République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A. Mira-Béjaia

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Faculté de Technologie
Département de Génie électrique

Polycopié pédagogique

Intitulé :

Cryptographie et sécurité réseaux

Cours déstiné aux étudiants de

Master 2 : Réseaux et Télécommunications

Réalisé par :
Dr. BENAMIROUCHE Nadir

Année : 2021

AVANT-PROPOS

Ce polycopié de cours «Cryptographie et sécurité réseaux», demande avant tout, un certain entendement de la théorie de l'information, des connaissances de base des réseaux informatiques et des mathématiques, et principalement, la maitrise des notions de l'algèbre.

Cependant, dans les annexes, une grande partie sur les principes et les outils mathématiques utilisés (algèbre, arithmétique, algorithmique, complexité,...) est introduite.

Toutefois, le premier chapitre se voit comme une introduction à la cryptographie, qui est essentiellement consacré pour introduire et illustrer les concepts de base de la cryptographie classique.

En outre, dans le second chapitre, des déscriptions détaillées sur les cryptosystèmes modernes, à savoir, les systèmes de chiffrement symétrique et asymétrique, les fonctions de hachage et les courbes elliptiques.

De plus, un troisième chapitre sur les principes fondamentaux de cryptanalyse et types d'attaques sur les différents cryptosystèmes sont aussi évoqués.

Enfin, dans le quatrième chapitre une variété de concepts et problèmes liés à la protection et la sécurité des systèmes d'information et des réseaux sont introduits. Il s'agit des problèmes informatiques, risques, menases, les protocoles et les normes de sécurité les plus utilisés.

Le cours de la cryptographie et sécurité réseaux se veut pour objectif primordial de donner aux étudiants ayant assistés, des techniques, des conceptions et d'exécutions des cryptosystèmes, de l'antiquité à nos jours, afin que ces derniers puissent assimiler la notion de sécurité dans la mise en place des systèmes informatiques dont ils ont vocation d'implémenter.

Table des matières

Table des figures

Introduction

De toute évidence, le monde d'aujourd'hui connaît des avancées très significatives dans le domaine des télécommunications ; par conséquent, les besoins en matière de sécurité réseaux sont un peu plus impérieux, et la prédisposition n'est forcément pas à la baisse.

Depuis quelques années déjà, on assiste à un développement constant des technologies de l'information qui ne cessent de remettre en cause certains protocoles sécuritaires, tandis que, d'autres ont vu le jour. Ainsi, de nouvelles techniques et protocoles de sécurité ont été mis en place ; qu'il s'agisse des techniques visant à sécuriser l'échange des données. D'où, la sécurité des données tend à s'améliorer. Et comme prône ce proverbe chinois : «l'art de la guerre est basé sur la tromperie», de même par analogie, la sécurité réseaux doit représenter une stratégie qui éradique cette tromperie.

Il est sans ignorer que, les équipements réseaux sont quasiment partout. En effet, d'une part le matériel est accessibles à un prix très abordable, et d'autre part, les logiciels tendent à se simplifier et permettent une prise en main rapide. En plus, les entreprises, informatisées, nécessitent davantage un réseau sécurisé pour le transfert des données aussi bien entre les machines de ladite entreprise qu'avec des hôtes externes. Cela étant, la sécurité de façon générale est présente à plusieurs niveaux, quelque soit la portée de l'information.

Ce cours est inspiré des cours de, Crystoff PAAR, [1], François ARNAULT, [2], Jean-Guillaume DUMAS et al., [3], Renaud DUMONT [4].

Des articles de techniques d'ingénieur, à savoir : Frédéric RAYNAL, [5], Thomas NOEL, [6], Daniel TREZENTOS [7], Claude CHIARAMONTI, [8].

Des livres comme, Douglas Stinson, [9], Gilles DUBERTRET, [10], Serge VAUDENAY [11], Damien VERGNAUD, [12], Bruno MARTIN, [13], Neal KOBLITZ, [14], John DAEMEN et al., [15], Lawrence WASHINGTON, [16], Jean-Guillaume DUMAS [17], Didier MULLER, [18], et de récits de WIKIPEDIA et du site Web, www.frameip.com.

Chapitre I

Introduction à la cryptographie

Contents

I.1 Cryptographie classique

On distingue traditionnellement deux systèmes : symétrique et asymétrique. En cryptographie conventionnelle, aussi appelée cryptographie symétrique (ou à clé secrète), les correspondants partagent la même clé pour chiffrer et déchiffrer.

I.1.1 Définition

La *cryptographie* est un outil qui se charge de régler les problèmes d'interception des informations échangées lors d'une transaction faite à travers un *canal* peu sûr.

Elle sert à protéger les données sensibles lors de leurs transmissions. Dans la terminologie cryptographique, le message à envoyer est appelé *text clair*.

Le processus de transformation d'une information claire en une information inintelligible *texte chiffré* est appelé *chiffrement*, inversement, le *déchiffrement* représente le processus de reconstruction du texte en clair à partir du texte chiffré grâce à la *clé* de déchiffrement.

Les systèmes cryptographiques reposent sur les principes de Kirchhoff énoncés comme suit :

I.1.2 Objectifs de la cryptographie

L'objectif fondamental de la cryptographie est de permettre à deux personnes appelées traditionnellement, *Alice* et *Bob* de communiquer à travers un canal peu sûr de telle sorte qu'un opposant passif *Eve* ne puisse pas comprendre ce qui est échangé et que les données échangées ne puissent pas être modifiées ou manipulées par un opposant actif.

La cryptographie utilise des concepts issus de nombreux domaines (Informatique, Mathématiques, Electronique). Toutefois, les techniques évoluent et trouvent aujourd'hui régulièrement racine dans d'autres branches (Biologie, Physique, etc.)

En effet, la cryptographie a pour but de garantir la protection des communications transmises sur un canal public contre différents types d'adversaires. La protection des informations se définit en termes de confidentialité, d'intégrité, d'authentification et de la nonrépudiation.

- **La confidentialité** : est la propriété qui permet de conserver les données secrètes et empêcher tout accès aux individus non autorisés.
- **L'intégrité** : permet de vérifier que les données n'ont subi aucune altération volontaire ou involontaire lors de leurs parcours par une entité tierce.
- **L'authentification** : le destinataire d'un message doit vérifier que la source des données est bien l'identité prétendue.
- **La non-répudiation** : c'est un mécanisme pour enregistrer un engagement ou un acte d'une entité lors de l'envoi des données, de telle sorte que celle-ci ne puisse pas nier avoir accompli cet acte.

La confidentialité est assurée par le chiffrement, par contre l'authenticité, l'intégrité et la non répudiation sont vérifiées par une signature numérique ; qui est considérée comme étant une version électronique d'une signature manuscrite. Nous pouvons décrire cette signature comme un code rattaché aux données qui sert de preuve que le message n'a été trafiqué d'aucune sorte entre l'expéditeur et le destinataire.

I.1.3 Principes de Kerckhoffs

En 1883 dans un article paru dans le Journal des sciences militaires, [19], Auguste Kerckhoffs (1835-1903) posa les principes de la cryptographie classique qui sont valables même à l'ère de la cryptographie moderne.

- Un cryptosystème sera d'autant plus résistant et sûr qu'il aura été conçu, choisi et implémenté avec la plus grande transparence et soumis ainsi à l'analyse de l'ensemble de la communauté cryptographique.
- Si un algorithme est supposé être secret, il se trouvera toujours quelqu'un soit pour vendre l'algorithme, soit pour le percer à jour, soit pour en découvrir une faiblesse ignorée de ses concepteurs. A ce moment là c'est tout le cryptosystème qui est à changer et pas seulement la clé. Les systèmes conçus dans le secret révèlent souvent rapidement des défauts de sécurité qui n'avaient pas été envisagés par les concepteurs.

Ces principes et en particulier le second, stipulant entre autre, que la sécurité d'un cryptosystème ne doit pas reposer sur le secret de l'algorithme de chiffrement mais qu'elle doit uniquement reposer sur la clef secrète du cryptosystème qui est un paramètre facile à changer, de taille réduite (actuellement de 64 à 2048 bits suivant le type du cryptage et la sécurité demandée) et donc assez facile à transmettre secrètement.

D'ailleurs, ce principe très exactement a été très respecté par le NIST *(National Institute of Standards and Technology)* dans le choix du dernier standard de chiffrement le plus puissant au mode d'aujourd'hui, l'algorithme symétrique AES *(Advanced Encryption Standard)*, [20]. Ce dernier a été choisi à la suite d'un appel d'offre international et tous les détails de conception sont rendus publiquement accessibles.

I.2 Concepts cryptographiques de base

D'une manière formelle, un système cryptographique est quintuplet $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ d'ou :

1. $\mathcal{P}$ est un ensemble fini de blocs de **textes clairs** possibles.
2. $\mathcal{C}$ est un ensemble fini de blocs de **textes chiffrés** possibles.
3. $\mathcal{K}$ est un ensemble fini de **clefs** possibles.
4. Pour tout $k \in \mathcal{K}$, il y a une **règle de chiffrement** $e_k \in \mathcal{E}$ et une **règle de déchiffrement** correspondante $d_k \in \mathcal{D}$:
   Chaque $e_k : \mathcal{P} \to \mathcal{C}$ et $d_k : \mathcal{D} \to \mathcal{P}$ sont des fonctions satisfaisant la propriété de base de la cryptographie telle que : $d_k(e_k(M)) = M$ pour tout texte clair $M \in \mathcal{P}$.

La principale propriété est la quatrième. Elle précise que si un texte clair x est chiffré en utilisant e_K, et si le texte chiffré y obtenu est ensuite déchiffré en utilisant d_K, on retrouve le texte clair x original.

Alice et Bob peuvent employer le protocole suivant pour utiliser un système cryptographique spécifique.
Tout d'abord, ils choisissent une clef quelconque $K \in \mathcal{K}$. Cette opération est effectuée lorsqu'il se rencontre en un même endroit et ne sont pas observés par Oscare, ou bien lorsqu'ils disposent d'un canal de communication sûr, auquel cas ils peuvent être à des endroits éloignés. Supposons qu'Alice souhaite ensuite communiquer un message à Bob par canal peu sûr, ce message étant chaîne, $x = x_1 x_2 \dots x_n$

Pour un entier $n \geq 1$, où chaque bloc $x_1 \in \mathcal{P}, 1 \leq i \leq n$. Chaque x_i est chiffré en utilisant la règle de chiffrement e_K spécifiée par la clé de chiffrement K choisie. Ainsi, Alice calcule $y_i = e_K(x_i), 1 \leq i \leq n$, et la chaîne chiffrée obtenue $y = y_1 y_2 \dots y_n$
est envoyée dans le canal. Lorsque Bob reçoit $y_1 y_2 \dots y_n$, il la déchiffre en utilisant la fonction de déchiffrement d_K et récupère le texte clair original $x_1 x_2 \dots x_n$. Le canal de communication est illustré sur la figure 1.1.
Il est évident que chaque fonction de chiffrement e_K doit être injective (c'est-à-dire ne pas chiffrer deux blocs différents en deux valeurs égales), sinon, le procédé de déchiffrement ne pourrait être fait sans ambiguÏté. Plus précisément, si

$y = e_K(x_1) = e_K(x_2)$

avec $x_1 \neq x_2$, Bob n'a aucun moyen de savoir si y doit être déchiffré en x_1 ou en x_2. On note ainsi que si $\mathcal{P} = \mathcal{C}$, chaque fonction de chiffrement doit être une permutation.
Autrement dit, si les espaces des blocs de messages clairs et chiffrés sont identiques, chaque fonction de chiffrement réarrange (ou permute) les éléments de cet espace.
Identifions l'alphabet usuel avec $\mathbb{Z}_{26}$ par $a = 0$, $b = 1, ..., z = 25$. On a alors les systèmes cryptographiques suivants :

I.2.1 Le chiffrement par décalage

Dans cette section, on décrit le **chiffrement par décalage**, basé sur l'arithmétique modulaire.

Supposons que l'on dévise a et b par m. On obtient des quotients et des restes, ceux-ci étant compris entre 0 et $m-1$. Précisément, on a $a = q_1 m + r_1$ et $b = q_2 m + r_2$ avec $0 \leq r_1 \leq m-1$ et $0 \leq r_2 \leq m-1$. Ainsi, il est facile de voir que $a \equiv b \ (mod\ m)$ si, et seulement si $r_1 = r_2$. On utilisera la notion $a \ mod\ m$ (sans parenthèses) pour représenter le reste dans la division de a par m. On a donc $a \equiv b \ (mod\ m)$ si, et seulement si, $a \ mod\ m = b \ mod\ m$. Si l'on remplace a par $a \ mod\ m$, on dit que l'on *réduit a modulo m*.

Définition I.2.1.1 *Si a, b et m sont des entiers, et si $m > 0$, on écrit $a \equiv b \ (mod\ m)$ si m divise $(b - a)$ que l'on note $m \ (a - b)$. L'écriture $a \equiv b \ (mod\ m)$ se lit a est congru à b modulu m. L'entier m est appelé modulus.*

Description I.2.1 *Soit $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$. Pour $0 \leq K \leq 25$, on définit la fonction de chiffrement par :*
$e_K(x) = x + k \ mod\ 26$, tandis que, $d_K(y) = y - k \ mod\ 26$ pour le déchiffrement.

Remarque. *Le chiffrement par décalage se définit dans $\mathbb{Z}_{26}$, car on utilise 26 lettres dans l'alphabet, mais on pourrait le définir sur n'importe quel $\mathbb{Z}_m$. Pour la clé particulière $k = 3$, le système cryptographique est souvent appelé **chiffrement de César**, car il était utilisé par Jules César.*

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

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

I.2.2 Le chiffrement par substitution

Description I.2.2 *Soit $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$. $\mathcal{K}$ est l'ensemble des permutations sur l'ensemble des 26 nombres : $0, 1, ..., 25$. Pour chaque permutation $\pi \in \mathcal{K}$, on définit, $e_\pi(x) = \pi(x)$ et $d_\pi(y) = \pi^{-1}(y)$.*
ou π^{-1} est la permutation réciproque (inverse) de π.

Exemple : *Considérons la permutation donnée par le tableau suivant,*

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

Le texte clair `cryptographie` *est chiffré en* `MDQFERUDPFJNI`*.*

I.2.3 Le chiffrement affine

Description I.2.3 *Soit $\mathcal{P} = \mathcal{C} = ZZ_{26}$ et soit $K = (a, b) \in ZZ_{26} \times ZZ_{26} : pgcd(a, 26) = 1$. Pour $K = (a, b) \in \mathcal{K}$, on définit, $e_k(x) = ax + b mod\ 26$, et $d_k(y) = a^{-1}(y - b) mod\ 26$. Tel que, $(x, y) \in ZZ_{26})$*

Exemple : *Pour $a = 7$ et $b = 11$ le texte clair* `cryptographie` *est chiffré en* `ZAXMOFBALMIPN`*.*

I.2.4 Le chiffrement de Vigenère

Description I.2.4 *Soit $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}^m$ ($m \in N^*$). Soit $K = (k_1, ..., k_m) \in \mathcal{K}$ alors $e_k(x_1, ..., x_m) = (x_1 + k_1, ..., x_m + k_m)$ et $d_k(y_1, ..., y_m) = (y_1 - k_1, ..., y_m - k_m)$.*

Exemple : *Pour $m = 3$ et $k = cle = (2, 11, 4)$, le texte clair* `cryptographie` *est chiffré en* `ECCRESICERSMG`*.*

I.2.5 Le chiffrement de Vernam

En 1917, pendant la première guerre mondiale, la compagnie américaine AT&T *(American Telephone & Telegraph)* avait chargé le scientifique Gilbert Vernam d'inventer une méthode de chiffrement que les Allemands ne pourraient pas casser. Le chiffrement de vernam connu aussi sous l'appelation *masque jetable* (One time pad).
Le masque jetable est le seul code connu à l'heure actuelle comme mathématiquement prouvé sûr.
Le système à clef jetable est un système cryptographique dit à clef secrète, c'est-à-dire que le secret réside dans un paramètre des fonctions de chiffrement et de déchiffrement connu uniquement de l'émetteur et du destinataire. C'est aussi le cas du chiffrement de César, dans lequel le paramètre est la taille du décalage des lettres ou des nombres.
Il est aussi identique au chiffrement de Vigenère mais on utilise la clé q'une seule fois, ce qui contraint à choisir une clé aussi longue que le message à transmettre.

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

I.2.6 Le chiffrement de Hill

Description I.2.5 *L'espace des clés est l'ensemble $GL_m(\mathbb{Z}_{26})$ des matrices inversibles d'ordre m à coefficients dans $\mathbb{Z}_{26}$. Soit $k \in \mathcal{K}$. On a alors $e_k(x_1, ..., x_m) = (x_1, ..., x_m)k$ et $d_k(y_1, ..., y_m) = (y_1, ..., y_m)k^{-1}$.*

Exemple : *Pour $m = 2$ et $k = \left(\begin{smallmatrix} 4 & 3 \\ 9 & 7 \end{smallmatrix}\right)$ le texte clair* `cryptographe` *est codé numériquement par ((2,17),(24,15),(19,14),(6,17),(0,15),(7,4)). Ainsi, il est chiffré alors en ((5,21), (23,21), (20,25), (21,7), (5,1),(12,23)), ce qui correspond finalement a* `FVXVUZVHFBMX`*.*

I.2.7 Le chiffrement par permutation

Description I.2.6 *Soit $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}^m$ $(m \in \mathbb{N}^*)$ et $\mathcal{K}$ l'ensemble des permutations de $\mathbb{Z}_m$. Pour $k \in \mathcal{K}$, on a alors $e_k(x_1, ..., x_m) = (x_{k(1)}, ..., x_{k(m)})k$ et $d_k(y_1, ..., y_m) = (y_{k^{-1}(1)}, ..., y_{k^{-1}(m)})$. On peut voir ce système comme un cas particulier du chiffrement de Hill.*

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

I.2.8 Le chiffrement en chaîne

Description I.2.7 *Soit $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}^m$ $(m \in \mathbb{N}^*)$. La clé $k = (k_1, ..., k_m)$ est un élément de $\mathbb{Z}_{26}^m$ donné par sa première composante k_1 et par une règle qui permet de calculer chaque $k_i (i > 1)$ en fonction de k_{i-1} et $x_1, ..., x_{i-1}$. La portion de k_i servira alors à chiffrer x_i ;*

Exemple : *On pourra poser $y_i = x_i + k_i$. Si la suite des k_i est périodique, on parle de chiffrement en chaîne périodique et c'est une généralisation du chiffrement de Vigenère.*

I.3 Le chiffrement par flots

C'est une méthode de chiffrement dont on possède la preuve de la sécurité. C'est d'ailleurs la seule méthode connue pour laquelle on ait prouvé son caractère parfait. Mais pour se servir de cette méthode en pratique, il faut savoir générer des clefs aléatoires, et ce problème est loin d'être trivial. D'autre part, comme la clef n'est pas réutilisable, les protocoles d'échange des clefs entre émetteurs et destinataires restent problématiques. Nous allons voir dans la section suivante comment construire des codes qui réutilisent une clef unique et rendent le protocole d'échange moins fastidieux. Sans être parfaits, ces codes tendent vers cette propriété.

De ce fait, il existe deux types de chiffrement à clé symétrique :

Le chiffrement par blocs : l'opération de chiffrement s'effectue sur des blocs de texte clair (ex : le DES *(Data Encryption Standard)* avec des blocs de 64 bits). L'idée sera de découper le message en blocs et de chiffrer chaque bloc séparément.

Le chiffrement par flots (ou par stream ou de flux) : l'opération de chiffrement s'opère sur chaque élément du texte clair (caractère, bits). On chiffre un bit/caractère à la fois. La structure d'un chiffrement par stream repose sur un générateur de clés qui produit une séquence de clés aléatoires.

La deuxième idée est de générer des nombres aléatoires de bonne qualité, ou plus précisément des nombres pseudo-aléatoires. Cela permet de réaliser un chiffrement bit à bit (par flot) comme illustré sur la figure I.1.

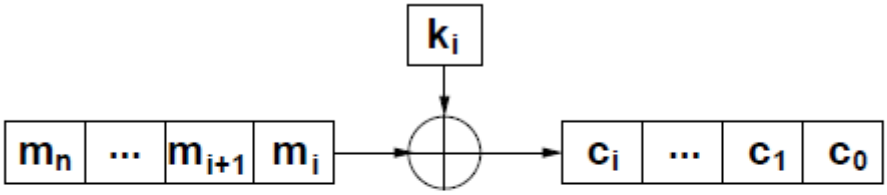

FIGURE I.1 – Mode de chiffrement bit à bit (par flot).

I.3.1 Les registres à décalage à rétroaction linéaire

Une généralisation des générateurs congruentiels linéaires consiste à ne plus utiliser seulement la précédente valeur pour fabriquer l'élément suivant de la séquence, mais plusieurs des précédentes valeurs, c'est-à-dire que x_n est calculé par des combinaisons linéaires de $x_{n-1}, ..., x_{n-k}$. Autrement dit :

$$x_n = (a_1 x_{n-1} + ... + a_k x_{n-k}) \quad mod\, m$$

Ces générateurs sont particulièrement intéressants si m est un nombre premier car leur période maximale est alors $m^k - 1$. Ainsi il est possible, même avec un petit modulo, d'avoir de très grandes périodes.
Par exemple, pour générer des suites aléatoires de bits, on choisit $m = 2$. Dans ce cas, les opérations peuvent être réalisées très rapidement sur machine par des *ou exclusifs* (`xor`) pour l'addition modulo 2 et par des décalages des bits x_i pour générer les bits suivants. Il existe même des puces spécialisées réalisant les opérations nécessaires. On parle alors de registres à décalage linéaire, abrégé par LFSR *(Linear Feedback Shift Registers)*. La figure I.2 résume le fonctionnement.

Pour certains calculs, il est intéressant d'écrire LFSR par une forme polynomiale : soit $\prod(X) = X^k - a_1 X^{k-1} - ... a_k$. Ainsi, $LFSR_{\prod}(x_0, ... x_{k-1})$ désigne la séquence infinie des x_i linéairement générée par le polynôme $\prod$ avec pour k valeurs initiales les $x_0, ..., x_{k-1}$.

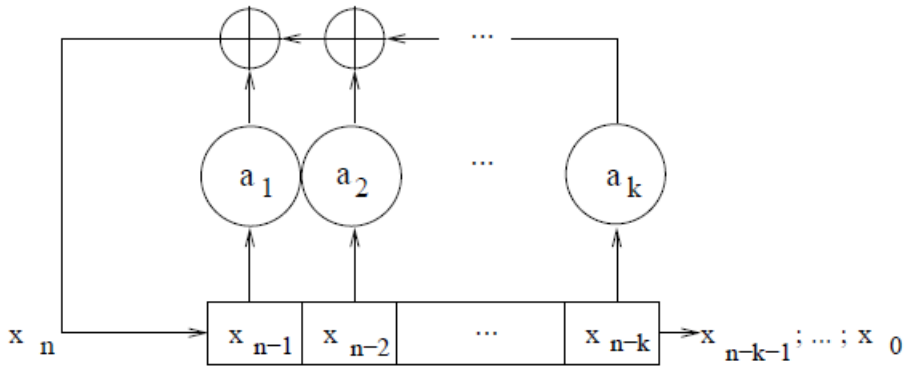

FIGURE I.2 – Schéma de fonctionnement d'un LSFR.

I.4 Le chiffrement par blocs

On peut coder chaque bloc d'un message par un même algorithme de façoon indépendante pour chaque bloc. C'est ce qui est appelé le mode de codage ECB, pour *Electronic Code Book*. Plus généralement, cette indépendance de codage entre les blocs n'est pas requise et les différentes façons de combiner les blocs sont appelés *modes de chiffrement*.

I.4.1 Le mode ECB : (Electronique Code Book)

Dans ce mode, le message M est découpé en blocs m_i de taille fixe. Chaque bloc est alors crypté séparément par une fonction E_k, paramétrée par une clef k comme sur la figure I.3.

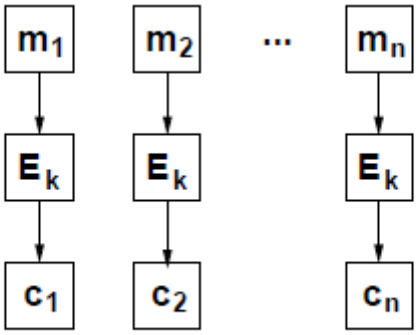

FIGURE I.3 – Mode de chiffrement par blocs ECB.

Ainsi un bloc de message donné m_i sera toujours codé de la même manière. Ce mode de chiffrement est le plus simple mais ne présente donc aucune sécurité et n'est normalement jamais utilisé en cryptographie.

I.4.2 Le mode CBC : (Cipher Bloc Chaining)

Le mode CBC a été introduit pour qu'un bloc ne soit pas codé de la même manière s'il est rencontré dans deux messages différents. Il faut ajouter une valeur initiale C_0 (ou IV pour *initial value*), aléatoire par exemple. Chaque bloc est d'abord modifié par XOR avec le bloc crypté précédent avant d'être lui même crypté conformément à la figure I.4 par :

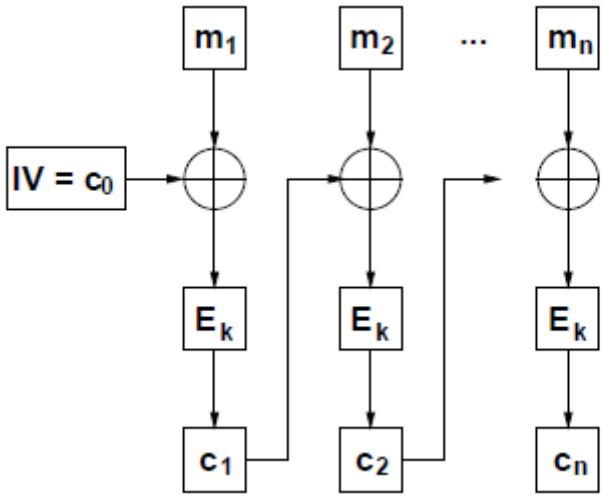

FIGURE I.4 – Mode de chiffrement par blocs CBC.

$$c_i = E_k(m_i \oplus c_{i-1}) \tag{I.1}$$

C'est le mode de chiffrement le plus utilisé. Le déchiffrement nécessite l'inverse de la fonction de codage $D_k = E_k^{-1}$ pour déchiffrer : $m_i = c_{i-1} \oplus D_k(c_i)$.

I.4.3 Le mode CFB : (Cipher FeedBack)

Pour ne pas avoir besoin de la fonction inverse pour décrypter, il est possible de faire un XOR après le cryptage, c'est le principe du mode CFB, comme on peut le voir sur la figure I.5.

$$c_i = m_i \oplus E_k(c_{i-1}) \tag{I.2}$$

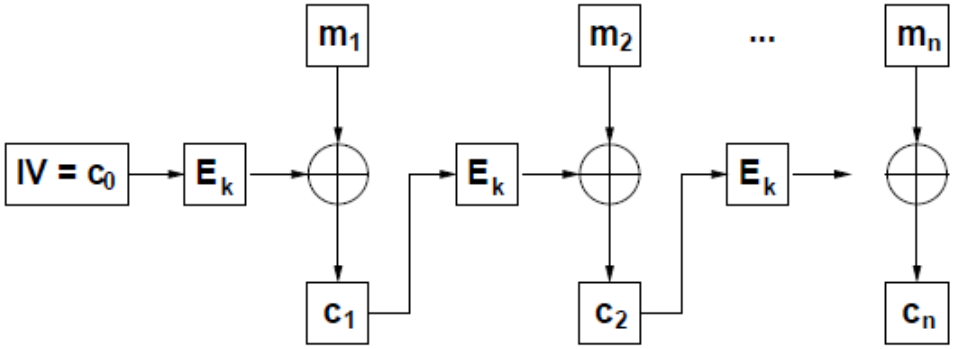

FIGURE I.5 – Mode de chiffrement par blocs CFB.

L'intérêt de ce mode est que le déchiffrement ne nécessite pas l'implémentation de la fonction D_k : $m_i = c_i \oplus E_k(c_{i-1})$. Ce mode est donc moins sûr que le CBC et est utilisé par exemple pour les cryptages réseaux.

I.4.4 Le mode OFB : (Output FeedBack)

Une variante du mode précédent permet d'avoir un codage et un décodage totalement symétrique, c'est le mode OFB suivant la figure I.6.

$$z_0 = c_0\,;\; z_i = E_k(z_{i-1})\,;\; c_i = m_i \oplus z_i \tag{I.3}$$

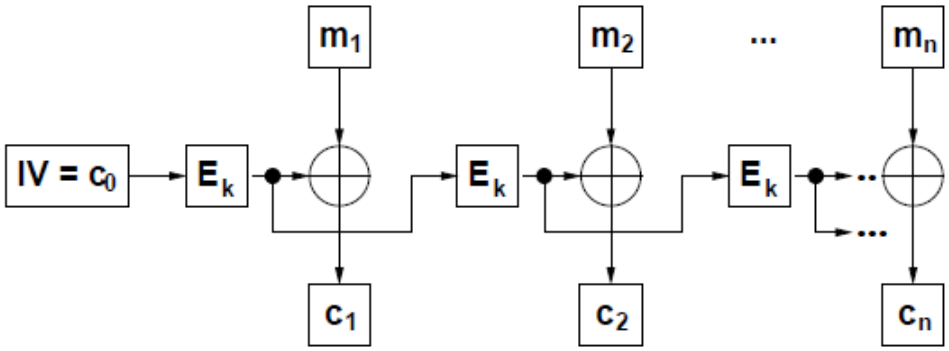

FIGURE I.6 – Mode de chiffrement par blocs OFB.

Ce qui se déchiffre par : $z_i = E_k(z_{i-1})\,;\; m_i = c_i \oplus z_i$. Ce mode est utile dans les satellites pour lesquels minimiser le nombre de circuits embarqués est crucial.

I.4.5 Le mode CTR : (Counter-mode encryption)

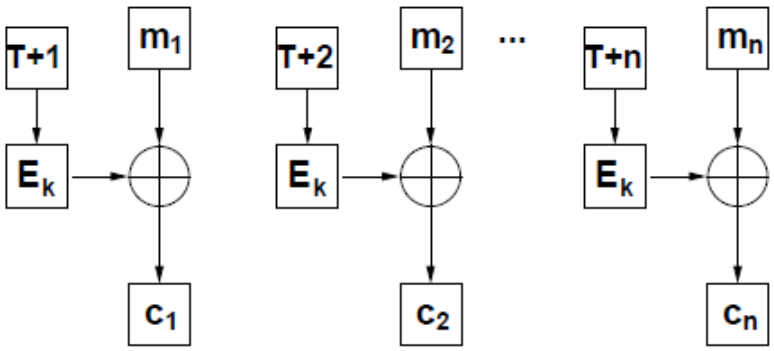

FIGURE I.7 – Mode de chiffrement par blocs CTR.

Ce mode est également totalement symétrique, mais en outre facilement parallélisable. Il fait intervenir le chiffrement d'un compteur de valeur initiale T :

$$c_i = m_i \oplus E_k(T + i) \tag{I.4}$$

Le déchiffrement est identique : $m_i = c_i \oplus E_k(T + i)$. L'intérêt d'un tel mode est que les différents calculs sont indépendants, comme pour le mode ECB, mais qu'un même bloc n'est *a priori* jamais codé de la même façon. Ainsi, le principe de chiffrement est donné par la figure I.7.

Chapitre II

Cryptographie moderne

Contents

Partie 1 :Systèmes cryptographiques à clé privée

II.1 Concept cryptographique

II.1.1 Historique

L'histoire de la cryptographie symétrique est très ancienne, mais les connaissances académiques dans ce domaine sont assez récentes.

Durant les années 70s, l'explosion des besoins de sécurité pour les données non-classifiées, c'est-à-dire, non militaires et non diplomatiques, ont poussé le NBS *(National Bureau of Standards)* des Etats-Unis à lancer un appel d'offre en 1973 avec un cahier des charges pour d'éventuelles propositions de cryptosystèmes.

En effet, cet appel d'offre a donné naissance au cryptosystème DES *(Data Encrytion Standard)* [21]. Il a été publié en 1975 et adopté par le NBS en 1977 comme standard de cryptage pour les applications non classifiées.

Il reprend les principes, schéma de Feistel, **(II.1.3)**, et une partie du système de cryptage d'IBM *(International Business Machines corporation)* denommé *Lucifer*.

Il est à la base d'autres cryptosystèmes plus récents comme IDEA *(International Data Encryption Algorithm)*, FEAL *(Fast Data Encipherment Algorithm)*, CAST *(Carlisle Adams et Stafford Tavares)*, RC5 *(Rivest's Cipher)*, et Blow-Fish.

Aujourd'hui le DES est remplacé par le AES *(Advanced Encrytion Standard)* de J. Daeme et V. Rijmen, [20], basé sur un principe un peu différent du moment qu'il utilise un réseau de substitution-permutation, avec une clé plus longue (128 à 256 bits), plus structuré et avec des fonctionnalités plus étendues. AES a été retenu en 2000 aprés un appel d'offre international.

II.1.2 Description du DES

C'est un système cryptographique produit , basé sur un schéma de Feistel, illustré par la figure, II.1.

Il compose des opérations de cryptage, autrement dit il effectue un produit d'opérations de cryptage. Il répète 16 fois un algorithme appelé la fonction d'étage qui dépend d'un paramètre la clef d'étage. La répétition de cet algorithme mélange les bits du message en clair en respectant les principes de C. Shannon, [22], : **Confusion et Diffusion**.

- *La confusion* gomme les relations entre le texte clair et le texte chiffré pour éviter les attaques par analyse statistique. Autrement dit un changement d'un seul bit dans le texte clair doit affecter un grand nombre de bits (idéalement tous) du texte chiffré.
- *La diffusion* disperse la redondance du texte clair dans le texte chiffré, par exemple deux lettres doublées ne doivent pas rester côte à côte après cryptage.

II.1.3 Schéma de Feistel

Un bon algorithme à clé secrète doit transformer le message clair en un message crypté qui ressemble autant que possible à une suite aléatoire, pour limiter au minimum les risques d'une attaque par analyse statistique du texte chiffré, de ses redondances, etc...
Le problème est que, si l'on sait depuis longtemps construire des fonctions qui ont l'air aléatoire, on ne savait pas avant les travaux de Feistel construire des bijections aléatoires. La solution apportée par Feistel est très élégante : on suppose par exemple qu'on a une fonction f presque aléatoire qui prend comme argument un mot de n bits, et renvoie un mot

de n bits (qui donne l'impression d'avoir été choisi au hasard). L'algorithme de chiffrement va procéder en chiffrant des blocs de $2n$ bits, qu'on partage en 2, partie gauche G, partie droite D. L'image du bloc (G, D) par le schéma de Feistel est le bloc (L, R), avec $L = D$, et $R = G \oplus (D)$ où $\oplus$ est l'opération XOR..
Cette transformation est cette fois bijective, car si on a un tel couple (L, R), on retrouve (G, D) par $D = L$ et $G = R \oplus f(L)$. Bien sûr, la partie droite n'a pas été transformée (juste envoyée à gauche). C'est pourquoi on répète le schéma de Feistel un certain nombre de fois (on parle de tour - le DES en comporte 16).
La plupart des algorithmes à clé secrète de la fin du XXème siècle étaient des schémas de Feistel. L'avénement de l'AES, qui n'en est plus un, marque la fin de la prédominance de tels algorithmes.

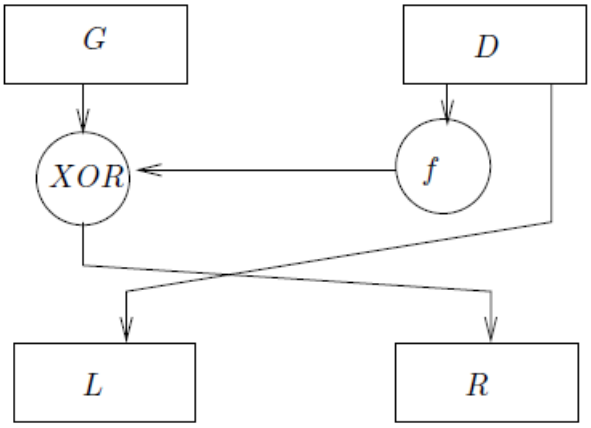

FIGURE II.1 – Schéma d'un tour de Feistel.

II.1.4 Organigramme du DES

DES utilise une clé K de 56 bits utiles, en fait une clé de 64 bits dont les bits 8, 16, 24, 32, 40, 48, 56 ne sont pas utilisés pour la clé mais participent à un code correcteur qui permet de vérifier que la clé n'a pas été altérée.
Le DES est un cryptosystème par blocs qui travaille sur des blocs de 64 bits. La clé de DES est trop courte pour les puissances de calcul actuelles. La taille de la clé secrète est de 56 bits, ce qui l'a rend aujourd'hui vulnérable aux attaques par force brute.

D'ailleurs, en 1997 la clé a été cassée en 3 semaines par une fédération de machines sur internet. Puis, elle a été cassée en 56 heures, en 1998.

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

Afin de prolonger la durée de vie de DES en attendant l'arrivée du AES, on a introduit le triple DES *(3DES)*. Il consiste à appliquer successivement au message DES muni de la clef K, à le décoder avec DES muni de la clef K' et de le recoder avec DES muni de la clef K. Au total tout se passe comme si on avait codé le message avec une clé nettement plus longue. C'est un système de chiffrement itéré à 16 étages

1. La clé K donne naissance à 16 sous-clés, les clefs d'étage, de 48 bits $K_1, K_2, ..., K_{16}$

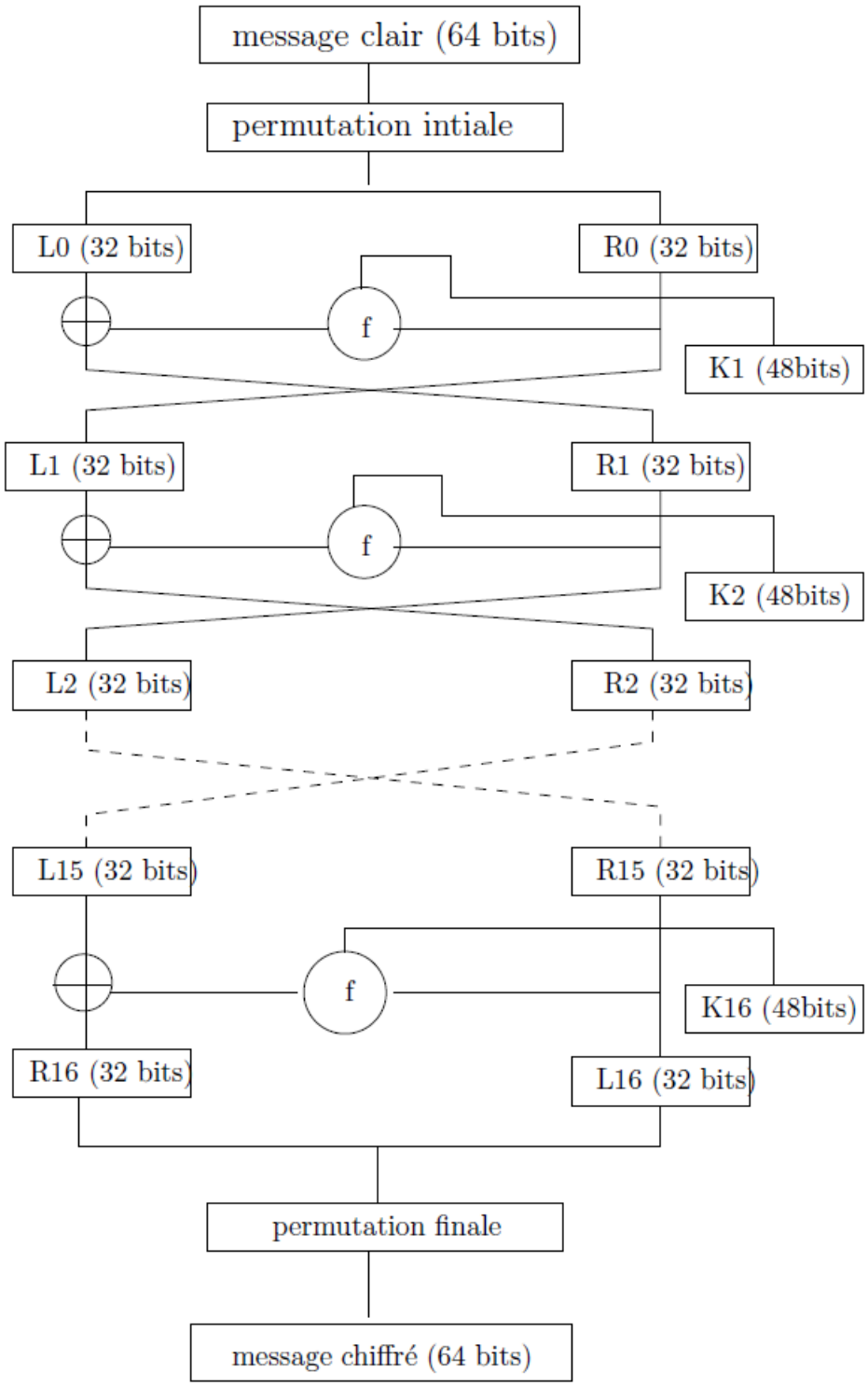

FIGURE II.2 – Architecture du DES.

2. Etant donné un bloc de texte clair de 64 bits, X, on construit, grâce à la fonction d'étage g_0, un nouveau texte, X_0, par permutation de l'ordre des bits grâce à la **permutation initiale** IP.
$$g_0(X) = X_0 = IP \tag{II.1}$$
3. on sépare les 32 bits de gauche, L_0, et les 32 bits de droites, R_0, de X_0, donc $X_0 = L_0R_0$
4. On effectue 16 itérations (ou tours ou étages). On calcule la valeur de la fonction d'étage $g(X_{i1}, K^i) = L_iR_i$, $1 \leq i16$ suivant la règle
$$L_i = R_{i-1}R_i = L_{i-1} \leq f(R_{i-1}, K_i) \tag{II.2}$$
où $\oplus$ est la somme bit à bit sans retenue ou ou exclusif ou XOR dans $\mathbb{Z}/2\mathbb{Z}$ de L_{i1} et $f(R_{i1}, K_i)$ et où f est une fonction non linéaire qui participe à la diffusion et qui est décrite par les **S-boîtes**
5. Pour finir on applique la permutation inverse de la permutation initiale, IP^{-1}, à $R_{16}L_{16}$.

Le schéma de la figure II.2 résume ce qui précède.

II.1.5 Description de l'AES.

Le principe de l'AES est très proche de celui du DES. C'est aussi un système cryptographique constitué d'une suite d'opérations de permutation et de substitution. Contrairement à AES ce n'est pas un schéma de Feistel mais basé sur un **réseau de substitution-permutation**.

L'AES opère sur des blocs de 128 bits avec des clés de longueurs de, 128, 192 ou de 256 bits.

A l'origine AES pouvait travailler sur des blocs de longueur $N_b \times 32$ bits où N_b variait de 4 à 8, finalement la taille de blocs d'AES a été fixée à 128 bits et donc N_b a été fixé à 4. Le passage à une clé de 128 bits minimum rend impossible dans le futur prévisible les recherches exhaustives de clefs. Si on suppose que l'on a un algorithme capable de comparer en une seconde 2^{56} clefs (i.e de casser DES en une seconde) il lui faudra 149 mille milliards d'années pour casser AES.

II.1.6 Organigramme de l'AES

Pour toute la partie mathématique on pourra consulter dans les rappels mathématiques dans le chapitre 1, la section sur les extensions de corps et les anneaux de polynômes sur un corps fini, **(A.1.3)**.

Le AES est un système de chiffrement itéré constitué d'un algorithme de chiffrement sur des blocs de 128 bits, la fonction d'étage, répété N_e fois, avec N_e allant de 10 à 14, et d'une clef secrète de 128, 192 ou 256 bits et pour chaque étage d'une clef d'étage de longueur fixe, 128 bits. Un algorithme de diversification de clef, **ExpandKey**$[i]$, permet de créer une clef pour chacun des étages, i, à partir de la clef secrète K.

On notera N_e le nombre d'étages. La fonction d'étage, g, est l'ensemble des opérations que l'on fait subir au texte qui arrive à l'étage i. Elle est la même pour tous les étages sauf le dernier. La fonction g consiste en l'application successive de 4 opérations **SubBytes, ShiftRows, MixColumns, AddRoundKey**. Pour le dernier étage on applique seulement les

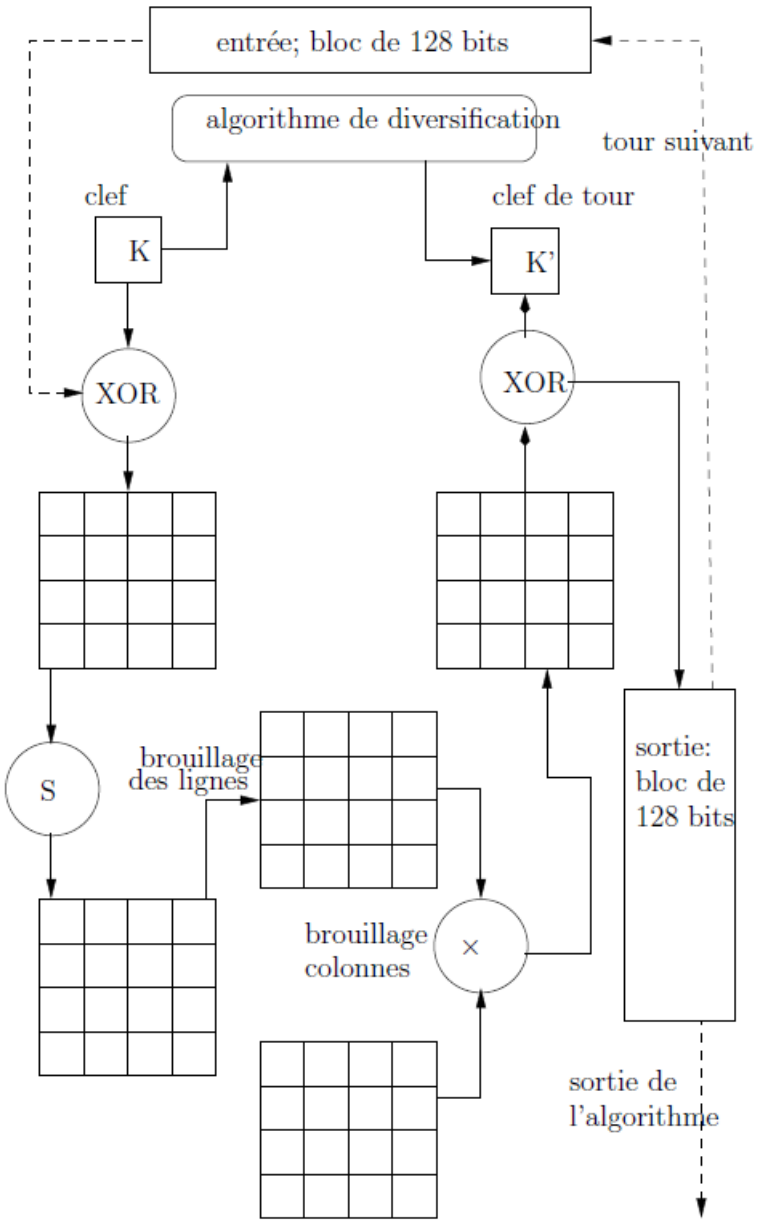

FIGURE II.3 – Architecture d'un tour de l'AES

opérations **SubBytes**, **ShiftRows** et **AddRound-Key**. A partir de la clef secrète, K, on génére des sous clefs d'étage par l'algorithme de diversification de la clef, **ExpandKey**$[i]$. On obtient les clefs d'étage : $K^1, ..., K^{N_e}$.

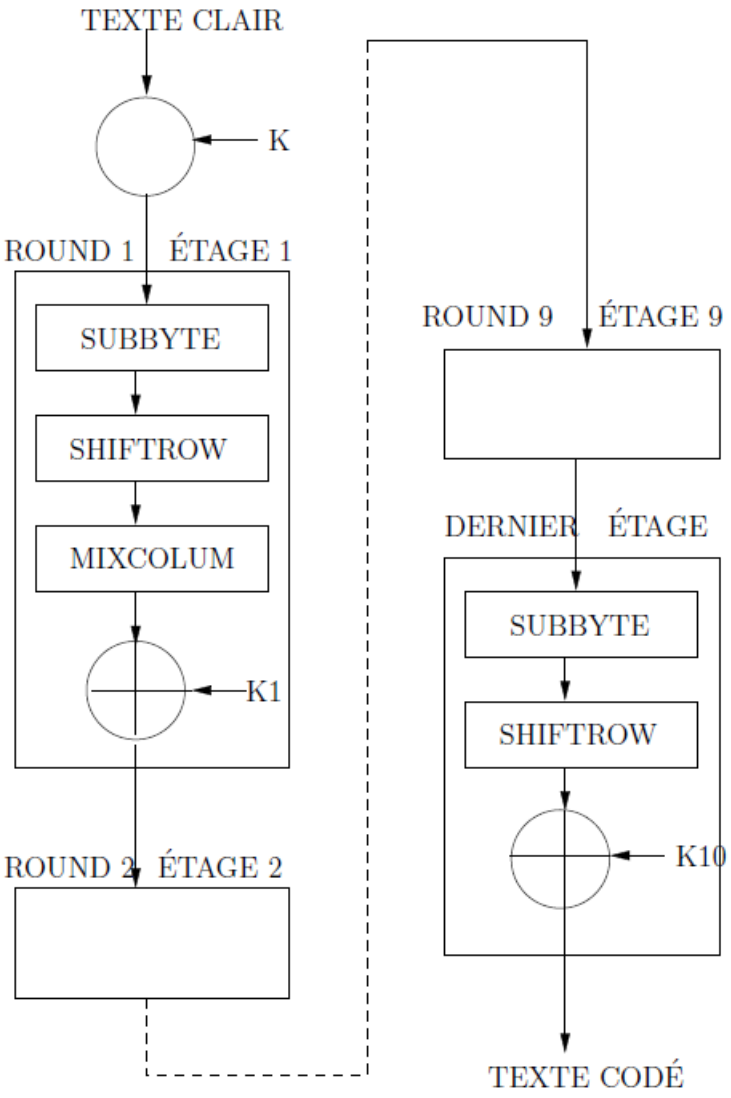

FIGURE II.4 – Diagramme du codage en AES.

On note x un bloc de 128 bits du texte initial (texte en clair), w^i le texte utilisé en entrée à l'étage i (c'est donc un bloc de 128 bits) et y sera le bloc correspondant crypté final.

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

$$K \oplus L \tag{II.3}$$

l'opération XOR entre le suite K et la suite L. Avec ces notations la description schématique d'AES est la suivante. On notera State l'écriture du flux d'entrée-sortie sur lequel opère chaque algorithme. Donc on prend le bloc de 128 bits sur lequel opère AES que l'on considère comme une suite de 16 octets que l'on range en une matrice 4×4. Un octet étant confondu

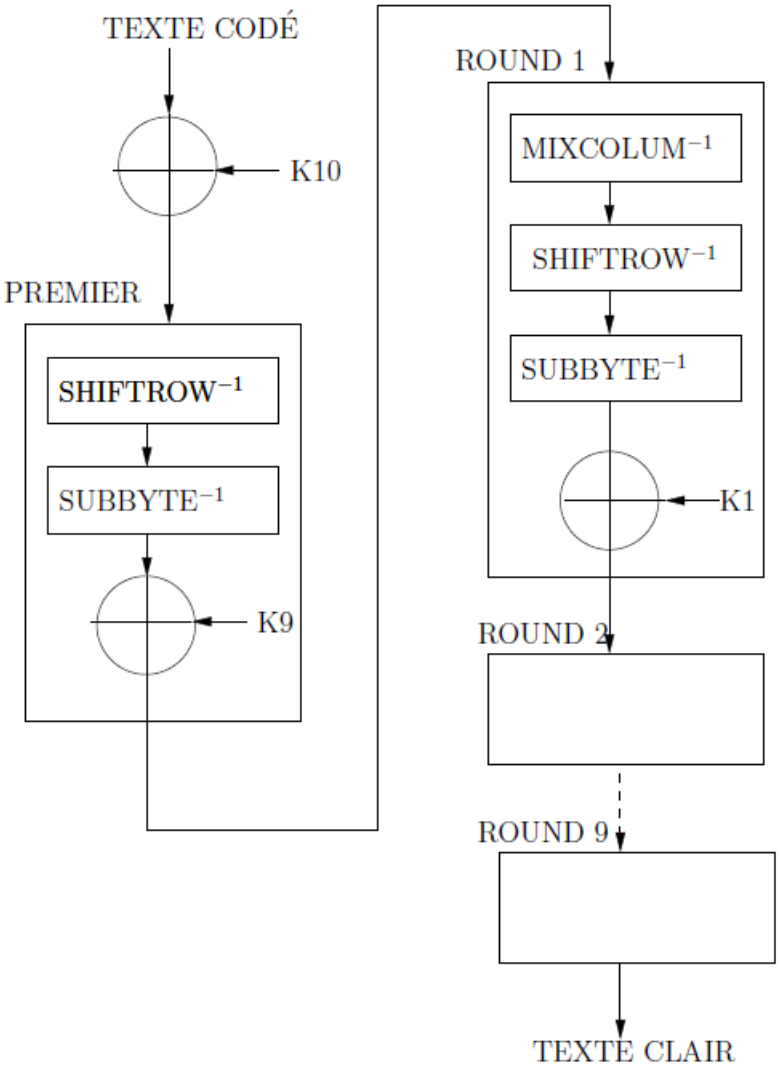

FIGURE II.5 – Diagramme du décodage en AES.

avec un élément de $\mathbb{F}_{256}$ de la manière suivante : $\mathbb{F}_{256}$ est identifié aux polynômes de $\mathbb{F}_2[x]$ de degré ≤ 7 par l'isomorphisme

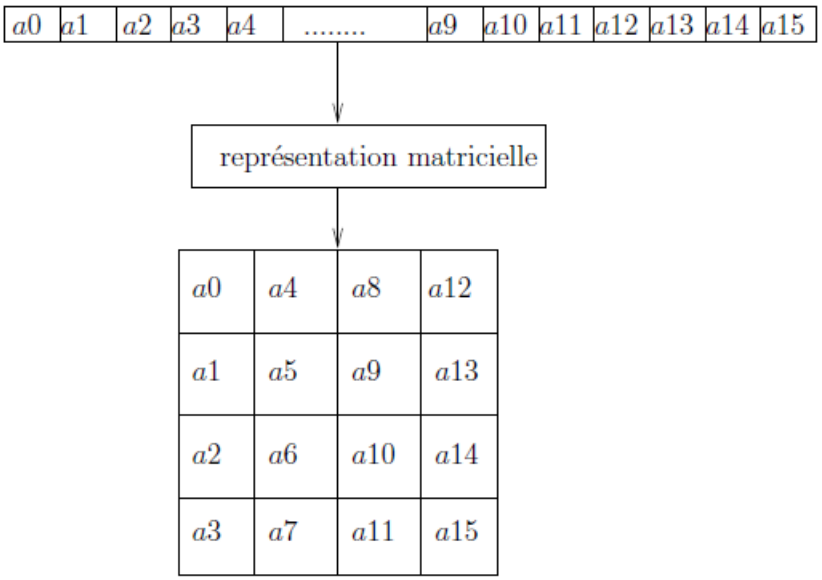

FIGURE II.6 – La matrice **State**.

$$\mathbb{F}_{256} \simeq \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1) \tag{II.4}$$

Donc **State** est une matrice de $M_{4\times4}(\mathbb{F}_{256})$ construite suivant le **schéma de la figure II.6**.

L'algorithme se déroule de la manière suivante :

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

$$W^0 \oplus K \tag{II.5}$$

2. Pour chacun des $1 < i \leq N_e - 1$ étages suivants on effectue sur w^i les opérations suivantes
   (a) l'opération de substitution **SubBytes**
   (b) sur le résultat de **SubBytes** une permutation notée **ShiftRows** (permutation circulaire sur les éléments des lignes de la matrice des octets $s_{i,j}$)
   (c) sur le résultat de **ShiftRows** une opération notée **MixColumns** (application linéaire sur l'espace des matrices sur F_{2^8})
   (d) Sur le résultat de **MixColumns** l'opération **AddRoundKey**$[i]$ avec la sous-clé de l'étage i
3. Pour le dernier étage N_e on supprime l'opération **MixColumns**

On va décrire maintenant chacune des opérations utilisées.

L'opération ExpandKey

L'opération **ExpandKey** de la figure II.7, dépend de la taille de la clé choisie, $N_k \times 32$, qui peut être égale à 128, 160, 192, 224 ou 256 bits. On supposera dans la suite que la longueur de la clef est de 192 bits, donc $N_k = 6$. Extension de la clef secrète K. On écrit la

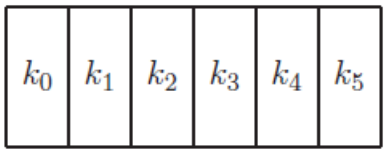

FIGURE II.7 – Opération ExpandKey.

clef K sous forme d'une matrice de N_k colonnes de 4 bytes chacune (donc 4 lignes, l'élément d'une ligne étant un mot de 8 bits), dénotées $k_0, ..., k_5$

Ensuite, cette matrice est étendue en une matrice de taille $4(N_e + 1)$ où N_e est le nombre d'étages par l'algorithme suivant

- Si i n'est pas un multiple de N_k (la longueur de la clef) alors la colonne d'indice i, k_i, est la XORisation de la colonne d'indice iN_k, k_{i-N_k}, et de la colonne d'indice $i-1$, $ki-1$. C'est donc l'addition bit à bit sans retenue de la colonne k_{i-N_k} et de la colonne $k_{i-1}, k_i = k_{i-N_k} \oplus k_{i-1}$.
- Si i est un multiple de N_k on applique à chacun des bytes de la colonne k_{i-1} la permutation π_S décrite au paragraphe suivant suivie d'une rotation dans les bytes de la nouvelle colonne k_{i-1}, notée **Rotword**, et de l'addition d'une constante d'étage définie en héxadécimal par [**FieldtoBinary**(x^{j-1})000000] pour l'étage j (autrement dit on ajoute **FieldtoBinary**(x^{j-1}) au premier byte de la clef d'étage après application de **Rotword**, c'est à dire à a_1 avec les notations ci-dessous). On XOR le résultat obtenu avec la colonne k_{iN_k} .

On obtient ainsi

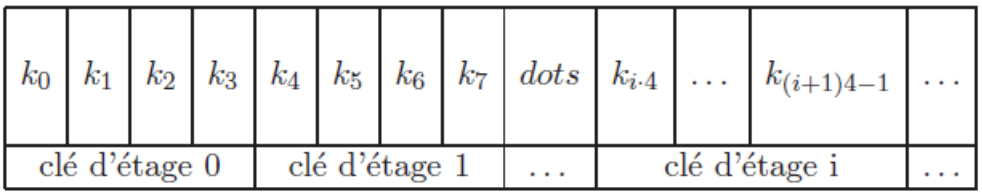

FIGURE II.8 – Opération RotWord.

La rotation Rotword est donnée par la figure II.1.6 et elle est définie comme suit

$$\textbf{Rotword} : \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \mapsto \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_0 \end{pmatrix}$$

L'opération SubBytes

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

$$S_i = \begin{matrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{matrix} \tag{II.6}$$

L'opération **SubBytes** consiste en une permutations π_S sur $\{0,1\}^8$, agissant indépendamment sur chacun des octets $s_{i,j}$ de l'étage i. Pour décrire π_S on travaille dans un surcorps du corps à 2 éléments, $\mathbb{F}_2$, le corps fini à 256 éléments, $\mathbb{F}_{2^8}$, que l'on identifie avec les polynômes de degré ≤ 8, munis de l'addition et de la multiplication modulo le polynôme irréductible de $\mathbb{F}[x], x^8 + x^4 + x^3 + x + 1$, **(A.1.2)**, de rappel sur les anneaux de polynômes,

$$\mathbb{F}_{256} \simeq \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1) \tag{II.7}$$

L'application **BinarytoField** associe à l'octet $a_7a_6...a_1a_0$ l'élément

$$\textbf{BinarytoField}(a_7a_6...a_1a_0) = \sum_{i=0}^{7} a_i x^i \tag{II.8}$$

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

Opération ShiftRows.

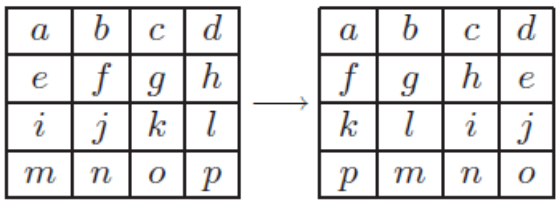

FIGURE II.9 – Opération ShiftRows

C'est la permutation cyclique dans les lignes de la matrice S_i décrite par le diagramme de la figure II.1.6 ci-dessous

L'opération MixColumns

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

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{II.9}$$

Polynomialement cette opération s'interprète de la manière suivante. Chacune des colonnes de la matrice State est considérée comme un polynôme de degré 3 à coefficients dans $\mathbb{F}_{256}$. L'opération **MixColumns** consiste alors à effectuer pour chaque colonne une multiplication du polynôme correspondant à la colonne par un polynôme fixe $c(x) = 03x^3 + x^2 + x + 02$ suivie d'une réduction modulo le polynôme $x^4 + 1$ de façon à obtenir un polyôme de degré inférieur ou égal à 3 de $\mathbb{F}_{256}[X]$ qui est interprété comme une colonne d'une matrice de $M_4(\mathbb{F}_{256})$.

L'opération AddRoundKey

AddRoundKey[i] donné par la figure II.10 est l'addition de la clé obtenue à l'étage i par l'algorithme de diversification des clés, **ExpandKey**[i], au texte obtenu à l'issue de l'étape MixColumns.

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

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

FIGURE II.10 – Opération AddRoundKey.

Partie 2 : Systèmes cryptographiques à clé publique

En 1976, Diffie et Hellman, [23], ont révolutionné la cryptographie en inventant un système asymétrique dans lequel l'émetteur et le récepteur ne partagent plus de clé commune : une clé, rendue publique, permet de chiffrer un message, la seconde est gardée secrète et elle permet au destinataire de déchiffrer le message.

Cette découverte a modifié la cryptographie car jusqu'alors, pour envoyer un message chiffré, les correspondants devaient s'échanger au préalable une information secrète, qui est : la clef.

En inventant la cryptographie asymétrique, Diffie et Hellmanont ont ainsi conçu un système fondé sur la notion de fonction à sens unique où les correspondants peuvent s'échanger une information secrète sans se rencontrer.

Les systèmes de chiffrement asymétriques les plus utilisés sont :

- **RSA** basé sur la difficulté calculatoire de la factorisation des grands nombres entiers.
- **El Gamal** basé sur la difficulté calculatoire de calculer le logarithme discret dans un corps fini.
- **Menezes-Vanstonne** basés sur basés sur le logarithme associé au groupe des points d'une courbe elliptique sur un corps fini. C'est une modification d'autres cryptosystèmes, comme El Gamal. Sa sureté peut être mieux analysée. La longueur de sa clé est bien inférieure à celle des systèmes RSA *(Rivest Shamir Adleman)* et El Gamal pour une sureté èquivalente. Ils offrent des fonctionnalités supplémentaires comme la possibilité de monter un cryptosysème basée sur l'identité.

Ils existent d'autres codes asymétriques peu utilisés en pratique comme

- **McEliece** basé sur la théorie algébrique des codes correcteurs d'erreurs. Il repose sur la difficulté calculatoire du décodage d'un code linéaire (problème NP-complet). Considéré comme sûr il nécessite des clefs extrèmement longues 10^{19} bits.
- **Chor-Rivest** basé sur une instance du problème du **sac-à-dos**.

II.2 Concept cryptographique

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

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

Autrement dit, on cherche des fonction $f : \mathcal{P} \to \mathcal{E}$ appartenant à la classe des problèmes P (calculables en temps polynomial en fonction de la taille des données) et telle que la fonction réciproque (ou l'image réciproque si on ne suppose pas f bijective) $f^{(-1)}$ appartienne à la classe des problèmes NP, *(Non calculables en temps Polynomial)* en fonction de la taille des données, mais vérifiables en temps polynomial.
La mise au point de telles fonctions a révolutionné la cryptographie et élargi son champ d'application.

II.2.1 Cryptosystème Merkle-Hellman.

Ce cryptosystème qui fut historiquement un des premiers proposés, (il a été publié en 1978), repose sur le problème du sac-à-dos. Génériquement, il a une sécurité prouvée, car le problème du sac-à-dos est génériquement dans la classe NP. Malheureusement, comme l'a montré Shamir sa sécurité ne repose pas sur un problème de sac-à-dos générique mais sur une instance particulière de ce problème qui elle peut ne pas être dans la classe NP.

Le problème du sac-à-dos

Le problème du sac-à-dos s'énonce de la manière suivante,

- Soit a un ensemble d'entiers $\{a_1, a_2, ..., a_n; s\}$.
- Existe-t-il un sous-ensemble de $\{a_1, a_2, ..., a_n\}$ dont la somme soit, s,

Intuitivement on peut considérer que les a_i représentent la taille de paquets que l'on veut mettre dans un sac-à-dos de taille s, ou encore qu'ils représentent des billets et des pièces monnaies destinées à payer un objet dont le prix est, s. L'interét de ce problème du sac-à-dos pour la cryptographie tient au théorème suivant :

Théorème II.2.1.1 (Karp 1972) *Le problème du calcul de la solution, supposée exister, du sac-à-dos est un problème génériquement NP-complet.*

De manière un peu approximative, ce théorème signifie que génériquement trouver une solution que l'on sait exister au problème du sac-à-dos, est difficile c'est à dire que le calcul de la solution nécessite un temps exponentiel en fonction de la taille des données. Le mot générique signifie qu'il existe au moins un ensemble a ayant une solution, tel que, le calcul de la solution ne se fasse pas en un temps polynomial en fonction de la taille des données. Le problème est de cacher une clef secrète, une porte dérobée, ou (backdoor/trapdoor) qui permette au destinataire du message de le déchiffrer.

On choisit un entier, M, on dira que $a = \{a_1, a_2, ..., a_n;\ s\}$ est un problème de sac-à-dos modulo M s'il existe,

$$\{a_{k_1}, ..., a_{k_m}\} \subset \{a_1, a_2, ..., a_n\} \tag{II.10}$$

tel que,

$$a_{k_1} +, ...,\ a_{k_m} \equiv s\ mod\ M \tag{II.11}$$

L'idée principale de Merkle et Hellman, consiste à choisir un sac-à-dos particulier dont la solution est triviale et à cacher cette forme particulière du sac-à-dos par une transformation secrète connue du seul destinataire du message. La forme retenue par Merkle et Hellman est le sac-à-dos super-croissant (super-increasing knapsack).

Description II.2.1 (Merkle-Hellman) *Une suite d'entier $\{b_1, b_2, ..., b_n, b_{n+1}\}$ est dite super-croissante si l'on a*

$$b_i > \sum_{j=1}^{i-1} b_j \; ; pour \; 1 \leq i \leq n+1. \tag{II.12}$$

On remarque que si la suite $\{b_1, b_2, ..., b_n, b_{n+1}\}$ *est super-croissante et si* $M = b_{n+1}$ *alors tout problème de sac-à-dos* $\{b_1, b_2, ..., b_n;\ s\}$ *modulo* M *est équivalent au problème de sac-à-dos* $\{b_1, b_2, ..., b_n;\ s\}$ *sur les entiers naturels. Ce dernier est un problème facile. On remarque que* b_n *est dans le sousensemble de la solution si et seulement si* $b_n \leq s$ *et on peut alors ramener le problème initial au problème de sac-à-dos* $\{b_1, b_2, ..., b_{n-1};\ s^{'}\}$ *avec*

$$s^{'} = \begin{cases} s - b_n \\ \quad ou \\ \quad s \end{cases} \tag{II.13}$$

La transformation secrète destinée à cacher le fait que l'on a un sac-à-dos super-croissant est simplement la multiplication par un nombre secret M*. Le crypto-système de Merkle et Hellman est donc le suivant*

1. Paramètre public n.
2. fabrication de la clef : choisir une suite super croissante

   $$\{b_1, b_2, ..., b_n, b_{n+1} \; = \; M\}.$$

   choisir un nombre $W \in \mathbb{Z}/M\mathbb{Z}$ et une permutation π de $\{1, ..., n\}$.
   Calculer $a_i = W b_{\pi(i)} \, (mod \, M)$ pour $1 \leq i \leq n$.
3. Clef Publique : $K_p = (a_1, a_2, ..., a_n)$
4. Clef secrète $K_s = (b_1, b_2, ..., b_n, M, W, \pi)$
5. Message : une suite de zéros et de uns de longueur n, $m = m_1 m_2 ... m_n$
6. Chiffrement $c = \sum_{i=1}^{n} m_i a_i$
7. Déchiffrer : calculer CW^{-1} $(mod \; M)$, résoudre le problème de sac-à-dos super croissant $x_1 b + ... + xb \; = \; cW^{-1} (mod \, M)$ et alors $m_i = x_{\pi(i)}$.

Le crypto-système Merkle-Hellman revient à trouver un vecteur dans un réseau. Un **réseau (lattice)** est l'ensembles des combinaisons à coefficients entiers relatifs de vecteurs de $\mathbb{R}^n$. Trouver un petit vecteur au sens de la norme euclidienne est aussi un problème difficile. mais il y a des cas où ce problème devient facile. Plus précisément l'**algorithme LLL**, de *(A. Lenstra, H. Lenstra et L. Lovász)* peut être utilisé. En particulier il peut être utilisé pour casser des crypto-systèmes comme celui de Merkle-Hellman.

II.2.2 Le cryptosystème RSA

Le cryptosystéme RSA est nommé d'après le nom de ses inventeurs : *(Rivest, Shamir, Adleman)*. Il est basé sur la fonction à sens unique **produit de deux entiers**

$$\begin{aligned} f : & \mathbb{N} \times \mathbb{N}\mathbb{N} \\ & (n, m) \mapsto f(n, m) = n \times m \end{aligned} \tag{II.14}$$

sa fonction réciproque étant la décomposition d'un nombre entier en un produit de deux facteurs non-triviaux (c'est à dire différents de 1 et du nombre lui même) et donc finalement sur la décomposition en facteurs premiers d'un entier. Le calcul du produit de deux nombres entiers est une opération **facile**, **rapide**, *calculatoirement facile* (c'est à dire faisable en temps polynomial en fonction de la taille des données) alors que la décomposition en facteurs premiers n'est pas **facile**, est **lente**, *calculatoirement difficile* (c'est à dire qu'elle n'est pas faisable en temps polynomial en fonction de la taille des données).

Description II.2.2 (Cryptosystème RSA) *On a un ensemble $(A_i)_{i\in I}$ de correspondants (personnes physiques ou ordinateurs). Le principe du cryptosystème RSA est le suivant :*

- *Chacun des correspondants A_i choisit deux nombres premiers p_i et q_i distincts. On note $n_i = p_i \times q_i$, le* ***module du cryptosystème*** *de A_i et $\varphi(n_i) = (p_i - 1)(q_i - 1)$, l'indicatrice d'Euler de n_i.*
- *A_i choisit ensuite l'exposant e_i de la* ***clé publique*** *premier avec $\varphi(n_i)$ (ce que l'on note $e_i \wedge \varphi(n_i) = 1$). D'après le petit théorème de Fermat, **(A.2.4.2)**, on peut imposer $1 \leq e_i \leq \varphi(n_i) - 1$, ce que l'on fait toujours en pratique.*
- *Puis A_i calcule l'inverse d_i de e_i modulo $\varphi(n_i)$, d_i est appelé l'**exposant de la clé secrète**. C'est à dire, d'après la définition de l'inverse modulo $n_i \in \mathbb{N}$ tel qu'il existe $k_i \in \mathbb{N}$ avec*

$$d_i \times e_i \;=\; 1 + \; k_i\, \varphi(n_i) \quad et \quad 1 \leq d_i \leq \varphi(n_i) - 1 \tag{II.15}$$

  Le calcul se fait par l'algorithme d'Euclide étendu.

*Chacun des correspondants A_j publie dans un annuaire public les nombres n_j et e_j qui forment sa **clé publique de chiffrement** et garde secret p_j , q_j et d_j qui forment sa **clé secrète de déchiffrement**. En pratique on choisit les nombres premiers p_i et q_i de taille comparable de telle sorte que leur produit $n_i = p_i \times q_i$ soit un nombre d'au moins 300 chiffres en base 10 et plutôt 500 pour une protection longue.*

Exemple : *Soient $p = 5$, $q = 17$, c'est-à-dire, $n = p \cdot q = 85$.*

Alors, $\varphi(n) = (5 - 1) \times (17 - 1) = 64$.

*Soit l'exposant choisi, $e = 5$, tel que, le $pgcd(e, \varphi(n)) = 1$. On calcule le couple de Bézout $(d, (\varphi(n))^{-1})$, qui sont respectivement les inverses de, e et $\varphi(n)$ (voir **(A.2.3.1)**), grâce à l'algorithe d'Euclide étendu (A.2.2). Ainsi, $d = 13$.*

Donc, la clé publique est $(5, 85)$, et la clé privée à garder secrète est $(12, 85)$. Tandis que, les valeurs de, p, q et $\varphi(n)$ sont à détruire.

Le chiffrement du message clair `M=10`, sera comme suit,

- *Le chiffrement : c'est de calculer : $M^e \equiv C (mod\, n)$, donc $10^5 \equiv 40\, (mod\, 85)$. Ainsi la lettre chiffrée `M=10`, est $C = 40$.*
- *Le déchiffrement : c'est de calculer : $c^d \equiv M (mod\, n)$, donc $40^{13} \equiv 10\, (mod\, 85)$. Ainsi la lettre déchiffrée `C=40`, est $M = 10$.*

II.2.3 Le cryptosystème El Gamal

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

$$G = \{g^i \,|\, 0 \leq i \leq n - 1\} \tag{II.16}$$

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

Autrement dit G est isomorphe au sous groupe additif de $\mathbb{Z}/n\mathbb{Z}$ et la sécurité du cryptosystème El-Gamal repose sur la difficulté à calculer explicitement cet isomorphisme en un temps raisonnable. On utilise les deux réalisations suivantes de G. On considère un nombre premier p et on choisit

$$G = (\mathbb{Z}/p\mathbb{Z})^* \simeq \mathbb{F}_p^* \tag{II.17}$$

c'est à dire que le G est le groupe des éléments inversible pour la multiplication de $\mathbb{Z}/p\mathbb{Z}$ les entiers modulo p, pour la définition voir dans les rappels mathématiques donnés dans **(A.1.3)**.

D'aprés un théorème de Gauss,le groupe multiplicatif de $\mathbb{Z}/p\mathbb{Z}$ est cyclique et le calcul de $g^i \, mod \, p$ est rapide alors qu'on ne connait pas d'algorithme en temps polynomial pour calculer i connaissant $a = g^i$ pour de bons choix de p. Plus généralement on peut considérer le groupe multiplicatif des éléments inversibles d'un corps fini à $q = p^s$ éléments, $\mathbb{F}_q$.
Une autre réalisation est de prendre comme groupe G le groupe des points d'une courbe elliptique sur un corps fini, $E(\mathbb{F}_q)$ qui donne lieu aux cryptosystèmes elliptiques. Ils se développent actuellement car ils possèdent des avantages en terme de taille de clé et de sécurité prouvée.

Description II.2.3 *On considère le cryptosystème suivant : Soit p un nombre premier tel que le problème du logarithme discret dans $\mathbb{Z}/p\mathbb{Z}$ soit difficile. Soit g une racine primitive modulo p, c'est à dire g est un générateur du groupe cyclique $(\mathbb{Z}/p\mathbb{Z})^*$.*
Soit $\mathcal{P} = (\mathbb{Z}/p\mathbb{Z})^*$, les messages en clair, et soit $\mathcal{C} = (\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/p\mathbb{Z})^*$ les messages chiffrés et enfin soit l'espace des clés,

$$\mathcal{K}_b = \{(p, g, \alpha, \beta);\ \beta \equiv g^\alpha \ mod \ p\} \ ou \ \begin{cases} (p, g, \beta) & est \, une \, clef \, publique \\ \alpha & est \, une \, clef \, privee \end{cases} \tag{II.18}$$

Alice veut transmettre un message, M, à Bob, donc,

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

$$y_1 \equiv g_b^{k_a} \ (mod \, p_b) \ et \ y_2 = \beta_b^{k_a} M \tag{II.19}$$

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

- *Bob calcule,*

$$M \equiv y_2 (y_1^{\alpha_b})^{-1} \ mod \ p_b \tag{II.20}$$

Comme $y_1^{\alpha_b} = g_b^{k_a \alpha_b} \, (mod \, p)$ on a :
$d_K(y_1, y_2) = y_2 (y_1^{\alpha_b})^{-1} \equiv \beta_b^{k_a} M (g_b^{k_a \alpha_b})^{-1} \equiv g_b^{\alpha_b k_a} M g_b^{-\alpha_b k_a} \equiv M \, mod \, p$
et comme M est inférieur à p_b il n'y a pas d'ambiguité dans le déchiffrement. Pour une bonne sécurité, Alice doit changer souvent k_a.

Exemple : *Soient* $p_b = 2579$, $g_b = 2$, $\alpha_b = 765$. *Il vient,*

- *Clé privée* $\alpha_b = 765$.
- *Clé publique* $K_b = (p_b, g_b, \beta_b) = (2579, 2, 949)$ *car* $\beta_b = 2^{765} \, mod \, 2579 = 949$.

Pour chiffrer `M=1299`, *on choisit une clé* $k_a = 853$. *Il vient,*

$$y_1 = 2^{853} \, mod \, 2579 = 435,$$

$$y_2 = 1299 * 949^{853} \, mod \, 2579 \; = 2396,$$

Donc, la paire $(y_1, y_2) = (435, 2396)$ *est le message chiffré qu'Alice envoie à Bob.*
On peut effectivement vérifier que,

$$M = y_2 \cdot (y_1)^{-1} \, mod \, 2579 = 2396 \cdot (435^{765})^{-1} \, mod \, 2579 = 1299.$$

Partie 3 : Courbes élliptiques

II.3 Concept cryptographique

II.3.1 Introduction

Il s'agit d'un concept proposé en 1985 par deux chercheurs Miller et Klobitz [14], de façon totalement indépendante. Ce type de cryptographie, toujours basé sur le modèle asymétrique permet aussi bien de chiffrer que de signer. On utilise souvent l'abréviation ECC, *(Elliptic Curve Cryptography)*. Les clés utilisées sont plus courtes pour une sécurité égale ou supérieure. La théorie sous-jacente, ainsi que l'implémentation sont plus complexes, ce qui explique le fait que cette technologie soit moins répandue. Toutefois, de par la nécessité de traiter plus rapidement l'information, de gérer des quantités de données importantes et de miniaturiser au maximum, les avantages de cette technique poussent la recherche. D'une manière générale, sur $\mathbb{R}$, les courbes elliptiques seront considérées comme l'ensemble des couples (x, y) tels que,

$$y^2 = x^3 + ax + b \tag{II.21}$$

dont le discriminant

$$\Delta = -(4a^3 + 27b^2) \tag{II.22}$$

soit différent de zéro (non nul).
Pour la dessiner, pour a et b fixés, on calcule y tel que

$$y = \sqrt{x^3 + ax + b} \tag{II.23}$$

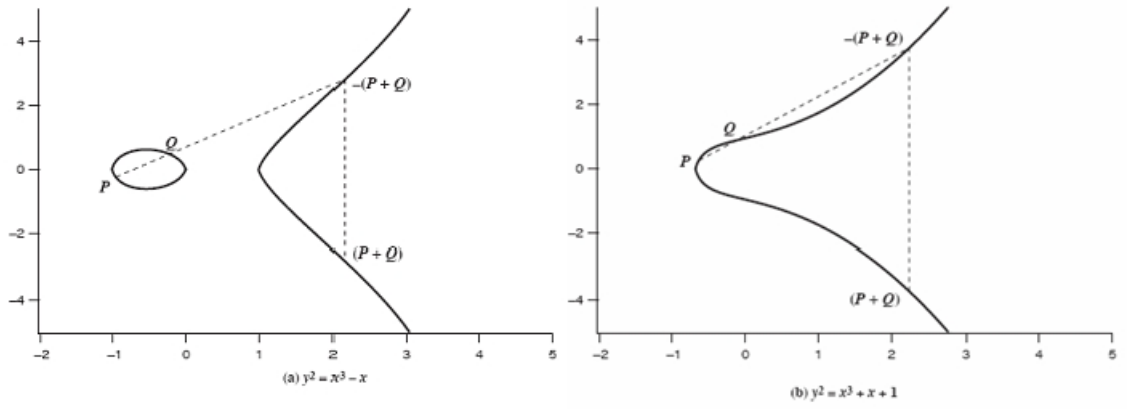

FIGURE II.11 – Deux exemples de courbes élliptiques

Remarque. *Contrairement à ce que l'on peut croire à première vue, il ne s'agit pas de travailler à partir d'ellipses. La raison en est que les équations utilisées (équations cubiques) sont similaires à celles permettant de déterminer la circonférence d'une ellipse.*

Description II.3.1 (Définition géométrique) *Soit une opération (l'addition, +) pour l'ensemble $E(a, b)$ tel que a et b répondent à la condition du discriminant. Si 3 points sur une EC sont alignés, leur somme vaut $\mathcal{O}$ (point à l'infini).*

1. *$\mathcal{O}$ est l'identité pour l'addition : $\mathcal{O} = -\mathcal{O}$.*

2. *Pour n'importe quel point* $P + \mathcal{O} = P$.
3. *L'opposé d'un point* $P(x, y)$ *est* $P(x, -y)$
4. *Pour additionner 2 points* P *et* Q*, on trace la droite les reliant. Cela nous donne un point d'intersection* R*. On définit l'addition telle que* $P + Q = -R$*. En conséquence, on définit* $P + Q$ *comme étant l'opposé de ce point* R.

II.3.2 Les courbes élliptiques sur $\mathbb{Z}_p$

Les variables et coefficients prennent des valeurs dans l'ensemble $[0, p-1]$ pour un certain nombre premier p, et où toutes les opérations sont calculées modulo p. L'équation devient,

$$y^2 \, mod \, p = (x^3 + ax + b) \, mod \, p \tag{II.24}$$

Cette équation est par exemple satisfaite pour $a = 1$, $b = 1$, $x = 9$, $y = 7$ et $p = 23$.

$$7^2 \, mod \, 23 = (9^3 + 9 + 1) \, mod \, 23 \tag{II.25}$$

$$49 \, mod \, 23 = 739 \, mod \, 23$$

$$3 = 3$$

On note $E_p(a, b)$ l'ensemble des couples d'entiers (x, y) qui satisfont cette équation. On parle de groupe elliptique.

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

FIGURE II.12 – Couples (x, y) répondant à l'équation $y^2 = x^3 + x + 1$.

Exemple : *Soient* $p = 23$ *et la courbe elliptique* $y^2 = x^3 + x + 1$*. On est donc dans* $E_{23}(1, 1)$*. Comme nous travaillons dans* $\mathbb{Z}_p$*, les couples* (x, y) *répondant à l'équation sont donnés à la figure II.12.*

Pour tous points $P, Q \in E_p(a, b)$:

1. $P + \mathcal{O} = P$
2. Si $P = (x_P, y_P)$, alors $P + (x_P, -y_P) = 0$ et $(x_P, -y_P) = -P$.
   Retour à l'exemple : Dans $E_{23}(1, 1)$, pour P = (13, 7), $-P = (13, 7) = (13, 16)$
3. Si $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ avec $P \neq -Q$, alors on détermine $R = P + Q = (x_R, y_R)$ comme suit,

$$x_R = (\lambda^2 - x_P - x_Q) \, mod \, p \tag{II.26}$$

$$y_R = (\lambda(x_P) - x_R) - y_P) \, mod \, p \tag{II.27}$$

où

$$\lambda = \begin{cases} \left(\frac{y_Q - y_P}{x_Q - x_P}\right) \; mod \; p \, si \, P \neq Q \\ \left(\frac{3x_P^2 + a}{2y_P}\right) \; mod \; p \, si \, P = Q \end{cases} \tag{II.28}$$

4. La multiplication est définie comme une répétition d'additions (ex : $3P = P + P + P$).

Retour à l'**exemple**.
Soient $P = (3, 10)$ et $Q = (7, 9)$ dans $E_{23}(1, 1)$. Il vient,

$$\lambda = \left(\frac{7 - 10}{3 - 9}\right) \; mod \; 23 = \left(\frac{-3}{6}\right) \; mod \; 23 = \left(\frac{-1}{2}\right) \; mod \; 23 = 11 \tag{II.29}$$

$$x_R = (11^2 - 3 - 9) \, mod \, 23 = 109 \, mod \, 23 = 17 \tag{II.30}$$

$$y_R = (11(3 - 17) - 10) \, mod \, 23 = -164 \, mod \, 23 = 20 \tag{II.31}$$

Et ainsi, $P + Q = (17, 20)$.

Pour trouver $2P$, on a,

$$\lambda = \left(\frac{3(3^2) + 1}{2 * 10}\right) \; mod \; 23 = \left(\frac{5}{20}\right) \; mod \; 23 = \left(\frac{1}{4}\right) \; mod \; 23$$

$$x_R = (6^2 - 3 - 3) \, mod \, 23 = 30 \, mod \, 23 = 17$$

$$y_R = (6(3 - 7) - 10) \, mod \, 23 = -34 \, mod \, 23 = 12$$

et donc $2P = (7, 12)$

Remarque. *La section précédente traite du problème dans $\mathbb{Z}_P$. Les équations sont différentes si nous travaillons dans le corps fini $GF(2^m)$, pour (Galois Field), ce qui est souvent connu sous le nom : corps de Galois.*

Chapitre III

Cryptanalyse

Contents

III.1 Introduction

Il s'agit de l'étude des mécanismes théoriques ou techniques visant à briser (casser) un algorithme de chiffrement, c'est-à-dire le fait de retrouver le message M à partir de C, sans connaître la clé K a priori. Dans certains cas, il s'agira également de retrouver cette clé K. On parlera d'*attaque* cryptanalytique.

Un système cryptographique ne se conçoit pas indépendamment des attaques dont il peut être l'objet. On indiquera donc pour chaque système cryptographique quelques attaques et sa résistance à ces attaques. L'accent sera mis sur les principes et les outils mathématiques utilisés (arithmétique, algèbre, algorithmique, complexité, probabilité, théorie de l'information,..). On évoquera aussi quelques grands types de menaces et d'attaques sur les systèmes cryptographiques. Ce chapitre présentera ensuite quelques attaques souvent évoquées dans la littérature spécialisée dans le domaine de la cryptologie.

Les travaux de Shannon dans les années 1940, [22], sur les communications (théorie de l'information) sont précurseurs des futurs travaux théoriques en cryptographie. En particulier, l'article intitulé Communication Theory of Secrecy Systems a dégagé et étudié la notion d'entropie et a prouvé la sécurité parfaite du schéma de chiffrement de Vernam **(I.2.5)**.

Shannon a aussi présenté des notions importantes sur la sécurité d'un protocole cryptographique en distinguant sécurité inconditionnelle et sécurité calculatoire. Dans le premier cas, on suppose que l'adversaire a une ressource de temps infini, alors que dans le second cas, l'adversaire est borné dans le temps. Il ne peut effectuer qu'un nombre fini d'étapes de calcul, d'où le nom de sécurité calculatoire.

Il est en effet raisonnable de penser que si, pour casser un système, il faut utiliser un million de machines pendant un million d'années, alors le système est sûr.

En pratique, on suppose que pour casser un système cryptographique, l'adversaire a accès à plusieurs ordinateurs. On peut quantifier le nombre d'opérations qu'il peut faire ainsi que le nombre d'informations qu'il est capable de stocker. Par exemple, un ordinateur cadencé à 1 GHz effectue un milliard de cycles par seconde, soit environ 2^{30} opérations élémentaires en 1 s. Si un million d'ordinateurs à 1 GHz fonctionnent pendant cent mille ans, alors ils sont capables d'effectuer environ 2^{92} opérations. D'un point de vue pratique, on dira qu'un système est sûr si le nombre d'opérations minimales pour le casser nécessite plus de 2^{80} opérations. Si on veut se protéger de manière plus sûre, on augmentera le niveau de sécurité à 2^{128}.

III.1.1 Attaques sur un chiffrement

La cryptanalyse est l'ensembles des procédés d'attaque d'un cryptosystème. Elle est indispensable pour l'étude de la sécurité des procédés de chiffrement utilisés en cryptographie. Son but ultime est de trouver un algorithme de déchiffrement des messages. Le plus souvent on essaye de reconstituer la clef secrète de déchiffrement. On suppose, en vertu des principes de Kerckhoffs, pour toutes les évaluations de sécurité d'un cryptosystème que l'attaquant connait le système cryptographique utilisé, la seule partie secrète du cryptosystème est la clef. Par exemple dans un cryptosystème basé sur des registres à décalage on suppose que l'attaquant connait la forme des récurrences linéaires ainsi que la fonction de combinaison mais pas les conditions intiales des récurrences qui constituent la clef du code. On doit distinguer entre les types d'attaques d'un adversaires et les buts des attaques d'un adversaire. Il en existe quatre grands types, chacun pouvant utiliser différentes techniques.

1. **attaque à texte chiffré connu :** l'opposant ne connait que le message chiffré y.

2. **attaque à texte clair connu :** l'opposant dispose d'un texte clair x. et du message chiffré correspondant $\vec{y}$.
3. **attaque à texte clair choisi :** l'opposant a accès à une machine chiffrante. Il peut choisir un texte clair et obtenir le texte chiffré correspondant $\vec{y}$, mais il ne connait pas la clef de chiffrement.
4. **attaque à texte chiffré choisi :** l'opposant a accès à une machine déchiffrante. Il peut choisir un texte chiffré, $\vec{y}$ et obtenir le texte clair correspondant $\vec{x}$, mais il ne connait pas la clef de déchiffrement.

   En plus de ces attaques basées sur une étude de messages codés, il y a aussi des attaques physiques. Le principe de ces attaques est d'essayer de reconstituer la clef secrète par exemple en espionnant la transmission entre le clavier de l'ordinateur et l'unité centrale ou en mesurant la consommation électrique du microprocesseur qui effectue le décodage du message ou encore en mesurant son échauffement. Ensuite on essaye de remonter de ces données physiques aux clefs de codage et décodage. Une méthode pour résister à ce type d'attaque sont les protocoles de preuve sans transfert de connaissance (zero-knowledge proof).

   Le but de l'attaque d'un adversaire peut être soit de *découvrir la clef du chiffrement* et de pouvoir ainsi décrypter tous les messages de l'émetteur ou plus modestement de *décryter un message particulier* sans nécessairement disposer de la clef du code. Garantir la confidentialité des communications entre Alice et Bob suppose donc qu'Eve ne peut pas

5. trouver M à partir de $E(M)$ (le crypto-système doit être résistant aux attaques sur le message codé)
6. trouver la méthode de déchiffrement D à partir d'une famille de couples, $E(M_i)$, (message clair, message codé correspondant).
7. accéder à des données contenues dans le micro-processeur qui code et décode et plus généralement ne puisse pas espionner les ordinateurs d'Alice et de Bob.

III.2 Différentes notions de sécurité d'un cryptosystème.

8. **La sécurité inconditionnelle** qui ne préjuge pas de la puissance de calcul du cryptanalyste qui peut être illimitée.
9. **La sécurité calculatoire** qui repose sur l'impossibilité de faire en un temps raisonnable, compte tenu de la puissance de calcul disponible, les calculs nécessaires pour décrypter un message. Cette notion dépend de l'état de la technique à un instant donné.
10. **La sécurité prouvée** qui réduit la sécurité du cryptosystème à un problème bien connu réputé difficile, par exemple on pourrait prouver un théorème disant qu'un système cryptographique est sûr si un entier donné \ ne peut pas être factorisé.
11. **La confidentialité parfaite** qualité des codes pour lesquels un couple (message clair, message chiffré) ne donne aucune information sur la clef.
    Toutes ces notions de sûreté reposent sur la théorie de l'information de Claude Shannon, cf. Théorie de l'information. Il a pu donner un sens précis basé sur les probabilités à la notion de sécurité inconditionnelle si l'on précise le type d'attaques permises. Il a

aussi donné un sens précis à la notion de confidentialité parfaite. La notion de sécurité calculatoire repose sur la théorie de la complexité.
Dans la pratique il faut préciser le type d'attaque. C'est la sécurité calculatoire que l'on utilise dans la plupart des évaluations de sécurité des systèmes cryptographiques. Elle repose sur la remarque suivante :
même avec des ordinateurs faisant 10^9 opérations élémentaires par seconde un calcul qui nécessite 2^{100} opérations élémentaires est hors de portée actuellement car pour l'effectuer il faut environ 4.10^{13} années !
La sécurité prouvée consiste à ramener la sécurité d'un cryptosystème à un problème que l'on sait ou que l'on espère être calculatoirement difficile comme par exemple la factorisation des entiers en facteurs premiers. Ceci permet de classifier les différents cryptosystèmes suivant leur sécurité et de procéder à une veille technologique rationnelle.

III.3 Autres techniques d'attaques cryptanalytiques

III.3.1 La force brute

Le principe est ici de tester toutes les clés possibles de manière exhaustive. La limite maximale est donnée par

$$T^N$$

avec T représentant la taille de l'alphabet, N est la taille de la clé. Par exemple, pour une clé de 128 bits, il y a 2^{128} clés possibles. Cette technique n'est "efficace" que pour des textes chiffrés avec une clé relativement courte.

Sur la figure III.1, la ligne rouge indique la limite actuelle conseillée. Un clé de 56 bits n'est plus à utiliser aujourd'hui si l'objectif premier est la confidentialité. Les algorithmes symétriques actuels utilisent en standard des clés variant entre 112 bits dans le 3DES et 256 bits pour le AES.

Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption/μs	Time required at 10^6 encryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\ \mu s = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\ \mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\ \mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\ \mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\ \mu s = 6.4 \times 10^{12}$ years	6.4×10^{6} years

FIGURE III.1 – Temps de calcul pour une taille de clé donnée.

Remarque. *Un algorithme est dit cassé quand il est possible de retrouver la clé en effectuant moins d'opérations qu'en utilisant la force brute.*
Un algorithme cassé est "moins sûr", mais pas inutile pour autant. Il faudra veiller à son utilisation si on souhaite assurer la confidentialité des données, mais rien n'empêche de l'utiliser à d'autres fins.

Level	Protection	Symmetric	Asymmetric	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash
1	Attacks in "real-time" by individuals *Only acceptable for authentication tag size*	32	-	-	-	-	-
2	Very short-term protection against small organizations *Should not be used for confidentiality in new systems*	64	816	128	816	128	128
3	Short-term protection against medium organizations, medium-term protection against small organizations	72	1008	144	1008	144	144
4	Very short-term protection against agencies, long-term protection against small organizations *Smallest general-purpose level, protection from 2008 to 2010*	80	1248	160	1248	160	160
5	Legacy standard level *Use of 2-key 3DES restricted to 10^6 plaintext/ciphertexts, protection from 2008 to 2016*	96	1776	192	1776	192	192
6	Medium-term protection *protection from 2008 to 2026*	112	2432	224	2432	224	224
7	Long-term protection *Generic application-independent recommendation, protection from 2008 to 2036*	128	3248	256	3248	256	256
8	"Foreseeable future" *Good protection against quantum computers*	256	15424	512	15424	512	512

FIGURE III.2 – Sécurité fournie selon la taille de la clé.

III.3.2 Attaques par dictionnaire

Lorsque la clé est un mot (p.ex. un mot de passe), on peut tenter de court-circuiter la Force Brute. Le principe est ici d'utiliser un recueil de mots possibles (le dictionnaire), et de tester tous les mots de ce dictionnaire. Attention à bien distinguer les deux attaques : on teste tous les mots **du dictionnaire** mais celui-ci ne contient pas *toutes* les possibilités. Par exemple, on pourra y trouver "unie", "unir" et "unis", mais pas "unih" (pour autant qu'il s'agisse d'un dictionnaire de mots existant dans la langue française).

III.3.3 Analyse de fréquence

Cette analyse repose sur l'obtention d'indices précieux : quelle est la langue utilisée ? quelle est le thème du texte ?
Il faut toutefois que la taille de K soit inférieure à la taille de C, au risque de ne pas permettre l'unicité de la solution. Il faut aussi que le texte C soit suffisamment long pour être représentatif. Et enfin que l'algorithme utilisé soit une substitution simple (mono- ou polyalphabétique).

III.3.4 Cryptanalyse différentielle

Il s'agit de l'étude (modélisation) des transformations subies par le message durant son passage dans l'algorithme de chiffrement. Le principe est de modéliser ce qu'une modification en entrée induira sur le résultat de l'algorithme.

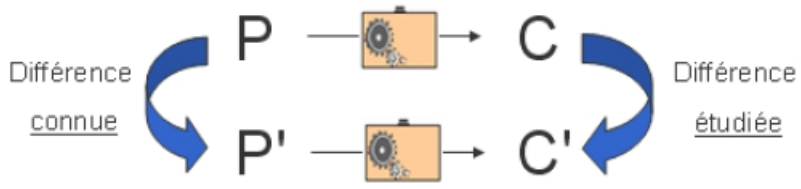

FIGURE III.3 – Cryptanalyse différentielle

III.3.5 Cryptanalyse linéaire

Le but est d'effectuer une approximation linéaire de l'algorithme de chiffrement. Il n'y a ici aucune possibilité de choisir le texte clair à chiffrer, on dispose tout au plus d'un ensemble de couples (M, C). On tente alors de découvrir une expression de la forme

$$M_{i1} \oplus M_{i2} \oplus ... \oplus M_{iu} \oplus Cj1 \oplus Cj2 \oplus ... \oplus C_{jv}$$

avec M_i représentant le i^{eme} bit de $M = [M_1, M_2, ...M_u]$ et C_j représentant le j^{eme} bit de $C = [C_1, C_2, ...C_v]$. Si c'est le cas, et selon la probabilité d'occurrence de l'expression, on peut déduire des faiblesses de l'algorithme (en termes de transformations aléatoires).

Remarque. *Cryptanalyses différentielle et linéaire sont des outils parfaits pour comparer la résistance de divers chiffrements. Aucun algorithme cryptographique n'est dit valable s'il ne résiste pas à ce type de cryptanalyse.*

La résistance à ces attaques ne signifie pas résistance contre toutes les autres méthodes inconnues.

Sans une connaissance approfondie des techniques de cryptanalyse, il n'est pas possible de créer un algorithme de chiffrement performant, sûr et robuste.

III.3.6 Meet in the middle

Cette attaque est souvent illustrée par l'attaque ayant démontré les faiblesses du DES. Nous allons dès lors reprendre les explications fournies dans le chapitre vu précédemment. Pour rappel, le 2DES fonctionne de la manière suivante

$$C = E_{K_2}(E_{K_1}(M))$$

On suppose que l'attaquant dispose d'un couple (M, C). Il chiffre M avec les 2^{56} clés f d'un coté et déchiffre C avec les 2^{56} clés g de l'autre. Lorsque $E_f(M) = D_g(C)$, il sait qu'il a découvert les 2 clés utilisées. La figure III.4 illustre le principe adopté dans cette attaque. Un nombre de clés possibles réduit rend cette attaque utilisable.

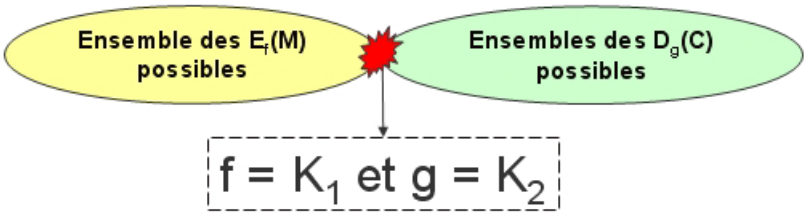

FIGURE III.4 – Meet in the middle.

III.3.7 Man in the middle

Souvent confondue avec l'attaque précédente en raison de son acronyme MITM *(Man In The Middle)*, elle est pourtant très différente dans les faits. Son déroulement est illustré à la figure III.5 ci-dessous : le pirate se fait passer pour B auprès de A et pour A auprès de B.

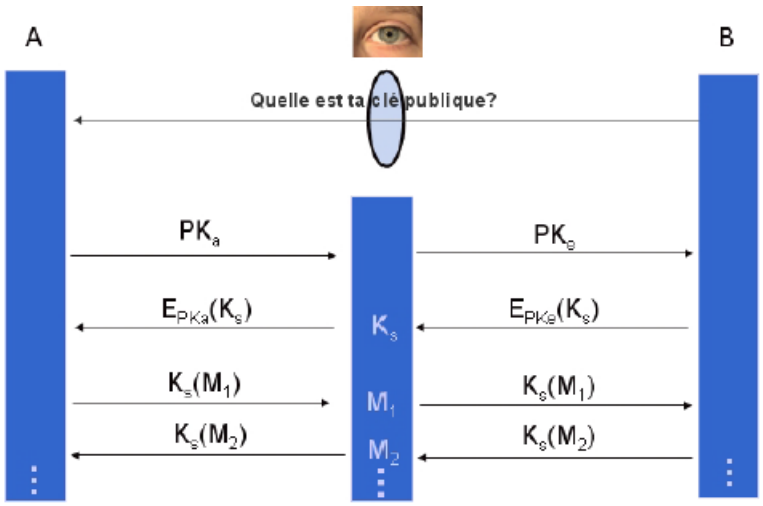

FIGURE III.5 – Man in the middle.

III.3.8 Quelques attaques logicielles

Quelques autres attaques logicielles répandues sont fréquemment citées :

1. Mascarade/Déguisement : attaquer le service d'authentification d'un système en se faisant passer pour une personne digne de confiance.
2. Attaque par Rejeu : répéter un envoi de données ou une séquence d'actions sans les modifier (répéter un ordre de crédit par exemple).
3.

La liste établie précédemment n'est bien évidemment pas exhaustive. Il existe un très grand nombre d'autres attaques, souvent plus spécifiques à un algorithme particulier.

III.4 Attaques des fonctions de hachage

III.4.1 Utilisation du paradoxe de l'anniversaire

Nous avons déjà traité le cas des attaques par le paradoxe de l'anniversaire *(Birthday Attack)*. Celui-ci consistait à traiter les variations d'un texte clair et d'un texte frauduleux pour déterminer un texte frauduleux de substitution.

III.4.2 Le compromis temps-mémoire

L'approche est ici radicalement différente. Il ne s'agit plus de substituer une chaine par une autre tout en conservant un même haché, mais bel et bien de retrouver la chaine initiale, ayant donc servi à créer un haché donné. En d'autre termes, retrouver le x dans $H(x) = h$, H et h étant connus.
Ce type d'attaque est principalement utilisé dans le contexte du cassage de mots de passe : on voudrait retrouver le mot de passe correspondant à un haché donné, sachant que la plupart des systèmes de protection par mot de passe, notamment les OS, *(Operating System)*, fonctionnent par comparaison du mot de passe entré par l'utilisateur avec le haché du mot de passe attendu (haché créé lors de la définition de ce mot de passe et stocké en clair dans l'OS).

- Tester tous les textes clairs possibles (mais selon la taille, cela demanderait trop de temps)
- Stocker tous les résultats (mais demanderait trop d'espace mémoire).

La solution intermédiaire porte ainsi le nom de compromis temps-mémoire. Une nouvelle notion entre en ligne de compte : la Fonction de Réduction (R). Elle consiste en le processus inverse d'une fonction de hachage : elle fournit un texte clair à partir d'un haché donné. Il s'agira d'une fonction spécifique mais dont le résultat devra être pseudo-aléatoire (par exemple, le fait de prendre les premiers caractères du haché).

Cas 1 : La fonction de réduction uniques

Soit une chaine de départ. Le principe consiste à lui appliquer en alternance un certain nombre de fois (ici, 3) une fonction de hachage puis une fonction de réduction, voir la figure III.6. Une table est conservée en mémoire et constituée des premières et dernières chaines respectives (on parle de *point de départ* et d'arrivée de chaines). D'où le terme de *compromis*. Pour un haché donné, l'algorithme de "cassage" est le suivant :

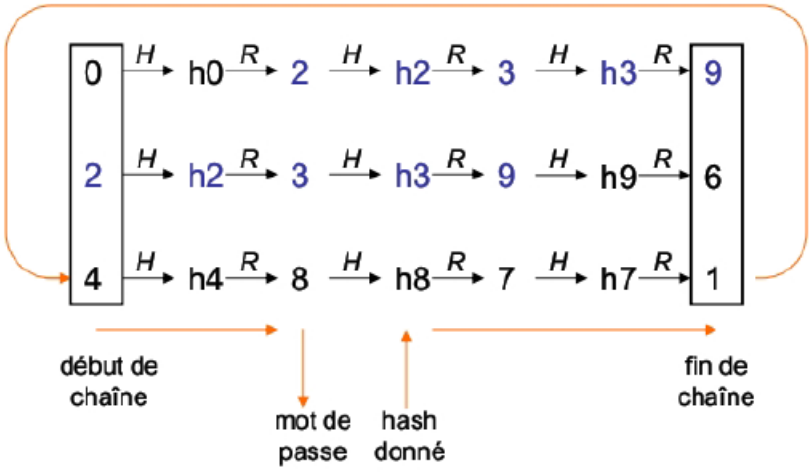

FIGURE III.6 – Compromis temps-mémoire par fonction de réduction unique.

1. On calcule sa réduction
2. Le résultat est-il dans la dernière colonne ?
   - Oui, on recalcule la chaine à partir de son point d'entrée, et on obtiendra la chaine initiale correspondant au haché donné.
   - Non, on lui applique H puis on retourne à 1.

On stoppe le processus une fois que le nombre de réductions appliquées sur le haché donné atteint le nombre de réductions présentes dans une chaine (ici, 3). Si aucun résultat n'est obtenu, c'est que notre table n'est pas assez complète, dans le cas d'une table non-exhaustive telle celle de la figure III.6.

Le problème dans l'exemple précédent est qu'il est fréquent de se trouver en présence d'une fusion de chaines, c'est-à-dire au fait qu'une réduction correspond à plusieurs hachés. Ce cas est inévitable lorsque la fonction de réduction produit un résultat de taille inférieure à la taille des hachés.

La solution consiste à utiliser des fonctions de réduction différentes à chaque étape intermédiaire. Les fusions sont alors beaucoup moins probables puisqu'il faudrait que le même mot de passe apparaisse au même endroit dans deux chaines distinctes.

Cas 2 : Attaque Arc-en-ciel

Pour un haché donné, l'algorithme de *cassage*, illustré par la figure III.7, est le suivant :

- On calcule sa réduction $R3$
- Le résultat est il dans la dernière colonne ?
  - Oui, on recalcule la chaine à partir de son point d'entrée, et on obtiendra la chaine initiale correspondant au haché donné.
  - Non, on lui applique $R2$ puis H puis $R3$
  - Le résultat est dans la dernière colonne ?
    - Oui, on recalcule la chaine à partir du début
    - Non, on lui applique $R1$ puis H puis $R2$ puis H puis $R3$

Si aucun résultat n'est obtenu, on sait que notre table n'est pas assez complète. Il faudra alors soit augmenter le nombre de chaines traitées dans la table, soit augmenter le nombre de tables avec utilisant d'autres fonctions de réduction.

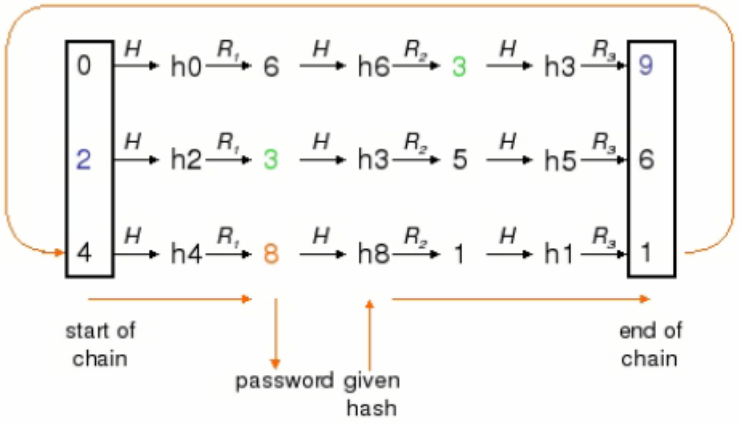

FIGURE III.7 – Compromis temps-mémoire par fonction de réduction unique.

III.5 Attaques par canaux auxiliaires

Une famille d'attaques n'est pas basée sur les données elles-même, mais plutôt sur leur environnement. On les appelle les attaques par canaux auxiliaires, ou Side Channel Attacks. Plus précisément, le principe de telles attaques est d'étudier l'aspect physique d'un cryptosystème sous différents angles au lieu d'étudier son aspect logique (algorithmique, mathématique).
On peut classer ces attaques de différentes façons :

III.5.1 Actives ou passives

1. Actives : tenter de modifier le comportement de l'élément (par exemple en induisant des erreurs)
2. Passives : observer le comportement de l'élément sans chercher à la modifier.

III.5.2 Invasives ou non-invasives

1. Invasives : ouvrir le système (puce, smartcard, etc.) pour accéder aux composants (cellules mémoires, bus, etc.)
2. Non-invasives : recueillir l'information depuis l'extérieur sans modifier l'élément. Ces catégories ne sont pas exclusives : une attaque peut être active et invasive, ou inversément.

Ces catégories ne sont pas exclusives : une attaque peut être active et invasive, ou inversément.
De nombreux types d'attaques appartiennent à cette branche de la cryptanalyse :

– **Fault Induction Attack** : elle consiste à modifier l'environnement du cryptosystème afin de provoquer d'éventuelles erreurs, celles-ci pouvant fournir des informations (stacktrace, etc.). On peut par exemple faire varier la température, le voltage, la fréquence, induire des champs magnétiques intenses,...etc

– **Optical Fault Induction Attack** : Grace à un faisceau lumineux, il est possible de modifier le contenu des cellules mémoire.

– **Power Analysis Attack** : analyser la consommation électrique des composants durant une phase de calcul.

– **Timing Attack** : évaluer les temps de calcul nécessaires (voir II.2.2)

– **Electromagnetic Analysis Attack** : étude du rayonnement électromagnétique d'un système afin de connaître les données traitées et les calculs réalisés. Une exemple très connu de ce type d'attaque est dénommée "Tempest Attack".

Chapitre IV

Sécurité réseaux

Contents

IV.1 Introduction

Les systèmes d'information sont omniprésents dans toutes les entreprises. La sécurité de ces systèmes doit les protéger contre de nombreuses menaces de diverses origines. L'analyse de risques permet de déterminer, en fonction de la vulnérabilité du système, sa criticité pour chacune de ces menaces. Elle permet ensuite de proposer les solutions nécessaires et suffisantes pour réduire les risques à un niveau résiduel acceptable.

Après une introduction sur les risques informatiques en général, ce chapitre se préoccupe plus particulièrement des systèmes d'information et présente les méthodes d'analyse de risques et quelques solutions en réduction de risques. En conclusion, un panorama des principales normes employées en cybersécurité est donné. Un dispositif de sécurité n'est efficace que s'il est correctement administré et supervisé.

IV.2 Services et mécanismes de sécurité

L'authentification est le premier rempart aux attaques informatique, il s'agit la plupart du temps du couple *nom d'utilisateur/mot de passe* (LOGIN/PASSWORD). Cette première méthode constitue une sécurité relativement fiable lorsqu'elle est bien utilisée : mot de passe correct, confidentialité assurée, fichier protégé,...etc.

Elle pose tout de même certains problèmes comme par exemple le cas où un utilisateur a besoin de se connecter sur plusieurs stations différentes. Il va alors rapidement trouver cette méthode d'authentification relativement lourde (considérons que l'utilisateur est une personne "sérieuse" qui possède un mot de passe différent sur chaque station ou que le réseau utilise NIS *(Network Information Service)*).

Il sera alors tenter d'utiliser une méthode moins contraignante : les services r^*. Ces services (rlogin, rsh, rcp,...) sont basés sur une authentification beaucoup moins lourde pour l'utilisateur mais également beaucoup moins sûre : l'adresse IP *(Internet Protocol)* de la station source ainsi que le "nom d'utilisateur". Ce type d'authentification est, malgrés de nombreux problèmes de sécurité, encore très utilisé dans les réseaux locaux actuels.

Une autre méthode d'authentification permettant un niveau de sécurité plus élevé utilise un serveur d'authentification AS, *(Authentification Server)*, qui permet d'éviter les problèmes engendrés par l'authentification IP. Le serveur d'authentification reçoit une demande de connection et, afin de vérifier que la station est bien celle qu'elle prétend être, établit une seconde connection TCP *(Transfert Control Protocol)* avec la station source (qui ne sera donc pas sous le contrôle d'un éventuel attaquant).

Une méthode beaucoup plus sécurisée, mais plus contraignante et plus difficile à mettre en oeuvre, appellée OTP *(One-Time Password system)* permet d'éviter les problèmes de circulation des mots de passe sur le réseau (Sniffing,...). Se sont des pseudo mots de passe qui circulent alors sur le réseau et dont la validité est restreinte à la dite connection. Le fonctionnement des One-Time Passwords peut se résumer ainsi : le client demande une connection à un serveur, ce dernier renvoit un Challenge au client qui devra alors générer un One-Time Password à l'aide d'un mot de passe (ou d'une phrase) connu des deux stations ainsi que du Challenge reçu. Ce pseudo mot de passe sera alors retransmis au serveur qui pourra ainsi le vérifier et autoriser (ou non) la connection.

Actuellement, le programme SSH *(Secure SHell)* est trés apprécié, il peut apporter la facilité des authentifications par adresses IP tout en assurant une trés bonne sécurité grâce à un système d'encryptage des données qui transitent sur le réseau. Il utilise pour cela un mécanisme d'encryptage assymétrique (principe des clés publiques et privées) bien connu. Il

s'agit du système RSA. Cette solution peut permettre aux utilisateurs de ne pas être obligé d'entrer à chaque fois leur mot de passe tout en évitant les problèmes des services r^* ou du telnet (Sniffing).

D'autres méthodes d'authentification existent également, comme par exemple Kerberos *(gestion de tickets)*, SSL *(Secure Socket Layer)*, S-HTTP *(Secure Hyper Text Transfert Protocol)*, les signatures digitales *(Digital Signature)*, les certifications, ...etc.

IV.2.1 Les firewalls

Le filtrage de datagrammes IPsec est délicat pour deux raisons :

- les normes adoptées ne précisent pas si, sur un système remplissant simultanément les fonctions de passerelle de sécurité et de Firewall *(pare-feu)*, le décodage de l'IPsec doit avoir lieu avant ou après l'application des règles de firewalling.
- il n'est pas possible au code de firewalling de lire certaines données, par exemple des numéros de port, dans des données chiffrées, ou transmises dans un format qu'il ne connaît pas.

Il est donc important de lire la documentation de l'implémentation IPsec utilisée sur les systèmes destinés à gérer simultanément IPsec et firewalling. Il est également utile de noter que les systèmes qui appliquent les règles de firewalling avant le décodage IPsec sont néanmoins souvent capables de filtrer les datagrammes décodés en utilisant des outils de tunneling comme ipip et gif, ou en filtrant au niveau de l'interface de sortie des données. Enfin, AH *(Authentication Header)* utilise le numéro de protocole 51 et ESP *(Encapsulating Security Payload)* le numéro de port 50. Quant à IKE *(Internet Key Exchange)*, il utilise le numéro de port 500 avec UDP *(User Datagram Protocol)*. Ces informations sont indispensables lors de la configuration d'un firewall qui doit laisser passer ou bloquer les échanges IPsec.

IV.2.2 Les VPNs

Les applications et les systèmes distribués font de plus en plus partie intégrante du paysage d'un grand nombre d'entreprises. Ces technologies ont pu se développer grâce aux performances toujours plus importantes des réseaux locaux. Mais le succès de ces applications a fait aussi apparaître un de leur écueil. En effet si les applications distribuées deviennent le principal outil du système d'information de l'entreprise, comment assurer leur accès sécurisé au sein de structures parfois réparties sur de grandes distances géographiques ? Concrètement comment une succursale d'une entreprise peut-elle accéder aux données situées sur un serveur de la maison mère distant de plusieurs milliers de kilomètres ? Les VPN *(Virtual Private Network)*, ont commencé à être mis en place pour répondre à ce type de problématique. Mais d'autres problématiques sont apparues et les VPN ont aujourd'hui pris une place importante dans les réseaux informatique et l'informatique distribuées. Nous verrons ici quelles sont les principales caractéristiques des VPN à travers un certain nombre d'utilisation type. Nous nous intéresserons ensuite aux protocoles permettant leur mise en place.

Principe de fonctionnement du VPN

Un réseau VPN repose sur un protocole appelé « protocole de tunneling ». Ce protocole permet de faire circuler les informations de l'entreprise de façon cryptée d'un bout à l'autre

du tunnel. Ainsi, les utilisateurs ont l'impression de se connecter directement sur le réseau de leur entreprise.

Le principe de tunneling consiste à construire un chemin virtuel après avoir identifié l'émetteur et le destinataire. Par la suite, la source chiffre les données et les achemine en empruntant ce chemin virtuel. Afin d'assurer un accès aisé et peu coûteux aux intranets ou aux extranets d'entreprise, les réseaux privés virtuels d'accès simulent un réseau privé, alors qu'ils utilisent en réalité une infrastructure d'accès partagée, comme Internet.

Les données à transmettre peuvent être prises en charge par un protocole différent d'IP. Dans Ce cas, le protocole de tunneling encapsule les données en ajoutant une entête. Le tunneling est l'ensemble des processus d'encapsulation, de transmission et de désencapsulation.

Fonctionnalités des VPN

Il existe 3 types standard d'utilisation des VPN. En étudiant ces schémas d'utilisation, il est possible d'isoler les fonctionnalités indispensables des VPN.

1. **Le VPN d'accès** est utilisé pour permettre à des utilisateurs itinérants d'accéder au réseau privé. L'utilisateur se sert d'une connexion Internet pour établir la connexion VPN.

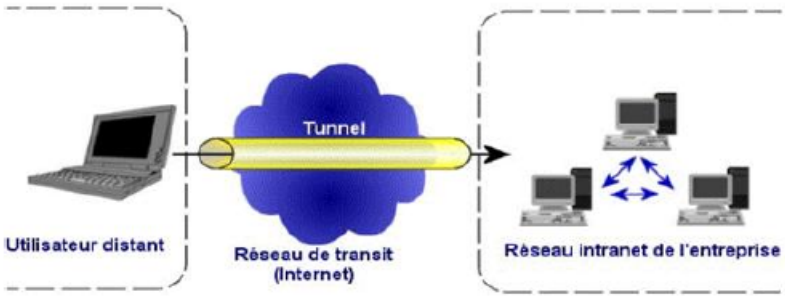

FIGURE IV.1 – VPN d'accès.

Il existe deux cas :

- L'utilisateur demande au fournisseur d'accès de lui établir une connexion cryptée vers le serveur distant : il communique avec le NAS *(Network Access Server)* du fournisseur d'accès et c'est le NAS qui établit la connexion cryptée.
- L'utilisateur possède son propre logiciel client pour le VPN auquel cas il établit directement la communication de manière cryptée vers le réseau de l'entreprise.

Les deux méthodes possèdent chacune leurs avantages et leurs inconvénients :

- La première permet à l'utilisateur de communiquer sur plusieurs réseaux en créant plusieurs tunnels, mais nécessite un fournisseur d'accès proposant un Nas compatible avec la solution VPN choisie par l'entreprise. De plus, la demande de connexion par le Nas n'est pas cryptée Ce qui peut poser des problèmes de sécurité.
- Sur la deuxième méthode ce problème disparaît puisque l'intégralité des informations sera cryptée dès l'établissement de la connexion. Par contre, cette solution nécessite que chaque client transporte avec lui le logiciel, lui permettant d'établir une communication cryptée.

  Nous verrons que pour pallier Ce problème certaines entreprises mettent en place des VPN à base de SSL, technologie implémentée dans la majorité des navigateurs Internet du marché.

Quelle que soit la méthode de connexion choisie, Ce type d'utilisation montre bien l'importance dans le VPN d'avoir une authentification forte des utilisateurs. Cette authentification peut se faire par une vérification « login / mot de passe », par un algorithme dit « Tokens sécurisés » (utilisation de mots de passe aléatoires) ou par certificats numériques.

2. **L'intranet VPN** est utilisé pour relier au moins deux intranets entre eux. Ce type

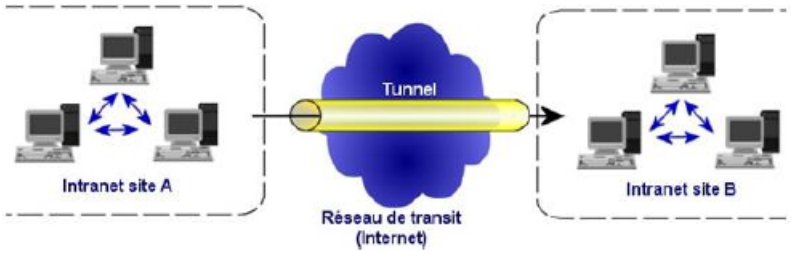

FIGURE IV.2 – VPN intranet.

de réseau est particulièrement utile au sein d'une entreprise possédant plusieurs sites distants. Le plus important dans Ce type de réseau est de garantir la sécurité et l'intégrité des données. Certaines données très sensibles peuvent être amenées à transiter sur le VPN (base de données clients, informations financières...).

Des techniques de cryptographie sont mises en oeuvre pour vérifier que les données n'ont pas été altérées. Il s'agit d'une authentification au niveau paquet pour assurer la validité des données, de l'identification de leur source ainsi que leur non-répudiation. La plupart des algorithmes utilisés font appel à des signatures numériques qui sont ajoutées aux paquets.

La confidentialité des données est, elle aussi, basée sur des algorithmes de cryptographie. La technologie en la matière est suffisamment avancée pour permettre une sécurité quasi parfaite. Le coût matériel des équipements de cryptage et décryptage ainsi que les limites légales interdisent l'utilisation d'un codage » infaillible « . Généralement pour la confidentialité, le codage en lui-même pourra être moyen à faible, mais sera combiné avec d'autres techniques comme l'encapsulation Ip dans Ip pour assurer une sécurité raisonnable.

3. **L'extranet VPN** : une entreprise peut utiliser le VPN pour communiquer avec ses

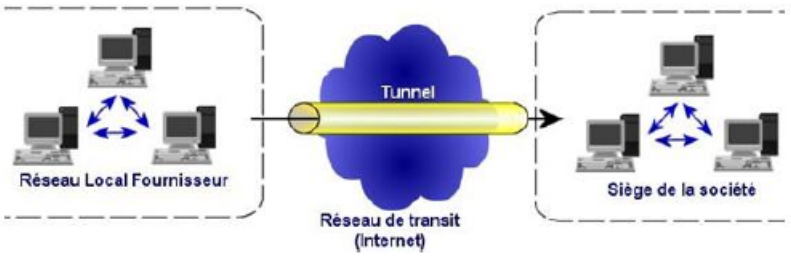

FIGURE IV.3 – VPN extranet.

clients et ses partenaires. Elle ouvre alors son réseau local à ces derniers. Dans Ce cadre, il est fondamental que l'administrateur du VPN puisse tracer les clients sur le réseau et gérer les droits de chacun sur celui-ci.

IV.2.3 IPsec

Le fonctionnement du protocole IPsec est similaire à celui du protocole SSL. Il est standardisé par l'IETF *(Internet Engineering Task Force)*. Le protocole IPsec a été conçu pour assurer la sécurité du protocole IP. Il faut ajouter ce protocole de sécurité au protocole IP. La sécurité du protocole est prise en compte de manière native dans la version 6 du protocole IP, ce qui n'est pas le cas de la version 4 utilisée actuellement.

Il existe deux modes : l'un permettant de chiffrer de bout en bout et l'autre permettant la création de tunnels sécurisés, c'est-à-dire que les données sont chiffrées uniquement entre deux routeurs par exemple et non à l'intérieur du réseau de l'entreprise supposé sécurisé. Dans le cas d'un tunnel, tout le paquet IP est chiffré alors que seule la partie « données » est chiffrée dans le cas d'un chiffrement de bout en bout car on ne peut pas chiffrer l'en-tête qui contient les adresses IP. Dans le cas d'un tunnel, les protocoles IPsec garantissent la confidentialité de l'adresse IP. Cela peut aussi être garanti en utilisant des mécanismes de translation d'adresse sur le routeur de sortie de l'entreprise.

Durant la première phase, le protocole IKE permet d'échanger la clé maître, d'authentifier les entités et de gérer les associations de sécurité SA *(Security Association)*. Les associations de sécurité contiennent les algorithmes de chiffrement et de MAC, et les clés de chiffrement et d'authentification des données. Ces SA sont stockées dans des bases de données de sécurité, SADB, pour *(Security Association DataBase)*, en utilisant comme clé de la base de données l'adresse IP de la machine émettrice et le numéro de SA. Ainsi, quand une station du réseau reçoit un paquet IP sécurisé, elle utilise le bon algorithme de chiffrement et de MAC *(Message Authentication Code)* et la bonne clé de chiffrement et de MAC.

Durant la deuxième phase, le protocole ESP permet de garantir la confidentialité et l'intégrité des données, et de l'en-tête dans le cas d'un tunnel, alors que le protocole AH permet de garantir l'intégrité des paquets IP, y compris certaines données de l'en-tête des paquets IP. En effet, les champs qui varient, comme le nombre de routeurs traversés, ne peuvent pas être utilisés pour calculer le MAC car les routeurs intermédiaires n'ont pas la clé de MAC.

IV.2.4 Les services offerts par IPsec

Les deux modes d'échange IPsec

Une communication entre deux hôtes, protégée par IPsec, est susceptible de fonctionner suivant deux modes différents : le mode transport et le mode tunnel. Le premier offre essentiellement une protection aux protocoles de niveau supérieur, le second permet quant à lui d'encapsuler des datagrammes IP dans d'autres datagrammes IP, dont le contenu est protégé. L'intérêt majeur de ce second mode est qu'il rend la mise en place de passerelles de sécurité qui traitent toute la partie IPsec d'une communication et transmettent les datagrammes épurés de leur partie IPsec à leur destinataire réel réalisable. Il est également possible d'encapsuler une communication IPsec en mode tunnel ou transport dans une autre communication IPsec en mode tunnel, elle-même traitée par une passerelle de sécurité, qui transmet les datagrammes après suppression de leur première enveloppe à un hôte traitant à son tour les protections restantes ou à une seconde passerelle de sécurité.

Les protocoles à la base d'IPsec

1. **AH (authentication header)** est le premier et le plus simple des protocoles de protection des données qui font partie de la spécification IPsec. Il a pour vocation de

garantir :

- L'authentification : les datagrammes IP reçus ont effectivement été émis par l'hôte dont l'adresse IP est indiquée comme adresse source dans les entêtes.
- L'unicité (optionnelle, à la discrétion du récepteur) : un datagramme ayant été émis légitimement et enregistré par un attaquant ne peut être réutilisé par ce dernier, les attaques par rejeu sont ainsi évitées.
- L'intégrité : les champs suivants du datagramme IP n'ont pas été modifiés depuis leur émission : les données (en mode tunnel, ceci comprend la totalité des champs, y compris les entêtes, du datagramme IP encapsulé dans le datagramme protégé par AH), version (4 en IPv4, 6 en IPv6), longueur de l'entête (en IPv4), longueur totale du datagramme (en IPv4), longueur des données (en IPv6), identification, protocole ou entête suivant (ce champ vaut 51 pour indiquer qu'il s'agit du protocole AH), adresse IP de l'émetteur, adresse IP du destinataire (sans source routing).

En outre, au cas où du source routing serait présent, le champ adresse IP du destinataire a la valeur que l'émetteur a prévu qu'il aurait lors de sa réception par le destinataire. Cependant, la valeur que prendront les champs type de service (IPv4), indicateurs (IPv4), index de fragment (IPv4), TTL *(Time To Live)* de (IPv4), somme de contrôle d'entête (IPv4), classe (IPv6), flow label (IPv6), et hop limit (IPv6) lors de leur réception n'étant pas prédictible au moment de l'émission, leur intégrité n'est pas garantie par AH.

L'intégrité de celles des options IP qui ne sont pas modifiables pendant le transport est assurée, celle des autres options ne l'est pas. Attention, AH n'assure pas la confidentialité : les données sont signées mais pas chiffrées.

Enfin, AH ne spécifie pas d'algorithme de signature particulier, ceux-ci sont décrits séparément, cependant, une implémentation conforme à la AH est tenue de supporter les algorithmes MD-5 *(Message Digest-5)* et SHA-1 *(Secure Hash Algorithm-1)*.

2. **ESP (encapsulating security payload)**

ESP est le second protocole de protection des données qui fait partie de la spécification IPsec. Contrairement à AH, ESP ne protège pas les entêtes des datagrammes IP utilisés pour transmettre la communication. Seules les données sont protégées. En mode transport, il assure :

- La confidentialité des données (optionnelle) : la partie données des datagrammes IP transmis est chiffrée.
- L'authentification (optionnelle, mais obligatoire en l'absence de confidentialité) : la partie données des datagrammes IP reçus ne peut avoir été émise que par l'hôte avec lequel a lieu l'échange IPsec, qui ne peut s'authentifier avec succès que s'il connaît la clef associée à la communication ESP. Il est également important de savoir que l'absence d'authentification nuit à la confidentialité, en la rendant plus vulnérable à certaines attaques actives.
- L'unicité (optionnelle, à la discrétion du récepteur).
- L'integrité : les données n'ont pas été modifiées depuis leur émission.

En mode tunnel, ces garanties s'appliquent aux données du datagramme dans lequel est encapsulé le trafic utile, donc à la totalité (entêtes et options inclus) du datagramme encapsulé. Dans ce mode, deux avantages supplémentaires apparaissent :

- Une confidentialité, limitée, des flux de données (en mode tunnel uniquement, lorsque la confidentialité est assurée) : un attaquant capable d'observer les données transitant par un lien n'est pas à même de déterminer quel volume de données est transféré entre deux hôtes particuliers. Par exemple, si la communication entre deux sous-réseaux est chiffrée à l'aide d'un tunnel ESP, le volume total de données échangées entre ces deux sous-réseaux est calculable par cet attaquant, mais pas la répartition de ce volume entre les différents systèmes de ces sous-réseaux.
- La confidentialité des données, si elle est demandée, s'étend à l'ensemble des champs, y compris les entêtes, du datagramme IP encapsulé dans le datagramme protégé par ESP).

Enfin, ESP ne spécifie pas d'algorithme de signature ou de chiffrement particulier, ceux-ci sont décrits séparément, cependant, une implémentation conforme à la norme donnée par *RFC* 2406 est tenue de supporter l'algorithme de chiffrement DES en mode CBC, et les signatures à l'aide des fonctions de hachage MD5 et SHA-1.

3. **Implantation d'IPSec dans le datagramme IP** La figure IV.4 montre comment les données nécessaires au bon fonctionnement des formats AH et ESP sont placées dans le datagramme IPv4. Il s'agit bien d'un ajout dans le datagramme IP, et non de nouveaux datagrammes, ce qui permet un nombre théoriquement illimité ou presque d'encapsulations IPsec : un datagramme donné peut par exemple être protégé à l'aide de trois applications successives de AH et de deux encapsulations de ESP.

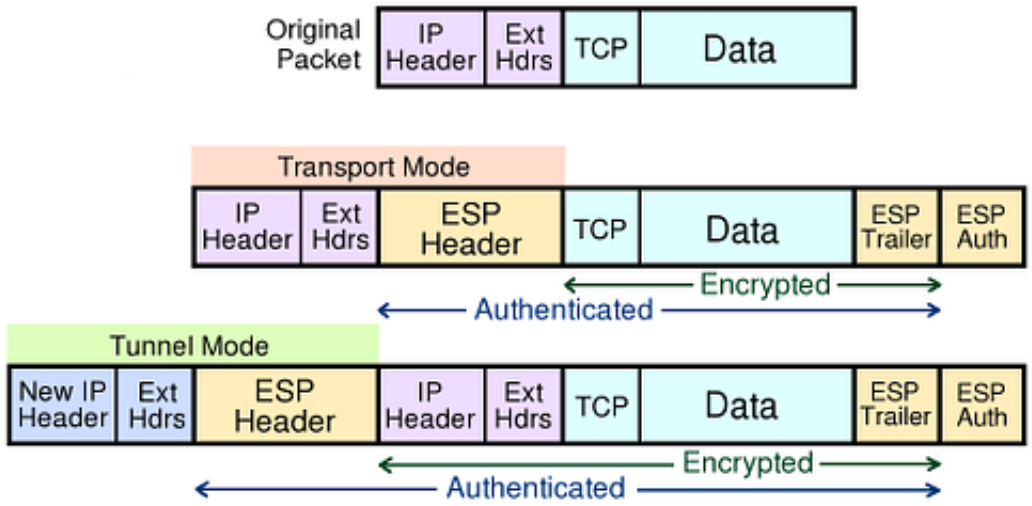

FIGURE IV.4 – Entête AH et ESP en mode, transport et tunnel, respectivement.

IV.3 Politique et architecture de sécurité

IV.3.1 Associations de sécurité

Le protocole SAIP *(Security Architecture for the Internet Protocol)* décrit le protocole IPsec au niveau le plus élevé. En particulier, elle indique ce qu'une implémentation est censée permettre de configurer en termes de politique de sécurité (c'est-à-dire quels échanges IP doivent être protégés par IPsec et, le cas échéant, quel(s) protocole(s) utiliser). Sur chaque système capable d'utiliser IPsec doit être présente une SPD *(Security Policy Database)*, dont

la forme précise est laissée au choix de l'implémentation, et qui permet de préciser la politique de sécurité à appliquer au système. Chaque entrée de cette base de données est identifiée par un SPI *(Security Parameters Index)* unique (32 bits) choisi arbitrairement.

Une communication protégée à l'aide d'IPsec est appelée une SA. Une SA repose sur une unique application de AH ou sur une unique application de ESP. Ceci n'exclut pas l'usage simultané de AH et ESP entre deux systèmes, ou par exemple l'encapsulation des datagrammes AH dans d'autres datagrammes AH, mais plusieurs SA devront alors être activées entre les deux systèmes. En outre, une SA est unidirectionnelle. La protection d'une communication ayant lieu dans les deux sens nécessitera donc l'activation de deux SA. Chaque SA est identifiée de manière unique par un SPI, une adresse IP de destination (éventuellement adresse de broadcast ou de multicast), et un protocole (AH ou ESP). Les SA actives sont regroupées dans une SAD *(Security Association Database)*.

La SPD est consultée pendant le traitement de tout datagramme IP, entrant ou sortant, y compris les datagrammes non-IPsec. Pour chaque datagramme, trois comportements sont envisageables : le rejeter, l'accepter sans traitement IPsec, ou appliquer IPsec. Dans le troisième cas, la SPD précise en outre quels traitements IPsec appliquer (ESP, AH, mode tunnel ou transport, quel(s) algorithme(s) de signature et/ou chiffrement utiliser).

Les plus importants de ces termes sont récapitulés dans le tableau suivant :

- SPD base de données définissant la politique de sécurité
- SA une entrée de la SPD
- SAD liste des SA en cours d'utilisation

Les règles de la SPD doivent pouvoir, si l'adminsitrateur du système le souhaite, dépendre des paramètres suivants :

- adresse ou groupe d'adresses IP de destination
- adresse ou groupe d'adresses IP source
- nom du système (DNS *(Domain Name System)* complète, nom X.500 distingué ou général)
- protocole de transport utilisé (typiquement, TCP ou UDP)
- nom d'utilisateur complet, comme foo@bar.net (ce paramètre n'est toutefois pas obligatoire sur certains types d'implémentations) importance des données (ce paramètre n'est toutefois pas obligatoire sur certains types d'implémentations)
- ports source et destination (UDP et TCP seulement, le support de ce paramètre est facultatif)

Certains paramètres peuvent être illisibles à cause d'un éventuel chiffrement ESP au moment où ils sont traités, auquel cas la valeur de la SPD les concernant peut le préciser, il s'agit de :

- l'identité de l'utilisateur
- le protocole de transport
- les ports source et destination

Il est donc possible de spécifier une politique de sécurité relativement fine et détaillée. Cependant, une politique commune pour le trafic vers un ensemble de systèmes permet une meilleure protection contre l'analyse de trafic qu'une politique extrêmement détaillée, comprenant de nombreux paramètres propres à chaque système particulier. Enfin, si l'implémentation permet aux applications de préciser elles-mêmes à quelle partie du trafic appliquer IPsec et comment, l'administrateur doit pouvoir les empêcher de contourner la politique par défaut.

IV.3.2 Architectures supportées

Le protocole IPsec permet théoriquement n'importe quelle combinaison, en un nombre quasiment illimité de niveaux d'encapsulation, c'est-à-dire d'accumulations de SA. Néanmoins, les implémentations ne sont pas tenues d'offrir une telle flexibilité. Les architectures qu'une application conforme à la RFC 2401 doivent supporter, au nombre de quatre, sont décrites ci-dessous.

Dialogue entre deux hôtes protégeant le trafic eux-mêmes

Deux hôtes engagent une communication IPsec, en encapsulant le protocole de haut niveau dans : en mode transport :

- des datagrammes AH
- des datagrammes ESP
- ou des datagrammes ESP encapsulés dans des datagrammes AH

en mode tunnel :

- des datagrammes AH
- ou des datagrammes ESP

Dialogue entre deux LANs à l'aide de passerelles de sécurité

Ici, deux passerelles de sécurité gèrent les conversations entre les hôtes de deux LANs *(Local Area Network)* différents, IPsec s'appliquant de manière transparente pour les hôtes. Les deux passerelles de sécurité échangent des datagrammes en mode tunnel, à l'aide de AH ou ESP (après routage, ceci correspond simplement à la deuxième partie du cas précédent), puis transmettent les datagrammes obtenus après traitement IPsec aux hôtes destinataires. Ainsi, les datagrammes IP émis par un système de l'un des deux LANs sont encapsulés dans d'autres datagrammes IP+IPsec par la passerelle du « LAN émetteur », encapsulation supprimée par la passerelle du « LAN récepteur » pour obtenir de nouveau les datagrammes IP originaux.

Dialogue entre deux hôtes traversant deux passerelles de sécurité

Le troisième cas n'ajoute pas vraiment de prérequis sur les implémentations IPsec. Ainsi, une passerelle de sécurité doit avoir la capacité de transmettre dans un tunnel IPsec du trafic déjà sécurisé par IPsec. Celà autorise les dialogues IPsec entre deux hôtes situés eux-mêmes dans des LANs reliés par des passerelles de sécurité.

Dialogue entre un hôte et une passerelle de sécurité

Le dernier cas est celui d'un hôte externe se connectant à un LAN protégé par une passerelle de sécurité. Les documentations techniques désignent souvent un tel hôte sous le nom de road warrior. Il engage une conversation IPsec avec un système du LAN en mode transport, le tout encapsulé dans un tunnel IPsec établi entre le road warrior lui-même et la passerelle de sécurité. Dans ce cas, les datagrammes du tunnel sont de type AH ou ESP et les datagrammes utilisés en mode transport doivent pouvoir être de type AH, ESP, ou ESP encapsulé dans AH.

IV.3.3 Gestion des clefs

Les services de protection offerts par IPsec s'appuient sur des algorithmes cryptographiques, et reposent donc sur des clefs. Si celles-ci sont gérables manuellement dans le cas où peu d'hôtes seront amenés à engager des conversations IPsec, ceci devient un véritable cauchemar quand le système commence à prendre un peu d'extension (le nombre de couples de clefs augmentant de manière quadratique avec le nombre de systèmes). C'est pourquoi la spécification IPsec propose également des procédés d'échange automatique de clefs, dont les aspects les plus importants sont évoqués ici. Un point important est que la gestion de clefs automatique est considérée comme plus sûre que la gestion manuelle.

Le protocole IKE permet l'échange de clefs entre deux hôtes d'adresses IP connues préalablement ou inconnues selon le mode d'échange de clefs sélectionné. Parmi ses avantages figure le mode agressif (cette caractéristique n'est pas obligatoire), qui permet d'accélérer la négociation, au prix de la protection d'identité. L'identité est toutefois protégée dans le cas d'une négociation IKE authentifiée à l'aide de signatures à clef publique.

Deux manières d'échanger des clefs sont abondamment utilisées : les clefs pré-partagées, et les certificats X.509 (dans ce dernier cas, deux systèmes d'adresses initialement inconnues pourront protéger leurs échanges).

La seconde manière de procéder a l'avantage de permettre à des clients d'adresses IP dynamiques et changeant éventuellement de créer des SAs.

Enfin, une définition qui peut s'avérer utile lors de la lecture de documentations techniques sur IPsec : la PFS *(Perfect Forward Secrecy)* est définie comme la notion selon laquelle la compromission d'une clef ne permettra l'accès qu'aux données protégées par cette clef, mais ne sera pas suffisante pour déchiffrer tout l'échange IPsec, seule la partie de la communication protégée par la clef corrompue sera déchiffrable.

IV.4 Les intrusions

IV.4.1 Port Scan

Il existe de nombreuses manières de scanner des ports. Mais le principe général est d'envoyer un paquet (TCP, IP, ...) et de regarder ce qu'il se passe. Selon la réaction, on pourra déterminer si le port est ouvert, fermé, ou filtré.

Selon cette réponse, le pirate pourra savoir quels protocoles sont utilisés ou les services auxquels il a accès. A la suite d'un port scan, le pirate aura donc plusieurs renseignements :

- l'adresse IP ayant répondu
- les services accessibles
- l'état des ports
- la liste des protocoles supportés par la machine (TCP, IP, SMTP *(Simple Mail Transfer Protocol)*, Telnet, ...)

La plupart du temps, cette attaque est un prémice à une autre attaque1, visant à s'introduire sur la machine.

Les attaques de ce type sont souvent facilement détectables. Une fois découverte, on peut bannir l'adresse source ordonnant le scan.

IV.4.2 OS Fingerprinting

Chaque OS possède sa propre gestion des protocoles réseaux. Il faut savoir que certains champs des paquets de données sont laissées à l'OS, et ce dernier doit décider ce qu'il y place. D'autre part, ces mêmes OS ne respectent pas toujours les documents RFC définissant les points importants. En conséquence, il existe des bases de données référençant ces caractéristiques propres à chaque OS.

En récupérant les paquets qu'émet la machine ciblée, on peut en extraire certains champs tels que le TTL (durée de vie du paquet) ou le ToS (permettant gérer la manière dont on traite la paquet, en spécifiant sa priorité, son importance,...).

Selon les OS, tous ces paramètres changent. La base de données contenant leur valeur par défaut permet alors de les identifier. Il suffit ainsi d'envoyer certains paquets différents à la machine pour tester les réponses et ensuite comparer ces dernières à une base de signatures pour identifier l'OS.

IV.4.3 DoS

Le but de l'attaque DoS, pour *(Denial of Service)* ou encore Déni de Service, est de saturer une machine en générant un gros trafic lui étant destiné. La conséquence de cette saturation pourra être la suppression d'un accès à un service, ou plus grave, la destruction de serveurs ou de sous-réseaux.

Les attaques par déni de service consistent à envoyer des paquets IP de taille ou de constitution inhabituelle, causant la saturation ou une mauvaise gestion de la part de la machine victime, qui ne peut plus assurer les services réseaux qu'elle propose. Cette attaque ne profite pas d'une faille du système d'exploitation, mais d'une faille de l'architecture TCP/IP.

On parlera de DDoS *(Distributed Denial of Service)* lorsque plusieurs machines sont à l'origine de l'attaque. Pour obtenir ces machines secondaires (on parle aussi de zombies), le pirate aura dû en prendre contrôle précédemment, en ayant acquis des droits administrateurs sur ces machines. Une fois qu'il les aura toutes en sa possession, il pourra lancer une attaque commune vers une machine cible.

La détection de ce type d'attaque est problématique, et reste un des grands enjeux des firewalls actuels.

IV.4.4 IP Spoofing

Cette attaque consiste à modifier les paquets IP dans le but d'usurper l'identité d'une machine, et de passer inaperçu aux yeux d'un firewall qui les considérera alors comme provenant d'une machine "de confiance".

Les nouveaux firewalls utilisant les protocoles VPN (par exemple IPSec) contrent cette attaque. En effet, le chiffrement des paquet permet d'éviter cette usurpation.

IV.4.5 Ping of Death

Un ping a normalement une longueur maximale de 65535 octets. Cette attaque consiste à envoyer un ping de taille supérieure. Lors de l'envoi, le ping sera fragmenté, et à sa réception par la machine sible, cette dernière devra réunir les morceaux du paquet. Il arrive que certains systèmes (anciens) ne gèrent pas correctement cette réunification du paquet. Dans ce cas, le système plantera.

IV.4.6 Ping Flooding

Comme la majorité des techniques de flooding, cette attaque consiste à envoyer un nombre très important de requêtes Ping sur une machine cible. Il peut y avoir une ou plusieurs machines émettrices (en cas de DDoS par exemple).

Cette attaque, comme la plupart de celles présentées rapidement ici, causera un ralentissement de la machine pouvant aller jusqu'au crash système. La seule façon de s'en protéger est d'utiliser un firewall correctement configuré.

IV.4.7 Autres types d'attaques

Il peut y avoir l'attaque par tunneling, les buffer overflow, les conditions de course, les DNS Poisoning, et beaucoup d'autres attaques.

Conclusion

Pour conclure, il faut vraiment savoir sécuriser intelligemment, c'est-à-dire bien étudier son emplacement et le niveau de sécurité,

- Ce qu'il faut sécuriser ?
- Comment le sécuriser ?

1. Assurer un bon niveau de sécurité, c'est,
   - Ne pas surprotéger ce qui n'en vaut pas/plus la peine
   - Protéger efficacement ce qui en vaut la peine
2. Placer la sécurité au bon endroit.
   - Ne jamais sous-estimer le facteur humain.

De plus,

- Rester attentif aux nouvelles technologies, aux mises à jour de sécurité des logiciels, etc.
- Veiller à la maintenance des bases de données du système informatique (BD *(BitDefender)* antivirus, sécurité du DBMS *(DataBase Management System)* utilisé, etc.)
- Ne pas se reposer sur une sécurité jugée bonne à un moment donné.

De ce fait, la sécurité est nécessaire à tous les niveaux d'utilisation de l'information, de sa source, pendant sa transmission et jusqu'à sa destruction. Et encore, de toute évidence le risque zéro n'existe pas.

"Un ordinateur en sécurité est un ordinateur éteint. Et encore..."

Bill Gates.

Annexe A

Annexe

A.1 Structures algébriques

L'élaboration des codes par blocs nécessite de pouvoir faire des opérations et des calculs sur les blocs. Par exemple, l'opération $\oplus$ sur un bloc de bits est l'addition modulo 2 de deux vecteurs de bits. D'autre part, les fonctions de chiffrement doivent être inversibles, ce qui nécessitera des structures où l'on peut facilement calculer l'inverse d'un bloc. Pour pouvoir faire ces calculs sur des bases algébriques solides, rappelons les structures fondamentales.

A.1.1 Groupes

Définition A.1.1.1 *Un groupe $(G, *)$ est un ensemble muni d'un opérateur binaire interne vérifiant les propriétés suivantes :*

1. *$*$ est associative : pour tous a, b, $c \in G$, $a * (b * c) = (a * b) * c$.*
2. *Il existe un élément neutre $e \in G$, tel que pour tout $a \in G$, on trouve $a * e = e * a = a$.*
3. *Tout élément a un inverse : pour tout $a \in G$, il existe $a^{-1} \in G$ tel que $a * a^{-1} = a^{-1} * a = e$.*

De plus, si la loi est commutative ($a*b = b*a$ pour tous $a, b \in G$), alors G est dit abélien.

On dit qu'un sous-ensemble H de G est un sous-groupe de G lorsque les restrictions des opérations de G confèrent à H une structure de groupe. Pour un élément a d'un groupe G, on note an la répétition de la loi $*$, $a * ... * a$, portant sur n termes égaux à a pour tout $n \in N^*$.
Si un élément $g \in G$ est tel que pour tout $a \in G$, il existe $i \in \mathbb{Z}$, tel que $a = g^i$, alors g est un générateur du groupe $(G, *)$ ou encore une *racine primitive*.

Définition A.1.1.2 *Un groupe est dit cyclique s'il possède un générateur. Autrement dit, si $\exists g \in G$, $\forall a \in G$, $\exists i \in \mathbb{Z}$, $a = g^i$*

Exemple : *Par exemple, pour un entier n fixé, l'ensemble des entiers $\mathbb{Z}_n = \{0, 1, ..., n-1\}$, muni de la loi d'addition modulo n est un groupe cyclique généré par 1 ; si $n = 7$, et si on choisit pour loi de composition la multiplication modulo 7, l'ensemble $\{1, ..., 6\}$ est un groupe cyclique généré par 3, car $1 = 3^0, 2 = 3^2 = 9, 3 = 3^1, 4 = 3^4, 5 = 3^5, 6 = 3^3$.*

Soit un groupe $(G, \in)$ et $a \in G$. L'ensemble $\{a_i, i \in N\}$ est un sous-groupe de G, noté $< a >$ ou G_a. Si ce sous-groupe est fini, son cardinal est l'ordre de a. Si G est fini, le cardinal de tout sous-groupe de G divise le cardinal de G.

Propriété A.1.1 (Lagrange) *Dans un groupe fini abélien $(G, *, e)$ de cardinal n, pour tout $x \in G$: $x_n = e$.*

Preuve. *Soit a un élément du groupe G, alors a est inversible. Donc, l'application $f_a : x \to a \times x$ définie de G dans G est une bijection. Donc $Im(f_a) = G$; d'où $\prod_{y \in Im(f_a)} y = \prod_{x \in G} x$. Or $\prod_{y \in Im(f_a)} y = \prod_{z \in G} a \times x = a^n \prod_{x \in G} x$ (commutativité de x). Ainsi $a^n \prod_{x \in G} x = \prod_{x \in G} x$, d'où $a^n = e$*

A.1.2 Anneaux

Définition A.1.2.1 *Un anneau $(A, +, \times)$ est un ensemble muni de deux opérateurs binaires internes vérifiant les propriétés suivantes :*

1. *$(A, +)$ est un groupe abélien.*
2. *$\times$ est associative : pour tous $a, b, c \in A, a \times (b \times c) = (a \times b) \times c$.*
3. *$\times$ est distributive sur $+$: pour tous $a, b, c \in A, a \times (b + c) = (a \times b) + (a \times c)$ et $(b + c) \times a = (b \times a) + (c \times a)$.*

Si de plus $\times$ possède un élément neutre dans A, A est dit *unitaire*. Si de plus $\times$ est commutative, A est dit *commutatif*. Tous les éléments de A ont un *opposé*, c'est leur inverse pour la loi $+$. Ils n'ont cependant pas forcément un inverse pour la loi $\times$. *L'ensemble des inversibles* pour la loi $\times$ est souvent noté A^*.
Pour un élément a d'un anneau A, on note $n \cdot a$ (ou plus simplement na) la somme $a + ... + a$ portant sur n termes égaux à a pour tout $n \in N^*$.
Si l'ensemble $\{k \in N^* : k \cdot 1 = 0\}$ n'est pas vide, le plus petit élément de cet ensemble est appelé la *caractéristique* de l'anneau.
Dans le cas contraire, on dit que l'anneau est de caractéristique 0.

Par exemple, $(Z, +, \times)$ est un anneau unitaire commutatif de caractéristique 0.

Définition A.1.2.2 *Deux anneaux $(A, +_A, \times_A)$ et $(B, +_B, \times_B)$ sont isomorphes lorsqu'il existe une bijection $f : A \to B$ vérifiant pour tous x et y dans A :*

$$f(x +_A y) = f(x) +_B f(y) \text{ et } f(x \times_A y) = f(x) \times_B f(y). \tag{A.1}$$

Si E est un ensemble quelconque et $(A, +, \times)$ un anneau tel qu'il existe une bijection f de E sur A, alors E peut être muni d'une structure d'anneau :

$$x +_E y = f^{-1}(f(x) + f(y)) \text{ et } x \times_E y = f^{-1}(f(x) \times f(y)). \tag{A.2}$$

L'anneau $(E, +_E, \times_E)$ ainsi défini est évidemment isomorphe à A. Lorsque deux anneaux sont isomorphes, on peut identifier l'un à l'autre.

Un anneau commutatif est *intègre*, s'il ne possède pas de diviseur de zéro, autrement dit si pour deux éléments x et y vérifiant $xy = 0$ alors forcément l'un deux au moins est nul. Un idéal I est un sous-groupe d'un anneau A pour la loi $+$ qui est *absorbant* pour la loi $\times$: pour $g \in I$, le produit $a \times g$ reste dans I pour n'importe quel élément a de l'anneau A.
Pour tout $x \in A$ la partie $Ax = ax; a \in A$ est un idéal de A appelé *idéal engendré* par x. Un idéal I de A est dit *principal* s'il existe un générateur x (tel que $I = Ax$). Un anneau est *principal* si et seulement si tout idéal y est principal.

A.1.3 Corps

Définition A.1.3.1 *Un corps $(A, +, \times)$ est un ensemble muni de deux opérateurs binaires internes vérifiant les propriétés suivantes :*

1. *$(A, +, \times)$ est un anneau unitaire.*
2. *$(A \setminus \{0\}, \times)$ est un groupe.*

Si $(A, +, \times)$ est commutatif, c'est donc un corps commutatif ; l'inverse (ou opposé) de x par la loi $+$ est noté $-x$; l'inverse de x par la loi $\times$ est noté x^{-1}. La caractéristique d'un corps est sa caractéristique en tant qu'anneau.

Exemple : *$(Q,+,\times)$ est un corps commutatif de caractéristique 0. Puisque tous les anneaux et corps qui nous intéressent dans ce cours sont commutatifs, nous dirons désormais anneau (respectivement corps) pour désigner un anneau unitaire commutatif (respectivement un corps commutatif). Deux corps sont isomorphes lorsqu'ils sont isomorphes en tant qu'anneaux. On dit qu'un sous ensemble W d'un corps V est un sous−corps de V lorsque les restrictions des opérations de V à W confèrent à W une structure de corps.*

A.1.4 Espace vectoriel

Définition A.1.4.1 *Un ensemble $\mathbb{E}$ est un espace vectoriel sur un corps V s'il est muni d'une loi de composition interne $+$ et d'une loi externe $(\cdot)$, telles que :*

1. *$(\mathbb{E}, +)$ est un groupe commutatif.*
2. *Pour tout $u \in \mathbb{E}$, alors $1_V \cdot u = u$.*
3. *Pour tout $\lambda, \mu \in V$ et $u \in \mathbb{E}$, alors $\lambda \cdot u + \mu \cdot u = (\lambda + \mu) \cdot u$.*
4. *Pour tout $\lambda, \mu \in V$ et $u \in \mathbb{E}$, alors $\lambda \cdot (\mu \cdot u) = (\lambda \cdot \mu) \cdot u$.*
5. *Pour tout $\lambda, \mu \in V$ et $u \in \mathbb{E}$, alors $\lambda \cdot (u + v) = \lambda \cdot u + \lambda \cdot v$.*

Un élément d'un espace vectoriel est appelé un *vecteur*, et les éléments du corps V sont des scalaires.

L'ensemble $\{ 0, 1 \}$ muni des opérations d'addition et de multiplication est un corps commutatif noté $\mathbb{F}_2$. L'ensemble des tableaux de bits de taille n peut donc être muni d'une structure d'espace vectoriel. L'ensemble des mots de code est alors $\mathbb{F}_{2^n}$.
Selon la structure choisie, on peut manipuler les mots de code par des additions et des multiplications, entières ou vectorielles. Toutes ces structures sont trés générales, et classiques en algèbre. Une particularité des codes est qu'ils constituent des ensembles finis. Les groupes et corps finis ont des propriétés additionnelles que nous utiliserons intensément au fils de ce cours.
On note $\mathbb{Z}$ l'ensemble des entiers, $\mathbb{Q}$ le corps des rationnels, $\mathbb{R}$ le corps des réels, $\mathbb{N}$ l'ensemble

des entiers positifs ou nuls, $\mathbb{Z}_n$ l'ensemble des entiers positifs ou nuls et strictement plus petits que n, pour $n \in \mathbb{N}\ \{0,1\}$..
L'ensemble $\mathbb{Z}_n$ muni de l'addition et de la multiplication modulo n est un anneau noté $\mathbb{Z}/n\mathbb{Z}$, qui sera trés utilisé en théorie des codes. On dit qu'un anneau est euclidien s'il est muni de la division euclidienne, c'est-à-dire que pour tout couple d'éléments a et b , $(a \geq b)$ de cet ensemble, il existe q et r tels que $a = bq + r$ et $\mid r \mid < \mid b \mid$.
Les nombres q et r sont respectivement le quotient et le reste de la division euclidienne, et notés $q = a\ div\ b$ et $r = a\ mod\ b$ (pour *a modulo b*). Or tout anneau euclidien est principal. Ceci implique l'existence d'un *plus grand commun diviseur* (*pgcd*) pour tout couple d'éléments (a, b). Ce *pgcd* est le générateur de l'idéal $Aa + Ab$.

Si p est un nombre premier, l'anneau $\mathbb{Z}/p\mathbb{Z}$ est un corps de caractéristique p. En effet, le théoréme classique de Bézout **(voir A.2.3.1)** nous apprend que quels que soient deux entiers a et b, il existe des entiers x et y tels que

$$ax + by = pgcd(a, b).$$

Si p est premier et a est un élément non nul de $\mathbb{Z}_p$, cette identité appliquée à a et p donne $ax + bp = 1$, soit $ax = 1\ mod\ p$, donc a est inversible et x est son inverse. On note $\mathbb{F}_p$ ce corps. Le corps des nombres rationnels $\mathbb{Q}$ et les corps $\mathbb{F}_p$, sont appelés **corps premiers**.

A.2 Elément d'arithmétique dans $\mathbb{Z}$

A.2.1 Plus Grand Commun Diviseur (pgcd).

Le plus grand commun diviseur *pgcd* en français et *gcd* (Greatest Common Divisor) en anglais est défini de la manière suivante

Théorème A.2.1.1 *Le plus grand commun diviseur de deux nombres entiers a et b est le plus grand des entiers qui divisent à la fois a et b, on le note pgcd(a, b) ou $a \wedge b$ ou encore (a, b). Si le pgcd de a et b vaut 1, on dira que a et b sont premiers entre eux ou sont des* ***nombres relativement premiers****.*

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

Remarque. *Attention bien distinguer entre nombres relativement premiers et nombres qui ne se divisent pas.*

Exemple : *35 et 49 ne se divisent pas car* $49 = 35 \cdot 1 + 14$ *et* $35 = 0 \times 49 + 35$*, mais ils ne sont pas premiers entre eux car 7 divise à la fois 35 et 49.*

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

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

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

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

Exemple : *Le pgcd de* $4200 = 2^3.3.5^2.7$ *et de* $10780 = 2^2.5.7^2.11$ *est*

$$pgcd(4200, 10780) = 4200 \wedge 10780 = 2^2.5.7 = 140$$

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

A.2.2 Algorithme d'Euclide

La version dite <étendue> de l'algorithme d'Euclide, celle que nous emploierons trés souvent dans ce cours, permet, en plus du calcul du pgcd de deux nombres, de trouver les coefficients de Bézout.

Il est étendu aussi parce qu'on veut le rendre un peu plus générique en lui donnant la possibilité de l'appliquer non seulement à des ensembles de nombres mais aussi à n'importe quel anneau euclidien. Ce sera le cas des polynômes, comme nous le verrons dans les sections suivantes. Le principe de l'algorithme est d'itérer la fonction G suivante :

$$G : \begin{bmatrix} a \\ b \end{bmatrix} \mapsto \begin{bmatrix} 0 & 1 \\ 1 & -(a \quad div \quad b) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}. \tag{A.3}$$

Exemple : *Pour trouver* x *et* y *tel que* $x \times 522 + y \times 453 = pgcd(522, 453)$*, on écrit les matrices correspondant à l'itération avec la fonction* G*, on a :*

$$\begin{aligned} \begin{bmatrix} 3 \\ 0 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & -6 \end{bmatrix} \\ &\quad \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 522 \\ 453 \end{bmatrix} \\ &= \begin{bmatrix} 46 & -53 \\ -151 & 174 \end{bmatrix} \begin{bmatrix} 522 \\ 453 \end{bmatrix} \end{aligned} \tag{A.4}$$

On obtient ainsi $-151522 + 174453 = 3$. D'où $d = 3$, $x = -151$ et $y = 174$. On donne une version de l'algorithme d'Euclide étendu qui réalise ce calcul en ne stockant que la première ligne de G. Il affecte les variables x, y et d de façon à vérifier en sortie : $d = pgcd(a, b)$ et $ax + by = d$. Pour une résolution (à la main), on pourra calculer récursivement les équations (E_i) suivantes (en s'arrêtant lorsque $ri + 1 = 0$) :

$$\begin{aligned} (E_0) &: \quad 1 \times a + 0 \times b = a \\ (E_1) &: \quad 0 \times a + 1 \times b = b \\ (E_{i+1}) = (E_{i-1}) - q_i(E_i) &: u_i \times a + v_i \times b = r_i \end{aligned} \tag{A.5}$$

A.2.3 Théorème de Bézout

Théorème A.2.3.1 *Soient a et b deux entiers relatifs non nuls et d leur pgcd.*

Il existe deux entiers relatifs x et y tels que $|x| < |b|$ et $|y| < |a|$ vérifiant l'égalité de Bézout $ax + by = d$ et l'algorithme d'Euclide étendu est correct.

Preuve. *Tout d'abord, montrons que la suite des restes est toujours divisible par $d = pgcd(a, b)$: par récurrence si $r_{j-2} = kd$ et $r_{j-1} = hd$ alors $r_j = r_{j-2} - q_j \times r_{j-1} = d(k - q_j h)$ et donc $pgcd(a, b) = pgcd(r_{j-1}, r_j)$.*

Ensuite, la suite des restes positifs r_j est donc strictement décroissante et minorée par 0 donc converge. Cela prouve que l'algorithme est arrivé à sa fin.

En outre, la suite converge forcément vers 0, puisque sinon on peut toujours continuer à diviser le reste. Il existe donc un indice i tel que $r_{i-1} = q_{i+1} r_i + 0$. Dans ce cas, $pgcd(r_{i-1}, r_i) = r_i$ et la remarque précédente indique donc que r_i est bien le pgcd recherché.

Il reste à montrer que les coefficients conviennent. Nous procédons par récurrence. Clairement le cas initial $a \bmod b = 0$ convient et l'algorithme est correct dans ce cas.

*Ensuite, notons $r = a \bmod b$ et $q = a \operatorname{div} b$, alors $a = bq + r$. Par récurrence, avec les notations internes de l'algorithme **(A.2.2)**, on a que $d = xb + yr$ avec $|x| < b$ et $|y| < r$.*

Cette relation induit immédiatement que $d = ya + (x - qy)b$ avec $|y| < r < b$ et $|x - qy| \leq |x| + q|y| < b + qr = a$. Ce qui prouve que l'algorithme est correct.

A.2.4 Théorème de Fermat

Soit $n \geq 2$ un entier, on note $\mathbb{Z}_n^*$ l'ensemble des entiers positifs plus petits que n :

$$\mathbb{Z}_n^* = \{x \in \mathbb{N} : 1 \leq x < n \quad et \quad pgcd(x, n) = 1\}$$

Le cardinal de $\mathbb{Z}_n^*$ est noté $\phi(n)$. La fonction ϕ est appelée fonction d'Euler. Par exemple, si p est premier, $\varphi(p) = p - 1$. De la même manière, dans $\mathbb{Z}/p^k\mathbb{Z}$, seuls les multiples de p ne sont pas premiers avec p. Ils sont $(p, 2p, 3p, ..., p^{k-1}p)$. Ainsi, $\varphi(p^k) = p^k - p^{k-1} = (p-1)p^{k-1}$. Enfin, si m et n sont premeirs entre eux, les entiers premeirs avec, $m \cdot n$, sont chacun des entiers premiers avec m multipliés par chacun des entiers premiers avec n et seulement ceux là ; d'où l'on tire $\varphi(mn) = \varphi(m)\varphi(n)$. Au final, on obtient la proposition suivante :

Proposition *Soient $p_1, ..., p_n$ des entiers premiers alors,*

$$\forall (k_1, ..., k_n) \in \mathbb{N}, \varphi(\prod p_i^{k_i}) = \prod (p_i - 1) p_i^{k_i - 1}$$

Ainsi, pour calculer $\varphi(n)$, il faut savoir factoriser n.

Dans $\mathbb{Z}_n^*$, tout élément x a un inverse : en effet, comme x est un premeir avec n, l'identité de Bézout assure l'existence de deux entiers de signes opposés u et v ($1 \leq |u| < n \quad et \quad 1 \leq |v| < x$), tels que :

$$u \cdot x + v \cdot n = 1$$

On a alors $u \cdot x \equiv 1 \ mod \ n$, i.e $u = x^{-1} \ mod \ n$. On appelle *u inverse de x modulo n*. L'existence de u montre que $\mathbb{Z}_n^*$, muni de loi de multiplication modulo n, est un groupe commutatif fini. Le calcul de u se fait grâce à l'algorithme d'Euclide étendu.

Théorème A.2.4.1 (Théorème d'Euleur) *Soit a un élément quelconque de* $\mathbb{Z}_n^*$.
On a : $a^{\varphi(n)} \equiv 1 \ mod \ n$

Théorème A.2.4.2 (Théorème de Fermat) *Si p est premeir, alors pour tout* $a \in \mathbb{Z}_p$: $a^p \equiv a \ mod \ p$

A.2.5 Les congruences

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

Définition A.2.5.1 *Soit a, b et m trois entiers. On dit que a est congru b modulo m et on écrit*

$$a \equiv b \, mod \, m$$

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

Une autre manière de dire la même chose,

Corollaire. A.2.1 *Soit m un entier non nul, alors a et b sont congrus modulo m si et seulement si les restes de la division euclidienne de a par m et de b par m sont les mêmes.*

Exemple : *Prenons m = 2 alors deux entiers a et b sont congrus modulo 2 s'il sont soit tous les deux pairs soit tous les deux impairs.*

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

Proposition *La relation de congruence est une relation d'´equivalence sur les entiers.*
c'est à dire que,

- $a \equiv a \, mod \, m$ *(réflexivité)*
- $a \equiv b \, mod \, m \Rightarrow b \equiv a \, mod m$ *(symétrie)*
- $(a \equiv b \, mob \, m \quad et \quad b \equiv c \, mod \, m) \Rightarrow a \equiv c \, mod \, m$ *(transtivité)*

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

$$((a \, (mod \, m)) + (b \, (mod \, m) \quad (mod \, m) = (a + b \quad (mod \, m))$$
$$((a \, (mod \, m)) \times (b \, (mod \, m) \quad (mod \, m) = (a \times b \quad (mod \, m))$$

A.2.6 Problème du logarithme discret

Si $g \in \mathbb{Z}_n^*$ est d'ordre $\varphi(n)$, alors $\mathbb{Z}_n^* = \{g^i \,(mod\, n) / \, i \in \mathbb{N}\}$. On dit que g est un *générateur* ou *racine primitive* de $\mathbb{Z}_n^*$. Un groupe qui admet un élément générateur est dit *cyclique*. Les valeurs de n pour lesquelles $\mathbb{Z}_n^*$ admet un élément générateur (i.e. $\mathbb{Z}_n^*$ est cyclique) sont 2, 4, p^e et $2p^e$ pour tout nombre premier $p \geq 3$ et tout entier positif e.
Si g est un générateur de $\mathbb{Z}_n^*$, alors, pour tout $a \in \mathbb{Z}_n^*$, $\exists z \in \mathbb{N}$ tel que $g^z = a$ modulo n.

Définition. *Pour a dans $\mathbb{Z}_n^*$, le plus petit entier positif z tel que $g^z = a$ modulo n est appelé le logarithme discret (ou encore l'index) en base g de a modulo n ; on note $z = log_g a \, modulo \, n$.*

Le résultat suivant est connu sous le nom de théorème du logarithme discret.

Théorème. (Logarithme discret) *Si g est une racine premitive de $\mathbb{Z}_n^*$, alors $\forall x, y \in \mathbb{N}$:*

$$g^x \equiv g^y \, mod \, n \Leftrightarrow x \equiv y \, mod \, \varphi(n)$$

Bibliographie

[1] Christoff Paar et Jan Pelzl. *Kryptografie verständlich*. Springer, 2016.

[2] François Arnault. "Contributions en Mathématiques Discrètes & Applications Cryptographiques". In : *Université de Limoges, Faculté des Sciences & Techniques* (2014).

[3] Jean-Guillaume Dumas et al. "Théorie des codes". In : *Dunod, Paris* (2007).

[4] Renaud Dumont. "Cryptographie et Sécurité informatique". In : *Eyrolles* 2010 (2009).

[5] Frédéric RAYNAL. "Canaux cachés". In : *Techniques de l'ingénieur. Sécurité des systèmes d'information* TI440 (2003), p. 42313-1.

[6] Thomas NOËL. "Ip mobile". In : *Techniques de l'ingénieur. Sécurité des systèmes d'information* TE7515 (2002), TE7515-1.

[7] Daniel TREZENTOS. "Standard pour réseaux sans fil : IEEE 802.11". In : *Techniques de l'ingénieur. Sécurité des systèmes d'information* TE7375 (2002), TE7375-1.

[8] Claude CHIARAMONTI. "Échange de données informatisé. (EDI)". In : *Techniques de l'ingénieur. Sécurité des systèmes d'information* H3598 (2001), H3598-1.

[9] Douglas Stinson et al. *Cryptographie-Théorie et pratique, 2ème édition*. Vuibert, 2003.

[10] Gilles Dubertret et Delphine Courtois. *L'univers secret de la cryptographie*. Vuibert, 2015.

[11] Serge Vaudenay. *A classical introduction to cryptography : Applications for communications security*. Springer Science & Business Media, 2006.

[12] Damien Vergnaud. *Exercices et problèmes de cryptographie 3e éd.* Dunod, 2018.

[13] Bruno Martin. *Codage, cryptologie et applications*. PPUR presses polytechniques, 2004.

[14] Neal Koblitz. *A course in number theory and cryptography*. T. 114. Springer Science & Business Media, 1994.

[15] Joan Daemen et Vincent Rijmen. *The design of Rijndael*. T. 2. Springer, 2002.

[16] Lawrence C. Washington. *Elliptic curves : number theory and cryptography*. CRC press, 2008.

[17] Jean-Guillaume Dumas et al. *Théorie des codes 2e éd. : Compression, cryptage, correction*. Dunod, 2013.

[18] Didier Müller. *Les codes secrets décryptés*. City éd., 2011.

[19] Auguste Kerckhoffs. "La cryptographic militaire". In : *Journal des sciences militaires* (1883), p. 5-38.

[20] Joan Daemen et Vincent Rijmen. “AES proposal : Rijndael”. In : (1999).

[21] Federal Information Processing Standards Publication. “Data encryption standard”. In : (1999).

[22] Claude E Shannon. “A mathematical theory of communication”. In : *The Bell system technical journal* 27.3 (1948), p. 379-423.

[23] Whitfield Diffie et Martin Hellman. “New directions in cryptography”. In : *IEEE transactions on Information Theory* 22.6 (1976), p. 644-654.