Université de Liège
Faculté des Sciences Appliquées

Cryptographie et Sécurité informatique

INFO0045-2

Notes de cours provisoires

2009 - 2010

Renaud Dumont

Table des matières

Remerciements

Vifs remerciements à Messieurs W. Stallings [1] (algorithmes cryptographiques modernes) et D. Müller [2] (histoire de la cryptographie et stéganographie) pour la permission de copie sur leurs illustrations.

Dans la mesure du possible, les sources des autres illustrations ont été mentionnées. Certaines d'entre elles me sont inconnues. Si vous en connaissez l'origine ou en êtes vous-même l'auteur, n'hésitez pas à me le signaler.

Remarque générale

Ce document s'accompagne d'un cours ex-cathedra. Ceci explique pourquoi certains passages peuvent paraitre succincts.

Chapitre 1

Introduction

1.1 Motivation

A l'heure actuelle, les besoins en matière de sécurité sont grandissants, et la tendance n'est certainement pas à la baisse. Mais pourquoi ?

Tout d'abord parce que le matériel informatique est omniprésent. En effet, d'une part le matériel est accessible à un prix très abordable, et d'autre part, les logiciels tendent à se simplifier (au niveau de l'utilisation !) et permettent une prise en main rapide.

D'un autre coté, les entreprises, elles aussi informatisées, nécessitent un réseau sécurisé pour le transfert des données, que ce soit entre les machines de cette entreprise, ou avec des machines externes, distantes de plusieurs milliers de kilomètres.

Si on observe la sécurité d'une manière plus générale, elle est d'ailleurs présente à plusieurs niveaux, qu'il s'agisse des différentes portées de l'information comme l'illustre la figure 1.1 ou les stades de vie de l'information, tels qu'illustrés à la figure 1.2.

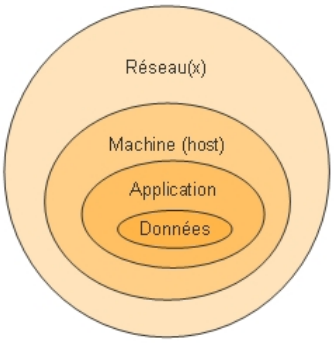

FIG. 1.1 – La sécurité existe à plusieurs niveaux (portée de l'information)

On assiste également à une évolution constante des techniques, qu'il s'agisse des techniques visant à sécuriser l'échange de ces données ou des techniques mises au point pour contourner ces systèmes sécurisés.

D'une manière générale, la sécurité des données tend à s'améliorer. La raison principale est qu'aujourd'hui, l'étude des contournements possibles est simultanée à l'étude des protections. La tendance actuelle veut que les résultats découverts, tous domaines confondus, soient publiés. Dans le cadre de la sécurité informatique, cela permet de découvrir plus rapidement les failles et/ou avantages de certaines techniques.

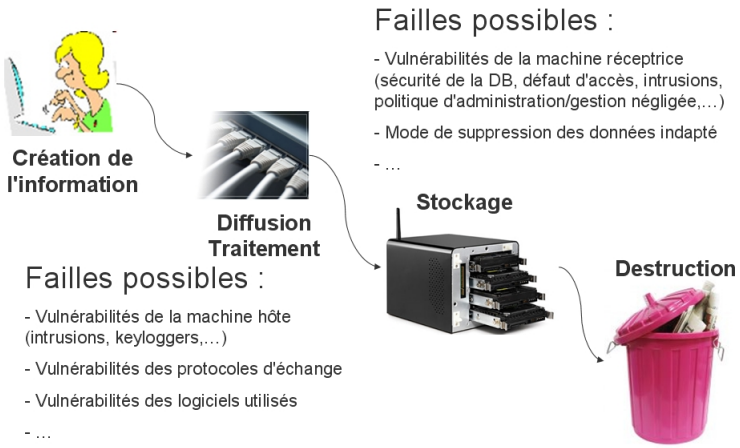

FIG. 1.2 – Stades de vie de l'information

A l'origine, c'est Shannon qui, en 1948 puis en 1949 avec Weaver, a le premier défini les base d'une transmission de données entre deux parties. Son idée est illustrée à la figure 1.3.

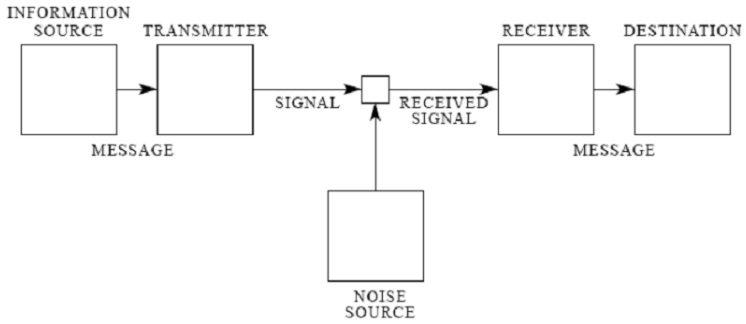

FIG. 1.3 – La transmission de données vue par Shannon-Weaver en 1949

Pour plus de facilités dans le cadre de ce cours de sécurité, nous simplifierons ce schéma en trois entités, telles que représentées à la figure 1.4

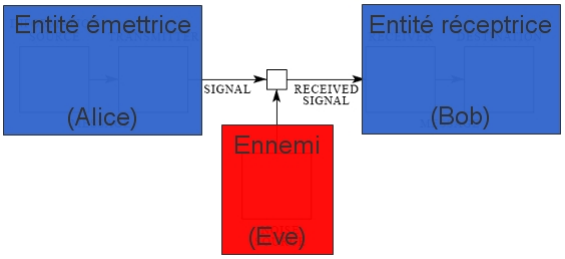

FIG. 1.4 – La transmission de données simplifiée

Ajoutons que les signaux émis et reçu seront ici l'envoi et la réception de messages, mais nous préciserons la terminologie employée par la suite.

Avec la popularité grandissante des réseaux, des échanges de données, et donc des transmissions entre individus, de nombreuses menaces émergèrent. Parmi celles-ci, on trouve diverses catégories :
- Les menaces accidentelles
- Les menaces intentionnelles :
  - passives
  - actives

Les menaces accidentelles ne supposent aucune préméditation. Dans cette catégorie, sont repris les bugs logiciels, les pannes matérielles, et autres défaillances "incontrôlables".

Les menaces intentionnelles quant à elles, reposent sur l'action d'un tiers désirant s'introduire et relever des informations. Dans le cas d'une attaque passive, l'intrus va tenter de dérober les informations par audit, ce qui rend sa détection relativement difficile. En effet, cet audit ne modifie pas les fichiers, ni n'altère les systèmes. Dans le cas d'une attaque active, la détection est facilitée, mais il peut être déjà trop tard lorsque celle-ci a lieu. Ici, l'intrus aura volontairement modifié les fichiers ou le système en place pour s'en emparer.

Les menaces actives appartiennent principalement à quatre catégories (illustrées à la figure 1.5) :
- Interruption = problème lié à la disponibilité des données
- Interception = problème lié à la confidentialité des données
- Modification = problème lié à l'intégrité des données
- Fabrication = problème lié à l'authenticité des données

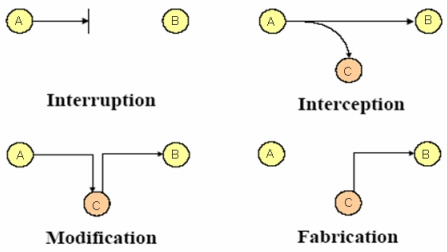

FIG. 1.5 – Types de menaces actives

Les auteurs de ces attaques sont notamment les hackers (agissant souvent par défi personnel), les concurrents industriels (vol d'informations concernant la stratégie de l'entreprise ou la conception de projets), les espions, la presse ou encore les agences nationales. Nous en reparlerons dans un prochain chapitre.

1.2 Différents modèles de sécurité

Tout en restant très générale, cette section présente par l'intermédiaire de quelques schémas la manière avec laquelle peut être appréhendée la notion de sécurité. Même si le concept n'est pas neuf, il a subi plusieurs modifications au cours du temps.

1.2.1 CIA (1987)

Le triangle CIA est le pilier immuable présentant les grands axes de la sécurité. La plupart des autres modèles utilisent cette représentation en tant que base.

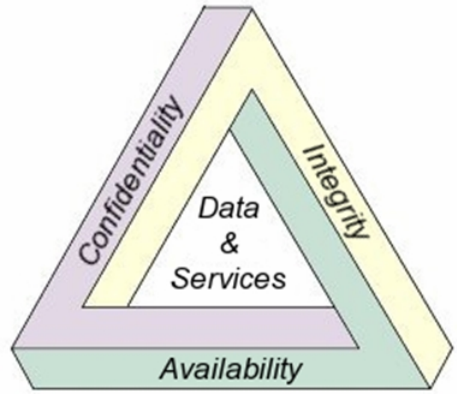

FIG. 1.6 – CIA

Le triangle opposé existe également. Il porte le nom de DAD, pour Disclosure, Alteration, Disruption.

On peut définir les différents termes employés comme suit :
- Confidentialité : l'information n'est connue que des entités communicantes
- Intégrité : l'information n'a pas été modifiée entre sa création et son traitement (en ce compris un éventuel transfert)
- Disponibilité : l'information est toujours accessible et ne peut être bloquée/perdue

1.2.2 Le contrôle d'accès : Le protocole AAA

On peut en effet se demander pourquoi la vérification de l'identité des parties communicantes n'est pas traitée dans ce premier modèle. Celui-ci est en réalité la base de tous les systèmes existants. Il fut complété au fil des années, mais reste historiquement la première modélisation de la sécurité d'un système.

Le contrôle d'accès se fait en 4 étapes :
1. Identification : Qui êtes-vous ?
2. Authentification : Prouvez-le !
3. Autorisation : Avez-vous les droits requis ?
4. Accounting/Audit : Qu'avez-vous fait ?

On parle du *protocole AAA* (les deux premières étapes sont fusionnées). Dans certaines situations, on scindera la dernière étape. On parlera d'Accounting lorsque le fait de comptabiliser des faits sera demandé, et d'Audit lorsque des résultats plus globaux devront être étudiés.

Notons également que l'authentification, visant à prouver l'identité d'un individu peut se faire de plusieurs manières :

- Ce que vous *savez* (mot de passe, code PIN, etc.)
- Ce que vous *avez* (carte magnétique, lecteur de carte, etc.)
- Ce que vous *êtes* (empreintes digitales, réseau rétinien, etc.)

L'**authentification forte** résultera de la combinaison de 2 de ces facteurs.

Le pentagone de confiance (2006)

Défini par Piscitello en 2006, il précise la notion d'accès à un système. Indépendamment des notions définies dans le triangle CIA, le modèle de Piscitello précise la confiance que peut/doit avoir l'utilisateur

en présence d'un système informatisé.

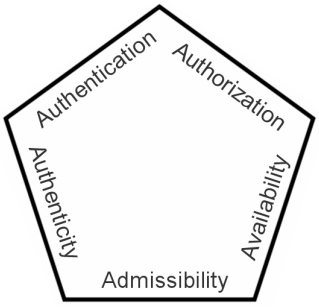

FIG. 1.7 – Le pentagone de confiance

On y retrouve les notions d'authentification, de disponibilité, d'autorisation et d'intégrité (Authenticity). Mais on y découvre un nouveau thème : l'admissibilité : La machine sur laquelle nous travaillons, à laquelle nous nous connectons, est-elle fiable ? En d'autres termes, peut-on faire confiance à la machine cible ?

1.2.3 Parkerian Hexad (2002)

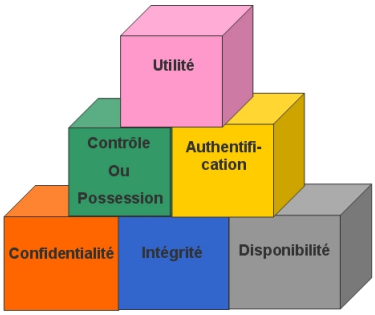

FIG. 1.8 – L'hexagone de Parker

Ce modèle, initié par Donn Parker, ajoute la nuance d'utilité (une information chiffrée pour laquelle on a perdu la clé de déchiffrement n'est plus d'aucune *utilité*, bien que l'utilisateur y ait accès, que cette information soit confidentielle, disponible et intègre).

1.2.4 McCumber Cube (1991)

On y retrouve les trois piliers de la sécurité (CIA), mais deux autres dimensions apparaissent :
- L'état des données : le stockage, la transmission, l'exécution
- Les méthodes : les principes et règles à adopter pour atteindre le niveau de sécurité souhaité

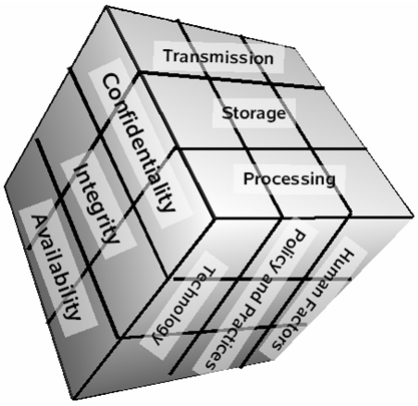

FIG. 1.9 – Le Cube de McCumber

1.2.5 Autres définitions

1.2.5.1 La sécurité en parallèle

On parle de sécurité en parallèle lorsque plusieurs mécanismes de sécurité protégeant un système possèdent le même rôle. Dans ce cas, le niveau de protection du système est équivalent à celui du mécanisme le moins sûr.

En tant qu'exemple, citons un ordinateur portable que l'on peut déverrouiller par mot de passe, dongle ou empreinte digitale.

1.2.5.2 La sécurité en série

Plusieurs mécanismes de sécurité protègent un système et ont des rôles différents. On parlera de « défense en profondeur ».

Citons par exemple le réseau d'une entreprise où :
- Le réseau est sécurisé par un FW hardware,
- Les liaisons entre machines sont protégées,
- Les machines individuelles sont munies d'un FW software,
- Les accès aux machines se font par empreinte biométrique,
- Le logiciel à utiliser est accessible par mot de passe,
- etc.

Chapitre 2

Introduction à la cryptographie

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.)

2.1 Vocabulaire de base

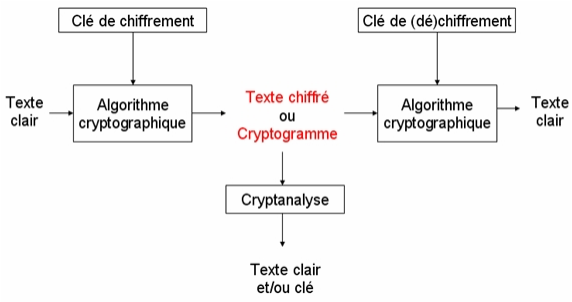

FIG. 2.1 – Protocole de chiffrement

Cryptologie : Il s'agit d'une science mathématique comportant deux branches : la cryptographie et la cryptanalyse

Cryptographie : La cryptographie est l'étude des méthodes donnant la possibilité d'envoyer des données de manière confidentielle sur un support donné.

Chiffrement : Le chiffrement consiste à transformer une donnée (texte, message, ...) afin de la rendre incompréhensible par une personne autre que celui qui a créé le message et celui qui en est le destinataire. La fonction permettant de retrouver le texte clair à partir du texte chiffré porte le nom de *déchiffrement*.

Texte chiffré : Appelé également *cryptogramme*, le texte chiffré est le résultat de l'application d'un chiffrement à un texte clair.

Clef : Il s'agit du paramètre impliqué et autorisant des opérations de chiffrement et/ou déchiffrement. Dans le cas d'un algorithme symétrique, la clef est identique lors des deux opérations. Dans le cas d'algorithmes asymétriques, elle diffère pour les deux opérations.

Cryptanalyse : Opposée à la cryptographie, elle a pour but de retrouver le texte clair à partir de textes chiffrés en déterminant les failles des algorithmes utilisés.

Cryptosystème : Il est défini comme l'ensemble des clés possibles (espace de clés), des textes clairs et chiffrés possibles associés à un algorithme donné.

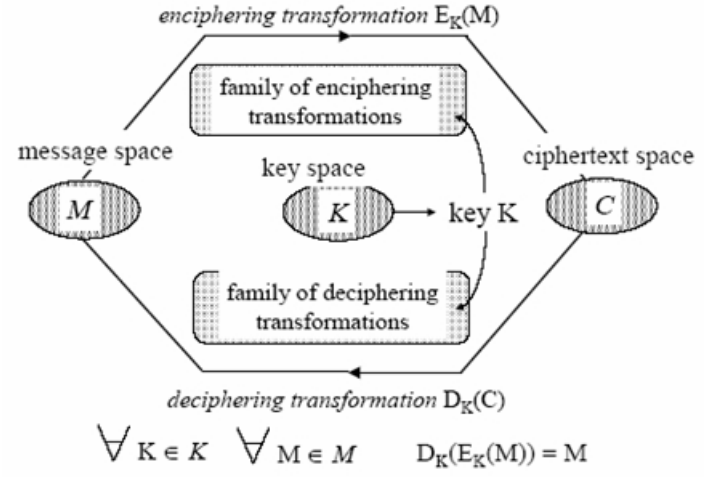

FIG. 2.2 – Schéma d'un cryptosystème

L'algorithme est en réalité un triplet d'algorithmes :
- l'un générant les clés K,
- un autre pour chiffrer M, et
- un troisième pour déchiffrer C.

Remarque : On parle de "décryptage" pour désigner l'action permettant de retrouver le texte clair sans connaître la clef de déchiffrement. On emploie également parfois les termes "cryptage" et "crypter" pour qualifier l'action de chiffrer un message. Les mots "encryptage" et "(en)cryptement" sont des anglicismes dérivés du verbe "to encrypt".

2.2 Notations

En cryptographie, la propriété de base est que

$$M = D(E(M))$$

où

- M représente le texte clair,
- C est le texte chiffré,
- K est la clé (dans le cas d'un algorithme à clé symétrique), E_k et D_k dans le cas d'algorithmes asymétriques,
- $E(x)$ est la fonction de chiffrement, et
- $D(x)$ est la fonction de déchiffrement.

Ainsi, avec un algorithme à clef symétrique,

$$M = D(C) \text{ si } C = E(M)$$

2.3 Principe de Kerckhoff

> La sécurité du chiffre ne doit pas dépendre de ce qui ne peut pas être facilement changé.

En d'autres termes, aucun secret ne doit résider dans l'algorithme mais plutôt dans la clé. Sans celle-ci, il doit être impossible de retrouver le texte clair à partir du texte chiffré. Par contre, si on connaît K, le déchiffrement est immédiat.

On parle aussi de la Maxime de Shannon, dérivée du principe énoncé ci-dessus : *L'adversaire connait le système.*

Remarque : Il faut distinguer les termes "Secret" et "Robustesse" d'un algorithme. Le secret de l'algorithme revient à cacher les concepts de celui-ci, ainsi que les méthodes utilisées (fonctions mathématiques). La robustesse quant à elle désigne la résistance de l'algorithme à diverses attaques qui seront explicitées dans la suite de ces notes.

2.4 La publication des algorithmes

Selon l'endroit où réside le secret, on peut parler d'algorithme secret ou d'algorithme publié[1]. Chacun possède ses atouts et inconvénients.

2.4.1 Algorithme secret

- La cryptanalyse, souvent basée sur le secret de la clé, doit ici en plus retrouver l'entièreté de l'algorithme (mécanisme de récupération).
- Souvent, de tels algorithmes sont utilisés par un plus petit nombre d'utilisateurs. Et comme souvent dans ce cas, moins il y a de monde l'utilisant, moins il y a d'intérêts à le casser.
- De tels algorithmes sont rarement distribués par delà les frontières, afin de garder un nombre d'utilisateurs restreint.

2.4.2 Algorithme publié

- Puisque l'algorithme est publié, tout le monde a le droit de l'explorer. Ainsi, les failles (laissées intentionnellement ou non par les concepteurs) peuvent être plus facilement découvertes. La sécurité en est donc améliorée.
- Comme la publication est autorisée, il n'est pas nécessaire de chercher à protéger le code contre le reverse-engineering.
- Cette publication permet d'étendre les travaux sur l'algorithme au niveau mondial. Toute une série d'implémentations logicielles peuvent donc être réalisées.
- Tout le monde utilise la même version publique ce qui permet une standardisation générale.

En conséquence, on préférera les algorithmes publiés, souvent plus sûrs pour les raisons explicitées ci-dessus.

[1] et donc, dans lequel le secret réside dans la clé.

2.5 Les principaux concepts cryptographiques

2.5.1 Cryptosystème à clé symétrique

Caractéristiques :

- Les clés sont identiques : $K_E = K_D = K$,
- La clé doit rester secrète,
- Les algorithmes les plus répandus sont le DES, AES, 3DES, ...
- Au niveau de la génération des clés, elle est choisie aléatoirement dans l'espace des clés,
- Ces algorithmes sont basés sur des opérations de transposition et de substitution des bits du texte clair en fonction de la clé,
- La taille des clés est souvent de l'ordre de 128 bits. Le DES en utilise 56, mais l'AES peut aller jusque 256,
- L'avantage principal de ce mode de chiffrement est sa rapidité,
- Le principal désavantage réside dans la distribution des clés : pour une meilleure sécurité, on préfèrera l'échange manuel. Malheureusement, pour de grands systèmes, le nombre de clés peut devenir conséquent. C'est pourquoi on utilisera souvent des échanges sécurisés pour transmettre les clés. En effet, pour un système à N utilisateurs, il y aura $N.(N-1)/2$ paires de clés.

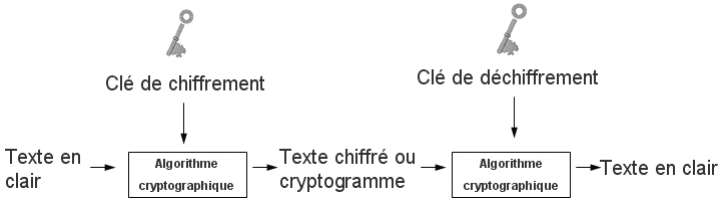

FIG. 2.3 – Chiffrement symétrique

2.5.2 Cryptosystème à clé publique

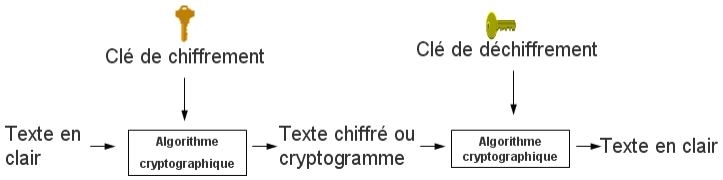

FIG. 2.4 – Chiffrement asymétrique

Caractéristiques :

- Une clé publique P_K (symbolisée par la clé verticale),
- Une clé privée secrète S_K (symbolisée par la clé horizontale),
- Propriété : La connaissance de P_K ne permet pas de déduire S_K,
- $D_{S_K}(E_{P_K}(M)) = M$,
- L'algorithme de cryptographie asymétrique le plus connu est le RSA,
- Le principe de ce genre d'algorithme est qu'il s'agit d'une fonction unidirectionnelle à trappe. Une telle fonction à la particularité d'être facile à calculer dans un sens, mais difficile voire impossible dans le sens inverse. La seule manière de pouvoir réaliser le calcul inverse est de connaître une

trappe. Une trappe pourrait par exemple être une faille dans le générateur de clés. Cette faille peut être soit intentionnelle de la part du concepteur (définition *stricte* d'une trappe) ou accidentelle.
- Les algorithmes se basent sur des concepts mathématiques tels que l'exponentiation de grands nombres premiers (RSA), le problème des logarithmes discrets (ElGamal), ou encore le problème du sac à dos (Merkle-Hellman).
- La taille des clés s'étend de 512 bits à 2048 bits en standard. Dans le cas du RSA, une clé de 512 bits n'est plus sûre au sens "militaire" du terme, mais est toujours utilisable de particulier à particulier.
- Au niveau des performances, le chiffrement par voie asymétrique est environ 1000 fois plus lent que le chiffrement symétrique.
- Cependant, à l'inverse du chiffrement symétrique où le nombre de clés est le problème majeur, ici, seules n paires sont nécessaires. En effet, chaque utilisateur possède une paire (S_K, P_K) et tous les transferts de message ont lieu avec ces clés.
- La distribution des clés est grandement facilitée car l'échange de clés secrètes n'est plus nécessaire. Chaque utilisateur[2] conserve sa clé secrète sans jamais la divulguer. Seule la clé publique devra être distribuée.

2.5.3 Fonction de hachage

Il s'agit de la troisième grande famille d'algorithmes utilisés en cryptographie. Le principe est qu'un message clair de longueur quelconque doit être transformé en un message de longueur fixe inférieure à celle de départ. Le message réduit portera le nom de "Haché" ou de "Condensé". L'intérêt est d'utiliser ce condensé comme empreinte digitale du message original afin que ce dernier soit identifié de manière univoque. Deux caractéristiques (théoriques) importantes sont les suivantes :

1. Ce sont des fonctions unidirectionnelles :

$$\textit{A partir de } H(M) \textit{ il est impossible de retrouver } M.$$

2. Ce sont des fonctions sans collisions :

$$\textit{A partir de } H(M) \textit{ et } M \textit{ il est impossible de trouver } M' \neq M \textit{ tel que } H(M') = H(M).$$

Il est bien entendu que le terme “impossible” n'est pas toujours à prendre au pied de la lettre ! Il s'agit ici de concepts théoriques. La réalité est quelque peu différente. Ainsi, pour le caractère “sans collision”, dans les faits, cela est “très difficile” dans le meilleur des cas, mais jamais impossible, comme le bon sens le laisse penser.

2.5.4 Protocoles cryptographiques

Dès que plusieurs entités sont impliquées dans un échange de messages sécurisés, des règles doivent déterminer l'ensemble des opérations cryptographiques à réaliser, leur séquence, afin de sécuriser la communication C'est ce que l'on appelle les protocoles cryptographiques.

Lorsque l'on parle de “sécuriser un échange”, on souhaite prêter attention aux 3 services suivants : la confidentialité, l'intégrité et l'authentification.

Signalons la distinction entre “services” (confidentialité, intégrité, etc.) et “mécanismes” (les moyens utilisés : chiffrement, signature, hachage, etc.).

[2]Pour désigner un utilisateur, on emploiera également le terme "entité"

2.5.4.1 Confidentialité

Elle est amenée par le chiffrement du message. Dans le cas de systèmes à clés symétriques, la même clé est utilisée pour $E_K(M)$ et $D_K(C)$. Ce type de chiffrement nécessite un échange sûr préalable de la clé K entre les entités A et B.

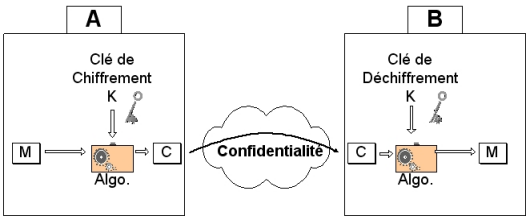

FIG. 2.5 – Confidentialité d'un système symétrique

Comme dit précédemment, à l'aide d'un cryptosystème asymétrique, cet échange préalable n'est pas nécessaire. Chaque entité possède sa propre paire de clés. On aura donc la paire P_{KA}, S_{KA} pour l'entité A et la paire P_{KB}, S_{KB} pour l'entité B.

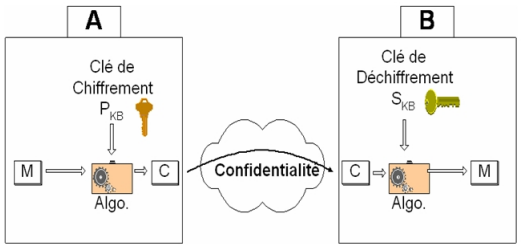

FIG. 2.6 – Confidentialité d'un système asymétrique

En marge de ces deux systèmes, existe également un système appelé "hybride" (figure 2.7), reposant comme son nom l'indique sur les deux systèmes précédents. Par l'intermédiaire du système à clé publique, on sécurise l'échange de la clé K. Ensuite, les deux parties ayant acquis de manière sécurisée cette clé de chiffrement K[3], on utilisera le système à clé symétrique pour chiffrer le message.

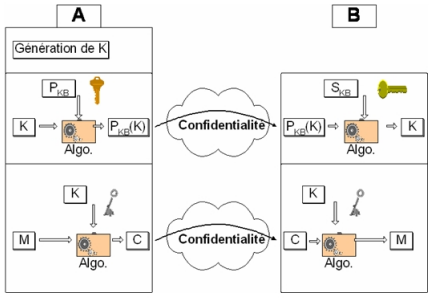

FIG. 2.7 – Confidentialité d'un système hybride

[3]Pour la clé K dans un système symétrique, on parle également de "clé de session"

2.5.4.2 Intégrité

Il faut ici vérifier si le message n'a pas subi de modification durant la communication. C'est ici qu'interviennent les fonctions de hachage.

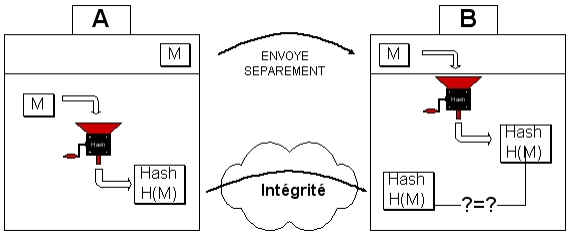

FIG. 2.8 – Vérification de l'intégrité par fonction de hachage

Dans la figure 2.8, on ne parle pas de l'envoi du message. On prête uniquement l'attention à la vérification de l'intégrité.

2.5.4.3 Authentification

Elle a lieu à plusieurs niveaux.

– Au niveau des parties communicantes, dans le cas d'un système symétrique (figure 2.9) ou asymétrique (figure 2.10). A la première figure, R_a est une nonce (p. ex. nombre aléatoire), propre à l'utilisateur A. Les lettres A et B représentent des identificateurs personnels.

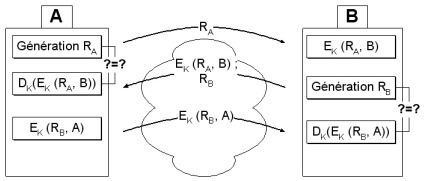

FIG. 2.9 – Authentification dans un système symétrique

A la seconde figure, la clé de chiffrement utilisée est bien la clé privée. Comme le propriétaire de cette clé est le seul à la connaitre, cela prouve qu'il est bien la personne ayant chiffré le message. Attention, dans cet exemple, seule l'authentification est souhaitée. Le message envoyé pourra être lu par toute personne possédant la clé publique, c'est-à-dire, n'importe qui. La confidentialité est ici nulle.

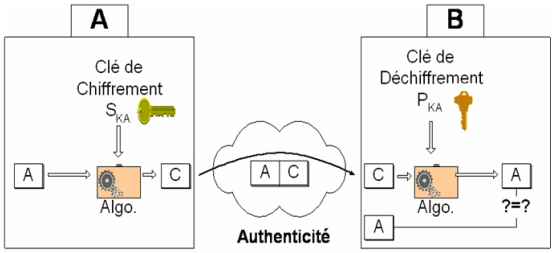

FIG. 2.10 – Authentification dans un système asymétrique

– Au niveau du message

  – Par l'utilisation d'un MAC (Message Authentication Code) généré à l'aide d'un cryptosystème à clé symétrique où le MAC est constitué des derniers digits de C (figure 2.11), ou généré à l'aide d'une fonction de hachage (figure 2.12), la clé secrète K utilisée étant partagée par les deux entités A et B. Dans les deux cas, l'authentification repose sur l'utilisation de la clé K.

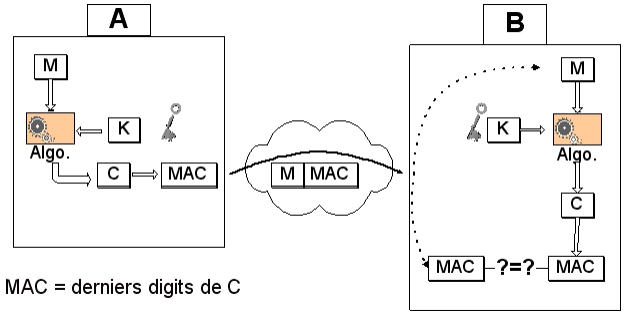

FIG. 2.11 – Authentification par MAC et système symétrique

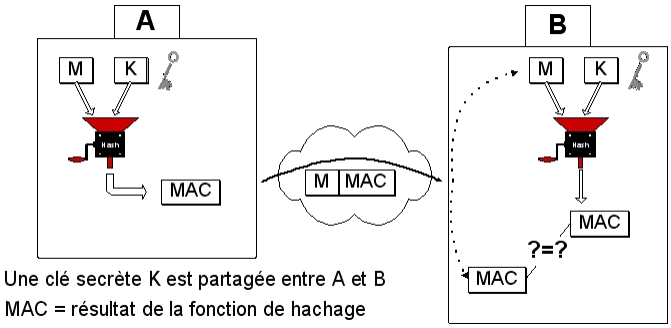

FIG. 2.12 – Authentification par MAC et fonction de hachage

  – Par l'utilisation d'une signature digitale. Parmi les propriétés remarquables de ces signatures, on peut dire qu'elles doivent être authentiques, infalsifiables, non-réutilisables, non-répudiables, et inaltérables. Dans la figure 2.13, on fait abstraction de la confidentialité. C'est l'authentification qui importe.

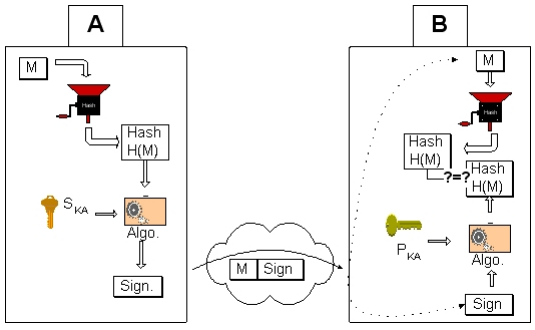

FIG. 2.13 – Authentification par signature (technique asymétrique)

2.5.4.4 Synthèse

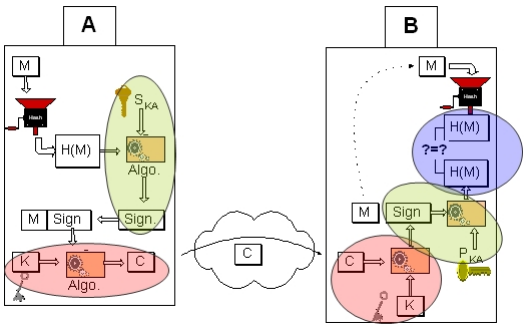

FIG. 2.14 – Confidentialité(Rouge), Intégrité(Violet), Authentification(Vert)

Chapitre 3

La cryptographie classique

Dans le schéma ci-dessous figurent les différentes branches de la cryptographie classique.

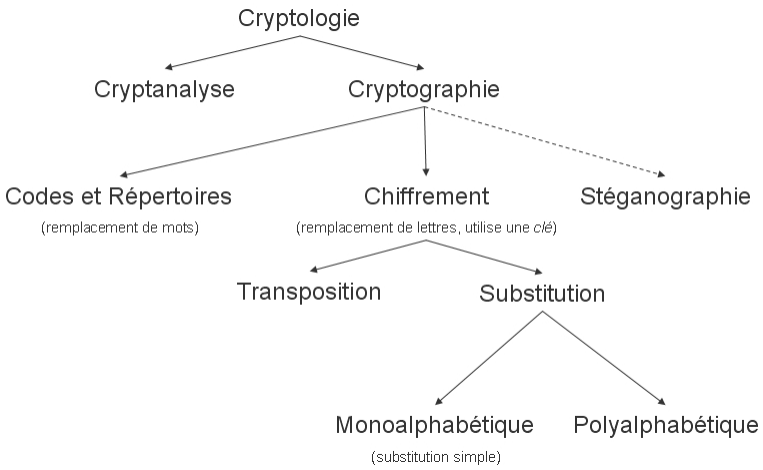

FIG. 3.1 – Domaines inclus dans la cryptologie.

3.1 Substitution monoalphabétique

Chaque lettre est remplacée par une autre lettre ou symbole. Parmi les plus connus, on citera le chiffre de César, le chiffre affine, ou encore les chiffres désordonnés. Tous ces chiffres sont sensibles à l'analyse de fréquence d'apparition des lettres (nombre de fois qu'apparait une même lettre dans un texte). De nos jours, ces chiffres sont utilisés pour le grand public, pour les énigmes de revues ou de journaux.

Historiquement, on recense des procédés de chiffrement remontant au Xème siècle avant JC. On trouve par exemple, l'Atbash des Hébreux (-500), la scytale à Sparte (-400), le carré de Polybe (-125), ... Des langues anciennes sont également parfois classifiées dans les codes secrets : le Rongo-Rongo, le linéaire A, les écritures du disque de Phaistos en sont des exemples. Intraduisibles à l'heure actuelle, on les place (à tort ?) dans ce domaine.

3.1.1 Chiffre de César (50 av. J-C)

Il s'agit d'un des plus simples et des chiffres classiques les plus populaires. Son principe est un décalage des lettres de l'alphabet. Dans les formules ci-dessous, p est l'indice de la lettre de l'alphabet, k est le décalage.

Pour le chiffrement, on aura la formule

$$C = E(p) = (p + k) \bmod 26$$

Pour le déchiffrement, il viendra

$$p = D(C) = (C - k) \bmod 26$$

Si on connait l'algorithme utilisé (ici César), la cryptanalyse par force brute est très facile. En effet, dans le cas du chiffre de César, seules 25 (!) clés sont possibles.

3.1.2 Analyse de fréquences

Lorsque la langue de départ et la technique de chiffrement sont connus, on peut exploiter les régularités du langage par le principe d'analyse de la fréquence d'une lettre. Cette technique ne fonctionne bien que si le message chiffré est suffisamment long pour avoir des moyennes significatives.

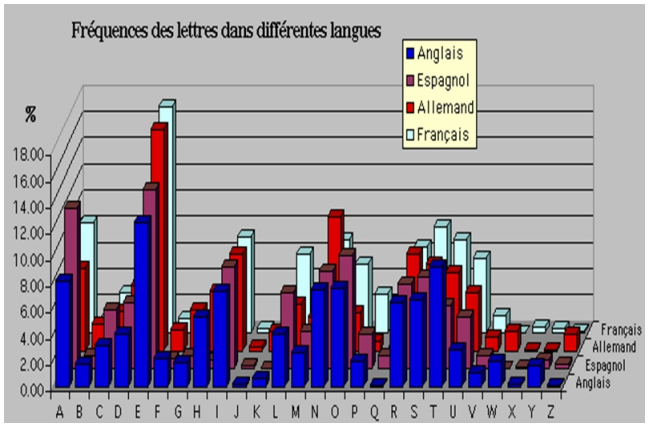

FIG. 3.2 – Exemple d'analyse de fréquence

Cependant, il existe également des cas où cette analyse ne fonctionne pas, comme le montre l'exemple ci-dessous.

Exemple : De Zanzibar à la Zambie et au Zaïre, des zones d'ozone font courir les zèbres en zigzags zinzins.

□

Pour éviter ce type d'attaque sur un texte chiffré, il existe différents moyens :

– On peut par exemple chiffrer le message par digrammes, trigrammes, etc.

Les 20 bigrammes les plus fréquents

Bigrammes	ES	DE	LE	EN	RE	NT	ON	ER	TE	EL	AN	SE	ET	LA	AI	IT	ME	OU	EM	IE
Nombres	3318	2409	2366	2121	1885	1694	1646	1514	1484	1382	1378	1377	1307	1270	1255	1243	1099	1086	1056	1030

Les 20 trigrammes les plus fréquents

Trigrammes	ENT	LES	EDE	DES	QUE	AIT	LLE	SDE	ION	EME	ELA	RES	MEN	ESE	DEL	ANT	TIO	PAR	ESD	TDE
Nombres	900	801	630	609	607	542	509	508	477	472	437	432	425	416	404	397	383	360	351	350

FIG. 3.3 – Bigrammes et Trigrammes

– On peut également utiliser des homophones :

Il s'agit de remplacer une lettre non pas par un symbole unique, mais par un symbole choisi au hasard parmi plusieurs. Dans sa version la plus sophistiquée, on choisira un nombre des symboles proportionnel à la fréquence d'apparition de la lettre. Ainsi, on obtient un renversement des fréquences ce qui permet de faire disparaître complètement les indications fournies par la fréquence. On parle de Substitution Homophonique ou de Substitution à représentation multiple.

Lettre	%	Chiffre
A	9	09 12 33 47 48 53 67 78 92
B	1	81
C	3	13 41 62
D	3	01 03 45
E	16	06 10 14 16 23 24 44 46 54 55 57 74 79 82 87 98
F	1	31
G	1	25
H	1	39
I	8	32 50 56 70 73 83 88 93
J	1	15
K	0	04
L	5	26 37 51 65 84
M	3	22 27 68

Lettre	%	Chiffre
N	7	18 58 59 66 71 91 99
O	5	00 05 07 54 72
P	3	38 90 95
Q	1	94
R	6	29 35 40 42 77 80
S	8	11 19 21 36 76 86 96 97
T	7	17 20 30 43 49 69 75
U	6	02 08 61 63 85 90
V	2	34 52
W	0	60
X	0	28
Y	0	24
Z	0	01

FIG. 3.4 – Exemple de table pour un chiffrement par homophones

3.1.3 Chiffre affine

On dit qu'une fonction est affine lorsqu'elle est de la forme $x \to a * x + b$, c'est-à-dire un polynôme de degré 1. Une fonction linéaire est une fonction affine particulière.

L'idée est d'utiliser comme fonction de chiffrement une fonction affine du type

$$y = (ax + b) \bmod 26,$$

où a et b sont des constantes, et où x et y sont des nombres correspondant aux lettres de l'alphabet (A=0,B=1,...). On peut remarquer que si $a = 1$, alors on retrouve le chiffre de César où b est le décalage (le k du chiffre de César).

Propriété de neutralité : si $b = 0$, alors "a" est toujours chiffré "A" car il ne subit aucun décalage. En effet, si aucun décalage n'a lieu, l'alphabet de départ se retrouve chiffré par lui même, et donc ne subit aucune modification.

□

Pour le chiffre affine, la clé est constituée de (k_1, k_2) où $k_1, k_2 \in [0, 25]$ et telle que

$$\gcd(k_1, 26) = 1.$$

Le chiffrement en lui-même est donné par

$$c_i = f(m_i) = k_1 * m_i + k_2 \bmod 26.$$

Pour le déchiffrement, il vient

$$m_i = f^{-1}(c_i) = k_1^{-1} * (c_i - k_2) \bmod 26.$$

Par le chiffre affine, on obtient 312 clés possibles. En effet, pour respecter la propriété de k_1, il n'y a que 12 choix possibles. Et puisque k_2 peut prendre n'importe quelle valeur dans $[0, 25]$, il vient $12 * 26 = 312$.

Exemple : Soient la clé $= (k_1, k_2) = (3, 11)$

Transformation de chiffrement :

$$c_i = f(m_i) = 3 * m_i + 11 \bmod 26$$

Transformation de déchiffrement :

$$k_1^{-1} = 3^{-1} \bmod 26 = 9 \ [car\ 3 * 9 \bmod 26 = 1]$$

$$m_i = f^{-1}(c_i) = 9 * (c_i - 11) \bmod 26$$

Ainsi, pour une suite de lettres telle que 'NSA' $\rightarrow$ 13 18 0 $\rightarrow$ 24 13 11 $\rightarrow$'YNL'.

□

3.1.3.1 Cryptanalyse du chiffre affine

1. Il faut tout d'abord établir la fréquence relative de chaque lettre du texte chiffré, par analyse de fréquence.

   HGAHY RAEFT GAGRH DGAGM OEHIY RAAOT ZGAGJ GKFDG AZGSB INNTG KGRHE NNIRG

   On dénombre 12 fois la lettre G et 8 fois la lettre A. Supposons que le langage original du texte est le français.

2. Sur base de l'analyse de fréquences, il faut ensuite dériver les équations correspondantes. E, A, S, I, N étant les lettres les plus fréquentes en français, on en déduit les équations suivantes (en émettant l'hypothèse que le bigramme ES est plus fréquent que EA) :

   $$E \rightarrow G \Rightarrow f(E) = G$$
   $$S \rightarrow A \Rightarrow f(S) = A$$

   Il en découle que :

   $$4 \rightarrow 6 \Rightarrow f(4) = 6$$
   $$18 \rightarrow 0 \Rightarrow f(18) = 0$$

3. On peut maintenant résoudre les équations pour que retrouver k_1 et k_2.

$$f(4) = 6,\ f(18) = 0$$
$$\downarrow$$
$$4 * k_1 + k_2 \equiv 6(\text{mod}26)$$
$$18 * k_1 + k_2 \equiv 0(\text{mod}26)$$
$$\downarrow$$
$$14k_1 \equiv -6 \mod 26$$
$$\downarrow$$
$$k_1 = 7 \Rightarrow k_2 = 4.$$

4. La fonction de déchiffrement est donc la suivante : $m_i = 15 * (c_i - 4) \mod 26$.
   Et donc,

   HGAHYRAEFTGAGRHDGAGMOEHIYRAAOTZGAGJGKFDGAZGSBINNTGKGRHENNIRG

   devient

   TESTONSAPRESENTLESEQUATIONSSURDESEXEMPLESDECHIFFREMENTAFFINE

3.2 Chiffrement polygraphique

Il s'agit ici de chiffrer un groupe de n lettres par un autre groupe de n symboles. On citera notamment le chiffre de Playfair et le chiffre de Hill. Ce type de chiffrement porte également le nom de *substitutions polygrammiques*.

3.2.1 Chiffre de Playfair (1854)

On chiffre 2 lettres par 2 autres. On procède donc par digramme. On dispose les 25 lettres de l'alphabet (W exclu car inutile à l'époque, on utilise V à la place) dans une grille de 5x5, ce qui donne la clef. La variante anglaise consiste à garder le W et à fusionner I et J.

Il y a 4 règles à appliquer selon les deux lettres à chiffrer lors de l'étape de substitution. Pour le déchiffrement, on procède dans l'ordre inverse.

1. Si les lettres sont sur des "coins", les lettres chiffrées sont les 2 autres coins.
   Exemple : OK devient VA, RE devient XI ...
2. Si les lettres sont sur la même ligne, il faut prendre les deux lettres qui les suivent immédiatement à leur droite.
3. Si les lettres sont sur la même colonne, il faut prendre les deux lettres qui les suivent immédiatement en dessous.
4. Si elles sont identiques, il faut insérer une *nulle* (habituellement le X) entre les deux pour éliminer ce doublon.
   Exemple : "balloon" devient "ba" "lx" "lo" "on".

Pour former ces grilles de chiffrement, on utilise un mot-clef secret pour créer un alphabet désordonné avec lequel on remplit la grille ligne par ligne. Ensuite, on comble la grille avec les lettres restantes de l'alphabet.

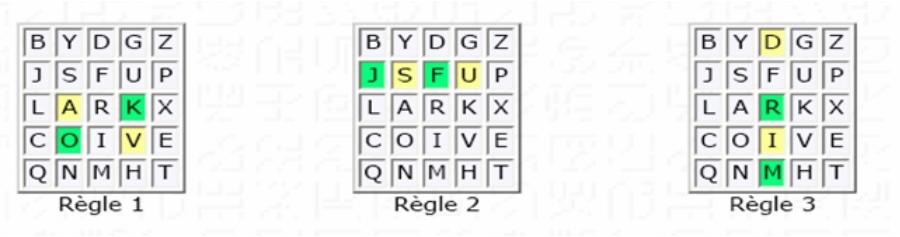

FIG. 3.5 – Exemple du chiffre de Playfair

3.2.2 Chiffre de Hill (1929)

Les lettres sont d'abord remplacées par leur rang dans l'alphabet. Les lettres P_k et P_{k+1} deviennent C_k et C_{k+1}

$$\begin{pmatrix} C_k \\ C_{k+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} P_k \\ P_{k+1} \end{pmatrix} (\text{mod} 26)$$

Les composantes de cette matrice doivent être des entiers positifs. De plus la matrice doit être inversible dans Z_{26}. Cependant, sa taille n'est pas fixée à 2. Elle grandira selon le nombre de lettres à chiffrer simultanément.

Chaque digramme clair (P_1 et P_2) sera chiffré (C_1 et C_2) selon :

$$C_1 \equiv aP_1 + bP_2 (\text{mod} 26)$$
$$C_2 \equiv cP_1 + dP_2 (\text{mod} 26)$$

Exemple de chiffrement : Alice prend comme clef de cryptage la matrice

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$$

pour chiffrer le message "je vous aime" qu'elle enverra à Bob. Après avoir remplacé les lettres par leur rang dans l'alphabet (a=1, b=2, etc.), elle obtiendra

$$C_1 \equiv 9 * 10 + 4 * 5 (\text{mod} 26) = 110 (\text{mod} 26) = 6$$

$$C_2 \equiv 5 * 10 + 7 * 5 (\text{mod} 26) = 85 (\text{mod} 26) = 7$$

Elle fera de même avec les 3e et 4e lettres, 5e et 6e, etc. Elle obtiendra finalement le résultat de la figure 3.6.

Lettres	j	e	v	o	u	s	a	i	m	e
Rangs (P_k)	10	5	22	15	21	19	1	9	13	5
Rangs chiffrés (C_k)	6	7	24	7	5	4	19	16	7	22
Lettres chiffrées	F	G	X	G	E	D	S	P	G	V

FIG. 3.6 – Exemple de chiffrement de Hill

□

Pour déchiffrer, le principe est le même que pour le chiffrement : on prend les lettres deux par deux, puis on les multiplie par une matrice

$$\begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} (\text{mod} 26)$$

Cette matrice doit être l'inverse de matrice de chiffrement (modulo 26). Ordinairement, cet inverse est

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Exemple de déchiffrement : Pour déchiffrer le message d'Alice, Bob doit calculer :

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}^{-1} = \frac{1}{43} \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} (\text{mod} 26) = (43)^{-1} \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} (\text{mod} 26)$$

Comme $\gcd(43, 26) = 1$, $(43)^{-1}$ existe dans Z_{26} et $(43)^{-1} = 23$. Bob a la matrice de déchiffrement :

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}^{-1} = 23 \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} (\text{mod} 26) = \begin{pmatrix} 161 & -92 \\ -115 & 207 \end{pmatrix} (\text{mod} 26) = \begin{pmatrix} 5 & 12 \\ 15 & 25 \end{pmatrix} (\text{mod} 26)$$

Bob prend donc cette matrice pour déchiffrer le message "FGXGE DSPGV". Après avoir remplacé les lettres par leur rang dans l'alphabet (A=1, B=2, etc.), il obtiendra :

$$P_1 \equiv 5 * 6 + 12 * 7 (\text{mod} 26) = 114 (\text{mod} 26) = 10$$

$$P_2 \equiv 5 * 6 + 25 * 7 (\text{mod} 26) = 265 (\text{mod} 26) = 5$$

Il fera de même avec les 3e et 4e lettres, 5e et 6e, etc. Il obtiendra finalement le résultat de la figure 3.7

Lettres chiffrées	F	G	X	G	E	D	S	P	G	V
Rangs chiffrés (C_k)	6	7	24	7	5	4	19	16	7	22
Rangs (P_k)	10	5	22	15	21	19	1	9	13	5
Lettres	j	e	v	o	u	s	a	i	m	e

FIG. 3.7 – Exemple de déchiffrement de Hill

□

3.3 Substitutions polyalphabétiques

3.3.1 Chiffre de Vigenère (1568)

C'est une amélioration décisive du chiffre de César. Sa force réside dans l'utilisation non pas d'un, mais de 26 alphabets décalés pour chiffrer un message. On parle du *carré de Vigenère*. Ce chiffre utilise une clef qui définit le décalage pour chaque lettre du message (A : décalage de 0 cran, B : 1 cran, C : 2 crans, ..., Z : 25 crans).

Exemple : chiffrer le texte "**CHIFFRE DE VIGENERE**" avec la clef "**BACHELIER**" (cette clef est éventuellement répétée plusieurs fois pour être aussi longue que le texte clair)

Clair	C	H	I	F	F	R	E	D	E	V	I	G	E	N	E	R	E
Clef	B	A	C	H	E	L	I	E	R	B	A	C	H	E	L	I	E
Décalage	1	0	2	7	4	11	8	4	17	1	0	2	7	4	11	8	4
Chiffré	D	H	K	M	J	C	M	H	V	W	I	I	L	R	P	Z	I

FIG. 3.8 – Application du carré de Vigenère

□

La grande force du chiffre de Vigenère est que la même lettre sera chiffrée de différentes manières d'où perte de la fréquence des lettres, ce qui rend inutilisable l'analyse de fréquence classique. La figure 3.9 illustre cette perte des fréquences dans une fable de Lafontaine, codée par substitution simple et par Vigenère.

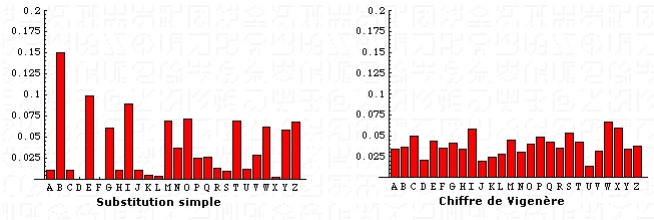

FIG. 3.9 – Perte de la fréquence des lettres

Pour utiliser le chiffrement de Vigenère, on a recours au *Carré de Vigenère*, illustré à la figure 3.10.

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

FIG. 3.10 – Le carré de Vigenère

La lettre de la clef est dans la colonne la plus à gauche, la lettre du message clair est dans la ligne tout en haut. La lettre chiffrée est à l'intersection des deux.

L'emploi du carré de Vigenère est souvent sujet à erreurs : la lecture en est pénible et, à la longue, fatigante. Beaucoup de cryptologues préfèrent se servir d'une "réglette", facile à construire, et d'un maniement plus rapide.

3.3.1.1 Cryptanalyse de Kasiski

Cette technique consiste à chercher des séquences de lettres qui apparaissent plus d'une fois dans le texte. En effet, dans ce cas, il n'y aura que deux possibilités :

- soit la même séquence de lettres du texte clair a été chiffrée avec la même partie de la clef,
- soit deux suites de lettres différentes dans le texte clair auraient par pure coïncidence engendré la même suite dans le texte chiffré (probabilité faible).

Exemple : Soit une séquence de 3 lettres répétée dans un message codé avec une distance d. On fait donc l'hypothèse sensée qu'il s'agit d'une même séquence de 3 lettres du texte initial, codée avec la même séquence de lettres de la clé. Par conséquent, si m est la longueur de la clé, pour que les 2 séquences soient codées avec les mêmes lettres de la clé, il est nécessaire que m divise d.
m sera donc le pgcd des distances de séquences répétées. m représentera la longueur de la clé de chiffrement !
Il faut toutefois baser ce raisonnement sur les valeurs significatives du pgcd. Ainsi, dans l'exemple 3.11, les suites FCS et WTV ne seront pas prises en compte.

CS AZZMEQM, CO XRWF, CS DZRM GFMJECV. X'IMOQJ JC LB NLFMK. CC LBM WCCZBM
KFIMSZJSZ CS URQIUOU. CS ZLPIE ECZ RMWWTV, SB KCCJ QMJ FCSOVJ GCI ZI ICCKS, MK
QMLL YL'CV ECCJ OKTFWTVM JIZ CO XFWBIWVV, IV ACCI CC C'OCKFM, JINWWB
U'OBKSVUFM

Séquence	Position	Distance	Décomposition
COX	11-140	129	3.43
FCS	16-99	83	83
ZRM	20-83	63	$3^2 7$
FMJ	24-162	138	2.3.23
CLB	37-46	9	3^2
KCC	44-92	48	$2^3 3$
WTV	87-133	46	2.23
CCJ	93-126	33	3.11
ICC	110-155	45	$3^2.5$
MJI	136-163	27	3^3

FIG. 3.11 – Méthode de Kasiski

□

Ce renseignement est capital. Cela signifie que les caractères de rang $1, 4, 7, 10, ..., 3k+1$, sont simplement décalés à la manière du chiffre de César. On peut donc appliquer maintenant l'analyse de fréquence à ces caractères et trouver la première lettre de la clef. Pour la deuxième lettre de la clef, on analysera les fréquences des caractères de rang $3k+2$ et pour le dernière lettre les fréquences des caractères de rang $3k$.

3.3.1.2 Cryptanalyse de Friedman

Cette méthode utilise la notion d'Indice de Coïncidence (IC). Celui-ci est défini comme la probabilité que deux lettres choisies aléatoirement dans un texte soient identiques. Soient n le nombre de lettres dans le texte, n_1 = nombre de A, ...,n_{26} = nombre de Z.

La probabilité de tirer deux A parmi les n lettres du texte est :

$$P(2\ fois\ A) = \frac{C_2^{n_1}}{C_2^n} = \frac{\frac{n_1(n_1-1)}{2}}{\frac{n(n-1)}{2}} = \frac{n_1(n_1-1)}{n(n-1)}$$

Plus généralement, la probabilité de tirer 2 lettres identiques est donnée par :

$$IC = \sum_{i=1}^{26} \frac{n_i(n_i-1)}{n(n-1)}$$

La figure 3.12 donne quelques indices calculés sur des textes contemporains dans différentes langues.

Langue	allemand	anglais	espagnol	esperanto	français	italien	norvégien	suédois
IC	0.072	0.065	0.074	0.069	0.074	0.075	0.073	0.071

FIG. 3.12 – IC des principales langues européennes

Plus n est grand, plus la probabilité est correcte.

Le test de Friedman (aussi appelé *Test Kappa*) a pour premier objectif de déterminer si un texte a été chiffré avec un chiffre monoalphabétique ou polyalphabétique. Comme second bénéfice, il suggère la longueur du mot-clef si le chiffre est polyalphabétique. Pour réaliser ces buts, le test de Friedman s'appuie sur l'IC.

Remarques importantes :

- Pour un langage de 26 lettres où chaque lettre a la même fréquence (1/26), l'IC vaut 0.038.
- Pour tout chiffre monoalphabétique, la distribution des fréquences est invariante, donc l'IC sera le même que pour le texte clair (respect des fréquences).
- Donc, si on applique ce test à un texte chiffré avec un chiffre monoalphabétique, on devrait trouver IC égal environ à 0.074 (en français). Si IC est beaucoup plus petit (p. ex. 0.050), le chiffre est probablement polyalphabétique.

Exemple de cryptanalyse par le test de Friedman

Soit le message suivant, supposé écrit en français, chiffré avec Vigenère (369 lettres) :

PERTQ UDCDJ XESCW MPNLV MIQDI ZTQFV XAKLR PICCP QSHZY DNCPW EAJWS ZGCLM QNRDE OHCGE ZTQZY HELEW AUQFR OICWH QMYRR UFGBY QSEPV NEQCS EEQWE EAGDS ZDCWE OHYDW QERLM FTCCQ UNCPP QSKPY FEQOI OHGPR EERWI EFSDM XSYGE UELEH USNLV GPMFV EIVXS USJPW HIEYS NLCDW MCRTZ MICYX MNMFZ QASLZ QCJPY DSTTK ZEPZR ECMYW OICYG UESIU GIRCE UTYTI ZTJPW HIEYI ETYYH USOFI XESCW HOGDM ZSNLV QSQPY JSCAV QSQLM QNRLP QSRLM XLCCG AMKPG QLYLY DAGEH GERCI RAGEI ZNMGI YBPP

On va considérer les sous-chaînes obtenues en prenant les lettres à intervalle donné :

– Intervalle de 1 : PERTQ UDCDJ XESCW MPNLV ... (texte original)
– Intervalle de 2 : PRQDD XSWPL ... et ETUCJ ECMNV ...
– Intervalle de 3 : PTDJS MLIIQ ..., EQCXC PVQZF... et RUDEW NMDTV
– ...

On calcule ensuite les IC pour toutes ces sous-chaînes, procédé illustré à la figure 3.13.

Intervalle	Indice de coïncidence
1	0.0456107
2	0.0476954, 0.0443098
3	0.044249, 0.0494469, 0.0426771
4	0.0465839, 0.0453894, 0.0449116, 0.0425227
5	0.0799704, 0.0925583, 0.0836727, 0.0795282, 0.0684932
6	0.0512956, 0.0407192, 0.0371585, 0.0382514, 0.0661202, 0.0431694

FIG. 3.13 – Indice de coïncidence des sous-chaines

On remarque que quand l'intervalle est de 5, l'IC correspond plus ou moins avec l'IC caractéristique du français (en tout cas, c'est cette ligne qui s'approche le plus de 0.074, les autres lignes étant plutôt proches de 0.038). La longueur de la clef utilisée est donc probablement 5.

□

D'autre part, si un message en français de longueur n et d'indice de coïncidence IC est chiffré avec un carré de Vigenère, alors on peut définir r, la longueur du mot-clef composé de lettres distinctes, qui est donné par la formule suivante

$$r \approx \frac{(0.074 - 0.038)n}{(n-1)IC - 0.038n + 0.074} \approx \frac{0.036n}{(n-1)IC - 0.038n + 0.074}$$

En appliquant cette formule au texte précédent, on trouve r = 4.69, ce qui confirme ce que l'on avait trouvé ci-dessus.

3.3.2 Chiffre de Vernam (One Time Pad - 1917)

Le masque jetable est défini comme un chiffre de Vigenère avec la caractéristique que la clef de chiffrement a la même longueur que le message clair.

Clair	M	A	S	Q	U	E	J	E	T	A	B	L	E
Clef	X	C	A	A	T	E	L	P	R	V	G	Z	C
Décalage	23	2	0	0	19	4	11	15	17	21	6	25	2
Chiffré	J	C	S	Q	N	I	U	T	K	V	H	K	G

FIG. 3.14 – Exemple de One Time Pad

Pour utiliser ce chiffrement, il faut respecter plusieurs propriétés :

– choisir une clef aussi longue que le texte à chiffrer,
– utiliser une clef formée d'une suite de caractères aléatoires,
– protéger votre clef,
– ne jamais réutiliser une clef.

Exemple illustrant l'inviolabilité :

Soit le texte chiffré : **cuskqxwmfwituk**

Soit le masque jetable possible : **bgfbcdfbfdecdg**

Résultat : BONJOURLATERRE

Soit un autre masque jetable : **quauwtedbdisjg**

Résultat : MASQUESJETABLE

Il est donc impossible de déterminer le bon masque !

□

Le système du masque jetable, avec les précautions indiquées ci-dessus, est absolument inviolable si l'on ne connaît pas la clef. Il est couramment utilisé de nos jours par les États. En effet, ceux-ci peuvent communiquer les clefs à leurs ambassades de manière sûre via la valise diplomatique.

Le problème de ce système est de communiquer les clefs de chiffrement ou de trouver un algorithme de génération de clef commun aux deux partenaires.

De plus, la création de grandes quantités des clefs aléatoires devient vite problématique. N'importe quel système couramment utilisé pourrait exiger des millions de caractères aléatoires de façon régulière.

La distribution des clés est également complexe. La longueur de la clé étant égale à celle du message, une bonne organisation est nécessaire.

3.4 Transpositions

Elles consistent, par définition, à changer l'ordre des lettres. C'est un système simple, mais peu sûre pour de très brefs messages car il y a peu de variantes. Ainsi, un mot de trois lettres ne pourra être transposé que dans 6 (=3!) positions différentes. Par exemple, "col" ne peut se transformer qu'en "col", "clo", "ocl", "olc", "lco" et "loc".

Lorsque le nombre de lettres croît, il devient de plus en plus difficile de retrouver le texte original sans connaître le procédé de brouillage. Ainsi, une phrase de 35 lettres peut être disposée de $35! = 10^{40}$ manières différentes. Ce chiffrement nécessite un procédé rigoureux convenu auparavant entre les parties.

Une transposition rectangulaire consiste à écrire le message dans une grille rectangulaire, puis à arranger les colonnes de cette grille selon un mot de passe donné (le rang des lettres dans l'alphabet donne l'agencement des colonnes).

Exemple :

A la figure 3.15, on a choisi comme clef GRAIN pour chiffrer le message SALUT LES PETITS POTS. En remplissant la grille, on constate qu'il reste deux cases vides, que l'on peut remplir avec des nulles (ou pas, selon les désirs des correspondants).

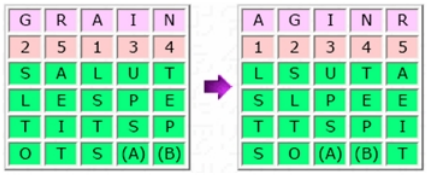

FIG. 3.15 – Application d'une transposition

3.5 Machines à rotor

Extrait en partie de [2].

L'entre-deux-guerres voit le début de la mécanisation de la cryptographie. Des outils mécaniques, comme les cylindres chiffrants, sont mis à disposition des opérateurs, et des machines électromécaniques, sont mises au point. Ces machines fonctionnent sur le principe des rotors et des contacts électriques, afin de réaliser des formes de substitution polyalphabétique dont la clef a une longueur gigantesque de l'ordre de centaines de millions de lettres, au lieu de quelques dizaines dans les méthodes artisanales, comme le chiffre de vigenère. Le tableau 3.1 donne quelques systèmes électro-mécaniques. En supplément, on pourra citer les machines Purple (Japon), Nema (Suisse) et Sigaba (USA) pour les plus célèbres, mais il en existe encore beaucoup d'autres (machines de Hebern (USA), CCM (USA), variantes d'Enigma,...)

Origine	**Nom**	**Périodes des rotors**
USA	Hagelin M-209	101 405 850 (26*25*23*21*19*17)
Allemagne	Enigma	17 576 (26*26*26)[1]
Angleterre	Typex	26*(26-k)*26 avec ($k = 5; 7; 9$)[1]
Pologne	Lacida	26 040 (24*31*35)

Tab. 3.1 – Quelques machines et leur période.

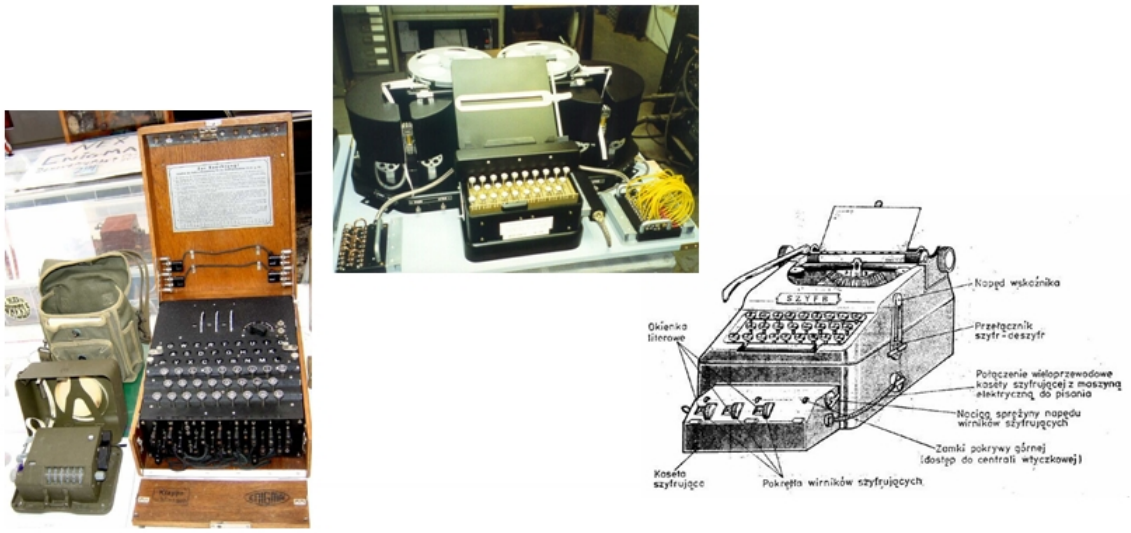

Fig. 3.16 – Différents systèmes électromécaniques

La machine Enigma

Enigma est la machine à chiffrer et déchiffrer qu'utilisèrent les armées allemandes du début des années trente jusqu'à la fin de Seconde Guerre Mondiale. Elle automatise le chiffrement par substitution.
Cette machine ressemble à une machine à écrire. Quand on presse sur une touche, deux choses se passent.

[1] Ce nombre représente la période des rotors, et ne tient pas compte des autres combinaisons possibles (connexions, ordres des rotors, sens de rotations, etc.)

Premièrement, une lettre s'allume sur un panneau lumineux : c'est la lettre chiffrée. Deuxièmement, un mécanisme fait tourner le rotor de droite d'un cran ; toutes les 26 frappes, le deuxième rotor tourne d'un cran, toutes les 676 frappes (26 au carré), c'est le troisième rotor qui tourne d'un cran. Certaines Enigmas avaient 3 rotors, celles de la Kriegsmarine en avaient 4 ou 5 (on peut apercevoir ces 4 cylindres gris sur le dessus de la machine ci-dessus).

Ces rotors tournants modifient les connexions électriques dans la machine, ce qui fait que la touche "A" allumera peut-être le "B" la première fois, mais le "X" la deuxième, le "E" la troisième, etc. Un "tableau de connexions" et un "réflecteur" complique encore le système. Le côté génial de cette machine est que même si elle tombe entre les mains ennemies, sa sécurité n'est pas compromise. En effet, c'est le nombre faramineux de réglages de la machine qui fait sa force et les réglages changeaient évidement chaque jour. On peut en effet changer l'ordre de rotors, leur orientation initiale et les branchements du tableau de connexions. Par exemple, on pouvait spécifier la clef du jour ainsi :

- Position des rotors : 2 - 3 - 1
- Orientations des rotors : 2 - 23 - 5
- Branchements des connexions : A/L - P/R - T/D - B/W - K/F - O/Y
- Indicateurs : B - W - E

Ainsi, connaître le fonctionnement de la machine n'aide (presque) pas à décrypter les messages qu'elle produit. Tout le problème est de retrouver le bon réglage.

3.5.0.1 Fonctionnement d'Enigma

Comme on le voit sur la figure 3.17, si on frappe la lettre b sur le clavier, un courant électrique est envoyé dans le rotor, suit le câblage interne, puis ressort à droite pour allumer la lettre A sur le tableau lumineux.

De plus, à chaque fois qu'une lettre est tapée au clavier, le rotor tourne d'un cran. Ainsi, b devient A la première fois, mais b devient C la deuxième fois, puis b devient E, etc.

Le principe de base des machines Enigma conçues par Scherbius repose sur l'utilisation de rotors qui transforment l'alphabet clair (noté en minuscules) en alphabet chiffré (en majuscules). Pour mieux l'illustrer, nous nous limiterons à un alphabet de six lettres. Voici la représentation de l'un de ces fameux rotors, ainsi que le schéma équivalent qui permet de mieux suivre l'opération "avec les doigts".

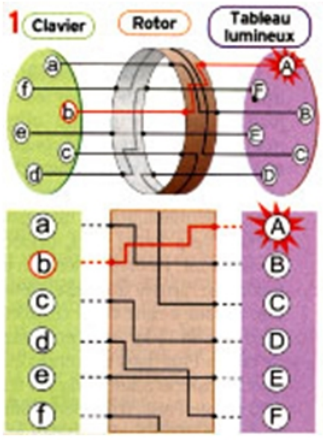

FIG. 3.17 – Première étape de chiffrement

Dans notre exemple le mot bac est chiffré ADD (et non ABD si le rotor était resté immobile).

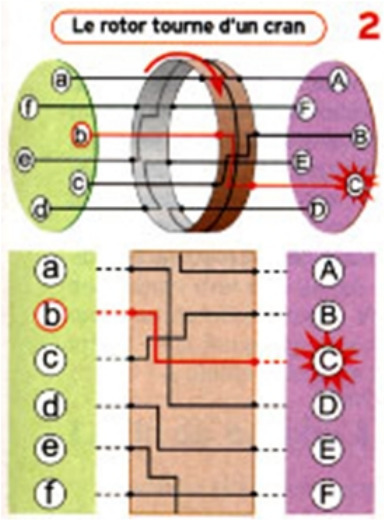

FIG. 3.18 – Deuxième étape de chiffrement

Pour augmenter le nombre de combinaisons possibles et déjouer les tentatives des cryptanalystes, Scherbius a associé plusieurs dispositifs :

- Le tableau de connexions est utilisé pour brouiller les pistes en reliant deux lettres du clavier entre elles.
- Les trois brouilleurs associés multiplient ainsi le nombre de combinaisons.
- Quant au réflecteur, il renvoie le courant dans le dispositif jusqu'au panneau lumineux où la lettre cryptée s'affiche.

Ainsi, quand on tape b, le courant prend en fait le circuit prévu pour a. Le deuxième et le troisième avancent respectivement d'un cran quand le premier et le deuxième ont fait un tour complet.

Le rôle du réflecteur n'est pas d'augmenter le nombre de combinaisons possibles, mais de faciliter considérablement la tâche du destinataire. En effet, si b devient C dans notre exemple (en rouge), on a aussi c devient B. Et c'est valable pour toutes les paires de lettres claire/cryptée. En conséquence, si le mot "efface" est chiffré "ACBFEB" par l'émetteur, il suffira à l'opérateur qui reçoit le message crypté de taper acbfeb sur son clavier pour voir les lettres E, F, F, A, C, E s'allumer. La seule condition est que les deux opérateurs distants règlent leur machine Enigma de la même façon.

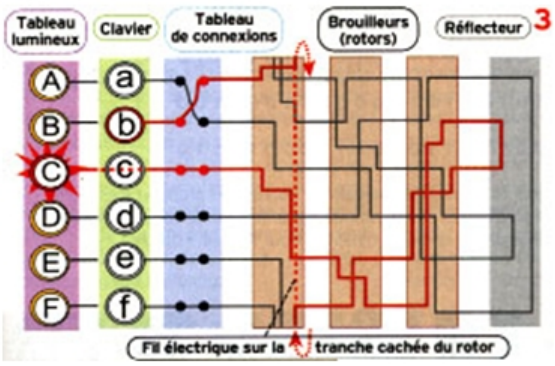

FIG. 3.19 – Troisième étape de chiffrement

Au final, on a :

- $26 * 26 * 26 = 17.576$ combinaisons liées à l'orientation des chacun des trois brouilleurs,
- 6 combinaisons possibles liées à l'ordre dans lequel sont disposés les brouilleurs,
- environ 10^{11} branchements possibles quand on relie les six paires de lettres dans le tableau de connexions.

Les machines Enigma peuvent donc chiffrer un texte selon $17.576*6*100.391.791.500 = 10^{16}$ combinaisons différentes !

3.5.0.2 La fin d'Enigma

La machine Enigma est donc un réseau électrique provoquant l'allumage d'une lampe. C'est Alan Turing, au départ de travaux de Marian Rejewski, qui créa une autre machine, appelée *Bombe de Turing* rendant sa cryptanalyse possible. Ces Bombes, illustrées à la figure 3.20, permettaient de calculer de manière exhaustive les combinaisons possibles des machines Enigma utilisées par l'ennemi. Pour gagner du temps, il fallait utiliser plusieurs bombes simultanément.

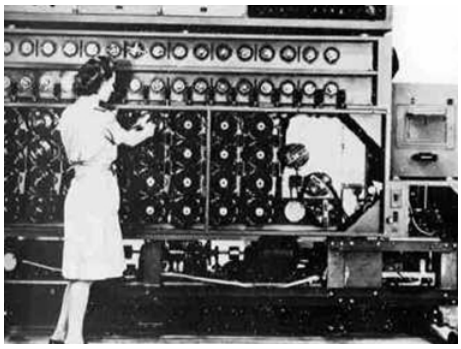

FIG. 3.20 – Bombe de Turing

Avec le temps, les machines Enigma furent modifiées, et bientôt, les attaques exhaustives devinrent trop lourdes. Mais deux méthodes furent encore utilisées :

- Test de mots courants (« wetter ») : certains termes revenaient régulièrement dans les envois de message. Ainsi, le bulletin météorologique se transmettait tous les matins à la même heure. Le terme "wetter" (= météo en Allemand) permettait d'obtenir des renseignements sur la façon dont était chiffrer les lettres. Et ces dispositions pouvaient être testées avec les Bombes, ce qui réduisait l'espace des clés possibles.
- Test de cillies : mois après mois, les envois devenaient répétitifs pour les Allemands. Certains des transcripteurs finirent par utiliser des clés de moins en moins sophistiquées, la plupart du temps, des suites de lettres au clavier (ex : azert, poiu, bhunji, etc.). Ces *cillies* permirent de retrouver les clés de chiffrement plus facilement.

3.6 Ressources supplémentaires

`http://nomis80.org/cryptographie/cryptographie.html`
`http://www.01adfm.com/win-xp/hacking/`
`http://www.apprendre-en-ligne.net/crypto/menu/index.html`
`http://perso.club-internet.fr/guidovdi/codes/lapagecryptologie.htm`
`http://www.pro-technix.com/information/crypto/pages/vernam_base.html`
`http://www.chez.com/nopb/crypto2.html#transposition`

Chapitre 4

Compléments mathématiques

4.1 Théorie de Shannon - Entropie

L'entropie est la mesure l'incertitude sur l'information et dénotée H(M).

On la définit également comme le nombre de bits de texte clair qui doivent être retrouvés pour retrouver le texte en clair en entier. Plus l'entropie est élevée, plus il y a d'incertitude et donc d'information liée à la source.

En général, H(M) est de type $\log_2 n$ (n = nbre de significations possibles).

4.1.0.3 Taux du langage (r) :

On le définit par :

$$r = H(M)/N$$

N représente la longueur du message M (par extension, M désignera l'ensemble des messages de longueur N). On peut considérer r comme le nombre moyen de bits d'information par caractère. Pour un N grand, on a en anglais $1 < r < 1.5$.

4.1.0.4 Taux absolu (R) :

C'est le nombre maximal de bits pouvant être codés par chaque caractère (séquences de caractères équiprobables) :

$$R = \log_2 L$$

L est le nombre de caractères dans un langage, R est l'entropie maximale des caractères isolés.

Pour les langues latines (alphabet de 26 lettres), on aura $R = \log_2\ 26 = 4,7$ bits par lettre. Cependant, il s'agit d'une valeur théorique, les langues réelles sont redondantes.

4.1.0.5 La redondance d'un langage (D) :

C'est la différence entre le nombre de bits maximal qui peuvent être codés par caractères et du nombre réellement nécessaire de bits pour obtenir une information sur le caractère :

$$D = R - r$$

Par exemple en anglais, D varie entre $3,2$ e t $3,7$ bits par lettre d'information redondante.

4.1.0.6 La redondance relative d'un langage :

Il s'agit du rapport

$$D/R$$

On obtient dès lors un pourcentage oscillant entre 68% et 79% (pour l'anglais).

4.1.1 Confidentialité parfaite

L'hypothèse est que les cryptanalystes disposent d'informations probabilistes concernant M (langage et redondance associée).

On définit la notion de confidentialité parfaite par le fait que le texte chiffré ne doit fournir aucune information concernant le texte clair :

- Shannon a prouvé que ce résultat est accessible uniquement si le nombre de clés possibles est aussi grand que le nombre de messages possibles.
- En pratique, seul le masque jetable y correspond car la longueur de la clé est égale à la longueur du texte et que la clé n'est pas réutilisée.

Le but des cryptanalystes est de déterminer la clef K, le texte clair M ou les deux. Souvent, ils possèdent des indices au sujet de M (i.e. s'il s'agit d'un son, d'un texte français, de données pour un tableur, etc.). Ils connaissent probablement le langage du texte en clair. Ce langage a une certaine redondance associée. Si c'est un message destiné à Bob, il commence probablement par « cher Bob ».

Le but de la cryptanalyse est, par *analyse*, de modifier les probabilités associées avec tous les textes en clair possibles. Finalement, un texte clair considéré comme certain émergera du paquet des textes clairs possibles.

4.1.1.1 Démonstration de la confidentialité parfaite du chiffre de Vernam

Soient :

- une phrase de m lettres,
- n, l'ensemble des messages possibles = 26^m (= ensemble des clés et des chiffrés possibles),
- des clés équiprobables.

On a :

$$P(K = k) = \frac{1}{n} = \frac{1}{26^m}$$

On veut prouver que la découverte du texte clair connaissant le texte chiffré est égale à la connaissance du texte clair (propriété issue de Vernam), c'est-à-dire :

$$P(M = x|C = y) = P(M = x)$$

Par définition, on a

$$P(M = x|C = y) = \frac{P(M = x, C = y)}{P(C = y)}$$

or on sait que C, K et M sont liés par la relation C=M+K (l'addition se faisant modulo 26). On obtient donc :

$$\begin{aligned} P(M = x, C = y) &= P(M = x, K = y - x) \\ &= P(M = x)P(K = y - x) \\ &= \frac{P(M = x)}{26^m} \end{aligned}$$

où on a utilisé le fait que M et K sont indépendants. D'autre part,

$$\begin{aligned} P(C = y) &= \sum_i P(m_i \ AND \ k_i) \\ &= \sum_i P(k_i)P(m_i) \\ &= \sum_i \frac{P(m_i)}{26^m} \\ &= \frac{1}{26^m} \end{aligned}$$

en utilisant le fait que M et K sont indépendants, et que la somme des probabilités de m_i est égale à 1. Ceci achève la démonstration.

De plus,

$$P(M|C) = P(M) \Rightarrow P(C|M) = P(C)$$

car $I(C; M) = 0$ (ce qui signifie que C et M sont indépendants). On en déduit également que :

$$H(M|C) = H(M)$$

Théorème 1 (Cryptosystème parfait) *Si un système cryptographique est parfait, alors :*

$$H(K) \geq H(M)$$

□

L'interprétation de ce théorème est que dans un cryptosystème parfait, il y a au moins autant d'incertitude sur les clés que sur les messages. En conséquence, dans un tel système, les clés sont aussi volumineuses que les messages.

Remarques :

- L'objectif principal en cryptographie est de minimiser l'information fournie par l'intermédiaire du texte chiffré.
- Plus un texte est redondant, plus il est simple à cryptanalyser. La compression réduit cette redondance.
- Un système parfait est à l'opposé d'un système pratique : la clé doit être de taille raisonnable et réutilisable.

4.1.2 Distance d'unicité

On définit la distance d'unicité (U) comme étant la longueur de texte chiffré minimale requise pour pouvoir s'attendre à ce qu'il n'y ait qu'un seul déchiffrement sensé. On utilise la formule suivante, que nous allons démontrer :

$$U = H(K)/D$$

Par cette formule, on remarque que U est inversément proportionnel à la redondance D. Ainsi, plus grande est U, plus faible est la redondance, meilleur est le cryptosystème.

La distance d'unicité indique la longueur minimale de texte chiffré pour laquelle il est probable qu'il n'y ait qu'un seul texte clair plausible correspondant quand une attaque exhaustive est menée.

Des textes chiffrés significativement plus courts que cette distance ont des chances d'avoir plusieurs déchiffrements également valables et donc augmentent la sécurité, car il est difficile à l'attaquant de choisir le bon déchiffrement.

La distance d'unicité n'est pas une mesure de la quantité de texte chiffré qu'il faut pour la cryptanalyse mais bien de la quantité de texte chiffré nécessaire pour qu'il n'y ait qu'une solution raisonnable à la cryptanalyse.

4.1.2.1 Elaboration de la formule de la distance d'unicité

Soient
- $M = M_1...M_n$
- K une permutation de l'alphabet $\rightarrow H(K) = \log(26!)$
- $C = C_1...C_n$

On définit la distance d'unicité comme le plus petit entier d tel que :

$$H(K|(C_1, ..., C_d)) = 0$$

D'autre part, on a :

$$\begin{aligned} H(K|(C_1, ..., C_d)) &= H(K, C_1, ..., C_d) - H(C_1, ..., C_d) \\ &= H(K, M_1, ..., M_d) - H(C_1, ..., C_d) \end{aligned}$$

La valeur de "d" désigne le nombre moyen de lettres du message chiffré qu'il faut connaître pour pouvoir déterminer la clé. Connaître la clé et le message chiffré équivaut à connaître la clé et le message en clair.

D'où :

$$H(K|(C_1, ..., C_d)) = H(K) + H(M_1, ..., M_d) - H(C_1, ..., C_d)$$

Cette égalité provient du fait que la clé est indépendante du message ($H(M, K) = H(K) + H(M)$).

On a donc :

$$H(K) + H(M_1, ..., M_d) - H(C_1, ..., C_d) = 0$$

Par la définition de la redondance d'un langage, on constate que :

$$H(M_1, ...M_d) \approx d.r \text{ et } H(C_1, ...C_d) \approx d.R$$

On obtient dès lors

$$\begin{aligned} H(K) + d.r - d.R = 0 &\Leftrightarrow H(K) = d.(R - r) \\ &\Leftrightarrow d = H(K)/(R - r)(= U) \end{aligned}$$

Par analyse statistique sur de nombreux textes français, on a obtenu une valeur pour $r \approx 3,97$. et $R \approx 4,67$.

On en déduit que

$$U \simeq \frac{H(K)}{R-r} = \frac{\log(26!)}{4.67 - 3.97} \simeq 126$$

En pratique, il faudra effectivement de l'ordre de ce nombre de lettres pour retrouver la clé.

Remarques :

- Nous avons présenté la notion de distance d'unicité au départ des caractères. D'une façon plus générale, on peut parler de textes chiffrés. Dans ce cas, les M_i correspondent à des textes clairs distincts, et C_i aux textes chiffrés correspondants.
- U est donne le nombre de messages suffisants pour pouvoir retrouver la clé. Elle ne donne pas de renseignements concernant la difficulté d'obtention de la clé.

4.2 Complexité en temps et en espace

Cette théorie fournit une méthodologie pour analyser la complexité de calcul de différents algorithmes et techniques cryptographiques. Elle compare les algorithmes et les techniques cryptographiques pour déterminer leur niveau de sécurité.

Cette section exprime un autre point de vue sur le cassage des algorithmes de chiffrement. Jusqu'ici, la théorie de l'information nous apprenait que tous les algorithmes peuvent être cassés. Avec la théorie de la complexité, on apprend s'ils peuvent être cassés avant la fin du monde.

Classe	Complexité	Nombre d'ops pour n=10^6	Temps pour 10^6 ops/sec
Linéaires	O(n)	10^6	1 seconde
Quadratiques	O(n^2)	10^{12}	11,6 jours
Cubiques	O(n^3)	10^{18}	32.000 ans
Exponentiels	O(2^n)	$10^{301\,030}$	$10^{301\,006}$ fois l'age de l'univers

FIG. 4.1 – Temps nécessaire pour "casser" un algorithme selon son ordre.

La complexité des problèmes détermine les temps et espaces minimaux nécessaires pour résoudre l'instance la plus difficile du problème sur un ordinateur théorique (appelée Machine de Turing). Il s'agit d'une machine à états finis avec une mémoire en écriture et en lecture sous la forme d'un ruban infini.

Les problèmes qui peuvent être résolus avec des algorithmes polynomiaux en temps sont appelés solubles, car ils peuvent généralement être résolus en un temps raisonnable. Les problèmes qui ne peuvent être résolus en temps polynomial sont appelés non solubles car calculer leur solution devient vite impossible. Il s'agit souvent des problèmes résolus par des algorithmes exponentiels.

Dans le cadre de ce chapitre, nous citerons deux ensembles de problèmes (appelés *Classes*).
- La classe P, qui contient les problèmes résolus en temps polynomial sur une MT.
- La classe NP, qui contient les problèmes résolus en temps polynomial sur une MTND.

Une MTND (Machine de Turing Non Déterministe) est une variante de la machine de Turing normale qui devine les solutions. La machine devine une solution d'un problème, soit en faisant par chance une bonne hypothèse, soit en essayant toutes les possibilités en parallèle. Elle vérifie son hypothèse en un temps polynomial. Ainsi, la vérification du problème se fait en un temps polynomial, mais le fait d'essayer toutes les possibilités apporte un nombre de cas à explorer très grand, ce qui rend l'algorithme inefficace.

4.2.1 Relations entre P et NP

$NP \supset P$ Car tout problème résoluble en un temps polynomial sur une MT l'est aussi sur un MTND.

$NP \overset{?}{\subset} P$ Beaucoup de problèmes semblent être plus durs que ceux de P, mais on n'est pas encore parvenu à prouver que $NP \not\subset P$

Remarque importante : on n'a pas trouvé d'algorithmes polynomiaux résolvant ces problèmes mais on n'a pas prouvé qu'il n'en existait pas.

4.2.2 Les problèmes NP-Complet

Soit le problème dit « de la satisfaisabilité » (SAT) :

Etant donné une formule booléenne propositionnelle, y a-t-il un moyen d'assigner des valeurs de vérité aux variables de telle manière que la formule soit vraie ?

A l'heure actuelle, on ne sait pas si SAT se situe dans P ou dans NP. On le place donc dans une autre classe, incluse dans NP, appelée Classe NP-Complet. Les problèmes équivalents au problème SAT forment cet ensemble des problèmes de la classe NP-Complet. Cette relation est issue du théorème de Cook.

Théorème 2 (Théorème de Cook) *Tout problème NP peut être réduit au problème de la satisfaisabilité en un temps polynomial.*

□

Objectifs en cryptographie :

- Les méthodes de chiffrement/déchiffrement appartiennent à P.
- Les méthodes de décryptement et cryptanalyse appartiennent à l'ensemble NP-Complet.

4.3 Autres concepts utiles

4.3.1 Résidus quadratiques

Théorème 3 (Résidus quadratiques) *Si p est premier et $a < p$ alors a est un résidu quadratique modulo p si $\lfloor x^2 = a (\mathrm{mod} p) \rfloor$ pour un certain x.*

□

Remarques :

- Une valeur quelconque de a ne satisfait pas toujours cette propriété.
- Il existe (p-1)/2 résidus quadratiques modulo p.
- Si $n = pq$ (p et q premiers), il y a exactement $(p-1)(q-1)/4$ résidus quadratiques modulo n.

Exemple : si p=7, les résidus quadratiques sont 1,2,4 :

$$1^2 = 1 = 1 \bmod 7$$
$$2^2 = 4 = 4 \bmod 7$$
$$3^2 = 9 = 2 \bmod 7$$
$$4^2 = 16 = 2 \bmod 7$$
$$5^2 = 25 = 4 \bmod 7$$
$$6^2 = 36 = 1 \bmod 7$$

Chaque résidu quadratique apparaît 2 fois dans la liste. Mais 3,6 et 5 ne satisfont pas cette équation.

4.3.2 Fonction totient d'Euler

Soit la fonction d'Euler définie par :

$$\phi(m) = \#\{n \leq m | (n, m) = 1\}$$

$\phi(n)$ est l'indicateur d'Euler, c'est-à-dire le cardinal des entiers inversibles de n.

Il faut lire : la fonction ϕ du nombre m a pour résultat un nombre n inférieur ou égal à m et tel que le plus grand commun diviseur de n et m soit 1.

Si n est premier, il vient $\phi(n) = n - 1$. La fonction $\phi(n)$ donne le nombre d'entiers positifs plus petits ou égaux à n relativement premiers à n.

On a ici une propriété remarquable : on peut compter le nombre d'entiers positifs plus petits que m et "relativement premiers" à m, c'est-à-dire :

$$\phi(m) = \sum_{k=1}^{m} 1$$

avec (k,m)=1. Cela n'est qu'une réécriture mathématique de la définition du totient d'Euler pour un nombre premier. Pour autant que k soit relativement premier à m, le totient d'Euler d'un nombre premier m est égal au nombre d'éléments inférieurs à m.

On a donc par exemple $n = 11 \rightarrow \phi(n) = 10$.

De plus, soient p et q deux nombres premiers et $n = pq$. Il vient

$$\begin{aligned} \phi(n) &= \phi(pq) \\ &= \phi(p) * \phi(q) \\ &= (p-1)(q-1) \end{aligned}$$

Démonstration :

$$\phi(n) = \phi(p)\phi(q)$$

Soit l'ensemble des résidus de n (Z_n) : $0, 1, ..., (pq-1)$. Les résidus qui ne sont pas premiers à n sont les ensembles $p, 2p, ..., (q-1)p, q, 2q, ..., (p-1)q$ et 0.

De la sorte :

$$\begin{aligned} \phi(n) &= pq - [(q-1) + (p-1) + 1] \\ &= pq - (p+q) + 1 \\ &= (p-1) * (q-1) \\ &= \phi(p) * \phi(q) \end{aligned}$$

On a par exemple $p = 11, q = 13 \rightarrow n = 143$ et $\phi(n) = 10 * 12 = 120$.

Théorème 4 (Euler) *Soit* $(a, m) = 1$,
alors

$$a^{\phi(m)} = 1 \bmod (m)$$

□

C'est une nouvelle technique pour le calcul de l'inverse modulo : $x = a^{\phi(m)-1} \bmod m$.

Exemple :

Quel est l'inverse de 5 mod 7 ?

Comme 7 est premier, sa fonction totient est $\phi(7) = 7 - 1 = 6$. On a bien $(5, 7) = 1$, d'où $5^5 \bmod 7 = (5^2 * 5^2 * 5) \bmod 7 = 3$.

4.3.3 Théorème des restes chinois

Théorème 5 (Restes chinois) *Prenons* $m_1, ..., m_n$ *des entiers supérieurs à 2 deux à deux premiers entre eux, et* $a_1, ..., a_n$ *des entiers.*

Le système d'équations :

$$x = a_1 \bmod m_1$$

$$...$$

$$x = a_n \bmod m_n$$

admet une unique solution modulo $M = m_1 * ... * m_n$ *donnée par la formule :*

$$x = a_1 M_1 y_1 + ... + a_n M_n y_n \bmod M$$

où $M_i = M/m_i$, *et* $y_i = M_i^{-1} \bmod m_i$ *pour* $i \in [1, n]$.

□

Si on connaît la décomposition en facteurs premier de n, on peut utiliser le CRT (Chinese Remainder Theorem) pour résoudre un système d'équations particulier.

Exemple

Une bande de 17 pirates s'est emparée d'un butin composé de pièces d'or d'égale valeur. Ils décident de se les partager également, et de donner le reste au cuisinier chinois. Celui-ci recevrait alors 3 pièces.

Mais les pirates se querellent, et six d'entre eux sont tués. Le cuisinier recevrait alors 4 pièces.

Dans un naufrage ultérieur, seuls le butin, six pirates et le cuisinier sont sauvés, et le partage donnerait alors 5 pièces d'or à ce dernier.

Quelle est la fortune minimale que peut espérer le cuisinier quand il décide d'empoisonner le reste des pirates ?

Si x est ce nombre, x est le plus petit entier positif tel que :

$$x = 3 \bmod 17.$$

$$x = 4 \bmod 11.$$

$$x = 5 \bmod 6.$$

On applique alors le théorème chinois :

$$M = 17 * 11 * 6 = 1122,$$
$$M_1 = 66,$$
$$M_2 = 102,$$
$$M_3 = 187.$$

L'inverse de chaque M_i donne $y_1 = 8$, $y_2 = 4$, $y_3 = 1$.

On obtient donc : $x = 3 * 66 * 8 + 4 * 102 * 4 + 5 * 187 * 1 \bmod 1122 = 785 \bmod 1122$.

Si le cuisinier tue tous les pirates, il empochera 785 pièces d'or !

4.3.4 Tests de primalité

4.3.4.1 Petit théorème de Fermat

Théorème 6 (Petit théorème de Fermat) *Soit p un nombre premier, (a,p)=1. Alors :*

$$a^{p-1} = 1 \bmod p$$

□

Si n ∈ N est donné, soit a tel que (a,n)=1. On calcule alors a^{n-1}. Si $a^{n-1} \neq 1 \bmod n$ alors n n'est pas premier. Il faut cependant remarquer que cette condition n'est pas nécessaire *et* suffisante.

4.3.4.2 Test de primalité de Solovay-Strassen

1. Choisir un nombre a < p avec p est impair
   - Si $(a, p) \neq 1$, alors p est composé
2. Calculer $j = a^{(p-1)/2} \bmod p$
3. Calculer le Jacobien J(a,p)
   - Si $j \neq J(a, p)$, p n'est pas premier
   - Si $j = J(a, p)$, la probabilité que p soit non premier est de maximum 50%

On répète alors l'opération pour d'autres valeurs de a.

Symbole de Jacobi :

- J(a,n) n'est défini que pour n impair
- J(0,n) = 0
- Si n est premier alors
  - J(a,n) = 0 si n|a
  - J(a,n) = 1 si a est un résidu quadratique modulo n
  - J(a,n) = -1 si a n'est pas un résidu quadratique modulo n
- Si n est un nombre composé alors $J(a, n) = J(a, p_1) * ... * J(a, p_n)$ où $p_1, ..., p_n$ est la décomposition en facteurs premier de n

Propriétés de calculs basés sur le symbole de Jacobi :

- $J(1,n) = 1$
- $J(a*b,n) = J(a,n)*J(b,n)$
- $J(2,n) = 1$ si $(n^2-1)/8$ est paire,-1 sinon
- $J(a,n) = J((a \bmod n),n)$
- $J(a,p_1*p_2) = J(a,p_1)*J(a,p_2)$
- Si $(a,b) = 1$ et a et b sont impairs
  - J(a,b) = +J(b,a) si (a-1)(b-1)/4 est pair
  - J(a,b) = -J(b,a) si (a-1)(b-1)/4 est impair

4.3.4.3 Test de primalité de Miller-Rabin

Si n est un nombre premier, alors on peut écrire

$$n - 1 = 2^s * d$$

avec s entier et d impair.

Le test de Miller-Rabin énonce que n est premier si les deux équations suivantes sont vérifiées :

$$a^d \neq 1 \bmod n$$

$$a^{2^r.d} \neq 1 \bmod n$$

et ce, $\forall r$ dans $[0; s-1]$.

4.4 Ressources supplémentaires

`http://www.bibmath.net/crypto/complements/entropie.php3`
`http://www-igm.univ-mlv.fr/~beal/Enseignement/TheorieInfo/`
`http://mathworld.wolfram.com/topics/NumberTheory.html`

Chapitre 5

Le chiffrement par blocs

5.1 Introduction

La cryptographie symétrique utilise la même clé pour les processus de chiffrement et de déchiffrement ; cette clé est le plus souvent appelée "secrète" (en opposition à "privée") car toute la sécurité de l'ensemble est directement liée au fait que cette clé n'est connue que par l'expéditeur et le destinataire. La cryptographie symétrique est très utilisée et se caractérise par une grande rapidité (opérations simples, chiffrement à la volée) et par des implémentations aussi bien software que hardware ce qui accélère nettement les débits et autorise son utilisation massive.

L'idée générale du chiffrement par blocs est la suivante :

1. Remplacer les caractères par un code binaire
2. Découper cette chaîne en blocs de longueur donnée
3. Chiffrer un bloc en l'"additionnant" bit par bit à une clef.
4. Déplacer certains bits du bloc.
5. Recommencer éventuellement un certain nombre de fois l'opération 3.
6. Passer au bloc suivant et retourner au point 3 jusqu'à ce que tout le message soit chiffré.

On distingue trois catégories de chiffrement par bloc :

- Chiffrement **par substitution** : Les substitutions consistent à *remplacer* des symboles ou des groupes de symboles par d'autres symboles ou groupes de symboles dans le but de créer de la confusion.

- Chiffrement **par transposition** : Les transpositions consistent à *mélanger* les symboles ou les groupes de symboles d'un message clair suivant des règles prédéfinies pour créer de la diffusion. Ces règles sont déterminées par la clé de chiffrement. Une suite de transpositions forme une permutation.

- Chiffrement **par produit** : C'est la combinaison des deux. Le chiffrement par substitution ou par transposition ne fournit pas un haut niveau de sécurité, mais en combinant ces deux transformations, on peut obtenir un chiffrement plus robuste. La plupart des algorithmes à clés symétriques utilisent le chiffrement par produit. On dit qu'un « round » est complété lorsque les deux transformations ont été faites une fois (substitution et transposition).

Ces successions des rondes portent également le nom de *réseaux S-P de Shannon*.

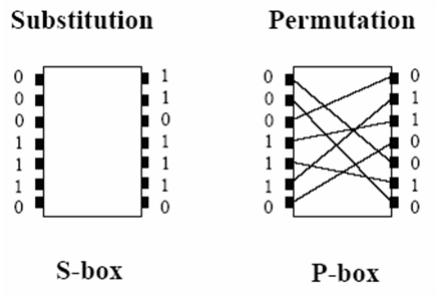

FIG. 5.1 – Substitution et permutation

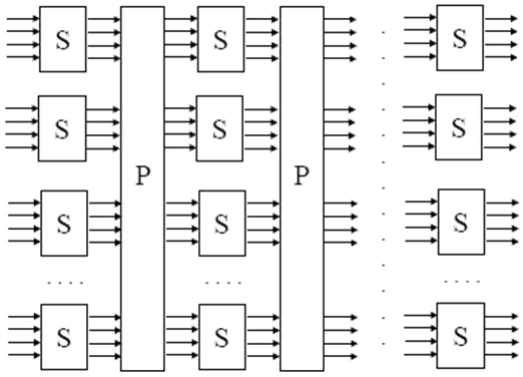

FIG. 5.2 – Succession de rondes

Effet d'avalanche

C'est une propriété des chiffrements par blocs composés de couches (layers) ou "rounds" caractérisés par un petit changement à l'entrée. Le changement d'un simple bit d'entrée produit généralement de multiples changements de bits après un round, plusieurs autres changements de bits après un autre round jusqu'au changement éventuel de la moitié du bloc.

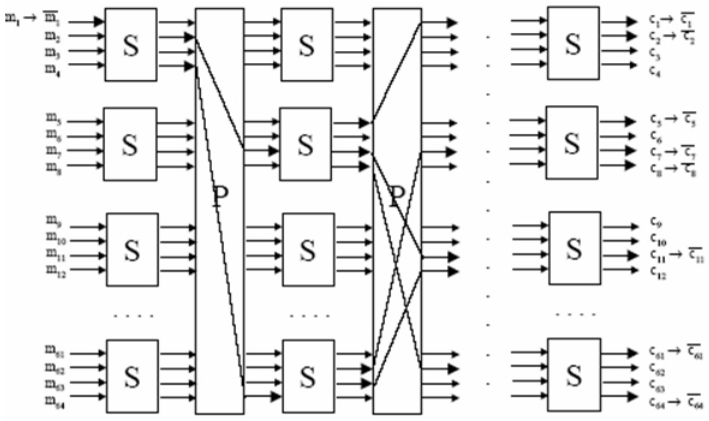

FIG. 5.3 – Effet d'avalanche

5.2 Les structures de Feistel

Cette structure fut décrite en 1973 (par Feistel, employé chez IBM). La plupart des chiffrements de la fin du XXè siècle sont basés sur cette structure. Elle découle des réseaux S-P de Shannon. Il adapte la structure de Shannon afin de la rendre inversible ce qui permet de réutiliser le matériel de chiffrement pour déchiffrer un message. La seule modification s'opère dans la manière dont la clé est utilisée.

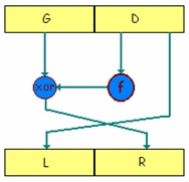

FIG. 5.4 – Structure de Feistel

Dans une construction de Feistel (figure 5.4), le bloc d'entrée d'un round est séparé en deux parties. La fonction de chiffrement est appliquée sur la première partie du bloc et l'opération binaire OU-Exclusif ($\oplus$) est appliquée sur la partie sortante de la fonction et la deuxième partie. Ensuite les deux parties sont permutées et le prochain round commence.

L'avantage est que la fonction de chiffrement et la fonction de déchiffrement sont identiques. Ainsi la fonction n'a pas à être inversible, c'est la structure qui l'est.

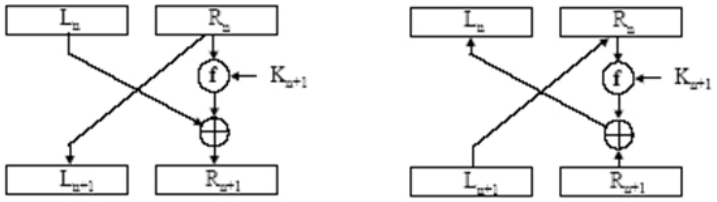

FIG. 5.5 – La structure est inversible

5.2.1 Exemple

A partir d'une table de correspondance, on peut déterminer le résultat du chiffrement d'un bloc après passage dans une structure de Feistel. C'est ce qui est illustré aux figures 5.6 et 5.7

entrée	f_1	sortie
00	→	01
01	→	11
10	→	10
11	→	01

entrée	f_2	sortie
00	→	11
01	→	00
10	→	00
11	→	01

FIG. 5.6 – Table de correspondance de fonctions

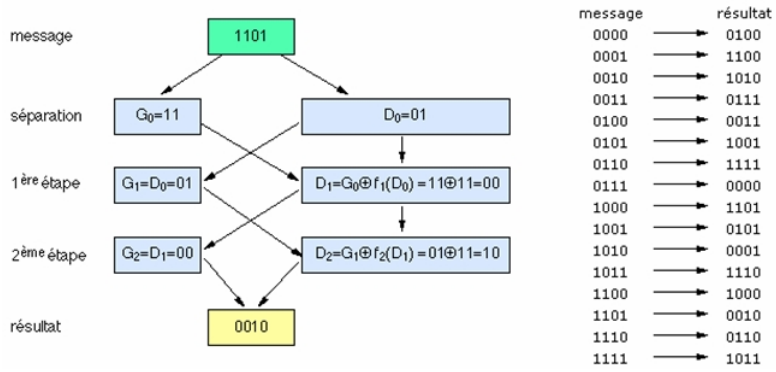

FIG. 5.7 – Exécution d'un schéma de Feistel

5.2.2 Choix des paramètres

La réalisation d'un tel réseau dépend des choix effectués pour les paramètres suivants :

- Taille du bloc : si elle augmente, la sécurité augmente également
- Taille de clé : si elle augmente, la sécurité aussi
- Nombre de cycle : plus il y en a, plus la sécurité est renforcée
- Algorithme de génération des sous-clés : plus il est complexe, plus la compréhension est rendue difficile.

5.3 D.E.S. - Data Encryption Standard

5.3.1 Présentation

Le D.E.S. (Data Encryption Standard, c'est-à-dire Standard de Chiffrement de Données) est un standard mondial depuis la fin des années 1970.

Au début de cette décennie, le développement des communications entre ordinateurs a nécessité la mise en place d'un standard de chiffrement de données pour limiter la prolifération d'algorithmes différents ne pouvant pas communiquer entre eux. Pour résoudre ce problème, L'Agence Nationale de Sécurité américaine (N.S.A.) a lancé des appels d'offres. La société I.B.M. a développé alors un algorithme nommé Lucifer, relativement complexe et sophistiqué. Après quelques années de discussions et de modifications (applications de S-Boxes et réduction à des clés de 56 bits), cet algorithme, devenu alors D.E.S., fut adopté au niveau fédéral le 23 novembre 1976.

Particularités

Le DES comporte plusieurs avantages qui en ont fait l'algorithme de chiffrement symétrique standard pendant longtemps, jusqu'il y a quelques années. En voici quelques-uns :

- il possède un haut niveau de sécurité,
- il est complètement spécifié et facile à comprendre,
- la sécurité est indépendante de l'algorithme lui-même,
- il est rendu disponible à tous, par le fait qu'il est public,
- il est adaptable à diverses applications (logicielles et matérielles),
- il est rapide et exportable,
- il repose sur une clé relativement petite, qui sert à la fois au chiffrement et au déchiffrement,
- il est facile à implémenter.

5.3.2 Algorithme de chiffrement

Le D.E.S. est un cryptosystème agissant par blocs. Cela signifie que D.E.S. ne chiffre pas les données à la volée quand les caractères arrivent, mais il découpe virtuellement le texte clair en blocs de 64 bits qu'il code séparément, puis qu'il concatène. Un bloc de 64 bits du texte clair entre par un coté de l'algorithme et un bloc de 64 bits de texte chiffré sort de l'autre coté. L'algorithme est assez simple puisqu'il ne combine en fait que des permutations et des substitutions.

C'est un algorithme de chiffrement à clef secrète. La clef sert donc à la fois à chiffrer et à déchiffrer le message. Cette clef a ici une longueur de 64 bits, c'est-à-dire 8 caractères, mais dont seulement 56 bits sont utilisés. On peut donc éventuellement imaginer un programme testant l'intégrité de la clef en exploitant ces bits inutilisés comme bits de contrôle de parité.

L'entière sécurité de l'algorithme repose sur les clefs puisque l'algorithme est parfaitement connu de tous. La clef de 64 bits est utilisée pour générer 16 autres clefs de 48 bits chacune qu'on utilisera lors de chacune des 16 itérations du D.E.S.. Ces clefs sont les mêmes quel que soit le bloc qu'on code dans un message.

Cet algorithme est relativement facile à réaliser matériellement et certaines puces chiffrent jusqu'à 1 Go de données par seconde. Pour les industriels, c'est un point important notamment face à des algorithmes asymétriques, plus lents, tels que l'algorithme R.S.A.

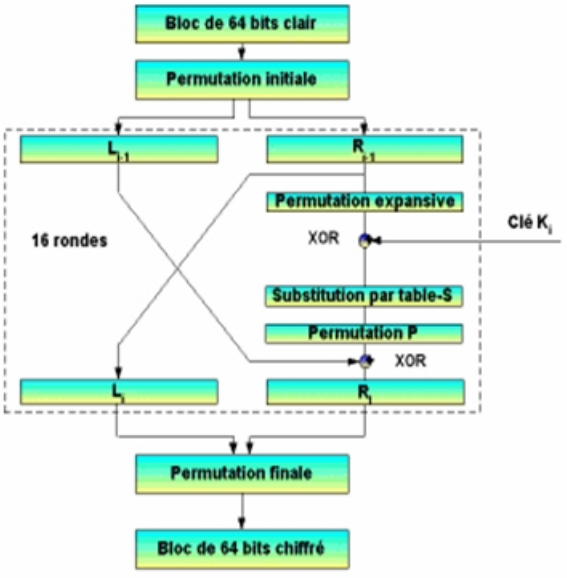

Fig. 5.8 – Algorithme principal du DES

L'algorithme repose principalement sur 3 étapes, en plus de la gestion spécifique de la clé :

1. Permutation initiale
2. Calcul médian (16 fois) : application d'un algorithme complexe appliqué en fonction de la clé
3. Permutation finale

5.3.2.1 La permutation initiale

Les 64 bits du bloc d'entrée subissent la permutation de la figure 5.9

IP

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

FIG. 5.9 – Matrice de permutations initiale

Cette "matrice" permet d'effectuer des changements internes au bloc (i.e. il n'y a pas d'apport de données extérieures). Le premier bit sera le bit 58, le second le bit 50, etc.

5.3.2.2 Le calcul médian

Les 64 bits initiaux de données sont divisés en 2 blocs (L et R).

Itérations :

- $L_n = R_{n-1}$
- $R_n = L_{n-1} \oplus F(R_{n-1}, K_n)$
- $K_n = G(K, n)$

avec

- $T_n = L_n R_n$
- $L_n = t_1 ... t_{32}$
- $R_n = t_{33} ... t_{64}$

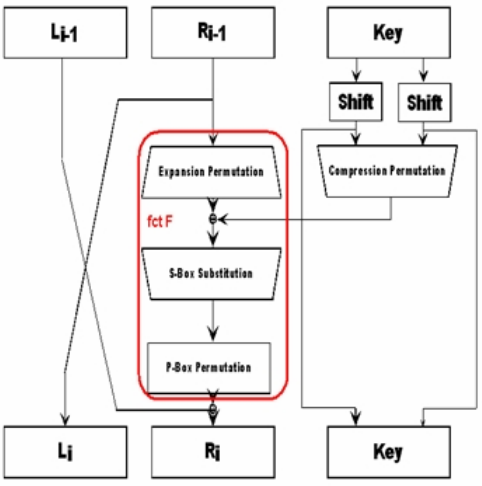

FIG. 5.10 – Etape générale du calcul médian

Le calcul médian s'effectue en 16 itérations. Le détail de la fonction F est donné à la figure 5.11. On traite 2 blocs simultanément : un bloc de 32 bits (données) et un bloc de 48 bits (clés). Le résultat forme un bloc de 32 bits.

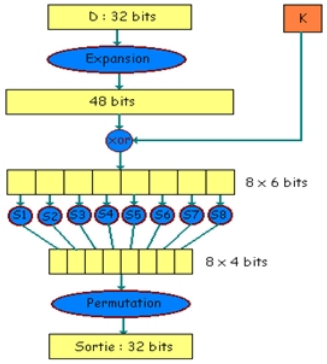

FIG. 5.11 – Fonction F détaillée

Expansion : Les 32 bits sont étendus à 48 bits grâce à une table d'expansion (également appelée matrice d'extension). On retrouve ici un effet d'avalanche.

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

FIG. 5.12 – Matrice d'expansion

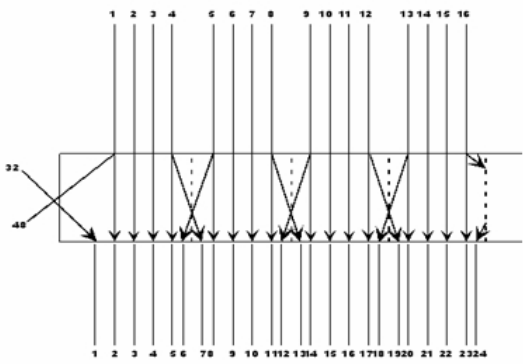

FIG. 5.13 – Phase d'expansion

Addition de la sous-clé : Le résultat de l'expansion est additionné (par une opération $\oplus$) à la sous-clé K_n correspondant à l'itération selon la formule :

$$E(R_{i-1}) \oplus K_i = B_1 B_2 \dots B_8$$

Les $B_1, B_2, ..., B_8$ sont des blocs de 6 bits :

$$B_j = b_1 b_2 b_3 b_4 b_5 b_6.$$

Transformations par S-Boxes : Chaque bloc B_j constitue ensuite l'entrée de l'opération de substitution réalisée sur base des S-Box.

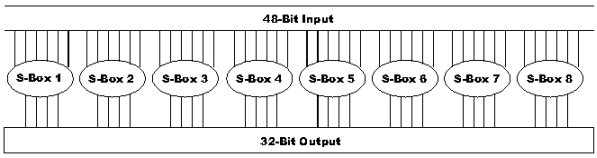

FIG. 5.14 – Transformations S-Box

L'opération de substitution consiste pour chaque S-box à calculer :

- $b_1\ b_6$ = n° de ligne
- $b_2\ b_3\ b_4\ b_5$ = n° de colonne

N° de colonne

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

N° de ligne

FIG. 5.15 – S-Box particulière

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
0	15	1	2	14	6	11	3	4	9	7	2	13	2	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	1	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	122	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

FIG. 5.16 – Les 8 S-Box du DES

Transformations par P-Box (permutation du calcul médian) : L'opération de permutation est réalisée sur le résultat de la substitution des S-box et est basée sur la table de la figure 5.17.

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

FIG. 5.17 – Matrice de permutation du calcul médian

Le résultat de cette dernière permutation est noté $F(R_{n-1}, K_n)$.

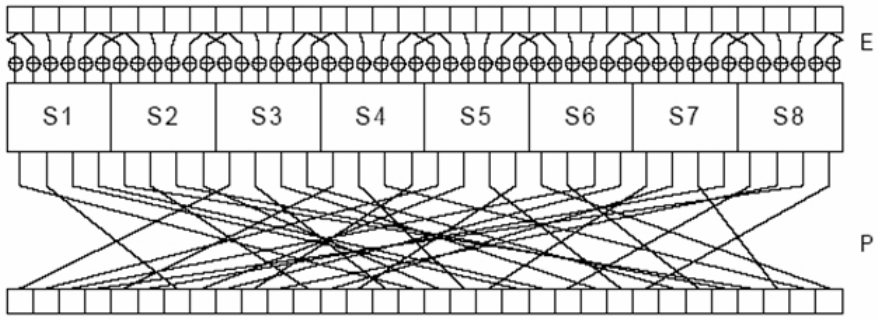

FIG. 5.18 – Ronde détaillée du calcul médian

5.3.2.3 Permutation finale

Une fois le calcul médian terminé, on pratique la permutation inverse de la permutation initiale. Attention toutefois : il s'agit de l'inverse de la permutation initiale, en d'autres termes, cette table permet de retrouver la position de départ. Ce n'est pas l'inverse de la "matrice" de départ !

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

FIG. 5.19 – Permutation finale

5.3.3 Algorithme du calcul de la clé $G(K, n)$

La clé est constituée de 64 bits dont 56 sont utilisés dans l'algorithme. Les 8 autres peuvent être utilisés pour la détection d'erreurs où chacun de ces bits sera utilisé comme bit de parité des 7 groupes de 8 bits. Ainsi, le nombre total de clés est de 2^{56}.

La clé initiale est de 64 bits. Le calcul a lieu en 4 étapes :

1. Réduction à 56 bits : les bits de parité sont enlevés. On procède ensuite à une permutation semblable à celle de la figure 5.20.

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

FIG. 5.20 – Matrice de réduction de la clé

2. Division en sous-clés de 28 bits : le résultat de l'étape précédente (56 bits) est scindé en deux sous-clés de 28 bits.
3. Rotation de la clé : à chaque itération, chaque sous-clé de 28 bits subit une rotation d'1 ou 2 bits vers la gauche selon la table de la figure 5.21,

Iteration i	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

FIG. 5.21 – Rotation de la clé

4. Réduction : après concaténation des deux sous-clés précédentes, la clé résultante (56 bits) est réduite à une sous-clé de 48 bits sur base de la matrice de la figure 5.22,

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

FIG. 5.22 – Matrice de réduction de la clé

Le résultat de cette réduction est la sous-clé K_n additionnée avec $E(R_{n-1})$.

5.3.4 Déchiffrement

Il suffit d'appliquer le même algorithme mais inversé en tenant bien compte du fait que chaque itération du déchiffrement traite les mêmes paires de blocs utilisés dans le chiffrement.

Il viendra

$$R_{n-1} = L_n \; et \; L_{n-1} = R_n \oplus f(L_n, K_n)$$

5.4 Faiblesses du D.E.S. et évolutions

5.4.1 Problème des compléments

Si $c = DES(p, k)$, alors $!c = DES(!p, !k)$ mais cela ne représente pas un problème sérieux.

5.4.2 Les clefs faibles

- Clefs « faibles » : telles que $E_k(E_k(x)) = x$. Il en existe 4.
- Clefs « semi-faibles » : ce sont les paires de clefs (K_1, K_2) dont la deuxième peut décrypter un message encrypté par la première. Ce sont les clés telles que $E_{K_1}(E_{K_2}(x)) = x$. Il existe six paires de ce genre.
- Clés « pouvant être faibles » : le problème est similaire aux clés semi-faibles. Il en existe 48.

5.4.3 Attaques sur le DES

Plusieurs attaques existent pour casser le DES :

- La recherche exhaustive par force brute : on teste toutes les clés possibles, l'une après l'autre, afin de déchiffrer un bloc de données. On a besoin, en moyenne, de 2^{55} essais.
  - Une machine dédiée au calcul des clés : Deep Crack (Des Cracker, élaboré par l'Electronic Frontier Foundation) a coûté moins de 250'000 dollars. Elle disposait de 1850 processeurs en parallèle pour un temps de recherche de 56 heures (1998).
  - Le calcul distribué : un regroupement d'ordinateurs par Internet qui partagent leur puissance de calcul. En 1998, Distributed.net a pu décrypter un message en 39 jours[1]. Le 19 janvier 1999, EFF et Distributed.net ont cassé en travaillant conjointement une clé en 22 heures et 15 minutes.
  - Depuis 2006, un autre ordinateur dédié, nommé COPACABANA[2], permet de casser une clé DES en 9 jours. En 2008, des améliorations logicielles ont même permis de diminuer le temps de recherche à 6,4 jours. Son avantage est son coût : 10.000$. On peut donc facilement en acquérir plusieurs et donc diminuer le temps de recherche en conséquence. Par exemple, pour 4 machines acquises (=40,000$), il faudra compter 1,6 jour de recherche.
- Cryptanalyse différentielle : il s'agit de la possibilité de produire au moyen de textes clairs choisis les textes chiffrés correspondants avec une clé inconnue. On analyse les différences résultantes sur les textes chiffrés et on donne des probabilités aux clés testées. En affinant, on trouve la clé la plus probable. La meilleure attaque différentielle connue demande 2^{47} textes clairs choisis
- Cryptanalyse linéaire : on dispose d'un dictionnaire de (M,C). On ne possède pas la « boite noire » comme dans le cas différentiel. L'objectif est de déterminer une équation linéaire modélisant au mieux les transformations du cryptogramme créé par l'algorithme de chiffrement.
- Il existe d'autres attaques spécifiques au DES (Davie's attack, ou des attaques sur des versions simplifiées du DES) mais qui n'entrent pas dans le cadre de ce cours.

5.4.4 2DES

Suite aux failles du DES, quelques modifications ont été apportées, mais pas toujours avec succès. Ce fut notamment le cas avec le 2DES.

Il faut tout d'abord choisir deux clefs k_1 et k_2. Le principe du 2DES est de chiffrer deux fois le message :

$$E(k_2, E(k_1, m))$$

Il a été prouvé que 2DES était équivalent à un DES avec une clé de 57 bits. Il faut donc seulement deux fois plus de travail pour le briser ($2^{57} = 2 * 2^{56}$).

[1] http ://lists.distributed.net/hypermail/announce/0039.html
[2] http ://www.copacobana.org

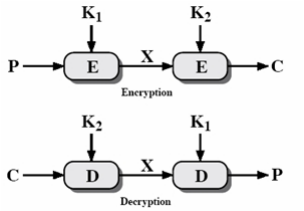

FIG. 5.23 – Double DES

2DES - Attaque Meet-in-the-middle

Le 2DES est sensible à l'attaque Meet-in-the-middle, autrement dit, un intrus peut s'introduire dans l'échange et retrouver la clé utilisée. Imaginons la situation suivante :

Alice transmet $C = f_2(f_1(M))$ à Bob où $f_1 = K_1$ et $f_2 = K_2$.

Si Jack connaît M et C, il peut construire deux listes de 2^{56} messages :

$$L_1 = \{f_K(M);\ \forall K\}\ et\ L_2 = \{f_K^{-1}(C);\ \forall K\}$$

Il cherche ensuite un élément commun.

Si $R = f_{K_3}(M) = f_{K_4}^{-1}(C)$, c'est que $f_{K4}(f_{K_3}(M)) = C$.

Jack a alors probablement retrouvé

$$K_3 = K_1\ et\ K_4 = K_2$$

Ainsi, l'attaque nécessite $X = 2^n$ opérations, et $Y = 2^n$ opérations, ce qui revient à $2 * 2^n = 2^{n+1}$.

5.4.5 3DES

Grâce à 2 clefs, on pratique 3 opérations :

$$E(k_1, D(k_2, E(k_1, m))).$$

C'est équivalent au fait de doubler la taille effective de la clé (ce qui est une longueur sûre actuellement). Il existe deux versions : la première utilise deux clés (cas de figure présenté ici), la seconde trois (le dernier chiffrement utilise une troisième clé).

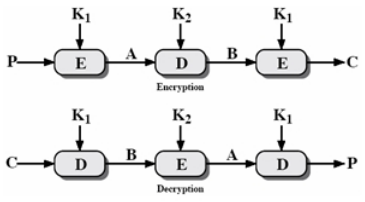

FIG. 5.24 – Triple DES

Il est très robuste contre toutes les attaques faisables connues. Cependant, il est beaucoup plus lent que le DES car on triple les opérations.

5.5 A.E.S. - Advanced Encryption Standard

5.5.1 Les résultats d'un concours

La progression de la puissance des ordinateurs a causé la mort du DES. Ce dernier n'est plus jamais utilisé lorsque la sécurité demandée est forte (utilisation militaire, documents "secrets", etc.). Pour cette tâche, on préfère utiliser l'algorithme connu sous le nom générique d'AES (Advanced Encryption Standard), issu d'un concours créé en raison des faiblesses avérées du DES. Le véritable nom de l'AES est le Rijndael, nom résultant de la contraction des noms de ses inventeurs : Rijmen et Deamen.

Le Triple DES demeure toutefois une norme acceptée pour les documents gouvernementaux aux U.S.A. Pour l'instant, il n'y a pas de projet ou d'obligation de rechiffrer les documents existants.

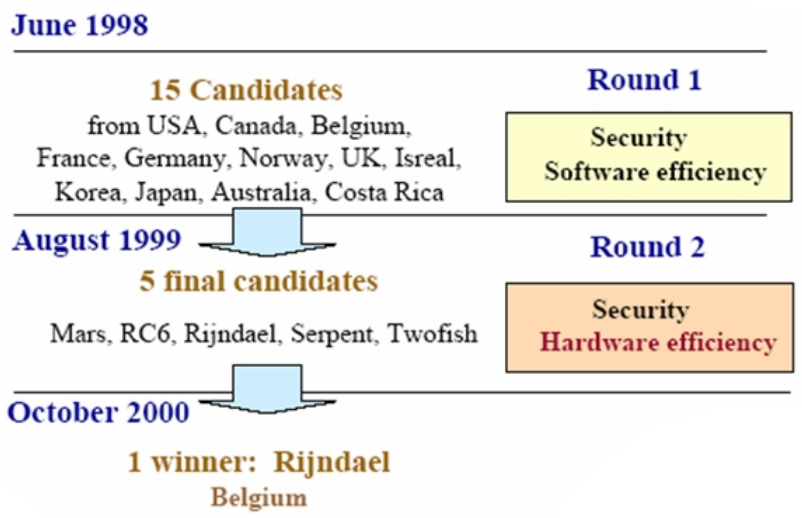

FIG. 5.25 – Différents participants au concours AES

Cahier des charges

Au second tour du concours, les jurys devaient juger différents critères :

- La sécurité générale,
- Le coût en terme de calculs (rapidité),
- La simplicité de l'algorithme et ses facilités d'implémentation,
- Une lecture facile de l'algorithme, puisqu'il est destiné à être rendu public,
- La résistance aux attaques connues,
- Flexibilité - Portabilité : l'algorithme devant remplacer le DES, il est destiné à servir aussi bien dans les cartes à puces, aux processeurs 8 bits peu puissants, que dans des processeurs spécialisés pour chiffrer des milliers de télécommunications à la volée.
- Techniquement, le chiffrement doit se faire par blocs de 128 bits, les clés comportant 128, 192 ou 256 bits.

Au niveau du chiffrement/déchiffrement, les résultats varient assez fortement. Cependant, Serpent reste le moins bon pour la majorité des plate-formes, Rijndael et RC6 étant les meilleurs.

5.5.2 Le choix : Rijndael

A la suite de nombreux tests, c'est finalement Rijndael qui a remporté la médaille, et est ainsi devenu le remplaçant officiel du DES.

Il possède les propriétés suivantes :
- Plusieurs longueurs de clef et de bloc sont possibles : 128, 192, ou 256 bits ;
- Le nombre de cycles ("rondes") varie en fonction de la longueur des blocs et des clés (de 10 à 14) ;
- La structure générale ne comprend qu'une série de transformations/permutations/sélections ;
- Il est beaucoup plus performant que le DES ;
- Il est facilement adaptable à des processeurs de 8 ou de 64 bits ;
- Le parallélisme peut être implémenté

À chaque ronde, quatre transformations sont appliquées :

1. substitution d'octets dans le tableau d'état
2. décalage de rangées dans le tableau d'état
3. déplacement de colonnes dans le tableau d'état (sauf à la dernière ronde)
4. addition d'une "clef de ronde" qui varie à chaque ronde

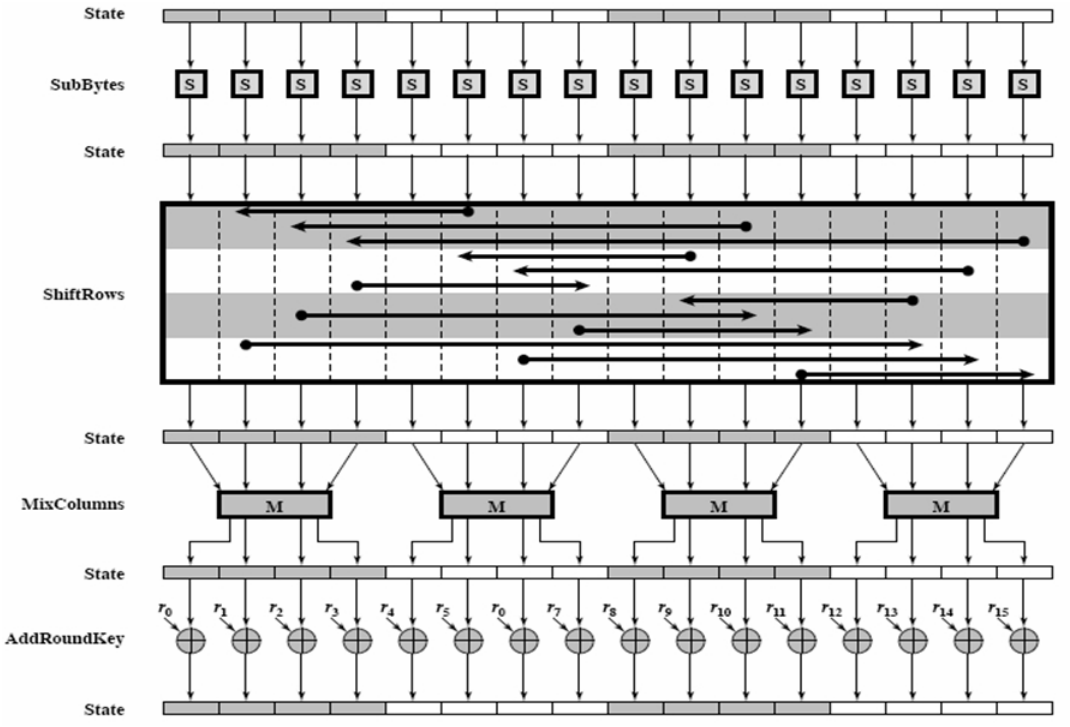

FIG. 5.26 – Schéma général de Rijndael

5.5.2.1 Chiffrement et Déchiffrement

L'ordonnancement des étapes est illustré à la figure 5.27.

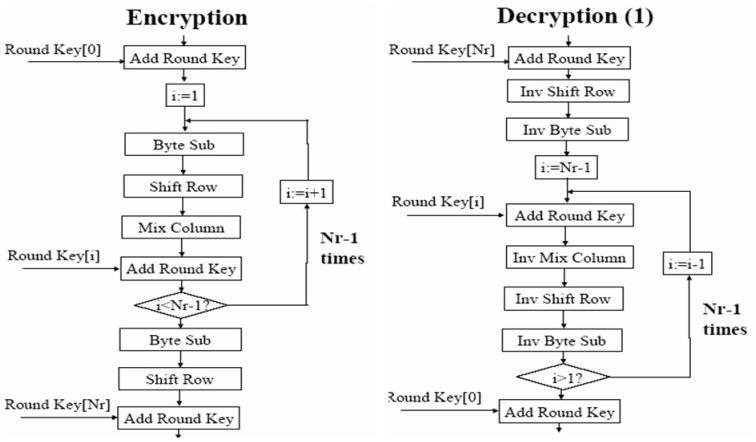

FIG. 5.27 – Schéma des différentes étapes

Table d'état du texte et des clés

Le message et la clé sont conservés sous forme de tables représentées respectivement aux figures 5.28 et 5.29. Le nombre de colonnes dépend des tailles des textes et clés.

$$Nb = L_{bloc}/32$$

$$Nk = L_{clef}/32$$

Une colonne du tableau correspond à un mot de 32 bits. Ainsi, chaque petit bloc représente 8 bits, donc 1 octet. L'input et l'output sont donc gérés comme des séquences linéaires d'octets.

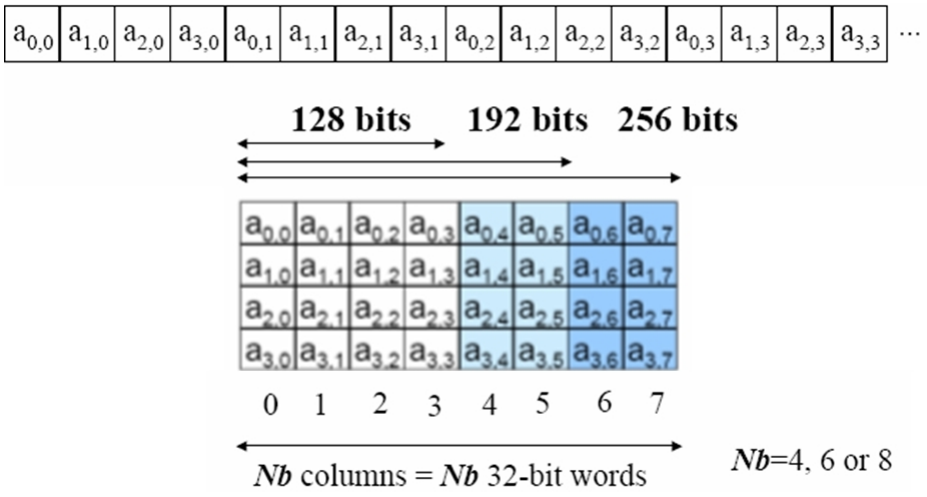

FIG. 5.28 – Table d'état du texte

SubByte

Les octets sont transformés en appliquant une S-Box inversible (afin de permettre un déchiffrement unique). Une seule S-Box est suffisante pour toute la phase de chiffrement.

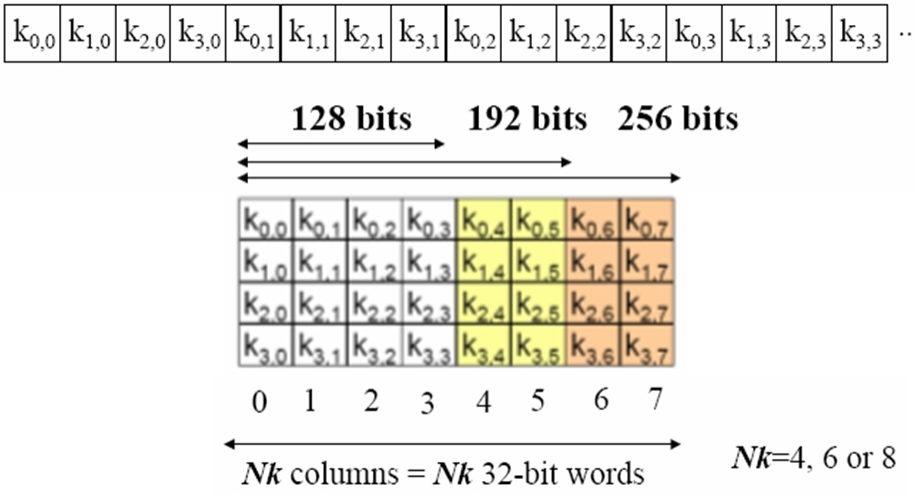

FIG. 5.29 – Table d'état des clés

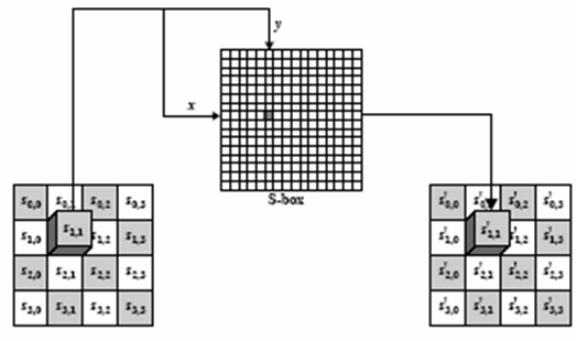

FIG. 5.30 – Fonction SubByte

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

FIG. 5.31 – S-Box inversible

ShiftRow

Cette étape augmente la diffusion dans la ronde.

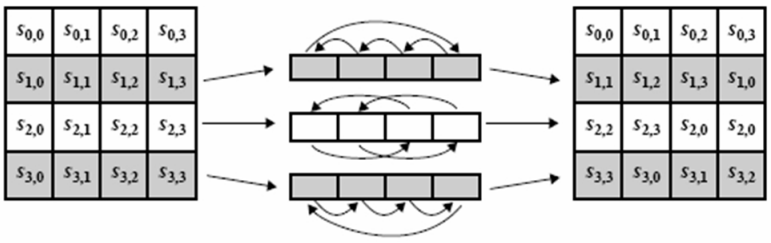

FIG. 5.32 – Schéma de l'étape ShiftRow

Selon la taille des blocs de message (c'est-à-dire la valeur de Nb), les décalages ne seront pas toujours identiques.

- La ligne 0 n'est jamais décalée,
- La ligne 1 est décalée de C_1,
- La ligne 2 est décalée de C_2,
- La ligne 3 est décalée de C_3.

	C_1	C_2	C_3
N_B=4	1	2	3
N_B=6	1	2	3
N_B=8	1	3	4

FIG. 5.33 – Décalage selon la taille des blocs de messages.

MixColumn

Une différence sur 1 byte d'entrée se propage sur les 4 bytes de sortie. On a donc encore une étape de diffusion. La matrice utilisée est définie par Rijndael. Elle contiendra toujours ces valeurs.

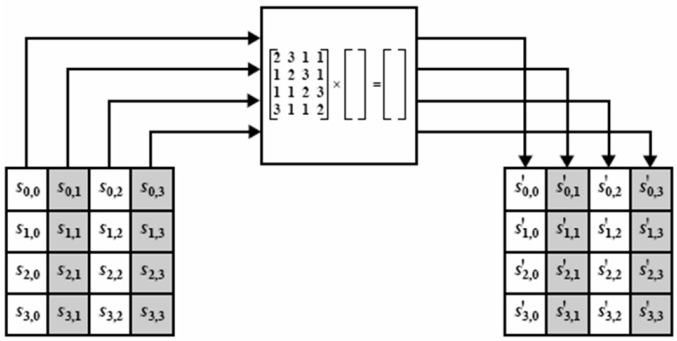

FIG. 5.34 – Etape du MixColumn

Add Round Key

C'est un simple $\oplus$ des clés. Il s'agit d'additionner des sous-clés aux sous-blocs correspondants.

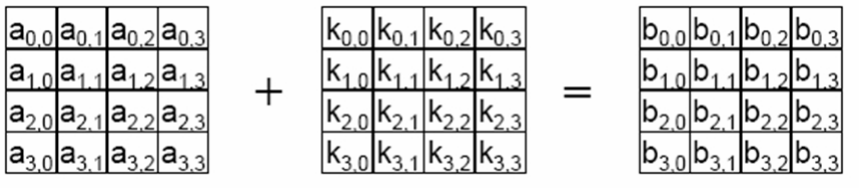

FIG. 5.35 – AddRound Key

Nombre de rondes

Selon la taille des blocs à traiter et la taille de la clé, le nombre de rondes évolue.

	Key length		
Block length	**128 bits Nk=4**	**192 bits Nk=6**	**256 bits Nk=8**
128 bits Nb=4	**10**	**12**	**14**
192 bits Nb=6	**12**	**12**	**14**
256 bits Nb=8	**14**	**14**	**14**

FIG. 5.36 – Nombres de rondes à effectuer

5.5.2.2 Calcul de la clé

Après avoir subi une extension (*Key Expansion*), la clé sera découpée en sous-clés (appelées clés de rondes), comme indiqué à la figure 5.37.

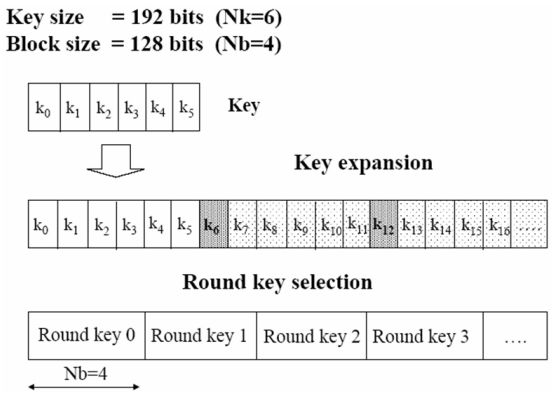

FIG. 5.37 – Schéma des opérations effectuées sur la clé

Le nombre de sous-blocs k_i dépendra bien sûr de la taille des clés et bloc du message.

Extension de la clé

Le calcul de l'expansion de la clé se fait de deux manières distinctes selon le sous-bloc de la clé concerné, comme l'illustrent les figures 5.38 et 5.39.

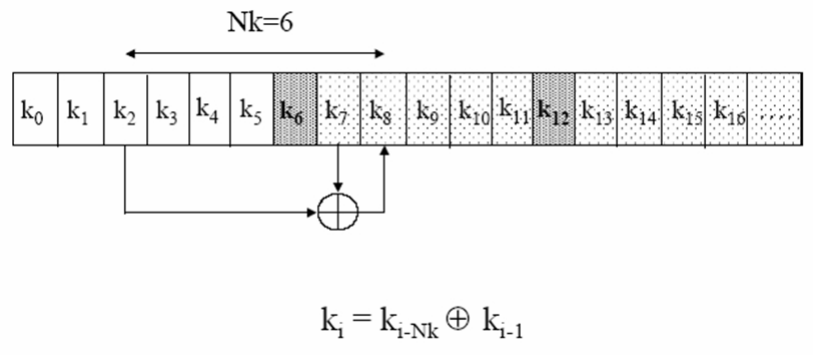

FIG. 5.38 – Expansion de la clé avec bloc "commun"

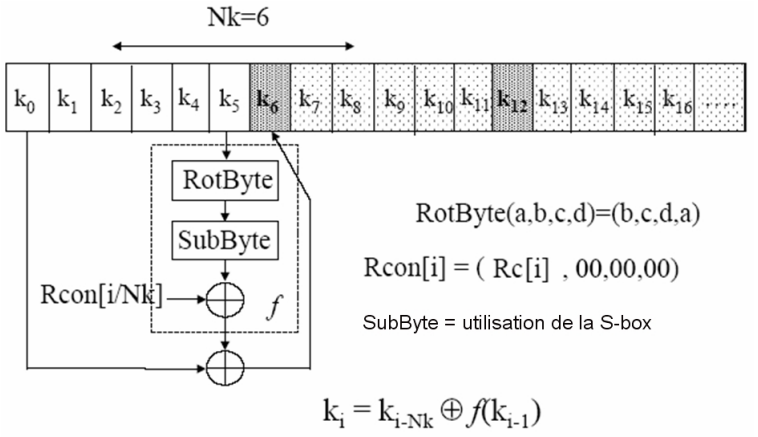

FIG. 5.39 – Expansion de la clé avec les blocs "multiples de N_k"

Remarques concernant la figure 5.39 :

L'ajout de "Rcon[x]" donne comme résultat un $\oplus$ sur les bits les plus significatifs. La table utilisée pour donner les valeurs de Rcon[] est donnée à la figure 5.40.

j	1	2	3	4	5	6	7	8	9	...
RC[j]	01	02	04	08	10	20	40	80	1B	...

FIG. 5.40 – Table de correspondance des Rcon[]

La règle de construction[3] de cette table est

$$RC[1] = 1$$
$$RC[j] = 2.RC[j-1]$$

5.5.3 Avantages et limites

Les principaux avantages sont :

- des performances très élevées,
- la possibilité de réalisation en "Smart Card" avec peu de code,
- la possibilité de parallélisme,
- il ne comprend pas d'opérations arithmétiques : ce sont uniquement des décalages et des XOR,
- il n'utilise pas de composants d'autres cryptosystèmes,
- il n'est pas fondé sur des relations obscures entres opérations,
- le nombre de rondes peut facilement être augmenté si c'est requis,
- il ne possède pas de clés faibles,
- il est résistant à la cryptanalyse différentielle et linéaire.

Il possède pourtant quelques inconvénients et limites :

- le code et les tables sont différents pour le chiffrement et déchiffrement,
- le déchiffrement est plus difficile à implanter en "Smart Card",
- dans une réalisation matérielle, il y a peu de réutilisation des circuits de chiffrement pour effectuer le déchiffrement.

5.5.4 Attaques

L'AES est sensible à certaines attaques par canal auxiliaire, n'impliquant donc pas directement l'algorithme.

Toutefois, en 2009, un article[4] met en évidence des problèmes de sécurité sur des versions allégées de l'AES-256. Bien que non applicables à l'algorithme standard (sur 128 bits), il est fort probable qu'à moyen terme, l'AES soit mis à mal.

[3] Il est à noter que la multiplication est définie sur le champ fini $GF(2^8)$.
[4] http ://eprint.iacr.org/2009/374

5.6 Modes de chiffrement symétrique

Les modes sont des méthodes pour utiliser les chiffrements par blocs. On parle de modes opératoires. Dans le cadre d'une implémentation pratique, l'algorithme 'pur' est combiné à une série d'opérations simples en vue d'améliorer la sécurité sans pour autant pénaliser l'efficacité de l'algorithme. Cette combinaison est appelée un mode cryptographique.

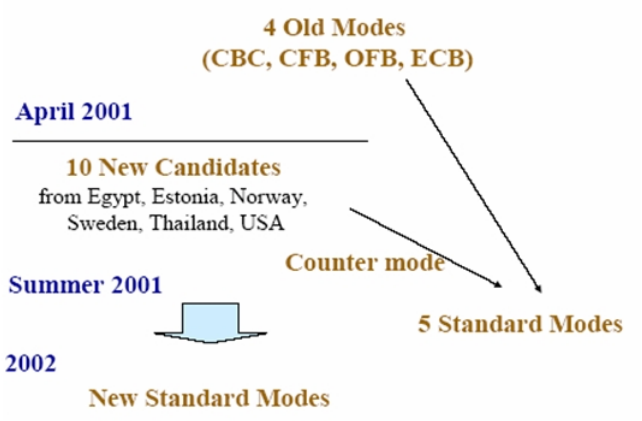

FIG. 5.41 – Apparition des modes cryptographiques

L'utilisation de ces modes repose sur différents critères :

- Sécurité :
  - Effacement des formats standards (ex. l'introduction d'un texte).
  - Protection contre la modification de C.
  - Chiffrement de plusieurs messages avec la même clé.
- Efficacité :
  - L'utilisation d'un mode cryptographique ne doit pas pénaliser l'efficacité du cryptosystème.
  - Limitation de la propagation des erreurs qui apparaissent dans M ou C.

5.6.1 ECB - Electronic Code Book

Les blocs sont chiffrés indépendamment bloc par bloc. Formellement, il vient

$$C_i = DES_K(P_i).$$

Son usage est limité à la transmission sûre de valeurs isolées.

5.6.1.1 Sécurité :

+/- indépendance des blocs
- conservation des formats $m = m' \Rightarrow c = c'$
- clé non-réutilisable

5.6.1.2 Efficacité :

+ parallélisme possible
+ accès aléatoire possible
+ bonne vitesse de chiffrement
- pas de préprocessing

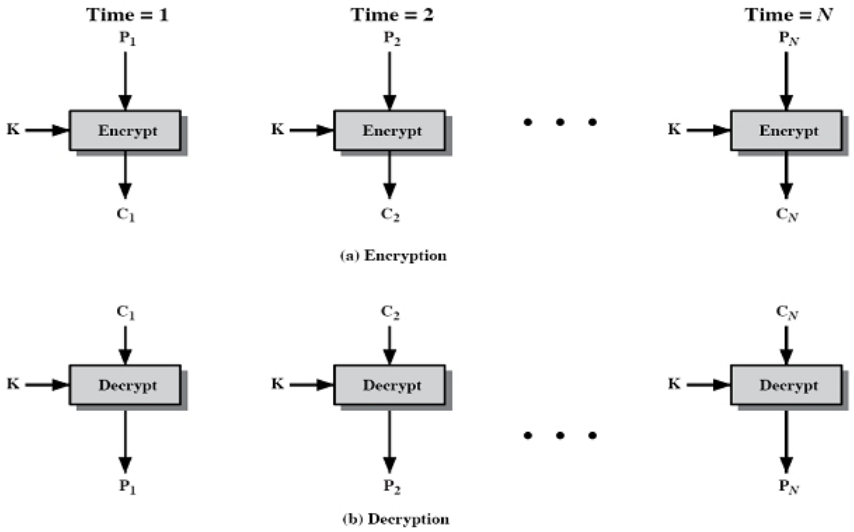

FIG. 5.42 – Mode ECB

5.6.1.3 Propagation des erreurs :

+ une erreur dans $m_i(c_i)$ n'affecte que le $c_i(m_i)$ correspondant
+ la perte d'un bloc c_i n'affecte qu'un bloc de M
- la perte ou l'ajout d'un bit de c_i affecte tous les blocs suivants $(m_i, m_{i+1}, ...)$
- Une erreur dans C_i affecte un bloc M_i entier

5.6.2 CBC - Cipher Block Chaining

Les blocs sont liées entre eux, d'où le terme "chaînage". Il y a donc un effet d'avalanche (un c_i est dépendant de m_i et c_{i-1}). Le vecteur d'initialisation (IV) peut être un mot de passe, un timestamp (marqueur temporel), etc. Il est utilisé dans certaines phases d'authentification.

5.6.2.1 Sécurité :

+ IV unique et aléatoire
+ effacement des formats grâce à l'enchaînement (si IV différents)
+ il n'y a plus de risque de répétition de bloc
+ clé réutilisable (si IV différents)
+ si $m_i = m'_i$, alors $c_i \neq c'_i$ (avec même K et IV différents)

5.6.2.2 Efficacité :

+ même vitesse de chiffrement que ECB
- pas de préprocessing
- pas de parallélisme
- IV connu des 2 cotés (problème pour sa communication)

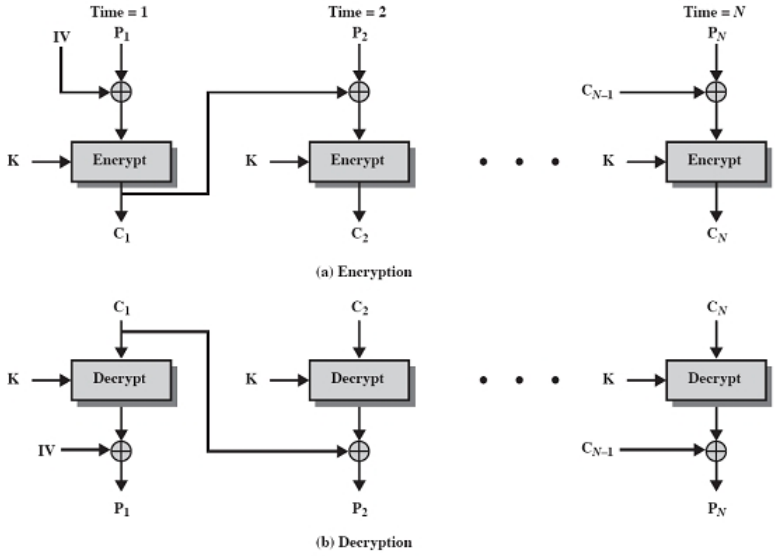

FIG. 5.43 – Mode CBC

5.6.2.3 Propagation des erreurs :

+ la perte d'un c_i n'affecte que deux blocs de M

Considérons que le bloc C_1 soit perdu. Les premiers blocs seront construits par les relations suivantes :

$$M_1 = D_K(C_2) \oplus IV$$
$$M_2 = D_K(C_3) \oplus C_2 \ (= P_3)$$
$$M_3 = D_K(C_4) \oplus C_3 \ (= P_4)$$

Au final, seuls les blocs P_1 et P_2 seront perdus.

+ une erreur dans m_i affecte tous les c_i suivants mais ne se retrouve que dans le m_i correspondant
- une erreur dans c_i affecte un bloc entier de m_i et le bit correspondant dans m_{i+1}
- la perte ou l'ajout d'un bit de c_i affecte tous les blocs $m_i, m_{i+1}, ...$ suivants (perte des limites de bloc)

5.6.3 CFB - Cipher FeedBack

Le message est ajouté à la sortie du bloc chiffré. Le résultat sert de feedback pour l'étape suivante. Le registre peut utiliser un nombre quelconque de bits : 1, 8, 64 bits (le plus souvent 64). Il est utilisé pour le chiffrement par flux ainsi que pour l'authentification.

5.6.3.1 Sécurité :

+ même fonction pour le chiffrement et le déchiffrement
+ pas de répétition de bloc si IV différents
+ effacement du format standard

5.6.3.2 Efficacité :

+ même vitesse de chiffrement que ECB
+ CFB peut débuter le chiffrement après réception de sous-blocs ⇒ utilisation réseaux

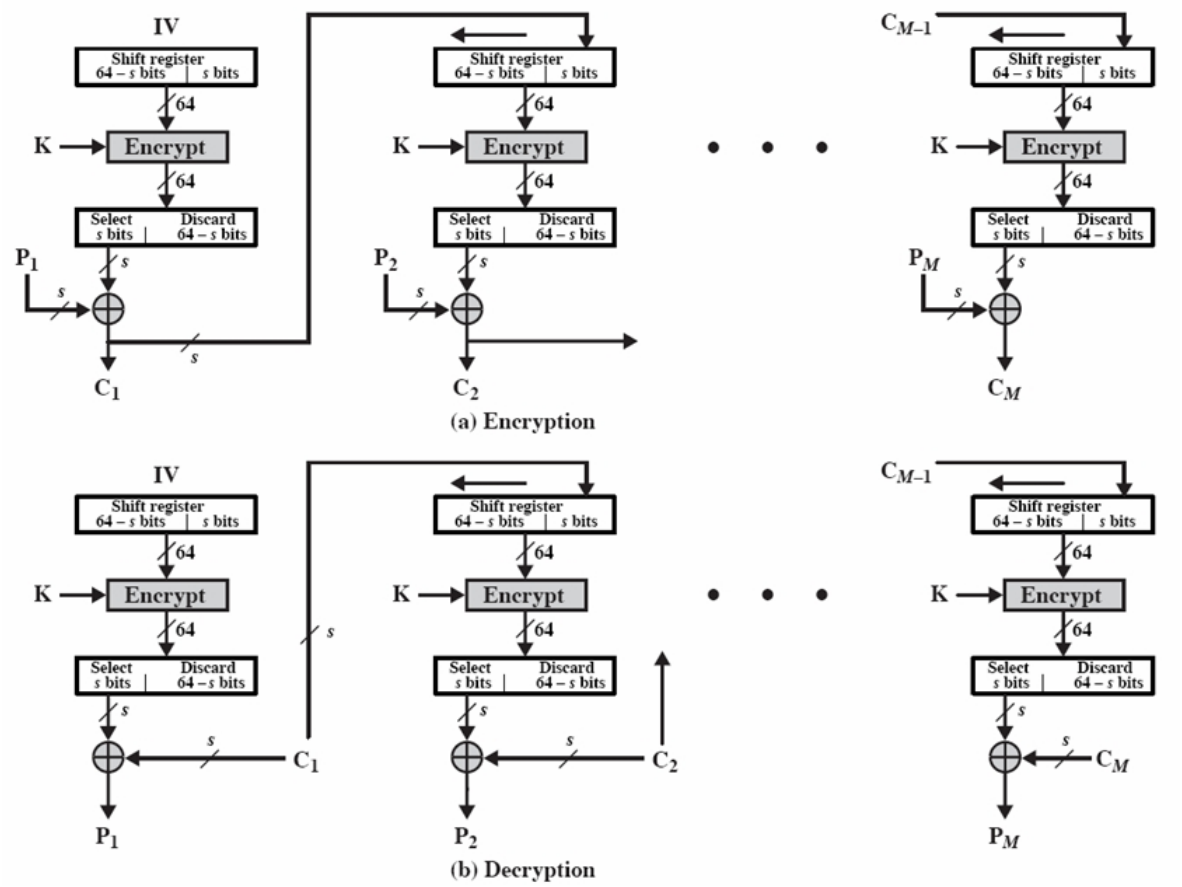

FIG. 5.44 – Mode CFB

- pas de préprocessing
- pas de parallélisme

5.6.3.3 Propagation des erreurs :

+ la perte d'un bloc de c_i : le synchronisme est récupéré dès que le c_i est sorti du registre
+ une erreur dans m_i affecte tous les c_i suivants mais uniquement le m_i correspondant lors du déchiffrement
- une erreur dans c_i affecte le m_i correspondant et les 64/s blocs suivants

5.6.4 OFB - Output FeedBack

Le feedback est indépendant du message. Tout le mécanisme est donc indépendant des blocs m_i et c_i. C'est une variante d'un chiffrement de Vernam avec réutilisation de la clé et de l'IV. Il est utilisé dans le cadre de chiffrement de flux sur un canal bruyant.

5.6.4.1 Sécurité :

+ IV unique et aléatoire
+ effacement du format de M

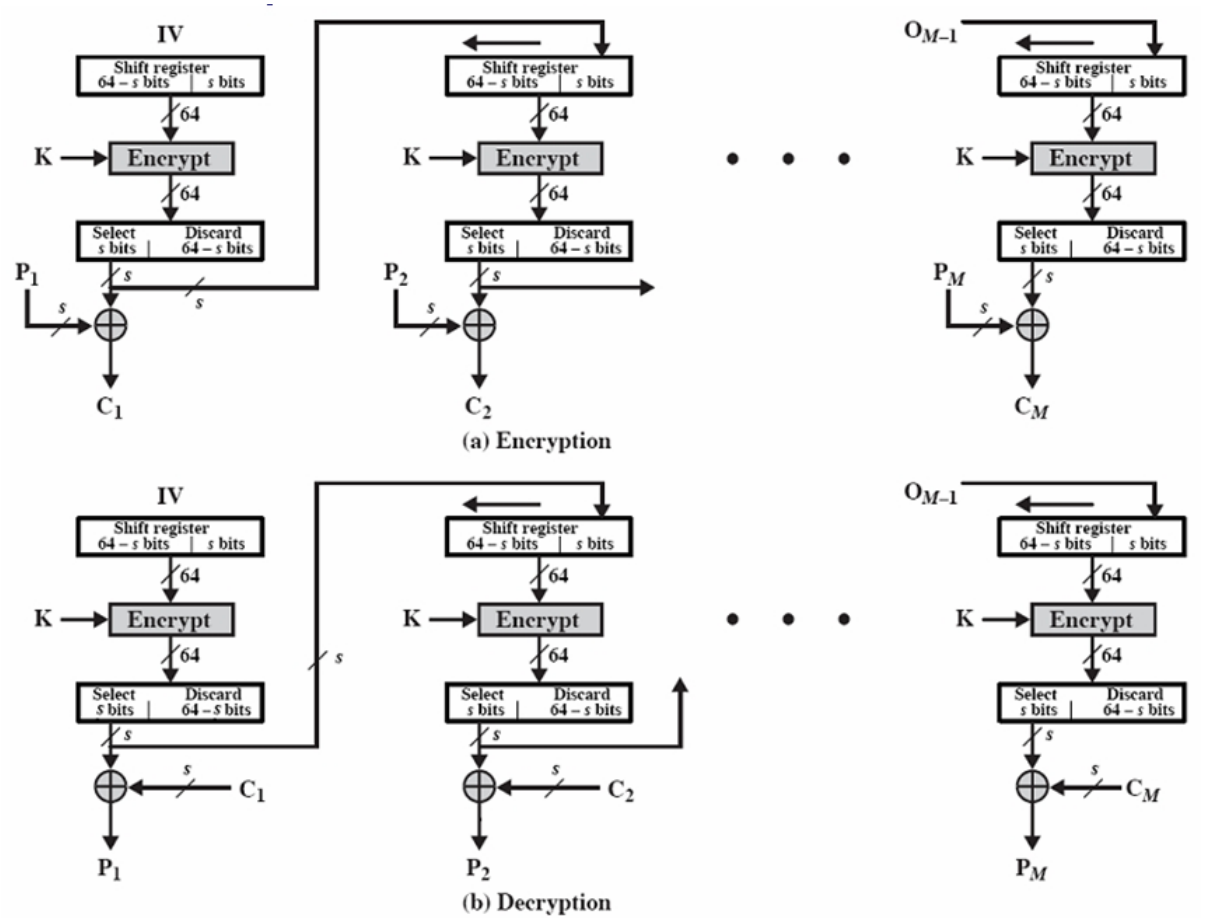

FIG. 5.45 – Mode OFB

+ clé réutilisable
+ pas de répétition de blocs
- le chiffrement ne consiste qu'en un XOR

5.6.4.2 Efficacité :

+ même vitesse de chiffrement
+ préprocessing
- pas de parallélisme

5.6.4.3 Propagation des erreurs :

+ une erreur dans c_i affecte uniquement le bit correspondant de m_i
- pas de mécanisme de récupération de synchronisation (il faut que les shifts registers soient identiques pour E et D)

5.6.5 CTR - CounTeR

Ce mode est très rapide, ce qui le rend utile dans les réseaux grande vitesse. On y trouve un compteur en remplacement d'un IV et une clé différente pour chaque texte clair. Le compteur est une fonction

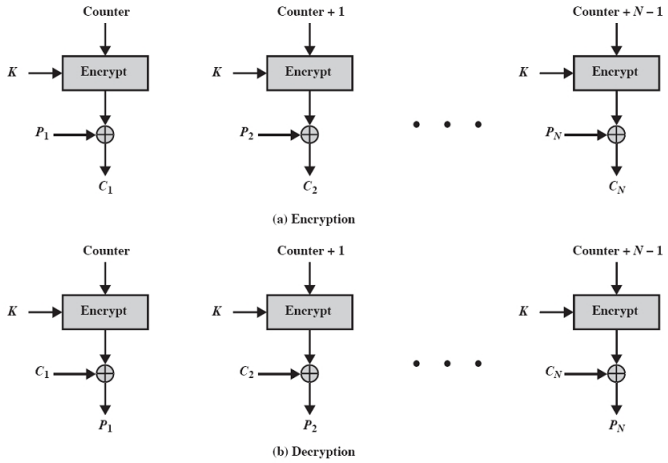

FIG. 5.46 – Mode CTR

simple mais garantissant que la séquence utilisée pour chiffrer ne sera pas réutilisée (cycle plus long que ne le nécessite le message).

5.6.5.1 Sécurité :

+ Sécurité démontrable
+/- non-réutilisation du compteur

5.6.5.2 Efficacité :

+ Parallélisme (architecture multi-coeur)
+ Accès aléatoire possible
+ Préprocessing possible
+ Simplicité du mode

5.6.5.3 Propagation des erreurs :

+ une erreur d'un bit dans c_i affecte le bit correspondant dans m_i

5.6.6 Tableau récapitulatif

	ECB	CBC	CFB	OFB	CTR
Sécurité	Faible	Forte	Forte	Forte	Forte
Paralélllisme	Oui	Non	Non	Non	Oui
Préprocessing	Non	Non	Non	Oui	Oui
Accès aléatoire	Oui	Non	Non	Non	Oui

FIG. 5.47 – Comparaison des modes

5.7 Références supplémentaires

`http://sic.epfl.ch/publications/FI00/fi-sp-00/sp-00-page8.html`

http://csrc.nist.gov/CryptoToolkit/aes/
http://www.uqtr.ca/~delisle/Crypto/prives/blocs_modes.php
http://www.cs.ucdavis.edu/~rogaway/papers/ctr.pdf

Chapitre 6

Chiffrement de flux

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 avec des blocs de 64 bits).
- 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 $k_1, k_2, ..., k_i$.

La sécurité du chiffrement dépend de la qualité du générateur : si $k_i = 0$ pour tout i, M=C. Mais si la séquence des clés k_i est infinie et complètement aléatoire, on obtient un One-Time-Pad. En pratique, on se situe entre les deux, c'est-à-dire une séquence pseudo-aléatoire.

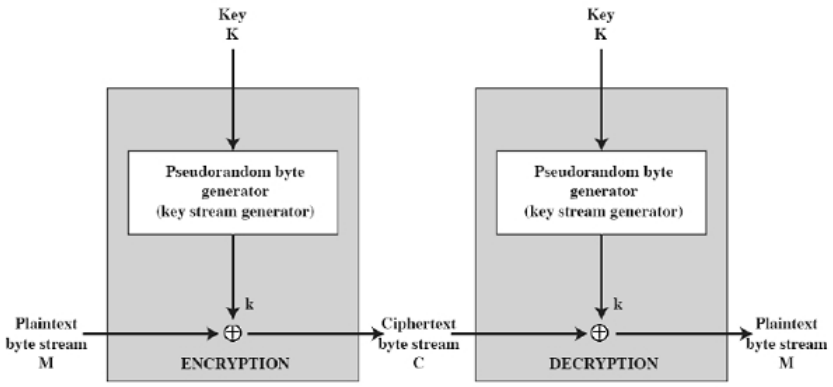

FIG. 6.1 – Schéma de chiffrement par flux

En termes de propagation d'erreur, une erreur dans C_i n'affecte qu'1 bit de M_i. La perte ou l'ajout d'un bit de C_i affecte tous les bits suivants de M après déchiffrement.

Le générateur de clés peut être considéré comme une machine à états finis. Un exemple de ce type de générateur est le FSR (Feedback Shift Register), illustré à la figure 6.2.

La clé notée sur les schémas 6.1 et 6.2 contient les valeurs d'initialisation et les coefficients (voir ci-dessous) du FSR.

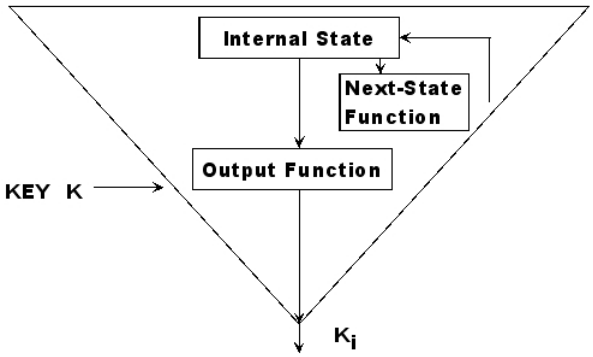

FIG. 6.2 – Schéma général d'un Feedback Shift Register

6.1 Les LFSR classiques

Un LFSR, illustré par la figure 6.3, est constitué de m sections contenant chacune 1 bit (état interne) et d'une fonction de retour (feedback). Les sections sont en réalité des cases mémoire (ou *flip-flops*). A chaque impulsion de l'horloge, les éléments des sections sont décalés d'une position vers la droite. L'élément de la section 0 constitue l'output du LFSR. L'élément de la section m-1 est le résultat de la fonction de retour.

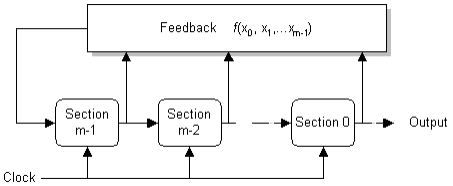

FIG. 6.3 – FSR : Fonction du registre

La fonction de retour qui donne la valeur au flip-flop en position m-1 est la suivante :

$$\begin{aligned} f(x_0, ..., x_{m-1}) &= c_0 x_0 + c_1 x_1 + ... + c_{m-1} x_{m-1} \\ &= \sum_{i=0}^{m-1} c_i x_i \end{aligned}$$

Les bits de sortie du LFSR constituent la suite pseudo-aléatoire. La qualité du générateur est définie par sa période de retour, c'est-à-dire la période après laquelle la même série d'output est répétée.

L'initialisation du LFSR (le contenu des flip-flops) et les valeurs des coefficients constituent ce qu'on nomme communément la *clé courte*.

Les coefficients c_i ont pour valeur 0 ou 1 et définissent les sections qui seront prises en compte par la fonction. Les valeurs de ces coefficients ainsi que la valeur initiale des éléments du registre sont fixées à l'avance. Ces valeurs sont représentées par la clé K des schémas 6.1 et 6.2.

Les valeurs des coefficients doivent restés secrètes, tout comme l'initialisation des registres. Dans le cas contraire, on pourrait observer M bits consécutifs et calculer les suivants. En effet, si ces c_i sont connus, il n'y a que 2^M suites différentes associées aux 2^M possibilités d'initialisation pour le registre.

6.2 Utilisation moderne des LFSR

Les LFSR ne sont pas cryptographiquement sûrs, de par l'existence d'un algorithme permettant de retrouver un polynôme engendrant une suite finie en un temps quadratique. Cet algorithme porte le nom de Berlekamp-Massey (voir [3] pour plus de détails).

Celà a conduit à une utilisation plus évoluée des LFSR.

6.2.1 Les LFSR combinés

Pour résoudre ce problème, on utilise m LFSR donnant chacun un bit en sortie. Ces bits constituent l'entrée d'une fonction booléenne qui elle-même donne un bit en sortie à chaque coup d'horloge. Les polynômes générateurs des LFSR sont souvent publics, de même que la fonction booléenne. Seules les initialisations sont maintenues secrètes. Ce type de LFSR est illustré à la figure 6.4.

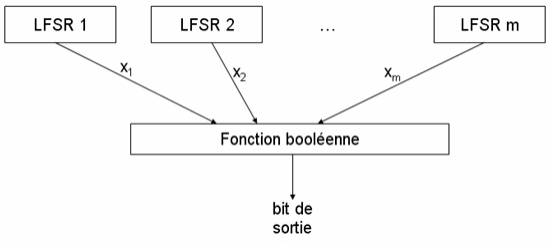

FIG. 6.4 – LFSR combiné

6.2.2 Les LFSR filtrés

Un seul LFSR est utilisé, et à chaque coup d'horloge, l'état du registre (ou d'au moins une partie du registre) constitue l'entrée d'une fonction booléenne. Les LFSR filtrés sont équivalents aux LFSR combinés. On peut effectivement le voir comme la combinaison de m LFSR dont la fonction de sélection est identique. Ce type de LFSR est illustré à la figure 6.5.

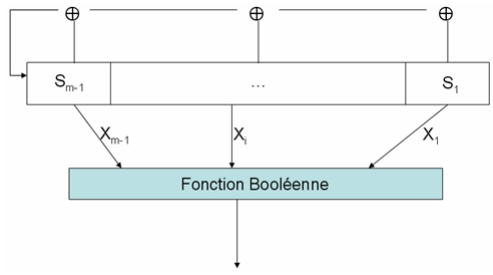

FIG. 6.5 – LFSR filtré

6.2.3 Le contrôle d'horloge

Dans le cas présent, on contrôle l'horloge d'un LFSR par celle d'un autre, ce qui introduit la non-linéarité.

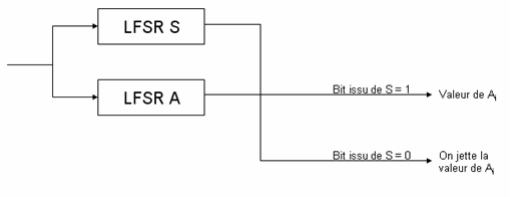

FIG. 6.6 – Le générateur rétrécissant

Le générateur rétrécissant

Soient deux LFSR dont un est "arbitre" de l'autre. Ainsi, sur la figure 6.6, le LFSR S est responsable de la sortie du système.
- Si S vaut 1, la valeur de sortie du système est donnée par la valeur du LFSR A.
- Si S vaut 0, la valeur du bit de A est supprimée, et aucune sortie n'est validée.

Ce type de système possède le désavantage d'avoir un flux de sortie assez irrégulier puisque tout dépend de l'arbitrage de S. Lorsque S supprime la sortie, il faut attendre le coup d'horloge suivant pour espérer trouver une valeur en sortie.

6.2.4 Sécurité des LFSR

Pour assurer une meilleure sécurité, on doit respecter 3 caractéristiques :

- La période doit être la plus longue possible.
- Le flux de sortie doit être le plus aléatoire possible.
- La clé K servant d'initialisation doit être la plus longue possible (au moins 128 bits) et rester secrète.

Une attaque très commune porte le nom d'*attaque par corrélation*. Le principe est de trouver un LFSR dont la sortie est corrélée avec la sortie des LFSR utilisés.

Attaque par corrélation : le générateur de Geffe

Soit la figure 6.7. Un générateur de Geffe est composé de 3 LFSR dont les longueurs L_1, L_2 et L_3 sont relativement premières deux à deux, et dont la fonction de combinaison est donnée par

$$f(x_1, x_2, x_3) = x_1x_2 \oplus (1 + x_2)x_3 = x_1x_2 \oplus x_2x_3 \oplus x_3$$

Ce générateur n'est pas sûr car des informations au sujet des LFSR 1 et 3 peuvent être découvertes à partir d'une étude probabiliste sur le flot de sortie.

Ainsi, notons $x_1(t), x_2(t), x_3(t)$ et $z(t)$ comme étant les sorties respectives du LFSR 1, 2, 3, et du système. On remarque rapidement que selon la valeur de $x_2(t)$, on peut déduire les valeurs de $x_1(t)$ ou de $x_3(t)$ avec une probabilité de 3/4.

En effet, il vient

$$\begin{aligned} P(z(t) = x_1(t)) &= P(x_2(t) = 1) + P(x_2(t) = 0).P(x_3(t) = x_1(t)) \\ &= 1/2 + 1/2 * 1/2 \\ &= 3/4 \end{aligned}$$

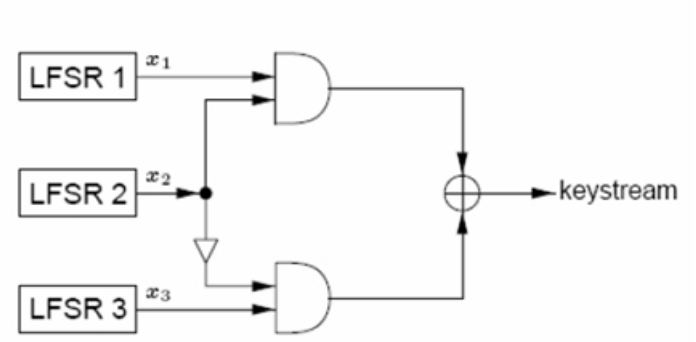

FIG. 6.7 – Le générateur de Geffe

et réciproquement

$$\begin{aligned} P(z(t) = x_3(t)) &= P(x_2(t) = 0) + P(x_2(t) = 1).P(x_3(t) = x_1(t)) \\ &= 1/2 + 1/2 * 1/2 \\ &= 3/4 \end{aligned}$$

L'attaque consiste alors à analyser les valeurs de la clé du LFSR 1 (ou du LFSR 3), et de vérifier que sa sortie correspond à celle de la sortie du système dans 75% des cas. Si on crée la table d'états pour les valeurs de bit x_1, x_2 et x_3, on voit que c'est bien le cas.

6.3 RC4

Il s'agit d'un chiffrement par flux créé en 1987 par Ron Rivest[1]. Diverses analyses ont démontré que la période du générateur est supérieure à 10^{100}.

Il s'agit probablement du chiffrement par flots le plus utilisé actuellement. On le retrouve notamment dans le standard SSL/TLS, dans Oracle Secure SQL, ou encore dans le protocole WEP (Wired Equivalent Privacy, de la norme 802.11). Ce dernier fut remplacé par le WPA (Wi-Fi Protected Access), mais celui-ci utilise toujours le RC4.

Initialement gardé secret par la RSA, cet algorithme a été publié anonymement en 1994 sur Internet. Il existe une version allégée du chiffrement RC4, portant le nom de ARC4 (Alleged RC4), utilisable légalement. Le RC4 reste la propriété de RSA Labs.

Un chiffrement par RC4 est très rapide, comme le montre le tableau 6.1

Algorithme	**Longueur de la clé**	**Vitesse (en Mbps)**
DES	56	9
3DES	168	3
RC4	Variable	45

TAB. 6.1 – Vitesses de quelques chiffrements symétriques.

[1]Selon la rumeur, RC signifierait *Ron's Code* ou *Rivest Cipher*.

6.3.1 Fonctionnement du RC4

Cet algorithme fonctionne sur les octets. Ainsi, la clé, de longueur variable, peut avoir une taille comprise entre 1 et 256 octets (de 8 à 2048 bits). Elle est utilisée pour initialiser un vecteur S de 256 octets. A tout moment, S contient une permutation de toutes les cellules le composant. La figure 6.8 illustre le principe du RC4.

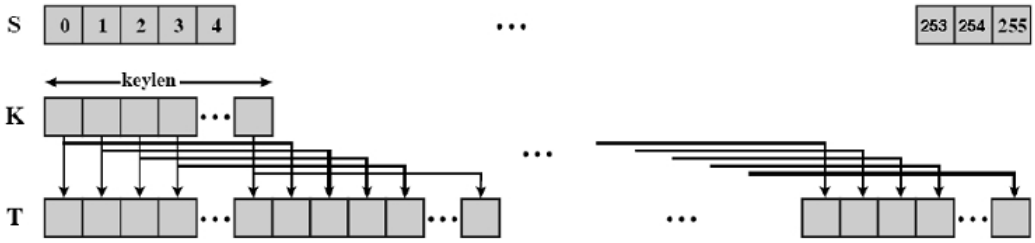

FIG. 6.8 – Fonctionnement du chiffrement RC4

6.3.1.1 Initialisation

Initialement, les cellules de S reçoivent une valeur égale à leur position (i.e., S[0]=0, S[1]=1, ...) Un vecteur temporaire de longueur T (de longueur égale à S) est également créé destiné à recevoir la clé. Si la longueur de la clé K est égale à 256 octets, K est simplement transféré dans T. Si K est inférieur à 256 octets, il est recopié dans T jusqu'à atteindre la taille de T. Ce procédé est illustré au point (a). On peut également le représenter algorithmiquement comme suit :

```
FOR i = 0 TO 255 DO
    S[i] = i;
    T[i] = K[i mod keylen];
```

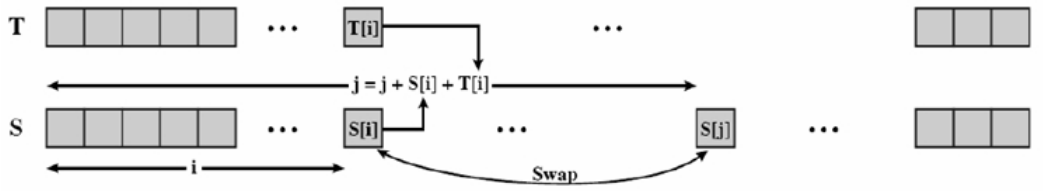

FIG. 6.9 – Initialisation (1)

Le vecteur temporaire T est ensuite utilisé pour produire la permutation initiale de S. Pour chaque cellule S[i] de S, celle-ci sera échangée avec une autre cellule de S selon un calcul basé sur la valeur comprise dans la cellule T[i] correspondante. Le point (b) illustre cette phase. Tout comme l'initialisation des vecteurs, on peut représenter la permutation initiale de S par un algorithme :

```
j = 0;
FOR i = 0 TO 255 DO
    j = (j + S[i] + T[i]) mod 256;
    SWAP(S[i], S[j]);
```

Ces deux portions de code portent le nom de KSA (*Key Schedule Algorithm*).

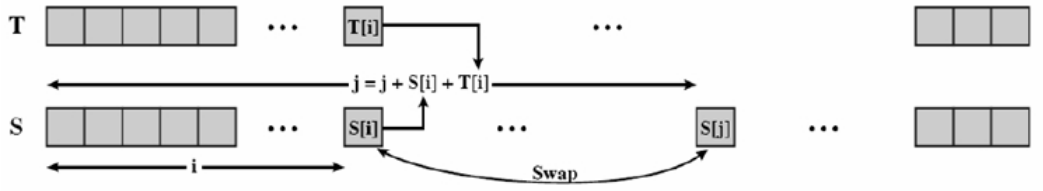

FIG. 6.10 – Initialisation (2)

6.3.1.2 Génération du flux

A partir de cet instant, la clé d'entrée n'est plus utilisée. Pour chaque S[i], on procèdera à un échange avec un autre octet de S, selon un schéma basé sur la configuration courante de S. Une fois arrivé à S[255], le processus redémarre à la cellule S[0]. Le point (c) de la figure 6.8 présente la procédure. A nouveau, on peut illustrer algorithmiquement la méthode (on parle de PRGA pour *Pseudo-Random Generation Algorithm*) :

```
i, j = 0;
WHILE (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    SWAP (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
```

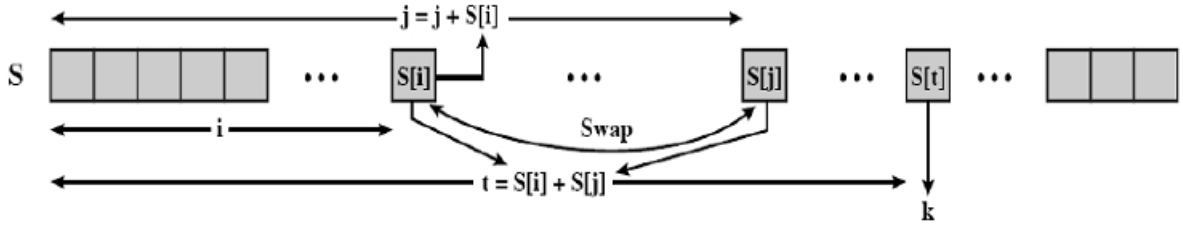

FIG. 6.11 – Génération du flux RC4

La valeur de k est alors utilisée pour le chiffrement ($\oplus$ avec le prochain octet de texte clair), ou pour le déchiffrement ($\oplus$ avec le prochain octet de texte chiffré).

6.3.2 Robustesse du RC4

On peut tout d'abord remarquer que la sécurité repose uniquement sur la clé. En effet, c'est elle, et uniquement elle, qui détermine le flux de sortie du générateur.

En 2001, une découverte affaiblit l'aspect sécuritaire du RC4. Fluhrer, Mantin et Shamir remarquèrent que les premiers octets en sortie du RC4 n'étaient pas tout à fait aléatoires. De fait, il semblerait que ces quelques premiers bytes divulgueraient des informations importantes sur la clé d'initialisation.

Le protocole WEP a été remplacé par le protocole WPA pour pallier les problèmes rencontrés avec la gestion des clés dans le RC4.

Il semblerait que le RC4 soit toujours sûre pour autant que certaines conditions soient remplies au niveau de la clé d'initialisation :

– utilisation d'une nonce
– une fonction de hachage

On utilisera par exemple la construction HMAC pour réaliser une telle procédure.

6.4 Comparaisons des chiffrements par blocs et par flots

Le tableau 6.2 regroupe les avantages et inconvénients de chaque famille. Il est toutefois à remarquer qu'au niveau des applications, chaque type de chiffrement peut être utilisé dans toutes les applications.

	par Blocs	**par Flots**
Avantages	- Réutilisation des clés	- Rapidité - Moins de code d'implémentation
Inconvénients		- Deux utilisations d'une même clé facilite la cryptanalyse
Applications	- Transfert de fichiers	- Chiffrement de canal de communication

TAB. 6.2 – Comparaison des chiffrements par blocs et par flots.

6.5 Ressources supplémentaires

`http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html`
`http://www.crimelabs.net/docs/stream.html`

Chapitre 7

Le chiffrement par clé publique

7.1 Concept

Dans le cas des systèmes symétriques, on utilise une même clé pour le chiffrement et le déchiffrement. Le problème repose dans la transmission de la clé : il faut une clé par destinataire. Dans le cas des systèmes asymétriques, chaque personne possède 2 clés distinctes (une privée, une publique) avec impossibilité de déduire la clé privée à partir de la clé publique. De ce fait, il est possible de distribuer librement cette dernière.

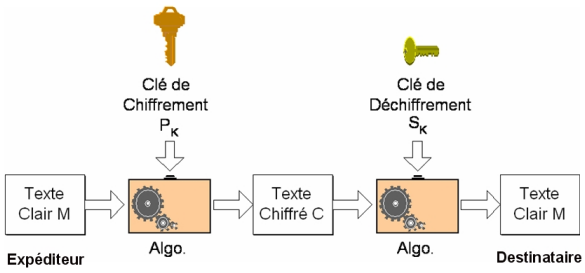

FIG. 7.1 – Chiffrement à clé publique

On peut classer l'utilisation des algorithmes à clé publique en 3 catégories :
- Chiffrement/déchiffrement : cela fournit le secret.
- Signatures numériques : cela fournit l'authentification.
- Échange de clés (ou des clefs de session).

Quelques algorithmes conviennent pour tous les usages, d'autres sont spécifiques à un d'eux.

Le concept date officiellement de 1976 de Diffie et Hellman. Officieusement, les bases existent depuis 1969 par Ellis. La première implémentation a lieu en 1978 par Rivest, Shamir et Adleman sous la forme de l'algorithme RSA bien que, là aussi, les fondements de ce système datent de 1973, par Cocks.

La sécurité de tels systèmes repose sur des problèmes calculatoires :

- RSA : factorisation de grands entiers
- ElGamal : logarithme discret
- Merkle-Hellman : problème du sac à dos (knapsacks)
- ...

La recherche des clés par force brute est toujours théoriquement possible mais les clefs utilisées sont trop grandes (> 512bits). La sécurité se fonde sur une assez grande différence en termes de difficulté entre les problèmes faciles (déchiffrement) et difficiles (décryptement) :
- généralement le problème difficile est connu, mais il est trop complexe à résoudre en pratique,
- La génération des clés exige l'utilisation de très grands nombres.

En conséquence, ce type de chiffrement est lent si on le compare aux chiffrements symétriques.

7.2 Merkle-Hellman

7.2.1 Définition du problème

Soit un havresac de capacité T et un ensemble S d'objets occupant les espaces $S = \{a_1, a_2, a_3, ..., a_n\}$. Il faut trouver un vecteur de sélection $V = \{v_1, v_2, v_3, ..., v_n\}$ satisfaisant la relation $\sum(a_i * v_i) = T$.

Ce problème n'a pas toujours une solution. Aussi, si T et S sont très grands, il est beaucoup plus difficile de trouver le vecteur associé.

Exemple : Soit S = $\{17, 38, 73, 4, 11, 1\}$ et T = 53 = 38 + 4 + 11. Donc V = $\{0, 1, 0, 1, 1, 0\}$.
Pour T = 45, il n'y a pas de solution.

7.2.2 Idée de base

Un bloc de texte clair de longueur égale au nombre d'objets d'un tas sélectionnerait des objets. Les bits du texte clair correspondraient aux valeurs des v_i : un 1 signifierait que l'objet est présent et 0 un objet absent. Et le texte chiffré serait la somme résultante.

Exemple :

M :	1	1	1	0	0	1	0	1	0	1	1	0
Tas :	1	5	6	11	14	20	1	5	6	11	14	20
C :	1+5+6+20 = 32						5+11+14 = 30					

7.2.3 Les empilements

Il y a 2 problèmes d'empilement :
- 1 soluble en temps linéaire
- 1 soluble en temps exponentiel

L'empilement facile peut être transformé pour créer un empilement difficile. Pour la clé publique, on utilisera un empilement difficile qui servira à chiffrer. La clé privée quant à elle, utilisera un empilement facile, qui donne un moyen simple de déchiffrer les messages. Bien sûr, ceux qui ne connaissent pas la clé privée sont obligés de résoudre le problème d'empilement difficile, ce qui est infaisable en pratique.

7.2.3.1 Empilement facile

Si la liste des poids est super-croissante[1], on utilise un algorithme (appelé *glouton*) de la manière suivante :

1. Prendre le poids total et le comparer avec le plus grand nombre de la suite.

[1] Une liste est super-croissante lorsque tout terme est plus grand que la somme des termes qui le précède.

- Si le poids total est inférieur à ce nombre, alors celui-ci n'est pas dans le tas. On recommence l'opération avec le nombre suivant dans le tas (qui, par définition de la suite, sera plus petit)
- Si le poids total est supérieur à ce nombre, alors celui-ci est dans le tas

2. Réduire le poids du tas à créer de ce nombre et passer au plus grand nombre suivant de la suite.
3. Répéter jusqu'à ce que ce soit terminé.
4. Si le poids total a pu être ramené à 0 : il y a une solution.

7.2.3.2 Empilement difficile

Dans le cas présent, on ne connaît pas d'algorithme rapide. Il faut tester méthodiquement toutes les solutions possibles, ce qui, si la suite des poids est suffisamment longue, est impraticable. Ces algorithmes sont exponentiels.

Le cryptosystème de Merkle-Hellman exploite cette propriété. La clé privée est une suite de poids super-croissante. A partir de celle-ci, on calcule la clé publique. Ce calcul consiste à prendre la suite super-croissante, et à la multiplier par (n modulo m), avec m supérieur à la somme de tous les termes de la suite, et n ne devant avoir aucun facteur commun avec m.

7.2.4 Algorithme

Le chiffrement consiste à additionner les termes où un 1 apparaît. Pour le déchiffrement, on calcule n^{-1} tel que

$$n * n^{-1} \equiv 1 \bmod m$$

Ensuite, on multiplie chaque valeur du texte chiffré par $n^{-1} \bmod m$.

En pratique, les sacs contiennent environ 250 éléments. Et chaque terme a une longueur de 200 à 400 bits. Le module a une longueur de 100 à 200 bits.

Exemple de calcul de la clé publique Soit S, une séquence super-croissante de h entiers : par exemple $S = \{1, 2, 4, 9\}$.

Choisissons un multiplicateur n et un module m : soit $n = 15$ et $m = 17$

- $1 * 15 \bmod 17 \Rightarrow 15$
- $2 * 15 \bmod 17 \Rightarrow 13$
- $4 * 15 \bmod 17 \Rightarrow 9$
- $9 * 15 \bmod 17 \Rightarrow 16$

Le havresac difficile est donc H = $\{15, 13, 9, 16\}$, et représente la clé publique.

□

Le message est ainsi traité comme une séquence de bits :

$$P = [p_1, p_2, p_3, ..., p_k]$$

On le divise en blocs de h bits :

$$P_0 = [p_1, p_2, p_3, ..., p_h], P_1 = [p_{h+1}, p_{h+2}, p_{h+3}, ..., p_{2*h}], ...$$

On utilise chaque bloc comme vecteur V du problème de havresac.

Exemple de chiffrement Soit $P = 0100101110100101 \Rightarrow 0100\ 1011\ 1010\ 0101$

- $[0,1,0,0] * [15,13,9,16] \Rightarrow 13$
- $[1,0,1,1] * [15,13,9,16] \Rightarrow 40$
- $[1,0,1,0] * [15,13,9,16] \Rightarrow 24$
- $[0,1,0,1] * [15,13,9,16] \Rightarrow 29$

Le message chiffré est donc $\{13, 40, 24, 29\}$ en utilisant le havresac public (la clef publique) H = [15, 13, 9, 16].

□

Pour le déchiffrement, le destinataire légitime connaît le havresac simple S et les valeurs de n et de m. Il peut donc déterminer n^{-1}.

Exemple de déchiffrement Avec $n = 15$ et $m = 17$, n^{-1} vaut 8 car $15 * 8 = 120 = 7 * 17 + 1$. On a alors, par l'algorithme glouton :

- $13 * 8 \bmod 17 = 104 \bmod 17 = 2 = [1,2,4,9] * [0100]$
- $40 * 8 \bmod 17 = 320 \bmod 17 = 14 = [1,2,4,9] * [1011]$
- $24 * 8 \bmod 17 = 192 \bmod 17 = 5 = [1,2,4,9] * [1010]$
- $29 * 8 \bmod 17 = 232 \bmod 17 = 11 = [1,2,4,9] * [0101]$

et le texte en clair est 0100 1011 1010 0101 ⇒ 0100101110100101. On a donc bien retrouvé le texte original.

□

7.2.5 Sécurité

Plusieurs failles ont été découvertes, qui ont rendu l'algorithme obsolète.

Herlestam a démontré en 1978 qu'un bit de texte clair pouvait souvent être retrouvé.

Malgré les modifications apportées à l'algorithme à la suite de ces découvertes, des travaux de Shamir (1982-84) et Brickell (1985) ont poussé à l'abandon de cet algorithme.

7.3 RSA : Rivest - Shamir - Adleman

Il est basé sur le calcul exponentiel. Sa sécurité repose sur la fonction unidirectionnelle suivante : le calcul du produit de 2 nombres premiers est aisé. La factorisation d'un nombre en ses deux facteurs premiers est beaucoup plus complexe.

Il s'agit du système le plus connu et le plus largement répandu, basé sur l'élévation à une puissance dans un champ fini sur des nombres entiers modulo un nombre premier. Le nombre d'exponentiation prend environ $O((\log n)^3)$ opérations ce qui est rapide et facile. Il emploie de grands nombres entiers (par exemple représentés sur 1024 bits).

Ce cryptosystème utilise deux clés d et e, interchangeables[2]. Le chiffrement se fait selon

$$C = M^e \bmod n$$

[2]Cette propriété sera à nouveau évoquée dans le chapitre portant sur l'authentification.

et le déchiffrement par

$$M = C^d \bmod n.$$

La sécurité repose sur le coût nécessaire pour factoriser de grands nombres. Le nombre de factorisation prend environ $O(e^{\log n \log(\log n))})$ opérations ce qui demande un temps de calcul trop important pour les machines actuelles, dans un cadre privé. On l'utilise pour la confidentialité, l'authentification, ou encore une combinaison des 2.

7.3.1 Principes

On possède une paire de clés, l'une publique (e,n) et une privée (d,n). La première étape revient à choisir n. Il doit s'agir d'une valeur assez élevée, produit de 2 nombres premiers très grands p et q. En pratique, si p et q ont 100 chiffres décimaux, n possèdera 200 chiffres. Selon le niveau de sécurité souhaité, la taille de n peut varier : 512 bits, 768, 1024 ou 2048[3].

Dans un second temps, on choisira un très grand entier e, relativement premier à (p-1)*(q-1). La clé publique sera formée par (e,n). On choisira ensuite un d tel que

$$e * d \equiv 1 \bmod (\Phi(n)).$$

La clé privée sera donnée par (d,n).

Dernière phase : on jette p et q. Le cryptanalyste devant retrouver ces valeurs, il faut les détruire pour éviter les fuites.

7.3.1.1 Justification de l'inversibilité

Par les théorèmes d'Euler et de Fermat, on sait que

$$a^{\Phi(n)} \equiv 1 \bmod n$$

et

$$a^{\Phi(n)} \bmod n = 1$$

où (a,n)=1.

Dans le RSA, on a $n = p * q$. De plus, $\Phi(n)$ donne le nombre d'entiers positifs plus petits que n et relativement premiers à n (si p est premier, $\Phi(p) = p - 1$). Si $n = p * q$, avec p et q premiers, il vient

$$\Phi(n) = \Phi(p) * \Phi(q) = (p - 1) * (q - 1)$$

De par la façon de choisir e et d,

$$e * d \equiv 1 \bmod \Phi(n) = k * \Phi(n) + 1$$

pour un certain k.

De plus, par définition et propriétés des opérations modulo n, on a :

$$D_k(E_k(M)) = ((M)^e \bmod n)^d \bmod n = (M^e)^d \bmod n = M^{e*d} \bmod n$$

Et donc :

$$M^{e*d} = M^{k*\Phi(n)+1} = M^{k*\Phi(n)}.M \bmod n = 1.M \bmod n = M \bmod n$$

[3]Le FBI utilise le RSA 4096.

7.3.2 Résumé

1. Génération de 2 nombres premiers p et q
2. Calcul de n = p*q
3. Déterminer e tel que $3 < e < \Phi(n)$ et $(e, \Phi(n)) = 1$
4. Calculer d tel que $e * d \equiv 1 \bmod \Phi(n)$
5. Clé publique : (e,n)
6. Clé privée : (d,n)
7. p et q doivent rester secrets, voire supprimés
8. $C = M^e \bmod n$ et $M = C^d \bmod n$

Exemple :

Soient p = 31, q = 53 c'est-à-dire n=1643. $\Phi(n) = 1560$ (nombre d'éléments relativement premiers à n et < n).
Soit e = 11 (par exemple, et on a bien (e,$\Phi(n)$)=1).
On détermine que d = 851 (inverse modulaire de e sur $Z_{\Phi(n)}$).
La clé publique est donc (11,1643) et la clé privée est (851,1643).

Soit le codage par la position dans l'alphabet du mot «ANEMONE». Il vient

01 14 05 13 15 14 05

On procède selon deux conditions :

1. Découpage en morceaux de même longueur, ce qui empêche la simple substitution :

   011 405 131 514 05_

   On ajoute un padding initial si nécessaire.

   001 140 513 151 405

   Cela provoque la perte des patterns (« NE »).
2. Découpage en morceaux de valeur inférieure à n, car opération modulo n.

Lors du chiffrement, on a

$001^{11} \bmod 1643$	0001
$140^{11} \bmod 1643$	0109
$513^{11} \bmod 1643$	0890
$151^{11} \bmod 1643$	1453
$405^{11} \bmod 1643$	0374

et pour le déchiffrement,

$0001^{851} \bmod 1643$	001
$0109^{851} \bmod 1643$	140
$0890^{851} \bmod 1643$	513
$1453^{851} \bmod 1643$	151
$0374^{851} \bmod 1643$	405

Lors du déchiffrement, sachant qu'il faut obtenir des blocs de 2 éléments (grâce au codage particulier de l'exemple), on a bien

01	14	05	13	15	14	05
A	N	E	M	O	N	E

7.3.2.1 Remarques

Il n'est pas très astucieux de choisir d'aussi petites valeurs car on peut retrouver d très facilement.

En pratique, il faut prendre de très grandes valeurs de p et q. Pour retrouver ces grandes valeurs, il faudra alors utiliser le Jacobien et le test de Solovay-Strassen par exemple.

7.3.3 Sécurité

7.3.3.1 Attaques

Il existe trois approches pour attaquer le RSA :

- recherche par force brute de la clé (impossible étant donné la taille des données),
- attaques mathématiques (basées sur la difficulté de calculer $\Phi(n)$, la factorisation du module n) :
  - factoriser n=p*q et par conséquent trouver $\Phi(n)$ et puis d,
  - déterminer $\Phi(n)$ directement et trouver d,
  - trouver d directement.
- attaques de synchronisation (sur le fonctionnement du déchiffrement).

A l'heure actuelle, la factorisation connait de lentes améliorations au cours des années. La meilleure amélioration possible reste l'optimisation des algorithmes. Excepté un changement dramatique, le RSA-1024 restera sûr pour les prochaines années. D'après les projections, une clé de 2048 bits est sensée tenir jusque 2079 si on tient compte de la loi de Moore. Mais ces valeurs sont correctes uniquement si on respecte les propriétés de e, d, p et q.

Attaque de synchronisation (timing attack)

Développé dans le milieu des années 90, il s'agit d'exploiter les variations de temps pris pour effectuer certaines opérations (par exemple la multiplication par un petit ou un grand nombre).

Plusieurs contre-mesures existent telles que l'emploi de temps constants d'élévation à une puissance, l'ajout de délais aléatoires, ou le fait de rendre non visibles les valeurs utilisées dans les calculs. Dans ce dernier cas, cela reviendrait à calculer :

$$(r^e * m^e)d \bmod n$$

7.3.3.2 La menace quantique

Les valeurs précitées sont valables si on pratique la factorisation. A coté de cela, la physique pourrait faire pencher la balance, par l' utilisation d'un ordinateur quantique[4]. Celui-ci existe d'un point de vue théorique depuis 1994 (algorithme de Shor), et son prototype depuis 1996. Si son évolution se poursuit, il permettrait de réaliser la factorisation d'un nombre en un temps polynomial. Le principe est que les 0 et 1 représentés par les portes logiques des transistors sont remplacés par l'orientation du champ magnétique émit par les atomes (que l'on nomme des q-bits).

7.3.4 Conseils d'utilisation du RSA

Pour garantir une bonne sécurité, il faut respecter certains règles telles que :

- Ne jamais utiliser de valeur n trop petite,
- N'utiliser que des clés fortes (p-1 et q-1 ont un grand facteur premier),
- Ne pas chiffrer de blocs trop courts,
- Ne pas utiliser de n communs à plusieurs clés,
- Si (d,n) est compromise ne plus utiliser n.

Level	Protection	Symmetric	Asymmetric
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
3	Short-term protection against medium organizations, medium-term protection against small organizations	72	1008
4	Very short-term protection against agencies, long-term protection against small organizations *Smallest general-purpose level, protection from 2008 to 2010*	80	1248
5	Legacy standard level *Use of 2-key 3DES restricted to 10^6 plaintext/ciphertexts, protection from 2008 to 2016*	96	1776
6	Medium-term protection *protection from 2008 to 2026*	112	2432
7	Long-term protection *Generic application-independent recommendation, protection from 2008 to 2036*	128	3248
8	"Foreseeable future" *Good protection against quantum computers*	256	15424

FIG. 7.2 – Taille des clés pour une utilisation sûre.

Un concours, connu sous le nom de "RSA Factoring Challenge", proposait une certaine somme d'argent à tout groupe ayant réussi la factorisation d'une clé de taille donnée (la récompense étant proportionnelle à la taille de la clé mise en défaut). La plus grande clé "cassée" atteignit 663 bits (2005). Le concours fut stoppé fin 2007.

7.4 El Gamal

C'est un algorithme à clef publique présent à la base de la norme U.S. de signature électronique. Il fut inventé par Taher ElGamal en 1984. Il est basé sur la difficulté de calculer des logarithmes discrets.

Le problème du logarithme discret consiste à retrouver un entier λ tel que

$$h = g^{\lambda} \mod p.$$

7.4.1 Principe du chiffrement

Soit un entier premier p très grand et $p-1$ doit avoir un grand facteur premier. On produit :
- une clé secrète s, telle que $s \in (1...p-2)$,
- une clé publique reposant sur l'entier p, un entier a premier avec p, et l'entier P tel que

$$P = a^s \mod p$$

Le nombre a est pris tel que $a \in (0...p-1)$ et $\forall\ k \in (1...p-2)$:

$$a^k \neq 1 \mod p$$

Soit un message M, avec $M < p$. On détermine un nombre aléatoire k qui n'est connu que de celui qui chiffre et différent à chaque message. On calcule alors

$$C_1 = a^k \mod p$$
$$C_2 = M.P^k \mod p$$

On obtient alors le message chiffré $C = (C_1, C_2)$. Le message chiffré est alors deux fois plus long que le message original.

[4] Voir chapitre "Le monde quantique".

7.4.2 Principe du déchiffrement

A la réception, on calcule

$$\begin{aligned} R_1 &= (C_1)^s \bmod p \\ &= a^{sk} \bmod p \\ &= P^k \bmod p \end{aligned}$$

Le destinataire possède la clé privée (s). Ayant P^k, on divise C_2 par cette valeur :

$$\begin{aligned} D_K(C) &= C_2/(R_1) \\ &= M.P^k \bmod p / P^k \bmod p \\ &= M \end{aligned}$$

P^k est donc considéré comme un masque appliqué sous forme multiplicative à M.

Pour décrypter le message, il faudra soit trouver directement un masque jetable, soit trouver la clé privée s, solution de $P = a^s \bmod p$ (et donc trouver le logarithme discret).

Exemple : Soient $p = 2579, a = 2, s = 765$. Il vient
- Clé privée $S_k = (765)$
- Clé publique $P_k = (2579, 2, 949)$ car $2^{765} \bmod 2579 = 949$

Pour chiffrer M = 1299, on choisit k = 853. Il vient

$$C_1 = 2^{853} \bmod 2579 = 435$$

$$C_2 = 1299 * 949^{853} \bmod 2579 = 2396$$

On peut effectivement vérifier que $2396/(435^{765}) \bmod 2579 = 1299$.

□

7.4.3 Efficacité et sécurité

El Gamal est 2 fois plus lent que le RSA. L'inconvénient majeur reste la taille des données chiffrées qui représente 2 fois celle des données en clair.

La recherche de la clé privée (s) à partir de la clé publique est équivalente au problème du logarithme discret (NP). **MAIS** il n'est pas prouvé que la cryptanalyse d'un message chiffré avec El Gamal est équivalente au logarithme discret. En d'autres termes, si le problème du logarithme est résolu polynomialement, alors El Gamal sera cassé. Cependant, rien ne prouve qu'il n'est pas cassable par un autre moyen.

7.5 L'utilisation des courbes elliptiques

Il s'agit d'un concept proposé en 1985 par deux chercheurs Miller et Klobitz, 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, pour 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 R, les courbes elliptiques seront considérées comme l'ensemble des couples (x,y) tels que

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

dont le discriminant

$$-(4a^3 + 27b^2)$$

est non nul.

Pour la dessiner, pour a et b fixés, on calcule y tel que

$$y = \sqrt{x^3 + ax + b}$$

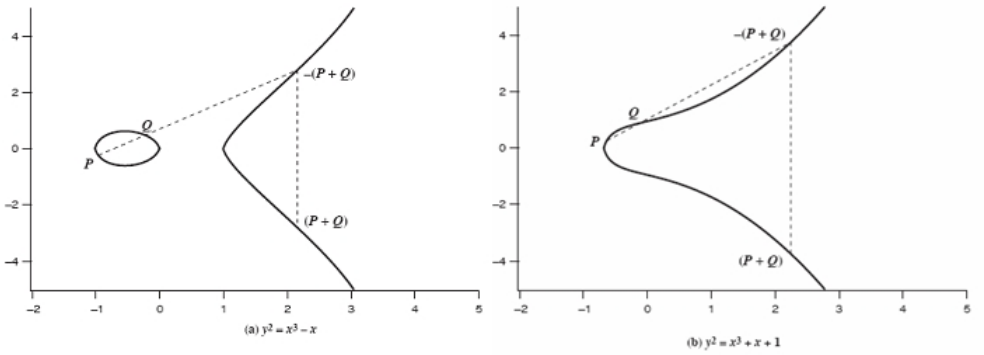

FIG. 7.3 – Deux exemples d'EC

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.

7.5.1 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 O (point à l'infini).

1. O est l'identité pour l'addition : $O = -O$.
2. Pour n'importe quel point $P + O = P$.
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.

7.5.2 Les EC sur 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

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

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

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$
$$49 \bmod 23 = 739 \bmod 23$$
$$3 = 3$$

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

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 Z_p, les couples (x, y) répondant à l'équation sont donnés à la figure 7.4

(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)

FIG. 7.4 – Couples (x,y) répondant à l'équation $y^2 = x^3 + x + 1$

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

1. $P + O = P$
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 :
   $$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$
   $$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$
   où
   $$\lambda = \begin{cases} \left(\dfrac{y_Q - y_P}{x_Q - x_P}\right) \bmod p \text{ si } P \neq Q \\ \left(\dfrac{3x_P^2 + a}{2y_P}\right) \bmod p \text{ si } P = Q \end{cases}$$
4. La multiplication est définie comme une répétition d'additions (ex : $3P = P + P + P$)

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

$$\lambda = \left(\frac{7 - 10}{9 - 3}\right) \bmod 23 = \left(\frac{-3}{6}\right) \bmod 23 = \left(\frac{-1}{2}\right) \bmod 23 = 11$$
$$x_R = \left(11^2 - 3 - 9\right) \bmod 23 = 109 \bmod 23 = 17$$
$$y_R = (11(3 - 17) - 10) \bmod 23 = -164 \bmod 23 = 20$$

Et ainsi, $P + Q = (17, 20)$. Pour trouver $2P$, on a

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

$$x_R = \left(6^2 - 3 - 3\right) \bmod 23 = 30 \bmod 23 = 17$$
$$y_R = (6(3-7) - 10) \bmod 23 = -34 \bmod 23 = 12$$

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

Remarque La section précédente traite du problème dans Z_P. Les équations sont différentes si nous travaillons dans le champ fini $GF(2^m)$.

7.5.3 La cryptographie sur courbes elliptiques (ECC)

Pour utiliser les courbes elliptiques en cryptographie, il faut trouver un problème difficile (tel que la factorisation d'un produit en ses facteurs premiers dans le cas du RSA).

Considérons l'équation

$$Q = kP$$

où $Q, P \in E_p(a, b)$ et $k < p$.

Il est facile de calculer Q connaissant k et P, mais il est difficile de déterminer k si on connait Q et P. Il s'agit du problème du logarithme discret pour les courbes elliptiques : $log_P(Q)$.

Dans une utilisation réelle, le k est très grand, rendant l'attaque par force brute inutilisable (rappelons qu'a priori, l'attaque par force brute est toujours possible...).

7.5.3.1 ECC pour l'échange de clés

Soit un grand entier premier q (en considérant que l'on va utiliser les équations présentées précédemment, et non les équations dans $GF(2^m)$) et les paramètres a et b satisfaisant l'équation $y^2 \bmod q = (x^3 + ax + b) \bmod q$. Cela nous permet de définir $E_q(a, b)$.

Prenons ensuite un point de départ $G(x_1, y_1)$ dans $E_q(a, b)$ dont l'ordre n est élevé. L'ordre n d'un point sur une EC est le plus petit entier positif tel que $nG = O$.

$E_q(a, b)$ et G sont rendu publiques.

L'échange d'une clé par ECC entre deux entités A et B se déroule comme suit :

- A choisit un n_A inférieur à n qui sera sa clé privée. A génère alors sa clé publique $P_A = n_A \times G$.
- B choisit un n_B inférieur à n qui sera sa clé privée. B génère alors sa clé publique $P_B = n_B \times G$.
- A génère la clé secrète $K = n_A \times P_B$ et B génère la clé secrète $K = n_B \times P_A$.

Exemple (Schaefer, Santa Clara University) :

- Soient $p = 211, E_p(0, -4) (\Rightarrow y^2 = x^3 - 4)$ et $G = (2, 2)$. On calcule que $240G = O$ et donc $n = 240$.
- A choisit $n_A = 121$, ce qui lui donne $P_A = 121(2, 2) = (115, 48)$.
- B choisit $n_B = 203$, ce qui lui donne $P_B = 203(2, 2) = (130, 203)$.
- La clé secrète K générée est $121(130, 203) = 203(115, 48) = (161, 69)$.

7.5.3.2 ECC pour chiffrer des données

Même si la cryptographie par courbes elliptiques est souvent employée pour l'échange d'une clé symétrique, elle est aussi utilisée pour chiffrer directement les données. Voici un exemple de cryptosystème les utilisant.

Il faudra ici encoder le texte clair m comme un point P_m de coordonnées x et y. C'est ce point qui sera chiffré. Il faut ici aussi rendre publique un point G et un groupe elliptique $E_q(a, b)$. Les utilisateurs

doivent également choisir une clé privée et générer la clé publique correspondante.

Pour chiffrer le message, A détermine aléatoirement un nombre entier positif k et produit C_m comme un couple de points tel que

$$C_m = \{kG, P_m + kP_B\}$$

On remarquera l'utilisation de la clé publique de B. Pour déchiffrer, B devra multiplier le premier point par sa clé privée, et soustraire le résultat au second point reçu :

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

7.6 Comparaisons

7.6.1 Longueur des clés

Protection	Symmetric	Asymmetric	Elliptic Curve
Attacks in "real-time" by individuals *Only acceptable for authentication tag size*	32	-	-
Very short-term protection against small organizations *Should not be used for confidentiality in new systems*	64	816	128
Short-term protection against medium organizations, medium-term protection against small organizations	72	1008	144
Very short-term protection against agencies, long-term protection against small organizations *Smallest general-purpose level,* *2-key 3DES restricted to 2^{40} plaintext/ciphertexts,* *protection from 2009 to 2012*	80	1248	160
Legacy standard level *2-key 3DES restricted to 10^6 plaintext/ciphertexts, protection from 2009 to 2020*	96	1776	192
Medium-term protection *3-key 3DES, protection from 2009 to 2030*	112	2432	224
Long-term protection *Generic application-independent recommendation, protection from 2009 to 2040*	128	3248	256
"Foreseeable future" *Good protection against quantum computers*	256	15424	512

FIG. 7.5 – Tailles des clés pour une utilisation sûre (2009)

7.6.2 Symétrique et Asymétrique : problèmes communs

Deux problèmes persistent dans ces 2 types de cryptosystèmes.
En premier lieu, la gestion des clés 'secrètes'. La confidentialité reposant exclusivement sur ces dernières, si elles sont égarées, divulguées ou oubliées, toute la sécurité s'effondre. Second point, la sécurité se base sur des arguments empiriques plutôt que théoriques. On suppose (et on espère, même si on en est pratiquement certain), que les problèmes NP le resteront. Cela n'a pas (encore ?) été démontré.

	Asymétrique(RSA)	Asymétrique(ECC)
Avantages	- Cryptosystème largement répandu - Nombreuses études au sujet de sa sécurité	- Taille de clé inférieure pour une sécurité égale - La taille des clés croit moins vite que le RSA si on souhaite une meilleure sécurité - Utilisation pour systèmes embarqués - Calculs moins lourds que l'exponentiation - Utilisation mémoire moindre - Cryptanalyse par algorithme exponentiel
Inconvénients	- Opérations de dé/chiffrement très inégales en termes de temps de calcul - Cryptanalyse par algorithme sous-exponentiel	- Complexe - Peu de développement sur des systèmes à grande échelle (mais tend à changer) - Travaux d'optimisation essentiellement destinés aux systèmes mobiles

	Symétrique	Asymétrique
Avantages	- Rapidité (jusqu'à 1000 fois plus rapide) - Facilité d'implantation sur hardware - Taille de clé : 128 bits ($\Rightarrow$ 16 caractères : mémorisable)	- Distributions des clés facilitées : pas d'authentification - Permet de signer des messages facilement - Nombre de clés à distribuer est réduit par rapport aux clés symétriques
Inconvénients	- Nombre de clés à gérer - Distribution des clés (authentification, confidentialité) - Certaines propriétés (p.ex. signatures) sont difficiles à réaliser	- Taille des clés - Vitesse de chiffrement

7.7 Ressources supplémentaires

`http://www.labri.fr/Perso/~betrema/deug/poly/premiers.html`
`http://www.cryptosec.org/article.php3?id_article=10`
`http://members.tripod.com/irish_ronan/rsa/attacks.html`

Chapitre 8

Authentification et intégrité

Jusqu'à présent nous avons considéré le chiffrement pour maintenir un secret. Cependant, d'autres facteurs peuvent entrer en ligne de compte pour garantir une communication sûre. Dans ce chapitre seront illustrées les notions d'authentification de l'expéditeur ainsi que d'intégrité de message.

L'authentification de message repose sur trois concepts :

- La validation de l'identité du créateur du message
- La protection de l'intégrité d'un message
- La non-répudiation de l'origine (résolution de conflit)

Pour assurer cette demande, trois méthodes sont possibles :

- chiffrer le message
- utiliser une fonction de hachage
- utiliser un code d'authentification de message (MAC - Message authentication code)

Dangers à contrer

Plusieurs problèmes peuvent survenir lors de la transmission de messages, aussi bien entre les parties, que face à un pirate. En voici quelques-uns :

- mascarade : insertion des messages d'une source frauduleuse dans le réseau.
- modification du contenu : changements du contenu d'un message, y compris l'insertion, la suppression, la transposition, et la modification.
- modification de séquence (ou d'ordre) : toute modification à un ordre des messages entre les parties, y compris l'insertion, la suppression, et commander à nouveau.
- modification de la synchronisation : retarde ou rejoue des messages.
- répudiation de la source : démenti de transmission de message par source.
- répudiation de la destination : démenti de la réception du message par la destination.

8.1 Par chiffrement du message

8.1.1 Par utilisation d'un cryptosystème symétrique

Si un chiffrement symétrique est employé, le récepteur sait que seul l'expéditeur peut l'avoir créé puisque seuls l'expéditeur et le récepteur connaissent la clé utilisée (ou en tout cas, sont sensés la connaitre).

En ce qui concerne l'intégrité, il faut s'assurer que le contenu n'a pas pu être changé. Si le message a la structure appropriée, on peut utiliser la redondance ou une somme de contrôle pour détecter un changement. On peut également réaliser un contrôle d'erreur interne et externe, comme illustré à la figure 8.1.

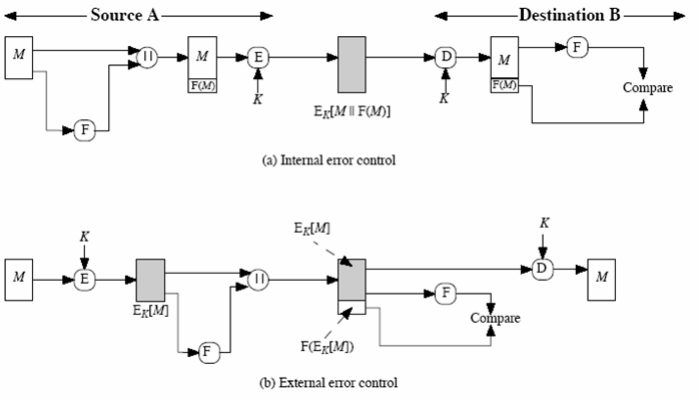

FIG. 8.1 – Controle d'erreur interne et externe

8.1.2 Par utilisation d'un cryptosystème asymétrique

Dans ce cas, on n'a aucune information sur l'expéditeur puisque n'importe qui peut potentiellement utiliser la cléf publique. Cependant si l'expéditeur signe le message en utilisant sa clé privée, et chiffre ensuite avec la clé publique du destinataire, on a le secret et l'authentification. Il est possible d'identifier les messages corrompus mais au coût de deux utilisations de ce chiffrement sur le message.

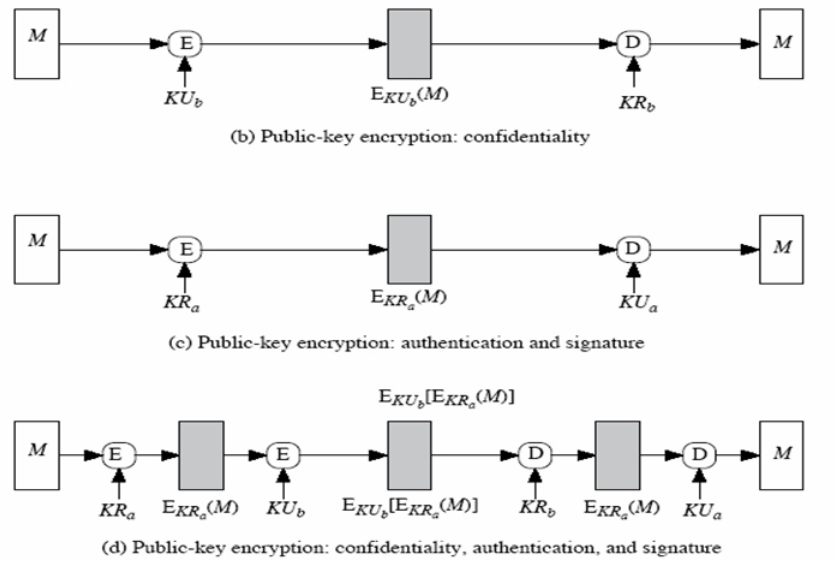

FIG. 8.2 – Authentification lors d'un chiffrement asymétrique

8.2 Fonctions de hachage

On les utilise en cryptographie pour :
– fournir un condensé de taille fixe
– représenter précisément les données : la détection des changements dans le message est simplifiée.

L'intérêt est que cela permet l'usage de la cryptographie asymétrique sans engendrer trop de ralentissement, mais également d'assurer la provenance d'un fichier ainsi que son intégrité. Dans la majorité des cas, elles seront utilisées pour créer une signature numérique. Enfin, la qualité principale de ces fonctions est que les algorithmes sont publics.

8.2.1 Propriétés

On parle de "*haché*", de "*résumé*", ou de "*condensé*" pour nommer la caractéristique d'un texte ou de données uniques. La probabilité d'avoir deux messages avec le même haché doit être extrêmement faible. Le haché ne contient pas assez d'informations en lui-même pour permettre la reconstitution du texte original. L'objectif est d'être représentatif d'une donnée particulière et bien définie (en l'occurrence le message).

Les fonctions de hachage possèdent de nombreuses propriétés :
– Elles peuvent s'appliquer à n'importe quelle longueur de message M
– Elles produisent un résultat de longueur constante
– Il doit être facile de calculer $h = H(M)$ pour n'importe quel message M
– Pour un h donné, il est *impossible* de trouver x tel que $H(x) = h$. On parle de propriété à sens unique.
– Pour un x donné, il est *impossible* de trouver y tel que $H(y) = H(x) \Rightarrow$ résistance **faible** de collision
– Il est *impossible* de trouver x, y tels que $H(y) = H(x) \Rightarrow$ résistance **forte** de collision

8.2.2 Exemple d'utilisations

A la figure 8.3, en (a), le message concaténé à un hash code est chiffré en utilisant le chiffrement symétrique. Puisque seuls A et B partagent la clef secrète, le message doit provenir de A et n'a pas été modifié. Le haché fournit la structure exigée pour réaliser l'authentification. Puisque le chiffrement est appliqué au message entier et au code de hachage, la confidentialité est également fournie.

En (b), seul le haché est chiffré, en utilisant le chiffrement symétrique. Ceci réduit le traitement pour les applications qui n'exigent pas la confidentialité. Ce point illustre en réalité le principe d'une fonction MAC (fonction de hachage + clé secrète).

En (c), à la figure 8.4, seul le code de hachage est chiffré, en utilisant le chiffrement par clef publique et la clef privée de l'expéditeur. Comme (b), ceci fournit l'authentification. Il fournit également une signature numérique, car seul l'expéditeur pourrait avoir produit le code de hachage chiffré.

En (d), toujours à la figure 8.4, si la confidentialité et la signature numérique sont désirées, alors le message et le code de hachage chiffré avec la clé privée sont chiffrés en utilisant une clef secrète symétrique.

Enfin, à la figure 8.5, en (e), on emploie une fonction de hachage mais aucun chiffrement pour l'authentification de message. La technique suppose que les deux parties communicantes partagent une valeur secrète commune S. A calcule la valeur de hachage à la suite de la concaténation de M et de S et ajoute la valeur de hachage résultante à M. Puisque B possède S, il peut recalculer la valeur de hachage à vérifier. Puisque la valeur secrète elle-même n'est pas envoyée, un adversaire ne peut pas modifier un message

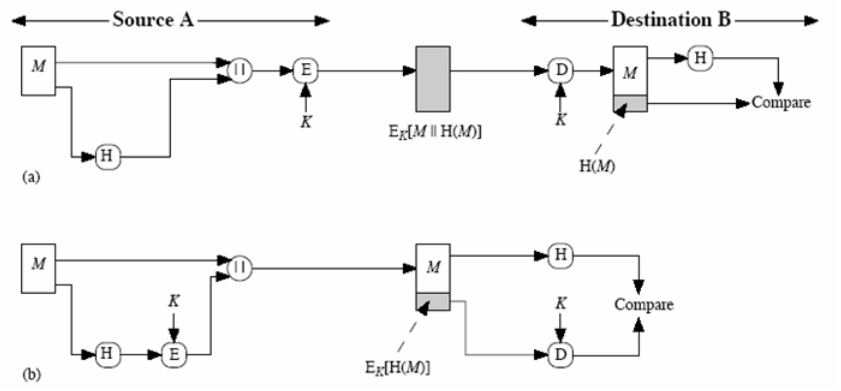

FIG. 8.3 – Exemple d'utilisation des fonctions de hachage

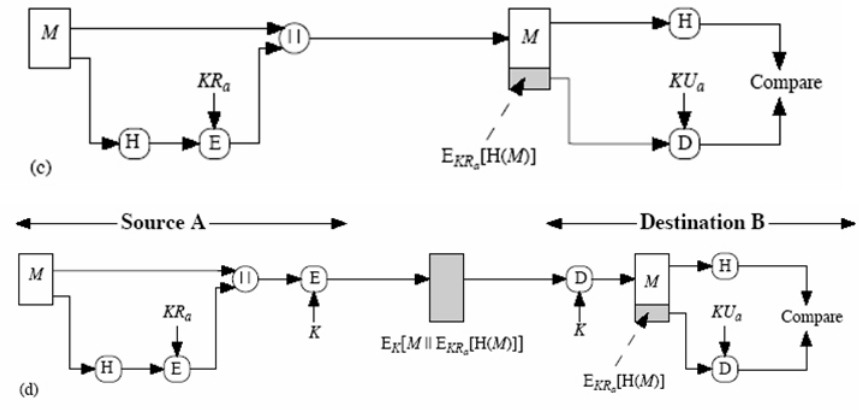

FIG. 8.4 – Exemple d'utilisation des fonctions de hachage

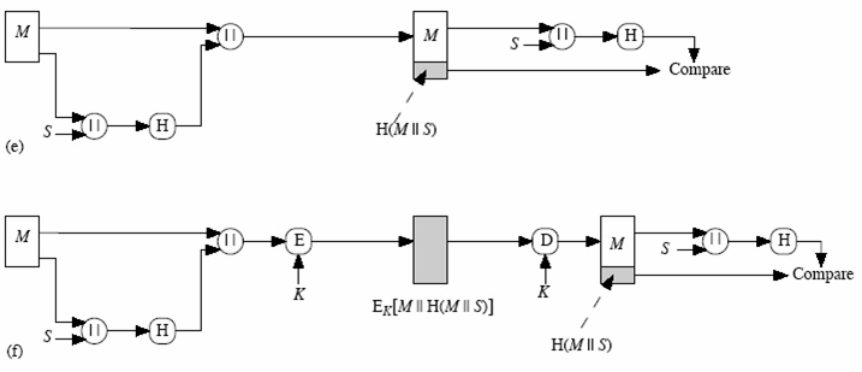

FIG. 8.5 – Exemple d'utilisation des fonctions de hachage

arrêté et ne peut pas produire un message faux.

On peut aussi, comme le montre (f), ajouter la confidentialité à l'approche de (e) en chiffrant le message entier et le code de hachage de M et de S.

8.2.3 Attaque sur les fonctions des hachage

On pourrait penser que des hachés de 64-bit sont sûrs. Pourtant, une technique largement connue, portant le nom de "Paradoxe de l'anniversaire" va nous prouver le contraire.

Ce paradoxe s'illustre par la question suivante : combien de personnes faut-il dans un groupe pour que la probabilité d'avoir deux personnes ayant la même date de naissance atteigne 0.5 ? La réponse, surprenante à première vue, est de 23.

Définition du problème

Pour ce problème, il faut ignorer le 29 février et les années bissextiles. Définissons P(n,k)=Pr[il y a au moins deux éléments communs dans le sous-ensemble de k éléments, et la répartition est uniforme dans l'intervalle 1 à n].

On cherche donc la valeur de k telle que $P(365, k) \geq 0,5$.

Dans un premier temps, cherchons la probabilité qu'il n'y ait pas d'éléments communs, ce que nous noterons $Q(365, k)$. Pour ce faire, nous notons par N, le nombre de manière de tirer k valeurs sans élément commun. Plus explicitement, nous pouvons choisir n'importe quel objet sur 365 pour le premier élément, n'importe lequel sur 364 pour le deuxième, etc. Ainsi, il vient

$$N = \ 365 \ X \ 364 \ X \ 363 \ X \ ... \ X \ (365 - k + 1) = \frac{365!}{(365 - k)!}$$

Si on enlève la restriction comme quoi il n'y a pas de doublons, il y a 365^k possibilités. Donc, la probabilité de ne pas avoir de doublons est donnée par

$$Q(365, k) = \frac{N}{365^k} = \frac{365!}{(365 - k)!(365)^k}$$

Par conséquent,

$$P(365, k) = 1 - Q(365, k) = 1 - \frac{365!}{(365 - k)!(365)^k}$$

Pour $k = 23$, on a $P(365, k) = 0.5073$.

Retour aux fonctions de hachage

Le problème équivalent est le suivant. Soient une fonction de hachage H possédant n résultats possibles différents, et une valeur $H(x)$. Si H est appliqué à k entrées aléatoires, quelle doit être la valeur de k pour que la probabilité qu'au moins une entrée y satisfaisant $H(y) = H(x)$ soit égale à 0.5 ?

Pour une valeur isolée y, la probabilité que $H(y) = H(x)$ est de $1/n$. A contrario, la probabilité que $H(y) \neq H(x)$ est de $[1 - (1/n)]$. Si on génère k valeurs d'entrées, la probabilité qu'il n'y ait aucune correspondance est de $[1-(1/n)]^k$. Donc, la probabilité qu'il y ait au moins une correspondance est égale à $1 - [1 - (1/n)]^k$.

Selon la formule du binôme de Newton,

$$(1 - a)^k = 1 - ka + \frac{k(k-1)}{2!}a^2 - ...$$

Pour de très petites valeurs de a, on peut l'approximer par $(1-ka)$. Ainsi, la probabilité d'avoir au moins une correspondance est approximativement égale à

$$1 - [1 - (1/n)]^k \approx 1 - [1 - (k/n)] = k/n$$

Pour obtenir une probabilité égale à 0.5, nous avons $k = n/2$.

Pour un haché de m bits, le nombre de code possible est 2^m et une probabilité égale à $1/2$ est obtenue lorsque $k = 2^{(m-1)}$.

Utilisation du paradoxe

Dans le cas qui nous occupe, le problème pourrait être adapté comme suit. Soit une fonction de hachage dont la distribution est uniforme et possédant n sorties possibles. Soient deux ensembles de k éléments ($k \leq n$), $X = \{x_1, x_2, ..., x_k\}$ et $Y = \{y_1, y_2, ..., y_k\}$. Quelle est la valeur de k pour que la probabilité d'avoir au moins un élément commun dans les deux ensembles soit égale à 0.5 ?

Comme nous venons de le voir,

$$Pr[\text{pas de correspondance entre } Y \text{ et } x_1] = (1 - (1/n))^k$$

Donc

$$Pr[\text{pas de correspondance entre } Y \text{ et } X] = [(1 - (1/n))^k]^k = (1 - (1/n))^{k^2}$$

Ainsi,

$$Pr[\text{au moins une correspondance entre } Y \text{ et } X] = 1 - (1 - (1/n))^{k^2}$$

Comme $(1 - x) \leq e^{-x}$,

$$Pr[\text{au moins une correspondance entre } Y \text{ et } X] > 1 - (e^{-1/n})^{k^2}$$

$$Pr[\text{au moins une correspondance entre } Y \text{ et } X] > 1 - (e^{-k^2/n})$$

Pour obtenir une probabilité égale à 0.5, il vient

$$1/2 = 1 - (e^{-k^2/n})$$

$$ln(2) = \frac{k^2}{n}$$

et donc,

$$k = 0,83\sqrt{n} \approx \sqrt{n}$$

Pour une fonction de hachage produisant 2^m sorties, on a $k = \sqrt{2^m} = 2^{m/2}$.

Un pirate utilisera donc le paradoxe de la manière suivante :

- l'adversaire produit de $2^{m/2}$ variations d'un message valide tous avec essentiellement la même signification
- l'adversaire produit également de $2^{m/2}$ variations du message frauduleux désiré
- les deux ensembles de messages sont comparés pour trouver une paire donnant les mêmes condensés (la probabilité de trouver une paire est supérieure à 0.5 d'après le paradoxe d'anniversaire)
- faire signer à l'utilisateur le message valide, puis substituer la contrefaçon qui aura une signature valide puisque les condensés sont semblables.

En conclusion, il est nécessaire d'utiliser de plus grands condensés.

La figure 8.6 représente une lettre à laquelle on apporte 2^{37} modifications.

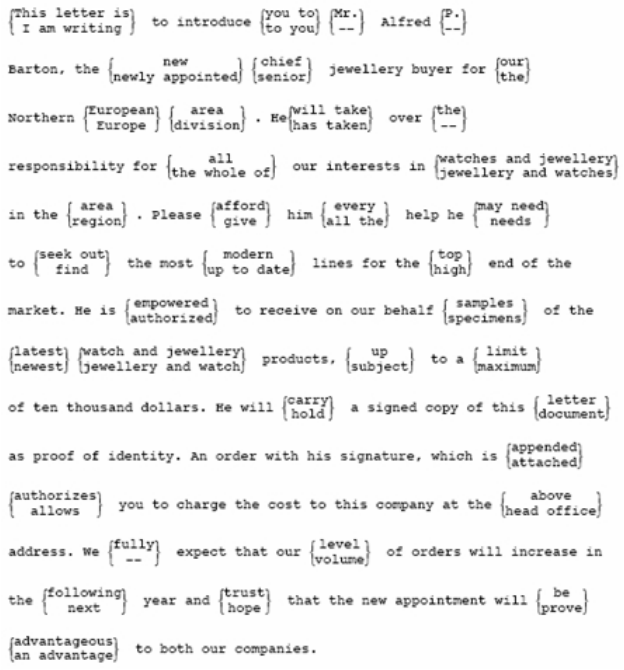

FIG. 8.6 – Message original et message approché

8.3 MAC - Message Authentication Code

Le concept est relativement semblable aux fonctions de hachage. Il s'agit ici aussi d'algorithmes qui créent un petit bloc authentificateur de taille fixe. La grande différence est que ce bloc authentificateur ne se base plus uniquement sur le message, mais également sur une clé secrète.

Tout comme les fonctions de hachage, les MAC n'ont pas besoin d'être réversibles. En effet, le récepteur exécutera le même calcul sur le message et le comparera avec le MAC reçu.

Le MAC assure que le message est inchangé (intégrité) et vient de l'expéditeur (authentification, par l'utilisation de la clé secrète). Il peut également être employé comme un chiffrement supplémentaire (rare) et peut être calculé avant ou après le chiffrement principal, bien qu'il soit généralement conseillé de le faire avant.

La figure 8.7 illustre 3 techniques de base utilisant un MAC. Au point (a), seule l'authentification du message (de la source A) a lieu. En effet, le message est envoyé en clair sur le réseau, et le MAC lui est concaténé. Cependant, par l'utilisation d'une clé secrète, et comme seul A connait cette clé, le destinataire B est sûr que l'expéditeur est bien A. A la réception du message, B applique la fonction MAC sur le message en clair, et vérifie que le MAC résultant correspondant à celui concaténé au message transmis.

Les points (b) et (c) ajoutent la confidentialité. Au point (b), le MAC est concaténé au message, et le tout est chiffré et envoyé à B. Au point (c), seul le message est chiffré. Le MAC est concaténé par la suite, et le tout est envoyé à B. Dans la majorité des cas, on préfèrera utiliser la technique illustrée au point (b).

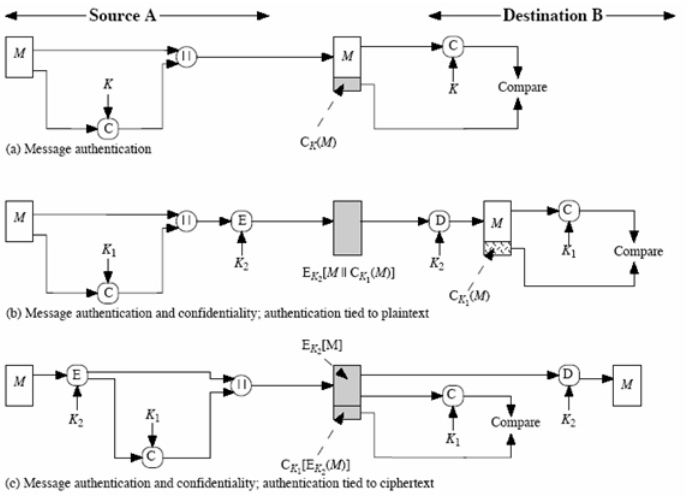

FIG. 8.7 – Fonctions MAC et authentification

8.3.1 Utilisation des chiffrements symétriques

Dans le cas de chiffrements symétriques, on peut employer n'importe quel mode chaîné, le bloc final étant le MAC.

Il y a quelques années, le CBC-MAC était un algorithme relativement répandu dans ce contexte. Il était basé sur le chiffrement DES et, comme l'indique son nom, le mode CBC.

Une norme américaine lui a aussi donné le nom de DAA (Data Authentication Algorithm).

Son fonctionnement est illustré à la figure 8.8. Le vecteur d'initialisation (IV) du mode CBC est ici nul. Les données à authentifier sont symbolisées par les blocs D_1 à D_N. Le DAC (Data Authentication Code) est soit le bloc O_N, soit les M bits les plus à gauche du bloc O_N, avec $16 \leq M \leq 64$.

Cet algorithme n'est plus guère utilisé aujourd'hui pour des raisons de sécurité (le MAC résultant est trop petit). Il sera donc souvent couplé à d'autres algorithmes.

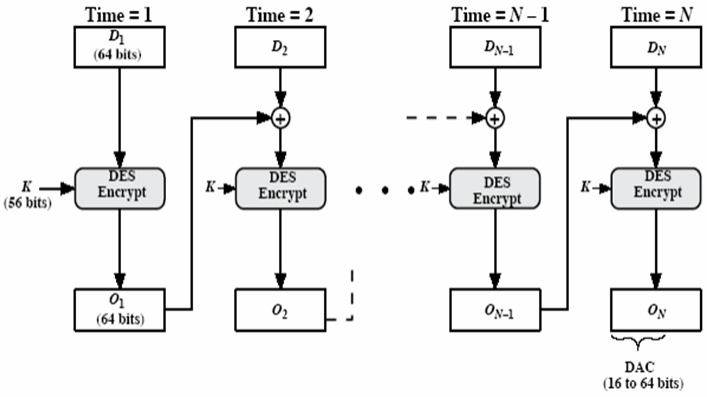

FIG. 8.8 – Data Authentication Algorithm

8.3.2 Sécurité des MAC et fonctions de hachage

La sécurité des algorithmes s'évaluent en fonction des résultats d'une attaque par force brute. L'objectif est que cette dernière soit la meilleure attaque possible.

Sur un haché, la recherche de collisions demande $2^m/2$ opérations. Les hachés de 128 bits étant vulnérables (2005), il est conseillé d'utiliser des condensés de 160 bits.

Dans le cas des MAC, il faut attaquer sur des paires connues message/MAC. On attaquera l'espace de clés (le pirate teste toutes les clés pour un texte donné et vérifie si le MAC obtenu est le même) ou le MAC directement (générer un MAC valide pour un message, ou un message pour un MAC donné). Il est actuellement conseiller d'utiliser des MAC d'une taille minimale de 128 bits.

8.4 Signatures digitales

Nous avons jusqu'ici regardé l'authentification de messages. Mais nous n'avons pas abordé les questions de manque de confiance. En effet, l'authentification des messages protège les deux entités communicantes d'un intrus. Cependant, cela ne les protège en rien l'un de l'autre. En cas de conflit, rien n'empêche la répudiation du ou des messages. Deux exemples illustrent ce principe :

- L'entité A pourrait forger un message donné et dire qu'il vient de B. Il lui suffit pour cela de créer un message et lui appliquer un MAC en utilisant la clé qu'ils ont en commun.
- Le cas inverse peut également exister. B peut nier le fait d'avoir envoyé un message, pour la simple raison que la construction expliquée au point précédent est possible. Il est donc impossible de prouver qu'un tel message a été ou non envoyé par B plutôt que par A.

C'est ici qu'interviennent les signatures numériques. Elles permettent notamment de vérifier l'auteur, la date et l'heure de la signature, d'authentifier le contenu d'un message et peuvent être vérifiées par des tiers pour résoudre des conflits.

8.4.1 Propriétés

Une signature digitale doit
- dépendre du message signé
- employer une information unique propre à l'expéditeur pour empêcher la contrefaçon et le démenti
- être relativement facile à produire, à reconnaître et à vérifier
- être mathématiquement infaisable à forger (par construction de nouveaux messages pour une signature numérique existante, ou par construction d'une signature numérique frauduleuse pour un message donné)
- être facile à stocker

8.4.2 Signatures digitales directes

Ce type de signature implique uniquement l'expéditeur et le récepteur. On suppose que le récepteur dispose de la clé publique de l'expéditeur.

La signature numérique est faite par l'expéditeur en signant le message entier ou le condensé avec sa clé privée. Par la suite, le message signé pourra être chiffré en utilisant la clé publique du récepteur. Il est important de signer d'abord et de chiffrer ensuite le message et la signature. La sécurité dépend de la clé privée de l'expéditeur.

8.4.3 Signatures digitales arbitrées

Un arbitre A est présent et validera n'importe quel message signé, le datera et l'enverra au destinataire. Ce système exige un niveau approprié de confiance en l'arbitre. Il peut être mis en application avec des algorithmes symétriques ou à clé publique. L'arbitre peut ou ne peut pas voir le message.

(a) Conventional Encryption, Arbiter Sees Message
(1) $X \rightarrow A$: $M \| E_{K_{xa}}[ID_X \| H(M)]$
(2) $A \rightarrow Y$: $E_{K_{ay}}[ID_X \| M \| E_{K_{xa}}[ID_X \| H(M)] \| T]$
(b) Conventional Encryption, Arbiter Does Not See Message
(1) $X \rightarrow A$: $ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_X \| H(E_{K_{xy}}[M])]$
(2) $A \rightarrow Y$: $E_{K_{ay}}[ID_X \| E_{K_{xy}}[M] \| E_{K_{xa}}[ID_X \| H(E_{K_{xy}}[M])] \| T]$
(c) Public-Key Encryption, Arbiter Does Not See Message
(1) $X \rightarrow A$: $ID_X \| E_{KR_x}[ID_X \| E_{KU_y}(E_{KR_x}[M])]$
(2) $A \rightarrow Y$: $E_{KR_a}[ID_X \| E_{KU_y}[E_{KR_x}[M]] \| T]$

Notation:
X = sender — M = message
Y = recipient — T = timestamp
A = Arbiter

FIG. 8.9 – Authentification par signature digitale avec arbitre

8.5 Le Zero-Knowledge

L'appellation courante française est "Preuve à divulgation nulle de connaissance". Il s'agit d'une autre méthode d'authentification/identification.

Le concept général est le suivant :

Une entité A (*le prouveur*) prouve sa connaissance d'un secret à une entité B (*le vérifieur*), sans divulguer ce secret.

Cette preuve doit avoir 3 propriétés :

1. *Consistante* : le vérifieur doit accepter la preuve si elle se présente et que le protocole a été correctement suivi.
2. *Significative* : en cas d'erreur, le vérifieur ne pourra pas être trompé, il finira par connaître la vrai nature du prouveur en répétant les expériences.
3. *Sans divulgation de connaissance* : la preuve fournie à B ne lui donne aucune information sur le secret lui-même. On dit que la preuve n'est pas **transférable** :
   – B n'apprend pas le secret
   – Il ne peut pas faire croire à un tiers qu'il dispose lui aussi du secret

Le principe peut être résumé en 3 phases :
– Le prouveur prétend connaître un secret
– Il veut convaincre le vérifieur qu'il connaît ce secret sans pour autant le lui révéler
– Le vérifieur veut être certain que le prouveur ne le trompe pas

Les preuves ZK sont dites *interactives* car elles se réalisent en 3 étapes :

1. Annonce : Le Prouveur envoie un élément destiné à prouver sa connaissance d'un secret
2. Challenge : Le Vérifieur renvoie une question visant à l'en assurer
3. Réponse : Le prouveur répond en utilisant son secret

8.5.1 Exemple : la caverne d'Ali-Baba

L'exemple le plus répandu pour illustrer le principe de Zero-knowledge est le problème de la caverne d'Ali-Baba illustré par la figure 8.10.

1. Bob attend à l'entrée de la caverne. Alice entre et choisit aléatoirement le couloir de gauche (A) ou de droite (B). Elle choisit le couloir B et s'arrête à la porte du fond qui nécessite un mot de passe.
2. Bob entre dans la caverne, s'arrête à la jonction des deux couloirs et dit à Alice par quel couloir il souhaite la voir revenir (il choisit arbitrairement celui de gauche ou de droite).
3. Obligée de passer par la porte, mais connaissant le mot de passe, elle rejoint Bob par le bon couloir et prouve donc qu'elle connaît le mot de passe.

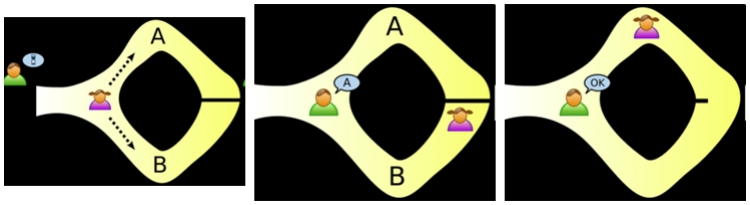

FIG. 8.10 – La caverne

Mais que se passe-t-il si Alice est déjà dans le couloir demandé par Bob ? Alice n'a alors pas besoin de connaître le mot de passe ! Il y a plusieurs possibilités :

– Alice connaît le mot de passe :
si elle se trouve en A et que Bob dit « B », elle satisfait la requête
si elle se trouve en B et que Bob dit « A », elle satisfait la requête
si elle se trouve en A et que Bob dit « A », elle satisfait la requête
si elle se trouve en B et que Bob dit « B », elle satisfait la requête

– Alice ne connaît pas le mot de passe :
si elle se trouve en A et que Bob dit « B », elle ne satisfait pas la requête
si elle se trouve en B et que Bob dit « A », elle ne satisfait pas la requête
si elle se trouve en A et que Bob dit « A », elle satisfait la requête
si elle se trouve en B et que Bob dit « B », elle satisfait la requête

En d'autres termes si Alice connaît le mot de passe, il n'y a jamais de problème. Si Alice ne le connaît pas, il y a une chance sur deux que Bob soit trompé à la première expérience. Ainsi, plus Bob la répétera, plus il pourra croire Alice : après n essais, il y a $1/2^n$ chances que Bob soit trompé (autrement dit, que Alice affirme posséder le secret sans le connaître en réalité).
Une seule erreur, et on peut déduire qu'Alice ne connaît pas le secret. Par contre, 100 bonnes arrivées ne permettent aucune conclusion sûre à 100% bien que la probabilité d'erreur soit extrêmement faible.

Pour reprendre les trois propriétés citées en début de section, on peut dire que

Répondre correctement 100 fois à l'expérience n'offre aucune information sur le secret de la porte (ZK), mais prouve (Consistante), avec une infime probabilité d'erreur (Significative), qu'Alice connaît ce secret.

8.5.2 ZK en pratique

Le principe de ZK est utilisé en tant que schéma d'identification. On l'utilise notamment dans le cadre des signatures digitales, souvent sur smart card.

De nombreux protocoles existent, et de nombreuses variantes (évoluant au gré des découvertes de certaines failles de sécurité). On citera notamment Feige-Fiat-Shamir (FFS) et Gillou-Quisquater (GQ), Schnorr, Ohta-Okamoto ou encore Brickell-McCurley. Les deux premiers seront traités dans le chapitre suivant.

Nous pouvons d'ores et déjà souligner les principaux avantages du schéma ZK :
- sa rapidité face au schéma similaire par RSA (25x pour FFS, 40x pour GQ) : Il n'y a pas d'élévation à une puissance, uniquement des multiplications.
- les signatures fournies sont de taille inférieure
- on peut en conséquence l'employer sur des systèmes légers (p.ex. cartes à puce)

8.6 Ressources supplémentaires

`http://www.bibmath.net/crypto/moderne/sigelec.php3`
`http://www.ssh.fi/support/cryptography/introduction/signatures.html`

Chapitre 9

Algorithmes pour l'authentification et l'intégrité

On constate des similitudes dans l'évolution des fonctions de hachage et des chiffrements symétriques : par la puissance croissante des attaques par force brute, les algorithmes évoluent constamment. On est passé du DES à l'AES dans les chiffrements symétriques et de MD5 à SHA et à Ripemd-160 dans les algorithmes de hachage. Il est aujourd'hui conseillé d'utiliser le SHA-256 (SHA-2), suite aux récentes attaques développées pour SHA-1.

9.1 MD5

Conçu par Ronald Rivest (le R dans RSA), c'est le dernier d'une série (MD2, MD4). Cet algorithme produit un condensé de 128 bits. Il était il y a encore quelques temps l'algorithme de hachage le plus largement répandu. La cryptanalyse et l'attaque par force brute (2004) l'ont affaibli. Ses spécifications sont disponibles sur Internet dans le RFC 1321.

9.1.1 Vue d'ensemble

Le déroulement général de l'algorithme est illustré à la figure 9.1.

1. Complétion : ajout de padding si nécessaire afin que le message ait une longueur de 448 mod 512. Cet ajout a toujours lieu.
2. Ajout de la longueur : on ajoute la longueur réelle du message (sur 64 bits) après les 448 bits. En conséquence, la taille totale du bloc atteint 512 bits. Si la longueur nécessite plus de 64 bits, on ne note que les 64 bits de poids faible.
3. Initialisation : initialiser 4 buffers de 32 bits chacun (A,B,C,D), qui constitue l'IV.
4. Calcul itératif : traiter le message par blocs de 512 bits. Il y a 4 rondes de 16 opérations qui sont réalisées en fonction du bloc (512), du contenu des buffers et de fonctions primitives, tel qu'illustré à la figure 9.3.
5. Le résultat final est obtenu en concaténant les résultats des additions des registres A,B,C,D avec la valeur de CV_q (voir 9.2).

9.1.2 Algorithme

La figure 9.2 schématise le détail de l'algorithme, dont voici les caractéristiques.
- $CV_0 = IV$
- $CV_{q+1} = \sum_{32}[CVq, RF_I(Y_q, RF_H(Y_q, RF_G(Y_q, RF_F(Y_q, CV_q))))]$

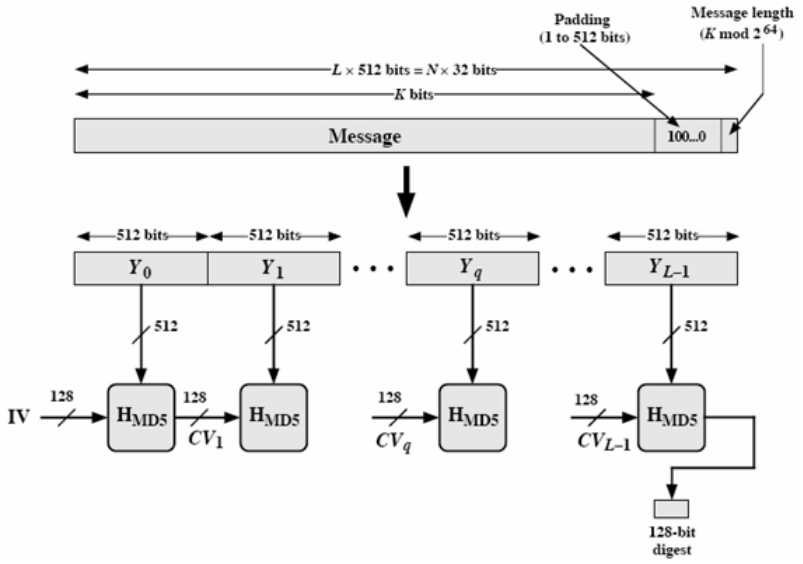

FIG. 9.1 – Vue d'ensemble du MD5

– MD = CV_L

– où :
IV : valeur initiale des registres ABCD
Y_q : le $q^{ème}$ bloc de 512 bits du message
L : le nombre de blocs de 512 bits dans le message
CV_q : variable chainée obtenue par la manipulation du $q^{ème}$ bloc
RF_x : fonction primitive dépendante de la ronde en cours
MD : résultat final
$\sum_{32}$: addition modulo 2^{32}

9.1.3 Fonction de compression

Chaque ronde comprend 16 itérations de la forme :

$$(A, B, C, D) \leftarrow (D, B + ((A + g(B, C, D) + X[k] + T[i]) <<< s), B, C)$$

Les lettres a, b, c, d se rapportent aux 4 buffers, mais sont utilisés selon des permutations variables. Il est à remarquer que cela ne met à jour qu'un seul des buffers (de 32 bits). Après 16 étapes, chaque buffer a été mis à jour 4 fois. La fonction g(b, c, d) est une fonction non-linéaire différente dans chaque ronde (notée F,G, H et I dans la figure 9.2). T[i] est une valeur constante dérivée de la fonction sinus. ($\lll s$) représente un décalage circulaire à gauche (pour chaque buffer séparément) de s bits.

9.1.4 MD4

Précurseur du MD5, il produit également des condensés de 128 bits. Il ne possède que 3 rondes de 16 étapes contre 4 dans MD5. Les buts à réaliser lors de sa conception étaient de le faire résistant aux collisions, et d'apporter une sécurité directe (c'est-à-dire qu'il n'a pas de dépendance envers des problèmes mathématiques). Il devait de plus être aussi rapide, simple et compact que possible.

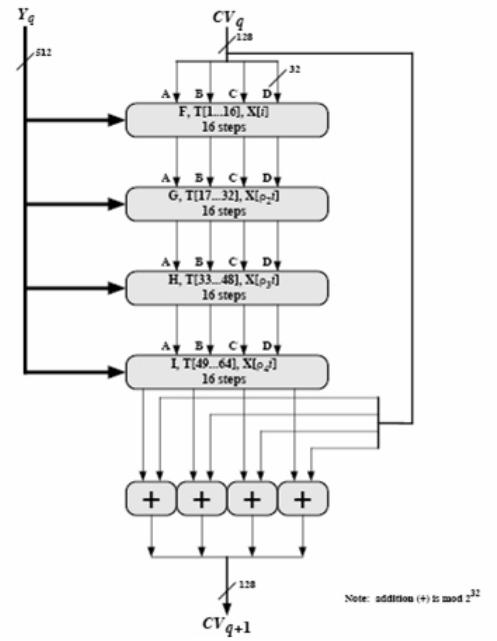

FIG. 9.2 – Rondes du MD5

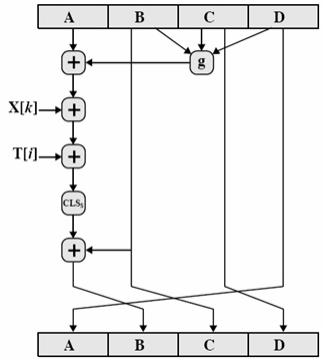

FIG. 9.3 – Fonction élémentaire du MD5

Le MD4 fut également optimisé pour les systèmes "little endian" tels que les PCs. Après plusieurs tests, il s'est avéré que les stations SUN (qui sont "big endian") étaient assez puissantes pour compenser les calculs nécessaires, ce qui décida les développeurs à l'optimiser pour les stations "little-endian".

9.1.5 Sécurité du MD5

Le condensé MD5 dépend de tous les bits du message, ce qui apporte un effet d'avalanche : chaque bit du haché est une fonction de chaque bit d'entrée.

Plusieurs attaques furent recensées. Dès 1992, Berson présenta diverses failles, suivi par Den Boer et Bosselaers en 93, puis Dobbertin en 96. On démontra la présence de collisions sur une ronde, à l'aide de la cryptanalyse différentielle.

Durant l'été 2004, des chercheurs chinois sont parvenus à casser le MD5 en quelques heures. En réalité,

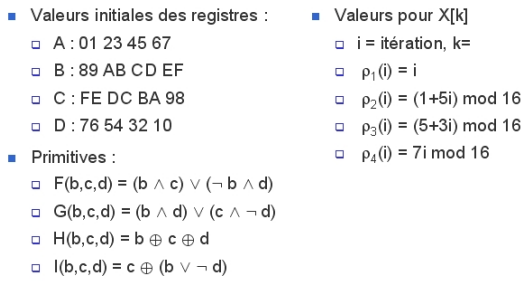

FIG. 9.4 – Valeurs définies pour le MD5

ils ont réussi à créer deux messages possédant le même haché. Le même résultat est aujourd'hui obtenu en quelques minutes, voire quelques secondes. En conclusion, le MD5 n'est plus sûr.

9.2 SHA-1

SHA (Secure Hash Algorithm) a été conçu par NIST et NSA en 1993, et révisé 1995 pour étendre ses capacités en matière de sécurité. Ses spécifications sont publiées dans le RFC 3174. L'algorithme est le SHA, la norme est SHS (Secure Hash Standard). Contrairement au MD5 qui produit des condensés de 128 bits, le SHA produit des valeurs condensées de 160 bits. Jusqu'à 2005, il était l'algorithme généralement préféré pour le hachage, mais des rumeurs de cassage le font peu à peu évoluer vers des versions plus sophistiquées. Il convient aujourd'hui (2009) d'utiliser le SHA-2.

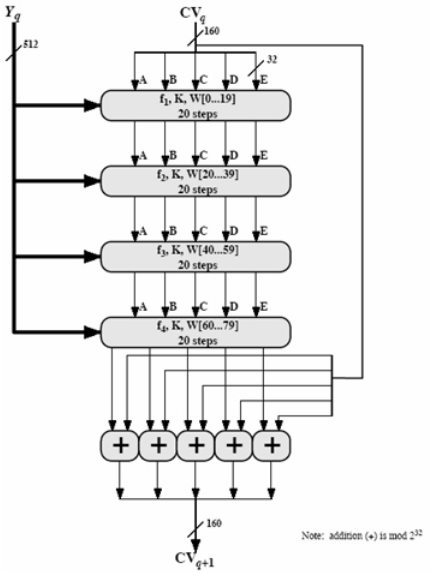

FIG. 9.5 – Rondes du SHA-1

1. Complétion : le message condensé doit être de longueur 448 mod 512
2. Ajout de la longueur : une valeur codée sur 64 bits
3. Initialisation : Initialiser 5 buffers de 32 bits (= 160 bits) - A,B,C,D,E

4. Calcul itératif : 4 rondes de 20 itérations chacune. Ces rondes ont une structure similaire mais utilisent des fonctions primitives différentes (f_1, f_2, f_3, f_4). Le SHA utilise des constantes additives $K_t (0 \leq t \leq 79)$. Le résultat est utilisé pour initialiser les buffers du bloc suivant.
5. Le condensé final constitue le condensé attendu

9.2.1 Fonction de compression

Chaque ronde comprend 20 étapes qui manipulent ainsi les 5 registres :

$$(A, B, C, D, E) \leftarrow (E + f(t, B, C, D) + (A << 5) + W_t + K_t), A, (B << 30), C, D)$$

où A,B,C,D,E se rapportent aux 5 registres, t est le numéro de l'itération, f(t, B, C, D) est une fonction primitive non-linéaire de la ronde pour l'itération t, W_t est dérivé du bloc de message(32 bits) et K_t est une valeur additive.

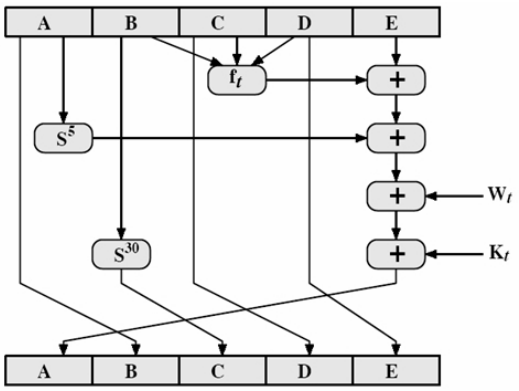

FIG. 9.6 – Fonction de ronde du SHA-1

- Fonctions primitives
  - Travaille sur 3*32 bits et fournit un résultat sur 32 bits
  - $0 \leq t \leq 19$ f_1=f(t,B,C,D) $(B \wedge C) \vee (\neg B \wedge D)$
  - $20 \leq t \leq 39$ f_2=f(t,B,C,D) $B \oplus C \oplus D$
  - $40 \leq t \leq 59$ f_3=f(t,B,C,D) $(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
  - $60 \leq t \leq 79$ f_4=f(t,B,C,D) $B \oplus C \oplus D$
- W_t
  - $0 \leq t \leq 15$: les 16 premières valeurs du bloc
  - $16 \leq t$: $W_t = ((W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}) <<< 1)$

FIG. 9.7 – Valeurs définies pour le SHA-1

Le calcul des W_t, dérivés des blocs de messages est donné à la figure 9.8. Pour ce calcul, il faut remarquer qu'il introduit beaucoup de redondance, ce qui introduit une forte interdépendance des blocs. En conséquence, il est donc relativement difficile de trouver deux messages avec le même haché. La formule résumant le calcul de W_t est donnée à la figure 9.7.

9.2.2 SHA-1 face à MD5

L'attaque par force brute est plus difficile (160 contre 128 bits pour MD5). Mais il est un peu plus lent que MD5 (80 contre 64 itérations). Il fut conçu comme simple et compact, et optimisé pour les processeurs big-endian.

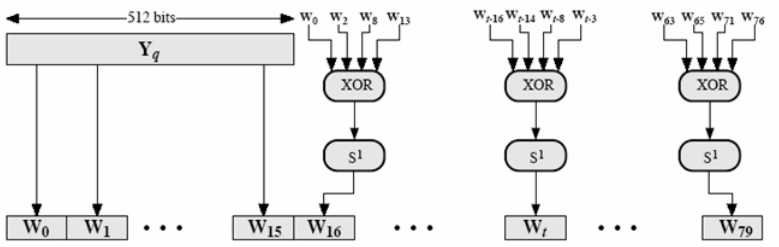

FIG. 9.8 – Calcul du W_t dérivé

	MD5	SHA-1
Haché	128	160
Taille bloc de base	512	512
Nombres d'étapes	64 (4*16)	80 (4*20)
Taille du msg max.	inf.	2^{64}-1
Nb fcts primitives	4	4
Nb const. Add.	64	4
Type d'architecture	Little-endian	Big-endian

FIG. 9.9 – Comparatif MD5 - SHA-1

9.2.3 Revised Secure Hash Standard

Le NIST a publié une révision en 1995 (FIPS 180-2). Cette révision ajoute 3 algorithmes de hachage : SHA-256, SHA-384, SHA-512. La structure et le détail de l'algorithme est semblable à SHA-1, par conséquent la résistance à la cryptanalyse devrait être semblable (hormis l'attaque par force brute qui devient plus longue).

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	80	80	80
Security	80	128	192	256

FIG. 9.10 – SHA et ses évolutions

La figure 9.10 compare les différentes versions du SHA. Toutes les tailles sont mesurées en bits. La sécurité se rapporte au fait qu'une attaque d'anniversaire sur un condensé de message de taille n produit une collision avec un facteur d'approximativement $2^{n/2}$.

9.3 Algorithmes pour les MAC

9.3.1 Fonction de hachage pour les MAC

Il est souhaitable de créer des MACs à partir de fonctions de hachage plutôt qu'à partir de fonction de chiffrement par bloc. Plusieurs raisons sont avancées telles que la rapidité des fonctions de hachage, et

l'absence de contrôle à l'exportation. La différence majeure entre les MAC et les Hachés étant la gestion d'une clé secrète, il faut intégrer cette clé dans l'algorithme de hachage.

Une idée simple fut avancée, et connue sous le nom de HashSur. Le principe était de concaténer directement le message à la clé (i.e. Hash(clé|Message)). Cependant quelques faiblesses ont été trouvées, ce qui a amené au développement de HMAC.

9.3.2 HMAC

Plusieurs objectifs devaient être réalisés tel que le fait d'utiliser sans modifications des fonctions de hachage existantes, de permettre un remplacement aisé de la fonction de hachage au cas où des fonctions plus rapides ou plus sûres seraient trouvées ou exigées, de préserver les performances initiales de la fonction de hachage et d'employer (et manipuler) les clefs de manière simple.

HMAC traite les fonctions de hachage comme des 'black box'. La fonction de hachage devient un module. Il devient alors simple de la remplacer si on met au point une nouvelle fonction de hachage plus rapide ou plus sûre, ce qui apporte une meilleure garantie de sécurité. De par cette modularité, il n'y aura pas de modification à apporter à l'algorithme.

9.3.2.1 Principe de l'algorithme

La fonction HMAC est la suivante et illustrée à la figure 9.11 :

$$\mathrm{HMAC_k(M) = H[(K \oplus opad)||H[(K \oplus ipad)||M]]}$$

Différents éléments interviennent dans l'algorithme :
- H : la fonction de hachage
- IV : un vecteur d'initialisation
- M : le message d'entrée de l'algo (+ padding pour le hachage)
- Y_i : $i^{ème}$ bloc de M
- L : le nombre de blocs de M
- b : le nombre de bits dans un bloc
- n : la longueur du condensé produit
- K : la clé permettant le calcul du MAC
- K^+ : la clé K 'paddée' par des '0' pour obtenir une longueur b
- ipad : 0x36 (répété b/8 fois)
- opad : 0x5C (répété b/8 fois)

Il vient dans l'ordre :

1. Padding de la clé : Ajout de 0 à gauche de K pour créer un flux de b bits (K+)
2. XOR de K+ et ipad pour produire S_i (longueur b bits)
3. Concaténation de M à S_i
4. Application de H à ce flux
5. XOR de K+ et opad pour produire S_o (longueur b bits)
6. Concaténation du haché de l'étape 4 avec S_o
7. Application de H au résultat de l'étape 6

9.3.3 Sécurité de HMAC

La sécurité de HMAC est dépendante de la fonction de hachage sous-jacente. Attaquer HMAC exige
- Soit une attaque par force brute sur la clef utilisée (2^n essais)
- Soit une attaque utilisant le paradoxe de l'anniversaire ($2^{n/2}$ mais puisque on l'utilise avec une clé, on devrait observer un nombre très élevé de messages)

Il faudra choisir la fonction de hachage à utiliser en se basant sur des contraintes de sécurité et de vitesse.

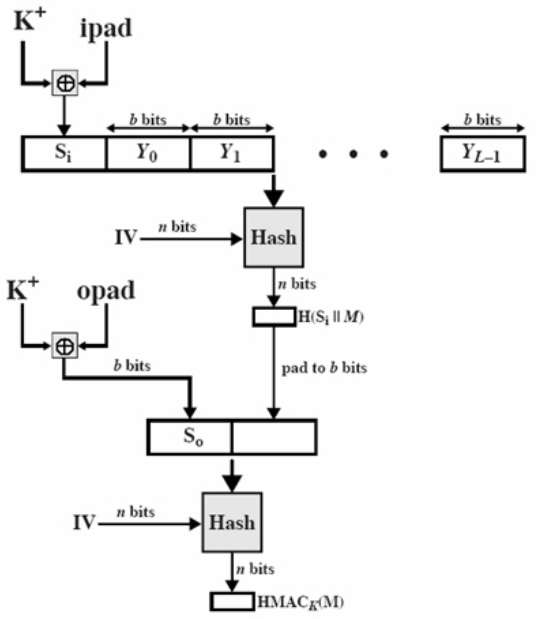

FIG. 9.11 – Algorithme HMAC

9.3.4 Optimisation de HMAC

Il est possible d'améliorer l'algorithme HMAC en utilisant un préprocessing :

$$\mathrm{f}(\mathrm{IV}, (\mathrm{K} \oplus \mathrm{ipad})) \text{ et } \mathrm{f}(\mathrm{IV}, (\mathrm{K} \oplus \mathrm{opad}))$$

La fonction f est le fonction de compression de la fonction de hachage. Elle prend en argument une variable chaînée de n bits et un bloc de b bits et produit une variable chaînée de n bits. Ces quantités ne sont calculées qu'à l'initialisation et quand la clé change. Ces quantités se substituent à l'IV dans la fonction de hachage. Cette technique n'a d'intérêt que lorsque les messages à hacher sont relativement courts.

9.4 Algorithmes de signatures

9.4.1 El Gamal

Soient un nombre p premier et une clé privée $S_K = x$ (où $x < p$).
Soit la clé publique $P_K = (y, g, p)$ où $y = g^x \bmod p$ avec $g < p$.

La signature représentée par (a, b) est calculée comme suit :

- soit un nombre entier k gardé secret tel que $(k, p-1) = 1$
- on calcule $a = g^k \bmod p$
- on calcule b tel que $M = (xa + kb) \bmod (p-1)$

Lors de la vérification, on pourra déterminer une signature valide si

$$y^a a^b \bmod p = g^M \bmod p.$$

Pour assurer la sécurité de la méthode, il faudra utiliser un nouveau k à chaque signature ou chiffrement. Si on utilise plusieurs fois le même k, il est possible de retrouver x aisément.

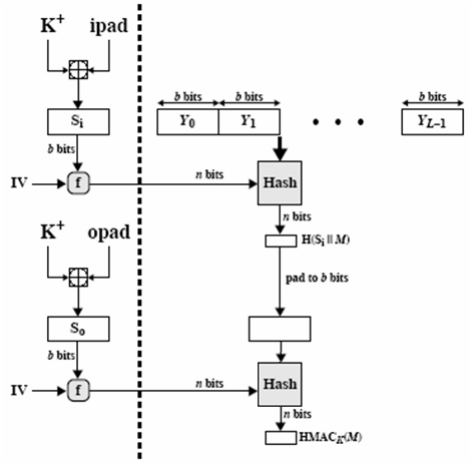

FIG. 9.12 – Préprocessing de HMAC

Exemple : Soit la clé privé $x = 8$ et les valeurs p = 11, g = 2. Il vient pour la clé publique $y = g^x \bmod p = 2^8 \bmod 11 = 3$ c'est-à-dire P_K = (3,2,11).

Soit le message M = 5. Pour l'authentification, prenons un nombre aléatoire k=9, où on a bien (9,10)=1. On calcule alors

- $a = g^k \bmod p = 2^9 \bmod 11 = 6$
- Par Euclide :
  $M = (xa + kb) \bmod (p-1)$
  $5 = (8 * 6 + 9 * b) \bmod 10$
  $b = 3 \rightarrow$ signature $= (a, b) = (6, 3)$

Pour la vérification, il faudra alors s'assurer que $3^6 6^3 \bmod 11 = 2^5 \bmod 11$, ce qui est bien le cas.

9.4.2 Digital Signature Standard (DSS)

Conçu par le NIST et la NSA dans le début des années 90, ce schéma de signature (FIPS-186) fut également approuvé par le gouvernement américain. Il emploie l'algorithme de hachage SHA-1. Comme dans le cas du SHA, on parle de DSS pour la norme, et de DSA pour l'algorithme. Il s'agit en réalité d'une variante d'ElGamal et de Schnorr[1]. La sécurité dépend donc une nouvelle fois de la difficulté de calculer des logarithmes discrets.

9.4.2.1 Algorithme

Ici aussi, différents éléments interviennent dans l'algorithme :

- p : nombre premier de L bits ($512 < L < 1024$)
- L doit être divisible par 64 et $2^{L-1} < p < 2^L$
- q : facteur premier de $p-1$, de 160 bits
- g : g = $h^{(p-1)/q} \bmod p$
- h : $1 < h < p-1$ tel que $h^{(p-1)/q} \bmod p > 1$
- x : valeur entière aléatoire t.q. $0 < x < q$ (clé secrète)

[1]Il s'agit d'un algorithme basé sur un envoi mutuel de valeurs aléatoires.

– y : $y = g^x \bmod p$ (clé publique)
– H(x) : fonction de hachage à sens unique (SHA)

9.4.2.2 Signature du DSS

Alice engendre tout d'abord un k tel que $0 < k < q$. Ce k doit être aléatoire, secret, détruit après utilisation et jamais réutilisé. Alice engendre ensuite les éléments composant la signature :

– r = $(g^k \bmod p) \bmod q$
– s = $(k^{-1}(H(M) + xr)) \bmod q$

Alice envoie ensuite (r, s) à Bob.

9.4.2.3 Vérification de la signature

Bob vérifie la signature en calculant :

– w = $s^{-1} \bmod q$
– $u_1 = (H(M) * w) \bmod q$
– $u_2 = rw \bmod q$
– $v = ((g^{u_1} * y^{u_2}) \bmod p) \bmod q$

Si $v = r$, alors la signature est vérifiée, et l'authentification réalisée.

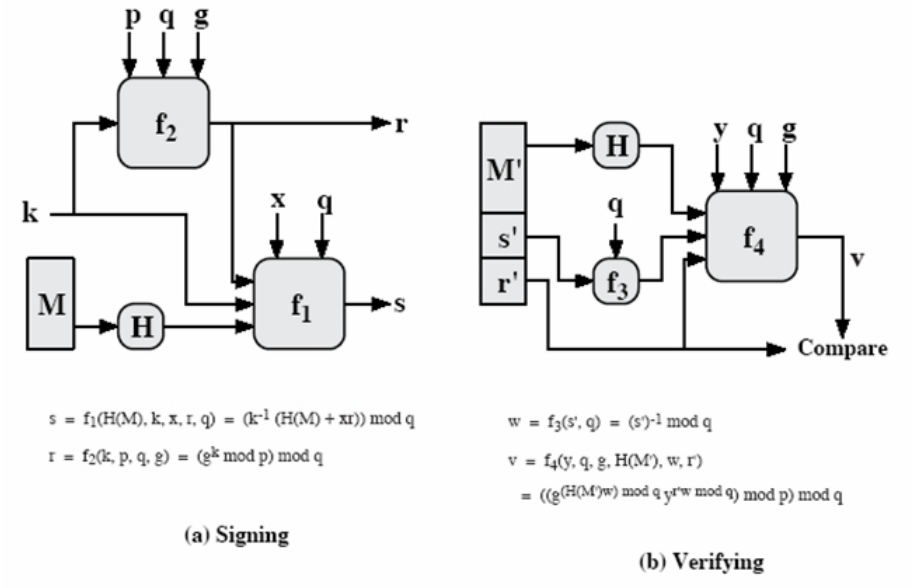

FIG. 9.13 – Illustration des procédés de signature et de vérification de DSA

9.4.3 Comparaison des approches

La figure 9.14 compare les méthodes d'authentification par chiffrement RSA et par DSA. Le KU_G est la "Global Public Key Component" comprenant (p, q, g).

A la différence du RSA, le DSA ne peut être utilisé pour chiffrer des messages ou transmettre des clés.

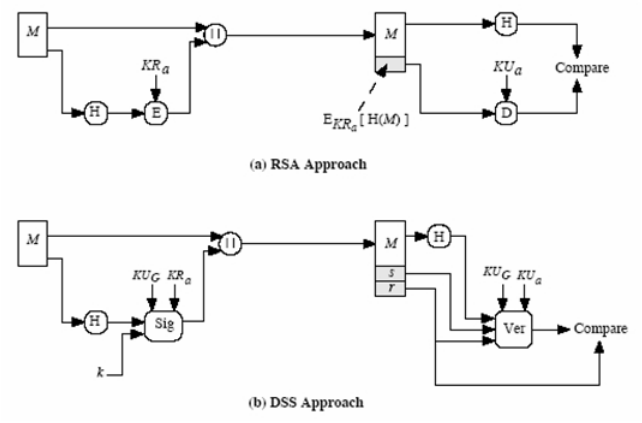

FIG. 9.14 – Signatures par RSA et DSA

9.5 Algorithme ZK

L'exemple donné ci-dessous concerne le protocole de Fiat-Shamir, datant de 1986.

Une autorité détermine aléatoirement un n ($\pm$1024 bits), produit de 2 grands nombres premiers.

Alice choisit un nombre s, avec $(s, n) = 1$. s sera la clé privée d'Alice. Alice calcule ensuite v tel que $v = s^2 \mod n$. v sera la clé publique d'Alice. Ne connaissant pas les facteurs de n, il sera impossible à un correspondant de retrouver la clé privée s à partir de v.

1. Alice détermine aléatoirement un nombre $r < n$
2. Alice calcule $x = r^2 \mod n$, et envoie x à Bob
3. Bob renvoie un bit aléatoire b à Alice
4. Alice envoie $y = r.s^b \mod n$ à Bob.
5. Bob vérifie que $y^2 = x.v^b \mod n$

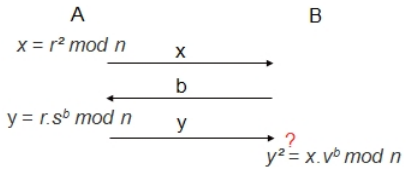

FIG. 9.15 – Fiat-Shamir

Un prouveur digne de confiance doit pouvoir répondre aux deux requêtes ($b = 0$ ou $b = 1$).

Si Alice ne connaît pas s, elle pourrait corrompre Bob en choisissant un r particulier. Mais comme Bob va répéter l'opération T fois, le r ne pourra pas répondre correctement aux deux questions simultanément.

En contexte réel, on travaille avec des vecteurs de données de longueur K, ce qui augmente la sécurité. La clé publique est $(v_1, v_2, \ldots, v_K)$, la clé privée est $(s_1, s_2, \ldots, s_K)$, et le vecteur de bits $(b_1, b_2, \ldots, b_K)$.

La probabilité que Bob soit trompé est de $1/2^{KT}$. En pratique, on conseille k=6 et T=5.

Chapitre 10

La gestion des clés

La gestion des clés est principalement constituée de quatre domaines :

1. La génération des clés : il faut prendre garde aux caractères choisis, aux clés faibles, ... et veiller à utiliser des générateurs fiables,
2. Le transfert de la clé : l'idéal est de se rencontrer, ou d'utiliser un canal de transmission protégé. Mais cela est souvent impossible. Aussi, si A et B ont des communications sûres avec un tiers C, ce dernier peut relayer la clé entre A et B. Un tiers est un intermédiaire de confiance, à qui tous les usagers font confiance, pour négocier l'établissement de transmissions sûres entre elles,
3. La vérification des clés : par hachage, ou utilisation de certificats,
4. Le stockage des clés : que ce soit dans des fichiers, sur supports extérieurs, par surchiffrement, ...

10.1 Distribution des clés

10.1.1 Clés symétriques

Dans ce cas, il est nécessaire pour les deux usagers de partager une clé secrète commune. Bien souvent, l'échec d'un système sûr est dû à une rupture dans le schéma de distribution des clés. Comment distribuer sûrement cette clé ?

- Physiquement : par une rencontre, un canal de transmission protégé, ...
- Utiliser un tiers de confiance. Celui-ci choisit et fournit la clé,
- Utiliser une ancienne clé pour chiffrer une nouvelle clé (ce qui suppose cependant un échange préalable de cette ancienne clé),
- Distribution automatique de clés à la demande des utilisateurs. Cette solution existe, mais elle nécessite une totale confiance au système.

La figure 10.1 illustre le protocole Needham-Schroeder, dont nous reparlerons dans le chapitre consacré à l'authentification des parties. Ce scénario suppose que les deux entités (A et B) possèdent chacune une clé secrète avec le KDC (l'autorité de confiance du système).

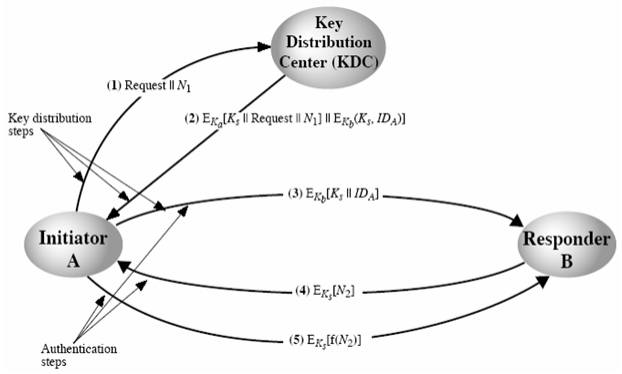

FIG. 10.1 – Exemple d'une distribution de clés dans le cas symétrique (Needham-Schroeder).

10.1.2 Clés asymétriques

Le chiffrement par clé publique permet de résoudre les problèmes de distribution de clés secrètes. Malgré tout, la distribution des clés publiques continue de poser problème, principalement au niveau de l'authentification des utilisateurs liés à ces clés. Il existe quatre solutions permettant un transfert des clés dans le cas asymétrique :

- Annonce publique
- Annuaire publiquement disponible
- Autorité de clés publique
- Certificats de clé publique

10.1.2.1 Annonce Publique

La distribution des clés publiques se fait directement aux destinataires ou par broadcast à la communauté. Il est par exemple possible d'apposer les clefs PGP aux emails ou les poster dans des newsgroups ou mailing-lists. Mais le risque majeur avec cette méthode est la contrefaçon : n'importe qui peut créer une clef en prétendant être quelqu'un d'autre et la publier. La mascarade continuera tant que la contrefaçon n'est pas découverte.

10.1.2.2 Annuaire Publique

On enregistre ici des clés dans un annuaire public, ce qui implique de faire confiance à cet annuaire. Ce dernier doit avoir plusieurs propriétés :

- Il doit contenir les entrées {nom, clef publique},
- Il doit être possible de s'inscrire de manière sécurisée dans l'annuaire,
- On doit pouvoir remplacer la clef à tout moment,
- L'annuaire doit être publié périodiquement,
- Il devrait également permettre la consultation électronique.

Même si ce schéma est clairement plus sûr que les annonces publiques individuelles, il reste vulnérable. Il est en effet nécessaire que l'annuaire soit sécurisé. Dans le cas contraire, un individu pourrait détourner l'annuaire et fournir des clefs publiques contrefaites, voire à ne pas transmettre les clés correspondant aux demandes des entités communicantes.

10.1.2.3 Autorité de clés publique

Il s'agit de renforcer le contrôle de la distribution des clefs à partir de l'annuaire. Il dispose des mêmes propriétés que ce dernier. Cependant, la sécurité est renforcée. En effet, dans le cas présent, chaque entité dispose de la clé publique de l'autorité. Ainsi, lorsqu'une entité désirera obtenir la clé publique d'un correspondant, il enverra une requête à l'autorité. Celle-ci contiendra la requête proprement dite et un marqueur temporel (*timestamp*). En retour, l'autorité renverra la clé demandée, le timestamp pour prouver le non-rejeu d'un ancien message, le tout chiffré avec sa clé privée. De cette manière, l'entité A, possédant la clé publique de l'autorité, pourra vérifier la bonne provenance de la clé publique de B. L'entité B pourra pratiquer de la même manière.

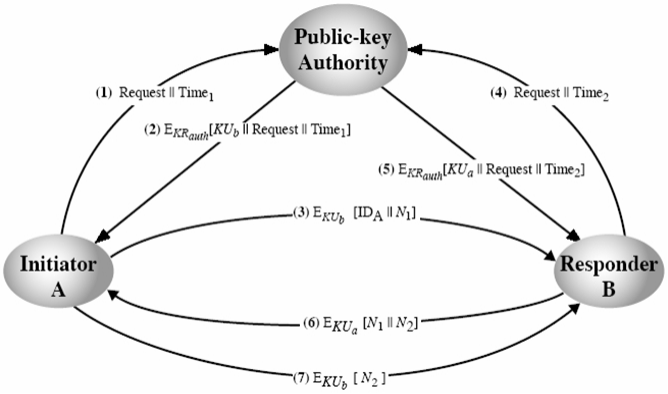

FIG. 10.2 – Utilisation d'une autorité de clés publiques

10.1.2.4 Certificats de clés publiques

Cette approche alternative (1978) consiste à employer des certificats pouvant être utilisés par des participants pour échanger des clefs sans entrer en contact avec une autorité de clés publiques, d'une façon fiable comme si les clefs avaient été obtenues directement à partir d'une telle autorité.

Chaque certificat contient une clef publique et des informations supplémentaires (la période de validité, les droits d'utilisation, etc.). Il est créé par une autorité de certification (*Certificate Authority*, CA) et est donné au participant disposant de la clef privée assortie. Un participant fournit l'information sur sa clé à un autre en transmettant son certificat. Son contenu est signé par la clé privée du CA. Cette situation est illustrée à la figure 10.3

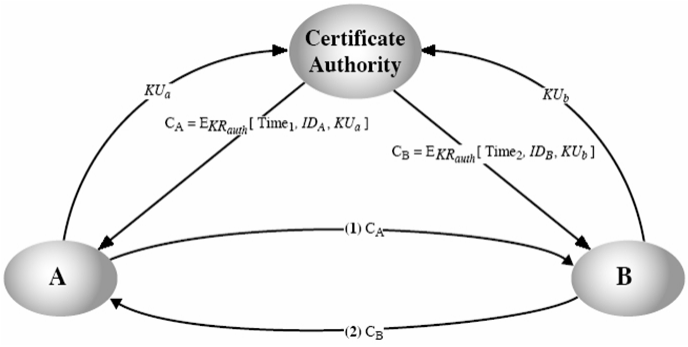

FIG. 10.3 – Certificats de clés publiques

Plusieurs propriétés permettent de garantir une meilleure sécurité :
- N'importe quel participant peut lire un certificat pour déterminer le nom et la clé publique du propriétaire du certificat.
- N'importe quel participant peut vérifier que le certificat provient réellement du CA et n'est pas contrefait.
- Seul le CA peut créer et mettre à jour des certificats.
- Tous les participants peuvent vérifier la validité des certificats (Denning - 1983)

Pour mieux comprendre ce concept, illustrons le par un exemple : si vous commandez des cds sur Internet, comment être sûr que vous envoyez bien votre numéro de carte bleue au commerçant, et non à un pirate qui aurait usurpé son identité et donné sa propre clé publique ?

Pour passer un examen, il vous faut prouver votre identité grâce à une carte d'identité, un passeport ou un permis de conduire. Un organisme supérieur (l'Etat) a signé ces certificats, s'assurant auparavant (par un acte de naissance,...) qu'il s'agit bien de vous.

Les certificats numériques fonctionnent sur le même principe. Alice veut certifier que sa clé publique lui appartient. Elle envoie sa clé à un organisme de certification, ainsi que différentes informations la concernant (nom, email, etc...). Cet organisme vérifie les informations fournies par Alice, et ajoute au certificat son propre nom, une date limite de validité, et surtout une signature numérique. Cette signature est réalisée grâce à sa clé privée et à un algorithme de hachage laissé au choix, bien que le RSA et le SHA soit maintenant préféré. En résumé, l'autorité de certification fournit un certificat tel que

$$C_A = E_{KR_{auth}}[T, ID_A, KU_a]$$

où $E_{KR_{auth}}$ représente la signature apposée au certificat, T est un timestamp, ID_A est l'ensemble des informations propres à l'entité A et KU_a est la clé publique de A.

Lorsque Bob veut envoyer un message à Alice ou simplement vérifier la validité du certificat que A lui a envoyé, il applique la clé publique de l'autorité de certification. Cette action permet de vérifier que le certificat est bien authentique :

$$D_{KU_{auth}}[C_A] = D_{KU_{auth}}[E_{KR_{auth}}[T, ID_A, KU_a]] = (T, ID_A, KU_A)$$

10.1.3 Les certificats - Service d'authentification X.509

Il s'agit d'une partie de la norme de service d'annuaire X.500, qui regroupe des serveurs distribués maintenant une base de données d'informations. Le service définit le cadre pour des services d'authentification, internationalement admis pour construire un certificat de clé publique. L'annuaire peut stocker des certificats de clé publique et les clés publiques des utilisateurs correspondants. Ces certificats sont également signés par une autorité de certification. Il utilise la cryptographie à clé publique et les signatures digitales. Aucun algorithme de chiffrement n'est imposé, mais le RSA est recommandé.

10.1.3.1 Obtention d'un certificat

N'importe quel utilisateur ayant accès au CA peut obtenir un certificat de celui-ci, mais seul le CA peut modifier un certificat. Comme il est difficile de forger un certificat, on peut sans trop de risque le placer dans un annuaire public.

La figure 10.4 illustre le format d'un certificat. L'exemple donné concerne un certificat signé par RSA et MD5.

10.1.3.2 Hiérarchie de certificats

On émet ici une hypothèse : si les deux utilisateurs partagent un CA commun alors on suppose qu'ils connaissent sa clef publique.

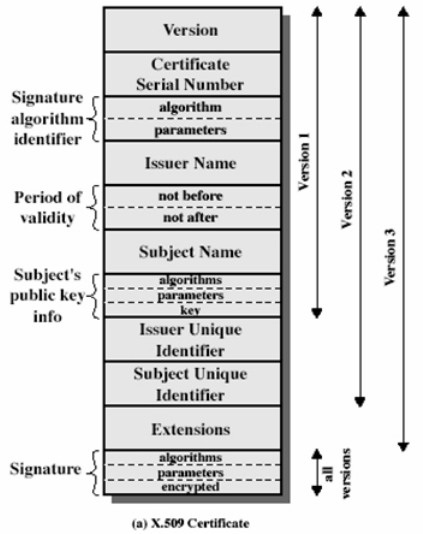

FIG. 10.4 – Format standard et exemple de certificat X.509

Sur la figure 10.5, chaque CA possède des certificats pour les clients (vers l'avant) et le parent (vers l'arrière). Chaque client fait confiance aux certificats parents. On parle alors de hiérarchie de CA. Le principe est d'employer les certificats liant les membres de la hiérarchie pour valider d'autres CA. L'objectif est de permettre la vérification de n'importe quel certificat d'un CA par des utilisateurs de tout autre CA dans la hiérarchie.

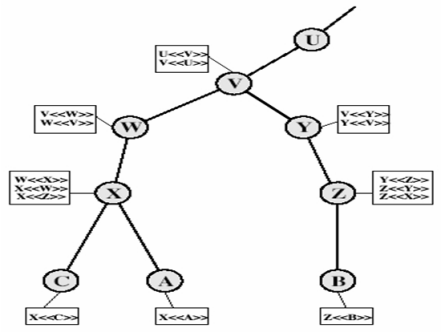

FIG. 10.5 – Hiérarchie de certificats

10.1.3.3 Révocation de certificats

Les certificats ont une période de validité. On doit pouvoir le retirer avant l'échéance car il se peut que la clef privée de l'utilisateur soit compromise, que l'utilisateur ne soit plus certifié par ce CA ou encore que le certificat du CA soit compromis.

Les CA maintiennent donc une liste des certificats retirés. Cette liste porte le nom de "liste de révocation de certificats" (CRL) et est vérifiable par tout utilisateur.

L'obtention des CRL sera réalisée auprès d'un CDP (*CRL Distribution Point*), sorte de serveur principal maintenant les CRL à jour.

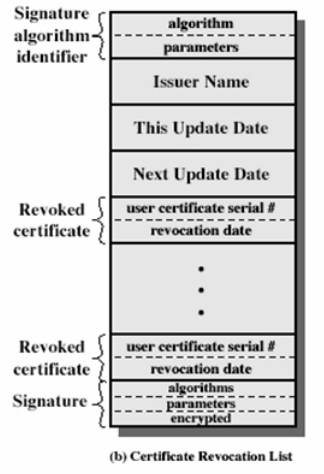

FIG. 10.6 – Format d'une CRL

Notons au passage que pour faciliter les vérifications des certificats, il existe différents protocoles dont le plus répandu est l'OCSP (*Online Certificate Status Protocol*, voir RFC 2560).

L'OCSP consiste en un échange de type requête/réponse entre un client (par exemple une banque) et un serveur de validation/vérification, appelé *répondeur* OCSP. Ce protocole est avantageux à plus d'un titre puisqu'il permet, par le travail du serveur, de diminuer le trafic réseau et la charge de travail du client (qui n'a plus à analyser la CRL).

Précisons enfin qu'il existe de nombreuses méthodes permettant de maintenir ces listes de certificats à jour. Nous avons cité l'utilisation basique d'une CRL et l'OCSP, mais on peut aussi nommer la Delta-CRL ou les CRL DP[1].

10.1.3.4 Procédures d'authentification

X.509 inclut trois procédures alternatives d'authentification :
- Authentification à sens unique (one-way) (telle que les mails)
- Authentification à 2 passages (two-way) (par utilisation de timestamps)
- Authentification à 3 passages (three-way) (pas de timestamps)

One-Way Un seul message ($A \rightarrow b$) est utilisé pour établir à la fois
- l'identité de A et l'origine du message,
- le destinataire du message,
- l'intégrité et l'authenticité du message.

Le message inclut l'horodateur (t_A), le nonce (r_A), l'identité de B (ID_B) et doit être signé par A (sgnData)

Two-Way Deux messages ($A \rightarrow b, B \rightarrow a$) sont utilisés. Le principe est le même que pour l'authentification One-Way mais détermine plus de caractéristiques des entités :
- l'identité de B et que cette réponse est de B,
- la réponse est destinée à A,

[1] Consulter "Méthodes de Révocation de Certificats : Etudes et Comparaisons" de BEKARA et MAKNAVICIOUS pour plus d'informations.

– l'intégrité et l'authenticité de la réponse.

La réponse inclut le nonce de A, l'horodateur et le nonce de B.

Three-Way Trois messages sont échangés ($A \rightarrow b, B \rightarrow a, A \rightarrow b$) permettant une authentification sans horloge synchronisée.

La réponse de A à B contient la copie signée du nonce de B ce qui signifie que les horodateurs n'ont pas besoin d'être vérifiés ou comptés.

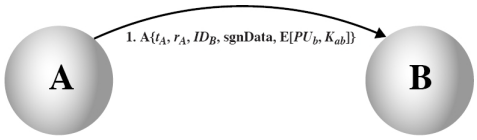

FIG. 10.7 – Authentification unidirectionnelle (One-Way)

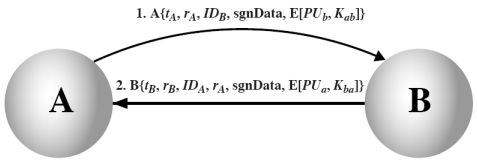

FIG. 10.8 – Authentification bidirectionnelle (Two-Way)

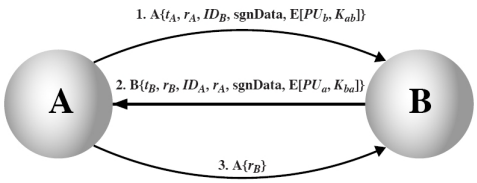

FIG. 10.9 – Authentification tridirectionnelle (Three-Way)

10.1.3.5 X.509 Version 3

Avec l'évolution des besoins et des technologies, est apparue la nécessité d'informations additionnelles telles que l'inclusion d'email/URL, de détails de politique ou de contraintes d'utilisation. La particularité de cette nouvelle version est qu'elle définit une méthode générale d'extension plutôt que la création de nouveaux champs statiques. Les extensions sont de plus optionnelles, ce qui permet de ne pas modifier toute la structure. Les logiciels doivent donc spécifier s'ils utilisent ou non ces extensions.

Celles-ci se composent principalement de trois éléments :
– un identificateur (numéro de l'extension utilisée),
– un marqueur d'importance,
– une valeur (ce que contient l'extension en question).

Cette nouvelle version apporte plus de flexibilité mais moins de compatibilité en contrepartie.

10.1.4 Clés de session

La distribution de clés publiques bien que simple et permettant la confidentialité et l'authentification, reste lente. L'objectif restant de protéger le contenu d'un message, une solution plus rapide est d'utiliser un système hybride permettant l'échange final d'une clé de session.

Merkle

Cette solution fut proposée par Merkle en 1979. Le déroulement est le suivant :

1. A produit une nouvelle paire de clés publique/privée provisoire
2. A envoie à B sa clef publique (et son identité)
3. B produit une clef K de session et l'envoie à A (la clé est chiffrée au moyen de la clef publique fournie par A)
4. A déchiffre la clef de session et tous les deux l'emploient

Cette technique est illustrée à la figure 10.10.

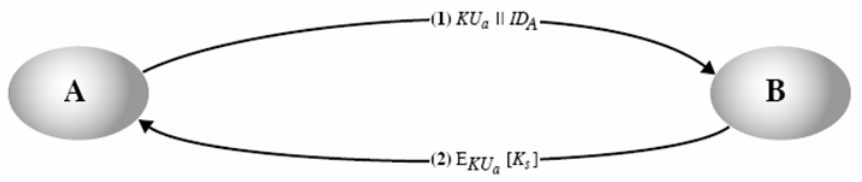

FIG. 10.10 – Echange de clés par le principe de Merkle

Le problème est qu'un adversaire peut arrêter et personnifier les deux moitiés de protocole (attaque man-in-the-middle).

Attaque Man-in-the-Middle Si un adversaire E a la commande de la voie de transmission intervenante, alors E peut compromettre la communication de la façon suivante sans être détecté :

1. A produit une paire de clés publique/privée $\{KU_a, KR_a\}$ et transmet un message destiné à B se composant de KU_a et d'une marque de A, ID_A.
2. E arrête le message, crée sa propre paire de clés publique/privée $\{KU_e, KR_e\}$ et transmet $KU_e||ID_A$ à B.
3. B produit une clef secrète, K_s, et transmet $E_{KU_e}(K_s)$.
4. E arrête le message, et apprend K_s par $D_{KR_e}(E_{KU_e}(K_s \textit{ de calcul}))$
5. E transmet $E_{KU_a}(K_s)$ à A.

Le résultat est qu'A et B connaissent K_s et ignorent que K_s a été également donné à E. A et B peuvent maintenant échanger des messages en utilisant K_s. E n'interfère plus activement sur la voie de transmission mais écoute clandestinement. Connaissant K_s, E peut déchiffrer tous les messages alors qu'A et B ignorent le problème.

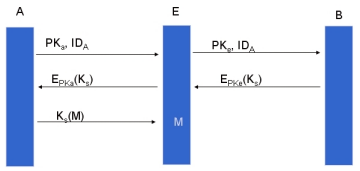

FIG. 10.11 – Attaque Man in the Middle

10.1.5 Distribution de clés secrètes avec confidentialité et authentification

Il est possible d'intégrer de l'authentification dans les échanges (on suppose que A et B se sont échangés préalablement leur clef publique). Ce processus est illustré à la figure 10.12 :

1. A utilise la clef publique de B pour chiffrer un message à B contenant une marque de A (ID_A) et un nonce (N_1), qui est employé pour identifier cette transaction uniquement.
2. B envoie un message à A chiffré avec KU_a contenant le nonce de A (N_1) ainsi qu'un nouveau nonce produit par B (N_2). Puisque seul B pourrait avoir déchiffré le message (1), la présence de N_1 dans le message (2) assure A que le correspondant est B.
3. A retourne N_2, chiffrés en utilisant la clef publique de B, pour assurer B que son correspondant est A.
4. A choisit une clé de session K_s secrète et envoie $M = E_{KU_b}[E_{KR_a}[K_s]]$ à B. Le chiffrement de cette clé avec la clef privée de A assure que seul A peut l'avoir envoyée.
5. B calcule $D_{KU_a}[D_{KR_b}[M]]$ pour récupérer la clef secrète.

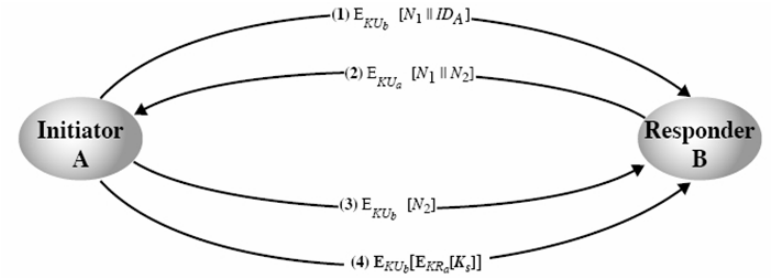

FIG. 10.12 – Distribution de clés secrètes avec confidentialité et authentification

10.2 Echange des clés - Diffie Hellman

C'est une méthode pratique pour l'échange public d'une clef secrète (ou de session). Contrairement à ce que beaucoup pensent, il ne peut pas être employé pour échanger directement un message arbitraire.

La méthode de Diffie-Hellman (DH) est basée sur l'élévation à une puissance dans un champ fini, ce qui est facile à calculer. La sécurité se fonde sur la difficulté de calculer des logarithmes discrets, problème difficile comme dans le cas d'El Gamal. La valeur de la clef dépend des participants (et de l'information sur leurs clés privée et publique).

10.2.1 Principe

Soient A et B, les deux parties de la communication. A génère un nombre premier p et un primitif a (p et a sont publics).

Définition 1 (Primitif) *Soient p premier et $a < p$. On dit que a est un primitif de p si $\forall b \in (1, p-1)$, $\exists g$ tel que $a^g \equiv b \bmod p$.*[2]

□

Une fois ces deux valeurs rendues publiques, A et B génèrent chacun un nombre aléatoire : x_A (< p) et x_B (< p) (gardés secrets).

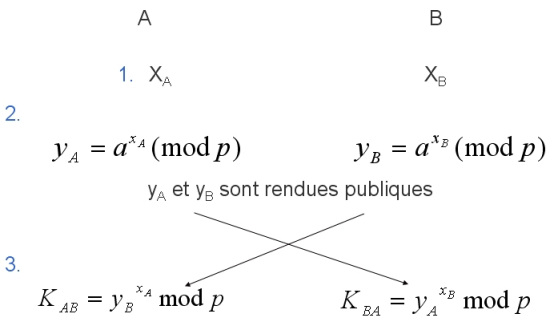

FIG. 10.13 – Principe de DH

A et B partagent ainsi la même clé puisque $K_{AB} = K_{BA}$. En effet :

$$\begin{aligned} K_{AB} &= y_B^{x_A} \bmod p \\ &= (a^{x_B})^{x_A} (\mathrm{mod} p) \\ &= a^{x_A x_B} (\mathrm{mod} p) \\ &= K_{BA} \end{aligned}$$

Exemple : Alice et Bob souhaitent échanger une clé.

1. Soient les valeurs $p = 353$ et $a = 3$ rendues publiques.
2. Ils sélectionnent chacun une valeur aléatoire : A choisit $X_a = 97$, B choisit $X_b = 233$.
3. Ils calculent la clé publique : $Y_A = 3^{97} \bmod 353 = 40$(pour Alice) et $Y_B = 3^{233} \bmod 353 = 248$(pour Bob).
4. Et calculent finalement la clé de session : $K_{AB} = (Y_B)^{X_a} \bmod 353 = 24897 = 160$ (pour Alice) et $K_{AB} = (Y_A)^{X_b} \bmod 353 = 40233 = 160$ (pour Bob).

Un imposteur pourrait cependant essayer de trouver K_{AB} à partir de y_A, de y_B, de a et de p en calculant

$$K_{AB} = y_B^{\log_a y_A} (\mathrm{mod} p)$$

mais la sécurité de cet algorithme repose sur le fait qu'il est impossible de calculer $\{log_a y (\mathrm{mod} p)\}$ en algèbre modulaire dans un temps raisonnable. Notons cependant que l'authentification de A et B n'est pas assurée par cet algorithme (sensible à l'attaque MITM).

10.2.2 Protocole Station à Station (STS)

Diffie-Hellman est sensible à l'attaque MITM (un ennemi peut s'immiscer dans l'échange et se faire passer pour le destinataire légal envers chacune des entités communicantes). La solution est d'insérer de

[2]Exemple : Pour p=11, 2 est un primitif. En effet, $2^{10} \equiv 1 \mod 11$, $2^1 \equiv 2 \mod 11$, $2^8 \equiv 3 \mod 11$, $2^2 \equiv 4 \mod 11$, $2^4 \equiv 5 \mod 11$, $2^9 \equiv 6 \mod 11$, $2^7 \equiv 7 \mod 11$, $2^3 \equiv 8 \mod 11$,$2^6 \equiv 9 \mod 11$ et $2^5 \equiv 10 \mod 11$.

l'information relative soit à l'identité de l'expéditeur, soit au message échangé. Plusieurs moyens existent, dont les signatures digitales ou les MACs.

Le protocole STS est illustré à la figure 10.14. Dans le cas présent, la fonction Sign est calculée au moyen de la clé définie par le nombre secret de B, valeur que pourra vérifier A grâce à la propriété d'inversibilité. Comme ces valeurs secrètes sont inconnues de l'ennemi, ce dernier ne pourra plus feindre son identité.

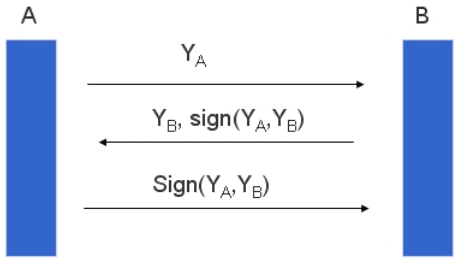

FIG. 10.14 – Protocole STS

10.3 Ressources supplémentaires

`http://home.ecn.ab.ca/~jsavard/crypto/pk0503.htm`
`http://home.ecn.ab.ca/~jsavard/crypto/mi0607.htm`
`http://developer.netscape.com/docs/manuals/security/pkin/contents.htm`
`http://www.itl.nist.gov/`

Chapitre 11

IPSEC

11.1 Présentation

IPSec n'est pas un protocole simple. Il fournit un cadre général qui permet à une paire d'entités communicantes d'employer n'importe quel algorithme fournissant la sécurité appropriée pour la communication. Cette technologie a pour but d'établir une communication sécurisée (un tunnel) entre des entités éloignées, séparées par un réseau non sécurisé voire public comme Internet, et ce de manière quasi-transparente si on le désire.

IPSec fournit trois types de services : l'authentification des deux extrémités, la confidentialité des données par chiffrement, et la gestion des clés. On l'utilise notamment pour protéger la connectivité de succursales au travers de l'Internet, établir la connectivité intranet et extranet avec des partenaires ou encore augmenter la sécurité du commerce électronique.

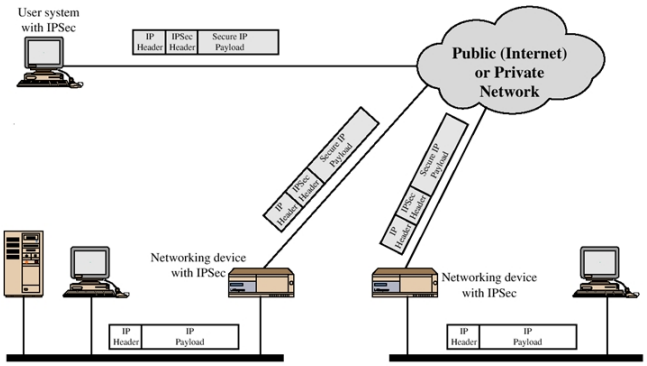

FIG. 11.1 – Emplacement du protocole IPSec

IPSec fournit 4 services de sécurité, permettant l'authentification et/ou la confidentialité.

- Authentification des données : IPSec permet de s'assurer que pour chaque paquet échangé, il a bien été émis par la bonne machine et qu'il est bien à destination de la seconde machine
- Confidentialité des données échangées : On peut décider de chiffrer le contenu des paquets IP pour empêcher qu'une personne extérieure ne le lise
- Intégrité des données échangées : IPSec permet de s'assurer qu'aucun paquet n'a subit une quelconque modification durant son trajet.

– Protection contre l'analyse de trafic : IPSec permet de chiffrer les adresses réelles de l'expéditeur et du destinataire, ainsi que tout l'en-tête IP correspondant. C'est le principe de base du tunneling.

11.2 Architecture

L'architecture d'IPSEC est définie en détails dans les RFC 2401, 2402, 2046 et 2408. Il est constitué de deux protocoles. Le premier, AH (Authentication Header), est responsable de l'authentification des parties de manière sûre, mais ne garantit aucune confidentialité. Le second, ESP (Encapsulating Security Payload), est responsable du chiffrement des données. Il peut également garantir l'authentification des parties, mais apporte alors une certaine redondance avec AH. Ces protocoles peuvent être utilisés séparément ou combinés.

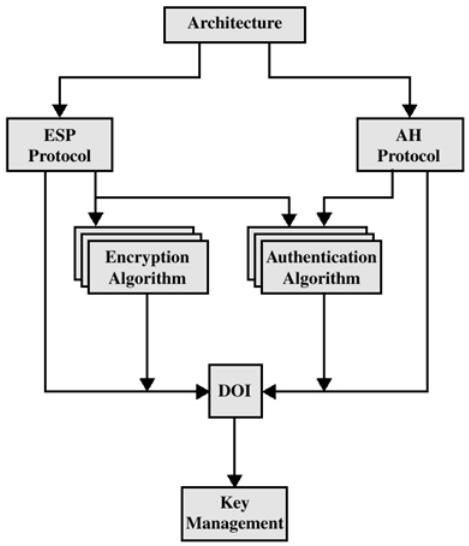

Fig. 11.2 – Architecture IPSec

11.3 Les modes d'IPSec

Trois modes sont possibles : Transport, Tunnel et Nesting.

11.3.1 Le mode Transport

Il se positionne entre le protocole Réseau (IP) et le protocole Transport (TCP, UDP).

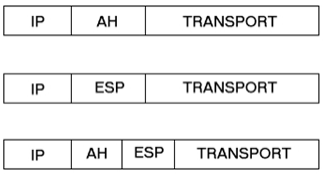

Fig. 11.3 – Mode Transport du protocole IPSec

Il peut être utilisé entre deux end-users. Il protège uniquement la zone de données du paquet. Ce mode n'est utilisé que ponctuellement.

11.3.2 Le mode Tunnel

Il remplace les adresses IP originales (IP_B) et encapsule tout le paquet IP. Pour ce faire, il place d'autres entêtes IP (IP_A) qui ne seront conservées que durant le transfert du paquet.

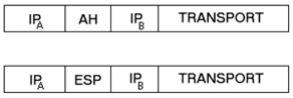

Fig. 11.4 – Mode Tunnel du protocole IPSec

Ce mode permet la transparence, et est habituellement utilisé entre des routeurs.

11.3.3 Le mode Nesting

C'est un mode hybride. On applique successivement les deux protocoles. Il s'agit dès lors d'une encapsulation IPSec dans IPSec

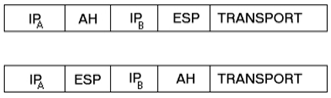

Fig. 11.5 – Mode Nesting du protocole IPSec

11.4 SPD et SA

La SPD (Security Policy Database) est une base de données présente sur chaque système capable d'utiliser IPSec. Elle permet de déterminer la politique de sécurité à appliquer à un certain trafic. Chaque entrée de cette base est identifiée grâce à plusieurs "sélecteurs" tels que l'adresse IP source et destination, le numéro de port ou le protocole de transport. Ce sont ces sélecteurs qui permettent de retrouver les SAs associées à un type de trafic. On parle de *granularité*.

La SPD est consultée pendant tout transfert de données, entrant ou sortant, IPSec ou Non-IPSec. Trois choix de traitement sont possibles :
- Appliquer les paramètres IPSec (c'est-à-dire protéger la connexion par une SA)
- Laisser passer le trafic
- Rejeter le trafic

Les SA (Security Association) sont le nom donné à toute communication protégée par l'intermédiaire d'IPSec. C'est une relation à sens unique entre un expéditeur et un récepteur. Elle repose sur une unique application de AH ou de ESP. Ainsi, pour une liaison protégée par AH entre 2 entités, il y aura 2 SA.

La SA est identifiée de manière unique par trois paramètres :
- Index de Paramètres de Sécurité (SPI - Security Parameters Index)
- Adresse de destination IP
- Identifiant du protocole de sécurité : AH ou ESP

Pour envoyer un paquet, il faudra donc :

1. Rechercher la politique de sécurité grâce aux sélecteurs
2. S'il faut protéger le trafic, on cherche la SA correspondante :
   - Si la SA existe, on applique le traitement associé (AH, ESP)
   - Sinon, on crée la SA, puis on applique le traitement

11.5 AH - Authentication Header

Ce protocole fournit le support pour l'intégrité des données et l'authentification des paquets IP. Pour l'authentification, l'extrémité du système/routeur peut authentifier l'utilisateur/application. Pour l'intégrité, on se basera l'utilisation d'un MAC, d'où la nécessité d'une clé secrète. L'algorithme HMAC est présent, et repose sur le MD5 ou SHA-1 (les deux doivent être supportés). AH permet aussi de vérifier l'unicité des paquets pour contrer les attaques de rejeu.

AH n'assure pas la confidentialité. En effet, les données sont signées, mais pas chiffrées.

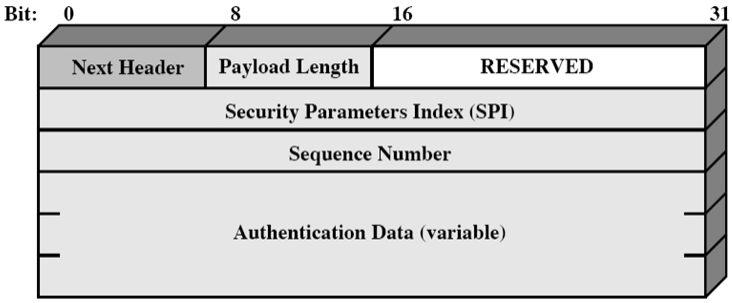

FIG. 11.6 – En-tête AH

La figure 11.7 illustre le mode Transport associé au protocole AH. On note la présence de la clé secrète (SK). On utilise l'algorithme HMAC pour obtenir le MAC à partir de la fonction de hachage MD5 ou SHA-1. La ligne noire continue représente ce qui est authentifié (ici, l'entièreté du paquet).

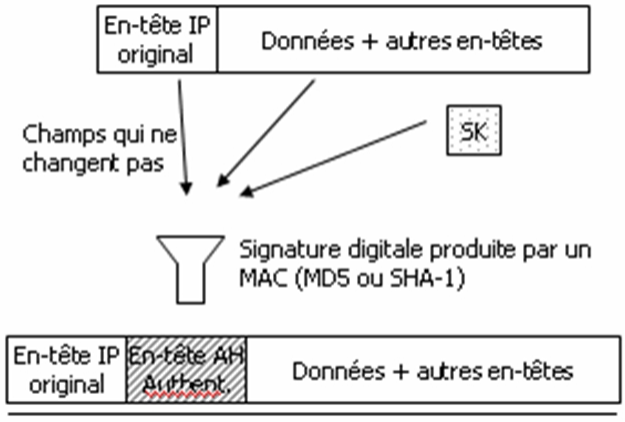

FIG. 11.7 – AH + Transport

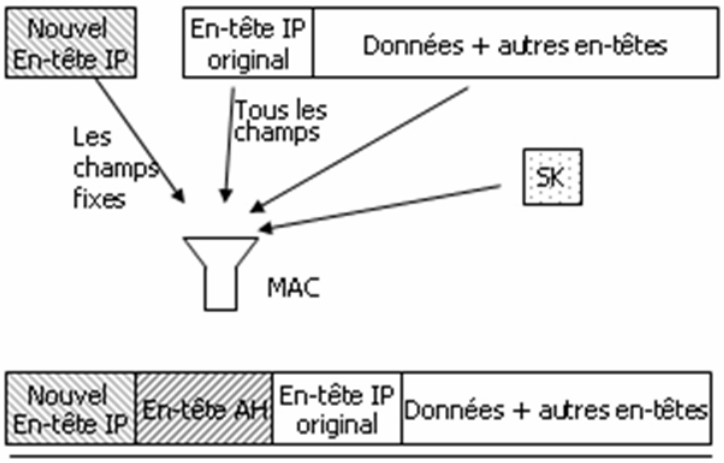

FIG. 11.8 – AH + Tunnel

11.6 ESP - Encapsulation Security Payload

Ce protocole fournit la confidentialité du contenu du message et une protection contre l'analyse de trafic. Il peut également fournir en option des services d'authentification semblables à ceux de AH. Il supporte les chiffrements usuels (DES, Triple-DES, RC5, IDEA, CAST,...), le mode CBC, et autorise le padding afin d'obtenir la taille de bloc nécessaire, le cas échéant.

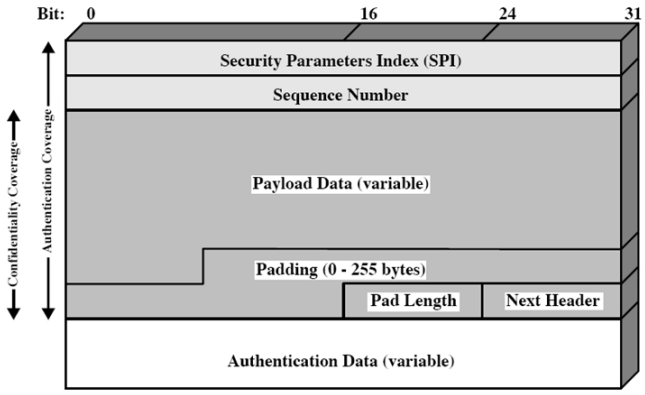

FIG. 11.9 – En-tête ESP

La figure 11.10 illustre le mode Transport avec le protocole ESP. La double flèche représente ce qui est chiffré, la droite continue, ce qui est authentifié. On remarque à ce propos que l'authentification est moindre que pour le protocole AH (où l'entête IP originale est également authentifiée). L'authentification est donc présente, mais moins forte. Dans la majorité des cas, elle sera cependant suffisante, et on préfèrera utiliser ESP plutôt que AH pour des raisons de confidentialité.

Le trailer ESP a deux raisons d'être :
– Il permet de camoufler la taille réelle du paquet,
– Il permet de remplir les conditions pour certains modes de chiffrement pour le DES par exemple.

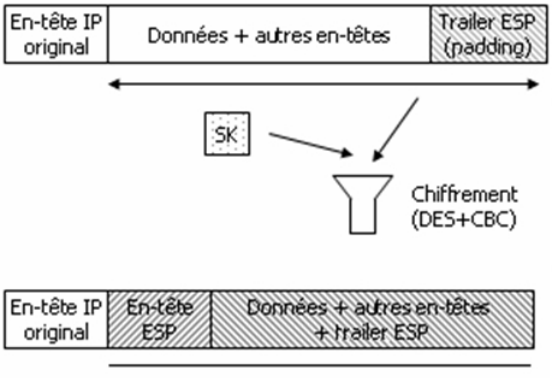

FIG. 11.10 – ESP + Transport

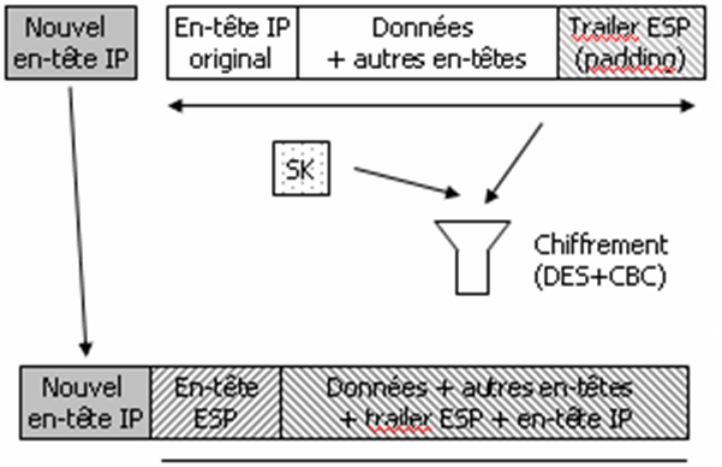

FIG. 11.11 – ESP + Tunnel

11.7 Gestion des clés

IPSEC gère la génération et la distribution des clés. Dans le cadre d'une gestion manuelle, l'administrateur système configure manuellement chaque système. Mais cela devient problématique lorsque le nombre de machines augmente car le nombre de couples de clefs évolue de manière quadratique avec le nombre de systèmes. Avec IPSEC, la gestion des clés est automatisée (IKE : Internet Key Exchange). Un système automatisé est présent pour la création sur demande des clefs pour la création des SAs dans de grands systèmes. La partie responsable de la gestion des clés dans IPSEC repose sur les protocoles OAKLEY et ISAKMP.

11.7.1 OAKLEY

C'est un protocole d'échange de clés basé sur la méthode de Diffie-Hellman. Il ajoute des dispositifs pour contrer certaines faiblesses. Ces dispositifs peuvent prendre la forme de cookies, de groupes (paramètres globaux), de nonces, ou encore d'un échange de clé DH avec authentification. Il peut pour ce faire employer des nombres premiers ou des courbes elliptiques.

I → R: CKY_I, OK_KEYX, GRP, g^x, EHAO, NIDP, ID_I, ID_R, N_I, $S_{KI}[ID_I \| ID_R \| N_I \| GRP \| g^x \| EHAO]$

R → I: CKY_R, CKY_I, OK_KEYX, GRP, g^y, EHAS, NIDP, ID_R, ID_I, N_R, N_I, $S_{KR}[ID_R \| ID_I \| N_R \| N_I \| GRP \| g^y \| g^x \| EHAS]$

I → R: CKY_I, CKY_R, OK_KEYX, GRP, g^x, EHAS, NIDP, ID_I, ID_R, N_I, N_R, $S_{KI}[ID_I \| ID_R \| N_I \| N_R \| GRP \| g^x \| g^y \| EHAS]$

Notation:

I	=	Initiator
R	=	Responder
CKY_I, CKY_R	=	Initiator, responder cookies
OK_KEYX	=	Key exchange message type
GRP	=	Name of Diffie-Hellman group for this exchange
g^x, g^y	=	Public key of initiator, responder; g^{xy} = session key from this exchange
EHAO, EHAS	=	Encryption, hash, authentication functions, offered and selected
NIDP	=	Indicates encryption is not used for remainder of this message
ID_I, ID_R	=	Identifier for initiator, responder
N_I, N_R	=	Random nonce supplied by initiator, responder for this exchange
$S_{KI}[X]$, $S_{KR}[X]$	=	Indicates the signature over X using the private key (signing key) of initiator, responder

FIG. 11.12 – Echange par OAKLEY

11.7.2 ISAKMP

Cet acronyme représente l'Internet Security Association and Key Management Protocol. Il fournit le cadre pour la gestion des clés et définit des procédures et le format de paquet pour établir, négocier, modifier, et supprimer des SAs. Il est indépendant du protocole d'échange de clé, de l'algorithme de chiffrement, et de la méthode d'authentification.

(a) Base Exchange	
(1) **I → R:** SA; NONCE	Begin ISAKMP-SA negotiation
(2) **R → I:** SA; NONCE	Basic SA agreed upon
(3) **I → R:** KE; ID_I; AUTH	Key generated; Initiator identity verified by responder
(4) **R → I:** KE; ID_R; AUTH	Responder identity verified by initiator; Key generated; SA established
(b) Identity Protection Exchange	
(1) **I → R:** SA	Begin ISAKMP-SA negotiation
(2) **R → I:** SA	Basic SA agreed upon
(3) **I → R:** KE; NONCE	Key generated
(4) **R → I:** KE; NONCE	Key generated
(5)* **I → R:** ID_I; AUTH	Initiator identity verified by responder
(6)* **R → I:** ID_R; AUTH	Responder identity verified by initiator; SA established

Notation:

I	=	initiator
R	=	responder
*	=	signifies payload encryption after the ISAKMP header

FIG. 11.13 – Les types d'échange par ISAKMP

(c) Authentication Only Exchange	
(1) **I → R:** SA; NONCE	Begin ISAKMP-SA negotiation
(2) **R → I:** SA; NONCE; ID_R; AUTH	Basic SA agreed upon; Responder identity verified by initiator
(3) **I → R:** ID_I; AUTH	Initiator identity verified by responder; SA established
(d) Aggressive Exchange	
(1) **I → R:** SA; KE; NONCE; ID_I	Begin ISAKMP-SA negotiation and key exchange
(2) **R → I:** SA; KE; NONCE; ID_R; AUTH	Initiator identity verified by responder; Key generated; Basic SA agreed upon
(3)* **I → R:** AUTH	Responder identity verified by initiator; SA established
(e) Informational Exchange	
(1)* **I → R:** N/D	Error or status notification, or deletion

Notation:
I = initiator
R = responder
* = signifies payload encryption after the ISAKMP header

FIG. 11.14 – Les types d'échange par ISAKMP (suite)

11.8 Ressources supplémentaires

`http://www.hsc.fr/ressources/articles/`
`http://web.mit.edu/tytso/www/ipsec/index.html`

Chapitre 12

Protocoles d'authentification

12.1 Authentification mutuelle

On l'utilise pour convaincre les parties de l'identité de l'une par rapport à l'autre et pour échanger des clés de session. Il faut veiller à la confidentialité (protection des clés de session) et au paramètre de temps (pour empêcher le rejeu).

12.1.1 Approche symétrique

Comme discuté précédemment, on peut employer une hiérarchie à deux niveaux des clefs :

- Transfert confidentiel de la clé
- Transfert confidentiel des données

Habituellement, on trouvera un centre de distribution de clés (KDC). Chaque partie partage une clé principale avec lui. Il produit des clefs de session utilisées pour les communications entre les parties. Les clés principales sont utilisées pour distribuer ces clés de session aux parties.

12.1.1.1 Needham-Schroeder :

Il s'agit d'un protocole de distribution de clé avec un tiers médiateur. Pour une session entre A et B, négociée par le KDC, il vient la séquence suivante :

1. $A \rightarrow KDC : ID_A||ID_B||N_1$
2. $KDC \rightarrow A : E_{K_a}[K_s||ID_B||N_1||E_{K_b}[K_s||ID_A]]$
3. $A \rightarrow B : E_{K_b}[K_s||ID_A]$
4. $B \rightarrow A : E_{K_s}[N_2]$
5. $A \rightarrow B : E_{K_s}[f(N_2)]$

On l'utilise pour distribuer sûrement une nouvelle clé de session entre A et B. Il est cependant vulnérable à une attaque par rejeu[1] si une vieille clé de session a été compromise.

Plusieurs modifications peuvent être apportées pour contrer ce risque : ce sont les méthodes de Denning et de Neuman.

[1] Rejeu du message 3 qui convainc B que A communique avec lui.

Utilisation d'un timestamp (Denning 81) :

1. $A \rightarrow KDC : ID_A||ID_B$
2. $KDC \rightarrow A : E_{K_a}[K_s||ID_B||T||E_{K_b}[K_s||ID_A||T]]$
3. $A \rightarrow B : E_{K_b}[K_s||ID_A||T]$
4. $B \rightarrow A : E_{K_s}[N_1]$
5. $A \rightarrow B : E_{K_s}[f(N_1)]$

Employer un nonce supplémentaire (Neuman 93) :

1. $A \rightarrow B : ID_A||N_a$
2. $B \rightarrow KDC : ID_B||N_b||E_{K_b}[ID_A||N_a||T_b]$
3. $KDC \rightarrow A : E_{K_a}[ID_B||N_a||K_s||T_b]||E_{K_b}[ID_A||K_s||T_b]||N_b$
4. $A \rightarrow B : E_{K_b}[ID_A||K_s||T_b]||E_{K_s}[N_b]$

12.1.2 Approche par clé publique

On dispose de différentes alternatives basées sur l'utilisation du chiffrement à clé publique. Mais il est nécessaire de s'assurer que les clefs publiques dont on dispose correspondent aux bonnes personnes. On emploie alors un serveur central d'authentification[2]. Divers protocoles existent utilisant des horodateurs ou des nonces.

12.1.2.1 Protocole de Denning utilisant un AS :

1. $A \rightarrow AS : ID_A||ID_B$
2. $AS \rightarrow A : E_{KR_{as}}[ID_A||KU_a||T]||E_{KR_{as}}[ID_B||KU_b||T]$
3. $A \rightarrow B : E_{KR_{as}}[ID_A||KU_a||T]||E_{KR_{as}}[ID_B||KU_b||T]||E_{KU_b}[E_{KR_a}[K_s||T]]$

La clef de session est choisie par A. Par conséquent, il n'est pas nécessaire de faire confiance à l'AS pour la protéger. Les horodateurs empêchent le rejeu mais exigent des horloges synchronisées.

Solution pour la désynchronisation [Woo & Lam 92] : (on fonctionne avec des certificats)

1. $A \rightarrow KDC : ID_A||ID_B$
2. $KDC \rightarrow A : E_{KR_{auth}}[ID_B||KU_b]$
3. $A \rightarrow B : E_{KU_b}[N_a||ID_A]$
4. $B \rightarrow KDC : ID_B||ID_A||E_{KU_{auth}}[N_a]$
5. $KDC \rightarrow B : E_{KR_{auth}}[ID_A||KU_a]||E_{KU_b}[E_{KR_{auth}}[N_a||K_s||ID_B]]$
6. $B \rightarrow A : E_{KU_a}[E_{KR_{auth}}[N_a||K_s||ID_B]||N_b]$
7. $A \rightarrow B : E_{K_s}[N_b]$

Renforcement [Woo & Lam 92]

1. $A \rightarrow KDC : ID_A||ID_B$
2. $KDC \rightarrow A : E_{KR_{auth}}[ID_B||KU_b]$
3. $A \rightarrow B : E_{KU_b}[N_a||ID_A]$
4. $B \rightarrow KDC : ID_B||ID_A||E_{KU_{auth}}[N_a]$
5. $KDC \rightarrow B : E_{KR_{auth}}[ID_A||KU_a]||E_{KU_b}[E_{KR_{auth}}[N_a||K_s||ID_A||ID_B]]$
6. $B \rightarrow A : E_{KU_a}[E_{KR_{auth}}[N_a||K_s||ID_A||ID_B]||N_b]$
7. $A \rightarrow B : E_{K_s}[N_b]$

[2] AS - Authentication Server.

12.2 Authentification par passage unique

Cette technique est requise lorsque l'expéditeur et le récepteur ne sont pas en communication en même temps (dans le cas d'un mail par exemple). L'en-tête est en clair, ainsi il peut être compris par le serveur mail. On souhaite que le contenu du corps du texte soit protégé et que l'expéditeur soit authentifié.

12.2.1 Approche symétrique

C'est un raffinement de l'utilisation d'un KDC :

1. $A \rightarrow KDC : ID_A||ID_B||N_1$
2. $KDC \rightarrow A : E_{K_a}[K_s||ID_B||N_1||E_{K_b}[K_s||ID_A]]$
3. $A \rightarrow B : E_{K_b}[K_s||ID_A]||E_{K_s}[M]$

Cela garantit un certain niveau d'authentification, mais ne protège pas du rejeu. On pourrait utiliser des timestamps mais les délais des mails rendent ce système problématique.

12.2.2 Approche à clé publique

Plusieurs alternatives ont déjà été présentées :

– Si la *confidentialité* est le souci principal :

$$A \rightarrow B : E_{KU_b}[K_s]||E_{K_s}[M]$$

La clé de session est chiffrée avec la clé publique de B, le message est chiffré avec cette clé de session.

– Si l'*authentification* est importante :
on utilise une signature numérique avec un certificat numérique.

$$A \rightarrow B : M||E_{KR_a}[H(M)]||E_{KR_{as}}[T||ID_A||KU_a]$$

On remarque que message, signature, et certificat sont fournis.

12.3 Kerberos

Kerberos est un système de serveurs de distribution de clés proposé par le MIT. Il fournit une authentification centralisée dans un réseau distribué et permet l'accès pour les utilisateurs aux services distribués proposés par le réseau sans devoir faire confiance à tous les postes de travail mais plutôt en faisant confiance à un serveur central d'authentification. Deux versions sont en service : v4 et v5.

12.3.1 Conditions à remplir

Le cahier des charges mentionnait les services logiques d'un serveur d'authentification, à savoir :
– la sécurité,
– la fiabilité,
– la transparence,
– la modularité.

Il utilise en outre un protocole d'authentification basé sur Needham-Schroeder.

12.3.2 Kerberos V4 - Vue générale

C'est un schéma d'authentification basique utilisant un tiers de confiance. L'architecture Kerberos dispose :

- d'un serveur d'authentification (AS). Les utilisateurs négocient avec l'AS pour s'identifier. Celui-ci fournit un ticket d'authentification non corruptible (Ticket Granting Ticket - TGT).
- d'un serveur de ticket (Ticket Granting Server - TGS). Les utilisateurs demandent un accès à des services proposés par le TGS sur base de leur TGT.

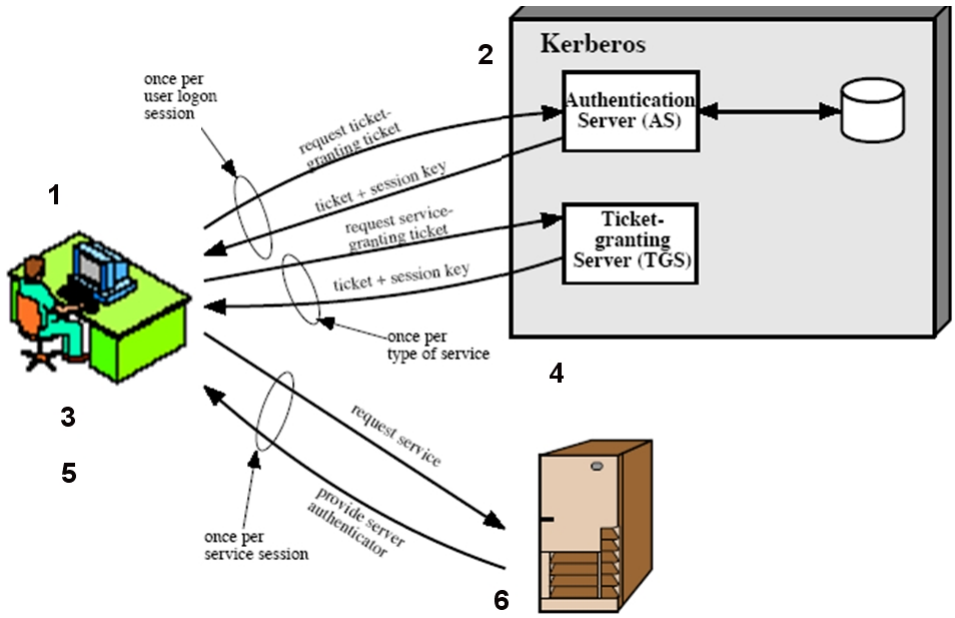

FIG. 12.1 – Système Kerberos

1. L'utilisateur se loggue sur un poste de travail (le mot de passe est utilisé pour établir la clé secrète K_C par l'intermédiaire d'une fonction de hachage) et demande un service à l'AS.
2. L'AS vérifie les droits d'accès de l'utilisateur dans la base de données, crée un ticket (TGT) et une clé de session. Ces résultats sont chiffrés en utilisant la clé K_C.
3. Le client déchiffre le message venant du serveur, puis il envoie le TGT et un authentificateur qui contient le nom de l'utilisateur, l'adresse réseau du poste de travail et un horodateur au TGS.
4. Le TGS déchiffre le ticket et l'authentificateur, vérifie la demande, puis crée le ticket pour le serveur (service) demandé.
5. Le poste de travail envoie ce ticket et l'authentificateur au serveur concerné.
6. Le serveur vérifie que ce ticket et l'authentificateur correspondent et accorde alors l'accès au service. Si l'authentification mutuelle est exigée, le serveur renvoie un authentificateur.

Dans la figure 12.2, K_C représente la clé calculée à partir du mot de passe de C, et $K_{c,tgs}$ est chiffrée par K_C. Seul C peut donc la lire.

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) C → AS: $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C: $E_{K_c}\left[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) C → TGS: $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C: $E_{K_{c,tgs}}\left[K_{c,v} \| ID_v \| TS_4 \| Ticket_v\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$$

$$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$$

$$Authenticator_c = E_{K_{c,tgs}}\left[ID_C \| AD_C \| TS_3\right]$$

(c) Client/Server Authentication Exchange: to obtain service

(5) C → V: $Ticket_v \| Authenticator_c$

(6) V → C: $E_{K_{c,v}}[TS_5 + 1]$ (for mutual authentication)

$$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$$

$$Authenticator_c = E_{K_{c,v}}\left[ID_C \| AD_C \| TS_5\right]$$

FIG. 12.2 – Synthèse des envois de messages

12.3.3 Domaines Kerberos

Un environnement Kerberos se compose de 3 éléments :

- un serveur Kerberos,
- un certain nombre de clients, tous inscrits au serveur,
- et un ou plusieurs serveurs d'application, partageant des clés avec le serveur.

Un tel environnement est appelé domaine Kerberos (realm). En cas de domaines Kerberos multiples, les différents serveurs Kerberos doivent partager des clés et se faire confiance, afin d'interagir comme sur la figure 12.3.

La séquence d'échange de tickets est la suivante :

1. $C \rightarrow AS : ID_c||ID_{tgs}||TS_1$
2. $AS \rightarrow C : E_{K_c}[K_{c,tgs}||ID_{tgs}||TS_2||Lifetime_2||Ticket_{tgs}]$
3. $C \rightarrow TGS : ID_{tgsrem}||Ticket_{tgs}||Authenticator_c$
4. $TGS \rightarrow C : E_{K_{c,tgs}}[K_{c,tgsrem}||ID_{tgsrem}||TS_4||Ticket_{tgsrem}]$
5. $C \rightarrow TGSrem : ID_{vrem}||Ticket_{tgsrem}||Authenticator_c$
6. $TGSrem \rightarrow C : E_{K_c,tgsrem}[K_{c,vrem}||ID_{vrem}||TS_6||Ticket_{vrem}]$
7. $C \rightarrow Vrem : Ticket_{vrem}||Authenticator_c$

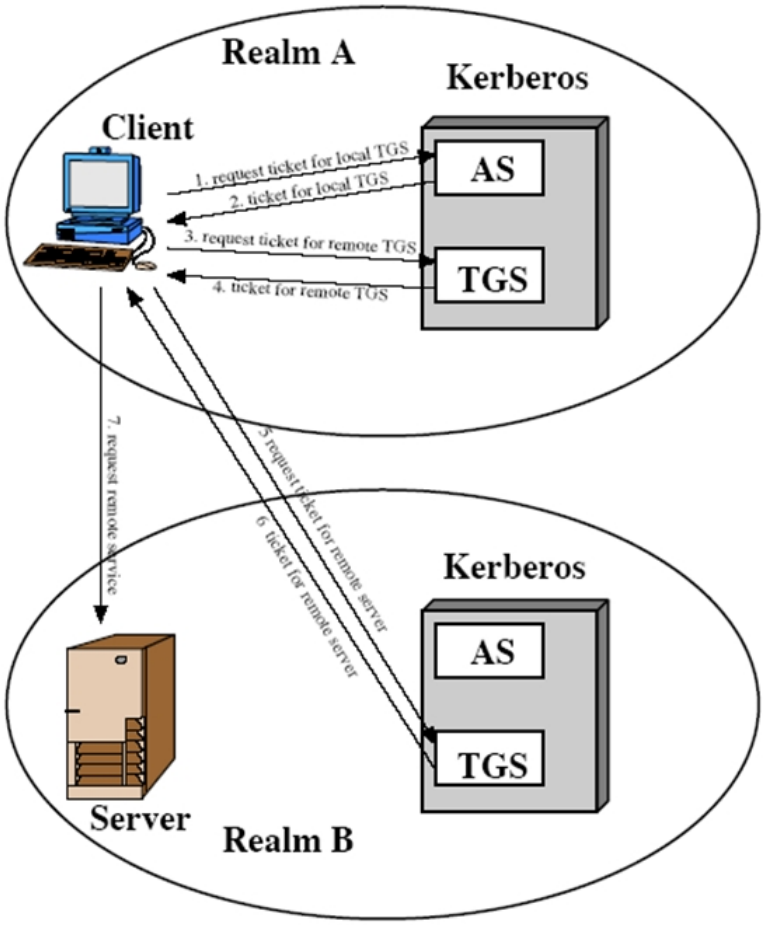

FIG. 12.3 – Domaines Kerberos

12.3.4 Kerberos V5

Développé dans le milieu des années 90, il fournit des améliorations de v4. Il corrige et met à jour quelques points clés du système tels que les algorithmes de chiffrement, l'utilisation de certains protocoles réseau, le temps de vie du ticket, les transferts d'authentification ou l'authentification inter-domaines. L'ensemble des caractéristiques est repris dans la RFC 1510.

Dans la figure 12.4, *Realm* représente le domaine de l'utilisateur, *Options* est utilisé pour demander que certains 'flags' soient levés dans le ticket obtenu en retour, *Times* permet de préciser les balises de durée de vie du ticket (from, till, rtime) et *Nonce* est une valeur aléatoire assurant de la fraîcheur de l'information obtenue.

La Subkey apparaissant dans la $6^{ème}$ phase est une proposition du client pour l'utilisation d'une clé de chiffrement à utiliser pour protéger les informations qui transiteront lors de cette session. Si le paramètre est omis, c'est $K_{c,v}$ qui sera utilisée. Dans cette même phase, le *Sequence number* précise le numéro de séquence à partir duquel seront numérotés les messages échangés pendant la session (cela permet de détecter les attaques par rejeu).

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) C → AS: $Options \| ID_c \| Realm_c \| ID_{tgs} \| Times \| Nonce_1$

(2) AS → C: $Realm_c \| ID_C \| Ticket_{tgs} \| E_{K_c}\left[K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times\right]$$

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) C → TGS: $Options \| ID_v \| Times \| \| Nonce_2 \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C: $Realm_c \| ID_C \| Ticket_v \| E_{K_{c,tgs}}\left[K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_V\right]$

$$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times\right]$$

$$Ticket_v = E_{K_v}\left[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times\right]$$

$$Authenticator_c = E_{K_{c,tgs}}\left[ID_C \| Realm_c \| TS_1\right]$$

(c) Client/Server Authentication Exchange: to obtain service

(5) C → V: $Options \| Ticket_v \| Authenticator_c$

(6) V → C: $E_{K_{C,V}}$ [$TS_2 \| Subkey \| Seq\#$]

$$Ticket_v = E_{K_v}\left[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times\right]$$

$$Authenticator_c = E_{K_{c,V}}\left[ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#\right]$$

FIG. 12.4 – Synthèse des messages de Kerberos v5

12.4 Secure Socket Layer

C'est un protocole de la couche Session (modèle OSI) qui permet de créer une connexion sécurisée à partir d'un transport TCP. Il fut à l'origine créé par Netscape, en 1992. Il offre des fonctions d'authentification forte du serveur et/ou du client. De plus, il vérifie l'intégrité des données ainsi que la confidentialité des données.

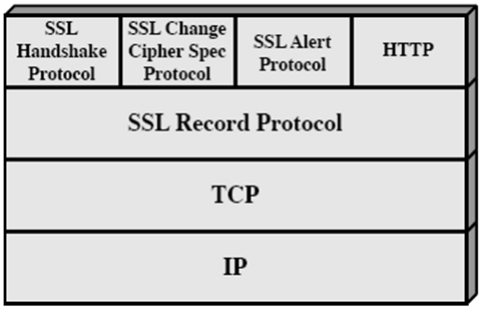

FIG. 12.5 – Emplacement du protocole SSL

SSL est composé de 2 sous-couches : la première définit la façon dont seront traitées les données à chiffrer et la seconde, gère l'établissement de la connexion sécurisée et la négociation des paramètres de la session.

12.4.1 SSL Record Protocol

Elle fournit 2 services :

- la confidentialité : via un chiffrement symétrique avec une clef secrète partagée définie dans le «Handshake Protocol». Les algorithmes pouvant être utilisés sont notamment IDEA, DES, 3DES et RC4.
- l'intégrité des messages par l'utilisation d'un MAC avec la clef secrète partagée.

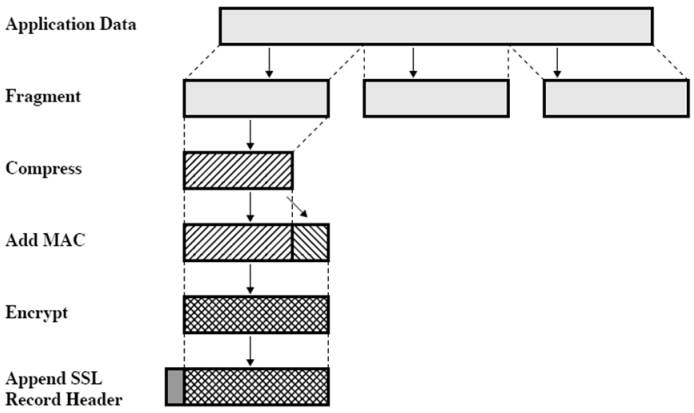

FIG. 12.6 – SSL Record Protocol

Le Record protocol est illustré à la figure 12.6. Chaque phase est définie ci-dessous :

1. Fragmentation : chaque message est fragmenté en blocs de 2^{14} bytes ou moins.
2. Compression (optionnelle) : doit être sans perte et ne doit pas augmenter la taille du contenu de plus de 1024 bytes. En effet, dans certains cas, si le message est trop court, il arrive que la compression gonfle les données. Dans SSL3, l'algorithme de compression à utiliser n'est pas spécifié.
3. MAC : on utilise une clé secrète connue des 2 parties.

   Le calcul du MAC est réalisé par la fonction suivante :

   $$Hash(MAC\_write\_secret||pad\_2||hash(MAC\_write\_secret||pad\_1||seq\_num||SSLCompressed.type||SSLCompressed.length||SSLCompressed.fragment))$$

   où

   - MAC_write_secret : la clé secrète partagée
   - Hash : algorithme MD5 ou SHA-1
   - Pad_1 : byte 0x36 (0011 0110) répété 48 fois (384 bits) pour le MD5 et 40 fois pour le SHA-1
   - Pad_2 : byte 0x5C (0101 1100) répété 48 fois pour le MD5 et 40 fois pour le SHA-1
   - Seq_num : numéro de séquence pour ce message
   - SSLCompressed.type : le protocole de haut niveau utilisé pour manipuler ce fragment
   - SSLCompressed.length : longueur du fragment compressé
   - SSLCompressed.fragment : le fragment compressé

4. Chiffrement symétrique : ne doit pas augmenter la longueur du contenu de plus de 1024 bytes et donc la longueur totale ne doit pas excéder $2^{14} + 2048$ bytes (chiffrement + compression).

5. Le header contient différents paramètres dont la manière de manipuler le fragment en question (Content Type, qui indique le protocole de niveau supérieur à utiliser pour traiter les données du paquet), la version SSL (Major et Minor Version) et la longueur du fragment compressé (ou du fragment de texte clair si aucune compression n'a lieu).

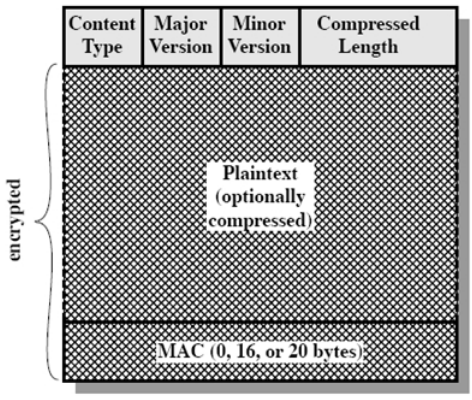

FIG. 12.7 – SSL Record Protocol - Header

12.4.2 SSL Change Cipher Spec Protocol

Il est lancé au cours de la phase de handshake. Il est constitué d'un seul byte. Son but est de synchroniser les stratégies de chiffrement utilisées par le client et le serveur.

Au cours du handshake, les paramètres de chiffrement sont négociés et calculés. Ces nouveaux paramètres constituent une stratégie 'temporaire'. A l'envoi du message 'Change_cipher_specs', les nouveaux paramètres sont copiés dans la stratégie 'courante' pour être directement utilisés par la couche SSL Record.

12.4.3 SSL Alert Protocol

Comme son nom l'indique, ce protocole transmet des alertes qui peuvent être un simple avertissement ou être fatale, et dans ce cas provoquer la fermeture de la connexion SSL.

Elles sont représentées sur 2 octets, le premier donne le niveau d'alerte (*Warning* ou *Fatal*), le second donne le code de l'alerte.

Ces alertes peuvent survenir notamment lors d'un message inattendu, d'un mauvais MAC, d'un échec de décompression, d'un échec durant la phase de handshake, ou encore d'un paramètre illégal. Mais une alerte peut aussi être émise lors de la fin de l'envoi de message, lorsqu'aucun certificat n'est utilisé, en cas de mauvais certificat ou de certificat non supporté, si le certificat est révoqué ou expiré, ou encore en cas de certificat inconnu.

12.4.4 SSL Handshake protocol

Ce protocole permet au serveur et au client de s'authentifier, de négocier les algorithmes de chiffrement et de MAC, et de négocier les clés cryptographiques à employer. Il comporte quatre phases d'échanges de messages :

1. *(HELLO)* L'établissement des possibilités en matière de sécurité. Cette phase détermine entre autres la version, la session, les algorithmes de chiffrement et compression ainsi que des nombres aléatoires.
2. L'authentification du serveur et échange de clé.
3. L'authentification du client et échange de clé.
4. *(FINISH)* La clôture.

Phase 1 : HELLO Au départ, les paramètres de chiffrement sont initialisés à 0 (c-à-d qu'aucun algorithme de chiffrement, de hachage et de signature ne sont définis).

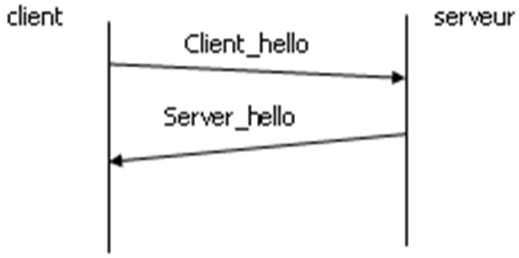

FIG. 12.8 – SSL Phase 1 - Hello

Client_hello est composé de :
- N° de version SSL (2 ou 3)
- N° aléatoire du client (ClientHello.random)
- N° de session (optionnel) (permet de récupérer les paramètres de sécurité d'une connexion antérieure)
- Liste des stratégies de chiffrements supportées, par ordre décroissant de préférences (DES, 3DES, IDEA, ...)
- Algorithmes de hachage supportés. (MD5, SHA-1, ...)

Server_hello comporte :
- N° de version SSL (en fonction de celui supporté par le client)
- N° aléatoire du serveur (ServerHello.random)
- N° de session
- Stratégie de chiffrement choisie (en fonction de celles du client)
- Algorithme de hachage choisi (en fonction de ceux du client)

Phase 2 : Authentification du serveur et échange de clé Plusieurs techniques existent pour échanger les clés. Elles sont basées sur une version de l'algorithme Diffie-Hellman et/ou du RSA :

- **DH anonyme** : il s'agit de l'algorithme de base. Il est sujet à l'attaque man-in-the-middle car les paramètres DH ne sont pas authentifiés.
- **DH fixé** : les paramètres de DH sont établis et publiés au moyen d'un certificat. Il contre l'attaque man-in-the-middle mais pas une attaque par force brute.
- **DH éphémère** : les paramètres DH peuvent changer d'une session à l'autre ce qui rend une attaque par force brute bien plus difficile à réaliser. Les paramètres DH sont de plus signés grâce à la clé RSA privée de l'expéditeur pour contrer l'attaque man-in-the-middle.
- **RSA** : la clé secrète est chiffrée grâce à la clé publique du receveur. Cela implique qu'un certificat du receveur soit disponible afin d'obtenir sa clé publique de façon fiable.

FIG. 12.9 – SSL Phase 2 - authentification du serveur

La phase 2 est illustrée à la figure 12.9.
'Certificate' envoie le certificat au client. Il n'est pas envoyé s'il s'agit d'un DH anonyme. Il contient aussi les paramètres de la clé publique DH s'il s'agit d'un DH fixé.

'Server_key_exchange' envoie les paramètres relatifs à la clé de session. Il n'est pas envoyé si on utilise un DH fixé (car ils sont déjà fournis dans le certificat) ou RSA sont utilisés.

'Certificate_request' demande le certificat du client. Il n'est pas envoyé si un DH anonyme est utilisé.

'Server_hello_done' marque la fin de la phase de Hello pour le serveur.

Phase 3 : Authentification du client et échange de clé Le client vérifie le certificat du serveur s'il l'a reçu, ensuite il enverra son certificat au serveur si celui-ci l'a demandé (message *Certificate* de la figure 12.10.

FIG. 12.10 – SSL Phase 3 - authentification du client

Client_key_exchange est similaire à Server_key_exchange. Il est toujours envoyé, même si son contenu est nul. Son contenu dépend du mécanisme d'échange de clé utilisé :

- S'il s'agit du RSA, le client génèrera une clé maîtresse primaire (PMK) représentée par un nombre aléatoire de 48 octets, et l'envoie au serveur.
- En cas de DH éphémère ou anonyme, les paramètres de la clé publique du client sont envoyés.
- En cas de DH fixé, le contenu de ce message est nul, puisque ces paramètres ont été envoyés dans le certificat.

Client_certificate_verify est envoyé après la génération de la valeur secrète maitresse (*Master_secret*). Il n'est pas envoyé si un DH fixé est utilisé. Pour cet envoi, le client signe le haché résultant de la concaténation de le MK, des messages qu'il aura envoyé précédemment, et de valeurs de padding.

Génération des clés : Selon qu'on utilise RSA ou DH, la gestion de la PMK sera différente :

- RSA : le client génère une valeur de 48 octets, la chiffre à l'aide de la clé publique du serveur et l'envoie à ce dernier. Le serveur utilise alors sa clé privée pour obtenir la PMK.
- DH : Les deux parties (client et serveur) génère une clé publique DH. une fois ces clés échangées, chaque partie calcule la PMK correspondante.

A ce moment, peu importe le mécanisme d'échange de clé utilisé, les deux parties disposent de la même PMK. A cet instant, serveur et client détermine la valeur secrète maitresse. Soit X, le ClientHello.random et Y, le ServerHello.random. La valeur secrète maitresse est donnée par

$$\begin{aligned} M\_Secret = & MD5(PMK||SHA('A'||PMK||X||Y))|| \\ & MD5(PMK||SHA('BB'||PMK||X||Y))|| \\ & MD5(PMK||SHA('CCC'||PMK||X||Y)) \end{aligned}$$

Phase 4 : FINISH Il s'agit des premiers messages utilisant la stratégie de chiffrement négociée au cours du « handshake ».

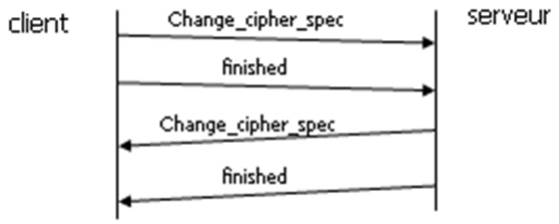

FIG. 12.11 – SSL Phase 4 - finish

'Change_cipher_spec' fait partie du protocole Cipher Spec. Il a pour but de transformer la stratégie temporaire en stratégie courante. Le message 'Finished' vérifie que l'échange de clé et les processus d'authentification se sont correctement déroulés.

12.4.5 TLS - Transport Layer Security

Aujourd'hui, on parle fréquemment du protocole TLS. Il s'agit en réalité de l'évolution du protocole SSL. Ses spécifications sont disponibles dans le RFC 2246. Les principales différences sont listées ci-dessous :

- utilisations HMAC pour le calcul de MAC
- une fonction pseudo-aléatoire intervient dans le calcul de clé
- des codes alertes additionnels
- modifications des chiffrements supportés
- le calcul de la valeur secrète maitresse est modifié

12.4.6 Utilisation du SSL - HTTPS

Le HTTPS[3] est une application du SSL pour Internet. De nombreux sites internet bancaires utilisent cette combinaison.

[3]HTTP over SSL, à ne pas confondre avec SHTTP (Secure HTTP) qui authentifie individuellement chaque page html avec un certificat.

Chapitre 13

Les architectures de paiement électronique

Le pourcentage d'internautes utilisant une carte de crédit pour effectuer leur paiement en ligne ne cesse de progresser. De nombreuses architectures de paiement électronique ont vu le jour. La plupart n'ont pas survécu. Après le SET, c'est aujourd'hui 3D-Secure qui est le plus couramment utilisé.

13.1 Le SET

Développé en 1996 par un ensemble de sociétés émettrices de cartes de crédit (MasterdCard, Visa,...), le SET (Secure Electronic Transaction) est utilisé pour protéger des transactions liées aux cartes de crédit sur Internet. Il ne s'agit pas d'un système de paiement, mais d'un ensemble de protocoles et de formats de sécurité basés sur trois principes :

- Des communications sécurisées entre les parties,
- L'utilisation des certificats X.509v3,
- Une certaine "intimité" en restreignant l'information à ceux qui en ont réellement besoin.

Les spécifications du SET sont reprises dans 3 livres.

- Un premier livre décrivant les données pour le Marché
- Un deuxième reprend les informations nécessaires aux programmeurs (API)
- Un troisième définit formellement le protocole.

Vue Générale

Le SET doit obéir à certaines contraintes, fixées par un cahier des charges :

- Fournir la confidentialité des paiements et des informations liées,
- Assurer l'intégrité des données transférées,
- Fournir l'authentification permettant de vérifier la légitimité de l'utilisateur d'une carte,
- Fournir l'authentification du marchand,
- Offrir une protection optimale de toutes les parties de la transaction,
- Etre indépendant des protocoles, plate-formes, OS, ...

Entités intervenantes

La figure 13.1 schématise les transferts de données entre entités. Les différents termes sont explicités ci-dessous :

- Cardholder : consommateur : personne autorisée à disposer d'une carte de paiement (MC, Visa, ...) fournie par un fournisseur (issuer)

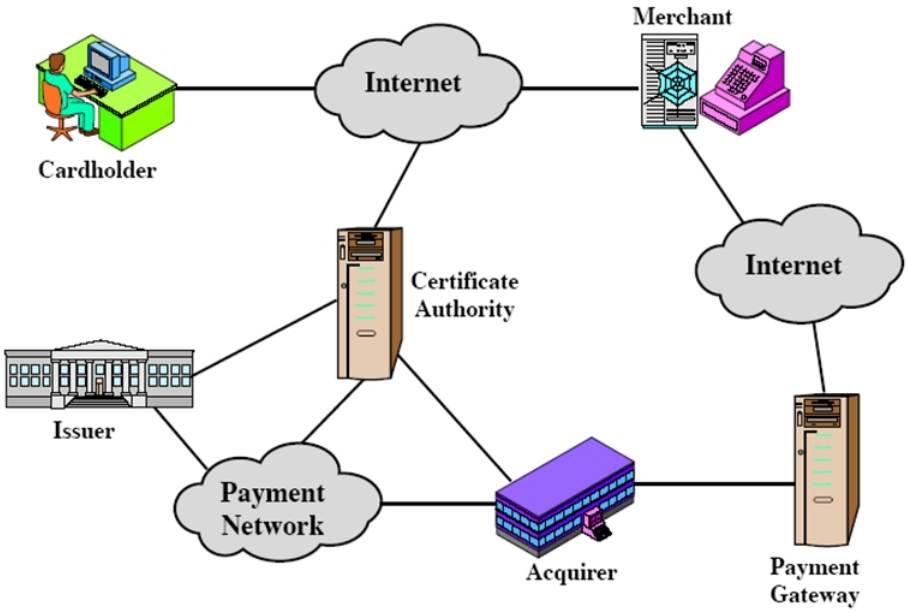

FIG. 13.1 – Composants du protocole SET

- Merchant : marchand : personne qui vend des biens ou des services au consommateur via un site Web ou un mail électronique. Un marchand qui accepte les cartes de paiement doit être en relation avec un acquéreur (acquirer)
- Issuer : fournisseur (banque de l'acheteur) : institution financière qui a fourni la carte au consommateur. Typiquement les comptes sont ouverts par mail ou par la personne directement. C'est le fournisseur qui est responsable du paiement de la dette du consommateur
- Acquirer : acquéreur (banque du marchand) : institution financière qui établit un compte avec un marchand, fournit les autorisations de paiement et gère les paiements. Le marchand accepte généralement plusieurs marques de carte de crédits mais préfère ne pas fonctionner avec de multiples associations de banque ou de consommateurs individuels. L'acquéreur fournit les autorisations au marchand lorsqu'une certaine carte est active et que la demande ne dépasse par la limite de crédit qui lui est autorisée.
- Payment gateway (passerelle de paiement) : il s'agit d'une fonction opérée par l'acquéreur ou un tiers désigné qui transmet les messages de paiement du marchand. Il s'agit d'une interface entre SET et le réseau existant de paiement par carte. Le marchand échange des messages SET avec ce gateway à travers internet, tandis que le gateway a une connexion directe vers l'acquéreur financier manipulant le système
- Certification authority : autorité qui fournit les certificats pour les propriétaires de carte, les marchands et les gateways de paiement

13.1.1 Quelques règles de sécurité

Pour assurer plus de confidentialité, et plus de sécurité, le SET doit respecter certaines règles, dont voici les principales :

- L'acheteur et la passerelle de paiement doivent être capables de vérifier l'authenticité du marchand
- Le marchand doit pourvoir vérifier que l'acheteur et la passerelle de paiement sont bien authentifiés
- Les personnes non autorisées ne doivent pas pouvoir modifier les messages échangés entre la passerelle de paiement et le marchand
- Le numéro de carte de l'acheteur ne doit pas être accessible au marchand

– Le banquier ne doit pas avoir accès à la nature de la commande passée par l'acheteur.

Le SET permet de vérifier l'authenticité des parties et l'intégrité des données grâce à l'utilisation de plusieurs algorithmes et protocoles précédemment présentés :
– La confidentialité de l'information : on utilise un chiffrement RSA-1024 pour chiffrer les clés de sessions DES ou 3DES, seule la banque peut lire le numéro de carte
– L'intégrité des données : par l'intermédiaire de signatures digitales RSA, SHA-1 et HMAC
– L'authentification du propriétaire de la carte : par des certificats X.509v3 et des signatures RSA
– L'authentification du marchand : par des certificats X.509v3 et des signatures RSA.

Pour garantir cette sécurité, et permettre une compatibilité entre toutes les entités, le choix des algorithmes fut imposé.

13.1.2 Déroulement général d'une transaction

Une transaction SET nécessite dix étapes entre l'ouverture d'un compte, et la demande de paiement :

1. le client ouvre le compte : cette première étape permet les échanges SET
2. le client reçoit un certificat : après vérification de son identité, le client reçoit un certificat. Ce dernier prouve l'authenticité de la clé publique RSA du client et établit un lien entre la paire de clés et la carte de crédit fournie.
3. les négociants ont leurs propres certificats : le négociant possède 2 paire de clés : une pour signer et une pour l'échange de clé. Le négociant dispose également du certificat de la passerelle de paiement
4. le client passe une commande
5. le négociant est vérifié : via l'envoi du certificat du négociant au client
6. l'ordre et le paiement sont envoyés : le paiement contient les informations sur la carte de crédit et est chiffré, le certificat du client permet de vérifier son identité
7. le négociant demande l'autorisation de paiement : il envoie les infos de paiement au gateway de paiement et demande l'autorisation de paiement (autrement dit, il demande si le crédit est suffisant)
8. le négociant confirme l'ordre : il envoie la confirmation au client
9. le négociant fournit des marchandises ou le service
10. le négociant demande le paiement : il envoie la demande de paiement au gateway de paiement

13.1.3 Les signatures duales

Comme dit précédemment, le négociant n'a pas besoin de connaître le numéro de carte de son client. De même la banque n'a pas besoin de savoir ce que le client a acheté. Ces deux entités doivent être liées, mais n'ont pas besoin de connaître les mêmes informations. La signature duale, tout en liant ces informations, permet d'éviter la diffusion de renseignements inutiles à chacune des parties. De plus, elle pourra être utilisée comme preuve en cas de litige.

Dans une signature duale, on trouvera donc deux messages destinés à deux receveurs distincts :
– l'information de commande (OI[1]) pour le négociant,
– l'information de paiement (PI[2]) pour la banque

Par l'intermédiaire de ce procédé, la sécurité est également garantie. D'une part, moins de données sont divulguées aux diverses parties. D'autre part, imaginez qu'un marchand obtienne les informations OI et PI de manière non liée. Le marchand passe alors le PI à la banque. Si le marchand obtient un autre OI de ce client, il pourrait dire que ce dernier correspond au PI envoyé précédemment. La signature duale

[1] Order Information
[2] Payement Information

ne permet pas cette fraude.

Le fonctionnement de la signature est donnée à la figure 13.2. Le client calcule en premier lieu individuellement le haché (par SHA-1) de son PI et celui de son OI. Le haché de leur concaténation est ensuite chiffré avec sa clé privée, ce qui constitue la signature duale. On peut illustrer l'entièreté du procédé par la formule suivante :

$$DS = E_{KR_c}[H(H(PI)||H(OI)]$$

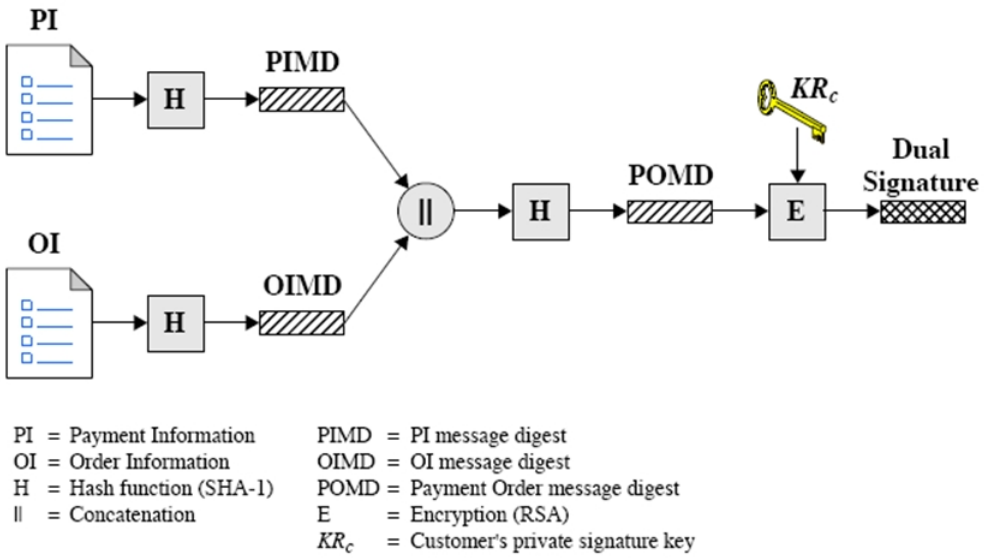

FIG. 13.2 – Fonctionnement d'une signature duale

13.1.4 Fonctionnement du paiement

De nombreux types de transaction sont possibles. Elles sont reprises sur la page du cours dans le document SET-trans.pdf.

Les principales sont "la Commande" (Purchase Request) , "l'Autorisation de paiement" (Payment Authorization) et "le Paiement" (Payment Capture). Elles sont explicitées ci-dessous.

13.1.4.1 La Commande

La première phase consiste en l'envoi d'une demande d'achat (*Purchase Request*) au marchand. Elle est illustrée par la figure 13.3.

L'enveloppe digitale représente le résultat du chiffrement de la clé de session K_s par la clé publique de la passerelle de payement. Seule la passerelle possède la clé privée correspondante. Ainsi, seule la passerelle pourra lire le PI.

A noter que le "cardholder certificate" représentant le certificat associé à la clé publique du client est également transmise à la passerelle afin que cette dernière puisse vérifier la signature duale.

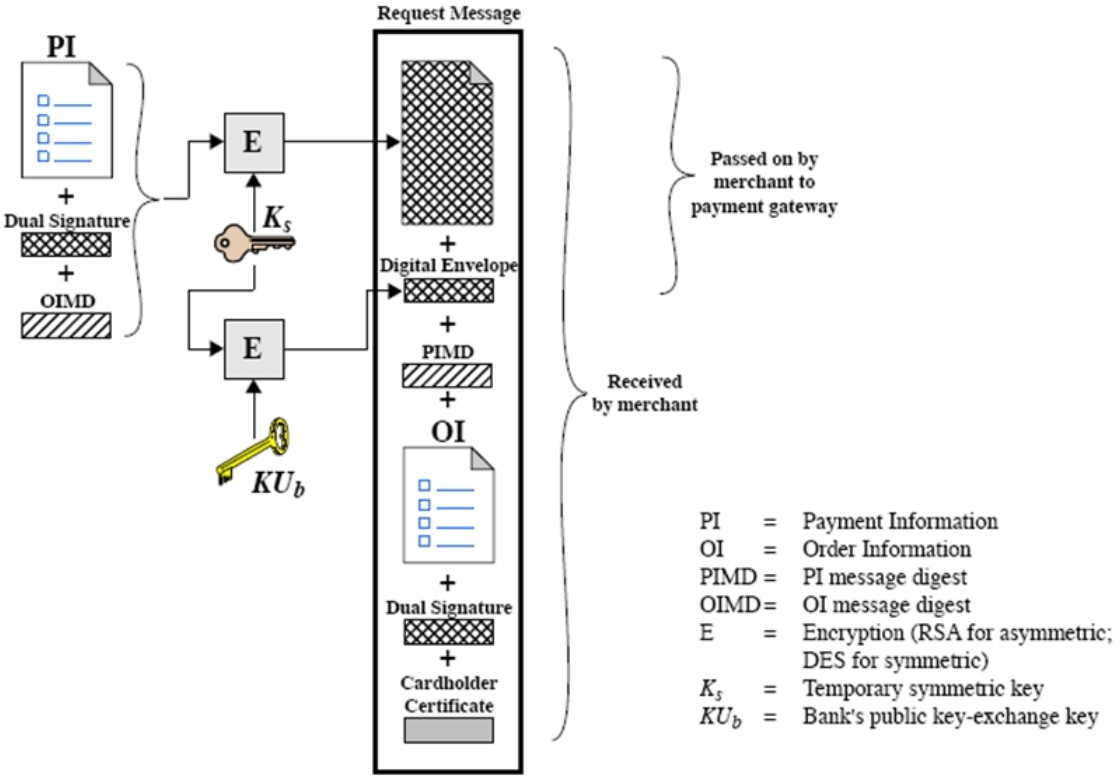

FIG. 13.3 – Commande - Requête du client

Lorsque le marchand reçoit la requête d'achat du client, il procède aux vérifications illustrées à la figure 13.4. Le déroulement est le suivant :

1. Il vérifie la signature duale en utilisant la clef publique de signature du client pour s'assurer que la commande n'a pas été manipulée pendant son transit et qu'elle a été signée en utilisant la clef privée de signature du détenteur de la carte
2. Il transmet l'information de paiement à la passerelle de paiement pour l'autorisation (décrite ci-après) et effectue la commande
3. Il envoie enfin une réponse d'achat au détenteur de la carte. Cette réponse confirme l'accord du marchand et renferme une référence à la transaction en cours. Cette réponse est signée par la clé privée du marchand.

13.1.4.2 Autorisation de Paiement

Lors de l'acceptation d'une commande, le marchand instaure une transaction avec le gateway de paiement. L'autorisation de paiement assure que la transaction a été approuvée par l'institution financière du client. Elle assure donc au marchand qu'il sera payé pour le service qu'il offre au client.

Sur la figure précédente, on voit une partie des données transmises à la passerelle par le marchand. De plus, ce dernier fournit des données supplémentaires : un bloc d'autorisation contenant un identificateur de la transaction, signé avec sa clé privée, et chiffré avec une clé symétrique qu'il aura généré pour l'occasion, ainsi qu'une enveloppe digitale protégeant cette clé symétrique.

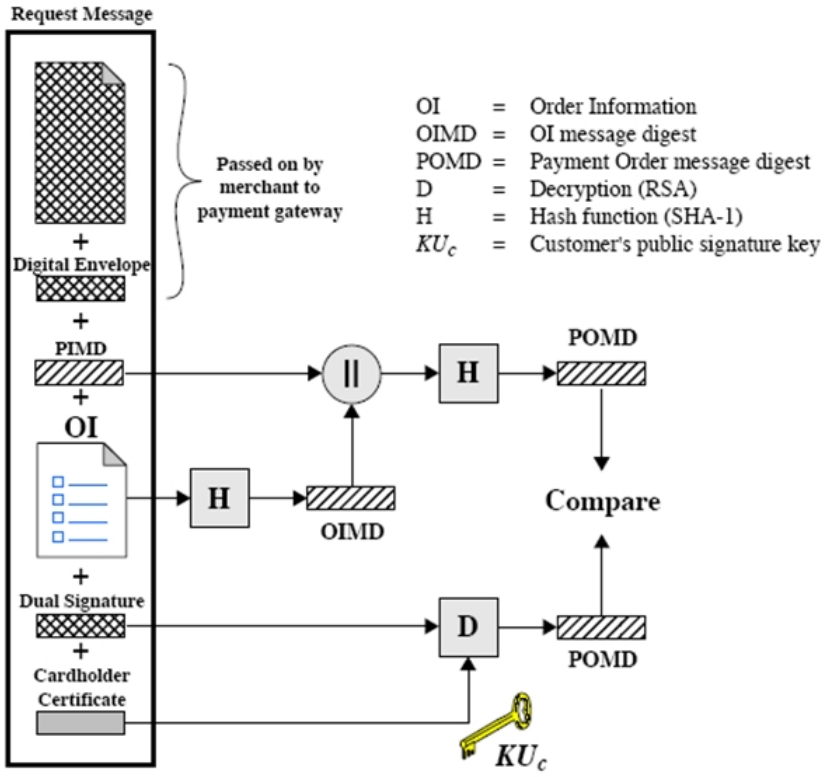

FIG. 13.4 – Commande - Vérification par le marchand

L'autorisation de la passerelle de paiement a lieu en 7 étapes :

1. Il déchiffre l'enveloppe numérique du bloc d'autorisation pour obtenir la clef symétrique et déchiffre alors le bloc d'autorisation
2. Il vérifie la signature du négociant sur le bloc d'autorisation
3. Il déchiffre l'enveloppe numérique du bloc de paiement (originaire du client et transmis par le marchand) pour obtenir la clef symétrique et déchiffre alors le bloc de paiement
4. Il vérifie la signature duale sur le bloc de paiement
5. Il vérifie que l'identificateur de transaction reçu du marchand correspond à celui reçu (indirectement) dans le PI du client
6. Il demande et reçoit une autorisation de l'institution financière
7. Il renvoie la réponse d'autorisation au marchand

13.1.5 Paiement

Cette dernière phase de paiement prend place en 4 étapes :

1. Le négociant envoie une demande de paiement au gateway de paiement
2. La passerelle vérifie la demande (vérification de la signature du marchand)
3. La passerelle effectue le transfert de fond vers le compte du négociant
4. Il informe enfin le négociant du transfert accompli.

13.2 3D-Secure

Le SET fut un échec, essentiellement dû au fait que les démarches à effectuer par le marchand sont relativement complexes. Il devait en effet établir plusieurs communications spécifiques avec le client, mais également avec sa banque et la passerelle de paiement. Au vu de ce désintérêt, un nouveau schéma de paiement fut créé à l'initiative de Visa. Par comparaison au SET, 3D-Secure repose sur un schéma simplifié, et permet une intégration et une utilisation plus simple pour le marchand et le client. La plupart des responsabilités sont maintenant transférées aux banques. La principale nouveauté en terme de sécurité est l'introduction du SSL/TLS (A l'origine, l'architecture 3D-Secure était nommé 3D-SSL). Il est maintenant généralisé (depuis le 01/03/2003) et est supporté notamment par Visa, Mastercard, American Express, ...

Le schéma global de 3D-Secure est représenté par trois domaines (d'où le nom 3D pour 3-Domains) :

1. Domaine du client - Domaine émetteur (Issuer)
2. Domaine d'interopérabilité - Domaine interbancaire
3. Domaine du marchand - Domaine acquéreur (Acquirer)

FIG. 13.5 – Aperçu de l'architecture

Chaque entité du schéma doit supporter le protocole TLS pour assurer des connexions sécurisées. Il en est de même pour les algorithmes RSA, 3DES et SHA.

Contrairement au SET, le marchand pourra ici disposer des informations bancaires du client.

13.2.1 Architecture générale

Domaine de l'émetteur (ISSUER Domain - authentifie le client)

- Le client (cardholder) :
  Il achète en ligne et fournit les infos nécessaires (numéro de carte de crédit, mot de passe pour l'authentification, date d'expiration)
- L'ACS (Access Control Server) :
  Son rôle est de vérifier si l'authentification 3D-Secure est possible pour un numéro de carte particulier, mais aussi d'authentifier un client pour chaque transaction.
- Le navigateur du client :
  Il permet une connexion sécurisée entre le MPI (Merchant Plug-In) présent dans le domaine du marchand et l'ACS

– L'émetteur (Issuer - souvent une banque) :
Il est responsable de l'obtention de la carte. Il fournit également les informations nécessaires au DS (Directory Server) présent dans le domaine d'interopérabilité. Enfin, il détermine si un client peut utiliser le service ou non.

Domaine de l'acquéreur (ACQUIRER Domain - authentifie le marchand)

– Le marchand.
– Le MPI (Merchant Plug-In) :
C'est une interface qui dialogue avec le client et le DS. C'est par son intermédiaire qu'ont lieu les principaux échanges, et qui a donc conduit à "déresponsabiliser" le marchand dans cette architecture.
– L'acquéreur (Acquirer - souvent une banque) :
Il détermine si un marchand peut participer au service 3D-Secure. Durant les transactions, c'est lui qui dialoguera avec les entités du domaine d'interopérabilité afin de gérer les demandes d'autorisation de paiement.

Domaine d'interopérabilité (INTEROPERABILITY Domain - gère les envois de messages)

– Le DS (Directory Server) :
Il détermine si un numéro de carte est valide ou non et transmet les requêtes d'authentifications des clients à l'ACS. Lorsqu'il reçoit la réponse, il la transmet au marchand.
– L'Authentication History Server (AHS) :
Il reçoit et conserve chaque message reçu de la part de l'ACS pour chaque authentification de paiement (réussie ou non). En cas de conflit entre la banque du client et celle du marchand, une copie de cet historique est consultable.
– L'établissement de crédit (ici VisaNet) :
Il reçoit les requêtes d'autorisations de l'institution financière et les transmet au fournisseur (Issuer). C'est lui aussi qui transmet la réponse reçue à la banque du marchand.

Un aperçu générale des 3 domaines est donné à la figure 13.6. Une vue plus détaillée des éléments composant chaque domaine est illustrée par la figure 13.8.

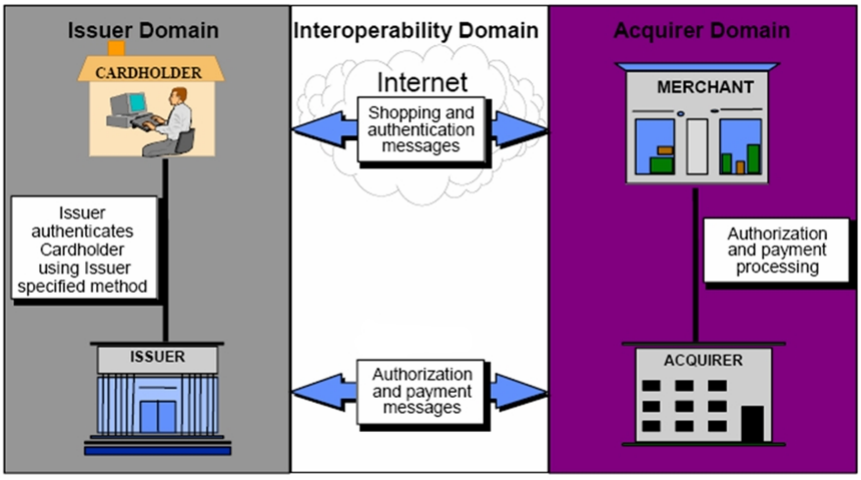

FIG. 13.6 – Architecture générale

13.2.2 Inscription

1. Le futur détenteur d'une carte se rend chez le fournisseur de service (sa banque) et fait la demande.
2. Il fournit les données nécessaires au Service d'inscription (Données d'identification, mot de passe, PAM (Personal Assurance Message)).
   La banque valide la demande du client si celui-ci est jugé apte à utiliser la carte de crédit.
3. Le serveur d'inscription (Enrollment Server) envoie la MAJ à l'ACS. Les informations y sont stockées pour les transactions futures ; le client peut maintenant faire ses achats en ligne.

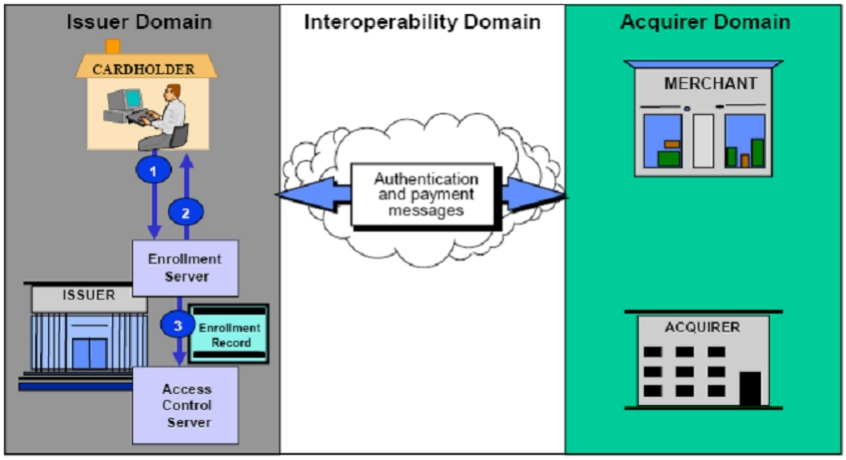

FIG. 13.7 – 3D-Secure : Inscription

13.2.3 Achat

1. Le client finalise un achat sur le site internet d'un marchand. Ce dernier a maintenant toutes les informations requises pour entamer la transaction : informations sur le client, numéro de compte personnel (Personal Account Number), ...
2. Le MPI envoie le PAN au DS.
3. Le DS interroge l'ACS afin de savoir si l'authentification est possible pour ce client.
4. Si c'est le cas, l'ACS lui répond positivement
5. Le DS transmet cette réponse au MPI
6. Le MPI envoie une PAReq (Payer Authentication Request) à l'ACS en passant par le browser du client.
7. L'ACS reçoit le PAReq
8. L'ACS authentifie le client en demandant des informations propres (mot de passe, code, etc.). L'ACS crée ensuite le PARes (PAResponse) et la signe.
9. L'ACS envoie des données d'authentification au Serveur d'historique d'authentification (AHS).
10. L'ACS envoie le PARes au MPI en passant par le browser du client.
11. Le MPI reçoit le PARes et vérifie la signature.
12. Le MPI poursuit les transactions avec son institution financière (par VisaNet).

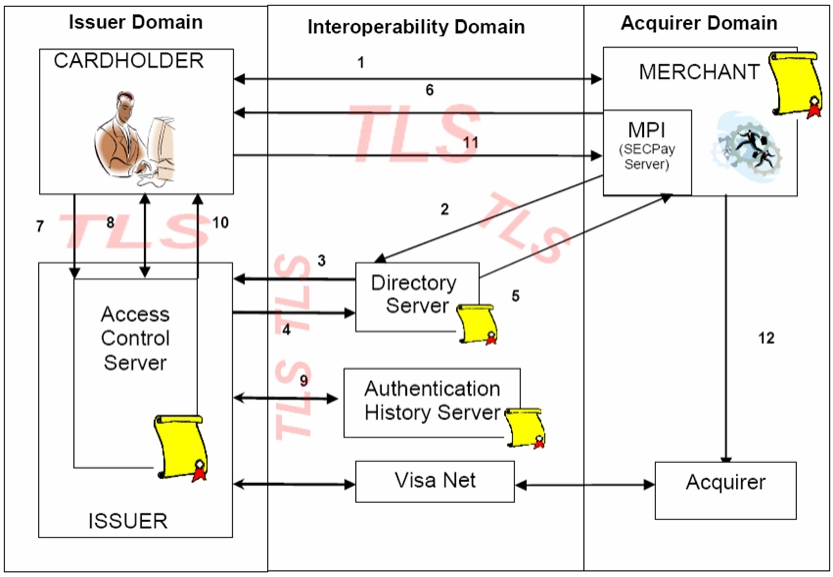

FIG. 13.8 – 3D-Secure : Achat

13.3 Autres solutions

13.3.1 C-SET

Il s'agit d'un projet français (Cyber-Comm) basé sur le protocole SET. C-SET est l'acronyme de Chip-Secure Electronic Transaction. La sécurité repose sur l'utilisation d'une carte à puce. C'est un système compatible avec le protocole SET initial moyennant une passerelle logicielle.

Le système requiert un lecteur de carte et un logiciel spécifique chez le client : la lourdeur de mise en place imposée et le coût des lecteurs (25 euros) ont fait que cette architecture n'a pas rencontré le succès souhaité.

En 2003, le projet a été officiellement stoppé.

13.3.2 SPA

C'est l'acronyme de Secure Payment Application, créé par Mastercard. Initialement destiné à concurrencer 3D-Secure, il nécessitait l'installation d'un applet, ce qui s'accompagnaient de certains problèmes de compatibilité. Sans cet applet, il était impossible d'effectuer un paiement. L'incription était plus fastidieuse et longue que celle du concurrent 3D-Secure.

En octobre 2002, ces deux principaux inconvénients ont poussé Mastercard à s'associer à Visa pour obéir à la norme 3D-Secure.

13.3.3 Ressources supplémentaires

`http://partnernetwork.visa.com/pf/3dsec/main.jsp`
`http://www.euroconex.com/products/3D_Secure.htm`
`http://www.secpay.com/secpay/index.php/content/view/full/414.html`

Chapitre 14

PGP

L'email est l'un des services réseau les plus utilisés. Le contenu des messages n'est pas sécurisé, il peut donc être inspecté durant son transit ou par les utilisateurs privilégiés sur le système de destination. On souhaite donc la mise en place de services de sécurité tels que la confidentialité, l'authentification, l'intégrité et la non-répudiation. Ces quatre services sont apportés par le logiciel PGP, que nous allons décrire en détails.

Développé par Phil Zimmermann, PGP fournit un service de confidentialité et d'authentification qui peut être employé pour des applications de type courrier électronique et stockage de dossiers. PGP repose sur une sélection et un emploi des meilleurs algorithmes cryptographiques (RSA, DSA, DH, 3DES, IDEA, CAST-128, SHA-1) disponibles intégrés dans un programme simple indépendant de l'OS. A l'origine libre, il dispose maintenant de versions commerciales. Il reste cependant disponible librement sur une grande variété de plateformes. Il possède un large éventail d'applicabilité, n'est pas développé ou commandé par des organismes de normalisation gouvernementaux et est pourtant maintenant reconnu comme un standard (RFC 3156).

14.1 Principes

PGP propose cinq services, qui seront explicités par la suite :
- Authentification
- Confidentialité
- Compression
- Compatibilité E-mail
- Segmentation

Notations utilisées

Diverses notations seront utilisées dans les schémas illustrant le principe de PGP :
- K_s : clé de session utilisée dans les schémas symétriques
- KR_a : clé privée de A
- KU_a : clé publique de A
- EP : chiffrement public
- DP : déchiffrement public
- EC : chiffrement symétrique
- DC : déchiffrement symétrique
- H : fonction de hachage
- $||$: concaténation
- Z : fonction de compression utilisant l'algorithme ZIP
- $R64$: conversion au format ASCII Radix 64

14.1.1 Authentification

Lors de cette première phase, l'expéditeur crée tout d'abord son message. Un fois terminé, SHA-1 génère un condensé de 160 bits associé. Ce dernier est chiffré avec RSA ou DSS en utilisant la clé privée de l'expéditeur, et le résultat est attaché au message.

Le récepteur utilise RSA ou DSS avec la clé publique de l'expéditeur pour déchiffrer et récupérer le condensé. Il produit ensuite un nouveau condensé du message et le compare avec le condensé déchiffré. S'ils correspondent, l'authenticité du message est démontrée. Le point (a) de la figure 14.1 illustre le mécanisme d'authentification.

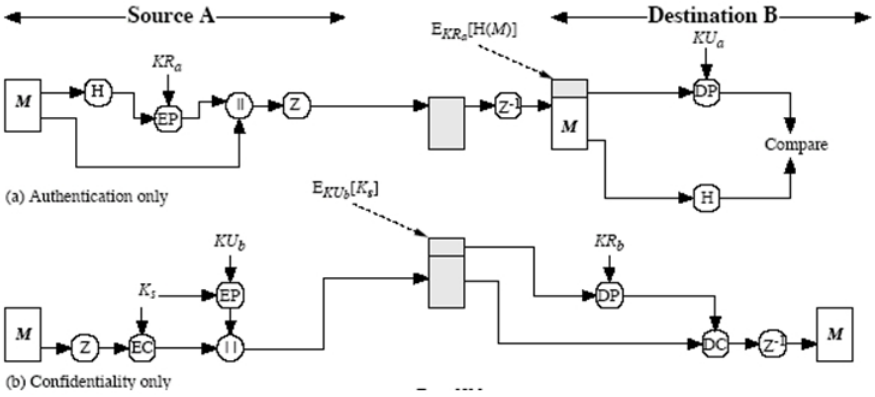

FIG. 14.1 – Authentification (a) et Confidentialité (b) de PGP

14.1.2 Confidentialité

L'expéditeur génère un message et un nombre aléatoire de 128 bits à employer comme clef de session pour ce message seulement. Le message est chiffré, en utilisant Cast-128/IDEA/3DES avec la clef de session. Celle-ci est chiffrée en utilisant le RSA avec la clef publique du destinataire, puis attachée au message.

Comme le montre le (b) de la figure 14.1, le récepteur emploie le RSA avec sa clé privée pour déchiffrer et récupérer la clé de session qui sera utilisée pour déchiffrer le message.

Confidentialité et authentification

Logiquement, on emploie les deux services sur le même message. On crée la signature que l'on attache ensuite au message. On chiffre le message et la signature. On attache ensuite la clef de session chiffrée par RSA. Cette association est illustrée à la figure 14.2.

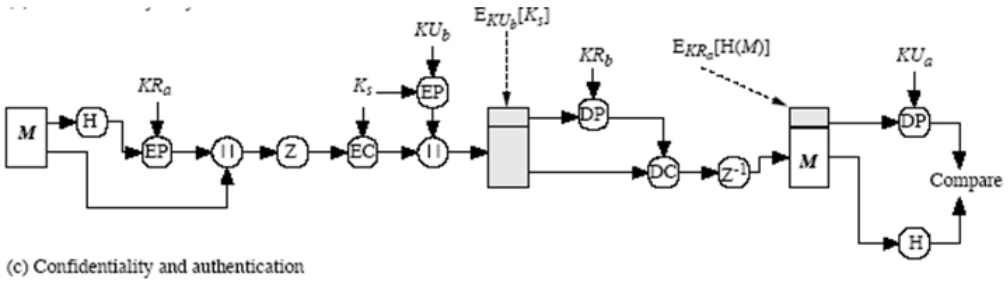

FIG. 14.2 – Authentification et Confidentialité associées dans PGP

14.1.3 Compression

Par défaut PGP compresse le message après la signature mais avant le chiffrement. Ainsi on peut stocker le message et la signature non compressés pour une vérification ultérieure. L'algorithme de compression utilisé est le ZIP. Son emplacement est critique (en raison de la redondance).

14.1.4 Compatibilité mail

L'emploi de PGP fournit des données binaires. Ainsi, tout ou partie du message consistera en une suite arbitraire d'octets. Cependant, la plupart des clients mail n'acceptent que l'utilisation de blocs ASCII. Par conséquent PGP doit coder les données binaires en caractères ASCII imprimables.

PGP emploie pour ce faire l'algorithme Radix-64 qui fait correspondre 3 bytes à 4 caractères imprimables. Il appose également un CRC. Si cela s'avère nécessaire, PGP procèdera également à la segmentation des messages.

14.1.5 Segmentation et Réassemblage

Selon les infrastructures en place, la taille des mails est souvent limitée (p. ex. à une longueur de message maximale de 50.000 octets). Ainsi, les messages plus longs doivent être découpés en segments. PGP va automatiquement subdiviser un message trop grand. En réception, il faudra enlever tous les en-têtes de mail et rassembler tous les blocs en un seul.

14.1.6 Résumé des opérations PGP

La figure 14.3 illustre schématiquement l'ordonnancement des services proposés.

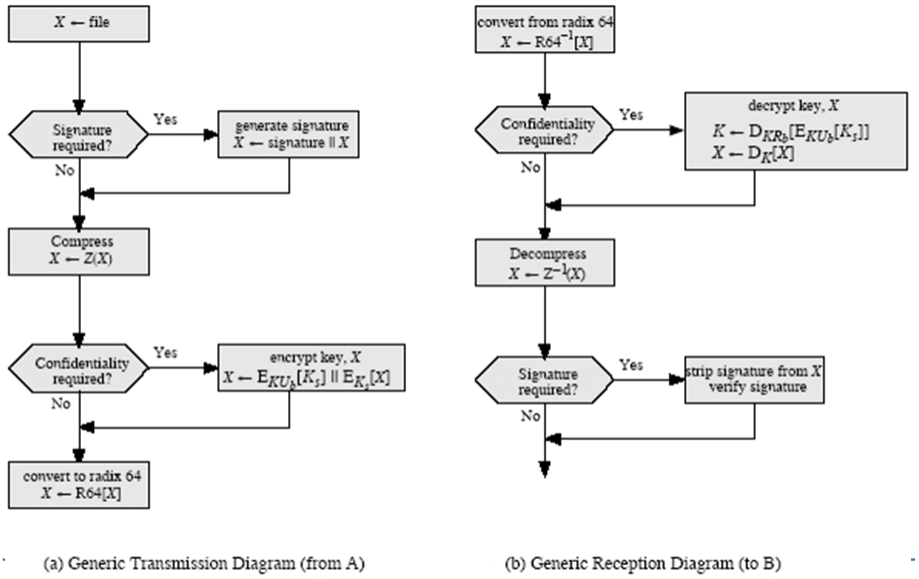

FIG. 14.3 – Opérations PGP

14.2 Format d'un message

Le format des messages utilisés par PGP est donné à la figure 14.4. La plupart des termes utilisés dans ce schéma seront explicités par la suite.

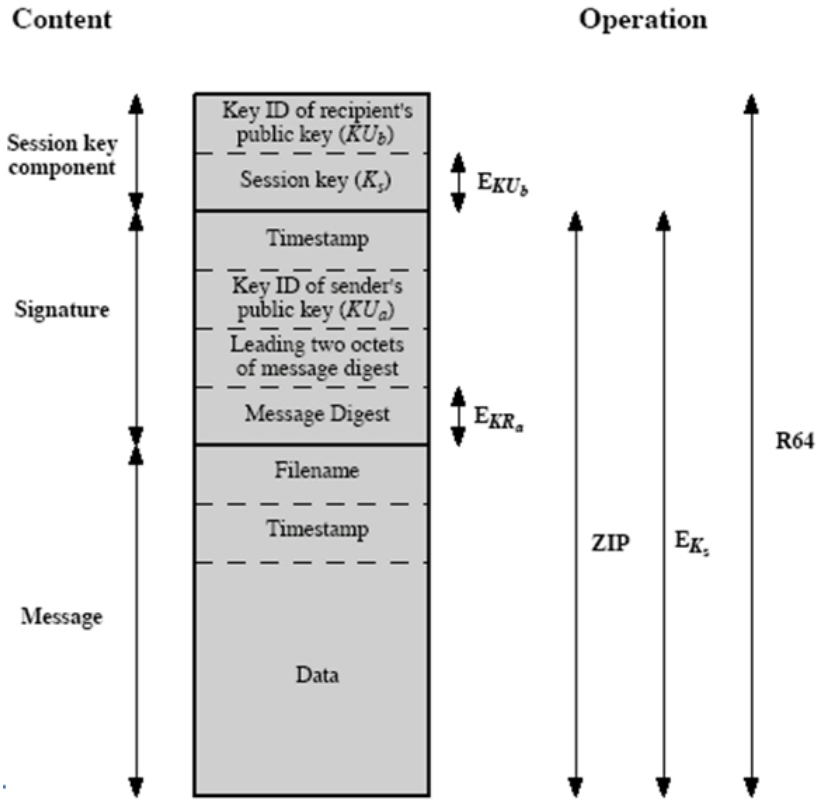

Fig. 14.4 – Format de message PGP

Identifiant de clé

Comme beaucoup de clefs publique/privée peuvent être en service (un utilisateur peut posséder plusieurs paires), il y a nécessité d'identifier quelles sont les clés employées réellement pour chiffrer la clef de session dans un message.

Il aurait été possible d'envoyer la clé publique complète avec chaque message, mais c'est inefficace et moins sûr. PGP emploie plutôt un identifiant de clé basé sur cette clé. Il prend les 64 bits les moins significatifs de la clé, ce qui permet avec une probabilité très élevée d'identifier de manière unique la clé ayant été utilisée. La même méthode est employée dans les signatures.

14.3 Clés cryptographiques et anneaux de clés

Il est nécessaire de générer des clés de session aléatoires. De plus comme on vient de le mentionner, il est possible de disposer de plusieurs paires de clés publique/privée. Il est donc obligatoire de conserver la liste de ses paires de clés mais également la liste des clés publiques de ses correspondants.

14.3.1 Clés de session

On a besoin d'une clé de session pour chaque message. Les tailles varieront selon l'algorithme utilisé : DES 56 bits, CAST ou IDEA 128 bits, Triple-DES 168 bits.

Pour introduire la notion aléatoire de la clé, PGP utilise des données issues des utilisations précédentes et de la dynamique de frappe de l'utilisateur (temps pris pour taper une séquence de lettres, le temps de pression sur une touche ou encore le temps entre deux pressions successives sur une même touche).

14.3.2 Anneaux de clés

Chaque utilisateur PGP dispose d'une paire d'anneaux :
- L'anneau de clés publiques contient toutes les clés publiques des autres utilisateurs de PGP connus de cet utilisateur, classées par identifiant de clé.
- L'anneau de clés privées contient les paires de clés publiques/privées de cet utilisateur, classées par identifiant de clé.

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
T_i	KU_i mod 2^{64}	KU_i	$E_{H(P_i)}[KR_i]$	User i
• • •	• • •	• • •	• • •	• • •

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
T_i	KU_i mod 2^{64}	KU_i	$trust\_flag_i$	User i	$trust\_flag_i$		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

* = field used to index table

FIG. 14.5 – Anneaux de clés privées et publiques

Remarque : La clé privée n'est pas conservée en tant que telle mais chiffrée (avec CAST ou IDEA ou 3DES) au moyen d'un condensé (SHA-1) obtenu à partir d'une passphrase selon la méthode suivante :

1. L'utilisateur sélectionne une passphrase, un simple mot de passe n'étant plus suffisant pour garantir une bonne sécurité.
2. Quand le système génère une nouvelle paire de clés RSA, il demande à l'utilisateur sa passphrase. En utilisant le SHA-1, il génère un condensé de 160 bits à partir de cette passphrase et la passphrase est effacée.
3. Le système chiffre la clé privée avec CAST en utilisant 128 bits du condensé. Le condensé est ensuite supprimé et la clé privée chiffrée est stockée dans l'anneau des clés privée. C'est la raison pour laquelle PGP demande la passphrase lorsque l'on désire obtenir sa clé privée.

14.3.2.1 Utilisation des anneaux pour un envoi

Dans un premier temps, on procède à la signature du message :

- Obtention de la clé privée dans l'anneau de clés privées
- Utilisation de la passphrase pour déchiffrer la clé
- Construction du composant signature

Ensuite, on procèdera au chiffrement du message :

- Génération d'une clé de session et chiffrement du message
- Obtention de la clé publique du correspondant dans l'anneau de clés publiques
- Création du composant de la clé de session

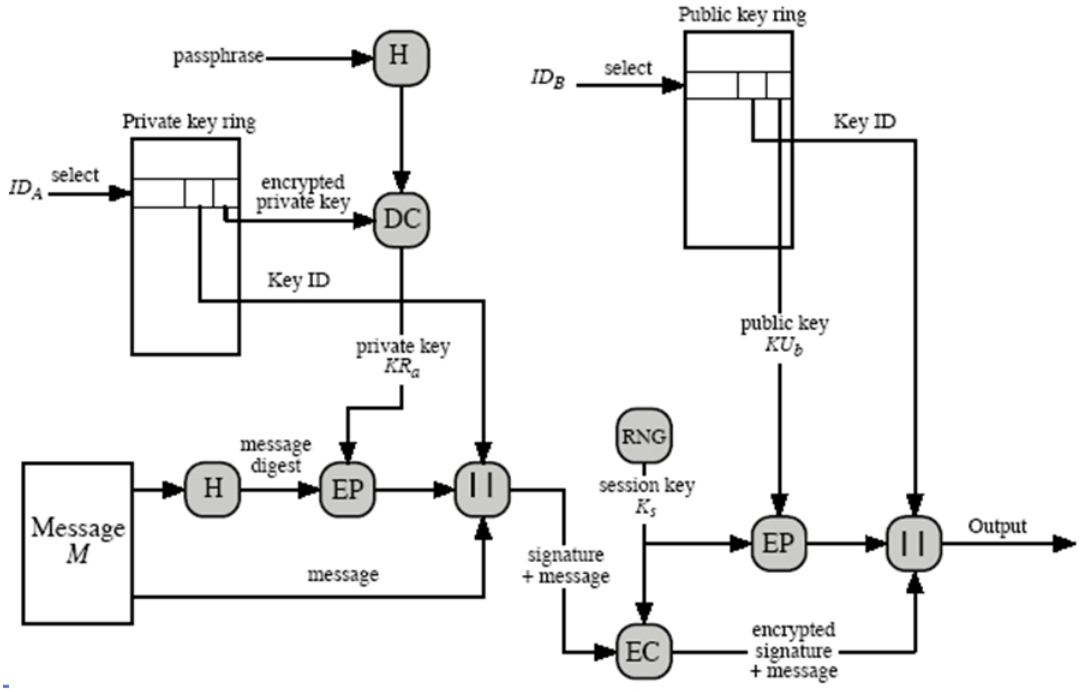

FIG. 14.6 – Génération d'un message PGP

14.3.2.2 Utilisation des anneaux pour une réception

Lors du déchiffrement du message, il viendra dans l'ordre :

- Récupération de la clé privée dans l'anneau de clés privées (obtention de l'ID de la clé dans le composant de la clé de session)
- Utilisation de la passphrase pour déchiffrer la clé
- Récupération de la clé de session et déchiffrement du message

Authentification du message :

- Récupération de la clé publique de l'expéditeur dans l'anneau de clés publiques (utilisant l'ID de la clé dans le composant de la signature)

– Récupération du condensé transmis
– Calcul du condensé du message reçu et comparaison.

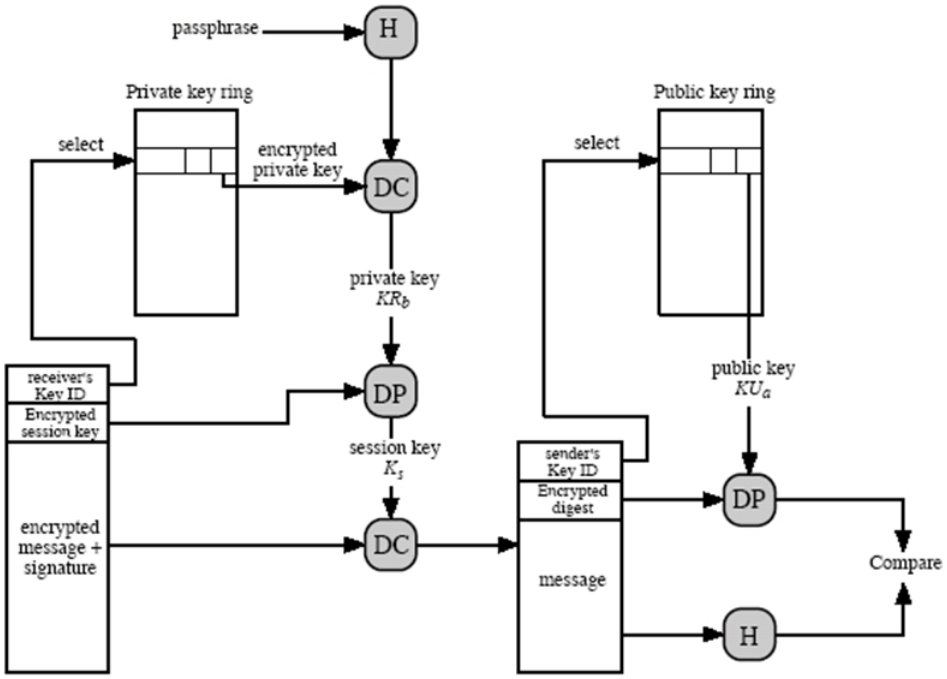

Fig. 14.7 – Réception d'un message PGP

14.3.3 Gestion des clés

Plutôt que d'employer des autorités de certificats, chaque utilisateur est son propre CA dans PGP. N'importe quel utilisateur peut signer des clefs pour des utilisateurs qu'ils connaissent directement. C'est ce que l'on appelle la *Toile de Confiance* de PGP. On peut faire confiance aux clefs d'autres utilisateurs pour autant qu'ils soient atteints par une chaîne de signatures. L'anneau des clés publiques inclut des indicateurs de confiance (Key legitimacy, Signature Trust, et Owner Trust de la figure 14.5).

Remarques relatives au schéma 14.8

– Vous signez toutes les clés appartenant à des utilisateurs à qui vous faites confiance (ou partiellement confiance) à l'exception du noeud L. Ces signatures personnelles ne sont pas obligatoires (comme on le voit avec L), mais sont souhaitables.
On le remarque avec le noeud E, qui est signé par F à qui vous faites confiance. Dans la majorité des cas, il sera préférable de "confirmer" la signature de F par une signature vous appartenant. Ainsi, il serait préférable que vous signiez la clé de L.
– On considère que deux utilisateurs à qui vous faites partiellement confiance sont suffisants pour certifier la clé d'un autre utilisateur. Ainsi, la clé de H est certifiée valide par la présence des signatures de A et de B, utilisateurs auxquels vous faites partiellement confiance.
– Il arrive que certaines clés soient certifiées, alors que vous n'avez pas confiance en son propriétaire. C'est le cas pour l'utilisateur N. Celui-ci est signé par E, sa clé est donc certifiée valide. Mais bien que vous ayez confiance en E, cette confiance n'est pas transitive. Pour l'utilisateur R, bien que signé par N dont la clé est validée, PGP ne le considère pas comme digne de confiance. Cette situation est tout à fait plausible : il peut arriver que vous souhaitiez envoyer des données chiffrées

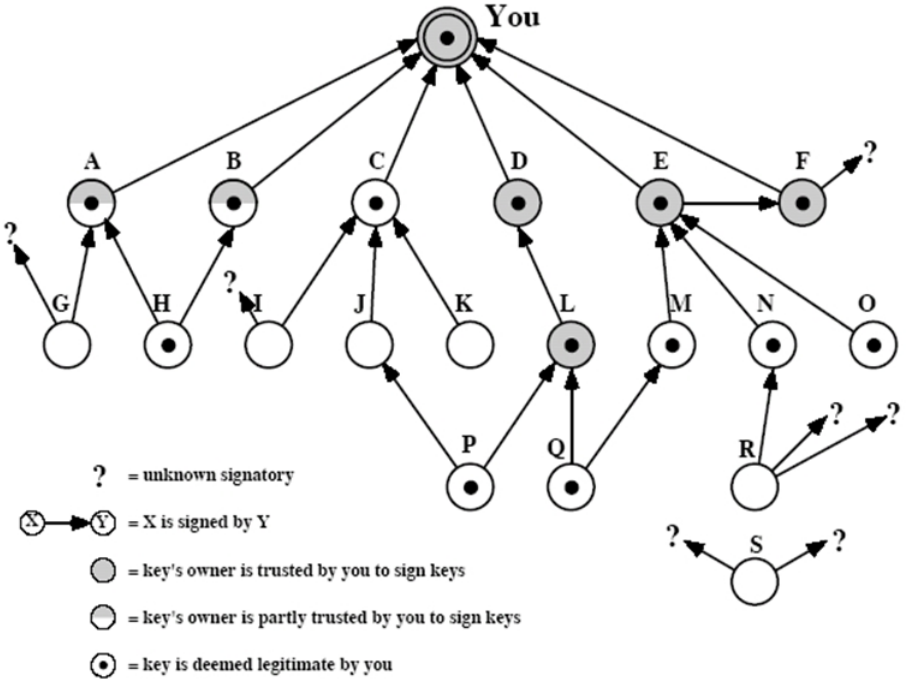

FIG. 14.8 – Toile de confiance

à un utilisateur quelconque en qui vous n'avez pas confiance, cependant, la clé utilisée pour chiffrer doit réellement être la clé de ce correspondant.
– Le noeud S est signé par deux utilisateurs inconnus (un tel cas peut avoir lieu si cette clé est obtenue à partir d'un serveur de clés.) PGP ne considère pas cette clé comme valide, même si elle provient d'un serveur certifié valide.

14.3.4 Révocation des clés publiques

Il est également possible pour les utilisateurs de révoquer leurs clefs. Le propriétaire génère alors *un certificat de révocation de clé*. Il s'agit d'un certificat de signature muni d'un indicateur de révocation. La clé privée correspondante est employée pour signer ce certificat.

14.4 Ressources supplémentaires

`http://www.gamers.org/~tony/pgp.html`
`http://axion.physics.ubc.ca/pgp-attack.html`
`http://www.faqs.org/faqs/pgp-faq/`

Chapitre 15

S/Mime

15.1 SMTP

Le protocole SMTP[1] est le protocole de base pour les communications de type mail. Ses spécifications sont disponibles dans le RFC 822. L'inconvénient majeur de ce protocole est qu'il est relativement limité pour les besoins actuels. Ainsi, il ne peut pas transmettre, ou en tout cas éprouve des difficultés à transmettre :

- les dossiers exécutables,
- les caractères "de langue nationale" (non-ASCII),
- les messages de taille supérieure à une certaine limite,
- les problèmes de traduction ASCII vers EBCDIC,
- les lignes de messages plus longues qu'une certaine longueur (72 à 254 caractères),
- ...

Le protocole MIME[2] tente de résoudre ces problèmes.

15.2 MIME

On peut trouver ses spécifications dans les documents RFC 2045 à 2049. Elles consistent en 3 modifications majeures :

- 5 nouveaux champs d'en-tête sont définis, donnant des informations sur le contenu du message,
- Un certain nombre de formats de contenu ont été définis, de manière à supporter en standard les mails multimédia
- Plusieurs codages permettant de convertir n'importe quel format de contenu en un autre pour éviter les altérations du message par les systèmes mails

Les en-têtes MIME

Les cinq en-têtes définies dans ce protocole sont reprises ci-dessous :

- **MIME-Version** : la version indiquée doit être "1.0".
- **Content-Type** : plusieurs types ont été ajoutés (par exemple "application/word", "text/plain",...)
- **Content-Transfer-Encoding** : indique comment le message a été encodé (radix-64)
- **Content-ID (facultatif)** : représente une chaine d'identification unique
- **Content Description (facultatif)** : description textuelle de l'objet transmis. C'est nécessaire quand le contenu n'est pas un texte lisible (p. ex. des données audio).

[1] Simple Mail Transfert Protocol.
[2] Multipurpose Internet Mail Extension.

15.3 S/MIME

Cet acronyme signifie Secure/Multipurpose Internet Mail Extension et concerne donc une amélioration sécurisée du format MIME. Il est défini dans un grand nombre de documents dont principalement les RFC 2630, 2632, 2633.

S/MIME va probablement devenir le futur standard de l'industrie, alors que PGP sera plutôt utilisé pour la sécurité des mails personnels. Actuellement, S/MIME est reconnu par la plupart des architectures mails.

15.3.1 Possibilités S/MIME

Les possibilités générales de S/MIME sont proches de celles apportées par PGP, à savoir signer et chiffrer les messages.

Mais S/MIME permet également quatre configurations de transferts :

- **Données enveloppées** :
  Le contenu du message et les clés de session sont chiffrées pour les destinataires.
- **Données signées** :
  Le condensé du message est chiffré avec la clé privée du signataire. Le texte et ce condensé sont ensuite encodés en base64. Un tel message ne peut être lu que si le destinataire est compatible S/MIME.
- **Données signées en clair** :
  Seul le condensé est encodé en base64. Le texte reste en clair. Ainsi, un système non compatible S/MIME peut lire le message mais ne peut pas vérifier la signature.
- **Données signées et chiffrées** :
  C'est une combinaison des modes précédents. Ainsi, des données chiffrées peuvent être signées, et inversément.

15.3.2 Algorithmes utilisés

Tout comme PGP, plusieurs algorithmes sont employés dans le protocole. En ce qui concerne les fonctions de hachage, SHA-1 et MD5 sont utilisés. Pour la création des signatures digitales, le DSS est disponible. Pour les opérations de chiffrement symétrique, on peut utiliser soit le 3DES, ou le RC2/40[3]. Enfin, dans le cas des chiffrements par clé publique, le RSA avec des clés de 1024 bits est conseillé, ainsi que Diffie-Hellman pour l'envoi des clés de session.

15.3.3 Gestion des certificats

S/MIME utilise des certificats X.509 v3. Cette gestion est un système hybride entre la hiérarchie de CA X.509 et la toile de confiance de PGP. Dans le cas présent, chaque client dispose d'une liste de clés auxquelles il fait confiance, ainsi qu'une CRL. Le client doit donc maintenir les certificats à jour pour garantir une certaine sécurité. De plus ces certificats doivent être signés par des CA.

15.3.3.1 Rôle de l'utilisateur

L'utilisateur doit gérer ses clés, et doit pour ce faire, veiller à trois points importants :

- Génération de clé : en utilisant Diffie-Hellman (obligatoire), DSS (obligatoire) et RSA (souhaité).
- Enregistrement : les clés publiques doivent être enregistrées auprès d'une CA X.509 afin de pouvoir obtenir un certificat.
- Stockage du certificat : les utilisateurs doivent pouvoir accéder à une liste locale des certificats pour permettre la vérification des signatures et le chiffrement des messages sortants.

[3] Chiffrement symétrique à clé de 40 bits.

15.3.3.2 Autorités de certification

Il en existe de nombreuses, Verisign étant une des plus utilisées et compatible avec S/MIME. Elle propose différents types de certificats avec des niveaux de confiance et de vérification croissants. Ces différents certificats sont explicités dans le tableau 15.1. Les certificats ainsi délivrés sont à la norme X.509. Ils sont tous identifiables par un Digital ID. Ce Digital ID reprend au minimum :

- La clé publique du propriétaire du certificat, ainsi que son nom ou son alias
- La date d'expiration du certificat ainsi que son numéro de série
- Le nom du CA l'ayant délivré
- La signature digitale de ce CA

Les certificats peuvent appartenir à une certaine classe selon le niveau de sécurité souhaité :

- Classe 1 : la vérification online est suffisante
- Classe 2 : la vérification se fait également par adresse postale
- Classe 3 : la vérification doit impliquer la personne physique (par signature manuelle notamment)

Classe	Vérif d'identité	Usage
1	vérif nom/email	web browsing/email
2+	vérif adress/inscrip	email, souscription online, validation software
3+	identification des documents	accès e-banking/services

Tab. 15.1 – Certificats VeriSign

15.3.4 ESS - Enhanced Security Services

Il s'agit d'un ensemble de fonctions de sécurité pour S/MIME. Ces fonctions permettent de réaliser 4 services :

- *un renvoi d'accusé de réception (d'un message M envoyé et signé par A) signé par B* : après avoir contrôlé la signature de A, B envoie cet accusé, contenant une référence du message M et la signature de A pour ce message, le tout signé par B. A vérifie alors que sa signature (présente dans l'accusé) correspond à la signature qu'il avait envoyé. Si c'est le cas, B a bien reçu M.
- *les labels de sécurité* : ils permettent d'identifier la politique de sécurité associée au contenu du message, ou encore d'indiquer le type d'utilisateurs autorisés à accéder à l'objet en question.
- *l'utilisation d'agents de diffusion (MLA - Message List Agent)* : lorsque A désire envoyer un message, il communique avec son MLA qui devra transmettre ce message aux destinataires. Ce dernier pourra au besoin communiquer avec d'autres MLA. L'avantage de cette technique est que A délègue le travail (à son MLA), ce qui simplifie la tâche de l'utilisateur. Bien sûr, les utilisateurs doivent avoir confiance en leur MLA.
- *l'utilisation de clés générées par un protocole de gestions de clés (type ISAKMP)*

Chapitre 16

Le monde quantique

16.1 Le calculateur quantique

L'informatique quantique est encore à ses balbutiements. On ne sait d'ailleurs pas trop où la recherche en est actuellement. Cependant, on peut voir que deux approches radicalement opposées existent. D'une part, les concepteurs de transistors qui tentent, tant bien que mal, de supprimer "l'effet quantique" (effet observé lorsque la taille des puces diminuent), et d'autre part, les chercheurs qui souhaitent travailler avec cet effet (l'effet étant présent et a priori inévitable, ils préfèrent en tirer profit). C'est à cette deuxième voie de recherche que nous allons nous intéresser, en restant toutefois très large dans les notions évoquées.

Le calcultateur quantique (ou ordinateur quantique) utilise les particules élémentaires (électrons, protons et photons) qui portent le nom de « quanta ». A la différence de l'ordinateur classique, utilisant des bits et agissant de manière séquentiel, le calculateur quantique utilise les qubits (ou qbits). Un qubit (QUatum Bit), ou bit quantique, est un état quantique représentant la plus petite unité de stockage d'information quantique. A la différence du bit qui prend une valeur 0 ou 1, le qubit peut identifier une valeur intermédiaire, un "mélange" des états 0 et 1. Par exemple, il peut représenté un état constitué de 25% de 1 et 75% de 0. On parle alors de *superposition*.

Le principe de superposition représente la possibilité pour une particule d'être en deux endroits en même temps. Ainsi, pour déterminer ces qubits, toute particule pouvant avoir deux états simultanés peut convenir. Il peut s'agir de photons présents en 2 endroits simultanément (telle l'expérience des fentes de Young), ou encore des électrons et leurs orbitales. Par la propriété de superposition, les calculs peuvent être réalisés en parallèle, d'où une puissance de calcul décuplée.

Prenons par exemple 3 bits. Un ordinateur classique utilisant 3 bits peut être dans 8 états différents (2^3). Dans le cas d'un ordinateur quantique, les 3 qubits peuvent être une superposition de ces états : il pourra donc traiter ces 8 états en parallèle. En général, des ordinateurs quantiques travaillant sur N bits pourront gérer 2^N informations en parallèle.
Théoriquement, un ordinateur quantique de 100 qubits pourrait simuler un cerveau humain. Un ordinateur de 300 qubits permettrait de simuler l'évolution de la planète depuis le Big Bang.

L'ordinateur quantique existe en théorie depuis le milieu des années 80 (Deutsch, Bennett, Benioff, Feynman). Un algorithme est paru en 1994 par Peter Shor, permettant de factoriser de très grands nombres en un temps polynomial à l'aide des ordinateurs quantiques. Ainsi, un nombre de 300 chiffres serait factorisé en à peine 10 secondes ! Cet algorithme, faute de réel ordinateur quantique, ne reste que théorique.

C'est Isaac Chuang (IBM) qui a établi le premier prototype. En 1996 il fonctionnait sur 2 qubits, en 1999 sur 3, en 2000 sur 5. Le meilleur résultat fut atteint en 2001 sur 7 qubits. Il permit de factoriser

la valeur 15 = 5 * 3. Même si ce résultat peut paraître dérisoire, il doit être rapporté aux recherches effectuées. Si des résultats plus significatifs peuvent être apportés, c'est toute la cryptographie classique (de type chiffrement symétrique, asymétrique, protection par mot de passe, etc.) qui s'effondrera.

L'ordinateur de 2001 utilise des atomes de carbone et de fluor ainsi qu'une machine à résonance magnétique (aimants supra-conducteurs plongés dans un bain d'hélium liquide à -269°C. D'autres types d'ordinateurs quantiques existent : intégrer des qubits dans un substrat solide (NEC), emprisonner des photons dans des cavités optiques.

Il y a cependant des contraintes de construction. L'ordinateur doit être par exemple être totalement isolé du monde extérieur, car ce dernier provoque la décohérence du système. De plus, il ne peut y avoir aucune perte d'information.

16.1.1 Ordinateur quantique VS ordinateur classique

Le calculateur quantique devance l'ordinateur classique sur plusieurs points :
- La résolution de problèmes complexes.
- L'aide aux physiciens dans les calculs quantiques, souvent trop complexes.
- La recherche accélérée dans des bases de données[1].

16.1.2 Risques

Les calculateurs quantiques permettent donc d'obtenir beaucoup plus rapidement des solutions à certains problèmes. En cas de mise en oeuvre du concept à grande échelle, c'est la sécurité de la cryptographie "moderne" qui est mise en jeu.

Cependant, malgré les recherches en cours, il ne semble plus y avoir aucune amélioration significative depuis les expériences citées plus haut. Le fait de ne rien découvrir de neuf pourrait être dû à deux raisons : le plafonnement (les techniques actuelles utilisées dans ce domaine ne permettent pas d'aller plus loin), et l'aspect secret d'une telle découverte (la mise sur pied d'un ordinateur quantique puissant donnerait un pouvoir énorme à celui le détenant).

Il existe tout de même une solution pour rendre les communications plus sûres, même face à l'ordinateur quantique : la cryptographie quantique. Celle-ci ne permet pas d'éviter l'interception, mais permet de la détecter.

16.2 La cryptographie quantique

16.2.1 Fondements et principes

La cryptographie quantique repose sur trois domaines distincts :
- La cryptographie, en ce sens qu'elle permet de garantir la confidentialité d'une clé.
- La physique quantique, et plus particulièrement la mécanique quantique.
- La théorie de l'information, car elle fournit un "système" inconditionnellement sûr.

Le problème majeur en cryptographie est le transfert de la clé entre les deux parties communicantes. Ici, on n'émet aucune hypothèse sur la sécurité du canal employé. La raison en est que la cryptographie quantique repose sur le principe d'Heisenberg :

Certaines quantités subatomiques ne peuvent être simultanément mesurées.

La conséquence de ce principe est qu'il est impossible de mesurer ces particules sans les modifier. Il est donc possible de construire un canal de communication que nul ne peut espionner sans modifier

[1] quantum database search : par l'algorithme de Grover (recherche exhaustive)

la transmission de manière détectable. Ainsi, il est possible de transmettre une clé secrète entre deux personnes sans qu'elles disposent d'informations secrètes communes préalables.

Dans le cadre du transport d'une clé, la technique qui nous occupera ici consistera en l'envoi de photons. On utilisera la technique dite de *Polarisation de photons*.

16.2.2 Polarisation de photons

Chaque photon peut être polarisé, c'est-à-dire qu'il est possible d'imposer une direction à son champ magnétique. Cette polarisation des photons est mesurée par un angle variant de 0° à 180°. Par simplification des modèles théoriques, on considère souvent 4 angles précis : 0°, 45°, 90° et 135°.

On parlera de polarisation rectiligne pour les angles de 0° et 90° (pour lesquelles on utilisera des filtres ou bases "standards"), et de polarisation diagonale pour les angles de 45° et 135° (filtres ou bases "diagonales"). Pour la détection, on aura recours à un filtre (ou base) polarisant(e) et un détecteur de photons. Ce type de filtre permet la lecture de photons polarisés d'une certaine manière. En conséquence, il bloque les photons polarisés dans la direction perpendiculaire.

Dans l'illustration 16.1, le filtre est positionné de telle sorte qu'il laisse passer les photons polarisés à 0° et par conséquent, bloque les photons polarisés à 90°. On est donc ici en présence d'un filtre standard.

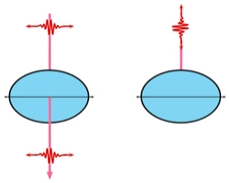

FIG. 16.1 – Filtre polarisant
Source : http ://www.iro.umontreal.ca/∼crepeau/PRL93/MQC.html

16.2.3 Détecteurs de photons (Photodétecteurs)

A la suite de ce filtre, il est nécessaire d'utiliser un outil permettant de détecter ces photons. On parle de photodétecteurs.

Soit un angle θ pour le photon à détecter. Après un passage dans un filtre standard, le photodétecteur placé à sa suite détectera le photon avec une probabilité de $cos^2\theta$ et de ne pas le détecter avec une probabilité de $sin^2\theta$.

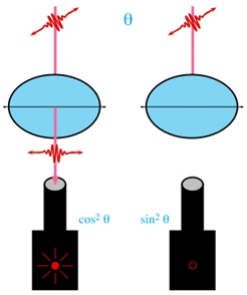

FIG. 16.2 – Détecteurs de photons
Source : http ://www.iro.umontreal.ca/∼crepeau/PRL93/MQC.html

Pour un filtre d'angle général, le raisonnement est semblable. Soit A, l'angle du photon et B l'angle du filtre. On obtient une transmission avec une probabilité de $COS^2(A-B)$ et non transmission avec une probabilité de $SIN^2(A-B)$.

Par convention, on considère qu'un photon polarisé à 0° ou 45° représentent un 0 et un photon polarisé à 90° ou 135° représente un 1. Le problème intervient lorsqu'un photon polarisé diagonalement (45° ou 135°) rencontre un filtre standard (et inversément). Il y a alors incertitude, comme on le voit sur la figure 16.3.

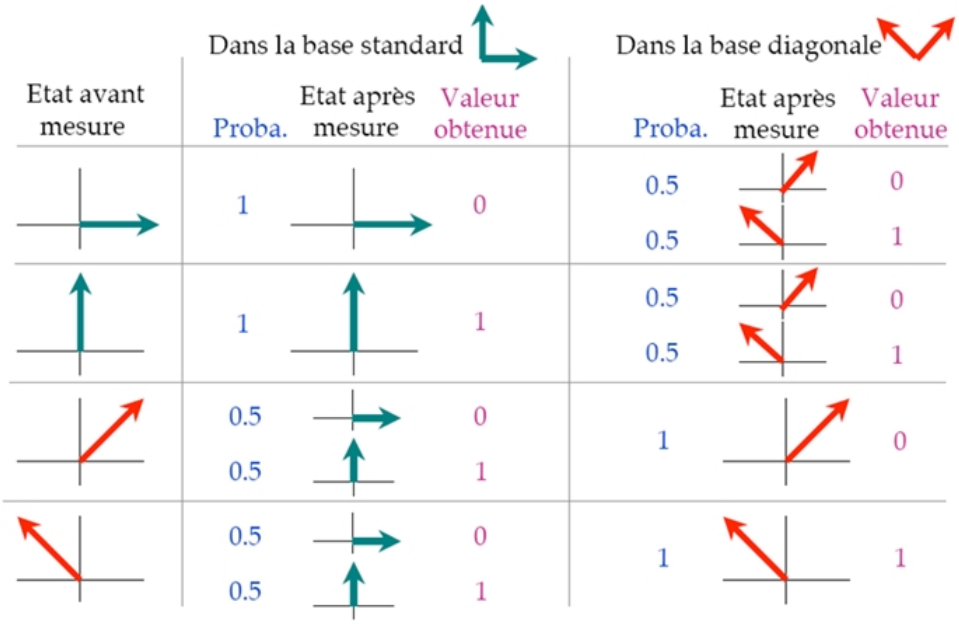

FIG. 16.3 – Détection sûre et incertitude
Source : P. Jorrand, CNRS Labo Leibniz, France,2006

16.2.4 Exemple de communication : Le protocole BB84

Soient Alice et Bob, souhaitant se transmettre une clé secrète sous forme binaire. Deux canaux sont disponibles : un canal quantique pour l'envoi de la clé, et un canal non protégé (canal radio par exemple) pour la vérification de la clé. Alice émet des photons polarisés et Bob mesure aléatoirement leur polarisation (rectiligne ou diagonale).

En moyenne, une mesure sur deux sera sans intérêt, puisque à priori, Bob ne connait pas l'orientation des filtres utilisés par Alice. L'exemple est illustré par la figure 16.4.

A la fin de la transmission, Bob possède un ensemble de bits, formant la *clé brute*. Après vérification sur le canal non protégé, seuls les bits 1,3,4 et 7 seront valables et constitueront *la clé tamisée*.

16.2.4.1 Pourquoi ce type de communication est-il sûr ?

La preuve de la sécurité de ce type de transmission est basée sur deux observations :

1. Pour réaliser une écoute sur le canal et ainsi intercepter la clé, il faut obligatoirement utiliser le même type d'outils (filtre et photodétecteur).

Alice	Valeur	0	0	1	1	1	0	0	1
	Filtre utilisé								
	Photon émis								
Bob	Filtre utilisé								
	Valeur lue	0	'1'	1	1	'1'	'0'	0	'1'
C. n. prtg.	Filtre de Bob								
	Même pour Alice								
Résultats	Bits à négliger		X			X	X		X
	Bits de la clé	0		1	1			0	

FIG. 16.4 – Protocole BB84

2. A la suite d'une interception, le pirate doit réémettre un photon polarisé comme celui qu'il aura lu.

Bob ayant reçu l'entièreté des bits, il communique avec Alice pour connaître l'orientation des filtres. Une fois les mauvaises mesures écartées, ils décident de tester au hasard quelques bits de leur clé. Par "tester un bit", il faut entendre : communiquer la valeur du bit sur le canal non protégé. Ce canal étant publique, les bits sont considérés perdus.
Dans l'exemple de la figure 16.5, en testant le troisième bit, ils se rendent compte de l'écoute. Pour un nombre assez élevé de tests, la sécurité est absolue (ou tout du moins considérée comme telle).

En effet, la probabilité qu'Eve choisisse un filtre incorrecte est de 50%, et donc elle a 50% de chance de se tromper en réémettant le photon. Si Bob lit ce photon réémis en utilisant le même filtre que celui utilisé pour l'émission par Alice (dans le cas contraire, le bit n'est pas retenu pour la clé), il a aussi 50% de chance de se tromper. La probabilité qu'un photon intercepté donne une erreur dans la clé est donc de 50% x 50% = 25%. Autrement dit, il y a 75% de chance que l'interception ne provoque pas d'erreur, ou encore, il y a 75% de chance qu'Eve ne soit pas détectée.

Si Alice et Bob comparent n bits de la clé sur le canal public, la probabilité qu'ils découvrent la présence d'Eve est égale à

$$P_d = 1 - \left(\frac{3}{4}\right)^n$$

Pour $n = 72$, on obtient $P_d = 0.999999999$. Il est conseillé d'envoyer $4n$ bits pour obtenir une clé tamisée de taille n.

Alice et Bob peuvent aussi utiliser des tests de parité :
- Alice choisit aléatoirement une suite de bits qu'elle pense partager avec Bob.
- Sur le canal publique, Alice envoie la parité de la suite et leur position dans le flux global. La parité pourra être simplement la réponse à la question "Y a t'il un nombre pair ou impair de 1 ?". La position sera transmise sous la forme "j'ai choisi le bit n°1, le bit n°4, le bit n°9, etc.".
- Bob vérifie avec la suite qu'il a détecté : si cela ne correspond pas c'est que la clé est compromise, si cela correspond, on ne peut encore rien conclure.
- Alice répète le processus plusieurs fois sur des suites distinctes.
- Après k vérifications, la probabilité de se tromper en assurant qu'ils n'ont pas été écoutés sera de 2^{-k}.

Alice	Valeur	0	0	1	1	1	0	0	1
	Filtre utilisé								
	Photon émis								
Caroline	Filtre utilisé								
	Valeur lue	'1'	0	'1'	'0'	'0'	'0'	0	'1'
	Photon réémis								
Bob	Filtre utilisé								
	Valeur lue	'0'	'0'	'1'	'0'	0	0	0	1
C. n. prtg.	Filtre de Bob								
	Même pour Alice								
Résultats	Bits à négliger		X			X	X		X
	Bits de la clé	0		1	0			0	

FIG. 16.5 – Protocole BB84 avec écoute pirate

En résumé, on peut représenter l'obtention d'une clé secrète en trois étapes, telles qu'illustrées à la figure 16.6.

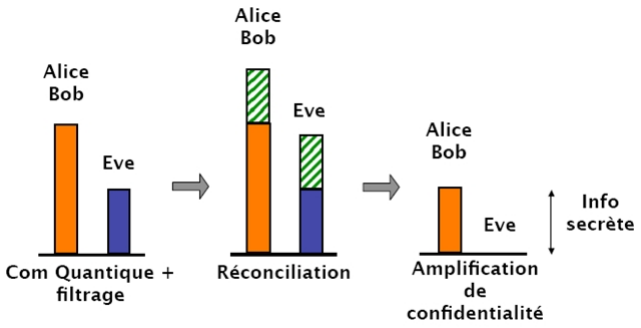

FIG. 16.6 – Les 3 étapes nécessaires à l'obtention d'une clé secrète
Source : R. Alléaume, Dép. Inf. et Réseaux, ENS Télécommunications, Paris, 2007

16.2.4.2 Résolution des problèmes

Le premier problème auquel il fallut faire face était la manipulation de photons isolés. La solution envisagée fut de travailler à partir d'impulsions de photons, courtes et polarisées, la modification de polarisation se faisant très rapidement grâce à une *Cellule de Pockels*.

Cependant, les impulsions impliquaient par définition plusieurs photons, et donc plus de facilité pour

déterminer la polarisation des photons par le pirate.

A l'heure actuelle, pour travailler photon par photon, il faut utiliser un système muni de 4 lasers possédant chacun un filtre polarisant différent. La puissance du laser est réduite de telle sorte qu'on n'émet qu'un photon à la fois.

Un deuxième problème est qu'il peut arriver que certains photons soient corrompus, même en l'absence d'écoute pirate (par une notion différente de l'horizontal entre les deux entités communicantes par exemple). Cela est dû à l'imperfection du matériel utilisé. Tout comme dans le cas classique, il existe une série de codes correcteurs d'erreurs, regroupé dans le monde quantique sous la notion de QEC (Quantum Error Correction).

Un autre domaine demande des recherches actives : la qualité du support du canal quantique. Il est actuellement composé d'une fibre optique, avec les inconvénients que cela apporte :
- La réflexion des photons sur les parois de la fibre modifie la polarisation.
- La perte de photons si la distance est trop longue.

On trouve aujourd'hui des fibres optiques spécialement dédiées au transport de photons. On parle dans ce cas de *Fibres Optiques Photoniques* (FOP).

16.2.4.3 En pratique

La première expérience de cryptographie quantique a eu lieu en 1991. Elle consista en l'envoi et à la réception de 2000 bits. En tenant compte des erreurs durant le transfert, de l'espionnage et des bits diffusés sur le canal public, la clé secrète finale aurait eu une taille de $\pm$800 bits. Par manque de technologie, cette expérience n'a permis que le transfert d'un bit par seconde sur une distance de 32 centimètres.

En 2003, Toshiba a réalisé une expérience similaire sur une centaine de kilomètres.

La technique décrite ici est aujourd'hui fonctionnelle et porte le nom de QKD (Quantum Key Distribution). La première expérience "commerciale" eut lieu à Genève en 2007 lors des élections fédérales.

Son adoption est toutefois lente, pour plusieurs raisons :
- Il s'agit d'une technologie émergente
- Son coût est assez élevé (environ 5000$ par capteur)
- Il existe une série de contraintes physiques (fibre optique, portée limitée, chiffrement P2P)
- Le débit offert actuellement est assez limité (autour d'1 Mbit maximum [10/2008])

16.3 Conclusions

Bien que reposant sur des lois de Physique quantique relativement abstraites pour le commun des mortels, la cryptographie quantique trouve ses fondements dans des concepts bien plus sûrs que la cryptographie "classique". Cette dernière, se fiant uniquement à une puissance de calcul limitée ne permettant pas de résoudre rapidement des problèmes complexes, pourrait bien disparaitre dans quelques décennies (années ?) si l'on parvient à mettre au point des ordinateurs quantiques à grande échelle.

Chapitre 17

La cryptanalyse

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. Il en existe 4 grands types, chacun pouvant utiliser différentes techniques. Ce chapitre présentera ensuite quelques attaques souvent évoquées dans la littérature spécialisée dans le domaine de la cryptologie.

17.1 Les 4 attaques cryptanalytiques

17.1.1 Attaque sur le texte chiffré uniquement (ciphertext-only)

A partir d'un texte chiffré, on recherche le texte clair et/ou la clé. On procède par analyse de fréquence des lettres utilisées dans le texte chiffré. Cette technique ne fonctionne que pour la plupart des chiffrements classiques basiques, les seuls permettant l'utilisation de l'analyse de fréquence. On peut aussi également procéder par force brute pour briser de tels chiffrements.

17.1.2 Attaque à texte clair connu (known-plaintext attack)

Etant donné un texte chiffré et un fragment de texte clair associé, on recherche le texte clair restant et/ou la clé. On utilise la technique dite de la cryptanalyse linéaire (1993) présentée rapidement dans la section suivante.

17.1.3 Attaque sur un texte clair sélectionné(chosen-plaintext attack)

Etant donné la capacité de chiffrer un fragment de texte clair choisi arbitrairement, on recherche la clé par la technique de la cryptanalyse différentielle.

17.1.4 Attaque sur le texte chiffré uniquement (chosen-ciphertext attack)

Etant donné la capacité de déchiffrer un fragment de texte chiffré choisi arbitrairement, on recherche la clé.

17.2 Quelques autres techniques

17.2.1 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 17.4, 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 (3DES) et 256 bits (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

FIG. 17.1 – Temps de calcul pour une taille de clé donnée

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

FIG. 17.2 – Sécurité fournie selon la taille de la clé

Remarques :

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.

17.2.2 Attaque 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).

17.2.3 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).

17.2.4 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.

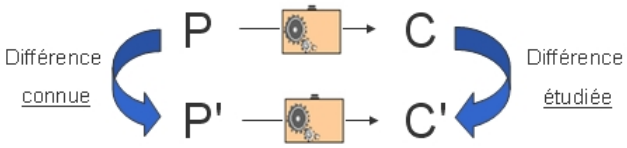

FIG. 17.3 – Cryptanalyse différentielle

17.2.5 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

$$M_{i1} \oplus M_{i2} \oplus \ldots \oplus M_{iu} \oplus C_{j1} \oplus C_{j2} \oplus \ldots \oplus C_{jv} = 0$$

avec M_i représentant le i^{eme} bit de $M = [M_1, M_2, \ldots M_u]$ et C_j représentant le j^{eme} bit de $C = [C_1, C_2, \ldots C_v]$. 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).

Remarques :

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.

17.2.6 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.

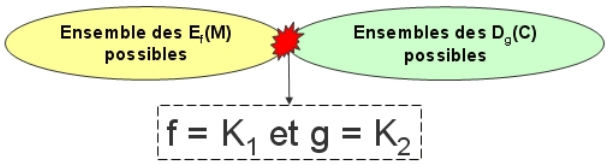

FIG. 17.4 – Meet-In-The-Middle

Un nombre de clés possibles réduit rend cette attaque utilisable.

17.2.7 Man-In-The-Middle

Souvent confondue avec l'attaque précédente en raison de son acronyme (MITM), elle est pourtant très différente dans les faits.

Son déroulement est illustré à la figure ci-dessous : le pirate se fait passer pour B auprès de A et pour A auprès de B.

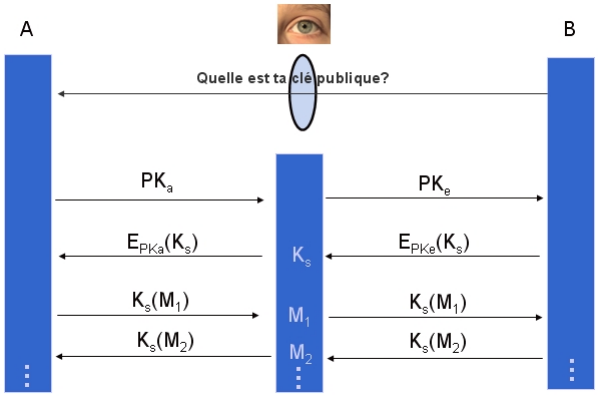

FIG. 17.5 – Man-In-The-Middle

17.2.8 Autres attaques logicielles

Quelques autres attaques logicielles répandues sont fréquemment citées :

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

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.

17.3 Attaquer les fonctions de hachage

17.3.1 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.

17.3.2 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) 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).

A priori, il y a deux solutions :
- 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).

17.3.2.1 Cas 1 : la fonction de réduction unique

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 fig.17.6). 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*.

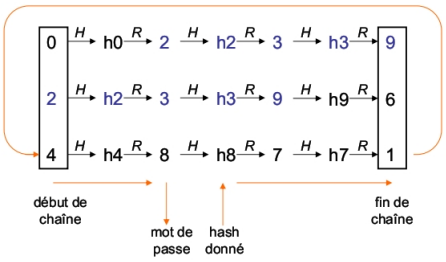

Fig. 17.6 – Compromis temps-mémoire par fonction de réduction unique
Source [4]

Pour un haché donné, l'algorithme de "cassage" est le suivant :

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 17.6).

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.

17.3.2.2 Cas 2 : Rainbow Attack

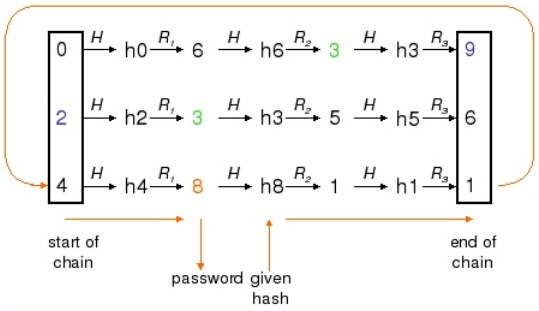

FIG. 17.7 – Compromis temps-mémoire par tables arc-en-ciel

Pour un haché donné, l'algorithme de "cassage" (figure 17.7) est le suivant :

1. On calcule sa réduction R3
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 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.

17.3.2.3 Contre-mesure

On peut se prémunir de ce type d'attaque en utilisant une nonce (que l'on appelle ici sel, ou *salt* en anglais). Ce sel est une donnée supplémentaire, non secrète. On applique alors l'algorithme de hachage à la concaténation du mot de passe avec ce sel :

$$h = H(\text{mdp}||\text{nonce})$$

Au niveau de l'attaque, cela a pour conséquence d'obliger l'attaquant à créer des tables arc-en-ciel spécifiques. En effet, pour une nonce de 16 bits, on obtiendra 65536 possibilités pour chaque haché.

En pratique, on utilisera, en plus de la nonce, des fonctions de hachage spécifiquement lentes, ce qui ralentira encore la création de ces tables.

Des précisions quant aux tailles des tables et à l'impact sur les coûts supplémentaires engendrés pour le pirate sont disponibles ici[1] et là[2].

17.4 Les 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 :

- 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.
- 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.

De nombreux types d'attaques appartiennent à cette branche de la cryptanalyse :

- **Fault Induction Attack** [5] : 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, ...
- **Optical Fault Induction Attack** [6] : 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 (cf. RSA)
- **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".

[1] http ://www.codinghorror.com/blog/archives/000949.html

[2] http ://chargen.matasano.com/chargen/2007/9/7/enough-with-the-rainbow-tables-what-you-need-to-know-about-s.html

– **Acoustic Analysis Attack** : écouter et analyser les sons produits par le processeur (ou cryptoprocesseur) durant ses calculs [7], ou le bruit des touches durant la frappe d'un mot de passe [8].

17.5 En guise de conclusion

Définissons encore 2 termes :

– Un (crypto)système est dit **inconditionnellement sûr** si le texte chiffré ne contient pas assez d'information pour retrouver le texte clair correspondant, peu importe la longueur du texte chiffré à disposition.

– Un système est dit **informatiquement sûr** :
  1. Si le coût nécessaire au décryptement excède la valeur intrinsèque de l'information
  2. Si le coût nécessaire au décryptement excède la durée de vie utile (durée de validité) de l'information

Dans la pratique, on trouve très peu de systèmes du premier type. Beaucoup plus du second.

Signalons enfin que la cryptanalyse et la cryptographie ont évolué conjointement. Comme dans de nombreux autres domaines, le jeu du chat et de la souris est bel et bien réel, même si la finalité de la cryptanalyse est souvent d'aider à la conception de meilleurs algorithmes.

Chapitre 18

Gestion des accès

18.1 Modèle de Lampson

En 2001, Butler Lampson a présenté un modèle [15] mettant en évidence les relations entre les différentes entités réelles d'un système d'accès aux données. Avant de voir un peu plus en détails les 3 méthodes d'accès principales existantes, nous allons donc présenter son modèle.

Lampson définit les entités de son modèle comme suit :
- Sujet = Entité pouvant effectuer des actions (humain, processus, machine,...)
- Objet = Entité considérée comme ressource nécessitant un contrôle d'accès (fichier, répertoire, port,...)
- Action = Opération effectuée par le sujet afin d'accéder à l'objet
- Gardien = Entité contrôlant l'accès proprement dit

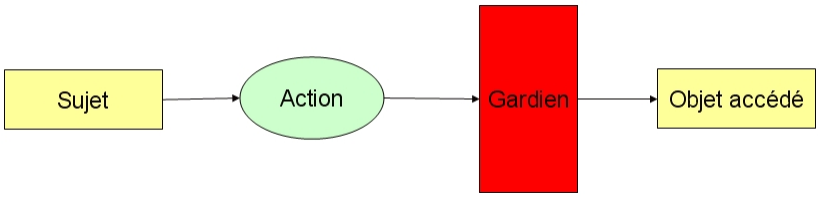

FIG. 18.1 – Le modèle de Lampson

Des exemples pratiques sont donnés dans le tableau ci-dessous.

utilisateur	requête	DBMS	Base de données
utilisateur	afficher une page web	Serveur Web	Page web
machine	envoi de paquet	firewall	intranet
programme	ouvrir un fichier	security manager Java	fichier

Le modèle de Lampson est particulièrement adapté à la présentation pratique du protocole AAA. En effet, par l'intermédiaire de son schéma, on représente facilement les endroits où ont lieu les différentes phase dudit protocole (voir figure 18.2).

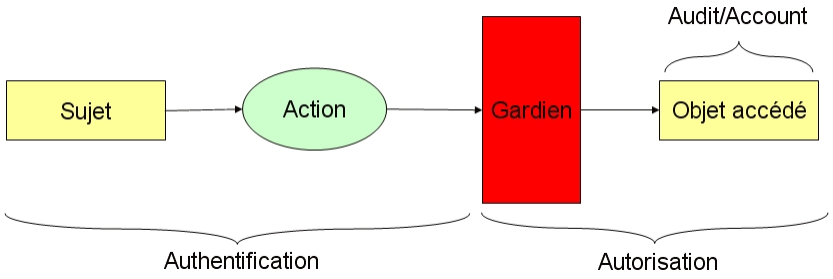

FIG. 18.2 – Le modèle de Lampson et le protocole AAA

18.2 Méthodes d'accès aux données

Trois techniques principales existent. De nombreuses méthodes en sont dérivées mais ne seront qu'en partie évoquées, toutes pouvant être rapprochées de l'une ou l'autre des 3 méthodes évoquées ci-dessous.

18.2.1 Discretionary Access Control (DAC)

L'accès est déterminé par le propriétaire de l'objet. Celui-ci détermine qui (*sujet*) peut utiliser l'objet, et comment (*action*) il peut l'utiliser.

On part du principe que tous les objets ont un propriétaire (qui sera souvent le sujet qui aura créé cet objet).

Quelques variantes de ce type d'accès permettent un transfert de propriété ou la délégation d'un droit entre utilisateurs non propriétaires.

Cette méthode d'accès est celle rencontrée la plupart du temps dans les systèmes d'exploitation courants (Linux, Windows). Le principe est d'associer une liste de contrôle d'accès à chaque fichier. On parle d'ACL (Access Control List).
Cette ACL renferme un certain nombres d'entrées (Access Control Entry - ACE). Le détail des accès permis ou refusé est mentionné.

Exemple d'ACL liée un objet T :

$ACE1$: L'utilisateur A peut le lire
$ACE2$: L'utilisateur B ne peut pas le modifier
$ACE3$: ...

18.2.2 Mandatory Access Control (MAC)

Il s'agit ici de contrôler l'accès en se concentrant sur les flux de données. L'accès est déterminé par le système.

- Chaque sujet possède un label lui donnant un niveau de confiance.
- Chaque objet possède un label permettant d'identifier le niveau de confiance requis pour l'utiliser.
- Le sujet doit avoir un label supérieur ou égal à celui de l'objet.

Cette méthode est beaucoup plus sûre, car elle ne dépend plus du propriétaire. Imaginez les pertes potentielles si le propriétaire d'un objet "sensible" venait à donner l'accès (volontairement ou non) à tous

les utilisateurs...

Un exemple concret est le modèle dérivé LBAC : Lattice-Based Access Control. Une lattice est une règle définissant un ensemble de niveaux de sécurité. Par exemple, la définition suivante, illustrée à la figure 18.3 :
- Les sujets avec un label L ne peuvent lire que des objets de label $L' <= L$ (**No Read-Up**)
- Les sujets avec un label L ne peuvent écrire que des objets de label $L' >= L$ (**No Write-Down**)

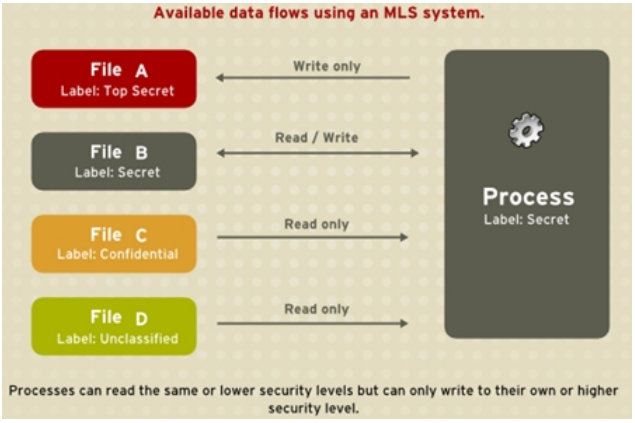

FIG. 18.3 – LBAC : no read-up et no write-down (Red Hat Inc., www.centos.org)

Comme on le voit, ce système est très contraignant. Peu adapté à un environnement commercial, il est peu répandu dans les contextes non-militaires. On le réserve à des lattices simples.

Bien que très sûr, il reste sensible aux attaques par canaux cachés (dialogue interprocessus à partir de l'utilisation du processeur par exemple).

Dans le contexte grand public, ce type de contrôle d'accès est rarement utilisé pour garantir la confidentialité en tant que telle. Ainsi, le système SELinux l'utilise pour la vérification de l'intégrité des données. Son modèle porte le nom de "Biba Model", qui utilise une règle "no write-up". Cette technique est notamment utilise pour empêcher la modification des données de l'OS à partir de logiciels indésirables circulant sur Internet.

18.2.3 Role-Based Access Control (RBAC)

L'accès est ici aussi déterminé par le système. Cette méthode d'accès est régulièrement utilisée en entreprise, ou chaque personne possède un rôle particulier qui lui donne accès à certaines informations. Le rôle va permettre d'attribuer un ensemble de permissions à un type d'utilisateurs.
- Plusieurs utilisateurs peuvent avoir le même rôle.
- Un utilisateur peut avoir plusieurs rôles, qu'il pourra activer au besoin.

Cette méthode est plus facile à mettre en oeuvre que le DAC pour la simple raison qu'il suffit de changer de rôle lorsque cela s'avère nécessaire (au lieu de modifier les accès pour chaque fichier).

Il existe plusieurs variantes à cette méthode :
- Utilisation de rôles "hiérarchiques" : un rôle "hiérarchiquement supérieur" possède toutes les permissions des rôles "inférieurs".
- Contrôle des activations simultanées des rôles : plusieurs rôles peuvent selon les cas être autorisés simultanément pour un même utilisateur.

Ce type d'accès aux données apparait très proche de ce que nous connaissons sous Windows notamment, par l'intermédiaire de la gestion des comptes utilisateurs. Cette filiation n'est pas anodine puisqu'il est en effet possible de simuler un système RBAC en utilisant la notion de "groupes utilisateurs". En réalité, il s'agit d'un DAC caché.

Windows utilise des ACL spécifiant les accès par groupe.
- $ACE1$: Le groupe A ne peut pas lire le fichier,
- $ACE2$: Le groupe B peut modifier le fichier,
- ...

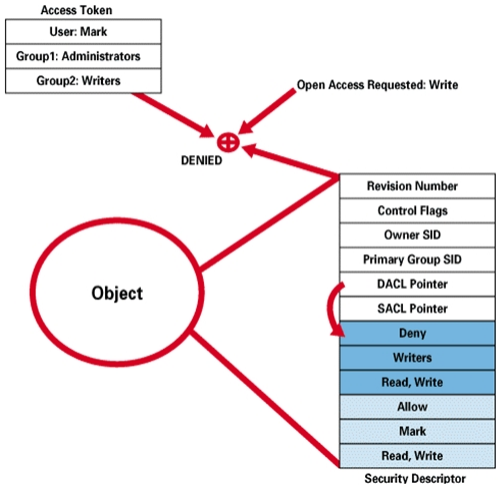

FIG. 18.4 – RBAC-DAC sous Windows (www.windowssecurity.com)

Remarques au sujet de la figure 18.4 :

- A l'ouverture d'une session, l'utilisateur reçoit un jeton d'accès (Access Token). Ce jeton définira les actions qu'il peut effectuer.
- SID : Security Identifier (unique)
- SACL : System ACL utilisé pour l'Audit
- L'ordre établi dans la DACL est important : la première entrée correspondante est prise en compte.
- Comme on le voit, la notion de groupe est présente, mais ce sont bel et bien les droits définis par le propriétaire qui détermine les actions possibles pour les autres utilisateurs (cf. Owner SID).

18.3 La problématique des mots de passe

Sans prêter attention au type de contrôle d'accès mis en place, s'il y a bien un domaine où la sécurité peut faire défaut, c'est dans la gestion des mots de passe utilisateur. Mais que peut-on dire à ce sujet ?

18.3.1 Quelques principes évidents, en théorie...

Lors de la création du mot de passe, une attention particulière doit être portée sur certains points :

- Si un générateur de mots de passe est utilisé, il devra employer une grande variété de caractères (pour le rendre plus robuste à la force brute).

- Il peut être utile d'utiliser un vérificateur de mots de passe afin de tester la vulnérabilité aux attaques par dictionnaire. Dans le même temps, il pourra tester la taille des mdp face aux attaques par force brute.
- Il est bon d'associer une durée de vie aux mdp. Un changement régulier permet une meilleure protection contre la force brute.
- On peut aussi limiter le nombre d'essais.

Quelques exemples de "mauvais" mots de passe :

- Par défaut : password, mdp, default, admin, ...
- Mots : bonjour, test, voiture, silence, ...
- Mots numérotés : clavier12, merci154, armoire98, ...
- Egaux au login : Albert84, ...
- Mots doublés : crabcrab, stopstop, treetree,...
- Séquences : qwerty, 12345678, bhunji,...
- Personnels : prénom, numéro d'imatriculation,...

Les besoins en sécurité étant ce qu'ils sont, la difficulté majeure aujourd'hui est que le nombre de mots de passe ne cesse de croître. Leur mémorisation reste donc problématique. Quelques solutions existent cependant :

1. Hardware : utiliser des clés usb comme accès aux données. Le problème est que c'est alors la clé qui authentifie, et non l'individu. De plus, que faire en cas de perte de la clé ?

2. Software :
   - utiliser un logiciel de gestion de mots de passe : un seul mot de passe (ou une phrase de passe) pour stocker tous les autres. Mais que faire en cas d'oubli du mot de passe maitre ?
   - utiliser la saisie semi-automatique. Mais quid en cas de formatage, virus, etc. ?

3. Au niveau de l'OS : une solution avancée est connue sous le nom de SSO (Single Sign On). Le fait de se logguer sur une machine permet d'accéder à toutes les données. Une seule phase d'authentification a donc lieu, et si elle réussi, l'utilisateur est libre d'agir avec les données et logiciels correspondant à ses droits, sans avoir à donner son mot de passe à chaque accès. Une application connue basée sur un principe similaire porte le nom de Kerberos (authentification d'utilisateurs sur les machines d'un réseau)
   - Avantages :
     - Les mots de passe de ces logiciels ne doivent plus être retenus (gain de mémoire)
     - Les mots de passe de ces logiciels ne doivent plus être tapés (gain de temps)
     - Un seul mot de passe à retenir
     - Gestion administrateur en cas de perte
   - Désavantages :
     - Un seul mot de passe à retenir. A nouveau, que faire en cas de divulgation ou d'oubli ?
     - Nécessite une authentification forte "à l'entrée" pour s'assurer de l'identité de l'utilisateur loggué

Une autre possibilité, mais qui n'est pas utilisable dans toutes les conditions, porte le nom de One Time Password (OTP).

18.3.2 One Time Password

Le mot de passe ici généré ne reste valable que pour une durée déterminée. Deux méthodes coexistent :

1. Méthode synchrone : Fonction du temps (Time-synchronous)
2. Méthode asynchrone : En réponse à un "challenge" (Challenge-response)

18.3.2.1 Méthode synchrone

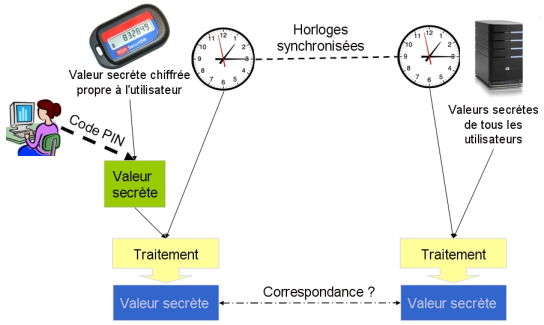

FIG. 18.5 – OTP par méthode synchrone

18.3.2.2 Méthode asynchrone

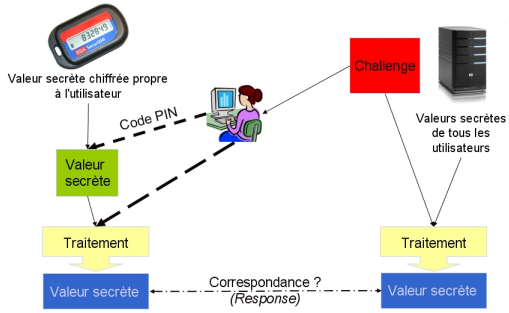

FIG. 18.6 – OTP par méthode asynchrone

18.3.3 Conclusions

Un bon mot de passe doit notamment (et principalement) être à la fois facile à retenir (pas besoin de le noter), et utiliser des caractères spéciaux, de casse différente (ce qui rend rend la force brute plus fastidieuse et l'attaque par dictionnaire presque impossible).

Idéalement, un mot de passe devra être :
- Long,
- Unique,
- Complexe,
- Modifié régulièrement,
- Mémorisé

Chapitre 19

Sécurité logicielle

Avec Internet, de nombreux problèmes surgissent concernant la sécurité des utilisateurs, mais également des données de ces utilisateurs. Nombreux sont les internautes n'ayant aucune notion de sécurité, et qui, même s'ils connaissent les termes "Virus" ou "Trojan", ne savent comment s'en protéger.

Pour tout problème ayant trait à la sécurité, il faut d'abord comprendre comment fonctionnent ces intrus, avant de pouvoir apporter une solution.

19.1 Introduction

Alors qu'Internet se démocratise et se popularise, de nombreux logiciels sont mis à la disposition des utilisateurs pour réaliser ce type d'attaques. Les créateurs de ces logiciels ont souvent des raisons bien précises. Parmi les plus courantes, on peut citer :

- Par défi personnel : nombreux sont les crackers n'agissant que par plaisir, afin de "prouver au monde" leurs capacités
- Pour des raisons politiques
- Pour des raisons dogmatiques
- Pour voler de l'information (espionnage industriel)
- Pour modifier des informations
- Par vengeance

www.vulnerabilite.com	Adolescent (script-kiddy)	Etudiant (script-kiddy)	Employé	Organisations (cybercriminalité, spammeurs)	Gouvernement
Temps	Limité	Moyen	Moyen	Elevé	Elevé
Budget (€)	Faible	Variable	Variable	Elevé	???
Créativité	Variable	Forte	Variable	Variable	Variable
Détectabilité	Forte	Forte	Faible/Moyen	Faible	Faible
Objectif	Challenge	Publicité	Argent	Argent	Divers
Représentativité	Forte	Moyenne	Faible/Moyen	Faible	???
Organisé	Non	Non	Non	Oui	Oui

FIG. 19.1 – Type de "pirate" et caractéristiques

Il est toutefois primordial de préciser la tendance actuelle. On assiste à une véritable révolution dans l'identité et l'objectif des hackers. Ainsi, aujourd'hui, d'après une étude de la société Kaspersky Labs, les hackers issus des pays tels que le Brésil, la Chine ou la Russie agiraient plutôt comme de véritables criminels, dans le but d'extorquer de l'argent ou des renseignements confidentiels. Au contraire, en Europe, on assisterait plutôt à une simple étude des techniques de hacking, destinée à l'approfondissement

global des connaissances en sécurité.

Autre différence entre ces deux régions : la taille même des groupes. En Europe, les hackers sembleraient isolés, ou agissant par petits groupes. En Chine, il s'agit de véritables organisations criminelles de plusieurs dizaines voire centaines de personnes.

Dans le même temps, la difficulté pour mettre en place ces attaques décroit, comme le montre une étude réalisée en 2002 présentée par la figure 19.2.

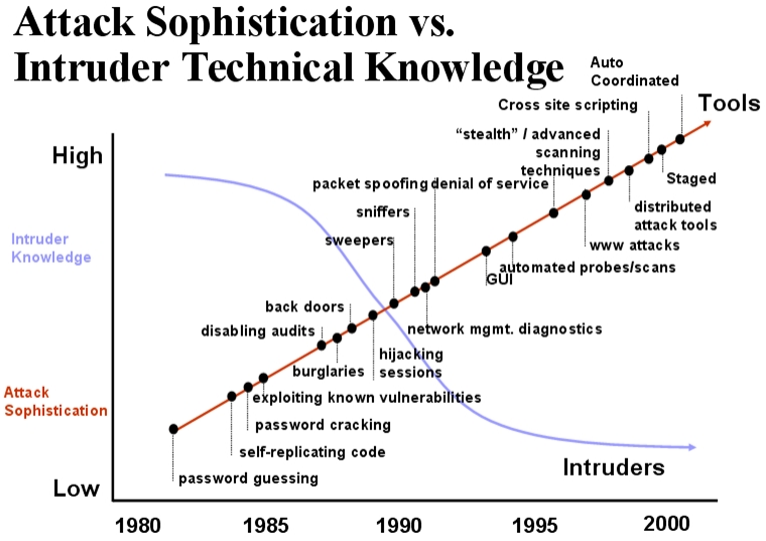

Fig. 19.2 – Cyberterrorism, Tim Shimeall, Carnegie Mellon University, 2002

Une autre représentation, plus axée sur l'impact de ces attaques est donnée à la figure 19.3. Remarquons ici que la notion d'impact "global" représente la taille des réseaux visés. Ainsi, on attaque plus aujourd'hui des réseaux bancaires de grandes envergures plutôt que le pc du voisin. On constate dans le même temps que les attaques lancées au "hasard" (à destination de tout Internet, tel qu'un ver laché à travers Internet par exemple) sont de moins en moins fréquentes.

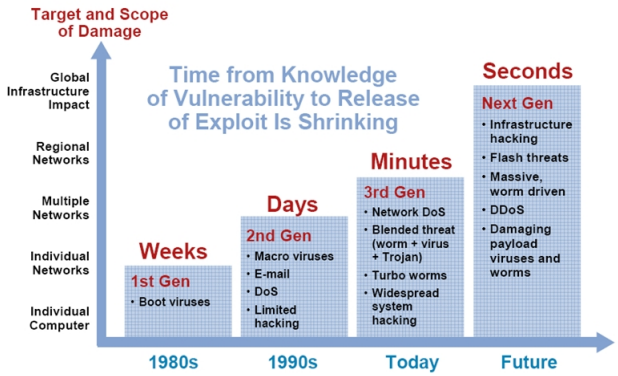

Fig. 19.3 – Cisco Expo 2007 Presentation, Cisco Systems Inc.

Les systèmes ciblés peuvent prendre n'importe quelle forme, de l'université aux infrastructures militaires, en passant par les FAI ou les banques.

Ces attaques (par virus ou par intrusion) provoquent souvent de nombreux problèmes, correspondant à des pertes de données, de fonctionnalités, d'image de marque, ou plus simplement une perte de temps, lorsqu'il s'agit d'un virus inoffensif.

19.2 Virus, Vers et dérivés

Leurs créateurs agissent souvent par défi personnel, afin de prouver à la communauté qu'il ont acquis une quantité suffisante de connaissance pour voler, modifier, ou détruire certaines données.

19.2.1 Les virus

Les virus se définissent par une portion de code, non nécessairement destructrice, capable de se reproduire sur l'ordinateur cible et/ou de se propager sur d'autres supports (ordinateur, disquettes, ...). Cette reproduction est d'ailleurs le seul véritable objectif d'un virus, l'aspect destructif étant propre aux bombes logiques, explicitées ci-après. Ils portent également le nom de CPA, pour Code Parasite Autopropageable.

Ces virus peuvent se propager de nombreuses manières, telles que par l'intermédiaire d'une disquette, d'une pièce jointe à un mail, ou encore d'un fichier téléchargé sur Internet.

Historiquement, les virus sont nés d'un jeu entre concepteurs de l'entreprise Bell AT&T. Dans les années 70, certains programmeurs avaient coutume de s'affronter à la "Core War". Le principe était que chaque programmeur devait créer un mini-logiciel, afin que ceux-ci s'affrontent dans un espace mémoire déterminé. A la suite de ce jeu, plusieurs articles sont parus expliquant les principes et les objectifs du jeu. Certains programmeurs en herbe se sont alors mis en tête de réaliser le même type de logiciel, mais à des fins moins ludiques.

19.2.1.1 Analogie entre virus informatique et virus biologique

Tout comme le virus biologique, le virus informatique poursuit plusieurs objectifs :

- Il essaye de se dissimuler le plus longtemps possible aux yeux de l'utilisateur infecté
- Il contamine tout ce qui est à sa portée
- Il tente de se répandre, sans se cantonner au support sur lequel il se trouve

19.2.1.2 Classification des virus

Il n'existe pas de classification stricte. En voici quelques exemples :
- Par le format visé (exécutable ou documents),
- Par leur comportement (rapide, lent, résident, polymorphe,...),
- Selon l'organe visé (boot sector, driver, ...),
- Selon le langage utilisé (virus assembleur, macro-virus, virus interprété, ...),
- ...

Quelques infos sur certains des noms donnés ci-dessus :
- **Les macro-virus** : Ces virus utilisent le langage de Microsoft Visual Basic. Ce langage permet d'écrire des macros pour les logiciels de la suite bureautique Office (Word, Excel,...). Un virus de ce type pourra donc se propager lorsque l'utilisateur ouvrira le fichier infecté. De plus en plus de logiciels sont compatibles avec le VB (VBScript). Le risque de propagation est donc d'autant plus important.
- **Les virus résidents** : Ils se placent en mémoire RAM et contaminent les fichiers au fur et à mesure de leurs exécutions. Ils peuvent par exemple prendre la forme de fichier pilote de Windows (.vxd). Ils sont alors chargés dès le démarrage du système, avant le chargement de l'antivirus.

- **Les virus de boot** : On parle aussi de *virus de secteurs d'amorce*. Leur objectif est de modifier le secteur de démarrage du disque et de le remplacer par une copie d'eux-même de sorte que le virus soit déjà en cours d'exécution dès le démarrage du système d'exploitation et de tout système d'éradication.
- **Les virus lents** : A la différence d'un virus rapide qui infecte les fichiers dès qu'ils sont manipulés par le système, les virus lents n'infectent les fichiers qu'en cas de modification, ce qui rend leur détection plus subtile.
- ... (voir aussi la section consacrée aux anti-virus)

Trois principales techniques (modes d'action) sont utilisées par les virus :
- Contamination par *recouvrement* : le virus écrase les premières instructions du fichier par ses propres instructions. L'avantage est que la taille du fichier n'est pas modifiée. Cependant, il n'est plus utilisable par la suite, le début de ses instructions ayant été supprimé.
- Contamination par *ajout* : le virus s'exécute avant le code original du fichier infecté, mais repasse la main à ce dernier à la suite de son exécution. Dans ce cas, la taille du fichier est modifiée.
- Contamination par *entrelacement* : il s'agit ici d'insérer du code entre les blocs valides du programme. Elle est plus difficile à mettre en place, mais est moins facilement détectée.

19.2.2 Les vers

Ils sont souvent confondus avec les virus mail. La différence à opérer entre les deux termes est que le second implique une intervention humaine dans son déploiement (exécution d'une pièce jointe par exemple).

Les vers sont moins répandus que les virus mail, de par leur complexité de mise en place (analyse des failles du système sur lequel il se situe, copie du ver sur la machine cible, ...). Un ver se propage entre les hôtes d'un réseau (de n'importe quel type) en exploitant les failles de sécurité de ces hôtes. On trouvera parfois des vers s'accompagnant d'une bombe logique.

Le premier ver recensé en tant que tel fut créer par Robert Morris, fils du directeur de la NSA, en 1988. Nommé "Internet Worm", il posa énormément de problèmes sur Internet (qui n'en était qu'à ses débuts). Bien qu'il ne fasse aucun dégât, tous les serveurs furent mis en quarantaine pour vérification. Tout le réseau fut paralysé en moins de 24 heures. A la suite de cette attaque (si on peut véritablement la considérer comme une attaque), apparut le CERT, groupe gérant les failles de sécurité d'Internet.

19.2.2.1 Utilité des vers

La plupart du temps, les vers n'ont d'autres utilités que la destruction (s'ils s'accompagnent d'une bombe logique), et la congestion du réseau. Malgré tout, le ver a parfois d'autres utilisations :

- Le ver créé par Morris n'avait d'autre but que d'explorer Internet. Il est vrai que ce réseau était relativement restreint à l'époque. Aujourd'hui, cette application ne semble plus réalisable.
- Certaines entreprises utilisent également des vers pour tester la sécurité de leur réseau intranet. Le ver tentant de s'infiltrer sur les machines distantes par l'intermédiaire des failles du réseau, s'il y parvient, cela signifiera que le réseau de l'entreprise n'est pas sécurisé.

19.2.2.2 Architecture d'un ver

Il est composé de deux parties :
- **Le vecteur** : c'est la partie principale du ver, qui s'occupe de la recherche de failles et est responsable de transmettre la seconde partie du ver.
- **L'archive** : c'est la partie "morte" du ver, qui sera envoyée sur le système distant et l'infectera. Cette archive peut exister sous deux formes distinctes :

- sous forme de sources : la reproduction est beaucoup plus aisée, car le nombre de machines potentiellement ciblées est beaucoup plus grand. L'inconvénient majeur de ce procédé est qu'un compilateur est nécessaire sur la machine à infecter. Les données nécessaires à l'infection sont plus volumineux et une compression des sources sera parfois opérée préalablement.
- sous forme binaire : la reproduction de ce type de ver n'est possible que sur une architecture compatible avec les données binaires. Cependant, ce format ne nécessite pas de compilateur sur la machine hôte et est souvent plus compacte que la version source.

Il est à remarquer qu'il existe des vers multi-formes, c'est-à-dire utilisant une archive en sources ou binaire selon le système à infecter.

19.2.2.3 Contamination d'un système

Avant d'infecter un système, le ver doit procéder dans l'ordre à une série d'étapes. Ce n'est qu'à la suite de celles-ci qu'il pourra attaquer la machine cible. La succession d'étapes est la suivante (et ne concerne que la partie "vecteur" jusqu'à la phase d'invasion) :

1. **Initialisation** : ce sont les premières instructions du ver. Il peut s'agir de la création de fichiers temporaires, ou la compilation d'une partie de l'archive. A cet instant, le ver est toujours sur la machine mère.
2. **Recherche de l'hôte** : Cette étape procède à un scan d'IP. Ce scan peut être aléatoire ou incrémental (on passe en revue des séries d'adresses IP). Pour chacune d'entre elles, on teste si le système répond ou non ("up" ou "down").
3. **Identification de l'hôte** : une fois une victime potentielle détectée, le ver (la partie vecteur) va tester le système en place sur la machine distante. Il s'agira de vérifier si le système possède des failles pouvant être exploitées par le ver, si il possède un compilateur approprié, ... Si ce n'est pas le cas, on retourne à l'étape précédente.
4. **Attaque** : c'est ici que le ver exploite les vulnérabilités mises au jour dans la phase précédente. Le vecteur obtient alors un accès sur la machine cible.
5. **Invasion** : Le vecteur est maintenant présent sur la machine cible, mais n'est pas encore actif. Les instructions sont toujours données par la machine mère. Dès cet instant, le vecteur va rappatrier la partie archive sur le système à infecter. Deux techniques sont possibles :

   - Soit le vecteur est toujours accompagné de sa partie archive, auquel cas l'archive est présente sur la machine à tout moment, le vecteur uploadant son archive de machines en machines. Cette technique est un peu plus complexe à réaliser car le vecteur doit avoir été programmé pour transmettre cette archive.
   - Soit le vecteur doit rappatrier la partie archive. Ce rappatriement se fait à partir de la machine mère, ou depuis un serveur fixe (de type FTP) et unique pour toutes les machines (mais dans ce cas, l'attaque se terminera si le serveur tombe en panne). C'est beaucoup plus facile à réaliser, mais également plus dangereux car l'utilisateur piraté pourra plus facilement tracer le ver.

   Si l'archive est sous forme de sources, le vecteur devra également procéder à la décompression et à la compilation de ces sources avant de passer à la phase suivante.
6. **Reproduction** : ici aussi, deux possibilités existent pour permettre la reproduction du ver :

   - Soit on tue le ver présent sur la machine mère. Le ver se déplacera alors de stations en stations.
   - Soit on ne le tue pas et chacun des deux vers continue à se propager selon la méthode décrite ci-dessus. Ce choix provoque une étendue exponentielle du ver et est logiquement plus dangereuse que la précédente.

19.2.3 Les chevaux de Troie

Ce type de programme est plus subtil, car ils se dissimulent dans un fichier anodin, et ne semblent même pas exister aux yeux de l'utilisateur.

Les chevaux de Troie (ou Trojan) sont les attaques par logiciel les plus fréquentes aujourd'hui, leur propagation étant relativement aisée, de par la naïveté de nombreux utilisateurs d'Internet.

Un trojan n'est à priori pas destructeur comme peut l'être un virus. Cependant, ils sont parfois associés à une bombe logique, ce qui les rend dangereux. Les effets peuvent alors être très variés, de la destruction de fichiers à une congestion mémoire, en passant par l'instauration d'une attaque DDoS.

Le cheval de Troie possède souvent un objectif tout autre, à savoir la création d'une backdoor (ou "porte dérobée"). Cette backdoor sera alors le point d'entrée d'un pirate qui pourra à sa guise recueillir des informations diverses propres à l'utilisateur (mot de passe, accès à la base de registre, fonctions keylogger, ...) ou interagir directement avec l'ordinateur (arrêt/redémarrage de la machine, déconnexion d'un réseau, blocage de périphériques, impression de documents, ...).

Enfin, à la différence d'un virus ou d'un ver, le trojan ne se reproduit pas. Il est ainsi plus difficile de le détecter car il n'a pas de signature semblable à celle d'un virus. Les trojans sont souvent furtifs, et utilisent des noms de programmes communs[1] pour éviter leur détection.

19.2.3.1 Mise en place d'un cheval de Troie

Comme dans le cas d'un ver, le cheval de Troie doit passer par plusieurs étapes pour être utilisable par le pirate :

1. **Intrusion sur l'ordinateur cible** : les trojans se répandent souvent par pièce jointe (par exemple : écran de veille).
2. **Modification de l'OS** : une fois introduit, il va modifier les fichiers systèmes nécessaires à son fonctionnement. Les fichiers modifiés sont principalement ceux régissant la phase de démarrage de l'OS. Le trojan sera alors chargé lors du prochain démarrage.
3. **Phase d'écoute** : le trojan en tant que tel n'est pas dangereux. Il ne fait qu'écouter les instructions du pirate. Ainsi, une fois le système redémarré, le trojan passe à une phase d'écoute, dans laquelle il attend les commandes du pirate distant.
4. **Intéraction avec le pirate** : le trojan sert de "porte d'accès" au pirate. Toutes ses commandes passeront par son intermédiaire.

19.2.3.2 Détection par les antivirus

Comme cité précédemment, un trojan n'a pas pour vocation de se reproduire. L'antivirus ne pourra donc pas scanner sa signature.

Cependant, les trojans sont souvent réutilisés par les pirates. La plupart d'entre eux sont donc connus des logiciels antivirus, et ces derniers peuvent relever des traces (ports utilisés, exécutables, ...) pour les identifier. La table 19.1 donne quelques ports utilisés par certains trojans.

Le problème se pose avec les trojans dont le code source est diffusé sur Internet. De ce fait, il existe de nombreuses versions d'un même trojan, donc les traces sont différentes entre chaque variante. L'antivirus devra donc relever non plus des caractéristiques connues, mais des indices afin de détecter leur activité. Ainsi, si l'antivirus détecte des ports ouverts suspects, tels que des numéros de ports aléatoires, il pourrait en conclure à un trojan.

Certaines variantes de trojans ont contourné la difficulté en utilisant des ports communs (port ftp, irc, telnet, ...), et si l'antivirus ne réagit pas, il faudra que l'utilisateur y prenne garde.

Dans certains cas, le pirate utilisera le trojan comme une passerelle pour attaquer un réseau entier. Dans ce cas, il faudra également procéder à une analyse de trafic, pour par exemple détecter un scan de ports.

[1] winamp5.exe, zip.exe, etc.

...	...
8685	Unin68
8732	Kryptonic Ghost Command Pro
8734	AutoSpY
8787	Back Orifice 2000
8811	Fear
8812	FraggleRock Lite
8821	Alicia
8848	Whirlpool
8864	Whirlpool
8888	Dark IRC
...	...

TAB. 19.1 – Liste de quelques ports utilisés

19.2.4 Les bombes logiques

Ces bombes n'ont pas, à priori, la possibilité de se reproduire. C'est pourquoi elles sont souvent associées à des vers ou des chevaux de Troie. Alors qu'initialement un ver et un trojan ne font pas de dégat (le ver se reproduit de machine en machine, et le trojan ouvre un accès), il peut arriver que certains d'entre eux provoquent de gros dégâts, semblables à ceux d'un virus. Une autre différence entre virus et bombe est que la bombe est toujours destructrice (d'où son nom).

La bombe logique est une portion de code qui, comme son nom l'indique, explosera à un moment voulu (après un temps déterminé suivant l'installation d'un logiciel, à une date précise, après une action de l'utilisateur, ...). Dans le cas d'un cheval de Troie associé à un écran de veille, la bombe logique pourra exploser après quelques heures de veille, ainsi, il y aura de fortes chances que l'utilisateur ne soit plus présent devant son écran. L'idéal est que la bombe n'explose pas directement après l'installation du logiciel, et ce pour éviter qu'elle ne soit trop rapidement détectée.

Certaines bombes logiques peuvent aussi exploser à un moment précis déterminé. Ainsi, la bombe logique "Tchernobyl" était de ce type, et, après s'être répandue sur de nombreuses machines, a explosé le 26 avril 1999, date du treizième anniversaire de la catastrophe du même nom. Une telle bombe porte le nom de *Bombe temporelle* ou *Bombe à retardement*.

19.2.5 Les intrusions

Avec l'extension des parcs informatiques, il est de plus en plus nécessaire d'augmenter le niveau de sécurité qui leur est attribué. A coté des systèmes d'échanges cryptographiques et de contrôle d'accès, il peut être utile d'analyser les flux internes et/ou externes du réseau.

A la différence des virus, vers et trojans, les attaques basées sur les failles de système (ou du réseau) sont souvent plus complexes à mettre en place. Elles serviront la plupart du temps à mettre en place une attaque ultérieure.
Il existe des dizaines voire des centaines d'intrusions si on tient compte des modifications possibles. Les conséquences peuvent être très variées, allant d'un simple redémarrage à une totale congestion du réseau en passant par un plantage système.

On citera par exemple les attaques DoS, DDoS, PortScan, DNS Poisoning, Ping Flooding, ou encore l'OS FingerPrinting[2].

[2]Voir les annexes pour quelques informations complémentaires sur ces attaques.

19.3 Les systèmes de protection

19.3.1 Les antivirus

Il est intéressant de noter qu'une fois un fichier infecté, il ne l'est jamais deux fois. En effet, un virus est programmé de telle sorte qu'il *signe* le fichier dès qu'il est contaminé. On parle ainsi de *signature de virus*. Cette signature consiste en une suite de bits apposée au fichier. Cette suite, une fois décelée, permettra de reconnaitre le virus.

Lorsque le virus est détecté par l'antivirus, plusieurs possibilités sont offertes pour l'éradiquer.
- Supprimer le fichier infecté
- Supprimer le code malicieux du fichier infecté
- Placer le ou les fichiers infectés en "quarantaine" pour un traitement futur.

Malgré tout, il existe plusieurs catégories de virus compliquant leur propre détection :

- **Les virus polymorphes** : Puisque les antivirus se basent sur la recherche de signature, les créateurs de virus ont élaboré une méthode de camouflage. Celle-ci réside dans le fait de chiffrer la signature inscrite dans le fichier infecté, de manière à rendre leur détection et leur identification plus complexe pour les antivirus. Ainsi, pour chaque fichier infecté, la signature aura une apparence différente. L'antivirus doit dans ce cas se baser sur la méthode de chiffrement utilisée et détecter les correspondances dans cette méthode de chiffrement pour pouvoir identifier un fichier infecté par tel ou tel virus.
- **Les virus défensifs ou rétrovirus** : Ces virus ont la particularité de pouvoir modifier les signatures de certains antivirus afin de rendre les détections impossibles par leur intermédiaire.
- **Les virus mutants** : La majorité des virus ne sont en réalité que des réécritures de virus originaux. Ces réécritures, qui portent le nom de *variantes*, restent pourtant uniques pour l'antivirus, car ce dernier doit toujours détecter la signature, qui elle, est différente pour chaque variante.
- **Les virus furtifs** : Ils peuvent restaurer la taille initiale des fichiers après leur infection.

La détection des virus

La guerre entre virus et antivirus est bien réelle. Dès qu'un clan agit, le camp opposé tente de trouver la parade. Pour détecter les virus, les antivirus doivent user de plusieurs techniques :

- **Le scanning des signatures** : La détection des virus consiste en la recherche de ces signatures à partir d'une base de données de signatures (on parle également de *définitions de virus*). Le principal avantage de cette technique est qu'il est possible de détecter le virus avant qu'il ne soit en action. Cependant, il est nécessaire que sa signature soit présente dans la base de données afin qu'il soit détecté. De plus, il est nécessaire de tenir la base régulièrement à jour afin de pouvoir détecter les nouveaux virus.

- **Le contrôleur d'intégrité** : Le principe est que l'antivirus maintienne une liste des fichiers exécutables associés à leur taille, leur date de création, de modification, voire un CRC. L'utilisation du CRC permet de vérifier qu'un exécutable n'a pas été modifié en comparant sa somme de contrôle avant et après son exécution. En effet, en dehors d'une mise à jour explicite du fichier, un fichier exécutable n'est pas sensé être modifié. Le même type de vérifications peut être instauré avec la date et l'heure de modification. Cependant, il suffira aux virus de mémoriser ces valeurs afin de pouvoir les restaurer par la suite.

- **Le moniteur de comportement** : Il s'agit ici de contrôler en continu toute activité suspecte telles que les lectures et écritures dans des fichiers exécutables, les tentatives d'écriture dans les secteurs de partitions et de boot du disque.

– **L'analyse heuristique** : A la différence du moniteur de comportement qui détecte les modifications causées par les virus, l'analyse heuristique tente de détecter les virus avant leur exécution, en cherchant des portions de code suspectes. Il pourrait par exemple chercher des séquences de lecture suivies de séquences d'écriture sur un même fichier exécutable. Cette technique permet donc de détecter des virus même s'il ne sont pas présent dans la base de données, puisque l'analyseur teste des séquences d'instructions communes à de nombreux virus.

19.3.2 Les systèmes de détection (et de prévention) d'intrusions

Ces logiciels, souvent appelés IDS[3], sont utilisés pour écouter, et analyser le trafic d'un réseau. Il en existe plusieurs dizaines, aussi bien dans le secteur commercial qu'en tant que projet libre (SNORT et PRELUDE étant les plus connus dans ce domaine).

Il existe principalement deux techniques d'analyse du trafic, chacune ayant des avantages et des inconvénients :

– **L'analyse comportementale** (*anomaly intrusion detection*) : à partir d'un comportement normal déterminé, l'IDS analyse le comportement des machines. Si un ordinateur se connecte en pleine nuit alors que personne n'est présent, cela pourrait lever une alerte pour l'IDS. Ainsi, dans ce type d'analyse, un profil est dressé et lorsque la machine liée s'éloigne du profil type, l'IDS réagit.
  – Avantages : ce type d'analyse permet de détecter des attaques inconnues, elle ne nécessite pas de base de données.
  – Inconvénient : cette détection est assez aléatoire, elle peut produire de fausses alertes relativement facilement.

– **L'analyse par scénario** (*misuse intrusion detection*) : l'IDS utilise ici une base de données de *signatures* d'attaques. Ces signatures peuvent être assimilées à des déroulements d'attaques. En effet, chaque attaque possède des caractéristiques propres (numéro de port, taille de paquet, protocole employé, ...). Ces caractéristiques peuvent être collectées et placées dans une base de données qu'interrogera l'IDS. Ce type d'IDS utilisent les fichiers journaux (log). Dès qu'il détectera des séquences suspectes (relatives à une signature de sa base de données), il déclenchera une alerte.
  – Avantage : on peut gérer les attaques de façon très précise.
  – Inconvénient : on doit maintenir une base de données à jour.

Certains IDS analysent uniquement les fichiers systèmes (fichiers d'historique), et d'autres uniquement le trafic réseau. On les nomme respectivement HIDS pour "Host IDS" (protection des machines) et NIDS pour "Network IDS" (protection du réseau).

Les IDS ne réagissent pas non plus de la même manière en présence d'une attaque. La plupart agissent *passivement*, c'est-à-dire qu'une fois l'attaque détectée, ils émettent simplement une alerte. D'autres, beaucoup moins répandus, tentent de contre-attaquer. On dit que ces derniers sont *actifs*.

Depuis quelques temps, les IDS actifs évoluent et se font appelés IDP ou IPS (respectivement *Intrusion Detection and Prevention* et *Intrusion Prevention System*. Deux grandes différences existent entre les IDS et les IPS. La première, comme il l'a déjà été mentionné, repose sur les capacités de l'IPS à répondre aux attaques. De par sa position sur le réseau (figure 19.4), il peut stopper l'intrusion dès qu'il la détecte. La seconde grande différence est la gestion des débits importants actuels. Là où les IDS éprouvaient des difficultés, les IPS, souvent associés à des composants hardware, peuvent gérer des flux beaucoup plus importants.

[3]Intrusion Detection System

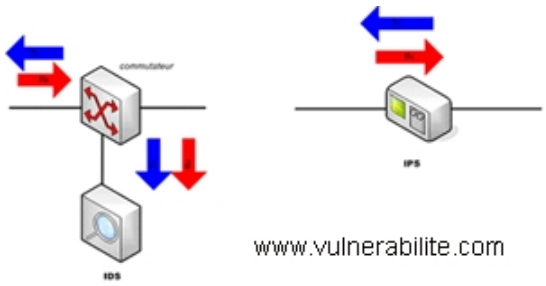

FIG. 19.4 – Positionnement d'un IDS et d'un IPS sur un réseau

19.3.2.1 Détail d'une signature (de SNORT)

```
ALERT tcp $REZO_EXTERNE any -> $REZO_INTERNE 21
(msg :"FTP EXPLOIT overflow"; flags : A+; content :"|5057 440A
2F69|"; classtype :attempted-admin; sid :340; rev :1;)
```

Sans entrer dans le détail des possibilités de détection de SNORT, cette signature signifie que l'IDS devra déclencher une alerte lorsqu'il détectera, sur le protocole TCP, une tentative de connexion d'une IP externe (à spécifier dans $REZO\_EXTERNE$) provenant de n'importe quel port vers une IP interne (à spécifier dans $REZO\_INTERNE$) sur le port 21 (FTP). L'alerte ne devra toutefois être déclenchée que si l'IDS repère les informations suivantes, spécifiées dans la suite de la signature :

- flags : demande à l'IDS de vérifier si le drapeau ACK est levé (A). SNORT peut tester tous les drapeaux TCP.
- content :"|*****|" : vérifie que la chaîne indiquée est présente dans le paquet. En effet, la plupart des attaques possèdent des caractéristiques représentatives.
- classtype-admin : est utilisé pour trier les attaques en sous-groupes.
- sid : identificateur de règle de SNORT (numérotation).
- rev : numéro de révision de la règle.
- msg : spécifie le nom de l'alerte qui devra être envoyée en cas de détection.

19.3.2.2 Les alertes

La détection d'intrusions utilisent des taux de vraies et fausses alertes :
- les fausses alertes négatives : l'IDS ne signale aucune intrusion alors qu'une a lieu.
- les vraies alertes positives : l'IDS signale à raison qu'une attaque est en cours ou a eu lieu.
- les fausse alertes positives : l'IDS signale une alerte alors qu'aucune attaque n'a eu lieu.
- les vraies alertes négatives : c'est la situation normale, rien ne se passe et l'IDS ne réagit pas.

19.3.2.3 Sécurité apportée par les IDS

Selon les différents critéres d'analyse, il est possible de déterminer l'IDS le mieux adapté à ses besoins. Malheureusement, il est bien rare de trouver l'IDS correspondant à tous les critères de sélection.

Il ne faut pas croire qu'un IDS assure la sécurité du réseau à lui seul. Comme on l'a vu, chaque type d'analyse possède ses avantages et ses inconvénients. De plus, en cas de mauvaise maitenance de la base de données, certaines attaques peuvent passer totalement au travers de l'IDS.

Il faut également remarquer que les IDS ne sont jamais robustes à 100%, bien loin de là. Les recherches sont très actives dans ce domaine. La majorité des attaques ne nécessitent qu'une modification mineure pour ne pas être détectée.

19.3.3 La tolérance d'intrusions

Comme nous venons de l'expliquer, les IDS ne sont pas assez fiables pour garantir la sécurité d'un réseau. En conséquence de cette remarque, on voit émerger depuis quelques temps des "systèmes ditribués à tolérance d'intrusions". Sous ce nom se cache en réalité une méthode de gestion des données entre plusieurs ordinateurs distants et autorisant une ou plusieurs intrusions, en se basant sur le principe que le pirate n'aura pas accès à la totalité des informations qu'il désire obtenir.

On définit un tel système comme étant un système capable de garantir la confidentialité, l'intégrité et la disponibilité des données, et ce même en cas d'intrusion dans une partie du système.

Ce concept peut s'implémenter par la technique de "fragmentation-duplication-dispersion".

La technique "fragmentation-redundancy-scattering"

Le principe est de diviser les données en plusieurs parties et les répartir en plusieurs endroits afin qu'en cas d'intrusion, le pirate n'ait pas accès au reste de l'information. Pour l'obtenir, il devra déterminer les différents sites de stockage, ce qui ne lui sera à priori pas possible. L'information proprement dite symbolise aussi bien les données, les droits d'accès, ou les programmes.

Grâce à cette méthode, on peut donc autoriser (*tolérer*) un certain nombre d'intrusions dans le système. Ainsi, on aura toujours un respect de la confidentialité et de l'intégrité. La disponibilité des données à tout moment est garantie par le principe de duplication de l'information et de sa répartition en plusieurs endroits.

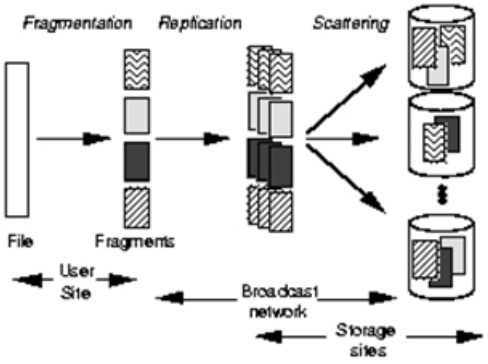

FIG. 19.5 – Technique de tolérance d'intrusions

19.3.4 Les honeypots

A l'opposé des firewalls et des IDS, les honeypots (littéralement, "pot de miel") ne cherchent pas à détecter ou prévenir une attaque particulière. Au contraire, un honeypot se définit comme étant un système de recueil d'informations sur les utilisations illicites et non autorisées de ce système.

Le principe du honeypot est qu'il représente une station accessible par les pirates, la plupart du temps dépourvue de toute protection. Théoriquement, un honeypot ne devrait générer aucun trafic car aucune activité ne lui est autorisée. Ainsi, tout intéraction avec lui ne peut être qu'illicite. On considérera donc toute activité du honeypot comme suspecte. Par l'intermédiaire du honeypot, il est alors possible de lister tous les accès et toutes les commandes qu'il subit ou reçoit afin, dans un second temps, d'analyser ces

accès et les résultats associés pour déterminer les méthodes d'attaques des pirates.

Les honeypots peuvent être de formes et de tailles très variables. Ils ont également des avantages et inconvénients, comme toute technologie.

Au niveau des avantages, on trouvera :

- des ensembles d'informations réduits mais de plus grande utilité : au lieu de générer des milliers d'alertes par jour, un honeypot ne pourra générer qu'une petite dizaine d'alertes, mais dont la qualité et l'utilité seront beaucoup plus grande. En effet, de par sa nature, le honeypot ne capture que l'activité suspecte, ce qui d'une part réduit le taux de fausses alertes, et d'autre part, facilite l'analyse des données recueillies.
- de nouveaux outils et de nouvelles techniques : comme un honeypot capture tout ce qu'il subit, il capture également les nouvelles méthodes d'attaques. Il ne travaille pas à partir d'une base de données comme la plupart des IDS.
- un consommation minimale de ressources : pour construire un honeypot, il n'est pas nécessaire d'avoir une station puissante, un simple Pentium avec 128 Mb de RAM est amplement suffisant.
- la simplicité : les honeypots sont relativement simples à comprendre et à mettre en place. Ils ne nécessitent pas d'algorithmes complexes, de bases de données à maintenir à jour, ou de listes de signatures.

En ce qui concerne les inconvénients, on peut citer :

- une vue limitée : bien que les honeypots collectent toute intéraction avec eux-même, ils ne peuvent relever les informations sur d'autres machines.
- le risque : il peut arriver qu'un pirate prenne entièrement le contrôle du honeypot cible. Si le honeypot est isolé, cela ne posera pas énormément de problèmes, mais s'il s'agit d'une station spécifique dans un réseau, le risque est plus élevé.

19.3.4.1 Les types de honeypots

La plupart du temps, on place les honeypots dans deux catégories : à basse intéraction ou à haute intéraction.

Les honeypots *à basse intéraction* travaillent en émulant des systèmes d'exploitation ou des services. Relativement simples d'emploi, ils présentent un risque minimal. Il fonctionne à la suite d'une installation software, et d'une sélection des services à émuler. L'avantage de ce type de honeypot est qu'il ne dispose pas réellement d'un système d'exploitation, il ne fait que l'émuler. Le pirate n'a donc jamais la possibilité d'attaquer ou d'affaiblir les autres systèmes. L'inconvénient majeur est que ces honeypots ne peuvent écouter que les attaques ayant traits aux services qu'ils émulent. Ils sont alors totalement insensibles aux attaques basées sur d'autres services.

Les honeypots *à haute intéraction* sont des solutions beaucoup plus complexes, en ce sens qu'ils n'émulent rien. Ils utilisent un système d'exploitation réel et de véritables applications. A la différence des honeypots à basse intéraction, le pirate a ici accès à la réalité et non à l'émulation. Le principal avantage est qu'il est alors possible de lister l'entièreté des accès qu'il subit, pour tous les services. De plus, en donnant l'entièreté du sytème au pirate, on peut analyser l'attaque d'une manière beaucoup plus complète. En contrepartie, puisque le pirate a accès au système réel, le risque d'affaiblir d'autres stations est plus grand.

Enfin, il faut noter qu'un honeypot à haute interaction peut faire tout ce que fait un honeypot à basse intération, et bien plus. Cependant, ils sont plus complexes à déployer et à maintenir en ordre.

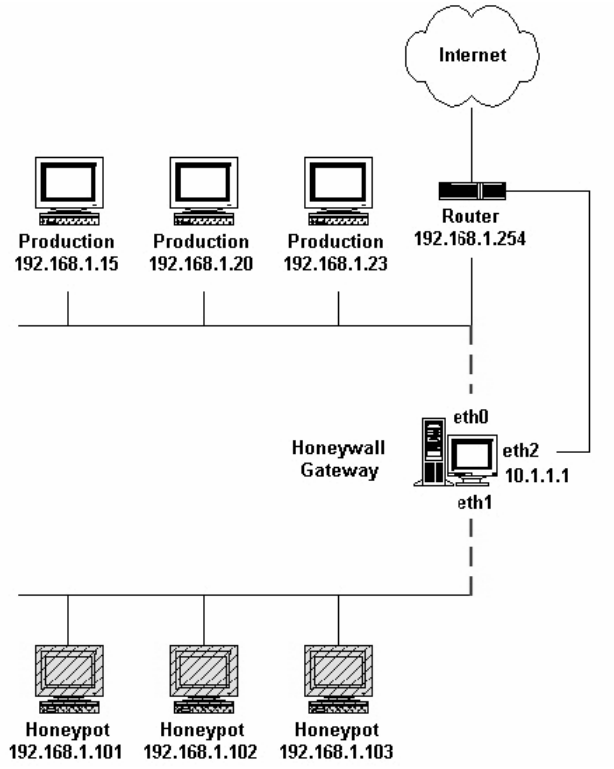

FIG. 19.6 – HoneyNet : un honeypot à haute intéraction

19.3.4.2 Utilisation des honeypots

En général, les honeypots peuvent avoir deux utilités distinctes : la protection et la recherche. Le premier point consiste à protéger une organisation, et plus particulièrement de les aider à prévenir, détecter et répondre aux attaques. En ce qui concerne la recherche, les honeypots sont essentiellement utilisés pour collecter l'information. En général, les honeypots à basse intéraction seront utilisés pour la protection tandis que les honeypots à haute intéraction seront employés pour la recherche.

Les tâches consistant à protéger une entreprise peuvent être explicitées comme suit :

- **la prévention** : le principe est de faire perdre du temps au pirate. Dans le cas d'un scan de ports, le honeypot pourrait ralentir sa vitesse de scan[4]. Grâce à ce ralentissement, l'entreprise aurait alors le temps de réagir. Le honeypot se rapproche ici d'un IDS, avec la faculté supplémentaire de tenter de "contrer" l'attaque. De plus, le simple fait d'utiliser un honeypot peut décourager le pirate. Comme ce dernier ne sait pas à priori quelle machine est le honeypot, il ne risquera pas de s'y enliser.
- **la détection** : les honeypots réduisent le nombre de fausses alertes positives, et peuvent détecter des attaques inconnues. Les honeypots à basse intéraction semblent les plus pratiques pour cette détection, de par leur facilité de mise en place. Un honeypot à haute intéraction permet de mieux étudier l'attaque, mais le risque que celle-ci s'étende est beaucoup plus grand, ce qui met en péril la protection de l'entreprise.
- **la réponse aux attaques** : lors d'une attaque, le pirate ayant le champ libre pour opérer, il laissera des traces sur son identité, sa localisation, comment il s'est introduit, ou les dommages qu'il a causé. Toutes ses informations pourront être réutilisées par l'entreprise pour répliquer.

[4]par l'intermédiaire de plages IP non utilisées par exemple.

19.3.5 Les firewalls

Qu'il s'agisse d'une machine personnelle, ou d'un réseau d'entreprise, tous les systèmes sont susceptibles d'être un jour la cible d'une attaque. La première protection, en marge de l'antivirus, sera d'utiliser un firewall. Le rôle du firewall sera d'empêcher les pirates de s'introduire sur le système placé derrière lui.

Un firewall peut être software ou hardware. Pour protéger un système, il se placera à la frontière entre Internet (ou un réseau) et la machine (ou les machines) du réseau. Il servira de filtre, et si un paquet ne correspond pas aux filtres qu'il renferme, le paquet en question sera rejeté.

Un firewall peut être configuré à de nombreux niveaux :
- Les adresses IP : on peut lui faire accepter les flux de données provenant d'une plage d'adresses, ou même d'une adresse uniquement.
- Les noms de domaine : il est également possible d'empêcher l'accès à certaines adresses Internet.
- Les protocoles : pour empêcher tout transfert FTP, tout accès Telnet, ou encore pour éviter le surf sur Internet (HTTP).
- Les ports : pour supprimer le FTP, on peut refuser les connexions sur le port 21.
- Par mots ou phrases : semblable aux expressions régulières, il est possible de refuser les paquets dont le contenu renferme des séquences de lettres données.

19.3.5.1 Principes de fonctionnement

Le filtrage de paquets (*Packet Filtering*)

Les paquets sont analysés en les comparant à un ensemble de filtres (c'est-à-dire à un ensemble de règles). Les paquets seront alors soit rejetés, soit acceptés et transmis au réseau interne.

Lorsque le paquet arrive au firewall, celui-ci analyse les champs IP et TCP/UDP. Ils sont confrontés à chacune des règles spécifiées dans la table des autorisations présente dans le firewall, et configurée par l'administrateur du système. Selon les règles qui autorisent ou refusent la transmission des paquets, le firewall obéira aux ordres. Si un paquet ne satisfait à aucune des règles, il est soit rejeté, soit accepté, suivant la philosophie choisie par l'administrateur réseau :

- Ce qui n'est pas expressément permis est interdit
- Ce qui n'est pas expressément interdit est permis.

La première de ces deux approches est beaucoup plus sûre. La seconde est plus risquée car elle suppose que l'administrateur est certain d'avoir envisagé tous les cas pouvant engendrer des problèmes.

On pourrait par exemple avoir une règle du type suivant :

Rule	Action	Src Host	Src H. Port	Dst Host	Dst H. Port	Flags TCP/Options IP
1	Allow	124.234.184.74	*	154.214.217.1	*	SYN

L'avantage du filtrage par paquet est sa rapidité. Il est de plus relativement simple à implanter dans un réseau.

Cependant, la sécurité ne peut se baser uniquement sur le filtrage par paquet. D'une part, il est difficile de maintenir un niveau suffisant de sécurité lorsque le nombre de règles augmente. D'autre part, le type d'informations accessibles à ce niveau est limité, en l'occurrence, il n'identifie que la machine et non son utilisateur.

Le filtrage du flux (*Circuit Filtering*)

Le filtrage de flux ne prête pas attention au contenu des paquets transitant sur la connexion. De ce fait, ce type de filtrage ne peut être utilisé pour assurer l'authentification des parties, ou la sécurité du protocole par l'intermédiaire duquel a lieu la connexion.

A la différence du filtrage de paquets, qui est considéré comme permissif, le filtrage de flux est restrictif. En effet, il n'autorisera le flux entre deux entités que si la connexion entre ces deux entités existe. On peut voir ce principe comme la création d'un tunnel entre deux machines. De ce fait, le circuit-filtering ne sera souvent utilisé qu'en complément de l'*application gateway*.

La passerelle applicative (*Application Gateway*)

A la différence du filtrage de paquets, qui analyse les paquets individuellement, l'application gateway permet de limiter les commandes à un service plutôt que de l'interdire.

Ce principe de fonctionnement empêche le trafic direct entre le réseau protégé et l'Internet, et ce dans les deux sens. Le trafic interne n'atteindra jamais Internet, et inversément, aucun trafic Internet ne voyagera sur le réseau interne. En effet, chaque client interne se connectera sur un serveur proxy (qui est la base de ce principe). Toutes les communications se feront par l'intermédiaire de celui-ci. Il déterminera si le service demandé par l'utilisateur est permis et se connectera avec le destinataire en cas d'autorisation. Le destinataire ne connaîtra pas l'adresse de son correspondant . Il ne communiquera qu'avec le serveur proxy, qui jouera en réalité le rôle d'un translateur d'adresse réseau (NAT).

La sécurité peut être ici très élevée. Agissant au niveau applicatif, on peut notamment la retrouver dans l'authentification par mot de passe des utilisateurs.

En tant qu'inconvénient, ce principe de fonctionnement supposera que toutes les machines sont configurées pour communiquer avec ce proxy, ce qui impliquera probablement une configuration individuelle de chaque machine ou l'installation de logiciels sur chacune d'elles.

19.3.5.2 Les types d'architectures

Il existe 3 types de firewalls. Chacun possède des avantages et inconvénients. Il faudra donc préalablement analyser les besoins réels en termes de sécurité, ainsi que les coûts engendrés.

Les firewalls *Bridge*

Ce format de mur de feu a l'apparence d'un simple cable réseau, sans machine spécifique. Il est invisible et indétectable pour un pirate, son adresse MAC ne circulant jamais sur le réseau. Placé sur le réseau, le pirate devra donc automatiquement passer par lui pour transmettre des requêtes. On trouvera notamment ce type de firewalls dans des switchs.

- **Avantages** :
  - Il n'est pas possible de l'éviter.
  - Il est relativement peu coûteux.
  - Il est transparent.

- **Inconvénients** :
  - Pour le contourner, il "suffit" d'adapter l'attaque.
  - Ses fonctionnalités sont souvent restreintes (il ne s'agit la plupart du temps que d'un filtrage de paquets).

Les firewalls *hardware*

Ils sont souvent assimilés à des boites noires, l'accès à leur code étant difficile. Ce type de matériel propriétaire renferme d'ailleurs souvent un système de protection permettant d'authentifier le logiciel associé (par signature RSA par exemple), et ainsi rendre toute modification pratiquement impossible.

- **Avantages** :
  - Il est facilement intégré au réseau.
  - Son administration est souvent simplifiée.
  - Le niveau de sécurité est assez élevé.
- **Inconvénients** :
  - Ce type de firewall étant propriétaire, les mises à jour dépendent entièrement du constructeur.
  - En raison de l'architecture hardware, peu de modifications sont autorisées.

Les firewalls *logiciels*

Ces firewalls existent autant sous forme commerciales que sous forme gratuites. Quelque soit son origine, la sécurité pourra fortement varier. Un logiciel commercial pourra parfois mettre en avant sa facilité de mise en place et de configuration, mais ce sera souvent aux dépends de la sécurité. Au niveau des logiciels gratuits et/ou libres, ils seront souvent plus flexibles (c'est-à-dire plus fournis en options), mais nécessiteront la plupart du temps de bonnes connaissances en réseau afin de les configurer finement sans abaisser le niveau de sécurité.

19.4 La notion de Vulnérabilité

Après avoir passé en revue une série de "malwares", traitons d'une façon un peu plus générale de ce que désigne la notion de vulnérabilité, les concepts voisins ainsi que les moyens de les gérer.

Donnons-en tout d'abord une définition. Nous dirons qu'une vulnérabilité est une faiblesse inhérente à un objet (software ou hardware). Elle a principalement deux origines :

- Faute de **conception** : un algorithme cryptographique qui présente une faiblesse qui peut être volontaire (présence d'une trappe) ou involontaire (ce que l'on appelle vulgairement une faille)
- Faute d'**utilisation** : une méprise ou une inattention dans la procédure d'utilisation d'un logiciel ou du système

Un terme souvent associé à la notion de vulnérabilité est le terme "exploit". Cet exploit sera défini comme attaque profitant d'une faille pour modifier le comportement du système ou en prendre le contrôle.

Plusieurs types d'exploits existent :
- Remote Exploit : l'attaque peut se faire à distance
- Local Exploit : un accès sur la machine cible est nécessaire
- Zero-Day Exploit : lorsqu'une faille est publiée, on donne cette désignation aux attaques l'exploitant le jour même de la publication. Par extension, on nomme ainsi les attaques publiées avant la sortie du patch lié à la vulnérabilité, et les attaques reposant sur les failles découvertes par les pirates eux-même (là aussi jusqu'à la sortie d'un patch).

Il n'existe pas à proprement parler un classement des vulnérabilités selon leur degré de gravité. On admet cependant couramment les catégories "Faible, moyenne et haute".
- High : pas d'interaction de l'utilisateur, globale
- Medium : faible interaction
- Low : forte interaction, localisé

On trouve également parfois le qualificatif “critique". Toutefois, le fait de classer une vulnérabilité dans telle ou telle catégorie est purement subjectif. Ainsi une faille classée “critique” par une société X sera peut-être classée “moyenne” par une société Y. Un exemple est donné à la figure 19.7.

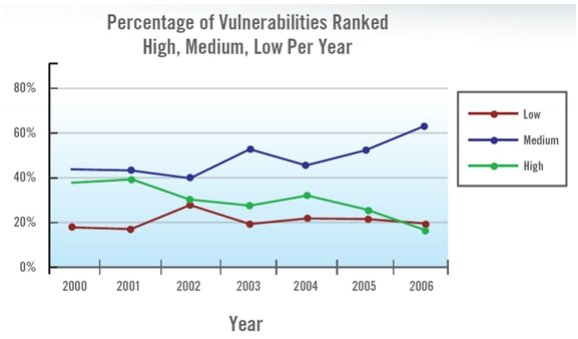

FIG. 19.7 – Classification des vulnérabilités en 2007 selon IBM

19.4.1 La fenêtre de vulnérabilité

Ce concept est défini comme le temps entre la mise en place d’une attaque exploitant une vulnérabilité et l’installation du correctif par l’utilisateur. Il existe trois cas distincts.

19.4.1.1 Cas favorable

1. Découverte de la faille par des whitehats/chercheurs
2. Correction de la vulnérabilité par le vendeur
3. Exploitation par les blackhats
4. Installation du correctif chez l’utilisateur

La fenêtre de vulnérabilité s’étend de T3 à T4.

Remarquons tout de même que ce n’est pas parce qu’un patch existe qu’il est installé par l’utilisateur. Dans la majorité des cas, le patch existera, mais par méconnaissance, ou inconscience, l’utilisateur omettra d’installer le correctif.

19.4.1.2 Cas défavorable

1. Découverte de la faille par des whitehats/chercheurs
2. Exploitation par les blackhats
3. Correction de la vulnérabilité par le vendeur
4. Installation du correctif chez l’utilisateur

La fenêtre de vulnérabilité s’étend de T2 à T4.

19.4.1.3 Cas critique

1. Découverte/exploitation de la faille par des blackhats
2. Correction de la vulnérabilité par le vendeur
3. Installation du correctif chez l’utilisateur

La fenêtre de vulnérabilité s’étend de T1 à T3.

19.4.2 Supprimer les vulnérabilités

Il existe un “certificat d’aptitude” au coding sûr. Il porte le nom de GSSP (Global Information Assurance Certification Secure Software Programmer). Ce certificat fut mis au point après la constatation que 85% des vulnérabilités étaient dues à 3 erreurs de programmation :

- Le buffer overflow
- L’absence de vérification des insertions de données par l’utilisateur
- L’integer overflow

L’absence de vérification des données (“Format String”), permet notamment le XSS (Cross Site Scripting) : corruption de forums autorisant par exemple l’insertion de code dans les messages utilisateur (HTML,...), ou encore l’obtention d’accès cachés aux serveurs en envoyant les paramètres adéquats dans les adresses URL.

L’Integer overflow, correspondant à la non-vérification du format des nombres utilisés, permet de rendre vulnérable la machine exécutant le code en question : la conséquence la plus courante est un simple crash, mais peut aller jusqu’à permettre l’introduction de code malicieux, selon l’état dans lequel se trouve la machine après exploitation de la faille.

19.4.2.1 Un exemple concret : Le buffer overflow

Le buffer overflow est défini comme étant une erreur logicielle par laquelle des données sont stockées dans un espace plus important que celui initialement réservé. Il s’agit de la vulnérabilité la plus souvent observée. Elle fut notamment exploitée par les vers CodeRed et Blaster.

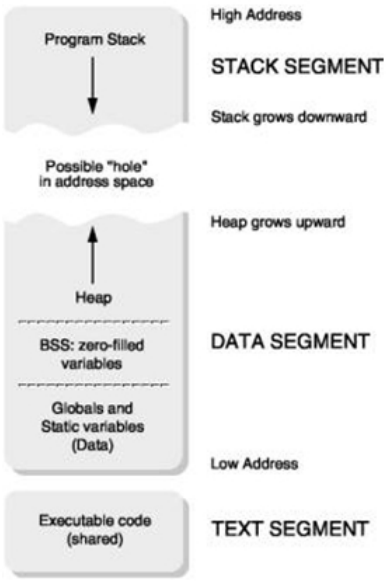

FIG. 19.8 – Organisation de la mémoire - http ://www.computerworld.com/

Sur la figure 19.9, on aperçoit le comportement normal de la fonction strcpy(). A priori, sans avoir été averti d’une utilisation malicieuse possible, on ne détecte pas directement pourquoi cette fonction peut poser un quelconque problème. Toutefois, la mise en situation évoquée à la figure 19.10 montre le contraire : si aucune vérification de la taille effective des données pointées par *bar* n’est faite, une corruption est possible, pouvant permettre l’insertion de code.

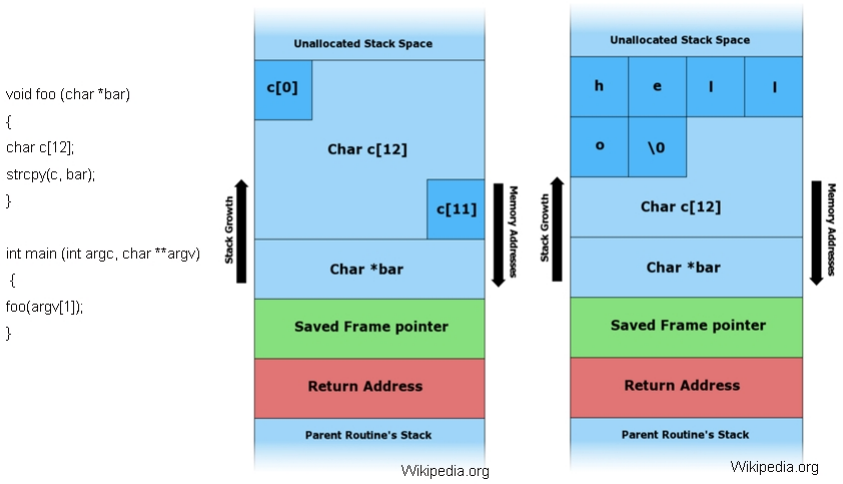

FIG. 19.9 – Utilisation normale de la fonction strcpy()

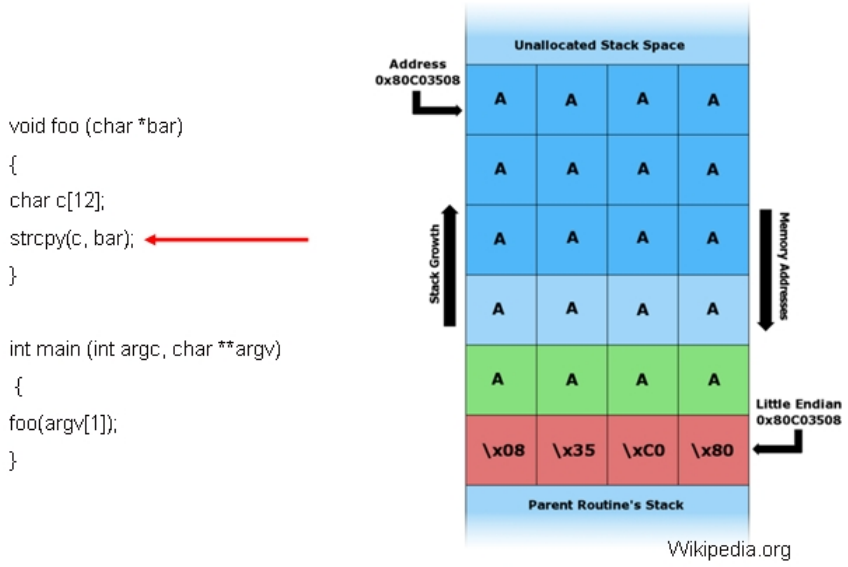

FIG. 19.10 – Utilisation possible de la fonction strcpy()

Il existe un certain nombre de fonctions dont il est bon de se méfier. Une liste est donnée à la figure 19.11.

Il existe d'autres solutions pour éviter ces problèmes de buffer overflow :

– Utiliser des canaris : il s'agit de placer sur la pile des nombres aléatoires pour chaque appel de fonction à la suite de l'adresse de retour. Ainsi, après vérification de ces valeurs, on peut déterminer si une erreur de type Buffer overflow a pu être commise (non concordance) ou non (concordance). L'inconvénient principal dans l'utilisation de ces canaris est le fait qu'il soit nécessaire de recompiler le code source.

Function	Severity	Solution
gets	Most risky	Use fgets(buf, size, stdin). This is almost always a big problem!
strcpy	Very risky	Use strncpy instead.
strcat	Very risky	Use strncat instead.
sprintf	Very risky	Use snprintf instead, or use precision specifiers.
scanf	Very risky	Use precision specifiers, or do your own parsing.
sscanf	Very risky	Use precision specifiers, or do your own parsing.
fscanf	Very risky	Use precision specifiers, or do your own parsing.
vfscanf	Very risky	Use precision specifiers, or do your own parsing.
vsprintf	Very risky	Use vsnprintf instead, or use precision specifiers.
vscanf	Very risky	Use precision specifiers, or do your own parsing.
vsscanf	Very risky	Use precision specifiers, or do your own parsing.
streadd	Very risky	Make sure you allocate 4 times the size of the source parameter as the size of the destination.
strecpy	Very risky	Make sure you allocate 4 times the size of the source parameter as the size of the destination.
strtrns	Risky	Manually check to see that the destination is at least the same size as the source string.
realpath	Very risky (or less, depending on the implementation)	Allocate your buffer to be of size MAXPATHLEN. Also, manually check arguments to ensure the input argument is no larger than MAXPATHLEN.
syslog	Very risky (or less, depending on the implementation)	Truncate all string inputs at a reasonable size before passing them to this function.
getopt	Very risky (or less, depending on the implementation)	Truncate all string inputs at a reasonable size before passing them to this function.
getopt_long	Very risky (or less, depending on the implementation)	Truncate all string inputs at a reasonable size before passing them to this function.
getpass	Very risky (or less, depending on the implementation)	Truncate all string inputs at a reasonable size before passing them to this function.
getchar	Moderate risk	Make sure to check your buffer boundaries if using this function in a loop.
fgetc	Moderate risk	Make sure to check your buffer boundaries if using this function in a loop.
getc	Moderate risk	Make sure to check your buffer boundaries if using this function in a loop.
read	Moderate risk	Make sure to check your buffer boundaries if using this function in a loop.
bcopy	Low risk	Make sure that your buffer is as big as you say it is.
fgets	Low risk	Make sure that your buffer is as big as you say it is.
memcpy	Low risk	Make sure that your buffer is as big as you say it is.
snprintf	Low risk	Make sure that your buffer is as big as you say it is.
strccpy	Low risk	Make sure that your buffer is as big as you say it is.
strcadd	Low risk	Make sure that your buffer is as big as you say it is.
strncpy	Low risk	Make sure that your buffer is as big as you say it is.
vsnprintf	Low risk	Make sure that your buffer is as big as you say it is.

FIG. 19.11 – http ://www.ibm.com/developerworks/library/s-buffer-defend.html

– Utiliser des librairies “sécurisées” : certaines librairies fournissent des fonctions sûrs.
– Utiliser des langages autres que C et C++, ceux-ci facilitant la mise en place de telles vulnérabilités. La figure 19.12 donne une liste de langage et leur éventuelle protection par rapport au problème du buffer-overflow. Attention au fait que la présence d’un “Safe” dans ce tableau ne concerne que le buffer-overflow, et pas d’autres types d’erreurs de programmation !
– Il est parfois possible d’empêcher l’exécution de code sur la pile ou sur le tas par l’intermédiaire de

Language/Environment	Compiled or Interpreted	Strongly Typed	Direct Memory Access	Safe or Unsafe
Java, Java Virtual Machine (JVM)	Both	Yes	No	Safe
.NET	Both	Yes	No	Safe
Perl	Both	Yes	No	Safe
Python - interpreted	Intepreted	Yes	No	Safe
Ruby	Interpreted	Yes	No	Safe
C/C++	Compiled	No	Yes	Unsafe
Assembly	Compiled	No	Yes	Unsafe
COBOL	Compiled	Yes	No	Safe

FIG. 19.12 – Liste de langages et protection face au buffer-overflow

flags CPU ou directement dans l'OS.
- On peut également distribuer aléatoirement les segments en mémoire. De ce fait, le pirate ne sait pas à priori où se situe le code à exécuter.

19.4.2.2 Conclusions

En règle générale, la protection contre ces vulnérabilités devra avoir lieu à trois moments distincts.
- Avant : Se documenter, être attentif lors du choix du langage
- Pendant : Être attentif aux fonctions employées, réaliser un audit du code (que ce soit en interne, ou par des sociétés spécialisées), utiliser des outils de scanning
- Après : Tester le programme et les entrées possibles (longueur, caractères aléatoires, etc.)

19.5 Ressources supplémentaires

http://www.vulnerabilite.com/dossier/
http://www.linuxfocus.org/Francais/July2003/article294.shtml
http://www.snort.org/docs/
http://teamlog.developpez.com/virus/
http://www.miscmag.com/articles/index.php3?page=103
http://www.simovits.com/trojans/trojans.html
http://www.tracking-hackers.com/papers/
http://www.tele.ucl.ac.be/EDU/ELEC2920/1997/firewall/Firewalls.html
http://www.interhack.net/pubs/fwfaq/

Chapitre 20

La sécurité en entreprise

20.1 La notion de risque

La sécurité d'une entreprise est très souvent attribuée au département IT de celle-ci. En tant qu'ingénieur système ou chef de sécurité, il faut pouvoir "gérer" cette sécurité. Dans l'idéal, cette gestion devra se faire avec le soutien d'instances supérieures de l'entreprise (p.e. le pouvoir organisateur).

Comme nous l'avons déjà mentionné, la sécurité est un compromis entre coûts, risques et contraintes :
- un système ultra-sécurisé mais très contraignant ne sera pas utilisable en pratique
- un système non sécurisé encore moins...

Il faudra peser le pour et le contre de chaque élément.

20.1.1 Définitions

Avant toute chose, définissons quelques termes, dont certains ont déjà été évoqués :

- **Vulnérabilité** : faiblesse inhérente à un objet (software ou hardware)
- **Menace** : danger (interne ou externe) tel qu'un hacker, un virus, etc.
- **Risque** : probabilité qu'une menace exploite une vulnérabilité
- **Contre-mesure** : moyen permettant de réduire le risque

On comprend mieux le poids d'un risque en se fiant à la modélisation suivante :

$$\text{Risque } = \frac{Menace \text{ X } Vulnerabilites}{Contremesures}$$

Conséquences :
- Le risque est d'autant plus réduit que les contremesures sont nombreuses
- Le risque est plus important si les vulnérabilités sont nombreuses

20.1.2 Gérer le risque

La gestion des risques consiste en trois actions majeures :

- Etudier les risques potentiels (identifier/mettre au jour ces risques)
- Imposer des règles de sécurité adéquates pour réduire ces risques
- Eduquer les employés

20.1.2.1 Etudier les risques : méthodologie

1. Définition de l'environnement
   - Définition des acteurs et leurs intérêts
   - Importance de la sécurité dans la stratégie de l'entreprise
     - Type de données impliquées
     - Visibilité extérieure de la sécurité (importance pour la clientèle, le public ?)
2. Etude des menaces
   - Nature : accidentelles (désastre, bugs, ...) ou volontaires (attaques, vols, ...)
   - Sources : personnel non autorisé, intrus, logiciel
   - Localisation : procédures manuelles, informatique (software, réseau, stockage, hardware), infrastructure (concrète et abstraite)
3. Etude des vulnérabilités
   - Etudes des faiblesses engendrées par l'exécution d'une menace
4. Etude des risques
   - Probabilité d'occurrence de ces menaces conduisant à une vulnérabilité
5. Estimation du risque et du plan stratégique
   - Risque :
     - Cout des pertes engendrées (à court, moyen et long terme)
     - Cout de la mise en place de la contre-mesure (tant au niveau logique que logistique)
     - Comparer la perte potentielle au coût de la contre-mesure (quantification mathématique)
   - Plan stratégique :
     - Planning de l'implémentation avec prise en compte des besoins futurs (en termes de sécurité ou non)
     - Planning du suivi de l'implémentation
6. Mise en place du plan de sécurité
   - Mise en place des correctifs
   - Définition de la police de sécurité
     - Objectifs, Portée, Responsables
     - Description de la sécurité (de l'infrastructure physique, des données informatiques, des applications, du réseau).
     - Plan en cas de sinistre
     - Sensibilisation du personnel aux nouvelles procédures
     - Sanctions en cas de manquements
7. Audit de sécurité
   - Système informatisé : scanneurs réseau, évaluation des algorithmes utilisés (force brute, outils de cracking, ...)
   - Mesurer l'efficacité de la solution implémentée dans sa globalité Champ plus large qu'une étude stricte des éléments mis en place

Les étapes 4 et 5 forment ce que l'on nomme communément la phase d'*Analyse des risques*.

20.1.2.2 Imposer des règles de sécurité adéquates

Cela consistera en la définition de procédures internes à l'entreprise basées sur :

1. des règles administratives
   - Suivre des standards de sécurité (normes ISO)
   - Suivre les lois
2. des règles physiques
   - Gardes, caméras, alarmes, verrous
   - Accès aux locaux sécurisés par biométrie
3. des règles techniques

- Déterminer des niveaux de classification des données
- Définir des niveaux d'accès à ces données
- Utiliser la cryptographie pour le traitement et le stockage de l'information
- Mettre en place un firewall matériel et/ou logiciel
- ...

20.1.2.3 Eduquer les employés

Il est de plus en plus admis que la sécurité est essentielle. Le CSI (Computer Security Institute) confirme ce fait par la publication d'un rapport annuel[16], dans lequel on constate que chaque année, l'attention des responsables d'entreprises est accentuée sur la sécurité. Les couts engendrés par les pertes de données dues aux attaques réseaux et autres malwares diminuent sensiblement d'années en années[1].

Une dernière remarque doit cependant être faite concernant les employés. Jusqu'ici, nous avons vu qu'il fallait tenter d'obtenir le meilleur compromis de sécurité (efficacité vs convivialité). Mais n'oublions qu'une faille béante peut également exister en la personne des employés de l'entreprise. Si ceux-ci ne sont pas éduqués aux notions de base de la sécurité, il est fort probable que la sécurité en pâtisse lourdement.

Nous avons déjà cité ce thème dans le chapitre d'introduction (cube de McCumber) : le facteur humain. Il est beaucoup plus simple de corrompre l'utilisateur et ce qui l'entoure que l'algorithme de chiffrement utilisé :

- L'utilisateur ne connait pas les risques engendrés par la conservation de la liste des mots de passe utilisés à coté de l'ordinateur
- Il est souvent plus simple de s'introduire dans l'ordinateur de l'utilisateur afin de retrouver le texte en clair (hacking, vol, ...)
- Il est possible de l'espionner, le pousser à la délation, pratiquer le shoulder-surfing ou tout autre technique dite de "social engineering"
- ...

Il ne s'agira donc pas ici d'expliquer aux employés comment fonctionnent les algorithmes qu'ils utiliseront, mais plutôt comment et dans quelles conditions ils devront les utiliser en définissant des règles qui ne devront pas être transgressées.

A méditer...

20.1.3 Réagir face à un risque

Il y a plusieurs façons de réagir à un risque, des plus "sûres" aux plus inconscientes :

- Transférer les risques à une compagnie d'assurances
- Réduire les risques en implémentant des contre-mesures qui peuvent être :
  1. Dissuasives : empêcher une attaque
  2. Préventives : faire échouer une attaque
  3. Correctrices : réduire les dommages causés par une attaque
- Ignorer/Négliger les risques
- Accepter les risques si les contre-mesures sont trop onéreuses

Il y a toujours un risque, aussi infime soit-il. Il faudra donc peser le pour et le contre lors de la mise en place éventuelle d'une contre-mesure.

[1] Toutefois, en 2007, on remarque une remontée de la somme totale des pertes, due à la fraude financière.

20.2 La destruction des données

S'il existe un domaine de l'entreprise où la sécurité peut être inconsciemment négligée, c'est dans la destruction de ses données. Qu'il s'agisse de documents physiques (*Hard Copy*, tels que des rapports écrits, feuilles imprimées, fax, etc.) ou de supports informatiques (*Soft Copy*, tels des disques durs, disques optiques, bandes magnétiques, etc.), il est souvent recommandé de détruire ces données de façon sécurisée. Il existe en effet plusieurs standards définissant la marche à suivre et les outils à employer pour garantir un niveau de sécurité suffisant.

20.2.1 Types de traitement

Il existe 4 stades de destruction des données :

Niveau 0 Élimination (Disposal) : Données ne présentant pas de "danger" pour l'entreprise. Elles ne subissent aucun traitement particulier.

Niveau 1 Nettoyage (Clearing) : Protection des données permettant de contrer une "attaque logicielle" (Keyboard Attack), c'est-à-dire l'utilisation d'un programme de récupération de données.

Niveau 2 Purification (Purging) : Protection des données permettant de contrer une "attaque matérielle" (Laboratory Attack), c'est-à-dire l'utilisation de techniques liées au traitement du signal, MFM, etc.

Niveau 3 Destruction (Destroying) : plusieurs techniques sont autorisées telles que l'incinération, la "pulvérisation" ou la décomposition chimique.

Remarque : depuis 2001, les niveaux 1 et 2 ont fusionné en raison des évolutions des techniques d'écriture des données sur les disques.

20.2.2 La copie physique

Ce type de copie étant principalement sous forme papier, l'utilisation de destructeurs de papier est évidemment recommandée. La norme DIN 32757 identifie les catégories de destructeurs selon le format des particules émises.

Niveau	Taille	Niveau	Type
DIN 1	12 mm (bande)	Général	Documents quotidiens, destruction en vrac
DIN 2	6mm (bande)	Quotidien	Documents internes, financiers ou de travail
DIN 3	2mm (bande)	Confidentiel	Données de vente, salariales du service du personnel
DIN 4	2*15 mm (morceau) ou 30 mm²	Secret	Propositions stratégiques, dossiers personnels
DIN 5	0.8*12 mm (morceau) ou 10 mm²	Top Secret	Documents du gouvernement et de l'armée
6 (non officiel)	0.8*4 mm (morceau) ou 5 mm²	Secret-Défense	Documents du gouvernement et de l'armée

FIG. 20.1 – Destruction des copies physiques papier

Dans le cas du papier, signalons qu'il est également possible de les détruire par incinération, compostage ou décomposition chimique.

D'autres formes de copies physiques existent tels les tampons encreurs ou les épreuves d'imprimerie par exemple. Pour ces situations, l'incinération sera préférée.

20.2.3 La copie logicielle

Lorsqu'on parle de suppression de données logicielles, on se rend compte que pour beaucoup, cette suppression se résume à un simple "vidage de corbeille". En y regardant de plus près, plusieurs niveaux de suppression sont envisageables, de la plus naïve à la plus sûre :

1. Effacement des données (Delete)
2. Formatage
3. Formatage bas-niveau
4. Réécriture
5. Démagnétisation
6. Incinération

Les trois premières solutions ne sont en rien garantes d'une suppression *efficace* des données. La première ne supprime que les pointeurs, la deuxième ne fait que réinitialiser le boot-sector et la troisième permettra dans certains cas de retrouver des résidus d'une magnétisation passée.

La première solution efficace pour détruire des données est donc la réécriture (*Wipe*). Sujette à beaucoup de controverses, elle trouve son origine dans un article publié en 1996 par Guttman[18], dans lequel ce dernier proposait un effacement sûr des données par 35 réécritures successives (suite de 1, suite de 0 et suites aléatoires). Pour argumenter sa solution, Guttman évoque le fait suivant : les écritures successives ne se superposent pas parfaitement. En simplifiant, celà se traduit par le fait qu'un "1" écrasant un 0 renvoie 0.95, sinon 1.05. Alors que ce fait était bien réel pour les premiers durs et les disquettes d'antan, il s'avère que les disques durs actuels, dont la technologie d'encodage des données a bien évolué, ne rendent plus possible cette analyse.

La figure 20.2 présente quelques solutions pour cette réécriture. La norme de suppression DoD 5220 nécessitant 3 passes n'est plus d'application. L'utilisation de la commande Secure Erase, disponible en tant qu'utilitaire pour les disques durs actuels à la norme ATA, est aujourd'hui mise en avant et fonctionne par réécriture d'un pattern aléatoire de tous les blocs accessibles.

Type of Erasure	Average Time (100 GB)	Security	Comments
Normal File Deletion	Minutes	Very Poor	Deletes only file pointers, not actual data
DoD 5220 Block Erase	Up to several days	Medium	Need 3 writes + verify, cannot erase reassigned blocks
Secure Erase	1-2 hours	High	In-drive overwrite of all user accessible records
Enhanced Secure Erase	Seconds	Very high	Change in-drive encryption key
Fast Secure Erase	Seconds	Fair but crackable	Drive is set to secure erase with a password and turned off, when turned on again secure erase must complete. Can be broken by sophisticated forensic techniques

FIG. 20.2 – DOD 5220 et SE[19]

Aujourd'hui, tant le NIST[2] que les départements responsables de la sécurité nationale des différents gouvernements du monde entier s'entendent sur le fait qu'un seul passage est suffisant. Pour vérifier la

[2]http ://csrc.nist.gov/publications/nistpubs/800-88/NISTSP800-88_rev1.pdf

véracité de cette proposition, Wright[17] publia en décembre 2008 un article présentant les résultats de tests effectués en laboratoire à l'aide d'un MFM. Ses résultats sont les suivants :

- sur un disque d'usage normal, il y a moins d'1% de chance de récupérer un octet valide sur un disque d'usage normal.
- Même sur un disque neuf (*pristine* disk) n'ayant subi qu'une seule réécriture, il y a moins de 7% de chance de récupérer un mot de 32 bits valide
- Sur ce même disque, il y a 0,00002317% de récupérer un mot de 4 lettres issu d'un traitement de texte

Probability of recovery	Pristine drive	Used Drive (ideal)
1 bit	0.92	0.56
2 bit	0.8464	0.3136
4 bit	0.71639296	0.098345
8 bits[5]	0.51321887	0.009672
16 bits	0.26339361	9.35E-05
32 bits	**0.06937619**	**8.75E-09**
64 bits	0.00481306	7.66E-17
128 bits	2.3166E-05	5.86E-33
256 bits	5.3664E-10	3.44E-65
512 bits	2.8798E-19	1.2E-129
1024 bits	8.2934E-38	1.4E-258

FIG. 20.3 – Résultats de Wright

De plus, indépendamment de la validité de ces données (le fait que les données retrouvées soient bien celles effectivement inscrites avant la réécriture), quel serait leur poids devant une cour de justice avec une probabilité si faible ?[3]

Signalons encore la masse de données à traiter. Les données fournies par le MFM sont très lourdes : pour un disque de 20 Go, 40 To de données brutes sont à analyser[4].

Autres méthodes de destruction des données :

A coté des techniques logicielles, il existe des techniques physiques, plus radicales, mais également plus onéreuses :

- Démagnétisation (*Degausser*) : il s'agit d'appliquer un champ magnétique suffisamment puissant (supérieur à la coercivité du disque à détruire) au disque. Toutefois, les disques les plus récents semblent avoir une coercivité trop élevée en raison d'un blindage de plus en plus fort.
- Destruction physique : il s'agit de casser le disque en lui faisant subir une forte pression (> 5 tonnes)
- Incinération : en exposant le disque dur à une température supérieure à son point de Curie (770°C), au delà duquel le disque perdra ses propriétés magnétiques.

Pour les disques optiques, on pratiquera une "pulvérisation", en le réduisant en particules n'excédant pas 250μm.

La plupart de ces outils font l'objet d'analyses et de certification par la NSA[5]

[3] Wright annonce un taux de certitude minimum devant varier entre 95 et 99% dans le cadre d'une affaire criminelle.
[4] http ://www.marsllc.net/tab_services/data_destruction.htm
[5] http ://www.nsa.gov/ia/guidance/media_destruction_guidance/index.shtml

Chapitre 21

La biométrie

21.1 Fonctionnement

La biométrie est une méthode d'identification basée sur certaines données humaines, physiques ou comportementales, qui peuvent aller de la rétine, à l'empreinte digitale en passant par la voix ou la forme de la main. Chacune de ces méthodes est ce que l'on nomme un "Moyen Biométrique".

Les techniques biométriques furent avant tout mises au point pour pallier le problème des pertes de mot de passe et autres vols de cartes à puce.

Comme cela vient d'être dit, on classe les moyens en deux catégories :
- Les moyens **physiques** (physiologiques) tels que l'iris, l'empreinte digitale, ...
- Les moyens **comportementaux** tels que la voix, la dynamique de signature ou de frappe, ...

Chaque moyen possède ses avantages, ses inconvénients et ses applications. Plusieurs questions devront être posées telles que :
- Le mode de saisie est-il optimal ?
- Quel type d'analyse est-il permis d'opérer ?
- Quelles sont les capacités de stockage disponibles ?
- Quel est le type de vérification demandé ?

Les différents moyens n'ont pas le même niveau de sécurité, ni la même facilité de mise en oeuvre. De plus, à la différence des systèmes travaillant à partir de mots de passe, les réponses ne seront jamais fiables à 100%. Dans le cas des mots de passe, on peut tout de suite dire si la phase d'authentification a réussi ou non. Avec la biométrie, il faudra travailler selon des taux de similitude prédéfinis.

Il faudra aussi veiller au fait que les données humaines se modifient avec le temps, et que la manière avec laquelle sont relevées ces mêmes données n'est jamais identique. Par exemple, le doigt n'est jamais posé tout à fait de la même manière sur le support de saisie. Un problème similaire se produit avec l'écartement des doigts de la main.

La figure 21.1 illustre les différents moyens biométriques, ainsi que les critères de référence utilisés pour les classer. On définira ces derniers comme suit :
- *Intrusiveness* : indique à quel point l'utilisateur se sent "agressé" par la méthode d'identification
- *Accuracy* : indique l'efficacité du système pour identifier un utilisateur
- *Cost* : indique le coût de la mise en place d'un tel système (lecteurs, capteurs, stockage, ...)
- *Effort* : indique l'effort nécessité par l'utilisateur pour permettre la mesure

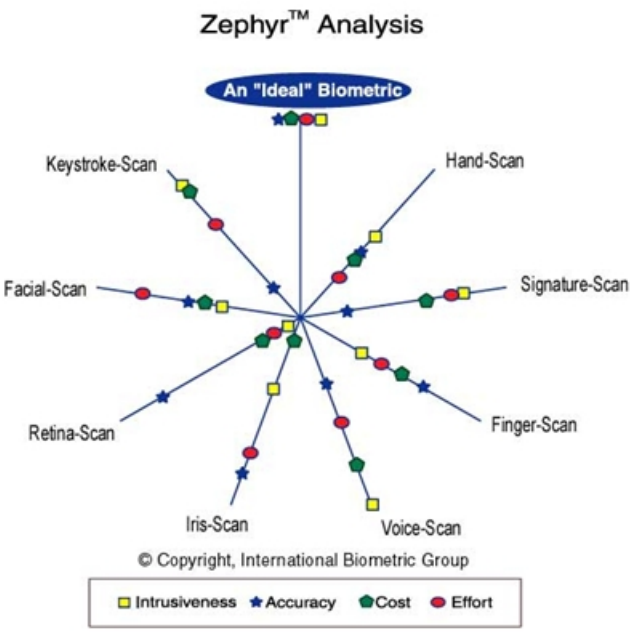

FIG. 21.1 – Les moyens biométriques et leurs caractéristiques

21.2 Mode de fonctionnement

Le processus biométrique se déroule en 3 étapes :

1. **Enregistrement** : l'utilisateur donne éventuellement un identifiant (PIN), puis utilise le capteur. Le système vérifie la qualité de la mesure, en extrait les points d'intérêt, et place la référence (template) dans sa base de données.

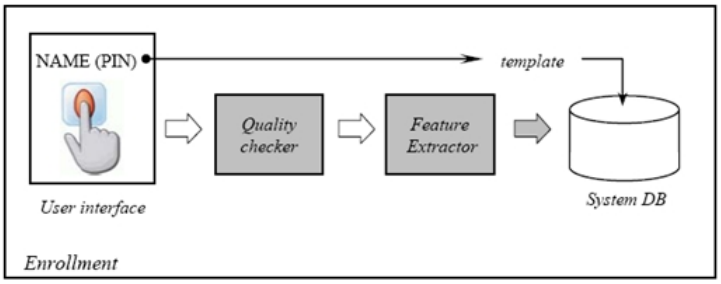

FIG. 21.2 – Enregistrement d'un utilisateur [11]

2. **Vérification** : l'utilisateur fournit son PIN et se présente au capteur. Le système cherche l'entrée correspondant au PIN dans sa BD et en extrait la référence associée. Le système compare la mesure et le template. Lorsqu'elle est possible, cette phase est la plus rapide. On parle de "positive recognition".
3. **Identification** : L'utilisateur se présente devant le capteur, sans fournir de code PIN. Le système extrait les points d'intérêt de la mesure et cherche dans sa BD s'il y a une correspondance. En conséquence, cette phase est plus lente que la précédente puisqu'il est plus rapide d'extraire une donnée en vue de la comparer que de comparer tous les templates. On parle de "negative recognition".

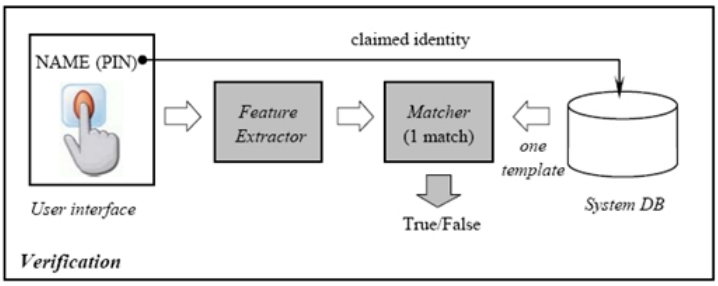

FIG. 21.3 – Vérification d'un utilisateur [11]

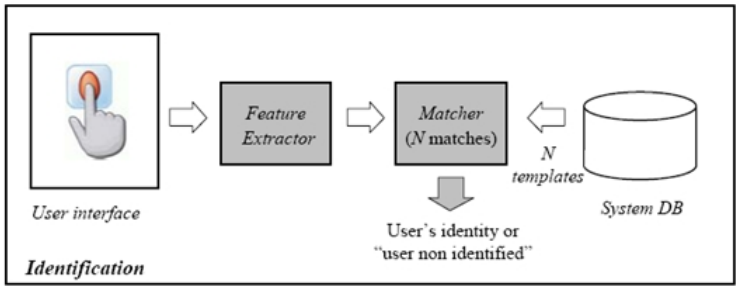

FIG. 21.4 – Identification d'un utilisateur [11]

21.3 Mesures des performances

L'inconvénient principal de la biométrie est qu'elle ne permet pas une authentification sûre à 100% : puisque les mesures se basent sur des propriétés physiques, elles peuvent se modifier avec le temps (âge, accident, blessure, ...). On parlera plutôt d'*identification*. L'objectif pour les créateurs de tels systèmes n'est donc pas la sécurité absolue, mais **un taux de certitude suffisant**.

Les performances s'expriment de plusieurs manières :

- T.F.R.[1] : Taux de Faux Rejets : nombre de personnes rejetées par erreur.
- T.F.A.[2] : Taux de Fausses Acceptations : nombre de personnes ayant été acceptées alors que cela n'aurait pas du être le cas.
- E.E.R. (x, t) : Equal Error Rate. Le seuil représentant le niveau d'erreurs acceptées.

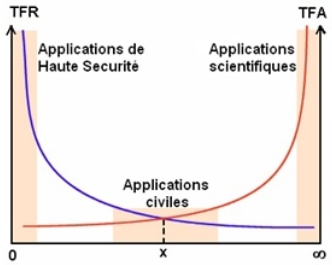

FIG. 21.5 – Minimisation des TFR et TFA et type des applications concernées

[1] ou FRR, pour False Rejection Rate, ou FNMR, False Non-Match Rate
[2] ou FAR, pour False Acceptation Rate, ou FMR, pour False Match Rate.

Un système sera *fonctionnel* lorsque son T.F.R. sera faible. Un système sera *sûr* lorsque son T.F.A. sera faible. L'objectif sera de minimiser ces deux valeurs, comme on le voit sur la figure 21.5. Le x (EER) représente la marge d'erreur autorisée. Plus il est grand, moins le système est sûr, mais plus on le réduit, moins le système est utilisable pour identifier des personnes. La recherche tente de limiter cet EER afin d'obtenir simultanément des TFA et TFR acceptables.

21.4 Moyens biométriques physiques

21.4.1 Les empreintes digitales (Finger-Scan)

Il s'agit de la plus ancienne technique d'identification, datant du début du 20^{me} siècle. Cette technique prend ses racines dans la découverte de la permanence des dessins de la naissance à la mort, son caractère individuel et son inaltérabilité par le chercheur britannique Galton.

Galton a défini la notion de *minuties* des empreintes digitales comme étant l'arrangement particulier des lignes des empreintes. Ces lignes forment des points caractéritiques permettant d'identifier de manière unique un individu. Ces points peuvent être des arrêts de lignes, des bifurcations, ou encore des îlots.

FIG. 21.6 – Type de minuties

Il existe de nombreux types de capteurs pour obtenir l'image et les minuties associées :
- Optiques par contact, avec ou sans contact : Le capteur utilise un CMOS (anciennement un CCD). Le scan est réalisé avec ou sans contact (doigt posé sur le capteur ou non). Si aucun contact n'a lieu, la lecture des empreintes a lieu par réflexion.
- Par balayage : le doigt défile sur le capteur et l'image est reconstruite au fur et à mesure de façon logicielle.
- Par capacité électrique : on mesure la capacité électrique existante entre la peau et le capteur. Elle varie comme l'inverse de la distance entre le doigt et le capteur. Ainsi, en présence d'une crête, la capacité électrique sera plus importante (car la distance est plus courte à cet endroit, puisque le doigt est posé sur le capteur).
- Par ultrasons : cette technique permet d'éviter le problème des doigts salis qui ne permettrait pas une bonne réflexion de la lumière.
- Par transmission de lumière : le capteur mesure l'intensité d'un faisceau traversant le doigt.

La technique d'extraction des empreintes porte le nom d'EDR : Empreinte Digitale Réduite. Le processus est illustré par la figure 21.7.

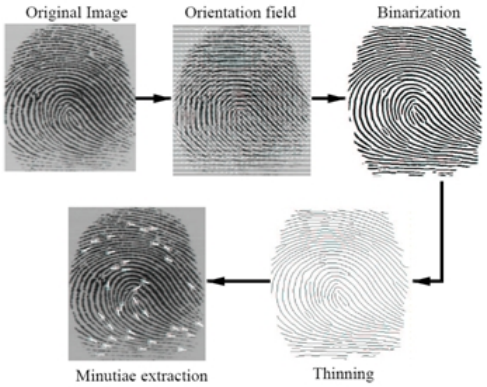

FIG. 21.7 – Processus de détection des minutcies [12]

21.4.1.1 Avantages

- C'est une technologie très connue du grand public, ce qui la rend moins intrusive
- La petite taille des nouveaux types de lecteurs permet une meilleure intégration dans de nombreuses applications
- Ces nouveaux lecteurs possèdent un moindre coût
- Le traitement des données est relativement rapide
- Les T.F.R. et T.F.A. sont assez faibles (si le nombre de points caractéristiques est assez élevé).

21.4.1.2 Inconvénients

- L'aspect "policier" donne un caractère négatif à l'identification
- Il est difficile de lire les données en présence de doigts blessés ou sales, selon la technique employée.
- L'utilisateur doit poser correctement son doigt sur le capteur, au risque de se voir rejeté.
- Le lecteur est exposé aux dégradations dûes aux passages réguliers des doigts et aux pressions exercées lorsqu'un contact est nécessaire.
- Il existe des techniques de contournement qui ne sont pas compliquées à mettre en oeuvre (ex : gummy fingers).

21.4.1.3 Applications

- Les domaines d'applications sont très larges : gsm, contrôle du démarrage d'un véhicule, accès à certains locaux,...
- Dans le domaine judiciaire, entre 8 et 20 points caractéristiques sont suffisantes pour une identification.

21.4.2 La forme de la main (Hand-Scan)

La silhouette de la main est également une caractéristique propre à chaque individu. Les paramètres pris en compte (juqu'à 90) peuvent être par exemple la longueur des doigts, la forme des articulations, ou encore leur épaisseur (dans une vue à trois dimensions). Le scanner utilisé est généralement à infra-rouges, et possède des guides afin de positionner correctement les doigts de l'individu.

21.4.2.1 Avantages

- C'est une technique simple à utiliser
- Elle est également très connue du grand public par l'intermédiaire du cinéma

21.4.2.2 Inconvénients

- La mise en place d'un tel système est parfois impossible au vu de la place nécessaire (il est impossible d'utiliser un tel lecteur dans une voiture, ou pour un téléphone portable ...)
- Ce moyen biométrique est sujet aux modifications de forme dûes au vieillissement de l'individu
- Il est également peu robuste en ce qui concerne l'identification de jumeaux, ou de membres d'une même famille

21.4.2.3 Application

- On l'utilise souvent dans les contrôles d'accès aux locaux

21.4.3 Les Veines (Vein pattern-Scan)

Cette techniques est rarement employée seule, mais régulièrement en tant qu'apport de sécurité d'un autre moyen biométrique. Il s'agit de cartographier le réseau des veines d'une partie du corps. Ainsi, dans le cas de la détection de la forme d'une main, cette technique permet d'obtenir d'autres points de comparaison que les formes ou les distances.

21.4.3.1 Avantages

- Il permet d'obtenir facilement des points caractéristiques supplémentaires

21.4.3.2 Inconvénients

- Cette méthode n'est pas utilisable seule

21.4.3.3 Application

- Utilisation en complément d'une autre technique
- Applications identiques à celle du Hand-Scan

21.4.4 Le visage (Facial-Scan)

Le visage possède également de nombreuses caractéristiques pouvant déterminer une personne. On pourra citer la forme des yeux, du nez, de la bouche, ou encore le creusement des joues. L'une des premières difficultés était de tenir compte des éléments additionnels physionomiques tels que la présence de lunettes, d'une barbe, ou du maquillage. Aujourd'hui, c'est un peu moins vrai, les techniques ayant fortement évolué. Il faudra aussi prendre garde aux conditions d'éclairage qui peuvent provoquer des ombres.

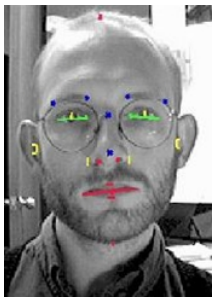

FIG. 21.8 – Extraction de caractéristiques d'un visage

Plusieurs techniques peuvent être utilisées telles que :

– Eigenface (MIT, 1987) : le visage est décomposé en plusieurs images mettant chacune en évidence une caractéristique du visage en question. Ces caractéristiques sont définies par une série de valeurs (vecteurs propres, valeurs propres et matrice de covariance) permettant d'établir un ensemble de statistiques en vue d'identifier les individus avec plus ou moins de précision.

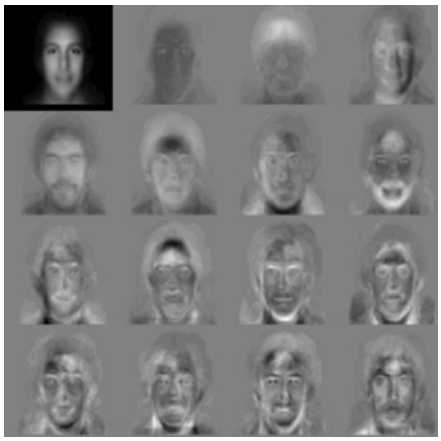

FIG. 21.9 – Facial-Scan par la méthode "Eigenface"

– Feature Analysis : basée sur la méthode Eigenface, elle calcule certaines distances entre éléments, les positions relatives de ces éléments, et semble donc plus souple pour l'identification. En effet, en cas d'inclinaison de la tête par exemple, la première méthode ne reconnaitra pas le visage, alors que le calcul des positions relatives permettra l'identification de l'individu.
– Etudes des expressions et postures : à l'aide des réseaux de neurones, ce système permet, à partir d'un nombre réduit de données sources, de déduire un nombre impressionnant d'images qui pourront par la suite être comparées à la saisie du visage de l'individu désirant être identifié. La figure 21.10 illustre les possibilités de cette technique.

FIG. 21.10 – Utilisation de réseaux de neuronnes en Facial-Scan

– Etude 3D : cela permet d'analyser la forme des éléments du visage tels que le creusement des joues, la profondeur des orbites, etc.
– Etude de la texture de la peau : avec une précision assez grande, cette technique permet d'en étudier certaines caractéristiques (pores, taches, défauts de peau, etc.)

21.4.4.1 Avantages

- Technique non invasive
- "user-friendly"
- Une des techniques les plus développées après les empreintes digitales
- Permet de sécuriser des organisations publiques de grande envergure

21.4.4.2 Inconvénients

- Selon les techniques : mauvaise adaptation au vieillissement, pilosité changeante, expression du visage, mouvement du visage, etc.

21.4.4.3 Applications

- Entrées de casino
- Hall d'aéroports

21.4.5 L'iris (Iris-Scan)

Il s'agit dans un premier temps de faire clairement la distinction entre iris et rétine (que nous évoquerons plus tard), ce qui est illustré à la figure 21.11.

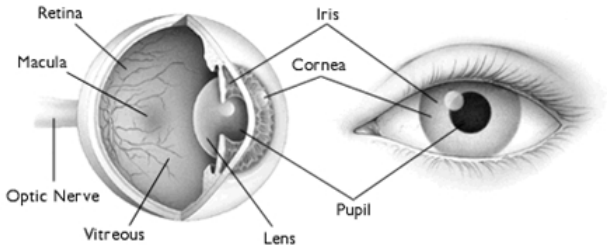

FIG. 21.11 – Vue détaillée de l'oeil humain

Chaque oeil est unique. Dans l'iris de l'oeil, il est possible de compter plus de 200 paramètres indépendants. Dès 1980, des opthalmologues ont remarqué que si la couleur de l'iris peut varier à travers le temps, il n'en est pas de même pour le motif (voir firgure 21.12) qui reste constant.

FIG. 21.12 – Motif de l'iris

Pour la capture, on procède à la prise de vue, souvent par infrarouge pour éviter la dilatation de la pupille. On réalise ensuite un applatissement de l'image, et après traitements mathématiques (ondelettes de Gabor), on obtient un code qui pourra prendre place dans une base de données. Le processus est illustré à la figure 21.13.

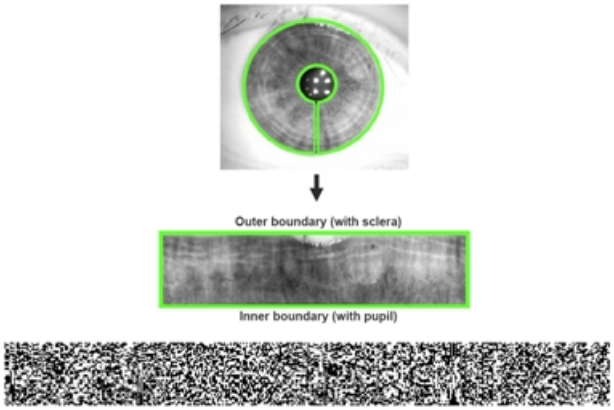

FIG. 21.13 – Processus d'enregistrement de l'iris [13]

Signalons aussi que cette mesure biométrique nécessite un nombre élevé de prises de vue. En effet, ici, non seulement la distance entre le capteur et l'oeil peut varier, mais également la position de la tête, l'ouverture de la paupière, l'éventuel maquillage des cils masquant une partie du motif, etc.

21.4.5.1 Avantages

- Grande quantité d'information présente dans l'iris
- Le capteur est moins exposé qu'un capteur tactile
- L'iris ne subit pas de modification à travers le temps

21.4.5.2 Inconvénients

- Le motif de l'iris peut être photographié en vue d'une usurpation. Tout dépend donc des mesures de sécurité prises (utilisation d'infra-rouge).
- Caractère plus intrusif que l'empreinte ou que la main.

21.4.5.3 Applications

- Identification dans les distributeurs de billets de banque.
- Contrôle d'accès aux locaux et machines.

21.4.6 La rétine (Retina-Scan)

Comme on le voit sur la figure 21.11, la rétine est composée en grande partie d'un ensemble de vaisseaux sanguins. La complexité du réseau sanguin ainsi constitué permet une identification unique. Cette technique est plus ancienne que la saisie de l'iris, mais bien moins utilisée.

Alors que l'aspect des vaisseaux change avec l'âge, leur position ne se modifie pas.

Pour effectuer la saisie, il faut éclairer le fond de l'oeil. On pourra alors opérer une cartographie du réseau et ainsi la stocker dans une base de données.

21.4.6.1 Avantages

- L'empreinte rétinienne est bien moins exposée aux blessures que les empreintes digitales
- Les T.F.R. et T.F.A. sont faibles
- On peut mesurer jusqu'à 400 points caractéristiques, contre "seulement" 30 à 40 pour une empreinte digitale

21.4.6.2 Inconvénients

- Souvent mal accepté par le public, en raison de l'aspect sensible de l'organe
- Mise en place assez coûteuse car le système n'est pas produit en masse

21.4.6.3 Applications

- Utilisation dans certains distributeurs de billets
- Controle d'accès aux locaux "haute sécurité"

21.5 Moyens biométriques comportementaux

21.5.1 La voix (Voice-Scan)

Chaque individu possède une voix unique caractérisée par une fréquence, une intensité et une tonalité. Même si deux voix peuvent sembler similaires pour l'oreille humaine, le traitement informatique permet de les isoler. Les prises sont réalisées à partir d'un simple microphone.

On distingue deux types de systèmes à reconnaissance vocale :

1. **Text Dependent System** : l'identification se fait sur des termes déterminés. Il existe 4 types de systèmes dépendants du texte :
   - *A texte suggéré* (text-prompted) : lors de chaque session d'identification, et pour chaque utilisateur, le système affiche un texte aléatoire.
   - *A traits phonétiques* (speech event dependent) : certains traits phonétiques doivent être mis en avant dans le texte prononcé par l'individu.
   - *A vocabulaire limité* (vocabulary dependent) : l'individu doit prononcer des mots issus d'un langage limité prédéfini (par exemple, une suite de chiffres)
   - *A texte personnalisé* (user-specific text dependent) : l'utilisateur possède un mot de passe, ou une passphrase.
2. **Text Independent System** (ou Free-Text) : l'utilisateur parle librement.

La sécurité est bien entendu plus forte dans le cas d'une identification par texte dépendant qu'indépendant.

La phase d'apprentissage nécessite souvent plusieurs phrases afin de prendre en compte les différentes intonations de la personne. Après l'acquisition, le signal est découpé en unités qui peuvent être soit des mots, soit des phonèmes.

Cette technique porte aussi le nom de AAL, pour Authentification Automatique du Locuteur.

21.5.1.1 Avantages

- Bien accepté par les utilisateurs, car il suffit de parler
- Il est facile de protéger le lecteur (par l'intermédiaire d'une grille, dans le cas d'un parlophone)
- C'est le seul moyen d'identifier les extrémités d'une communication téléphonique

21.5.1.2 Inconvénients

- Ce système est sensible à l'état physique de l'individu
  1. selon son état : stress, fatigue, maladie, ...
  2. selon ses émotions : joie, peine, ...
  3. selon son âge
- La fraude est possible par l'intermédiaire d'un enregistrement de bonnne qualité
- Le système est sensible aux bruits parasites (environnement, usure du micro)

21.5.1.3 Applications

- Utilisé dans la reconnaissance téléphonique
- Identification des propriétaires dans des immeubles à appartements, dont les lecteurs peuvent être protégés par une grille

21.5.2 La dynamique de frappe (Keystroke-Scan)

Ce type de reconnaissance identifie les personnes selon leur manière de taper sur un clavier. On parle dans ce cas précis de *solution biométrique essentiellement logicielle*, car elle consiste uniquement en un relevé de données basées sur la dynamique de frappe des utilisateurs. Par dynamique de frappe, on entend le temps utilisé entre deux frappes sur la même touche (**flight time**), le temps de pression sur chaque touche (**Dwell Time**), ou encore le temps pour taper un mot donné.

Lors de la phase d'apprentissage, il sera demandé à l'utilisateur de taper un certain nombre de fois un mot de passe ou une passphrase, et un algorithme sera chargé de moyenner les temps relevés. Par la suite, et selon le niveau de ressemblance demandé, les utilisateurs seront acceptés ou refusés.

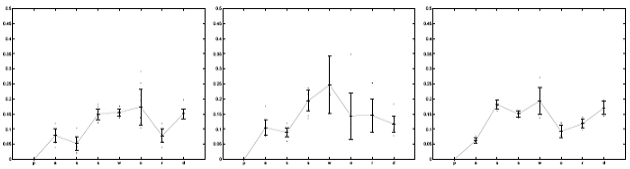

FIG. 21.14 – Moyenne obtenue pour la frappe du mot "password" par trois utilisateurs [14]

21.5.2.1 Avantages

- Pas de système hardware, seul le logiciel suffit, ce qui permet de réaliser des économies substancielles.
- Très pratique lorsque le nombre d'utilisateurs est élevé, de par sa simplicité de mise en place et d'apprentissage.
- T.F.A. inférieur à 0.5% si le mot de passe dépasse 8 caractères.

21.5.2.2 Inconvénient

- Il faut faire attention au clavier utilisé (distinction d'emplacement des touches en QWERTY et en AZERTY). Les vitesses de frappe changeant d'un type de clavier à l'autre, un profil d'identification sera nécessaire pour chaque configuration.

21.5.2.3 Applications

- Il est régulièrement utilisé en complément de sécurité (notamment pour des cartes à puces)

21.5.3 La signature dynamique (Signature-Scan)

Le principe de cette technique est d'identifier une personne à partir des parties fixes de sa signature. En effet, une signature, même si elle reste approximativement identique, possède des parties fixes et des parties variables. La différentiation de ces parties permettra l'identification. Celle-ci pourra également prendre d'autres paramètres en compte, tels que la pression exercée avec le stylo, la vitesse d'écriture, ou encore les accélérations.

Le système se compose principalement d'une tablette digitale et de son crayon. La phase d'apprentissage nécessite quelques signatures. Par la suite, le logiciel en extraira les caractéristiques, et tentera d'en calculer les variantes possibles et acceptables.

21.5.3.1 Avantages

- La sécurité contre la copie est plus forte que pour un simple mot de passe
- Faible coût du matériel nécessaire

21.5.3.2 Inconvénients

- Dégradation du matériel avec le temps
- L'écriture change au cours de la vie de l'individu.

21.5.3.3 Applications

- E-Commerce : Validation de la signature avant toute transaction
- Gestion banquaire : toutes les opérations sont signées
- Sécurisation des sessions sur ordinateur : au lieu d'un simple mot de passe, l'utilisateur doit soumettre sa signature, ce qui est beaucoup plus difficile à copier
- Verrouillage de fichiers : on protège les fichiers par signature, ce qui permettra de l'ouvrir, de le modifier, de l'imprimer, de façon sécurisée

21.6 Moyens biométriques expérimentaux

21.6.1 La thermographie

Dans ce système, une caméra thermique réalise une photographie infrarouge du visage. Apparait alors une répartition de la chaleur. Cette répartition est propre à chaque individu, y compris les vrais jumeaux. Ce système reste cependant encore très coûteux.

21.6.2 L'oreille

Peu répandue, cette technique consiste à comparer les empreintes d'oreille que peuvent laisser certains individus dans le cadre d'un délit. Cette méthode est uniquement utilisée par la police, mais est admissible devant une cour de justice.

21.6.3 L'ADN

Cette technique est la plus sûre car elle ne se base pas sur des points caractéristiques, mais sur l'entièreté des données. Il donnerait des T.F.R. et T.F.A. nuls (si on fait exceptions des vrais jumeaux monozygotes qui possèdent le même ADN). Cependant, sa mise en place pose énormément de problèmes en raison de son caractère "intrusif". En effet, le relevé de cet ADN permettrait la création d'un répertoire des utilisateurs, et ouvrirait la porte à de nombreuses dérives.

Il existe toutefois des BD réelles recensant l'ADN de détenus (Angleterre, France).

21.6.4 Autres

D'autres méthodes sont encore exploitables, telles que la dentition, l'odeur, les battements de coeur, la démarche ou encore les pores de la peau.

21.7 La biométrie multimodale

Dans le cas où un seul moyen biométrique est mesuré, on parle de biométrie unimodale. Lorsque ces moyens sont couplés, ou que plusieurs mesures distinctes sont réalisées, on parlera de biométrie multimodale. Les différents types de biométrie multimodale sont représentés à la figure 21.15

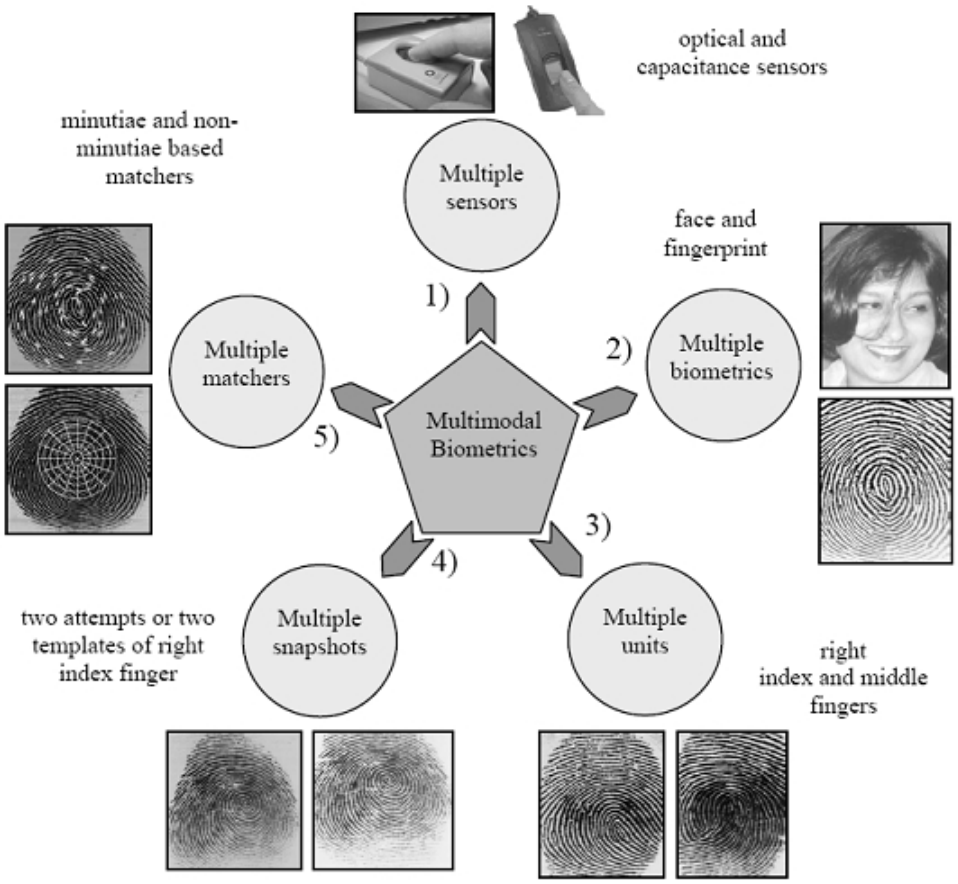

FIG. 21.15 – Biométrie multimodale [11]

Ce type de biométrie offre de nombreux avantages :

- Plus de fiabilité car les vérifications sont indépendantes (moins de faux positifs et faux négatifs)
- Plus de compatibilité : un moyen biométrique pris seul n'est jamais utilisable par 100% de la population.
- Plus de sécurité : la multiplication des moyens rend plus compliquée la tâche du pirate (plusieurs mesures à contourner). Il est aussi possible de vérifier le caractère "réel" de l'utilisateur en interagissant avec lui, par exemple en testant plusieurs doigts dans un ordre précis inconnu à l'avance.

21.8 Avantages, inconvénients et conclusions

Avantages	Inconvénients
Souvent non-invasif	Atteinte à la vie privée
Facile, sans effort	Sensible aux changements du quotidien
Généralement bien accepté	Coût
Evite les oublis…	Hygiène…
	Fiabilité (TFA, TFR)

Remarques importantes :

- Les vrais jumeaux sont aujourd'hui distinguables avec à peu près n'importe quel technique biométrique : rétine, iris et même la forme du visage. Signalons également que deux vrais jumeaux, même monozygotes, n'ont pas les mêmes empreintes digitales (elles ne dépendent pas de l'ADN qui lui est identique).
- Aucune technique ne permet d'identifier quelqu'un avec 100% de certitude (au sens strict), de même qu'aucun moyen n'est utilisable avec 100% de la population.

21.9 Ressources supplémentaires

http://www.biometricgroup.com/
http://biometrie.online.fr/
http://pagesperso-orange.fr/fingerchip/index.htm
http://www.facedetection.com/
http://www.minefi.gouv.fr/minefi/ministere/documentation/revuesdeweb/biometrie.htm
http://www.securiteinfo.com/conseils/biometrie.shtml
http://luc.bo.free.fr/SiteProfessionnel/RapportFinal.htm
http://big.chez.com/gipp/oraux/aal/index.html
http://www.techno-science.net/?onglet=articles&article=021&page=1

Chapitre 22

La Stéganographie

22.1 Définition

La stéganographie est l'art de "cacher" une information privée ou secrète dans un support, apparement anodin. Le support peut être de plusieurs types tels qu'un fichier texte, image, audio, ou vidéo, mais peut également être un système de fichier, ou un code source. La caractéristique principale est que le fichier doit sembler ne contenir aucune information sensible, i.e. ne renfermer aucune information secrète. Le support associé à son message porte le nom de "stégo-médium".

La stéganographie part du principe que la perception humaine n'est pas assez évoluée pour détecter les petites modifications introduites dans les images et sont destinées à renfermer un message. De plus, personne ne sait, à priori, qu'un fichier renferme un message stéganographié (en dehors des personnes visées par le message).

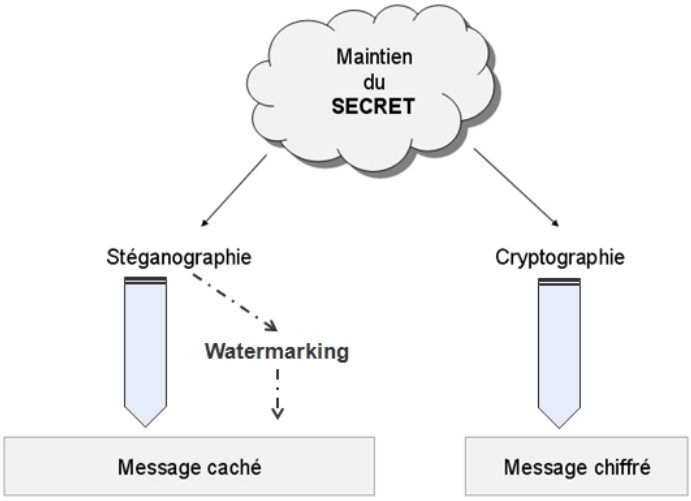

Fig. 22.1 – Stéganographie et concepts liés

On parle également de la stéganographie comme d'une possibilité pour établir un canal secret de communication (*cover channel*). En effet, par son intermédiaire, il est possible de communiquer avec d'autres individus sans se faire remarquer. Il serait alors possible par exemple, de passer à travers un firewall en se bornant aux protocoles et liens autorisés, mais en transmettant des messages cachés à l'intérieur de ceux-ci.

Trois critères existent pour mesurer l'efficacité d'un système cryptographique :

- L'imperceptibilité : comme mentionné ci-haut, la stéganographie repose sur le fait qu'une tierce personne ne peut percevoir la présence d'un message.
- La capacité : il s'agit du nombre de bits qu'il est possible de dissimuler dans le support.
- La robustesse : elle représente la manière avec laquelle le message dissimulé résiste aux modifications apportées au support.

22.1.1 Cryptographie et Stéganographie

Tout comme la cryptographie, le but principal de la stéganographie est de protéger une information. La différence intervient dans la manière avec laquelle chaque technique procède. La cryptographie va chiffrer l'information pour qu'elle ne soit pas compréhensible par un tiers, alors que la stéganographie va la rendre invisible[1].

On utilisera dès lors cette dernière notamment lorsque les techniques cryptographiques sont inapplicables, ou lorsqu'on souhaite garder secret le fait même de communiquer.

C'est en effet là le principal avantage de la stéganographie par rapport à la cryptographie : le fichier renfermant le message caché reste anodin. Personne ne peut se douter qu'il cache un message. Au contraire, un message chiffré annonce clairement qu'il renferme une information, puisque, par définition, il est chiffré.

Si on désire transmettre le message "*Jean arrêté à Paris. Mettre réseau en sommeil. Envoyer un OPR*[2]*.*", on obtiendra

- Par stéganographie (Stéganographie linguistique - Code de Barn) :

  Mon cher Pierre,

  J'espère que tu voudras bien m'excuser, mais j'ai eu tellement de travail à la maison que je n'ai pas pris le temps d'écrire aux amis. Cependant je t'envoie ce petit mot d'urgence pour te faire savoir que si tu veux des pneus, tu ferais bien de te dépêcher ; en effet :

  Hier, Jean est venu nous rendre visite, il descendait du train et s'est arrêté un moment chez nous pour bavarder et donner des nouvelles à mon père de son Paris. En principe, il doit rester quelques jours ici pour mettre en ordre ses affaires avant de repartir pour la capitale. A Paris, c'est calme, mais la veille il avait été dérangé en plein sommeil par les sirènes deux fois dans la nuit ! Ceci mis à part, il doit nous faire envoyer par un ami à lui des pneus neufs pour nos vélos. Il en a pour le moment, profitons-en ! A bientôt de tes nouvelles.

  P.-S. Nous irons au mariage de Simone et Henri, dimanche en quinze. Henri est un garçon sympathique qui a connu Simone l'an dernier chez Xavier, notre vieil ami. Il a deux ans de plus qu'elle et nous pensons que Simone va être très heureuse.

- Par chiffrement (Vigénère - Clé = Cryptographie) :

  LVYCT FXKPP YQWOV RIKSX JEPBM RUFKB XWRVN KVGIT LLDIF

Bien que le texte chiffré soit beaucoup plus court, il est cependant beaucoup plus explicite quant au fait qu'il renferme un secret.

On utilisera parfois la cryptographie en supplément de la stéganographie. Ainsi, même si le message est découvert, il faudra encore le déchiffrer, ce qui apporte un gain de sécurité non négligeable.

[1] Stéganographie vient du grec "steganos", signifiant "couvrir".
[2] OPérateur Radio.

22.1.2 Watermarking et Stéganographie

Le watermarking consiste à cacher une information dans le but de protéger le support contre la copie, contre toute modification de ce support ou dans un but d'identification. On considère souvent le watermarking comme la forme moderne de la stéganographie, ou en tout cas comme la principale application de sécurité reposant sur la stéganographie.

On trouve aujourd'hui régulièrement ce type de sécurité dans les images protégées par droits d'auteurs, dans certains supports audios et vidéos, ou encore sur des cds ou dvds. La détection du message permettra alors de distinguer un original d'une copie, de valider un accès, ou encore de tracer l'utilisation d'un média.

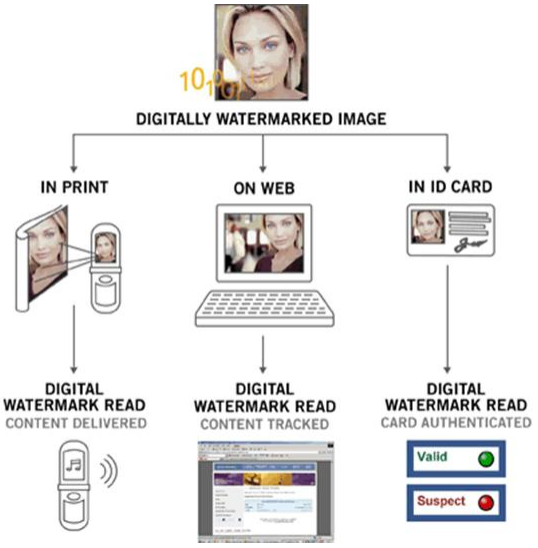

Fig. 22.2 – Quelques applications des techniques de watermarking
Source : http ://www.willamette.edu/wits/idc/mmcamp/watermarking.htm

Hormis l'utilisation qui en sera faite, une autre différence majeure existe entre les deux concepts :

1. Dans le cadre de la stéganographie, le message doit être caché, imperceptible.
2. Dans le cadre du watermarking, le message devra être impossible à enlever, impossible à modifier (mais peut éventuellement être visible).

Il est également fréquent de considérer la stéganographie comme une technique utilisée entre deux personnes (one-to-one), le watermarking étant préféré pour les communications one-to-many.

22.2 La stéganographie dans l'Histoire

La première utilisation de la stéganographie remonte à la Grèce Antique où on utilisait certains esclaves pour transmettre les messages. Ceux-ci étaient écrits sur les crânes des messagers, et passaient donc inaperçu lorsque les cheveux repoussaient. Une fois suffisamment longs, le messager pouvait être envoyé, avec l'ordre de se faire raser le crâne une fois arrivé à destination. Le principal désavantage de cette méthode était l'attente pour l'envoi d'un message.

Une autre technique était d'utiliser des tablettes de cire. Une fois la cire raclée, on gravait le message dans le bois de la tablette. Il suffisait ensuite d'y remettre de la cire, et le message était parfaitement caché.

En Chine ancienne, les messages étaient écrits sur de la soie, qui était ensuite roulée en boule, elle-même recouverte de cire. Un messager devait enfin avaler cette boule.

Vers 1500, l'abbé Trithème utilisa des louanges afin de faire passer des messages de manière anodine. Chaque lettre du message était associée à un groupe de mots, comme indiqué à la figure 22.3. Sa technique se rapproche d'une technique cryptographique, mais le concept de "recouvrement du message" en fait un exemple de stéganographie linguistique.

A	dans les cieux	N	en paradis
B	à tout jamais	O	toujours
C	un monde sans fin	P	dans la divinité
D	en une infinité	Q	dans la déité
E	à perpétuité	R	dans la félicité
F	sempiternel	S	dans son règne
G	durable	T	dans son royaume
H	sans cesse	U, V, W	dans la béatitude
I, J	irrévocablement	X	dans la magnificence
K	éternellement	Y	au trône
L	dans la gloire	Z	en toute éternité
M	dans la lumière		

FIG. 22.3 – Les Ave Maria de l'abbé Trithème

Au XVI^e siècle, Giovanni Porta, un scientifique italien, découvrit le moyen de cacher un message dans un oeuf dur : en écrivant sur la coquille avec une encre contenant une once d'alun par pinte de vinaigre. L'encre pénètre alors la coquille et le message se voit écrit sur le blanc de l'oeuf sans apparaitre sur la coquille. Pour lire le message, il suffit d'éplucher l'oeuf.

Enée le Tacticien, un historien de la Grèce Antique, imagina de percer de minuscules trous sous certaines lettres d'un texte anodin. Une fois toutes les lettres ainsi marquées relévées, on pouvait lire le message caché. Ces trous étaient invisibles pour une personne n'étant pas au courant de la méthode employée.

Durant la seconde guerre mondiale, on utilisa l'encre invisible. Cette encre, pouvant être créée notamment à partir de lait, de vinaigre, d'urine, de jus de citron ou encore de chlorure d'ammoniac, permettait d'écrire sur du papier sans pour autant afficher les caractères ainsi notés.

Différentes méthodes linguistiques existent également. Pour cacher un message dans un texte, on peut jouer sur l'espace entre les mots, la ponctuation, ou encore l'orthographe. A l'origine, c'est l'acrostiche qui permettait de cacher ces messages. L'acrostiche est un poème dont la première lettre de chaque vers compose un mot ou une phrase. Dès qu'un enfant commence à écrire, il pratique donc souvent l'acrostiche pour son premier cadeau de fête des mère ! Un des exemples les plus imposant fut sans doute le livre *Hypnorotomachia Poliphili* publié en 1499 par un anonyme. On découvre en effet dans cet ouvrage que si on regroupe la première lettre de chacun des chapitres, au nombre de 38, on peut recomposer la phrase *Poliam frater Franciscus Columna peramavit* ce qui signifie *Frère Francesco Colonna aime Polia passionnément.*

Mais il existe également des techniques plus évoluées où, pour pouvoir comprendre le message caché, il faut savoir quels mots ou quelles lettres lire. Par exemple, ci-dessous, un texte à première vue innocent envoyé par un espion allemand pendant la seconde guerre mondiale renfermait un message très important :

Apparently neutral's protest is thoroughly discounted and ignored. Isman hard it. Blockade issue affects pretext for embargo on byproducts, ejecting suets and vegetable oils.

Dans la cas présent, en prenant la deuxième lettre de chaque mot, on voit apparaitre le message suivant :

Pershing sails from NY June 1.

Parmi les exemples les plus célèbres illustrant le principe de stéganographie linguistique (ou littéraire), on trouvera encore l'échange de courier entre George Sand et Alfred de Musset.

Cher ami, Je suis toute émue de vous dire que j'ai
bien compris l'autre jour que vous aviez
toujours une envie folle de me faire
danser. Je garde le souvenir de votre
baiser et je voudrais bien que ce soit
une preuve que je puisse être aimée
par vous. Je suis prête à montrer mon
affection toute désintéressée et sans cal-
cul, et si vous voulez me voir ainsi
vous dévoiler, sans artifice, mon âme
toute nue, daignez me faire visite,
nous causerons et en amis franchement
je vous prouverai que je suis la femme
sincère, capable de vous offrir l'affection
la plus profonde, comme la plus étroite
amitié, en un mot : la meilleure épouse
dont vous puissiez rêver. Puisque votre
âme est libre, pensez que l'abandon où je
vis est bien long, bien dur et souvent bien
insupportable. Mon chagrin est trop
gros. Accourez bien vite et venez me le
faire oublier. À vous je veux me sou-
mettre entièrement.

Votre poupée

Quand je mets à vos pieds un éternel hommage,
Voulez-vous qu'un instant je change de visage ?
Vous avez capturé les sentiments d'un coeur
Que pour vous adorer forma le créateur.
Je vous chéris, amour, et ma plume en délire
Couche sur le papier ce que je n'ose dire.
Avec soin de mes vers lisez les premiers mots,
Vous saurez quel remède apporter à mes maux.

Alfred de Musset

Cette insigne faveur que votre coeur réclame
Nuit à ma renommée et répugne à mon âme.

George Sand

Une autre méthode ingénieuse fut inventée par Gaspar Schott (1608 - 1666). Le principe, illustré à la figure 22.4 était de coder le message selon des notes de musique. L'avantage du procédé était que le message apparaissait comme une partition musicale, et donc passait totalement inaperçu. Cependant, si la partition était jouée, il y avait très peu de chance pour que la mélodie soit agréable.

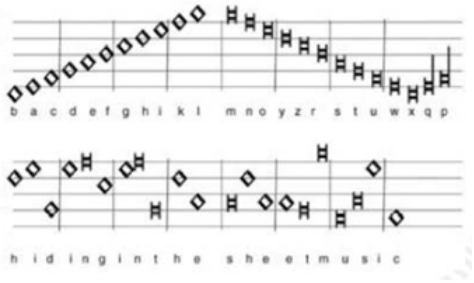

FIG. 22.4 – La stéganographie par partition musicale.

La ponctuation permit aussi d'identifier certains messages cachés. Au $XVII^e$ siècle, Sir John Trevanion fut arreté et emprisonné dans un chateau. Il reçut alors une lettre, que les gardiens avaient jugés sans danger. Lorsque John lut celle-ci, il détecta la présence suspecte de certaines virgules étrangement placées. Il repéra également qu'en prenant la troisième lettre de chaque mot suivant ces virgules, il pouvait former la phrase *Panel at east of chapel slides* ce qui signifiait *Le panneau à l'extrémité Est de la chapelle peut glisser.* C'est ainsi qu'il demanda un instant de recueillement dans la chapelle et s'évada.

Dans les années 1940, la puissance des techniques stéganographiques était si crainte que les Etats-Unis mirent en place un service de censure constitué d'environ 10.000 personnes, chargées d'étudier et de détecter les messages cachés. On interdit les parties d'échecs internationnales, les dessins d'enfant accompagnant les lettres pour les grands-parents, les diffusions de disques à la radio, ou encore les annonces pour chiens perdus.

La stéganographie reste également un sujet d'actualité. Après les incidents du 11 septembre 2001, beaucoup ont avancé l'hypothèse que des échanges entre terroristes avaient eu lieu afin de préparer les attaques.

Remarque : Un autre "phénomène stéganographique" est apparu avec le livre "The Bible Code" de Michael Drosdin. Ce dernier prétendait que la Bible renfermait des messages cachés, tels que l'assassinat d'Yitzhak Rabin. Après plusieurs expériences, il s'avéra que ce "code" n'était rien d'autre que le fruit du hasard. Ainsi, on découvrit notamment l'assasinat de JFK dans Moby Dick (fig. **??**), ou encore le décès accidentel de la princesse Diana.

L'explication prend sa source dans un théorème, portant le nom de Théorème de Borel. Celui-ci indique que si on prend un nombre suffisant de combinaisons de lettres, il est possible de retrouver à peu près n'importe quel sujet ayant trait au passé ou au futur. Ainsi, sur un texte d'environ 1000 lettres, il est possible de trouver plusieurs millions de mots, selon leur agencement par la méthode appelée ELS (Equidistant Letter Sequence).

22.3 Principes

Aujourd'hui, les messages cachés se transmettent de manière digitale et non plus par des techniques manuelles (bien que le système du message écrit sur le crâne d'individus semblent encore avoir été utilisé au début du XX^e siècle par des espions allemands). Le schéma du processus stéganographique est illustré à la figure 22.5.

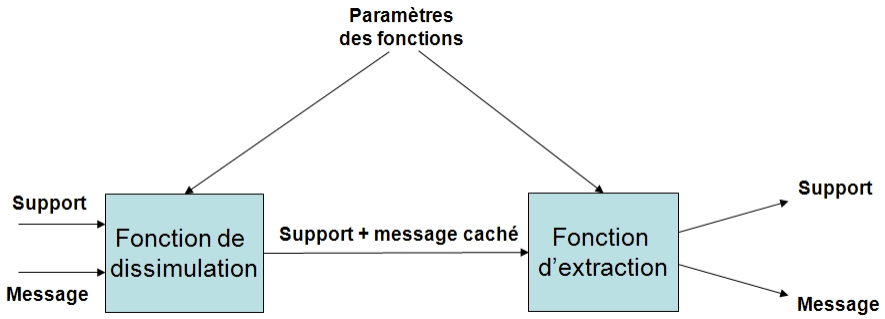

FIG. 22.5 – Principe d'un système stéganographique.

Il est à remarquer que plus le message à dissimuler est long, plus la modification du fichier support sera importante, et donc plus la détection sera facile.

On distingue 3 utilisations de la stéganographie :
- la stéganographie simple : seul l'algorithme à utiliser doit être connu des deux parties communicantes.
- la stéganographie à clé secrète : les deux parties utilisent une clé commune (ce sont les "paramètres" de la figure 22.5) afin d'introduire et d'extraire le message du support.
- la stéganographie à clé publique : il est possible d'utiliser un système à clé publique/privée (utilisable en paramètre), comme dans le cadre cryptographique.

22.3.1 Première Technique : la substitution

La substitution consiste à utiliser des zones inutilisées ou de faibles importances dans un fichier. Une méthode employée dans ce contexte porte le nom de LSB (Least Significant Bit).

Le LSB utilise comme son nom l'indique les bit de poids le plus faible pour transmettre un message. Si on regarde le codage d'une image RGB, on remarque que chaque pixel est codé selon 3 octets (un octet par couleur). En modifiant le bit de poids le plus faible, l'image sera effectivement modifiée, mais les nuances résultantes seront tellement faibles que l'oeil humain ne pourra que très rarement s'en rendre compte[3].

Le message caché pourra être d'autant plus important que l'on utilisera un grand nombre de bits pour le cacher. En contrepartie, utiliser plusieurs bits modifiera plus fortement l'image de base, et donc rendra la présence d'un message beaucoup plus détectable.
L'avantage du LSB est que la taille du fichier n'est pas modifiée, puisque le message est encodé dans les parties peu ou pas utilisées du fichier. C'est également une méthode rapide et facile à mettre en oeuvre. Cependant, le LSB possède un désavantage de taille, à savoir la perte du message lorsque des changements importants ont lieu sur le support, comme par exemple une rotation, ou un redimmensionnement de l'image.

Une amélioration du LSB consiste à introduire un paramètre aléatoire permettant de distribuer les bits de poids faible utilité. Ainsi, les modifications n'auront pas lieu "uniquement" dans les premiers octets de l'image, mais seront au contraire répartis aléatoirement dans l'entièreté de l'image.

[3]Si la modification avait eu lieu sur le bit de poids le plus fort, les différences auraient été très marquées, et on aurait pu en déduire la présence d'un message caché.

22.3.2 Deuxième Technique : l'injection

L'injection repose sur le fait d'inclure le message directement dans le fichier support. L'inconvénient de cette approche est que la taille du fichier support est modifiée. La présence d'un message est alors plus facilement détectable, bien que ce problème ne se pose que lorsque des regards indiscrets possède une copie du fichier support original.

22.3.3 Troisième Technique : la création d'un nouveau fichier

Contrairement aux deux autres méthodes qui utilisent un fichier support comme base, c'est le message qui servira ici de base. A partir de celui-ci, on construira une "enveloppe" qui le renfermera. La lettre de George Sand à Alfred de Musset peut être vu comme un exemple de cette technique. En effet, pour transmettre son message à Alfred de Musset, elle a créé une lettre anodine *autour* de son message.

22.4 Les types de support

Les techniques de stéganographie peuvent s'utiliser sur pratiquement tous les types de support. Les plus communs sont explicités ci-dessous.

22.4.1 Les images

La technique du LSB a déjà été explicitée dans une section précédente. On trouvera principalement 2 types d'images : les images 24 bits et 8 bits. On remarquera cependant qu'il est plus facile de cacher un message dans une image 24 bits en raison du nombre plus important d'octets.

22.4.2 L'audio

Il existe 4 techniques principales permettant d'intégrer des données dans un flux audio :

- **LBE (Low Bit Encoding)** : les données sont stockées dans le flux audio d'une manière semblable à celle utilisée dans le cas du LSB pour les images.
- **SSE (Spread Spectrum Encoding)** : cette méthode ajoute un bruit aléatoire au signal sonore. Le message est ensuite dissimulé dans ce bruit additif.
- **EDH (Echo Data Hiding)** : dans la plupart des cas, un echo dans une mélodie permet de grandement améliorer la qualité sonore. Plus le temps séparant le son original et son echo est réduit, moins l'oreille humaine peut le détecter. La durée de ce délai peut alors être utilisée pour dissimuler des bits d'information. Ces délais une fois mesurés, il est alors possible de recréer le message initial.

22.4.3 La vidéo

Plusieurs techniques existent, mais sont pour la plupart des améliorations ou modifications de la technique de la transformée en cosinus discret (DCT). Celle-ci utilise la quantification des parties les moins importantes des images (arrondies aux valeurs supérieures par exemple). L'oeil étant relativement peu sensible aux hautes fréquences, la DCT pourra appliquer des modifications plus importantes dans cette plage, sans pour autant créer de modifications visibles dans l'image. Le message est alors injecté dans l'image en jouant sur les facteurs d'arrondi. Une compression sans perte est ensuite appliquée.

22.4.4 Autres supports

La plupart des fichiers existants peuvent être utilisés pour transmettre un message de manière cachée. Cela peut aller de la simple page HTML à des formes plus évoluées telles que l'insertion d'octets dans une portion inutilisée de code assembleur. La stéganographie ne se cantonne donc pas aux simples images

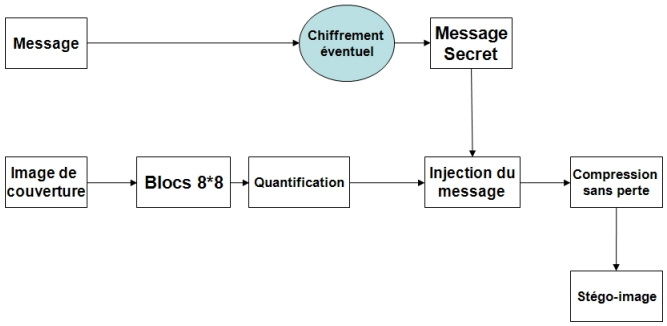

FIG. 22.6 – Processus d'insertion du message par DCT

ou fichiers musicaux.

L'exemple de code assembleur suivant illustre le principe de l'insertion d'octets dans les trous du code.

Soit le célèbre code source suivant :

```
/* hello.c */
main() {
    printf("Bonjour\n");
}
```

Sa version assembleur, à la figure 22.7, est quant à elle plus longue :

```
Dump of assembler code for function main:
0x80483c8 <main>:       push   %ebp
0x80483c9 <main+1>:     mov    %esp,%ebp
0x80483cb <main+3>:     push   $0x8048430
0x80483d0 <main+8>:     call   0x8048308 <printf>
0x80483d5 <main+13>:    add    $0x4,%esp
0x80483d8 <main+16>:    leave
0x80483d9 <main+17>:    ret
0x80483da <main+18>:    nop
0x80483db <main+19>:    nop
0x80483dc <main+20>:    nop
0x80483dd <main+21>:    nop
0x80483de <main+22>:    nop
0x80483df <main+23>:    nop
End of assembler dump.
```

FIG. 22.7 – Code assembleur du code source hello.c

L'insertion d'octets sera rendue possible grâce à la présence d'instructions inutiles (NOP, NO operation). On peut dès lors remplacer ce code "mort" par des octets de notre choix, sans que cela n'affecte le déroulement du programme, pour autant que l'on soit certain de la non-utilité de cette partie du code. Dans le cas contraire, il est très probable que le programme plante en tentant d'exécuter ces instructions.

Il existe également des techniques permettant de cacher des informations sur un disque dur en utilisant des blocs non utilisés et donc non référencés dans la table d'allocations des fichiers. Ainsi, seul l'utilisateur ayant connaissance de ces blocs pourra lire les informations cachées. Un amélioration consiste à placer l'information cachée de manière redondante. En effet, le système d'exploitation n'ayant pas connaissance du message dissimulé, il pourrait écraser certains blocs le contenant en partie.

22.5 La stéganalyse

La stéganalyse consiste à identifier la présence d'un message. Seule cette présence doit être déterminée. Dans un second temps, le message lui-même pourra être retrouvé, mais ce n'est pas l'objectif principal de la stéganalyse.

Il existe deux catégories d'attaques :

1. Les attaques passives : elles visent à identifier la présence d'un message, dans le but secondaire de pouvoir le reconstituer par la suite. Ces attaques peuvent prendre plusieurs formes :
   - la lecture ou l'écoute du fichier,
   - la comparaison avec le fichier original (si il est disponible),
   - certaines attaques statistiques (attaques sur le LSB),
   - la détection des signatures des logiciels utilisés (étude du code hexadécimal)
2. Les attaques actives : elles consistent à détruire le message caché, sans prêter attention à ce qu'il signifie. L'objectif est de supprimer le message, ou en tout cas, à le rendre inutilisable. Cette destruction aura souvent lieu par l'intermédiaire de modifications du support (redimensionnement (image, vidéo), filtrage (sons), compression, ...).

22.6 Les Anamorphoses

Une anamorphose est une image déformée qui retrouve ses proportions d'origine quand on la regarde sous un certain angle ou réfléchie dans un miroir adapté. Les miroirs utilisés peuvent avoir de nombreuses formes dont les plus répandues sont les miroirs coniques, cylindriques, ou encore spériques.

On trouve les premiers exemples au début du 17ème siècle, mais cette technique est toujours régulièrement utilisée de nos jours dans la vie courante. On peut notamment citer les marquages routiers ou les publicités marquées au sol dans les grands prix de F1 par exemple. Si on se tient à proximité de l'image, elle apparait déformée, mais une fois une certaine distance respectée, ou selon un angle de vue précis, l'image apparait "normalement" (comme on le voit sur la figure 22.8).

FIG. 22.8 – Anamorphose sur mur

Cependant, alors que l'anamorphose n'est rien d'autre qu'une représentation particulière d'un dessin (mais reposant tout de même sur une solide base mathématique), certains ont trouvé en cette technique, une méthode originale pour transmettre une image tout en la cachant dans une autre. Il ne s'agit pas à proprement parler d'une technique de stéganographie (puisqu'il n'y a pas d'écriture), mais l'idée de dissimulation reste tout de même assez proche dans certains cas. Quelques exemples sont présentés ci-dessous.

FIG. 22.9 – Anamorphose par miroir conique.

FIG. 22.10 – Anamorphoses d'Istvan Orosz

22.7 Ressources supplémentaires

http://www.apprendre-en-ligne.net/crypto/stegano/barncode.html
http://www.echu.org/articles/securite/RapportStegano.pdf
http://www.fbi.gov/hq/lab/fsc/backissu/july2004/research/2004_03_research01.htm
http://neuro.caltech.edu/~seckel/mod/
http://www.anamorphosis.com/

Chapitre 23

Conclusions

Il faut protéger intelligemment, c'est-à-dire étudier

- **Ce qu**'il faut protéger
- **Comment** le protéger
  1. Assurer un **niveau adéquat** de sécurité
     (a) Ne pas surprotéger ce qui n'en vaut pas/plus la peine
     (b) Protéger efficacement ce qui en vaut la peine
  2. Placer la sécurité **au bon endroit**.
- Ne jamais sous-estimer le facteur humain

Mais c'est aussi

- 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 antivirus, sécurité du SGBD utilisé, etc.)
- Ne pas se reposer sur une sécurité jugée bonne à un moment donné

La sécurité est omniprésente et nécessaire à tous les niveaux d'utilisation de l'information, de sa création à sa destruction.

Et enfin, pour rappel :

La sécurité est un compromis entre efficacité et convivialité

Chapitre 24

Annexe 1 : Rappels mathématiques

24.1 Entropie

On dit qu'une variable aléatoire X renseigne totalement sur le déroulement de l'expérience E si

$$H(X) = H(E)$$

Exemples :

- Jours de la semaine : 3 bits d'information → 000 = dimanche, 001 = lundi, ..., 110 = samedi
- Genre : 1 bit d'information → 1 = masculin, 0 = féminin
- On lance un dé non pipé et on considère les 3 variables aléatoires suivantes :
  X_1 qui vaut 0 si le nombre tiré est pair, 1 s'il est impair.
  X_2 qui vaut 0 si le nombre tiré est 1 ou 2, 1 si le nombre tiré est 3 ou 4, 2 si le nombre tiré est 5 ou 6.
  X_3 qui vaut le nombre tiré.
  Il est intuitivement clair que la connaissance de X_3 renseigne plus sur le déroulement de l'épreuve aléatoire que la connaissance de X_2, qui elle-même renseigne plus que celle de X_1
  On a
  $$H(X_1) = \log\ 2 < H(X_2) = \log\ 3 < H(X_3) = \log\ 6$$
- On vous présente 10 cartons sur lesquels sont inscrits sur la face cachée un nombre (tous les nombres sont différents).
  Vous pouvez poser des questions du type : "Est-ce que le nombre sur ce carton est plus élevé que sur celui-là ?".
  Vous payez un franc par question, et vous en recevez 15 lorsque vous savez réordonner les cartons. Acceptez-vous de jouer ?

  Il y a 10 ! façons d'ordonner les cartons. L'incertitude sur l'ordre dans lequel ils sont rangés vaut donc $\log_2(10!) = 21,8$ bits environ. Chaque réponse apporte au plus un bit d'information. Il faudra, du point de vue de la théorie de l'information, 22 questions en moyenne pour reconstituer l'ordre. Il ne faut pas jouer !

24.1.1 Entropie - Approche ponctuelle

On définit l'entropie par :

$$h(x) = \log[1/p(x)] = -\log p(x)$$

Dès lors, un événement certain apporte une information nulle ($h(1) = 0$).

24.1.1.1 Deux évènements indépendants :

Si x et y sont 2 événements indépendants :

$$p(x,y) = p(x).p(y)$$

$$\begin{aligned} h(x,y) &= \log[1/p(x,y)] \\ &= -\log p(x,y) \end{aligned}$$

ou encore

$$\begin{aligned} h(x,y) &= \log[1/(p(x).p(y))] \\ &= \log[1/p(x)] + \log[1/p(y)] \\ &= h(x) + h(y) \end{aligned}$$

h(x) ne dépend pas de la valeur de x mais seulement de la probabilité qui lui est associée.

24.1.1.2 Deux évènements consécutifs :

La quantité d'information associée à x conditionnelle à la réalisation de y est donnée par :

$$h(x|y) = -\log p(x|y)$$

Or, par Bayes :

$$p(x,y) = p(y|x).p(x) = p(x|y).p(y)$$

Donc,

$$\begin{aligned} h(x,y) &= -\log(p(x,y)) \\ &= -\log(p(y|x).p(x)) \\ &= -\log(p(y|x)) - \log(p(x)) \\ &= h(y|x) + h(x) \end{aligned}$$

24.1.1.3 Information mutuelle :

Elle détermine la quantité d'information que la connaissance d'une des variables apporte sur l'autre. Il vient

$$\begin{aligned} i(x;y) &= h(x) - h(x|y) \\ &= \log(p(x|y)) + \log(1/p(x)) \\ &= \log(p(x|y)/p(x) \\ &= \log(p(x,y)/(p(x).p(y)) \\ &= i(y;x) \end{aligned}$$

24.1.2 Entropie - Approche générale

Soient 2 variables aléatoires X et Y à valeurs respectives dans $(x_1, ..., x_n)$ et dans $(y_1, ..., y_n)$. Il vient :

$$H(X) = \sum_i p(x_i) \log(1/p(x_i))$$

On définit l'entropie conjointe par :

$$H(X,Y) = -\sum_i \sum_j p(x_i, y_j) \log(p(x_i, y_j))$$

Donc,

$$\begin{aligned} H(X,Y) &= -\sum_i \sum_j p(x_i, y_j) \log(p(x_i|y_j).p(y_j)) \\ &= -\sum_i \sum_j p(x_i, y_j)[\log(p(x_i|y_j)) + \log(p(y_j))] \\ &= -\sum_i \sum_j p(x_i, y_j) \log(p(x_i|y_j)) - \sum_i \sum_j p(x_i, y_j) \log(p(y_j)) \\ &= H(X|Y) + H(Y) = H(Y|X) + H(X) \end{aligned}$$

Remarques :
- On suppose les choix successifs des x_i indépendants.
- Le remplacement d'un symbole par un autre ne change pas la valeur de H (probabilités équidistribuées).

H(X|Y) est l'incertitude qu'il reste sur X lorsque Y est connu. L'information apportée par les 2 variables X et Y vaut donc l'information apportée par Y seule plus l'information apportée par X connaissant déjà la valeur de Y.

24.1.2.1 Information mutuelle moyenne :

On sait que

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Mais aussi que

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

Donc,

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

Remarque : Le maximum de H(X) est atteint pour n fixé, lorsque $p_i = \frac{1}{n} \forall i$.

24.2 Distance d'unicité

Le nombre de clés différentes traduisant un message chiffré en texte clair intelligible est[1] :

$$2^{H(K)-nD} - 1$$

24.3 Arithmétique modulaire

En cryptographie on fait un usage intensif de l'arithmétique modulo n. Celle-ci permet de restreindre la taille de tous les résultats intermédiaires et de la valeur finale. C'est indispensable, par exemple, pour le calcul de logarithmes discrets ou de racines carrées qui sont des opérations très coûteuses en ressources de calcul.

[1] Pour une démonstration détaillée, voir : `http ://www-ee.stanford.edu/ hellman/publications/25.pdf`

24.3.1 Diviseurs

Soient a, b et $m \in N$. $b(\neq 0)$ divise a si

$$a = mb$$

On dit que b est un diviseur de a.
Exemple : les diviseurs de 24 sont 1,2,3,4,6,8,12,24.

24.3.1.1 Propriétés :

- Si $a|1$ alors $a = \pm 1$
- Si $a|b$ et $b|a$ alors $a = \pm b$
- Tout b différent de 0 divise 0
- Si $b|g$ et $b|h$ alors $b|(mg + nh)$ pour m et n arbitraires
- Si $a = 0 \bmod n$ alors $n|a$

24.3.1.2 Quelques critères de divisibilité :

- n est divisible par 2 s'il se termine par 0,2,4,6,8.
- n est divisible par 3 si la somme de ses chiffres est divisible par 3.
- n est divisible par 4 si ses deux derniers chiffres forment un multiple de 4 (ex : 256628).
- n est divisible par 5 s'il se termine par 0 ou 5.
- n est divisible par 8 si ses 3 derniers chiffres forment un multiple de 8 (ex : 176072).
- n est divisible par 9 si la somme de ses chiffres est un multiple de 9 (ex : 37521=3+7+5+2+1=18=2*9).
- n est divisible par 11 si la différence (1^{er}chiffre + $3^{\text{ième}}$chiffre + $5^{\text{ième}}$chiffre + ...) - ($2^{\text{ième}}$chiffre + $4^{\text{ième}}$chiffre + $6^{\text{ième}}$chiffre + ...) est divisible par 11. Par exemple, 1485 est divisible par 11, car (1+8)-(4+5)=0 est divisible par 11.

24.3.2 Congruence

Soit n, un entier non nul (dans Z), et a, b des entiers. a et b sont dits congruents modulo n si

$$(a \bmod n) = (b \bmod n)$$

ce qui s'écrit $a = b \bmod n$.

Exemples : $73 = 4 \bmod 23$, $21 = 1 \bmod 10$

Deux entiers a et b sont égaux (ou congrus) modulo n si $n|a - b$.

24.3.2.1 Propriétés :

1. $a = b \bmod nssin|a - b$
2. $a = b \bmod n \longleftrightarrow ca = cb \bmod (cn)$
3. $a = b \bmod n \longleftrightarrow ac = bc \bmod n$
4. $a = b \bmod m \longleftrightarrow b = a \bmod m$
5. $a = b \bmod netb = c \bmod n \rightarrow a = c \bmod n$
6. $((a \bmod n) + (b \bmod n)) \bmod n = (a + b) \bmod n$
7. $((a \bmod n) - (b \bmod n)) \bmod n = (a - b) \bmod n$
8. $((a \bmod n) * (b \bmod n)) \bmod n = (a * b) \bmod n$

Propriété 1 :
$23 = 8(\text{mod}5)$ *car* $23 - 8 = 15 = 5 * 3$
$-11 = 5(\text{mod}8)$ *car* $-11 - 5 = -16 = 8 * (-2)$

Propriété 6 :
$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = (3 + 7) \bmod 8 = 2$
$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$

Propriété 7 :
$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = (-4) \bmod 8 = 4$
$(11 - 15) \bmod 8 = (-4) \bmod 8 = 4$

Propriété 8 :
$[(11 \bmod 8) * (15 \bmod 8)] \bmod 8 = (3 * 7) \bmod 8 = 5$
$(11 * 15) \bmod 8 = 165 \bmod 8 = 5$

Il existe une autre technique pour la propriété n°8 : la décomposition par facteurs plus simples.

Exemple : Pour trouver 117 mod 13, on peut procéder comme suit
$11^2 = 121 = 4 \bmod 13$
$11^4 = (11^2)^2 = 4^2 = 3 \bmod 13$
$11^7 = 11 * 11^4 * 11^2 = (11 * 3 * 4) = 132 = 2 \bmod 13$

24.3.3 L'ensemble Z_n

Soit Z_n l'ensemble des entiers, $Z_n = \{0, 1, ..., (n-1)\}$. Z_n représente l'ensemble des résidus ou classes de résidu modulo n (chaque entier dans Z_n est une classe de résidu). On peut écrire ces classes $[0], [1], ..., [n-1]$ où $[r] = \{a : a \in Z \; et \; a = r \bmod n\}$

Exemple :
Classes de résidu modulo 4 :

0 = {..., -8, -4,0,4,8,...}
1 = {...,-7,-3,1,5,9,...}
2 = {...,-6,-2,2,6,10,...}
3 = {...,-5,-1,3,7,11,...}

De tous les nombres entiers dans une classe de résidu, le plus petit nombre entier non négatif est celui habituellement utilisé pour représenter la classe de résidu. Trouver le plus petit nombre entier non négatif pour lequel k est congruent modulo n s'appelle la réduction de k modulo n.

24.3.4 Le PGCD

Il s'agit du Plus Grand Commun Diviseur (c) de a et b et tel que c est un diviseur de a et de b. Par cette définition, il vient que tout diviseur de a et b sera un diviseur de c.

- $\gcd(a, b) = max[k : k|a \text{ et } k|b]$
- $\gcd(a, b) = \gcd(|a|, |b|)$
- $\gcd(a, 0) = |a|$

Property	Expression
Commutative laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Associative laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive laws	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$ $[w + (x \times y)] \bmod n = [(w + x) \times (w + y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive inverse ($-w$)	For each $w \in Z_n$, there exists a z such that $w + z \equiv 0 \bmod n$

FIG. 24.1 – Propriétés remarquables de l'ensemble Z_n

24.3.5 Nombres premiers

Théorème 7 (Nombres premiers) *Tout nombre naturel $n \geq 1$ peut s'écrire comme un produit de nombres premiers, et cette représentation est unique, à part l'ordre dans lequel les facteurs premiers sont disposés.*

□

On dit qu'un nombre est premier si les deux seuls diviseurs positifs qu'il admet sont 1 et lui-même. Deux entiers sont relativement premiers si leur unique facteur commun positif est 1. On note lorsque a et c sont relativement premiers :

$$(a, c) = 1 \; ou \; \gcd(a, c) = 1$$

24.3.6 Autres propriétés de Z_n

24.3.6.1 Propriété de l'addition

$$(a + b) = (a + c) \bmod n) b = c \bmod n$$

Exemple : $(5 + 23) = (5 + 7) \bmod 8 \rightarrow 23 = 7 \bmod 8$

24.3.6.2 Propriété de la multiplication

$$Si \; (a, n) = 1 \; alors \; (a * b) = (a * c) \bmod n) b = c \bmod n$$

La *condition* doit absolument être vérifiée. En effet, si par exemple $(a = 6, n = 8) \neq 1 \rightarrow 6 * 3 = 2 \bmod 8$ et $6 * 7 = 2 \bmod 8$ or $3 \neq 7 \bmod 8$.

Explications : Avec a = 6 et n = 8, on obtient le résultat de la figure 24.2.

$\mathbb{Z}_8$	0	1	2	3	4	5	6	7
Multiplié par 6	0	6	12	18	24	30	36	42
résidus	0	6	4	2	0	6	4	2

FIG. 24.2 – Ensemble incomplet de résidus

et il n'y a pas d'ensemble complet de résidus. On a plusieurs résidus identiques (plusieurs nombres donnent le même résidu). Il n'y a donc pas d'inverse unique.

Comme on n'obtient pas un ensemble complet de résidus quand on multiplie par 6, plus de un entier de Z_8 donne le même résidu. Ainsi, $6 * 0 \mod 8 = 6 * 4 \mod 8, 6 * 1 \mod 8 = 6 * 5 \mod 8$, etc.

Cependant, avec a = 5 et n = 8, on le résultat de la figure 24.3

$\mathbb{Z}_8$	0	1	2	3	4	5	6	7
Multiplié par 5	0	5	10	15	20	25	30	35
résidus	0	5	2	7	4	1	6	3

FIG. 24.3 – Ensemble complet de résidus

La ligne des résidus contient tous les entiers de Z_8 dans le désordre.
Un entier a donc un inverse multiplicatif dans Z_n si cet entier est relativement premier à n.

24.4 Algorithme d'Euclide

24.4.1 Algorithme d'Euclide simple

L'objectif est de permettre de déterminer le PGCD de deux nombres entiers positifs sans avoir besoin de faire leur décomposition en facteurs premiers.

Théorème 8 (Principe d'Euclide) *Si a et $b \in N$ et $a \geq b$, si $\forall b > 0, a = r \mod b$, alors*

$$\gcd(a, b) = \gcd(b, r).$$

□

Tout repose ici sur le fait que si a et $b \in N$, $a \geq 0$ et $b > 0$, alors

$$\gcd(a, b) = \gcd(b, a \mod b)$$

Exemple : $\gcd(55, 22) = \gcd(22, 55 \mod 22) = \gcd(22, 11) = 11$.

Algorithme d'Euclide $(a > b > 0)$:

1. $A \longleftarrow a$; $B \longleftarrow b$
2. $IF\ (B = 0)\ RETURN\ A = \gcd(a, b)$
3. $R = A \mod B$
4. $A \longleftarrow B$
5. $B \longleftarrow R$
6. $GOTO\ 2$

24.4.1.1 Exemple :

Soit le calcul de pgcd(1970,1066). Il vient

$$
\begin{aligned}
1970 &= 1 * 1066 + 904 \;\; \gcd(1066, 904) \\
1066 &= 1 * 904 + 162 \;\; \gcd(904, 162) \\
904 &= 5 * 162 + 94 \;\; \gcd(162, 94) \\
162 &= 1 * 94 + 68 \;\; \gcd(94, 68) \\
94 &= 1 * 68 + 26 \;\; \gcd(68, 26) \\
68 &= 2 * 26 + 16 \;\; \gcd(26, 16) \\
26 &= 1 * 16 + 10 \;\; \gcd(16, 10) \\
10 &= 1 * 6 + 4 \;\; \gcd(6, 4) \\
6 &= 1 * 4 + 2 \;\; \gcd(4, 2) \\
4 &= 2 * 2 + 0 \;\; \gcd(2, 0)
\end{aligned}
$$

24.4.2 Algorithme d'Euclide étendu

Algorithme d'Euclide étendu ($m > b > 0$) :

1. $(A_1, A_2, A_3) \longleftarrow (1, 0, m)$
2. $(B_1, B_2, B_3) \longleftarrow (0, 1, b)$
3. $IF\ (B_3 = 0)\ RETURN\ A_3 = \gcd(m, b)$
4. $IF\ (B_3 = 1)\ RETURN\ B_3 = \gcd(m, b)$
5. $Q = \lfloor A_3 / B_3 \rfloor$
6. $(T_1, T_2, T_3) \longleftarrow (A_1 - Q.B_1, A_2 - Q.B_2, A_3 - Q.B_3)$
7. $(A_1, A_2, A_3) \longleftarrow (B_1, B_2, B_3)$
8. $(B_1, B_2, B_3) \longleftarrow (T_1, T_2, T_3)$
9. $GOTO\ \ 2$

Remarques
- $\lfloor\ \rfloor$ représente le nombre directement inférieur ou égal au nombre entre crochets
- Si la condition du point 3 est remplie, alors b n'a pas d'inverse modulo m
- Si la condition du point 4 est remplie, alors le pgcd vaut 1 et $b^{-1} \mod m = B_2$

24.4.2.1 Exemple d'exécution de l'algorithme :

Soit $\gcd(550, 1759) = 1$, l'inverse multiplicatif de 550 est 355. En effet, $(550 * 355) = 1 \mod 1759$

Q	A1	A2	A3	B1	B2	B3
—	1	0	1759	0	1	550
3	0	1	550	1	–3	109
5	1	–3	109	–5	16	5
21	–5	16	5	106	–339	4
1	106	–339	4	–111	355	1

FIG. 24.4 – Exemple d'exécution d'Euclide étendu

24.4.2.2 Exemple d'exécution manuelle :

Soit le problème suivant : trouver $(43)^{-1} \bmod 26 [= (17)^{-1} \bmod 26]$. Il vient

$$9 = 26 - 1 * 17$$
$$8 = 17 - 1 * 9 \rightarrow 17 - 1 * (26 - 1 * 17) = 2 * 17 - 1 * 26$$
$$1 = 9 - 1 * 8 \rightarrow 26 - 1 * 17 - 1 * (2 * 17 - 1 * 26) = 2 * 26 - 3 * 17$$

Donc, $(-3) \bmod 26 = 23 \bmod 26$: l'inverse modulo 26 de 43 est 23.

Chapitre 25

Annexe 2 : Concours AES

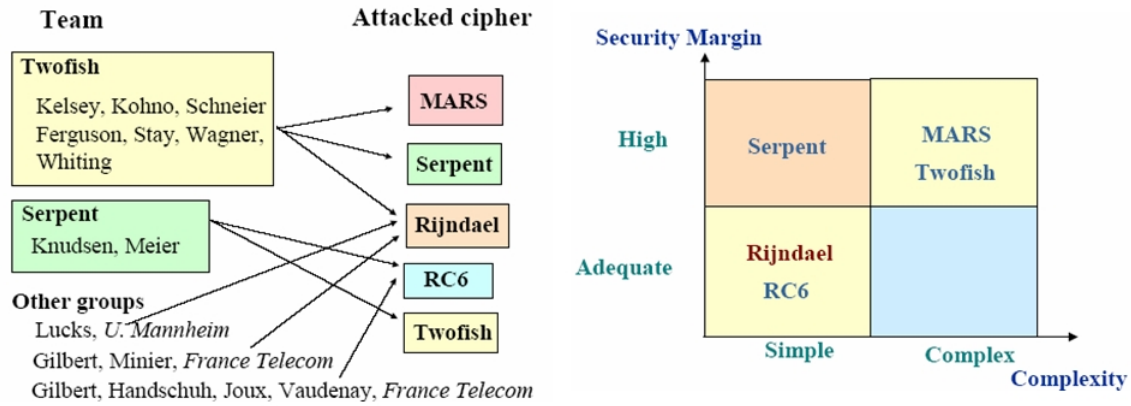

FIG. 25.1 – Distribution des jurys et Sécurité

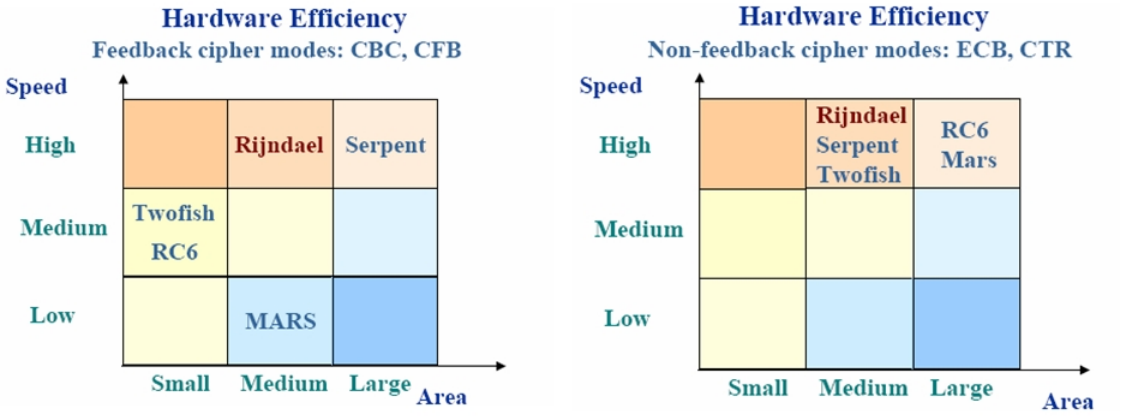

FIG. 25.2 – Efficacité matérielle

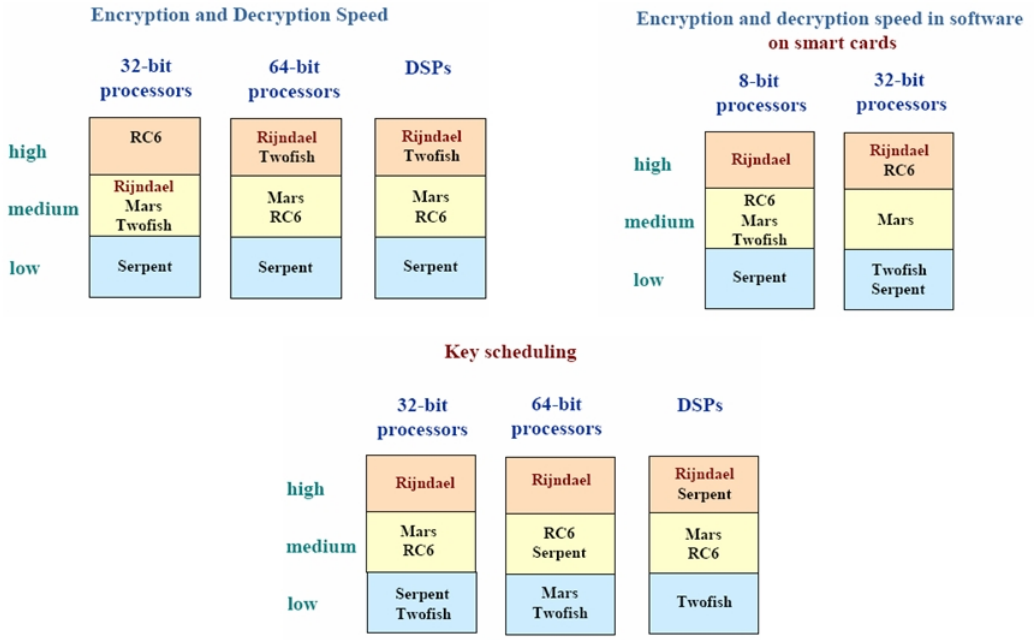

FIG. 25.3 – Efficacité logicielle des algorithmes participants

25.1 Rijndael - Daemen, Rijmen

Avantages :

- Le plus rapide en implantation matérielle
- Proche des plus rapides en implantation logicielle
- Très grande flexibilité

Inconvénients :

- Marge de sécurité

25.2 Serpent - Anderson, Biham, Knudsen

Avantages :

- Large marge de sécurité
- Construction conservatrice
- Très rapide en implantation matérielle
- Réputation cryptanalytique de ses auteurs (Anderson, Biham, Knudsen)

Inconvénients :

- Lent en implantation logicielle
- Flexibilité modérée

25.3 Twofish - Schneier, Wagner

Avantages :

- Bonne marge de sécurité
- Rapide pour le chiffrement/déchiffrement en implantation logicielle
- Américain
- Beaucoup de publicité

Inconvénients :

- Modérément rapide en implantation matérielle
- Configuration lente de la clé en implantation logicielle
- Flexibilité modérée

Chapitre 26

Annexe 3 : Sécurité logicielle

26.1 Intrusions

26.1.1 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, Telnet, ...)

La plupart du temps, cette attaque est un prémice à une autre attaque[1], 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.

26.1.2 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.

[1] On parle d'"étape de découverte".

26.1.3 DoS

Le but de l'attaque DoS, pour Denial of Service (ou 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.

26.1.4 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.

26.1.5 Ping of Death (obsolète)

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.

26.1.6 Ping Flooding

Comme toutes les 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.

26.1.7 Autres attaques

On pourra mentionner l'attaque par tunneling, les buffer overflow, les conditions de course, les DNS Poisoning, et beaucoup d'autres. Il s'agissait ici d'en présenter quelques-unes.

26.2 Les firewalls

26.2.1 Packet Filtering

Ce filtrage a lieu sur les couches Réseau (IP) et Transport (TCP/UDP).

Au niveau de la couche Réseau, le firewall vérifiera 3 informations (illustrées sur la figure 26.1) :
- L'adresse IP de destination
- L'adresse IP de la source
- Quelques options présentes à ce niveau

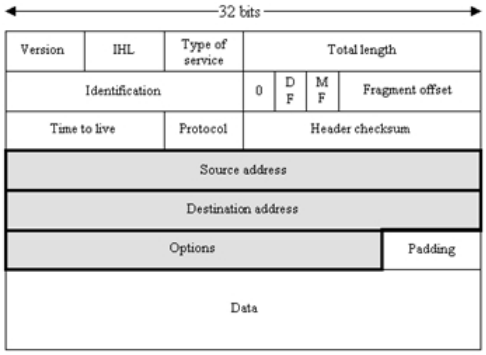

FIG. 26.1 – Packet Filtering - Couche Réseau (IP)

Au niveau de la couche Transport, le firewall analysera d'autres informations (illustrées sur la figure 26.2) telles que :
- Le port de destination
- Le port source
- Le type de protocole utilisé (TCP ou UDP)
- Les flags TCP

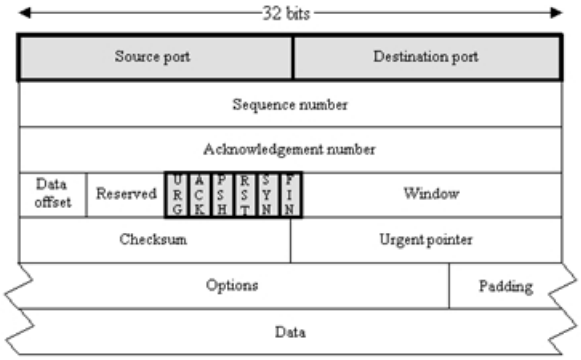

FIG. 26.2 – Packet Filtering - Couche Transport (TCP)

Bibliographie

[1] William Stallings. *Cryptography and Network Security : Principles and Practice, 3rd ed.* Prentice Hall, 2003.

[2] Didier Müller. *Les Codes secrets décryptés.* City Editions, 2007.

[3] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1996.

[4] Philippe Oechslin. Les compromis temps-memoire et leur utilisation pour casser les mots de passe windows. 2004.

[5] Travis Spann. Fault induction and environmental failure testing, 2005.

[6] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks, 2002.

[7] Adi Shamir and Aran Tromer. Acoustic cryptanalysis : On nosy people and noisy machines, 2004.

[8] Li Zhuang, Feng Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. 2005.

[9] Jan C. A. van der Lubbe. *Basic methods of cryptography.* Cambridge University Press, 1998.

[10] Douglas R. Stinson. *Cryptography : Theory and Practice, 2nd ed.* CRC Press, Inc., 2002.

[11] Anil K. Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition, 2004.

[12] Salil Prabhakary, Anil K. Jain, and Sharath Pankant. Learning fingerprint minutiae location and type, 2001.

[13] Dr. Marios Savvides. Introduction to biometric recognition technologies and applications, 2006.

[14] Jarmo Ilonen. Keystroke dynamics, 2003.

[15] B. Lampson. Computer security in the real world, 2001.

[16] Computer Security Institute. Csi/fbi 2007 computer crime and security survey, 2007.

[17] Craig Wright, Dave Kleiman, and Shyaam Sundhar. Overwriting hard drive data : The great wiping controversy. 2008.

[18] Peter Gutmann. Secure deletion of data from magnetic and solid-state memory. 1996.

[19] Gordon Hughes and Tom Coughlin. Tutorial on disk drive data sanitization, 2006.