^3 + X^5 + X^6 + \cdots$ est produite par le LFSR de dimension $\max(2, 3 + 1) = 4$ suivant :



b) Suites finies

Lorsque la suite est finie, une version spéciale de l'algorithme d'EUCLIDE permet de déterminer le LFSR de taille minimale qui la synthétise.

Soit une suite binaire $s = (s_i)_{i \in \{0,\ldots,N-1\}}$ de taille N Cet algorithme définit deux suites de polynômes $Q = (Q_i)_{i \geq 0}$ et $R = (R_i)_{i \geq 0}$. Ces suites sont définies par récurrence.

ALGORITHME D'EUCLIDE POUR RECONSTITUER UN LFSR

Entrée : Une suite binaire finie donnée par sa série génératrice S.

Initialisation : $Q_0 = X^N$; $Q_1 = S$; $R_0 = 0$ et $R_1 = 1$.

Itération : Supposons les suites (Q_i) et (R_i) définies jusqu'au rang i et que Q_i est non nul. Soit $Q_{i-1} = q_i Q_i + Q_{i+1}$ avec $\deg(Q_{i+1}) < \deg(Q_i)$ la division euclidienne de Q_{i-1} par Q_i. Cela définit le quotient q_i et le reste Q_{i+1}. On pose $R_{i+1} = R_i q_i + R_{i-1}$.

À chaque itération, si le polynôme R_i a un terme constant, il est le polynôme de connexion d'un LFSR dont la dimension est

$$L_i = \max\bigl(\deg(Q_i) + 1, \deg(R_i)\bigr)$$

La suite (Q_i) est une suite de polynômes dont les degrés décroissent strictement d'au moins une unité à chaque itération.

À l'inverse, la suite des degrés des R_i croît strictement. De plus, au moins un R_i sur deux a un terme constant.

Par conséquent, il existe nécessairement un rang k pour lequel R_k a un terme constant et la dimension L_k est minimale. Rappelons que si R_i n'a pas de terme constant, alors il ne peut pas être le polynôme de connexion d'un LFSR.

Cet indice k définit le LFSR le plus court produisant s. Soit $\ell = L_k$ la dimension de ce LFSR et d le degré de R_k Le polynôme caractéristique de ce LFSR est $P = X^{\ell - d}\widetilde{R}_k$. L'état initial est donné par les premiers termes de la suite.

Preuve de l'algorithme: À chaque itération, *i.e.* pour tout $i \in \{0, \ldots, k\}$, la relation suivante est satisfaite:

$$SR_i \equiv Q_i \mod X^N$$

Cette relation est vraie pour $i = 0$ et 1, puis se montre par récurrence en utilisant la définition des suites Q et R. Modulo X^N, on a:

$$SR_{i+1} = SR_i q_i + Q_i q_i + Q_{i-1} = Q_{i+1}$$

De plus, si R_{i-1} a un terme constant, mais R_i n'en a pas, alors $R_{i+1} = R_i q_i + R_{i-1}$ a nécessairement un terme constant, ce qui prouve, comme R_1 a un terme constant, qu'au moins un R_i sur deux a un terme constant.

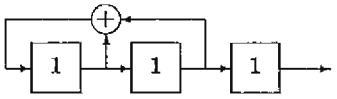
□

Exemple 1.

$s = 11101$ de taille 5.

$Q_0 = X^5$	$R_0 = 0$	
$Q_1 = X^4 + X^2 + X + 1$	$R_1 = 1$	$q_1 = X$
$Q_2 = X^3 + X^2 + X$	$R_2 = X$	$q_2 = X + 1$
$Q_3 = X^2 + 1$	$R_3 = X^2 + X + 1$	

À l'itération suivante, on aura $\deg(R_4) \geq 3$. On ne pourra pas obtenir de LFSR plus court.

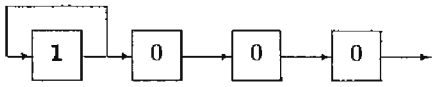


Exemple 2.

$s = 000111$ de taille 6.

$Q_0 = X^6$	$R_0 = 0$	
$Q_1 = X^5 + X^4 + X^3$	$R_1 = 1$	$q_1 = X + 1$
$Q_2 = X^3$	$R_2 = X + 1$	$q_2 = X^2 + X + 1$
$Q_3 = 0$		

Le LFSR le plus court est celui obtenu à la dernière itération de l'algorithme.

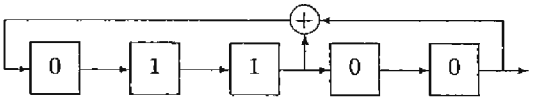


Exemple 3.

$s = 001101110$ de taille 9.

$Q_0 = X^9$	$R_0 = 0$	
$Q_1 = X^7 + X^6 + X^5 + X^3 + X^2$	$R_1 = 1$	$q_1 = X^2 + 1$
$Q_2 = X^6 + X^5 + X^3$	$R_2 = X^2 + X$	$q_2 = X$
$Q_3 = X^5 + X^4 + X^3 + X^2$	$R_3 = X^3 + X^2 + 1$	$q_3 = X$
$Q_4 = X^4$	$R_4 = X^4 + X^3 + X^2$	$q_4 = X + 1$
$Q_5 = X^3 + X^2$	$R_5 = X^5 + X^3 + 1$	$q_5 = X + 1$
$Q_6 = X^2$	$R_6 = X^6 + X^5 + X^2 + X + 1$	

La dimension minimale est celle obtenue à la cinquième itération.

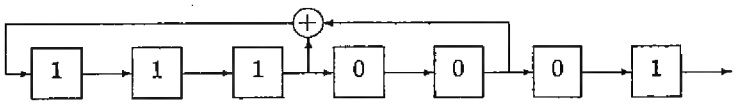


Exemple 4.

$s = 10001111100011$ de taille 14.

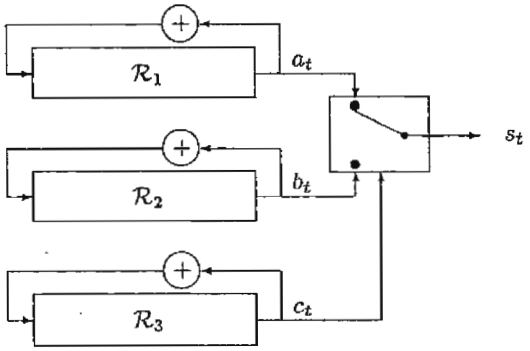
$Q_0 = X^{14}$	$R_0 = 0$	
$Q_1 = X^{13} + X^{12} + X^8 + X^7 + X^6 + X^5 + X^4 + 1$	$R_1 = 1$	$q_1 = X + 1$
$Q_2 = X^{12} + X^9 + X^4 + X + 1$	$R_2 = X + 1$	$q_2 = X + 1$
$Q_3 = X^{10} + X^9 + X^8 + X^7 + X^6 + X^2$	$R_3 = X^2$	$q_3 = X^2 + X$
$Q_4 = X^9 + X^7 + X^3 + X + 1$	$R_4 = X^4 + X^3 + X + 1$	$q_4 = X + 1$
$Q_5 = X^6 + X^4 + X^3 + 1$	$R_5 = X^5 + X^3 + 1$	$q_5 = X^3 + 1$

À l'itération suivante, au aura $\deg(R_6) = \deg(R_5) + \deg(q_5) = 8$, ce qui est supérieur à la dimension du LFSR obtenu à la cinquième itération, qui est égale à 7.



5.1 – Le générateur de GEFFE

Un LFSR seul est insuffisant pour les applications cryptogaphiques, car les suites produites sont aisément prédictibles. Leur complexité linéaire est insuffisante. Il est nécessaire d'introduire un composant non-linéaire.
Le générateur de GEFFE (1973) est illustré ci-dessous.



Il combine trois registres $\mathcal{R}_1$, $\mathcal{R}_2$ et $\mathcal{R}_3$. Ces trois registres produisent respectivement trois suites $(a_t)_{t\in\mathbb{N}}$, $(b_t)_{t\in\mathbb{N}}$ et $(c_t)_{t\in\mathbb{N}}$. Ces trois suites sont combinées par un multiplexeur pour produire une suite $(s_t)_{t\in\mathbb{N}}$. La valeur de s_t est définie par :

$$s_t = \begin{cases} a_t & \text{si } c_t = 0 \\ b_t & \text{si } c_t = 1 \end{cases}$$

Cette définition correspond à la fonction booléenne f de trois variables binaires dont la table est donnée par :

a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

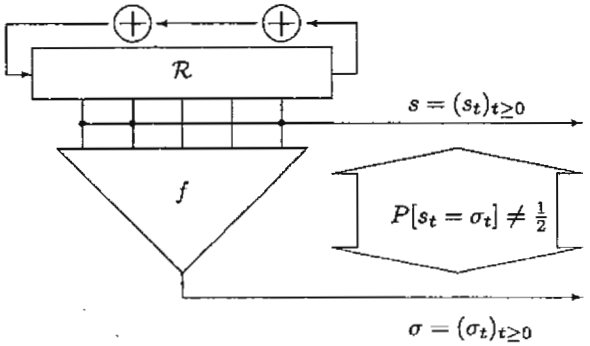
Cette fonction booléenne admet l'expression algébrique suivante :

$$f(a,b,c) = ac + b(1+c) = ac + bc + b$$

L'examen de la table de valeur de f conduit à constater que $f(a,b,c) = a$ dans 6 cas sur 8, $f(a,b,c) = b$ dans 6 cas sur 8 également et $f(a,b,c) = k$ dans 4 cas sur 8. On dit que la valeur de f est *corrélée* aux deux premières variables, mais n'est pas corrélée à la troisième. En conséquence, l'observation de s_t donne une information sur suites produites par les LFSR $\mathcal{R}_1$ et $\mathcal{R}_2$, mais pas sur $\mathcal{R}_3$.
La présence de ces corrélations, ainsi que la faible complexité de la fonction de combinaison rendent le générateur de GEFFE aujourd'hui insuffisamment sûr.

5.2 – Le registre filtré

Le registre filtré constitue un modèle classique de générateur pseudo-aléatoire cryptographique.
Il comprend un registre à décalage rebouclé, associé à une fonction booléenne non-linéaire de filtrage $f : \{0,1\}^k \to \{0,1\}$.
Lorsque le polynôme caractéristique du registre est de degré n, irréductible et primitif, il produit des suites binaires de période maximale égale à $2^n - 1$.

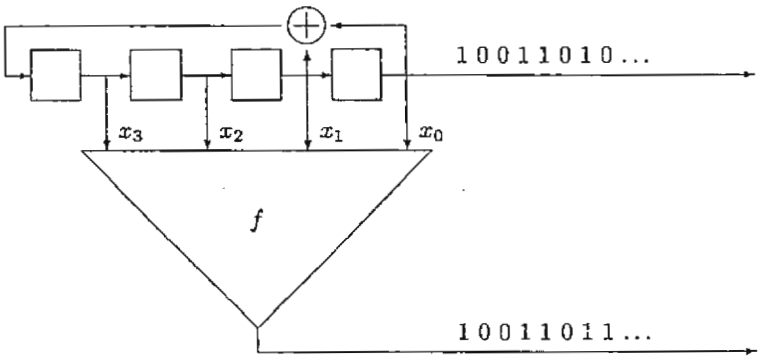


La dimension k du registre doit être suffisante pour produire des suites de très longue période et pour empêcher l'attaque qui consiste à tester systématiquement tous les états initiaux possibles ($k \geq 100$).
De plus, la fonction booléenne f de filtrage doit satisfaire les propriétés de non-linéarité suivantes :

- la corrélation entre $f(x)$ et n'importe quelle fonction linéaire $x \mapsto a \cdot x = a_1 x_1 + \cdots + a_k x_k$ doit être faible ;
- l'expression de f comme polynôme de k variables binaires doit avoir un degré élevé ;

Attaque par corrélation
Le registre filtré est sensible à l'attaque par corrélation.
Supposons que l'adversaire connaisse les N premiers termes de la suite $\sigma = (\sigma_t)_{t\geq 0}$ produite par le registre filtré.
On peut montrer que cette suite est nécessairement corrélée à une combinaison linéaire des états du registre.
Soit $s = (s_t)_{t\geq 0}$ la combinaison linéaire la plus corrélée à σ. Cela signifie que la probabilité que les termes s_t et σ_t soient égaux est notablement différente de $1/2$.
Soit k la taille du registre. Les N premiers termes de la suite s appartiennent à un sous-espace vectoriel de dimension k de $\{0,1\}^N$. Ils constituent donc les éléments d'un *code linéaire* $\mathcal{C}$ de longueur N et de dimension k.
La fonction de filtrage agit comme un canal de transmission produisant des erreurs. Le taux d'erreur est défini par la probabilité $P[s_t = \sigma_t]$.
L'adversaire observe les N premiers termes de la suite σ. S'il parvient à *décoder* cette observation, c'est-à-dire à trouver l'élément du code $\mathcal{C}$ le plus proche de σ, alors il aura déterminé la suite s correspondante avant filtrage par f.
Il est alors aisé de reconstituer l'état initial du registre et donc reconstituer la suite σ dans son ensemble.
Pour résister à cette attaque, la fonction de filtrage f doit présenter de très faibles corrélations entre la valeur $f(x)$ et toute combinaison linéaire $a \cdot x$. Une telle fonction est dite *fortement non-linéaire*.

Exemple
Considérons un LFSR dont le polynôme caractéristique $P(X)$ est $1 + X + X^4$, filtré par la fonction booléenne f définie par $f(x_0, x_1, x_2, x_3) = x_0 + x_1 x_2 x_3$. La valeur de f présente une très forte corrélation avec la variable x_0 puisque $f(x_0, x_1, x_2, x_3) = x_0$ sauf si $x_1 x_2 x_3 = 1$, c'est-à-dire dans 14 cas sur les 16 valeurs possibles de (x_0, x_1, x_2, x_3).



On suppose que l'adversaire a observé que les 8 premiers termes de la suite produite sont $\sigma = (1,0,0,1,1,0,1,1)$. Les 8 premiers termes produits par le LFSR dépendent des 4 composantes de l'état initial du registre. Ils appartiennent au code de longueur 8 et de dimension 4 défini par la matrice génératrice

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Le mot de code le plus proche de l'observation σ est $s = (1,0,0,1,1,0,1,0) = (1,0,0,1) \times G$. L'état initial du LFSR est donc $(1,0,0,1)$ et la suite produite par le registre filtré est maintenant parfaitement prédictible : $(1,0,0,1,1,0,1,1,1,0,1,1,0,0,0,1,0,\ldots)$.

5.3 – Forme algébrique normale

DÉFINITION. *Une fonction booléenne est une application* $\mathbb{F}_2^n \longrightarrow \mathbb{F}_2$.

L'ensemble $\mathcal{F}_n$ des fonctions booléennes sur $\mathbb{F}_2^n$ est un espace vectoriel de dimension 2^n sur $\mathbb{F}_2$ dont une base est constituée des indicatrices des singletons $\varphi_{\{x\}}$, pour $x \in \mathbb{F}_2^n$.
En effet, cette famille est génératrice puisque toute fonction booléenne f s'écrit

$$\forall x \in \mathbb{F}_2^n \quad f(x) = \sum_{t \in \mathbb{F}_2^n} f(t)\varphi_{\{t\}}(x). \tag{1}$$

Cette somme n'a qu'un seul terme non nul pour $t = x$.
Elle est également libre car une combinaison linéaire d'indicatrices de singletons est comme le le second membre de la relation (1). Cette égalité énonce que si elle est nulle, alors tous les coefficients sont nuls.
La fonction du générateur de GEFFE s'exprime dans cette base

$$f = \varphi_{\{011\}} + \varphi_{\{100\}} + \varphi_{\{110\}} + \varphi_{\{111\}}$$

Pour tout vecteur $u = (u_1, \ldots, u_n) \in \mathbb{F}_2^n$, le monôme m_u est par définition la fonction booléenne

$$m_u : \begin{array}{ccc} \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2 \\ x & \mapsto & x^u = x_1^{u_1} \cdots x_n^{u_n} \end{array}$$

Le *degré* du monôme m_u est égal au nombre de facteurs dans $x_1^{u_1} \cdots x_n^{u_n}$, c'est-à-dire au poids de u qui est le nombre de composantes égales à 1.

PROPOSITION. *L'ensemble des monômes constitue une base de l'ensemble $\mathcal{F}_n$ des fonctions booléennes.*

Preuve. Comme il existe 2^n monômes, il suffit de montrer que la famille des monômes est génératrice et pour cela, montrons que l'indicatrice $\phi_{\{a\}}$ du singleton $\{a\}$ peut s'exprimer en fonction des m_u. Or

$$\phi_{\{a\}}(x) = (x_1 + a_1 + 1) \cdots (x_n + a_n + 1).$$

En développant cette expression, on obtient bien une expression de $\phi_{\{a\}}$ en fonction des monômes. □

Par exemple dans $\mathbb{F}_2^3$, on a

$$\varphi_{\{101\}}(x_1, x_2, x_3) = x_1(x_2 + 1)x_3 = x_1x_2x_3 + x_1x_3.$$

La *forme algébrique normale* d'une fonction booléenne $f \in \mathcal{F}_n$ est sa décomposition dans la base des monômes.
Le *degré algébrique* de f est le degré maximal d'un monôme présent dans sa forme algébrique normale. Par exemple, $\varphi_{\{101\}}$ est de degré 3.
Les fonctions affines sont les fonctions de degré ≤ 1.

5.4 – Transformation de FOURIER

La transformation de FOURIER est l'outil qui permet d'évaluer les corrélations d'une fonction booléennes avec les fouctions affines.
Pour une fonction booléenne $f : \mathbb{F}_2^n \to \mathbb{F}_2$, la fonction *signe* associée, notée f_χ, prend respectivement les valeurs 1 et -1 au lieu des valeurs 0 et 1 :

$$f_\chi : \begin{array}{ccc} \mathbb{F}_2^n & \to & \{-1, 1\} \\ x & \mapsto & (-1)^{f(x)} = 1 - 2f(x) \end{array}$$

Sur l'espace vectoriel $\mathbb{F}_2^n$, le *produit intérieur* de deux vecteurs $x = (x_1, \ldots, x_n)$ et $(y_1, \ldots, y_n)$, noté $x \cdot y \in \mathbb{F}_2$ est par définition

$$x \cdot y = x_1y_1 + \cdots + x_ny_n$$

Pour tout élément $u \in \mathbb{F}_2^n$, la fonction $\lambda_u : x \mapsto u \cdot x$ est linéaire, car $u \cdot (x + y) = u \cdot x + u \cdot y$. Réciproquement, toute forme linéaire $\lambda : \mathbb{F}_2^n \to \mathbb{F}_2$ est déterminée par ses valeurs $(u_i)_{i \in \{1,\ldots,n\}}$ sur les vecteurs de la base canonique de $\mathbb{F}_2^n$. Elle est donc de la forme $\lambda : x \mapsto u \cdot x$ pour un certain vecteur $u \in \mathbb{F}_2^n$.

DÉFINITION. *Soit $f : \mathbb{F}_2^n \mapsto \mathbb{R}$ une application sur $\mathbb{F}_2^n$ et à valeurs réelles, la transformée de* FOURIER *de f est l'application*

$$\widehat{f} : \begin{array}{ccc} \mathbb{F}_2^n & \to & \mathbb{R} \\ u & \mapsto & \displaystyle\sum_{x \in \mathbb{F}_2^n} f(x)(-1)^{u \cdot x} \end{array}$$

La transformée de FOURIER de f est la fonction dont les composantes sont les valeurs de f dans ce qu'on appelle la base des fonctions de WALSH $\chi_u : x \mapsto (-1)^{u \cdot x}$ (voir exercices). En particulier la transformation $f \mapsto \widehat{f}$ est linéaire sur l'espace des fonctions booléennes, c'est à dire, pour tout $\lambda \in \mathbb{R}$ et pour toute fonction booléenne f et g, on a :

$$\widehat{\lambda f} = \lambda \widehat{f}$$
$$\widehat{f + g} = \widehat{f} + \widehat{g}$$

La transformée de WALSH d'une fonction booléenne $f : \mathbb{F}_2^n \to \mathbb{F}_2$ est par définition la transformée de FOURIER de sa fonction signe. Son expression est :

$$\begin{array}{rccc} & \mathbb{F}_2^n & \to & \mathbb{R} \\ \widehat{f_\chi} : & u & \mapsto & \displaystyle\sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+u\cdot x} \end{array}$$

La valeur en u de la transformée de WALSH de la fonction booléenne f exprime le nombre de fois où f est égale à la forme linéaire $\lambda_u : x \mapsto u \cdot x$, moins le nombre de fois où elle est différente. $\widehat{f_\chi}(u)$ représente une évaluation de la corrélation entre f et λ_u. En particulier, si $\widehat{f_\chi}(u) = 0$, alors f est *indépendante* de λ_u. Cela signifie que la connaissance de la valeur de $u \cdot x$ n'apporte aucune information sur la valeur de $f(x)$. Par exemple, $\widehat{f_\chi}(0) = 0$ signifie que f est équidistribuée.

Si pour tout vecteur $u \in \mathbb{F}_2^n$ de poids 1 (*i.e.* qui n'a qu'une seule composante non nulle) alors $\widehat{f_\chi}(u) = 0$, alors f est indépendante des variables x_i. On dit alors que f est *résistante aux corrélations à l'ordre 1*. Cela signifie que la connaissance de la valeur d'une quelconque des variables n'apporte aucune information quant à la valeur prise par f. C'est l'absence de cette propriété qui a permis la cryptanalyse du générateur de GEFFE.

Algorithme de transformée de FOURIER rapide

Une fonction réelle de n variables binaires peut s'exprimer à l'aide de deux sous-fonctions réelles de $n-1$ variables, notées f_0 et f_1 :

$$f(x_1, \ldots, x_n) = \begin{cases} f_0(x_1, \ldots, x_{n-1}) & \text{si } x_n = 0 \\ f_1(x_1, \ldots, x_{n-1}) & \text{si } x_n = 1 \end{cases}$$

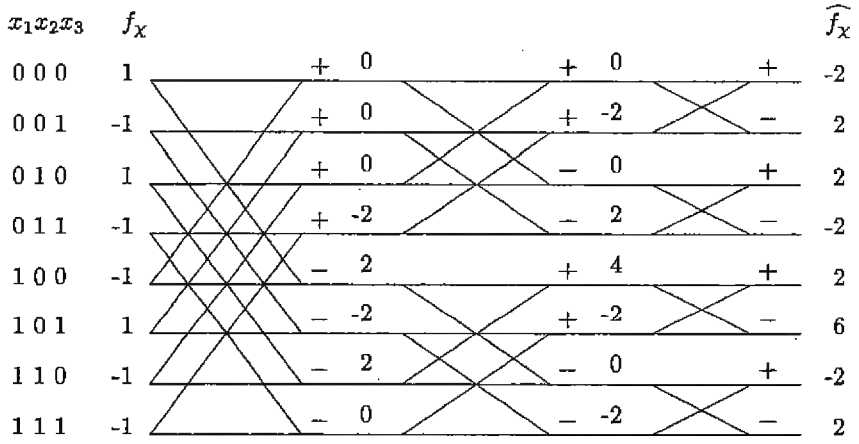
Cette décomposition est la décomposition de SHANNON de f. Ponr un vecteur $x = (x_1, \ldots, x_n)$ appartenant à $\mathbb{F}_2^n$, on note x' le vecteur de $\mathbb{F}_2^{n-1}$ constitué des $n-1$ premières composantes : $x' = (x_1, \ldots, x_{n-1})$. La décomposition de SHANNON s'exprime algébriqnement par

$$\begin{aligned} f(x) &= x_n f_1(x') + (1 - x_n) f_0(x') \\ &= x_n \big(f_1(x') - f_0(x')\big) + f_0(x') \end{aligned}$$

Cette décomposition est le fondement de l'algorithme rapide pour calculer les valeurs de la transformée de FOURIER de f. Pour un vecteur $x = (x_1, \ldots, x_n)$ appartenant à $\mathbb{F}_2^n$, on note x' le vecteur de $\mathbb{F}_2^{n-1}$ constitué des $n-1$ premières composantes : $x' = (x_1, \ldots, x_{n-1})$.

$$\begin{aligned} \widehat{f}(u) &= \sum_{x \in \mathbb{F}_2^n} \big(x_n f_1(x') + (1 - x_n) f_0(x')\big)(-1)^{u_n x_n + u' \cdot x'} \\ &= \sum_{x' \in \mathbb{F}_2^{n-1}} f_0(x')(-1)^{u' \cdot x'} + (-1)^{u_n} \sum_{x' \in \mathbb{F}_2^{n-1}} f_1(x')(-1)^{u' \cdot x'} \\ &= \sum_{x' \in \mathbb{F}_2^{n-1}} \Big(f_0(x') + (-1)^{u_n} f_1(x')\Big)(-1)^{u' \cdot x'} \\ &= \begin{cases} \widehat{f_0 + f_1}(u') & \text{si } u_n = 0 \\ \widehat{f_0 - f_1}(u') & \text{si } u_n = 1 \end{cases} \end{aligned}$$

Cette formule exprime la transformée de FOURIER de f en fonction des transformées de FOURIER de $f_0 + f_1$ et de $f_0 - f_1$. La première étape consiste donc à calculer ces fonctions. En continuant le même procédé avec deux fonctions, cela conduit à l'algorithme rapide illustré par le schéma qui suit pour l'exemple de fonction signe de la fonction booléenne de 3 variables $(x_1, x_2, x_3) \mapsto x_1 + x_3 + x_1 x_2 x_3$



L'examen des valeurs de $\widehat{f_\chi}$ montre que $\widehat{f_\chi}(1,0,1) = 6$ et correspond à la plus forte corrélation entre f et la forme linéaire $x_1 + x_3$. En effet, $f(x_1, x_2, x_3) = x_1 + x_3$ dans 7 cas sur 8, alors que l'égalité a lieu dans 5 cas sur 8 pour les autres formes linéaires.

6 – Le chiffrement par blocs

6.1 Principes de conception

Une primitive de calcul par bloc qui opère sur des blocs de n bits est la donnée d'une famille de bijections $(T_k)_{k\in\mathcal{K}}$, indexée par la clé. Pour tout $k \in \mathcal{K}$, la fonction T_k est une bijection $\{0,1\}^n \mapsto \{0,1\}^n$. Cette famille doit satisfaire des critères de performance et de sécurité.
La conception d'une primitive de chiffrement par bloc moderne repose sur les principes de diffusion et confusion énoncés par Shannon (1949).

Diffusion. Une modification minime de la clé ou du clair doit se répercuter sur l'ensemble du cryptogramme. La modification d'une seule composante binaire de la clé doit changer chaque composante du cryptogramme avec probabilité 1/2. Cette propriété implique que le cryptogramme n'est pas statistiquement distinguable de données aléatoires. Elle est généralement atteinte avec un calcul de transformations linéaires.

Confusion. La dépendance du cryptogramme vis à vis du clair et de la clé soit être suffisamment complexe pour empêcher de dégager une structure sur la fonction de calcul qui permettrait de la distinguer d'une fonction aléatoire. Cette propriété implique en particulier qu'il est difficile de retrouver la clé à partir de l'observation d'un clair et de son cryptogramme. La propriété de confusion est atteinte avec des fonctions non linéaires dont la valeur est consultée dans une table. Ces fonctions s'appellent des *boîte de substitution* ou *boîte-S*.

Indistinguabilité Ce principe, plus récent, énonce que pour être sure, une primitive de chiffrement par bloc ne doit pas être distinguable d'une permutation aléatoire. Cette notion est formalisée par un jeu auquel se livre l'adversaire.
Un oracle à qui l'adversaire peut formuler des requêtes de chiffrement contient,

- soit une instance aléatoire T_k d'une fonction de chiffrement, obtenue par tirage aléatoire de la clé k;
- soit une permutation aléatoire sur $\{0,1\}^n$ obtenue par tirage aléatoire d'une permutation parmi les $2^n!$ possibles.

Ces deux événements sont équiprobables.
L'adversaire peut demander le calcul de toute image appartenant à $\{0,1\}^n$. S'il peut deviner ce que contient l'oracle entre ces deux possibilités avec une probabilité de bonne réponse différente de 1/2, on dit qu'il a un avantage.
L'avantage de l'adversaire dépend de sa capacité de calcul et du nombre de requêtes qu'il a besoin de formuler avant de proposer sa réponse. En effet, une stratégie possible est d'essayer toutes les clés. Si aucune clé ne convient, alors l'oracle contient avec certitude une fonction aléatoire. Si une clé convient, d'autres requêtes permettent de confirmer ou d'infirmer l'autre hypothèse.

6.2. Architecture d'un calcul par bloc

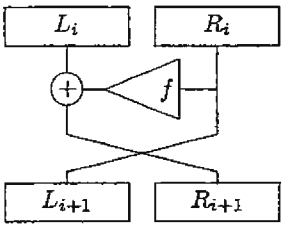
Pour atteindre les propriétés requises définies au paragraphe précédent, un calcul par bloc itère des calculs plus simples sur plusieurs tours. Chaque tour utilise une *sous-clé* qui est dérivée de la clé principale. Un calcul par bloc est constitué de:

- une fonction de tour t_i qui opère sur un bloc en fonction d'une sous-clé.
- un dispositif de *génération des sous-clés* qui calcule la sous-clé de chaque tour à partir de la clé principale.

Le nombre de tours h d'un calcul par bloc doit être suffisant pour assurer la sécurité, mais pas trop élevé pour ne pas pénaliser la rapidité du calcul.
La fonction de calcul par bloc T_k est la composée de h fonctions de tour plus simples: $T_k = t_h \circ \cdots \circ t_1$.
Chaque fonction de tour t_i est paramétrée par une sous-clé sk_i qui est dérivée de la clé k.
La principale difficulté pour le concepteur est de réaliser, pour chaque fonction de tour t_i, une bijection qui opère sur un ensemble très grand. Les paragraphes qui suivent présentent deux solutions.

6.3. Schéma de FEISTEL (1973)

Le paramètre de la fonction de tour t_i est partagé en deux composantes L_i (*left*) et R_i (*right*) dont la taille est la moitié de celle du bloc. La valeur est (L_{i+1}, R_{i+1}) et sert de paramètre pour le tour suivant.



$$\begin{cases} L_{i+1} &= R_i \\ R_{i+1} &= L_i + f(R_i) \end{cases}$$

L'opération + est en général l'addition modulo 2 terme à terme.

Quelle que soit la fonction f, ce schéma est inversible : $\begin{cases} L_i &= R_{i+1} - f(L_{i+1}) \\ R_i &= L_{i+1} \end{cases}$

La fonction inverse est également un schéma de FEISTEL, quitte à échanger les parties gauche et droite. Cela permet de réaliser le déchiffrement à partir d'une fonction similaire.

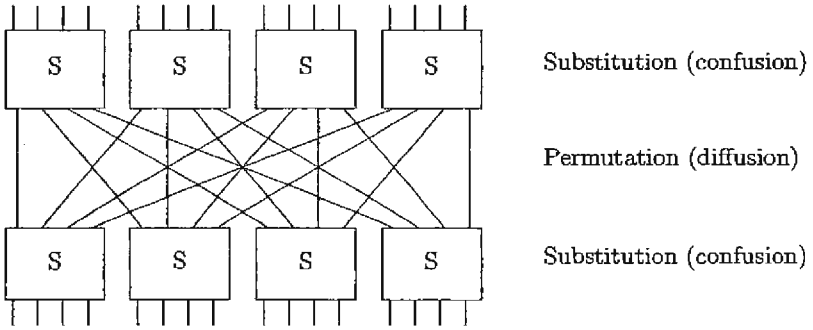
On peut montrer qu'il faut effectuer au moins trois tours pour qu'on ne puisse pas distinguer un schéma de FEISTEL d'une fonction aléatoire (voir TD).

Le DES (§ 6.5) est un exemple de schéma de FEISTEL.

Exemple. La taille des blocs est $n = 6$. Il est partagé en deux sons-blocs de taille 3. Supposons f : $\{0,1\}^3 \to \{0,1\}^3$ définie par $f(x_1, x_2, x_3) = (x_1x_2, x_2x_3, x_1 + x_3)$. Si $L_i = (0,1,0)$ et $R_i = (1,1,0)$, alors $f(R_i) = f(1,1,0) = (1,0,1)$, $L_{i+1} = (1,1,0)$ et $R_{i+1} = (0,1,0) + (1,0,1) = (1,1,1)$.

6.4 Réseaux substitutions-permutations

Un réseau de substitutions-permutations est constitué de transformations bijectives simples, facilement réalisables qui sont assemblées entre elles de manière à obtenir une transformation globale complexe.



Chaque substitution S doit être inversible. Le déchiffrement utilise les substitutions inverses S^{-1}.

Les substitutions sont des bijection non linéaires définies généralement par une table de valeurs. Elles assurent la propriété de confusion.

Les permutations réalisent l'interconnexion entre les substitutions. Elles permettent de s'assurer que le cryptogramme obtenu dépend bien de toutes les composantes du clair. Elles assurent la propriété de diffusion.

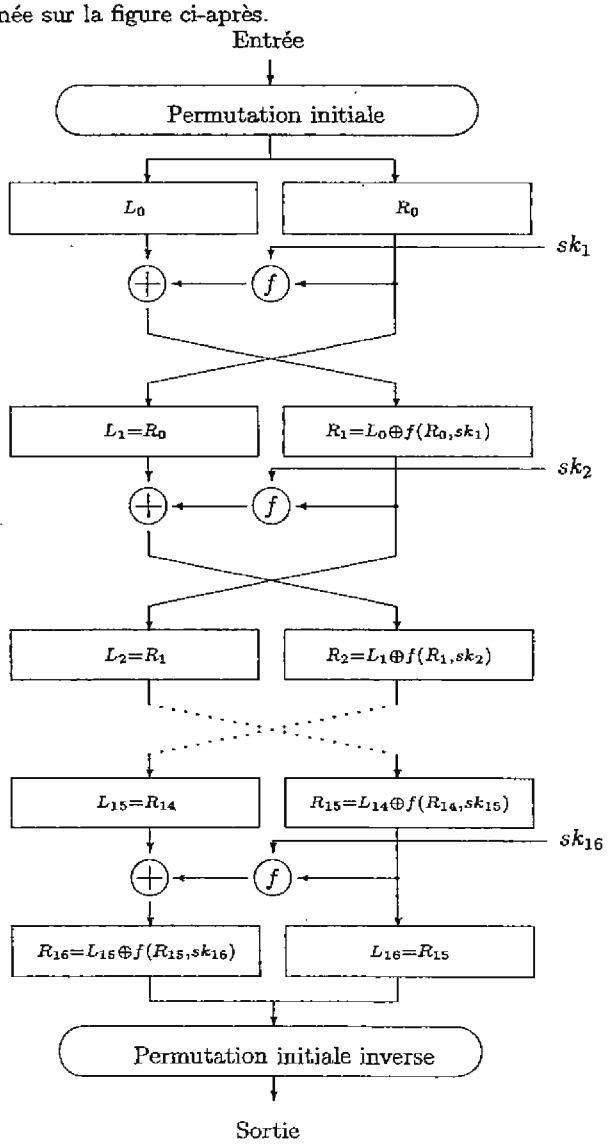
Dans les systèmes les plus récents comme l' AES (voir définition de l'AES au § 6.8), la permutation est souvent remplacée par une transformation linéaire plus élaborée.

6.5 Le DES

Le DES a été développé par IBM en 1977 sous contrôle de la NSA (*National Security Agency* américaine). Il a été le standard pour le chiffrement civil jusqu'en 2000, date où il a été remplacé par l'AES. Il reste cependant toujours largement utilisé. Aucune faiblesse vraiment critique n'a été découverte et la seule attaque envisageable en pratique reste la recherche exhaustive de la clé.

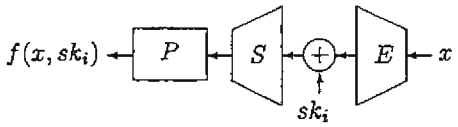
6.5.1. Architecture générale du DES.

Le DES est un schéma de FEISTEL à 16 tours, précédé et suivi d'une permutation des composantes binaires du bloc de données. Il opère sur des blocs de taille 64 avec une clé de 56 bits. L'architecture générale est donnée sur la figure ci-après.



La fonction f de chaque tour agit sur des mots de taille 32. Pour tout $x \in \{0,1\}^{32}$,

$$f(x, sk_i) = P(S(E(x) + sk_i))$$



- E est une expansion $\{0,1\}^{32} \to \{0,1\}^{48}$ qui consiste à dupliquer certaines composantes.
- sk_i est la sous-clé du i^e tour qui comprend 48 symboles binaires extraits de la clé, spécifiques à chaque tour.
- S est la juxtaposition de 8 boîtes de substitutions (boîtes-S) $\{0,1\}^6 \to \{0,1\}^4$, données par la table de leurs 64 valeurs dans $\{0,\ldots,15\}$. Elles assurent la propriété de confusion.
- P est une permutation des 32 composantes de sortie des boîtes-S. Elle assure la propriété de diffusion.

6.5.2. Génération des sous-clés. La clé de 56 bits est constituée de 8 mots de 7 bits. Chaque mot est complétée avec un symbole de parité de telle sorte que le poids binaire obtenu soit impair. Puis elle subit une permutation des termes. Elle est ensuite partagée en deux composantes C et D, de 32 bits chacune. A chaque tour, les deux composantes subissent une rotation. La sous-clé du tour courant est définie comme une extraction de 48 bits de ces deux composantes.

6.5.3. Déchiffrement. Les parties gauche et droite du dernier tour ne sont pas permutées, ce qui assure que la fonction de déchiffrement est identique à la fonction de chiffrement, mais avec les sous-clés prises dans l'ordre inverse.

6.5.4. Triple DES. La taille de la clé, égale à 56 bits, était suffisante lors de la création du DES en 1977. Elle est aujourd'hui trop faible. L'attaque par recherche exhaustive sur les 2^{56} clés possible est maintenant envisageable. Pour pallier cette faiblesse, la NSA a proposé le *triple*-DES. Il utilise deux clés DES k_1 et k_2, soient 112 bits. La fonction de chiffrement est

$$m \mapsto \mathrm{DES}_{k_1} \circ \mathrm{DES}_{k_2}^{-1} \circ \mathrm{DES}_{k_1}(m)$$

Remarque lorsque $k_1 = k_2$ cette fonction équivaut à un simple DES. Ceci permet la compatibilité avec les correspondants qui ne disposent que du simple DES.

6.6. La cryptanalyse différentielle du DES (Biham et Shamir, 1990)

C'est une attaque à clairs choisis. Elle requiert 2^{47} couples (clair, cryptogramme), ce qui la rend pratiquement inexploitable. On lui préfère l'attaque exhaustive. Cependant, cette attaque est applicable à tout chiffrement par bloc. Elle met en évidence les faiblesses et conduit à des critères de conception.

6.6.1. Principe. L'attaque différentielle cherche à reconstituer la sous-clé du dernier tour. Celle-ci étant connue, les autres sous-clés sont retrouvées de la même façon. Lorsqu'un nombre suffisant de sous-clés est connu, le reliquat peut être retrouvé par recherche exhaustive. On chiffre des messages de différence connue et on observe les cryptogrammes. Pour un bloc de clair $x = x_0$, notons $x_1, \ldots, x_{15}, x_{16} = y$ les données produites par les tours successifs.

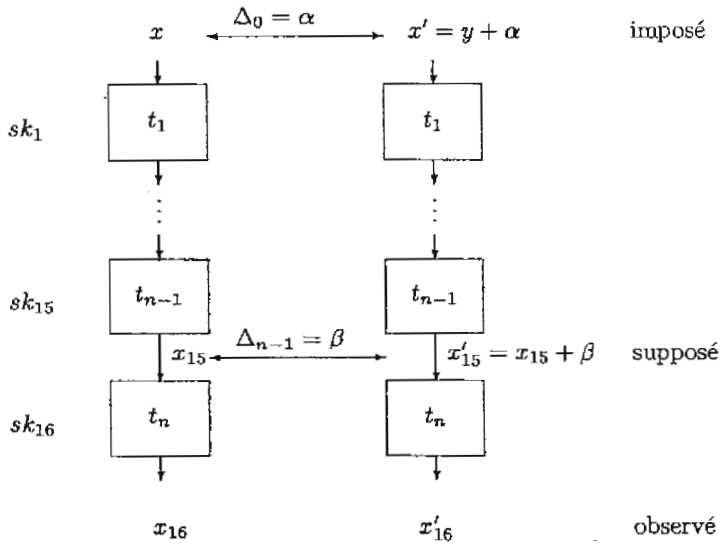
6.6.2. Caractéristique différentielle. Soient x et x' deux clairs et soit $\delta_0 = x - x'$ leur différence. On s'intéresse à la différence obtenue à l'avant dernier tour, $\delta_{15} = x_{15} - x'_{15}$. Cette qnantité inconnue dépend de la clé, et peut être considérée comme une variable aléatoire. Cependant, pour tout couple $(\alpha, \beta) \in \mathbb{F}_2^{64} \times \mathbb{F}_2^{64}$, la probabilité $P_{\alpha,\beta} = P\big[\delta_{15} = \beta / \delta_0 = \alpha\big]$ d'obtenir une différence β à l'avant dernier tour, sachant que la différence entre les clairs est α, n'est pas uniforme. Une *caractéristique différentielle* est un couple (α, β) tel que la probabilité $P_{\alpha,\beta}$ soit notablement supérieure à la probabilité uniforme, égale à $1/2^{64}$.
Trouver une caractéristique différentielle est un travail délicat et demande un examen attentif et détaillé de tous les éléments de la fonction de calcul. Le principe est de composer des caractéristiques optimales de chaque tour. Toutefois, ce travail n'est à effectuer qu'une seule fois.

6.6.3. Extraction de la sous-clé du dernier tour. Supposons qu'on ait trouvé que les données à l'entrée du dernier tour diffèrent de $x_{15} - x'_{15} = \beta \in \mathbb{F}_2^{64}$ et qu'on observe les cryptogrammes x_{16} et x'_{16}. Une recherche exhaustive sur les sous-clés du dernier tour permet de déterminer quelles sont les sous-clés compatibles avec la différence estimée β du 15^e tour et les observations x_{16} et x'_{16}. Pour cela, on fait opérer la transformation inverse du 16^e tour aux observations x_{16} et x'_{16}. Pour chaque sous-clé du 16^e tour candidate sk_{16}, on obtient $y_{15}(sk_{16})$ et $y'_{15}(sk_{16})$. Les sous-clés compatibles sont celles pour lesquelles

$$y_{15}(sk_{16}) - y'_{15}(sk_{16}) = \beta$$

Les valeurs de la sous-clé à l'entrée de chaque boîte-S peuvent être recherchées indépendamment les unes des autres. Le coût de cette recherche exhaustive est donc de $8 \times 2^6 = 512$ essais.

La cryptanalyse différentielle peut être illustrée ainsi :



6.6.4. Algorithme. Étant donnée une caractéristique différentielle (α, β) la meilleure possible, l'attaque consiste à itérer les étapes suivantes :

1. choisir un message aléatoire m, chiffrer m et $m + \alpha$. Soit γ la différence observée entre les cryptogrammes ;
2. compter les valeurs possibles de la sous-clé sk_{16} du dernier tour compatibles avec β et l'observation (x_{16}, x'_{16}) comme indiqué en 6.6.3 ;
3. répéter les étapes 1 et 2 jusqu'à ce qu'une valeur de sous-clé sk_{16} ait été comptée significativement plus que les autres.

Preuve de l'algorithme : Dans tous les cas où la différence δ_{15} à l'avant dernier tour n'est pas β, les sous-clés trouvées à l'étape 2 sont aléatoires. Si au contraire $\delta_{15} = \beta$, la bonne sous-clé est comptée une fois de plus. Comme $P_{\alpha,\beta}$ est une caractéristique différentielle, cette situation survient plus souvent. La bonne sous-clé est donc comptée plus que les autres.

6.7. La cryptanalyse linéaire du DES (MATSUI, 1993)

C'est une attaque à clairs connus. La cryptanalyse des 16 tours du DES nécessite la connaissance de 2^{43} couples (clair, cryptogramme). Comme l'attaque différentielle, elle est applicable à d'autres algorithmes et permet de dégager des critères de conception.

Principe. L'attaque linéaire retrouve une équation linéaire satisfaite par la clé. Pour retrouver la clé dans son ensemble, il faut l'appliquer plusieurs fois, par exemple 20 fois. Cela permet d'exprimer 20 composantes binaires de la clé en fonction des 36 autres qui, eux sont finalement déterminés par recherche exhaustive.

Approximation linéaire. Soit $K \in \mathbb{F}_2^{56}$ une clé aléatoire et $X \in \mathbb{F}_2^{64}$ un clair aléatoire. Soit $Y = \text{DES}_K(X)$ le cryptogramme obtenu. Une approximation linéaire du DES est un triplet $(\alpha, \beta, \gamma) \in \mathbb{F}_2^{64} \times \mathbb{F}_2^{64} \times \mathbb{F}_2^{56}$ tel que la probabilité $P[\alpha \cdot X + \beta \cdot Y = \gamma \cdot K]$ est nettement différente de la probabilité uniforme qui est égale à $1/2$. Un tel triplet s'obtient en composant les approximations linéaires optimales de chaque tour. Pour conduire l'attaque linéaire, il faut plusieurs triplets (α, β, γ).

Algorithme. L'attaque consiste à itérer les étapes suivantes jusqu'à obtenir assez d'équations linéaires satisfaites par la clé pour la reconstituer :

1. Choisir une approximation linéaire (α, β, γ) telle que $P[\alpha \cdot X + \beta \cdot Y + \gamma \cdot K = \varepsilon] = p > 1/2$.
2. Pour N couples (clair= x, cryptogramme= y) qui correspondent à la clé K cherchée, soit T le nombre de couples (x, y) tels que $\alpha \cdot x + \beta \cdot y = 0$. Si $T > N/2$, alors on estime que $\gamma \cdot K$ vaut ε et sinon, on estime que $\gamma \cdot K$ vaut $1 - \varepsilon$.

L'attaque sera d'autant plus efficace que la probabilité p sera élevée. MATSUI a montré que si $N \simeq \frac{1}{(p-\frac{1}{2})^2}$, alors le taux de succès est de 97,7%.

6.8 l'AES

L'algorithme AES a fait l'objet d'un appel d'offre international datant de 1997. Après l'évaluation publique de 15 candidats, le NIST a retenu en 2000 l'algorithme RIJNDAEL, dû aux chercheurs belges RIJMEN et DAEMEN. Il est devenu le nouveau standard pour le chiffrement civil.

Il s'agit d'un réseau de substitution-permutation. Il peut opérer sur des blocs de taille 128, 196 ou 256, avec une clé de 128, 196 ou 256 symboles binaires. La version qui opère sur des blocs de taille 128 itère un calcul élémentaire sur 10 tours.

6.8.1. Le corps $\mathbb{F}_{256}$. Dans l'algorithme AES un octet est interprété comme un élément du corps $\mathbb{F}_{256}$ à 256 éléments, dans sa représentation $\mathbb{F}_2[X]/(1 + X + X^3 + X^4 + X^8)$.

Exemple. L'octet dont l'écriture hexadécimale est `0x4D` s'écrit, en numération binaire, `01001101`. Il correspond au polynôme $1 + X^2 + X^3 + X^6$. Pour la définition de l'AES, la convention est de considérer les termes de poids faible de la numération binaire comme correspondant aux termes de plus bas degré du polynôme qui lui correspond.

6.8.2. La fonction de diffusion. En complément d'une simple permutation, la fonction de diffusion utilise également une transformation linéaire. Les mots de 32 bits sont interprétés comme des vecteurs colonne X de dimension 4 sur $\mathbb{F}_{256}$ et sont multipliés par la matrice M pour obtenir $M \times X$ avec

$$M = \begin{pmatrix} X & 1+X & 1 & 1 \\ 1 & X & 1+X & 1 \\ 1 & 1 & X & 1+X \\ 1+X & 1 & 1 & X \end{pmatrix}.$$

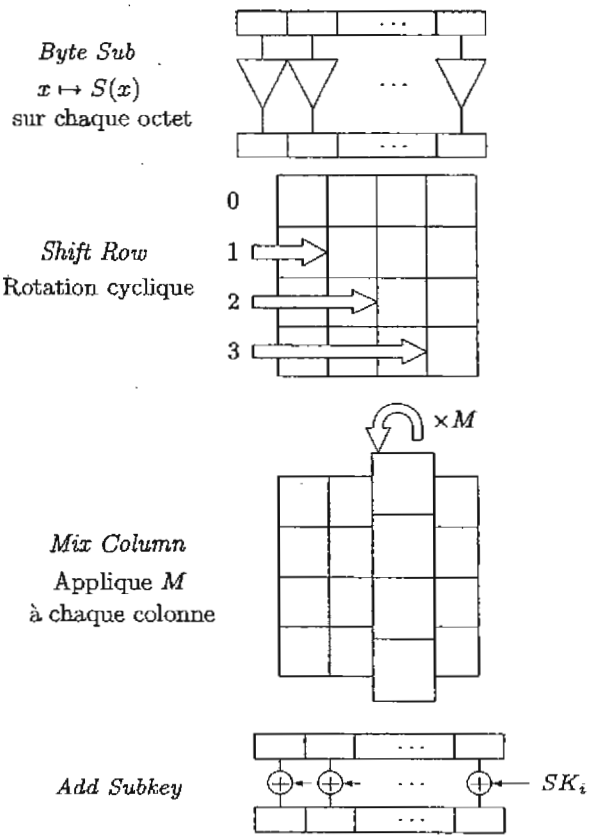
Exemple. Le mot de taille 32 représenté par `0x1234a9c5` en notation hexadécimale correspond au vecteur $\begin{pmatrix} 1+X^2+X^6+X^7 \\ 1+X^3+X^5+X^7 \\ X^2+X^4+X^5 \\ X+X^4 \end{pmatrix}$. La première composante de ce vecteur correspond aux 8 symboles de poids faible `0xc5`. Son image par M est $\begin{pmatrix} 1+X+X^2+X^4+X^6 \\ X^3+X^6+X^7 \\ X+X^4+X^5 \\ 1+X+X^2+X^3+X^5+X^6+X^7 \end{pmatrix}$, ce qui correspond à la notation hexadécimale `0xef32c857`.

6.8.3. La boîte-S. La fonction qui assure la confusion dans l'AES est définie sur $\mathbb{F}_{256}$. Elle est la composée de $x \mapsto x^{254}$ et d'une fonction affine. la première transformation correspond à l'inversion dans le corps $\mathbb{F}_{256}$, prolongée en 0 par 0. Cette fonction inversible possède de bonnes propriétés de non linéarité, ce qui assure à l'AES une bonne résistance aux cryptanalyses linéaires et différentielles (voir § 6.6 et § 6.7).

6.8.4. La fonction de tour. La fonction de tour agit sur une valeur de 128 symboles binaires, constituée de 16 octets arrangés en un tableau de 4×4 : $x = (x_{ij})_{i,j \in \{1,2,3,4\}}$, où $x_{ij} \in \mathbb{F}_{256}$.

$$x \mapsto MC \circ SR \circ BS(x) + SK_i$$

- BS (*Byte Substitution*) consiste à appliquer la boîte-S à chaque composante de x ;
- SR (*Shift Row*) opère une rotation cyclique sur les lignes de x, différente pour chaque ligne.
- MC (*Mix column*) transforme les colonnes de x en appliquant la fonction de diffusion M. Cette transformation est absente du dernier tour.
- *Add Subkey* ajoute, modulo 2, composante à composante, la sous-clé du i^{e} tour.



6.8.5. La génération des sous-clés. Les sous-clés SK_i de chaque tour sont dérivées à partir de la clé mère par un registre à décalages opérant sur des mots de 32 symboles binaires, avec un rebouclage non-linéaire qui utilise la boîte-S.

6.8.6. Le déchiffrement. La fonction de déchiffrement réalise la fonction inverse à chaque étape des transformations. Les sous-clés sont prises dans l'ordre inverse et chaque transformation de tour est réalisée dans l'ordre inverse.

7 – Le protocole Diffie-Hellman

Le protocole Diffie-Hellman est le premier système à clés publiques qui a été publié (1976). Il a constitué un véritable progrès. Il a permis de simplifier la distribution des clés sur les réseaux ouverts, comme l'Internet, et d'assurer les nouveaux services de sécurité comme l'authentification et la non répudiation.
Kerckoffs a énoncé que la sécurité de la cryptographie ne devait reposer que sur le secret de la clé. Le procédé de chiffrement doit pouvoir, lui, être rendu public. Les procédés à clés publiques vont encore plus loin dans ce sens, en autorisant une partie de la clé à être publiée.
Les systèmes à clés publiques reposent sur les notions de *fonction à sens unique* et de *fonction à sens unique avec porte dérobée*. Ces notions sont construites à partir de problèmes mathématiques réputés difficiles, comme la factorisation et le calcul du logarithme discret.

7.1. Le corps $\mathbb{Z}/p\mathbb{Z}$

Pour un entier $n \geq 2$, l'ensemble $\{0, 1, \ldots, n-1\}$, muni de l'addition et de la multiplication modulo p est un anneau commutatif unitaire noté $\mathbb{Z}/n\mathbb{Z}$. Il n'est pas toujours intègre comme le montre $2 \times 2 = 0$ dans $\mathbb{Z}/4\mathbb{Z}$.
On rappelle que si p est premier, alors tout élément x non nul de $\mathbb{Z}/p\mathbb{Z}$ est inversible. L'ensemble $\mathbb{Z}/p\mathbb{Z}$ est donc un corps.

Théorème.(Petit théorème de Fermat)
Soit p un nombre premier. Si a est premier avec p, alors

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

Preuve. Si p est premier et k est tel que $1 \leq k < p$, le coefficient binomial

$$\binom{p}{k} = \frac{p(p-1)\cdots(p-k+1)}{1 \times 2 \times \cdots k}$$

est un entier multiple de p. En effet, comme p est premier, il n'est divisible par aucun des entiers $2, 3, \ldots k$. Dans le développement de $(a+1)^p$, les seuls coefficients non multiples de p sont ceux de a^p et de 1^p, d'où $(a+1)^p \equiv a^p + 1 \pmod{p}$. Cette relation permet de prouver par récurrence sur a que $\forall a \in \mathbb{N} \quad a^p \equiv a \pmod{p}$. Lorsque a est premier avec p, la division par a conduit au résultat. □

L'inverse de x est donc x^{p-2}.

Définition. *Soit p un nombre premier et x un élément non nul de $\mathbb{Z}/p\mathbb{Z}$. On appelle ordre de x le plus petit entier non nul k pour lequel $x^k = 1$. On le note $o(x)$.*

Lorsque p n'est pas premier, un tel entier k n'existe pas toujours, par exemple, dans $\mathbb{Z}/10\mathbb{Z}$, aucune puissance de 2 ne vaut 1. Lorsque p est premier, le petit théorème de Fermat assure que l'ordre existe bel et bien.

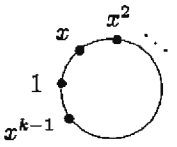
Proposition. *Soit p un nombre premier, x un élément de $\mathbb{Z}/p\mathbb{Z}$ et e l'ordre de x.*

$$\forall n \in \mathbb{N} \quad x^n = 1 \Longleftrightarrow n \text{ es multiple de } e$$

Preuve. $\Rightarrow$: Si $n = ke$, alors $x^n = x^{ke} = (x^e)^k = 1^k = 1$.

$\Leftarrow$: réciproquement, supposons $x^n = 1$. Soit $n = qe + r$ avec $0 \leq r < e$ la division euclidienne de n par e. $1 = x^n = x^{eq+r} = x^r$. En raison de la minimalité de e et comme $r < e$, nécessairement $r = 0$ et n est multiple de e. □

L'ordre d'un élément $x \neq 0$ de $\mathbb{Z}/p\mathbb{Z}$ est donc toujours un diviseur de $p-1$. L'*orbite* de x l'ensemble $\mathcal{O}(x) = \{1, x, x^2, \ldots, x^{k-1}\}$ des puissances successives de x.



Dans ce qui suit, p un nombre premier. Le groupe multiplicatif de $\mathbb{Z}/p\mathbb{Z}$ est engendré par un élément. Pour le démontrer, montrons tout d'abord deux lemmes.

LEMME 1. *Soit x et y deux éléments non nuls de $\mathbb{Z}/p\mathbb{Z}$, d'ordres respectifs m et n. Si m et n sont premiers entre eux, alors le produit xy a pour ordre mn.*

Preuve. Soit r l'ordre de xy. Comme $1 = (xy)^{rn} = x^{rn}$, on a m divise rn. Comme m et n sont premiers entre eux, m divise r. Symétriquement, n divise r, donc mn divise r. Comme par ailleurs $(xy)^{mn} = 1$, on a r divise mn, d'où $r = mn$. □

LEMME 2. *Soit r l'ordre maximal d'un élément de $\mathbb{Z}/p\mathbb{Z}$. L'ordre de tout élément non nul de $\mathbb{Z}/p\mathbb{Z}$ divise r.*

Preuve. Soit g un élément d'ordre r et x un élément d'ordre n. Élever x à une puissance convenable pour obtenir un élément y d'ordre m premier avec r. Si n ne divise pas r, alors $m > 1$ et d'après le lemme 1, le produit gy a pour ordre rm qui est strictement supérieur à r. Ceci contredit l'hypothèse de maximalité de r. □

THÉORÈME. *Si p est premier, $(\mathbb{Z}/p\mathbb{Z}^*, \times)$ est un groupe cyclique.*

Ceci signifie qu'il existe un élément générateur g, appelé *élément primitif*, tel que $\mathbb{Z}/p\mathbb{Z}^* = \{1, g, g^2, \ldots, g^{p-2}\}$.

Preuve. Soit g un élément d'ordre maximal r. On a bien sûr $r \leq p-1$. D'autre part, d'après le lemme 2, tout élément de $\mathbb{Z}/p\mathbb{Z}^*$ est racine du polynôme $X^r - 1$. Comme ce polynôme est de degré r, il a au plus r racines, d'où $p-1 \leq r$. Finalement, $r = p-1$ et g est un générateur de $(\mathbb{Z}/p\mathbb{Z}^*, \times)$. □

Un élément primitif est finalement un élément d'ordre $p-1$. Pour vérifier que g est primitif, il suffit de vérifier que pour aucun diviseur strict k de $p-1$ on a $g^k = 1$. On sait qu'il en existe toujours, mais on ne sait pas en trouver autrement que par tirage au hasard et vérification.

7.2. Fonctions à sens unique

Une fonction injective $f : E \to F$ est dite *à sens unique* si :

1. étant donné $x \in E$, la valeur y de $f(x)$ se calcule efficacement, c'est-à-dire, il existe un algorithme polynomial en la taille de x qui calcule y et
2. pour la plupart des y dans $f(E)$, il n'est pas possible de trouver un antécédent x dans E qui vérifie $f(x) = y$, sauf
   - à exécuter un nombre d'opérations qui dépasse les capacités de calcul actuelles, ou
   - à avoir une chance sur laquelle il est déraisonnable de compter.

En particulier, la taille de l'ensemble E doit être suffisante pour interdire la recherche de x par essais systématiques (recherche exhaustive).

En l'absence de résultats théoriques prouvés, la notion de fonction à sens unique est expérimentale et marquée dans le temps. On utilise en pratique des *problèmes difficiles* pour lesquels on ne connaît pas à ce jour d'algorithme de résolution polynomial, et sans que ce fait soit prouvé. Certaines fonctions à sens unique aujourd'hui peuvent ne plus l'être demain du fait des progrès mathématiques, algorithmiques ou technologiques.

7.3. La négociation de clé DIFFIE-HELLMAN (1976).

Ce protocole permet à deux personnes qui ne disposent que d'une ligne non sécurisée pour échanger des informations, de convenir d'une information secrète sans qu'un adversaire qui observe leurs échanges ne puisse la reconstituer.

Soit p un nombre premier et g un élément primitif de $\mathbb{Z}/p\mathbb{Z}$.

$$\begin{array}{ccc} (A) & & (B) \\ \text{choisit } a \in \{1,\dots,p-2\} & \xrightarrow{\quad g^a \quad} & \text{choisit } b \in \{1,\dots,p-2\} \\ \text{calcule } g^a & & \text{calcule } g^b \\ K = (g^b)^a = g^{ba} & \xleftarrow{\quad g^b \quad} & K = (g^a)^b = g^{ab} \end{array}$$

Exemple. Soit $p = 601$ et $g = 7$ un générateur de $\mathbb{Z}/p\mathbb{Z}$. Tous les calculs sont effectués modulo 601.

A choisit a au hasard, par exemple $a = 166$. Il calcule $g^a = 7^{166} = 92$ qu'il transmet à B.

B choisit b au hasard, par exemple $b = 499$, il calcule $g^b = 347$ qu'il transmet à A.

A calcule la clé de session $347^a = 347^{166} = 555$.

B calcule la clé de session $92^b = 92^{499} = 555$.

A et B ont donc convenu d'un secret commun 555 qui leur servira à chiffrer leurs messages. Ils ont transmis publiquement les données 92 et 347. Lorsqu'on utilise un entier p suffisamment grand, l'observation de l'échange ne permet pas à un adversaire de reconstituer le secret qu'ont partagé A et B.

7.4. L'exponentiation et le logarithme discret

La fonction à sens unique du protocole DIFFIE-HELLMAN est la fonction puissance. Soient un groupe G et un générateur g de G. L'élévation de g à la puissance x est facile en utilisant la décomposition binaire de x et par élévations au carré et multiplications successives. Par exemple, si $x = 13 = 2^3 + 2^2 + 1$, alors $g^{13} = ((g^2)^2)^2 \times (g^2)^2 \times g$.

Le calcul de la fonction réciproque, appelée *logarithme discret en base* g est un problème difficile (voir § 7.7).

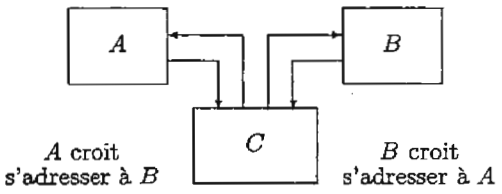
La sécurité du protocole repose sur l'hypothèse selon laquelle il n'est pas possible de résoudre le *problème* DIFFIE-HELLMAN, à savoir déterminer g^{ab} à partir de g, p, g^a et g^b avec les ressources de calcul supposées limités de l'adversaire.

Cette hypothèse est en pratique satisfaite lorsque le groupe G est de la forme $\mathbb{Z}/p\mathbb{Z}$ et lorsque p est un nombre premier de l'ordre de $2^{1\,000}$ (*i.e.* dont l'écriture en base 2 nécessite un millier de chiffres) et que $p - 1$ n'a pas que des *petits* facteurs premiers.

Dans des groupes moins structurés, tels ceux des courbes elliptiques, le problème du logarithme discret est plus difficile encore. Cela permet de définir des systèmes cryptographiques plus efficaces, puisque les calculs opèrent sur des nombres plus petits (de l'ordre de 2^{200}), et avec un niveau de sécurité équivalent.

7.5. L'attaque de l'*homme au milieu*.

Un adversaire qui peut effectuer une intrusion sur la ligne de communication saura se faire passer pour B auprès de A et pour A auprès de B. Il pourra alors capturer, déchiffrer et rechiffrer tous les échanges entre A et B.



Man in the Middle

Pour cette raison, les échanges doivent être authentifiés à l'aide d'un mécanisme de *signature numérique* tel que ceux définis au chapitre 10.

7.6. Le chiffrement ElGamal

Il s'agit d'une variante du protocole Diffie-Hellman. Soit p un nombre premier et g un élément primitif de $\mathbb{Z}/p\mathbb{Z}$. Le destinataire B dispose

- d'une clé privée s qui appartient à l'ensemble $\{1, \ldots, p-1\}$;
- d'une clé publique égale à $g^s \bmod p$.

Lorsque A veut transmettre un message chiffré à B, il doit

- choisir un aléa k dans $\{1, \ldots, p-1\}$;
- calculer la clé de session K par $(g^s)^k$;
- chiffrer son message m avec la clé K à l'aide d'un algorithme symétrique quelconque (DES, AES, *etc.*) pour obtenir le cryptogramme c. Dans la présentation originale de ce système, le message m était un élément de $\mathbb{Z}/p\mathbb{Z}$ et le chiffrement consistait simplement en une multiplication par K modulo p.
- transmettre le couple (g^k, c). La quantité g^k est l'entête du cryptogramme.

Lorsque B reçoit (g^k, c), il calcule la clé de session K par $K = (g^k)^s$ grâce à l'entête et sa clé privée, puis il déchiffre c à l'aide de la clé K.

Exemple

Soit $p = 1\,999$. Le nombre 3 est un générateur du groupe multiplicatif de $\mathbb{Z}/1\,999\mathbb{Z}$.

Soit $a = 732$ la clé privée de A. Sa clé publique Pub_A est $g^a = 3^{732} = 1\,077$.

B souhaite transmettre le message $m = 500$ à A. Il génère une valeur k aléatoire. Par exemple $k = 1\,756$. La clé de session est $K = (Pub_A)^{1\,756} = 1\,137$. Le cryptogramme est constitué des deux valeurs $g^k = 3^{1\,756} = 1\,122$ et $c = K \times m = 1\,137 \times 500 = 784$.

B reçoit donc le couple $(1\,122, 784)$ calcule tout d'abord la clé de session K à l'aide de sa clé privée en effectuant $K = 1\,122^a = 1\,122^{732} = 1\,137$, puis déchiffre le message par $m = 784/1\,137 = 500$.

7.7. Algorithmes de calcul du logarithme discret

7.7.1. Pas-de-bébé, pas-de-géant. (SHANKS) Cet algorithme s'applique aux groupes cycliques les plus généraux. Sa complexité est exponentielle.
Soit G un groupe cyclique d'ordre N et g un générateur. Étant donné un élément $x \in G$, il s'agit de trouver l'entier n tel que $x = g^n$. On pose $K = \lceil \sqrt{N} \rceil$. L'idée est de parcourir les éléments de G de deux façons. Lors du premier parcours, par *pas-de-bébé*, on établit la liste des puissances successives du générateur :

$$A = \{g^i \mid i = 0, 1, \ldots, K-1\}.$$

Lors du second parcours, par *pas-de-géants*, on saute de K en K :

$$B = \{xg^{-Kj} \mid j = 0, 1, \ldots, K-1\}.$$

Ces deux listes ont nécessairement un élément commun $g^i = xg^{-Kj}$, *i.e.* $x = g^{i+Kj} = g^n$ qui correspond à la division euclidienne de n par K.
On crée et mémorise et on trie la liste A. Ensuite les éléments de la liste B sont générés les uns après les autres, puis recherchés dans la liste A. On s'arrête dès qu'un élément généré xg^{-Kj} se trouve dans A au rang i. Le logarithme de x en base g est alors $n = i + Kj$.
La complexité de l'algorithme est exponentielle puisqu'il nécessite le stockage de $\lceil \sqrt{N} \rceil$ éléments et en moyenne $1/2\lceil \sqrt{N} \rceil$ calculs avant de trouver l'élément commun à A et B.

Exemple. Soit $p = 457$. Le nombre $g = 13$ est un générateur de $(\mathbb{Z}/457\mathbb{Z}, \times)$.
Quel est le logarithme en base 13 de 255 modulo 457 ? Tout d'abord, notons que $\lceil \sqrt{457} \rceil = 22$. On construit la suite $A = (13^i)_{i=1\ldots22} = (13, 169, 369, 227, 209, 432, 132, 345, 372, 266, 259, 168, 356, 58, 297, 205, 380, 370, 240, 378, \underline{344}, 359)$.
Ensuite, on calcule les termes de la suite $B = (255 \cdot 13^{-22j})_{j \geq 1}$ jusqu'à trouver un terme de la suite A. Les premiers termes de la suite B sont $(86, 167, 404, 285, 123, 106, \underline{344}, \ldots)$. On trouve que $344 = 13^{21} = 255 \cdot 13^{-22\times7}$ est l'élément commun aux deux listes. Le résultat est donc $\log_{13}(255) = 21 + 22 \times 7 = 175$. En effet, $13^{175} \equiv 255 \pmod{457}$.

7.7.2. Calcul de l'indice (*index calculus*).
Il s'agit d'une méthode bien plus efficace, mais qui ne s'applique que dans un corps fini $\mathbb{F}$. Soient α un élément primitif, et x un élément de $\mathbb{F}$ dont on cherche à évaluer le logarithme en base α.
On opère en deux étapes. Au cours de la première étape, on cherche les logarithmes en base α des éléments d'une *base* B constitués d'éléments dans lesquels il y a de forte chances de pouvoir factoriser un élément tiré au hasard. Lorsque $\mathbb{F}$ est un corps de la forme $\mathbb{Z}/p\mathbb{Z}$, la base B est constituée des h plus petits nombres premiers : $b_1 = 2$, $b_2 = 3$, $b_3 = 5, \ldots, b_h$. Lorsque $\mathbb{F}$ est une extension, B est constituée des h premiers polynômes irréductibles pris dans l'ordre lexicographique. Pour tout $i = 1, \ldots, h$, soit $\ell_i = \log_\alpha(b_i)$. Pour trouver les ℓ_i, on génère t au hasard, puis on calcule α^t et on regarde par divisions successives si α^t se décompose en fonction des éléments b_i de B. Si c'est le cas, on peut écrire $\alpha^t = b_1^{c_1} \cdots b_h^{c_h} = (\alpha^{\ell_1})^{c_1} \cdots (\alpha^{\ell_h})^{c_h}$. Les ℓ_i sont alors solutions de l'équation linéaire $c_1\ell_1 + \cdots c_h\ell_h = t$. On recommence avec d'autres valeurs de t. Lorsqu'on dispose d'assez d'équations linéaires indépendantes, la résolution du système linéaire fournit toutes les valeurs des ℓ_i.
La deuxième étape consiste à calculer le logarithme de x. Pour cela, on tire au hasard un entier t jusqu'à pouvoir décomposer $x\alpha^t$ en fonction des b_i par divisions successives comme à la première étape. Soit $x\alpha^t = b_1^{d_1} \cdots b_h^{d_h} = (\alpha^{\ell_1})^{d_1} \cdots (\alpha^{\ell_h})^{d_h}$ la décomposition trouvée. On en déduit :

$$\log_\alpha(x) = d_1p_1 + \cdots d_hp_h - t$$

Le choix de la taille h de la base est délicat et critique pour l'efficacité de l'algorithme. Avec une petite valeur de h, peu d'équations sont requises, mais il y a peu de chance qu'un élément α^t pris au hasard se décompose en fonction des α^{p_i}. Un t convenable sera long à trouver. *A contrario*, avec une valeur élevée de h, il faut un grand nombre d'équations pour trouver tous les ℓ_i.

Exemple. Soit $p = 2\,777$. Le nombre 3 est générateur de $(\mathbb{Z}/2\,777\mathbb{Z}, \times)$.
On considère la base $B = \{2, 3, 5, 7\}$. La première étape consiste à calculer les logarithmes en base 3 de 2, 3, 5 et 7. Soient $\ell_2 = \log_3(2)$, $\ell_3 = \log_3(3)$, $\ell_5 = \log_3(5)$ et $\ell_7 = \log_3(7)$. Notons que $\ell_3 = 1$. Il faut générer t au hasard, calculer 3^t jusqu'à obtenir un résultat qui se décompose dans B. On trouve par exemple :
$3^{1\,642} = 8 = 2^3 = 3^{3\ell_2}$
$3^{2\,356} = 35 = 5 \times 7 = 3^{\ell_5} \times 3^{\ell_7}$
$3^{2\,548} = 1\,029 = 3 \times 7^3 = 3^{\ell_3} \times 3^{3\ell_7}$
La résolution du système $\begin{cases} 1\,642 &= 3\ell_2 \\ 2\,356 &= \ell_5 + \ell_7 \\ 2\,548 &= \ell_2 + 3\ell_7 \end{cases}$ conduit à $\begin{cases} \ell_2 &= 2\,398 \\ \ell_5 &= 1507 \\ \ell_7 &= 849 \end{cases}$
Ensuite, il faut générer t au hasard jusqu'à ce que $1\,000 \times 3^t$ se décompose dans B. Par exemple $t = 1\,412$ donne $1\,000 \times 3^{1\,412} = 108 = 2^2 \times 3^3 = 3^{2\ell_2} \times 3^{3\ell_3}$, d'où on déduit que $1\,000 = 3^{2\ell_2 + 3\ell_3 - 1412}$ et finalement $\log_3(1\,000) = 2 \times 2\,398 + 3 - 1\,412 = 611$.

8 – Le cryptosystème de Rabin

Dans le cryptosystème de Rabin, la fonction de chiffrement est l'élévation au carré modulo un entier n qui est le produit de deux nombres premiers. On montre que, dans ce cas, savoir extraire une racine carrée modulo n revient à savoir factoriser n. Le détenteur de la factorisation de n peut déchiffrer. Pour les autres, le déchiffrement est, en pratique, impossible.

8.1. Calcul de la racine carrée modulo n

8.1.1. Racines carrées modulo un nombre premier.

Proposition. *Soit p un nombre premier congru à 3 modulo 4. Soit x un élément de $\mathbb{Z}/p\mathbb{Z}$ qui est un carré. Alors les racines carrées de x modulo p sont $\pm x^{(p+1)/4}$.*

Preuve. Remarquons tout d'abord que comme p est supposé congru à 3 modulo 4, l'entier $p+1$ est divisible par 4. Comme x est un carré, posons $x = t^2$ et $y = x^{(p+1)/4}$. On a $y^2 = x^{(p+1)/2} = t^{p+1} = t^{p-1} \cdot t^2 = t^2 = x$. □

Exemple. Une racine carrée de 5 dans $\mathbb{Z}/19\mathbb{Z}$ est $5^{(19+1)/4} = 5^5 = 9$. L'autre racine est -9 c'est-à-dire 10.
Dans $\mathbb{Z}/11\mathbb{Z}$, on a $5^{(11+1)/4} = 5^3 = 4$. Une racine carré de 5 est 4.

Lorsque p n'est pas congru à 3 modulo 4, le calcul de la racine carrée modulo p est un peu plus complexe, mais il existe des algorithmes efficace.

8.1.2. Racines carrées modulo un produit de deux nombres premiers.

Établissons tout d'abord le résultat suivant qui énonce que si un entier est connu modulo p et modulo q, alors il est déterminé de manière unique modulo le produit pq.

Lemme. [Théorème chinois] *Soient p et q deux entiers premiers entre eux et soient u et v tels que $up + vq = 1$. Si $\begin{cases} y \equiv a \bmod p \\ y \equiv b \bmod q \end{cases}$, alors $y \equiv aqv + bpu \bmod pq$.*

Preuve. La relation $up + vq = 1$ signifie que u est l'inverse de p modulo q. Par hypothèse, il existe un entier k tel que

$$(*) \qquad y = a + kp$$

Donc, dans $\mathbb{Z}/q\mathbb{Z}$, on a

$$\begin{aligned} a + kp &= b \\ kp &= (b - a) \\ k &= (b - a)u \end{aligned}$$

Comme cette dernière égalité est dans $\mathbb{Z}/q\mathbb{Z}$, il existe un entier h tel que

$$k = (b - a)u + hq$$

En utilisant ce résultat dans l'égalité (*), on déduit

$$y = a + \big((b - a)u + kq\big)p.$$

En réduisant modulo pq, on obtient, dans $\mathbb{Z}/pq\mathbb{Z}$:

$$\begin{aligned} y &= a + (b - a)up \\ &= a(1 - up) + bup \\ &= avq + bup \end{aligned}$$

□

Cette proposition montre même que les anneaux $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ et $\mathbb{Z}/pq\mathbb{Z}$ sont isomorphes.

PROPOSITION. *Soit n le un produit de deux nombres premiers p et q et soit $x \in \mathbb{Z}/n\mathbb{Z}$, un carré modulo n. Alors x est un carré dans $\mathbb{Z}/p\mathbb{Z}$ et dans $\mathbb{Z}/q\mathbb{Z}$. Soit y_p une racine carrée de x dans $\mathbb{Z}/p\mathbb{Z}$ et y_q une racine carrée de x dans $\mathbb{Z}/q\mathbb{Z}$. Soient également u et v les éléments de $\mathbb{Z}$ qui vérifient $up + vq = 1$. Les racines carrées de x dans $\mathbb{Z}/n\mathbb{Z}$ sont les quatre éléments $\pm puy_q \pm qvy_p$.*

Preuve. Comme x est un carré dans $\mathbb{Z}/n\mathbb{Z}$, il existe des entiers t et k tel que $x = t^2 + pqk$. En réduisant respectivement cette égalité modulo p et modulo q, on établit que x est un carré dans $\mathbb{Z}/p\mathbb{Z}$ et dans $\mathbb{Z}/q\mathbb{Z}$.
Les racines carrées de x dans $\mathbb{Z}/p\mathbb{Z}$ sont $\pm y_p$.
Les racines carrées de x dans $\mathbb{Z}/q\mathbb{Z}$ sont $\pm y_q$.
Le lemme ci-dessus permet de conclure. □

Exemple 19 et 11 sont relativement premiers et vérifient $7 \times 11 - 4 \times 19 = 77 - 76 = 1$. Les racines carrées de 5 modulo $11 \times 19 = 209$ sont $\pm 77 \times 9 \pm 76 \times 4$, c'est-à-dire les éléments de $\{29, 48, 161, 180\}$.

8.1.3. Extraire une racine carrée permet de factoriser. On a montré que la connaissance de la factorisation de n permet d'extraire les racines carrées modulo n. Nous montrons ici la réciproque. Supposons toujours que n, est un produit de deux facteurs premiers, mais dont on ignore la factorisation. Supposons également qu'un algorithme nous fournisse une racine carrée modulo n. Choisissons x au hasard et demandons à l'algorithme de trouver une racine carrée de x^2. Sa réponse y satisfait

$$x^2 \equiv y^2 \pmod{n} \qquad i.e. \qquad n \mid (x+y)(x-y)$$

Si x est aléatoire, avec une probabilité $1/2$, au aura $x \not\equiv \pm y \pmod{n}$. Dans ce cas, n divise $(x+y)(x-y)$ sans diviser aucun des deux facteurs. Par conséquent, le pgcd de n et de $x+y$ est un facteur non trivial de n.

Exemple Soit $n = 61\,181$. Soit $x = 196$. On a $x^2 = 38\,416$. Si un oracle nous informe que $34\,357$ est une racine carrée de $38\,416$, on peut aisément le vérifier : $34\,357^2 = 38\,416$. Mais alors on peut aussi factoriser n en calculant $\text{pgcd}(61\,181, 196 + 34\,357) = 317$. Par division, $61\,181 = 317 \times 193$.

8.2. Fonction à sens unique avec porte dérobée (*trapdoor*)

Une fonction à sens unique à porte dérobée est une fonction $f : E \to E$ injective qui est à sens unique, sauf si on dispose d'une information (la « *trappe* ») qui permet de l'inverser.
Lorsque n est un produit de deux nombres premiers, l'élévation au carré modulo n en est un exemple.
Pour tout le monde, élever au carré modulo n est une opération facile.
Effectuer l'opération inverse est réservé à ceux qui connaissent la factorisation de n. Si la factorisation de n n'est pas révélée, la porte dérobée est aussi difficile à trouver que l'entier n à factoriser. C'est sur cette propriété qu'est fondé le système à clé publique de RABIN.

8.3. Définition du système de RABIN

Paramètres.
- La clé privée est constituée de deux nombres premiers p et q qui sont choisis congrus à 3 modulo 4 afin de faciliter l'extraction des racines carrées.
- La clé publique n est le produit $p \times q$.
- L'espace des messages et des cryptogrammes est l'ensemble $\mathbb{Z}/n\mathbb{Z}$.

Chiffrement. Le chiffrement d'un message m consiste à l'élever au carré modulo n :

$$c = m^2 \pmod{n}$$

Déchiffrement. Le déchiffrement consiste à extraire une racine carrée du cryptogramme modulo n par la formule de la proposition du § 8.1.1. Cette formule donne 4 racines qu'il faut distinguer. En pratique, ce n'est pas un obstacle, à condition d'imposer un format particulier aux messages clairs comme dans l'exemple ci-après.

Exemple. Soit $n = 103 \times 139 = 14\,310$. Le produit est publié, mais la factorisation est maintenue secrète. Il est convenu que les messages clairs finissent par deux zéros. Soit à chiffrer le message $m = 10\,000$. Le cryptogramme est $c = m^2 = 10\,072$.
Pour déchiffrer c, il faut trouver des racines carrées de c modulo 103 et modulo 139. Ceci est réalisé en calculant modulo 103 : $c' = c^{(103+1)/4} = c^{26} = 9$ et modulo 139 : $c'' = c^{(139+1)/4} = c^{35} = 131$. De la relation de BÉZOUT $27 \times 103 - 20 \times 139 = 1$, on déduit que les quatre racines carrées de c modulo n sont $\pm 131 \times 27 \times 103 \pm 9 \times 20 \times 139$ *i.e.* les éléments de $\{2\,772, 4\,310, 10\,000, 11\,538\}$. La convention sur le clair permet de choisir $m = 10\,000$ parmi ces quatre solutions.

8.4. Sécurité du système de RABIN

Le système de RABIN *est sûr contre une attaque à clair choisi.*
Il s'agit du niveau minimal de sécurité requis pour un système à clé publique, puisque tout le monde peut chiffrer le message de son choix.
Si un adversaire sait déchiffrer tout message chiffré à l'aide du système de RABIN, alors il sait trouver une racine carrée d'un cryptogramme donné. Comme cela est expliqué au § 8.1, l'adversaire sait alors factoriser la clé publique. *Tant que la factorisation est un problème difficile, le système de* RABIN *est sûr.*

Par contre, *le système de* RABIN *n'est pas sûr contre une attaque à cryptogramme choisi.*
Si un adversaire a le moyen de faire déchiffrer un cryptogramme de son choix, cela revient à pouvoir extraire des racines carrées et donc à factoriser la clé publique. Pour parer cette attaque, il faut que les messages clairs valides ne constituent qu'une partie de $\mathbb{Z}/n\mathbb{Z}$ en proportion infime. Le dispositif de déchiffrement ne doit délivrer le message clair que s'il est valide.
C'est sans doute cette faiblesse du système de RABIN qui fait qu'il est peu utilisé en pratique.

8.5. Factoriser les grands entiers.

Pour résister aux algorithmes de factorisation actuels, l'entier n doit avoir une représentation binaire sur au moins 768 chiffres binaires, voire 1024 ou 2048. Ce paragraphe présente l'algorithme de factorisation d'où sont dérivés les réalisations les plus efficaces actuellement.

Algorithme du crible quadratique.
Étant donné un entier n à factoriser, l'objectif est de trouver une relation non triviale de la forme $x^2 = y^2$ dans $\mathbb{Z}/n\mathbb{Z}$. Pour cela, on se fixe une base de factorisation B constituée des h plus petits nombres premiers : $B = \{2, 3, 5, 7, 11, \ldots, p_h\}$. On génère au hasard des éléments x de $\mathbb{Z}/n\mathbb{Z}$. On ne conserve que ceux dont le carré modulo n se factorise dans B. Chaque élément trouvé fournit une relation du type

$$x_i^2 \equiv p_1^{c_{i,1}} \cdots p_h^{c_{i,h}} \pmod{n}.$$

Il y a peu de chance pour qu'une relation de ce type conduise à une égalité entre deux carrés, mais il est possible d'en combiner plusieurs pour arriver au résultat souhaité. Commençons par diviser modulo n les deux membres par les facteurs carrés. On obtient alors une famille de relations de la forme

$$y_i^2 \equiv p_1^{\alpha_{i,1}} \cdots p_h^{\alpha_{i,h}} \pmod{n},$$

où les $\alpha_{i,j}$ appartiennent à $\{0, 1\}$. Chaque vecteur $\alpha_i = (\alpha_{i,j})_{j=1,\ldots,h}$ est un élément de l'espace vectoriel de dimension h sur $\mathbb{F}_2$.
En accumulant un nombre suffisant de relations de ce type, on finit immanquablement par obtenir un système de vecteurs liés par une dépendance linéaire modulo 2. Il existe un sous-ensemble I tel que dans $\mathbb{F}_2^h$,

$$\sum_{i \in I} \alpha_i = 0$$

Lorsqu'on effectue la somme des composantes des vecteurs α_i dans $\mathbb{N}$ et non pas dans $\mathbf{F}_2$, on obtient un vecteur $a = \sum_{i \in I} \alpha_i = (a_1, \ldots, a_h)$ dont toutes les composantes sont paires.
En effectuant le produit terme à terme des relations correspondantes sur les y_i on obtient une relation

$$\prod_{i \in I} y_i^2 \equiv p_1^{a_1} \cdots p_h^{a_h} \pmod{n},$$

qui, puisque a est un vecteur à composantes toutes paires, est maintenant une égalité entre deux carrés modulo n. Si cette relation n'est pas triviale, alors on en déduit la factorisation de n. Sinon, il faut chercher d'autres relations.

Exemple. On souhaite factoriser n égal à 39 713. Un tirage au hasard de x et le calcul de $x^2 \mod n$ conduit aux relations suivantes modulo n :

$$\begin{cases} 20\,598^2 = 3^3 \times 5^3 \times 7 \\ 23\,650^2 = 2^9 \times 3^2 \\ 16\,742^2 = 2 \times 3 \times 5 \times 7 \end{cases}$$

En divisant par les carrés modulo n, on trouve respectivement les relations suivantes et les vecteurs α_i correspondants dont les composantes sont égales aux exposants dans la décomposition en facteurs premiers des carrés du premier membre.

$$\begin{cases} 25\,201^2 = 3 \times 5 \times 7 \\ 38\,551^2 = 2 \\ 16\,742^2 = 2 \times 3 \times 5 \times 7 \end{cases} \qquad \begin{cases} \alpha_1 = (0,1,1,1) \\ \alpha_2 = (1,0,0,0) \\ \alpha_3 = (1,1,1,1) \end{cases}$$

La relation $\alpha_1 + \alpha_2 + \alpha_3$ est satisfaite dans $\mathbf{F}_2^4$. En effectuant le produit terme à terme des trois relations, on trouve $9\,152^2 \equiv 210^2 \pmod{n}$, d'où on déduit un facteur non trivial par $\operatorname{pgcd}(39\,713, 9\,152 + 210) = 151$. Par conséquent :

$$39\,713 = 151 \times 263.$$

Comme pour le calcul du logarithme discret, le choix de la taille h de la base B de factorisation est critique pour l'efficacité de l'algorithme. Si h est trop petit, il y a peu de chances que les x_i^2 se factorisent dans B. Si h est trop élevé, il faudra beaucoup de relations pour trouver une dépendance linéaire entre les α_i.
Les progrès récents dans l'efficacité de la factorisation résident dans des techniques à la fois algorithmiques et algébriques pour trouver des carrés modulo n qui se factorisent plus facilement dans une base de nombres premiers.

9 – LE RSA

Le système RSA (1977), du nom de ses trois inventeurs RIVEST, SHAMIR et ADLEMAN, est sans doute le plus populaire des systèmes à clés publiques. Outre le chiffrement, il permet également de signer des messages. Comme le système de RABIN, il repose sur la difficulté de factoriser les grands entiers. Contrairement à ce dernier, il n'est pas prouvé que le décryptage du RSA soit équivalent à la factorisation des grands entiers, ce qui le rend sûr également contre une attaque à cryptogrammes choisis.

9.1. Indicateur d'EULER

Soit $n \geq 2$ un entier. On note U_n le groupe des éléments inversibles du groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z} \setminus \{0\}, \times)$. Il est constitué des entiers $\in \{1, 2, \ldots, n-1\}$ qui sont premiers avec n. On appelle *indicateur d'*EULER la fonction qui à n entier ≥ 2, associe l'ordre du groupe U_n, c'est-à-dire le nombre d'entiers compris entre 1 et $n-1$ qui sont premiers avec n.

Par exemple, pour $n = 15$, le groupe des éléments inversibles modulo 15 est constitué des entiers compris entre 1 et 14 et qui sont premiers avec 15, soit $U_{15} = \{1, 2, 4, 7, 8, 11, 13, 14\}$

Si n est le produit de deux nombres premiers différents p et q, les éléments de $\{1, 2, \ldots, n-1\}$ qui ne sont pas premiers avec n sont les multiples de p et les multiples de q, *i.e.* $\{p, 2p, \ldots, (q-1)p\}$ et $\{q, 2q, \ldots, (p-1)q\}$. Comme p et q sont premiers entre eux, ces deux ensembles sont disjoints et contiennent respectivement $q-1$ et $p-1$ éléments. On déduit que

$$\varphi(pq) = pq - 1 - (p-1) - (q-1) = (p-1)(q-1).$$

Cette égalité montre que si p et q sont connus, alors $\varphi(n)$ se calcule aisément.
Réciproquement, si n et $\varphi = \varphi(n)$ sont connus et si on sait que n est le produit de deux nombres premiers, alors on retrouve les facteurs de n en résolvant une équation du second degré :

$$(x-1)\left(\frac{n}{x} - 1\right) = \varphi$$
$$x^2 - p(n + 1 - \varphi) + n = 0$$

THÉORÈME DE LAGRANGE. *Si a est premier avec n, alors*

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

Preuve. Si a et x sont premiers avec n, alors ax est aussi premier avec n. L'application $x \mapsto ax$ est une bijection de U_n. En changeant l'ordre des facteurs, on a

$$\prod_{x \in U_n} x = \prod_{x \in U_n} ax.$$

Le membre de droite est $a^{\varphi(n)}$ fois le membre de gauche qui est inversible. □

Soit $n \in \mathbb{N}^*$ et e premier avec $\varphi(n)$. La fonction

$$\begin{array}{ccc} U_n & \to & U_n \\ x & \mapsto & x^e \end{array}$$

est une fonction à sens unique avec porte dérobée :

- Retrouver x à partir de $y = x^e$ est un problème difficile. Il s'agit, par définition, du *problème* RSA.
- Lorsqu'on connaît $\varphi(n)$, ce problème devient facile. Soit d l'inverse de e modulo $\varphi(n)$. Les applications $x \mapsto x^e$ et $y \mapsto y^d$ sont inverses l'une de l'autre dans U_n

Remarque: Sans le « *modulo* n », le calcul de x connaissant $y = x^e$, c'est-à-dire le calcul de la racine e-ième, est facile, par exemple en utilisant $x = \exp\left(\frac{1}{e}\log(y)\right)$ avec la précision suffisante.

9.2. Description du système RSA

La clé privée est constituée de deux nombres premiers p et q et d'un entier d premier avec $\varphi(n)$ qui, rappelons le, vaut $(p-1)(q-1)$.

La clé publique est constituée du produit n de p et q, sans que les facteurs p et q ne soient révélés, et d'un entier e, premier avec $\varphi(n)$.

Les entiers e et d vérifient $ed = 1$ dans $\mathbb{Z}/\varphi(n)\mathbb{Z}$.

L'espace des messages est l'ensemble U_n des éléments inversibles modulo n.

L'opération de chiffrement est l'élévation à la puissance e modulo n : $x \mapsto x^e \bmod n$. L'entier e s'appelle l'*exposant public*.

L'opération de déchiffrement est l'élévation à la puissance d modulo n : $x \mapsto x^d$. L'entier d s'appelle l'*exposant privé*.

Exemple. Soit $p = 11$ et $q = 17$. Le module est $n = pq = 187$ avec $\varphi(n) = 10 \times 16 = 160$. En pratique, p et q devront être tels que le produit pq soit pratiquement impossible à factoriser. Soit $e = 3$ l'exposant public. L'exposant privé correspondant est l'inverse de 3 modulo 160, qui vaut 107.

Soit $m = 12$ le message à chiffrer. Le cryptogramme c est donné par $c = 12^3 = 45$ dans $\mathbb{Z}/187\mathbb{Z}$. On retrouve le clair par l'égalité $45^{107} = 12$ dans $\mathbb{Z}/187\mathbb{Z}$.

On doit en toute rigueur se restreindre à des représentations du clair qui soient des entiers premiers avec n. Dans la pratique, on ignore cette condition et on considère comme message valide tout élément de $\mathbb{Z}/n\mathbb{Z}$, et ceci pour deux raisons :

1. La probabilité pour qu'un entier x ne soit pas premier avec n est extrêmement faible. Elle vaut $\dfrac{1}{n-\varphi(n)} = \dfrac{1}{p+q-1}$. Lorsque p et q sont représeutables avec le même nombre de chiffres binaires, cette quantité est de l'ordre de $\dfrac{1}{2\sqrt{n}}$. Lorsque n est grand, il s'agit d'une quantité véritablement négligeable. Par exemple, si n vaut environ $2^{1\,024}$, la probabilité pour qu'un élément, pris au hasard dans $\mathbb{Z}/n\mathbb{Z}$, ne soit pas inversible modulo n est de l'ordre de $1/2^{513}$. D'ailleurs, la connaissance d'un entier x non premier avec n permettrait de factoriser n par calcul de $\operatorname{pgcd}(n, x)$. De tels entiers sont donc aussi difficiles à trouver que l'entier n l'est à factoriser.
2. Si le message m n'est pas inversible modulo n, l'application des formules ci-dessus permet quand-même un chiffrement et un déchiffrement correct. Supposons par exemple $m = up$. Soit $c = m^e = (up)^e$. Alors $c^d = (up)^{ed}$. Comme d est inverse de e modulo $\varphi(n) = (p-1)(q-1)$, il existe un entier k qui vérifie $ed = 1 + k(p-1)(q-1)$. En conséquence $c^d = (up)^{1+k(p-1)(q-1)}$. Bien évidemment, c^d est multiple de p donc s'écrit $c^d = ap$. D'autre part, d'après le théorème de FERMAT, $c^d = up$ dans $\mathbb{Z}/q\mathbb{Z}$. Il existe donc un élément b de $\mathbb{Z}$ tel que $c^d = up + bq$. En conséquence, $ap = up + bq$. L'entier b est donc multiple de p, ce qui prouve que $c^d = up = m$ dans $\mathbb{Z}/n\mathbb{Z}$. Le cryptogramme c est donc correctement déchiffré.

Exemple Avec les mêmes clés que dans l'exemple précédent, soit m un message, égal à 33 qui n'est pas inversible modulo 187. Le chiffrement de m est donné par $c = 44^3 = 99$. Le déchiffrement de c est bien donné par $99^{107} = 44$.

9.3. La sécurité du RSA

Bien qu'on ne sache pas encore le démontrer, la seule attaque générale connue actuellement contre le RSA reste la factorisation du module n. Cependant, certains choix malheureux des paramètres ou une mauvaise utilisation peuvent compromettre la sécurité.

9.3.1. La connaissance de l'exposant privé permet de factoriser le module. Cependant, on ne sait pas s'il existe ou non d'autres méthodes pour inverser la fonction $x \mapsto x^e$.
Supposons qu'un adversaire connaisse l'exposant public e et l'exposant privé d. On a $ed \equiv 1$ modulo $\varphi(n)$. Donc $ed-1$ est multiple de $\varphi(n)$ qui est pair. La quantité $ed-1$ est divisible par deux k fois jusqu'à obtenir $ed-1 = 2^k u$ avec u impair. Soit a un élément non nul quelconque de $\mathbb{Z}/n\mathbb{Z}$. Il vérifie $a^{ed-1} = (a^u)^{2^k} = 1$. Soit $b = a^u$. Lorsqu'on élève successivement au carré b, on obtient 1 au bout d'au plus k étapes. L'avant dernière étape fournit une racine carrée de 1. Si c'est -1, cela ne donne rien. Il faut recommencer avec une autre valeur de a. Mais si c'est une racine carrée non triviale de 1, alors cela permet de factoriser n.

Exemple 1. Le module est $n = 187$, l'exposant public est $e = 3$ et l'exposant privé est $d = 107$. On a $ed - 1 = 320 = 2^6 \times 5$. Prenons $a = 2$. Élevons successivement au carré modulo 187 à partir de $2^5 = 32$. On trouve successivement $32^2 = 89$; $89^2 = 67$; $89^2 = 67$; $67^2 = 1$. L'entier 67 est une racine carrée de 1 différente de ± 1 qui permet de trouver un facteur de 187 par $\operatorname{pgcd}(187, 67-1) = 11$.

Exemple 2. Un adversaire connaît $e = 3$, $d = 17\,563$ et $n = 26\,671$. Il calcule $ed - 1 = 52\,688 = 16 \times 3\,293$, puis observe que $2^{4\times 3\,293} = 1$ et $2^{2\times 3\,293} = 24\,882$ est une racine carrée non triviale de 1. Par conséquent $\operatorname{pgcd}(24\,881, 26\,671) = 179$ et $\operatorname{pgcd}(24\,883, 26\,671) = 149$ donnent la factorisation de n.

9.3.2. Exposant public court. Pour accélérer l'opération de chiffrement, on a intérêt à choisir un exposant public le plus petit possible. Le choix $e = 3$ est souvent adopté, car $m^3 = m^2 \times m$. Le calcul de m^3 ne nécessite qu'une élévation au carré et une multiplication. Mais il est alors interdit de chiffrer le même message avec des modules différents (voir TD). Les messages doivent être complétés avec du bourrage aléatoire (*padding*).

9.3.3. Attaque de WIENER sur les exposants privés courts. Lorsque l'exposant privé est trop court, un adversaire peut le retrouver par un calcul simple sur les paramètres publics, et donc factoriser le module, comme le montre la proposition suivante :

PROPOSITION. *On considère un système* RSA *de module n et d'exposant public e. Si les facteurs p et q de n sont représentés en numération binaire avec le même nombre de chiffres et si l'exposant privé d est inférieur à $\sqrt[4]{n}/2$, alors d est le dénominateur d'une des réduites du développement de $\dfrac{e}{n - 2\lfloor\sqrt{n}\rfloor + 1}$ en fraction continue.*

La condition sur la taille de p et q en numération binaire est parfaitement vraisemblable, car elle conduit à la plus grande complexité pour factoriser n.
Comme il existe un entier k tel que $ed = 1 + k\varphi(n)$, on a, $\dfrac{e}{\varphi(n)} - \dfrac{k}{d} = \dfrac{1}{d\varphi(n)}$. La fraction $\dfrac{k}{d}$ est une approximation de $\dfrac{e}{\varphi(n)}$. On ne connaît pas cette dernière fraction, mais il est possible d'approcher $\varphi(n)$ par une quantité connue $\widetilde{n}$, égale à n, ou mieux encore, à $n - 2\lfloor\sqrt{n}\rfloor + 1$. Si l'approximation n'est pas trop mauvaise, le développement en fractions continues de $\dfrac{e}{\widetilde{n}}$ donnera une réduite égale à $\dfrac{k}{d}$. En testant tous les dénominateurs des réduites comme candidat pour l'exposant privé, on finit par trouver le véritable exposant privé d.
Preuve de la proposition. Soit $\widetilde{n}$ égal à $n - 2\lfloor\sqrt{n}\rfloor + 1$ et k tel que $ed = 1 + k\varphi(n)$. On a :

$$\left|\frac{e}{\widetilde{n}} - \frac{k}{d}\right| = \left|\frac{ed - k\varphi(n) + k\varphi(n) - k\widetilde{n}}{\widetilde{n}}\right| = \left|\frac{1 + k(\varphi(n) - \widetilde{n})}{\widetilde{n}}\right|.$$

De l'égalité $\dfrac{e}{\varphi(n)} - \dfrac{k}{d} = \dfrac{1}{d\varphi(n)}$ et comme $\dfrac{e}{\varphi(n)} < 1$, on déduit $\dfrac{k}{d} < 1$. De plus, si p et q s'écrivent avec autant de chiffres en numération binaire, ils vérifient $p + q < 3\lfloor\sqrt{n}\rfloor$. De $2\widetilde{n} > n$, se déduit :

$$\left|\frac{e}{\widetilde{n}} - \frac{k}{d}\right| < \frac{\lfloor\sqrt{n}\rfloor}{\widetilde{n}} < \frac{2\lfloor\sqrt{n}\rfloor}{n}$$

Dès lors que $d \leq \frac{\sqrt[4]{n}}{2}$, l'inégalité $\left|\frac{e}{n} - \frac{k}{d}\right| < \frac{1}{2d^2}$ est satisfaite et $\frac{k}{d}$ est bien une réduite du développement en fractions continues de $\frac{e}{n}$.

Exemple. Soit $n = 9\,047$ et $e = 7\,591$ les paramètres publics d'un RSA. On approche $\varphi(n)$ par $9\,047 - 2\lfloor\sqrt{9\,047}\rfloor + 1 = 8\,858$. Le développement en fraction continue de $\frac{7\,591}{8\,858}$ est

$$\cfrac{1}{1+\cfrac{1}{5+\cfrac{1}{1+\cfrac{1}{114+\cfrac{1}{5+\frac{1}{2}}}}}}$$

Les réduites successives sont $1, \frac{5}{6}, \frac{6}{7}, \frac{689}{804}\cdots$

Le message 2 se chiffre par $2^{7\,591} = 8\,485$. En tentant de déchiffrer ce cryptogramme avec les dénominateurs des réduites, on trouve successivement $8\,485^6 = 1\,481$, puis $8\,485^7 = 2$. L'exposant privé est $d = 7$.

9.3.4. Chaque destinataire doit avoir un module qui lui est propre. Dans le cas contraire, un adversaire peut décrypter un même message qui a été transmis à plusieurs destinataires (voir TD).

10 – Signature numérique

Grâce à l'avènement de la cryptographie à clés publiques, il est désormais possible de signer numériquement des documents. Le processus de signature est essentiellement asymétrique :

- seule la personne autorisée peut signer un document ;
- tout le monde doit pouvoir vérifier la signature.

Une signature doit permettre

- de s'assurer de l'origine du document, c'est la fonction d'*authentification* ;
- d'empêcher de se rétracter d'un engagement, c'est la fonction de *non-répudiation*.

Un mécanisme de signature est requis pour garantir l'authenticité des clés publiques utilisés dans un schéma de chiffrement asymétrique et éviter l'attaque de l'homme au milieu (voir § 7.5). Les clés sont ainsi assorties d'un *certificat* qui est une signature délivrée par une *autorité de certification*. En pratique, certaines compagnies, comme la compagnie Verisign, délivrent un *certificat*, c'est-à-dire une signature, pour les clés publiques des utilisateurs qui en font la demande. La vérification des certificats utilise la clé publique de l'autorité de certification dont l'authenticité ne fait aucun doute en raison de sa diffusion à grande échelle.

10.1. Définition d'un schéma de signature

Une signature numérique est une donnée σ qui est ajoutée au message m. Le message m n'est pas nécessairement chiffré. Le destinataire reçoit le couple (m, σ). La donnée σ dépend du message et d'une clé privée. Elle permet d'authentifier l'origine du message m. La vérification utilise une clé publique.

Plus précisément, un schéma de signature à clé publique comporte :

- une clé privée K_{priv} ;
- une clé publique K_{pub}, associée à K_{priv} ;
- un algorithme de génération de signature G, qui calcule une signature σ, à partir d'un message m et de la clé privée K_{priv}, donné par $\sigma = G(m, K_{\text{priv}})$;
- un algorithme de vérification de signature V, qui rend une valeur binaire $V(m, \sigma, K_{\text{pub}}) \in \{\text{bonne}, \text{mauvaise}\}$ à partir d'un message m, d'une signature σ et de la clé publique K_{pub}.

Le détenteur légitime de la clé privée peut produire la signature d'un document, et lui seul peut le faire. Ceci permet d'*authentifier* l'auteur du document. Cela implique également que le signataire ne peut pas *répudier* sa signature.

L'objectif de l'adversaire est de *forger* la signature d'un document, ce qui signifie pouvoir la générer sans disposer de la clé privée.

10.2. Fonctions de hachage (ou de condensation)

Une fonction de hachage permet de réduire une donnée binaire de n'importe quelle taille en un *condensé* de taille fixe. Elles ne comportent aucun élément secret. On les utilise pour la préparation des données en vue d'une signature numérique. Elles peuvent également servir à vérifier l'intégrité des données.
Les algorithmes standards sont SHA (*secure hash algorithm*), MD5 (*message digest*), RIPEMD.
Il existe des méthode pour définir des fonctions de hachage à partir d'une primitive de calcul par bloc comme le DES, l'AES, *etc.*
Il existe également des construction de fonctions de hachage à partir de fonctions à sens unique comme la fonction exponentielle (voir TD).
Une fonction de hachage est une application h, définie sur l'ensemble $\{0,1\}^*$ des mots de toute longueur sur l'alphabet $\{0,1\}$ et à valeurs dans $\{0,1\}^n$, et qui a les propriétés suivantes :

- Elle est simple à calculer à l'aide d'un algorithme efficace (complexité polynomiale).

- C'est une fonction à sens unique: pour presque tout y dans $\{0,1\}^n$, il est pratiquement impossible de trouver x dans $\{0,1\}^*$ tel que $h(x) = y$.
- Elle résiste aux *collisions faibles*: étant donné un couple $(x, h(x))$ dans $\{0,1\}^* \times \{0,1\}^n$, il est pratiquement impossible de trouver un autre x' dans $\{0,1\}^*$, différent de x tel que $h(x) = h(x')$.
- Elle résiste aux *collisions fortes*: il est pratiquement impossible de trouver deux éléments différents x et x' dans $\{0,1\}^*$, tels que $h(x) = h(x')$.

La valeur de n doit être suffisante (supérieure à 2^{128}, voire à 2^{160}) pour que h résiste à l'attaque du paradoxe des anniversaires (voir TD).
La signature d'un document procède donc en deux étapes:

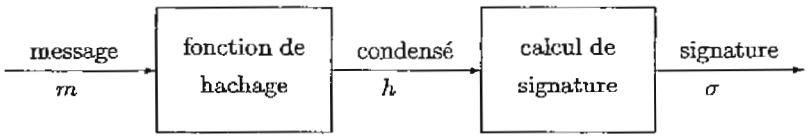
1. Soumettre le document à une fonction de hachage de manière à produire un condensé h de taille fixe.
2. Appliquer à h une fonction de signature telle que celles qui sont décrites aux paragraphes 10.3 et 10.4.

Le message et sa signature (m, σ) sont transmis au destinataire.

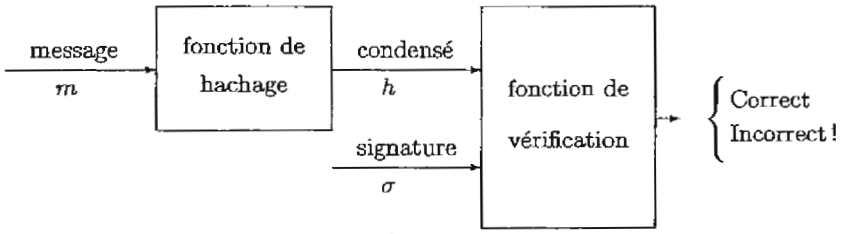
Le destinataire du document vérifie également en deux étapes:

1. Soumettre le document à la même fonction de hachage pour calculer le condensé h.
2. Soumettre le condensé h et la signature σ à la fonction de vérification pour déterminer si la signature est correcte ou non.

Calcul de signature:



Vérification de signature:



10.3. La signature RSA

On considère un schéma RSA défini par
- un module n;
- un exposant public e;
- un exposant privé d.

Le principe de la signature RSA est d'inverser les rôles des exposants e et d par rapport à leur utilisation dans un schéma de chiffrement.
L'exposant privé d sert à chiffrer le condensé du message. Seul le détenteur de la clé privée peut le faire. La signature est le résultat de ce chiffrement:

$$\sigma = h^d$$

L'exposant public e sert à déchiffrer la signature. La vérification consiste à vérifier que le résultat de ce déchiffrement vaut bien le condensé du message:
- Si $\sigma^e = h$ alors la signature est valide;
- Si $\sigma^e \neq h$ alors la signature est incorrecte.

Exemple Soient $p = 113$, $q = 131$ et $n = pq = 14\,803$. Soit $e = 3$ l'exposant public. L'exposant privé correspondant est l'inverse de 3 modulo $\varphi(n) = 112 \times 130 = 14\,560$ qui vaut $d = 9\,707$.
Le condensé h d'un document à signer et 10 000. Pour le signer, le détenteur de la clé privée effectue $\sigma = h^d = 10\,000^{9\,707} = 618 \pmod{14\,803}$.
Pour vérifier cette signature, on effectue $\sigma^3 = 618^3 = 10\,000 \pmod{14\,803}$ et on vérifie qu'il s'agit bien de la valeur du condensé du document qui a été signé.

10.4. La signature ElGamal (1985)

Ce mécanisme de signature repose sur la difficulté du problème du logarithme discret. Le standard DSA (*Digital Signature Algorithm*) est une modification de ce procédé.

Paramètres :

- un nombre premier p et un générateur g du groupe multiplicatif de $\mathbb{Z}/p\mathbb{Z}$.
- La clé privée est un entier non nul x compris au sens large entre 1 et $p-2$.
- La clé publique est $y = g^x$ modulo p.

Calcul d'une signature :

1. Choisir un entier k aléatoire, $1 \leq k \leq p-1$, premier avec $p-1$.
2. Calculer $r = g^k$ modulo p ;
3. Résoudre l'équation dite *équation de signature* d'inconnue s : $h = xr + ks \pmod{p-1}$. L'entier k étant par hypothèse inversible dans $\mathbb{Z}/(p-1)\mathbb{Z}$, la solution de l'équation de signature est $s = (h - xr)k^{-1} \pmod{p-1}$.

La signature du message de condensé h est le couple (r, s).

Vérification d'une signature. Cela consiste à tester l'égalité $g^h = y^r r^s (= g^{xr+ks})$ dans $\mathbb{Z}/p\mathbb{Z}$.

Exemple 1.

- *Paramètres*: $p = 11$ et $g = 2$ est générateur du groupe multiplicatif de $\mathbb{Z}/11\mathbb{Z}$.
- *Clé privée*: $x = 8$.
- *Clé publique*: $y = 2^8 = 256 = 3$ modulo 11.
- *Signature* Un message a un condensé $h = 7$. Le signataire tire au hasard un entier k, premier avec 11 dans $\{1, \ldots, 22\}$, par exemple $k = 3$. Cette valeur est bien inversible modulo 10. Son inverse est 7 ($3 \times 7 = 21$).
  Soient $r = 2^3 = 8$ et $s = (7-4) \times 7 = 2 = 3 \times 7 = 21 = 1$ modulo 10. La signature est le couple $\sigma = (11, 2)$.
- *Vérification* D'une part $g^h = 2^7 = 128 = 7$ modulo 11. D'autre part $y^r \times r^s = 3^8 \times 8^1 = 6\,561 \times 8 = -6 \times 8 = -48 = -4 = 7$ modulo 11. On a bien $2^7 = 3^8 \times 8^1$ modulo 11. La signature est correcte.

Exemple 2.

- *Paramètres*: $p = 23$ et $g = 5$ est générateur du groupe multiplicatif de $\mathbb{Z}/23\mathbb{Z}$.
- *Clé privée*: $x = 3$.
- *Clé publique*: $y = 5^3 = 10$ modulo 23.
- *Signature* Un message a un condensé $h = 7$. Le signataire tire au hasard un entier $k \in \{1, \ldots, 22\}$, premier avec 22, par exemple $k = 9$. Cette valeur est bien inversible modulo 22. Son inverse est 5.
  La signature est le couple (r, s) avec $r = 5^9 = 11$ dans $\mathbb{Z}/23\mathbb{Z}$ et $s = 5(7 - 3 \times 11) = 2$ dans $\mathbb{Z}/22\mathbb{Z}$, par conséquent, la signature est $\sigma = (11, 2)$.
- *Vérification* D'une part $5^7 = 17$ modulo 23. D'autre part $10^{11} \times 11^2 = 22 \times 6 = 17$ modulo 23. On a bien $5^7 = 10^{11} \times 11^{20}$ dans $\mathbb{Z}/23\mathbb{Z}$.

Sécurité du schéma de signature ElGamal.
Un adversaire qui sait calculer le logarithme discret dans $\mathbb{Z}/p\mathbb{Z}$ sait bien sûr forger de fausses signatures puisqu'il peut retrouver la clé privée. On ne sait pas si la réciproque est vraie.

Des précautions d'emploi sont à prendre, sans lesquelles aucune sécurité n'est assurée :

- Deux messages différents ne doivent pas être signés avec le même aléa k (voir TD).
- La valeur du nombre aléatoire k ne doit être ni révélée ni prédictible (voir TD).
- La valeur de r doit absolument être le représentant compris entre 0 et p dans $\mathbb{Z}/p\mathbb{Z}$ (voir TD).

BIBLIOGRAPHIE

BRIAN BECKETT, *Introduction aux méthodes de la cryptologie*, Ed. Masson, 1990.

RICHARD CRANDALL, CARL POMERANCE, *Prime Numbers, A computational perspective*, Ed. Springer, 2001.

MICHEL DEMAZURE, *Cours d'algèbre, primalité, divisibilité, codes*, Ed. Cassini, 1997.

DAVID KAHN, *The Codebreakers, The Comprehensive History of Secret Communication from Ancient Times to the Internet*, Ed. Hardcover, 1967–1996.

ALFRED J. MENEZES, PAUL C. VAN OORSCHOT, SCOTT A. VANSTONE, *Handbook of applied cryptography*, CRC Press, 1996.

PATRICE NAUDIN, CLAUDE QUITTÉ, *Algorithmique algébrique*, Ed. Masson, 1992.

SIHEM MESNAGER, *Chapitre Corps finis et codage*, dans le livre *"Mathmatiques L2, cours complet avec 700 tests et exercices"*, Editon Pearson, 2007.

BRUCE SCHNEIER, *Cryptographie appliquée*, Ed. International Thomson Publishing France, 1997.

ALAIN POLI, PHILIPPE GUILLOT, *Algèbre et protection de l'information*, Ed. Hermès, 2005.

VICTOR SHOUP, *A computational Introduction to Number Theory and Algebra*, Cambridge University Press, 2005

DOUGLAS STINSON, *Cryptographie, théorie et pratique*, Ed. Vuibert, 1996

GILLES ZÉMOR, *Cours de cryptographie*, Ed. Cassini 2000.