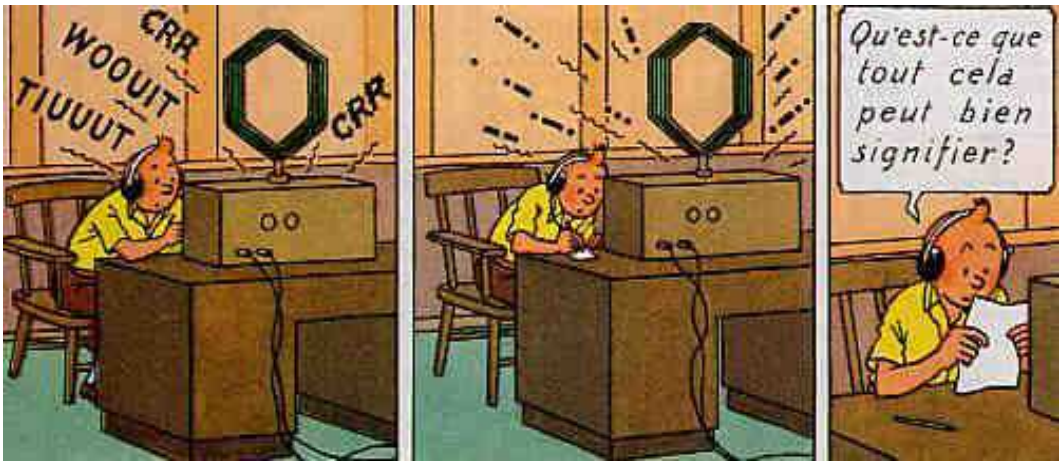
Jonathan BLANC
Adrien DE GEORGES
Licence Informatique

Enseignant : Sandrine JULIA
Année universitaire 2003/2004

# TECHNIQUES DE CRYPTOGRAPHIE



# TABLE DES MATIERES


**INTRODUCTION** 3

**1. TECHNIQUES DE CRYPTOGRAPHIE AU COURS DE L'HISTOIRE** 4

**1.1 SYSTEME DE CESAR** 4
**1.2 SYSTEME DE VIGENERE** 4
**1.3 SYSTEME DE PLAYFAIR** 5
**1.4 SYSTEME ADFG(V)X** 6
**1.5 MACHINES A ROTORS** 6

**2. CRYPTOSYSTEMES ACTUELS** 7

**2.1 CRYPTOGRAPHIE A CLEFS PRIVES** 7
2.1.1 CRYPTOSYSTEMES PAR FLOTS 7
2.1.2 CRYPTOSYSTEMES PAR BLOCS 8
2.1.3 ALGORITHME DATA ENCRYPTION STANDARD (DES) 9
2.1.4 RIJNDAEL (AES) 13
**2.2 CRYPTOGRAPHIE A CLEFS PUBLIQUES** 15
2.2.1 PRINCIPE 15
2.2.2 RSA 16
2.2.3 SSL 17

**3. SECURITE ET ATTAQUES DE SYSTEMES ACTUELS** 18

**3.1 RSA** 20
**3.2 DES** 21
**3.3 SSL** 22

**4. FONCTIONS DE HACHAGE, SIGNATURES ET CERTIFICATS ELECTRONIQUES** 23

**4.1 FONCTIONS DE HACHAGE** 23
4.1.1 PRINCIPE 23
4 .1.2 MD5 23
**4.2 SIGNATURES** 24
4.2.1 PRINCIPE 24
4.2.2 STANDARD DSS 24
4.2.3 STANDARD PKCS 25
**4.3 CERTIFICATS** 26
4.3.1 PRINCIPE 26
4.3.2 CERTIFICATS X.509 28

**CONCLUSION** 29

**REFERENCES BIBLIOGRAPHIQUES** 30

# INTRODUCTION

Dès que les hommes apprirent à communiquer, ils durent trouver des moyens d'assurer la confidentialité d'une partie de leurs communications : l'origine de la cryptographie remonte sans doute aux origines de l'homme.
En effet, le mot cryptographie est un terme générique désigant l'ensemble des techniques permettant de chiffrer des messages c'est-à-dire de les rendre inintelligibles sans une action spécifique.
Du bâton nommé « scytale » au Vie siècle avant JC, en passant par le carré de Polybe ou encore le code de César, on assista au développement plus ou moins ingénieux de techniques de chiffrement expérimentales dont la sécurité reposait essentiellement dans la confiance que leur accordaient leurs utilisateurs. Après la première guerre mondiale a lieu une première révolution technologique.
Mais ce n'est qu'à l'avènement de l'informatique et d'Internet que la cryptographie prend tout son sens. Les efforts conjoints d'IBM et de la NSA conduisent à l'élaboration du DES (Data Encryption Standard), l'algorithme de chiffrement le plus utilisé au monde durant le dernier quart du XXème siècle. A l'ère d'Internet, le nombre d'applications civiles de chiffrement (banques, télécommunications, cartes bleues,…) explose. Le besoin d'apporter une sécurité accrue dans les transactions électroniques font naître les notions de signature et authentification électronique. La première technique de chiffrement à clef publique sûre (intimement liée à ces notions) apparaît : le RSA.
Nous nous tournerons dans un premier temps vers les techniques cryptographiques qui ont marqué l'histoire, suivis par les techniques actuelles du monde de l'informatique.
Par ailleurs, nous arrêterons également sur la sécurité des algorithmes les plus connus ainsi que sur les notions de signatures, certificats et fonctions de hachage.

# 1. Techniques de cryptographie au cours de l'histoire

Contrairement à ce que l'on peut penser, la cryptographie n'est pas seulement une technique moderne, ni un produit de l'ère informatique. En effet de tout temps, les hommes ont ressenti le besoin de cacher des informations confidentielles. Bien évidemment depuis ses débuts la cryptographie a grandement évolué. Au cours des siècles, de nombreux systèmes de chiffrage ont été inventés, tous de plus en plus perfectionnés, et il est vrai que l'informatique y a beaucoup contribué. Mais au commencement les algorithmes étaient loin d'être aussi complexes et astucieux qu'à notre époque. La majeure partie des méthodes d'antan reposait sur deux principes fondamentaux : la substitution (remplacer certaines lettres par d'autres) et la transposition (permuter des lettres du message afin de le brouiller).

## 1.1 Système de César

L'un des systèmes les plus anciens et les plus simples est le codage par substitution mono alphabétique (ou alphabets désordonnés). Il consiste à remplacer chaque lettre par une lettre différente. Il existe donc grâce à cette technique 26 façons de coder un message, ce qui fait que ce système a été longtemps utilisé par les armées pendant l'antiquité. Ce procédé très fiable à l'époque est tout de même problématique car il nécessite que les interlocuteurs se souviennent tous deux de la clef. De plus, il est évident que la sûreté de ce codage est quasi nulle et qu'il pourrait être déchiffré par n'importe quelle personne qui y mettrait le temps nécessaire.

Voici un exemple de substitution :

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

La méthode la plus ancienne admise par l'histoire (par substitution alphabétique) est le non moins connu code de César, consistant en un décalage simple de lettres. Par substitution si l'on remplace le A par le C, alors le B devient D, le D un F, etc.… César utilisait ce code simple pour transmettre via un message des consignes à ces généraux d'armées sans qu'il puissent être exploité par un quelconque ennemi dans le cas ou le message serait intercepté. Malheureusement il n'y a que 26 façons différentes de chiffrer à l'aide de ce code ce qui en fait un code très peu sûr. Mais ce qui est d'autant plus insolite, c'est le fait que ce code de « César » est encore utilisé de nos jours sur Internet avec le ROT13 (rotation de 13 lettres) qui consiste à cacher des messages afin qu'ils ne soient pas lus involontairement, comme par exemple s'ils dévoilent le dénouement d'un film ou encore qui donne la réponse à une devinette.

## 1.2 Système de Vigenère

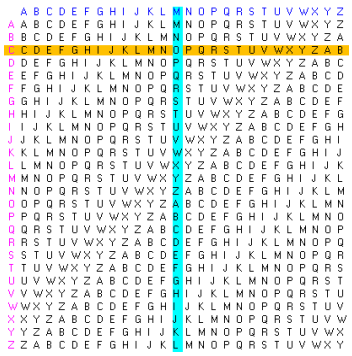
Un autre système de cryptographie des plus anciens est cette fois-ci, la substitution polyalphabétique, qui utilise plusieurs alphabets décalés pour crypter un message. L'algorithme de substitution polyalphabétique le plus connu est le chiffre de Vigenère, mis au point par Blaise de Vigenère en 1586, qui fut utilisé pendant plus de 3 siècles. Son chiffre consiste à utiliser le chiffre de César, mais en changeant le décalage à chaque fois. Il utilise alors un carré composé de 26 alphabets alignés, décalés de colonne en colonne d'un caractère.

Il place également au dessus de ce carré, un alphabet pour la clef et à sa gauche un autre alphabet pour le texte à coder. Il suffit alors, pour chiffrer un message, de choisir un mot de longueur quelconque, de l'écrire sous le message à coder (de façon répétée s'il le faut) et de regarder dans le tableau l'intersection de la lettre à coder et de la lettre de la clef.

Pour mieux comprendre le fonctionnement du Carré de Vigenère nous vous proposons cet exemple :

Supposons que nous voulons coder le texte « CARRE DE VIGENERE » avec la clef « MALICE ». On commence par écrire la clef sous le texte à coder :

| C | A | R | R | E | | D | E | | V | I | G | E | N | E | R | E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M | A | L | I | C | | E | M | | A | L | I | C | E | M | A | L |



Pour coder la lettre C, la clef est donnée par la lettre M. On regarde dans le tableau l'intersection de la ligne donnée par le C, et de la colonne donnée par le M. On trouve O. Puis on continue, jusqu'à ce qu'on ait fini de chiffrer notre texte. En chiffrant le texte « Carre de Vigenere », on obtient donc le texte « OAUZG HG VTOGRQRP ».
Cet algorithme de cryptographie ainsi que celui de César sont les premiers des algorithmes à clef privée.

## 1.3 Système de Playfair

Il existe d'autres systèmes presque aussi anciens basés également sur des techniques par substitution mais moins connus que ceux vus précédemment. Il s'agit des systèmes par substitution de polygrammes. En effet au lieu de substituer des caractères, on substitue par exemple des diagrammes (des groupes de lettres le plus souvent). Le système de « Playfair » inventé par Sir Charles Wheatstone, popularisé par L.Playfair utilise ce stratagème au moyen d'une table. Cet algorithme remplace chaque paire de lettre du texte en clair par une autre paire. Il utilise pour cela une table (matrice) carrée de coté 5, construite à partir d'une clef, qui contient toutes les lettres de l'alphabet hormis une (souvent le J par similitude avec le I). Chaque couple de lettre donne les coordonnées d'un rectangle dans la matrice. On remplace donc ce couple par les lettres formant les deux autres coins du rectangle. Si les deux lettres du couple sont sur la même ligne, on prend les deux lettres suivantes. Si elles sont sur la même colonne, on prend les 2 lettres du dessous. Si les 2 lettres sont identiques, on intercale entre elles une lettre convenue à l'avance (X ou Y).

Malheureusement, ce chiffre ingénieux ne fut pas utilisé souvent en raison du fait qu'il se déchiffre aisément en regardant quel couple de lettres apparaît le plus souvent dans le texte chiffré, et en supposant qu'ils représentent le couple de lettres le plus courant.

## 1.4 Système ADFG(V)X

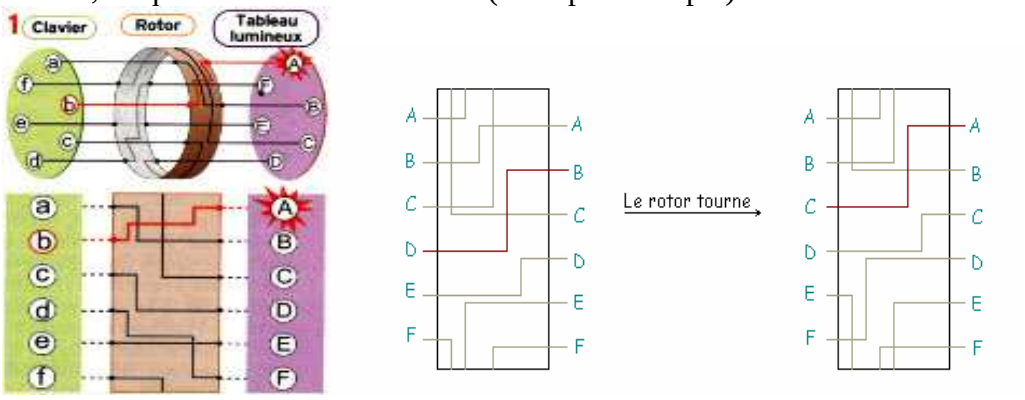
Mais même si la cryptographie était beaucoup utilisée de tout temps, jusqu'au début du XXe siècle, elle gardait une importance mineure, et les méthodes utilisées étaient bien souvent rudimentaires. Avec la 1e guerre mondiale, la cryptographie subit un véritable « boum ». La communication entre état major et les troupes se fait par radio et il faut donc empêcher les ennemis de comprendre les messages s'ils sont interceptés. C'est là que la cryptographie intervient, permettant à l'armée française de stopper de nombreuses offensives allemandes. Le chiffre ADFGVX en est un exemple concret. En effet ce code était utilisé par l'armée allemande depuis mars 1918 et malgré leurs efforts et leur talent, l'équipe française de cryptographie n'arrivaient pas à le déchiffrer. Pour réaliser ce code, qui mélange transpositions et substitutions, il faut ranger les 26 lettres de l'alphabet ainsi que les 10 chiffres dans un tableau de 6 cases sur 6. Au dessus et à côté de ce tableau est ajouté le fameux mot ADFGVX. Chacune des lettres du texte en clair est remplacée par le couple de lettres qui correspond à sa ligne et sa colonne.

| | A | D | F | G | V | X |
|---|---|---|---|---|---|---|
| A | Q | Y | A | L | S | E |
| D | Z | X | R | C | H | 0 |
| F | F | O | 4 | M | 8 | 7 |
| G | 3 | I | T | G | U | K |
| V | P | D | 6 | 2 | N | G |
| X | 1 | 5 | J | 9 | W | B |

Le 2 juin 1918, l'équipe de cryptographie perce à jour ce système et déchiffre un message allemand ordonnant une attaque éclair près de Compiègne. Les troupes françaises s'y massent, permettant de stopper l'ennemi.

## 1.5 Machines à rotors

Très vite après la première guerre, on s'est rendu compte que si l'on souhaitait diffuser beaucoup de documents chiffrés rapidement, et pouvoir changer de clef de chiffrement facilement, il fallait fabriquer des machines à chiffrer et à déchiffrer. Les machines utilisées à ces fins sont les machines à rotors (dont la plus célèbre était la machine ENIGMA à 3 rotors inventée dans les années 30). Cette machine électrique est composée d'un clavier alphabétique, d'un écran lumineux et de trois rotors. Le système est simple : l'utilisateur tape une lettre sur le clavier et le texte chiffré apparaît alors sur l'écran. A chaque frappe sur le clavier, le premier rotor tournait d'une unité puis à la fin d'un tour complet décalait le deuxième rotor d'une unité et ainsi de suite. On positionnait initialement les rotors comme on voulait, ce qui définissait ainsi la clef (FAC par exemple).




Grâce à cette machine un texte pouvait être codé d'un million de façons différentes. La frappe au clavier d'une lettre en allumait une autre sur l'écran de manière symétrique (si A donnait C alors C donnait A). Ainsi pour chiffrer un message, une fois la clef fixée, il suffisait de le

taper sur la machine et pour le déchiffrer de mettre les rotors dans la même position initiale et de taper le message chiffré. Chaque message commençait par la donnée de la clef choisie par l'opérateur, qu'il cryptait elle aussi selon une liste de clef changeant tous les jours. La machine ENIGMA a été utilisée pendant toute la seconde guerre mondiale par l'armée allemande qui croyait en son inviolabilité. Une équipe de mathématiciens (spécialisée en cryptanalyse, art de déchiffrer des messages) anglais dirigée par A.Turing finit par la décrypter.
On voit donc ici que la cryptographie est une technique de guerre à part entière et qu'elle joue un vrai rôle dans l'ère moderne. En effet on se rend compte qu'en l'espace de quelques années, la cryptographie et la cryptanalyse sont passées de simples techniques désuètes, à véritables sciences. Cette progression des techniques et algorithmes de cryptage ne s'est pas faite toute seule et c'est totalement à cause des attaques incessantes, visant à « casser » les techniques adverses, que l'on a pu assister à un tel bond.

# 2. Cryptosystèmes actuels

## 2.1 Cryptographie à clefs privés

La cryptographie à clefs privées, appelée aussi cryptographie symétrique est utilisée depuis déjà plusieurs siècles. C'est l'approche la plus authentique du chiffrement de données et mathématiquement la moins problématique.
La clef servant à chiffrer les données peut être facilement déterminée si l'on connaît la clef servant à déchiffrer et vice-versa. Dans la plupart des systèmes symétriques, la clef de cryptage et la clef de décryptage sont une seule et même clef.
Les principaux types de cryptosystèmes à clefs privés utilisés aujourd'hui se répartissent en deux grandes catégories : les cryptosystèmes par flots et les crytosystèmes par blocs.

### 2.1.1 Cryptosystèmes par flots

Dans un cryptosystème par flots, le cryptage des messages se fait caractère par caractère ou bit à bit, au moyen de substitutions de type César générées aléatoirement : la taille de la clef est donc égale à la taille du message. L'exemple le plus illustratif de ce principe est le chiffre de Vernam. Cet algorithme est aussi appelé « One Time Pad » (masque jetable), c'et à dire que la clef n'est utilisée qu'une seule fois.
Voici un exemple simple de l'application du chiffre de Vernam :

Exemple:

Message en clair: "**SALUT**"

=> (conversion en binaire)

*01010011 01000001 01001100 01010101 01010100*

XOR

Clef (générée aléatoirement)

*01110111 01110111 00100100 00011111 00011010*

=

*00100100 00110110 01101000 01001010 01001110*

=> (conversion en caractère)

"Message chiffré: **$6jJM**"

Il a été démontré par le mathématicien Claude Elwood Shannon qu'il était impossible de retrouver un message crypté par le principe de Vernam sans connaître la clef. Ce qui ferait en théorie du chiffre de Vernam un cryptosystème incassable. Mais dans la pratique, le cryptosystème par flots pose des problèmes délicats : canaux sûrs de distribution des clefs, taille des clefs encombrantes car de même taille que le message et surtout caractère aléatoire des générateurs de bits de clefs utilisés. En revanche, un des avantages du système est qu'il est insensible aux phénomènes de propagation d'erreurs : un bit erroné donne une erreur à la réception ou à l'émission, mais est sans incidence sur les bits suivants.

### 2.1.2 Cryptosystèmes par blocs

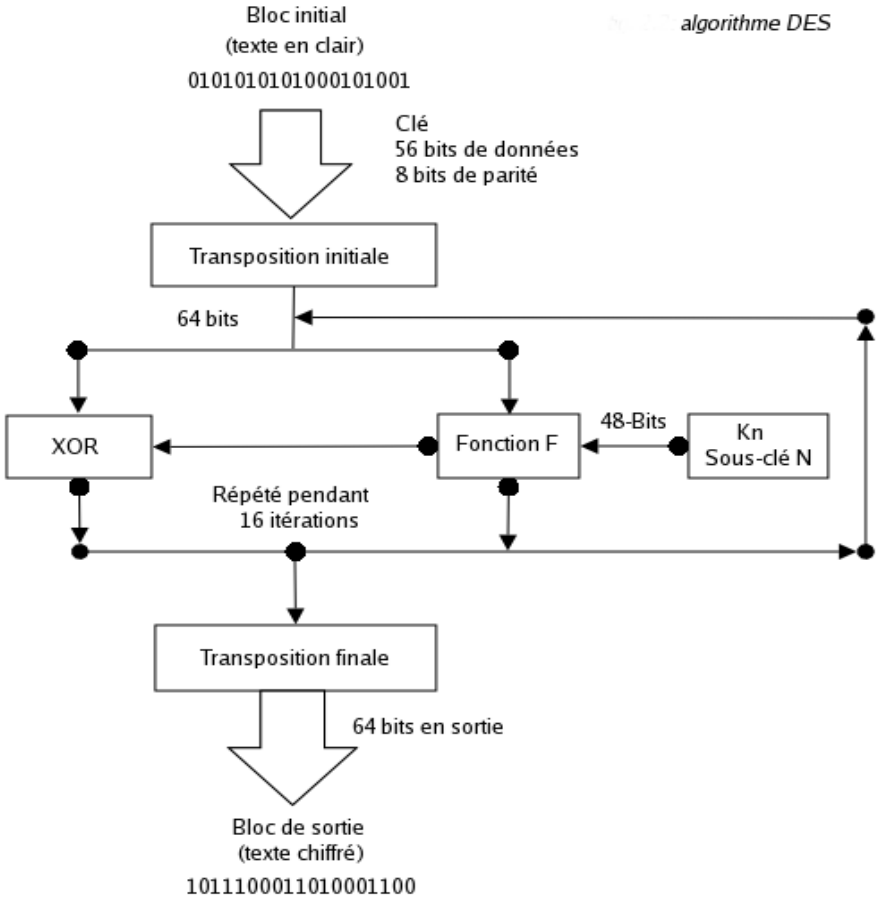
La deuxième classe de cryptosystèmes utilisée aujourd'hui est celle des cryptosystèmes par blocs. Dans ce mode de cryptage, le texte clair est fractionné en blocs de même longueur à l'aide d'une clef unique. Les algorithmes de chiffrement par blocs sont en général construits sur un modèle itératif. Ce modèle emploie une fonction F qui prend en paramètres une clef k et un message de n bits. F est répétée un certain nombre de fois, on parle de ronde. A chaque ronde, la clef k utilisée est changée et le message que l'on chiffre est le résultat de l'itération précédente.

$C_1 = F(k_1, M)$
$C_2 = F(k_2, C_1)$
...
$C_r = F(k_r, C_{r-1})$

Emetteur et destinataire se partagent une clé K secrète. L'algorithme qui engendre les clefs $k_i$ à partir de K se nomme l'algorithme de cadencement des clefs.
La fonction F doit être inversible, ce qui veut dire qu'il faut pour toute clef k et message M pouvoir recalculer M a partir de F (k′, M*)*, sinon le déchiffrement est impossible et on ne dispose pas d'un algorithme utilisable. C'est-à-dire qu'il existe une fonction G vérifiant
G (k, F (k, M)) = M et que F est une permutation.
La sécurité d'un algorithme de chiffrement par blocs réside principalement dans la conception de l'algorithme de cadencement des clefs et la robustesse de la fonction F. Si l'algorithme de cadencement est mal élaboré, les $k_i$ peuvent être déductibles les unes des autres. La fonction F doit donc être difficile à inverser sans connaître la clef k ayant servie dans le calcul de C = F (k, M). En d'autres termes, connaissant seulement *C,* F et G, on ne doit pouvoir retrouver le message M seulement en effectuant une recherche exhaustive de la clef.
Les caractéristiques de ces systèmes sont en général liées à leur très forte sensibilité à la dépendance inter-symboles, ainsi qu'à leur mécanisme de propagation d'erreurs. Toute erreur commise sur un bloc de texte clair ou chiffré peut perturber gravement le chiffrement/déchiffrement de ses voisins.

### 2.1.3 Algorithme Data Encryption Standard (DES)

Publié en 1977 par le NBS (National Bureau of Standards), le DES est un algorithme de chiffrement de données recommandé pour les organisations à caractère fédéral, commercial ou privé. Le DES tire son origine des travaux menés par le groupe cryptographique d'IBM dans le cadre du projet LUCIFER. Le DES a été l'objet de nombreuses implémentations, à la fois en matériel et en logiciel, depuis sa publication. Après une décennie de succès, pendant laquelle les moyens et techniques de cryptanalyse mis en œuvre pour en étudier les caractéristiques n'ont pas permis d'en découvrir des faiblesses rédhibitoires, le DES a, depuis peu, révélé des sensibilités à des attaques nouvelles et puissantes, parfois réalisées sur un simple micro-ordinateur. Aussi l'ISO (International Organization for Standardization) a-t-il récemment refusé la normalisation du DES, ce qui n'empêche pas cet algorithme d'être, de loin, aujourd'hui encore comme le moyen de chiffrement le plus sûr (et le plus largement utilisé) pour des données non militaires.
Le DES est un algorithme de chiffrement  symétrique par blocs qui permet de chiffrer des mots de 64 bits à partir d'une clef de 56 bits(56 bits servant à chiffrer + 8 bits de parité servant à vérifier l'intégrité de la clef en réalité).



*algorithme DES*

Voici les différentes étapes de l'algorithme du DES :

**Fractionnement du message**
Dans un premier temps le message en clair est découpé en blocs de 64 bits.

**Transposition initiale**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

Chaque bit d'un bloc subit une permutation selon l'arrangement du tableau ci-contre c'est-à-dire que le 58ème bit du bloc se retrouve en 1ère position, le 50ème en seconde position,etc…

**Scindement en bloc de 32 bits**

Le bloc de 64 bits est scindé en deux blocs de 32 bits notés G et D. On notera $G_0$ et $D_0$ l'état initial de ces deux blocs.

$G_0$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

$D_0$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

On remarque que $G_0$ contient tous les bits pairs du message initial et $D_0$ tous les bits impairs.

**Rondes**

Les blocs $G_i$ et $D_i$ sont soumis à un ensemble de transformation appelées rondes.
Une ronde est elle-même composée de plusieurs étapes :

- **Fonction d'expansion :**

| | | | | | |
|---|---|---|---|---|---|
| 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 |

Les 32 bits du bloc $D_0$ sont étendus à 48 bits grâce à une table d'expansion dans laquelle 32 bits sont mélangés et 16 d'entre eux sont dupliqués.
Ainsi, le 32ème bit devient le premier, le premier devient le second… Les bits 1,4,5,8,9,12,13,16,17,22,21,24,25,28,29 et 32 sont dupliqués et disséminés pour former un bloc de 48 bits que l'on nommera $D'_0$ .

- **OU exclusif (XOR) avec la clef :**

DES procède ensuite à un OU exclusif entre $D'_0$ et la première clef $k_1$ générée à partir de la clef K (que doivent se partager émetteur et destinataire) par l'algorithme de

cadencement des clefs que nous décrirons plus bas. Nous appellerons $D''_0$ le résultat de cette opération.

- **Boîtes de substitution :**

$D''_0$ est découpée ensuite en 8 blocs de 6 bits, noté $D''_{0i}$ . Chacun de ces blocs passe par des boîtes de substitution(S-boxes), notées généralement $S_i$ .

Les premier et dernier bits de chaque $D_{0i}$ déterminent la ligne de la fonction de substitution, les autres bits déterminent la colonne. Grâce à cela la fonction de substitution « choisit » une valeur codée sur 4 bits (de 0 à 15).

Voici la première boîte de substitution :

$S_1$

| | 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 |

Soit $D_{0i}$ égal à 010101, les premiers et derniers bits donnent 01, c'est-à-dire 1 en binaire.

Les bits autres bits donnent 1010, soit 10 en binaire. Le résultat de la fonction de substitution est donc la valeur située à la ligne n°1, dans la colonne n°10. Il s'agit de la valeur 6, soit 0110 en binaire.

Chacun des 8 blocs de 6 bits est passé dans la boîte de substitution correspondante.

Voici les autres S-Boxes :

$S_2$

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

$S_3$

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

$S_4$

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

$S_5$

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

$S_6$

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

$S_7$

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

$S_8$

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

On obtient donc en sortie 8 blocs de 4 bits. Ces bits sont regroupés pour former un bloc de 32 bits.

- **Permutation :**

Le bloc de 32 bits subit une permutation dont voici la table :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

- **OU exclusif :**

Le bloc de 32 bits ainsi obtenu est soumis à un OU exclusif avec le $G_0$ de départ pour donner $D_1$ et le $D_0$ initial donne $G_1$.

L'ensemble de ces étapes est itérée seize fois.

**Transposition initiale inverse**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

Au bout des seize itérations, les deux blocs $G_{16}$ et $D_{16}$ sont « recollés » pour reformer un seul bloc de 64 bits puis subit la transposition initiale inverse selon l'arrangement du tableau ci-contre.
On obtient alors le bloc initial chiffré.

**Reconstruction du message chiffré**

Tous les blocs sont collés bout à bout pour obtenir le message chiffré.

**Algorithme de cadencement des clefs**

Nous allons décrire l'algorithme qui permet de générer à partir d'une clef de 64 bits, 8 clefs diversifiées de 48 bits chacune servant dans l'algorithme du DES.
De prime abord les clefs de parité sont éliminées pour obtenir une clef de 56 bits.

Ce bloc subit une permutation puis est découpée en deux pour obtenir 2 blocs de 28 bits décrits par les matrices ci-dessous :

| | | | | | | |
|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 |

Ces deux blocs subissent une rotation à gauche, c'est-à-dire que les bits en seconde position prennent la première position, ceux en troisième position la seconde, celle en première position la dernière…

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 | 3 | 28 | 15 | 6 | 21 | 10 |
| 13 | 19 | 12 | 4 | 26 | 8 | 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 | 30 | 40 | 51 | 45 | 57 | 45 |
| 44 | 49 | 39 | 56 | 34 | 53 | 46 | 42 | 50 | 36 | 29 | 32 |

Les 2 blocs sont regroupés pour faire un bloc de 56 bits qui passe par une permutation fournissant un bloc de 48 bits représentant la clef $k_i$. :

Des itérations de l'algorithme permettent de donner les 16 clefs utilisées dans l'algorithme du DES.

### 2.1.4 Rijndael (AES)

En Janvier 1997, la NIST (National Institute of Standards and Technology) lance un appel d'offre international pour remplacer le vieillissant DES : il en résulte 15 propositions.
Parmi ces 15 algorithmes, 5 furent choisis pour une évaluation plus avancée en avril 1999 : MARS, RC6,Rijndael, Serpent et Twofish. Finalement, en octobre 2000 la NIST élit Rijndael comme nouveau standard qu'on nomme aussi AES (Advanced Encryption Standard).
Rijndael, du nom condensé de ses concepteurs Rijmen et Daemen, est un algorithme de chiffrement par blocs à plusieurs tours similaire à DES mais avec une taille de blocs et de clefs supérieures et variables, choisis entre 128, 196 et 256 bits.

**Le Corps GF($2^8$)**

Un octet b composé des 8 bits $b_7,b_6,b_5,b_4,b_3,b_2,b_1,b_0$ peut être vu comme un polynôme de degré inférieur ou égal à 7 avec des coefficients dans {0,1} :

$$b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0$$

L'addition de deux polynômes de ce type revient à additionner modulo 2 les coefficients de chacun. Cette addition correspond au OU exclusif.
Pour la multiplication, c'est la multiplication usuelle suivie d'une réduction modulo un polynôme binaire irréductible de degré 8.
Dans Rijndael, ce polynôme est **$m(x) = x^8 + x^4 + x^3 + x + 1$.** Le résultat sera à nouveau un polynôme de degré inférieur ou égal à 7.
Pour tout polynôme binaire de degré inférieur ou égal à 8, l'algorithme d'Euclide étendu permet de calculer b(x) tel que a(x) b(x) mod m(x) soit égal à 1, autrement de calculer l'inverse de a(x) : $a^{-1}(x)$.
On peut voir que l'ensemble des 256 éléments possibles, avec l'addition et la multiplication ci-dessus,ont la structure du corps GF($2^8$) : le corps fini de polynômes de degré $\leq 7$ avec des coefficients dans {0,1}.

**Structure d'état dans l'AES**

On appelle état un bloc vu comme un tableau de 4 x $N_b$ octets où $N_b$ est égal à Taille du bloc / 32. On représente la clef de la même façon, le nombre de colonnes étant $N_k$ = longueur de la clef / 32.

| | | | |
|---|---|---|---|
| $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ |
| $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ |
| $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ |
| $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ |

| | | | |
|---|---|---|---|
| $k_{0,0}$ | $k_{0,1}$ | $k_{0,2}$ | $k_{0,3}$ |
| $k_{1,0}$ | $k_{1,1}$ | $k_{1,2}$ | $k_{1,3}$ |
| $k_{2,0}$ | $k_{2,1}$ | $k_{2,2}$ | $k_{2,3}$ |
| $k_{3,0}$ | $k_{3,1}$ | $k_{3,2}$ | $k_{3,3}$ |

Exemple d'état (avec des blocs de 128 bits, $N_b = 4$) et de clef (de longueur 128 bits, $N_k = 4$)

**Nombre de tours**

Le nombre de tours dans l'AES dépend à la fois de la taille des blocs et de la clef. Le nombre r de tours est donné par le tableau :

| $N_r$ | $N_b = 4$ (128 bits) | $N_b = 6$ (192 bits) | $N_b = 8$ (256 bits) |
|---|---|---|---|
| $N_k = 4$ (128 bits) | 10 | 12 | 14 |
| $N_k = 6$ (192 bits) | 12 | 12 | 14 |
| $N_k = 8$ (256 bits) | 14 | 14 | 14 |

Chaque tour utilise une sous-clef $k_i$ différente et est composée de quatre étapes :
`ByteSub, ShiftRow, MixColumn et AddRoundKey.`

**ByteSub**

ByteSub est une substitution qui agit isolément sur tous les octets $a_{i,j}$ d'un état en 2 étapes :

1. on regarde $a_{i,j}$ comme polynôme dans GF($2^8$), et on prend son inverse $a^{-1}_{i,j.}$
2. on calcule l'image du résultat par la fonction y = f(x) suivante :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

**ShiftRow**
ShiftRow effectue un décalage des lignes de l'état courant. La ligne 0 n'est pas décalée, la ligne 1 l'est de $\mathbf{C_1}$ octets, la 2 de $\mathbf{C_2}$ octets et la ligne 3 de $\mathbf{C_3}$ octets. Les valeurs de $C_1$, $C_2$ et $C_3$ dépendant de la taille du bloc, selon la table suivante :

| $\mathbf{N_b}$ | $\mathbf{C_1}$ | $\mathbf{C_2}$ | $\mathbf{C_3}$ |
|---|---|---|---|
| 4 | 1 | 2 | 3 |
| 6 | 1 | 2 | 3 |
| 8 | 1 | 3 | 4 |

**MixColumn**
La transformation MixColumn consiste à prendre chaque colonne de l'état et à la multiplier par la matrice suivante:

$$\begin{pmatrix} b_{0,x} \\ b_{1,x} \\ b_{2,x} \\ b_{3,x} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} = \begin{pmatrix} a_{0,x} \\ a_{1,x} \\ a_{2,x} \\ a_{3,x} \end{pmatrix}$$

**AddRoundKey**
AddRoundKey consiste en un OU exclusif de l'état courant et de la clef du tour.

## 2.2 Cryptographie à clefs publiques

### 2.2.1 Principe

Tous les algorithmes évoqués jusqu'à présent sont symétriques en ce sens que la même clef est utilisée pour le chiffrement et le déchiffrement. Le problème essentiel de la cryptographie symétrique est la distribution des clefs : pour que n personnes puissent communiquer de manière confidentielle il faut n (n-1)/2 clefs.
L'idée de base des cryptosystèmes à clefs publiques a été proposée dans un article fondamental de Diffie et Hellman en 1976. Le principe fondamental est d'utiliser des clefs de chiffrement et déchiffrement différentes, non reconstructibles l'une à partir de l'autre :

- une clef publique pour le chiffrement
- une clef secrète pour le déchiffrement

Ce système est basé sur une fonction à sens unique, soit une fonction facile à calculer dans un sens mais très difficile à inverser sans la clef privée.
Pour faire une explication imagée, la clef publique joue le rôle d'un cadenas. Imaginons que seul Bob possède la clef (clef secrète), Alice enferme son message dans une boîte à l'aide du cadenas et l'envoie à Bob. Personne n'est en mesure de lire le message puisque seul Bob possède la clef du cadenas.

Le gros avantage de ce système est qu'il n'y ait pas besoin d'avoir partagé un secret au préalable pour s'échanger des messages cryptés. En revanche les implémentations de tels systèmes (RSA, ElGamal,…) ont un inconvénient majeur : leur lenteur par rapport à leurs homologues à clefs secrètes qui tournent eux jusqu'à près de mille fois plus vite.

### 2.2.2 RSA

L'algorithme le plus célèbre d'algorithme à clef publique a été inventé en 1977 par Ron Rivest, Adi Shamir et Len Adleman, à la suite de la publication de l'idée d'une cryptographie à clef publique par Diffie et Hellman. Il fut appelé RSA, des initiales de ces inventeurs.
RSA est basé sur la difficulté de factoriser un grand nombre en produit de deux grands facteurs premiers. L'algorithme fonctionne de la manière suivante :
Imaginons que Bob souhaite recevoir d'Alice des messages en utilisant RSA.

1. **génération des clefs** :
    a. p et q, deux grands nombres premiers sont générés au hasard grâce à un algorithme de test de primalité probabiliste, avec n = pq.
    b. Un nombre entier e premier avec (p-1)(q-1) est choisi. Deux nombres sont premiers entre eux s'ils n'ont pas d'autre facteur commun que 1.
    c. L'entier d est l'entier de l'intervalle [2, (p-1)(q-1)[ tel que ed soit congrue à 1 modulo (p-1)(q-1), c'est-à-dire tel que ed-1 soit un multiple de (p-1)(q-1).
2. **distribution des clefs** : le couple (n, e) constitue la clef publique de Bob. Il la rend disponible à Alice en lui envoyant ou en la mettant dans un annuaire. Le couple(n, d) constitue quand à lui sa clef privée.
3. **chiffrement du message** : Pour crypter le message Alice représente le message sous la forme d'un ou plusieurs entiers M compris entre 0 et n-1. Elle calcule $C= M^e \bmod n$ grâce à la clef publique (n, e) de Bob et envoie C à Bob.
4. **déchiffrement du message** : Bob reçoit C et calcule grâce à sa clef privée $C^d \bmod n$.
   Il obtient ainsi le message initial M.

**Exemple :**
Bob choisit p = 17 et q = 19, n = p x q = 323 et e = 5.
Sa clef privée est alors d=173 car 173 x 5 = 1 (mod (16 x 18))
Supposons qu'Alice veuille lui envoyer le message « BONJOUR » en se servant du tableau suivant pour transformer les lettres en nombre :

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |

Cela donne :

| B | O | N | J | O | U | R |
|---|---|---|---|---|---|---|
| 2 | 15 | 14 | 10 | 15 | 21 | 18 |

Après avoir chiffré en remplaçant chaque nombre b par ($b^e \bmod n$) on obtient :

| 32 | 2 | 29 | 193 | 2 | 89 | 18 |
|---|---|---|---|---|---|---|

Qu'Alice envoie à Bob.

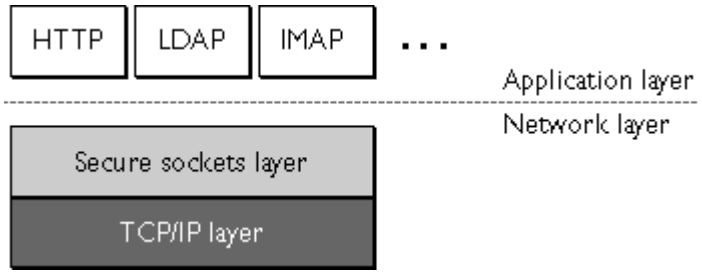
Bob réalise pour chaque nombre b du message $b^d$ mod n pour trouver :

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 15 | 14 | 10 | 15 | 21 | 18 |
| B | O | N | J | O | U | R |

qui est bien le message initial.
RSA est basé sur la difficulté de factoriser n. En effet celui qui arrive à factoriser n peut retrouver facilement la clef secrète de Bob connaissant seulement sa clef publique. C'est pourquoi dans la pratique la taille des clefs est au minimum de 512 bits.

### 2.2.3 SSL

Le protocole SSL (Secure Sockets Layers, que l'on pourrait traduire par « couche de transport sécurisé »), est un procédé développé par Netscape ayant pour but de sécuriser les transactions effectuées sur Internet. De nombreux sites de commerces de nos jours sont sécurisés avec SSL (afin de communiquer sûrement avec leurs clients et d'obtenir le paiement de leurs ventes). Ce système repose à la fois sur les algorithmes à clef publique (RSA, Diffie-Hellmann), sur les algorithmes à clef privée et sur les certificats électroniques (que nous traiterons plus tard) afin de garantir au maximum la sécurité de la transmission de données avec un tel site. Cependant un utilisateur quelconque ignore en principe totalement qu'il utilise SSL, car celui-ci agit de manière transparente. En fait SSL est indépendant des protocole de communications, il agit directement entre la gestion des commandes et la gestion du transport des données (il agit comme une couche supplémentaire de protection).

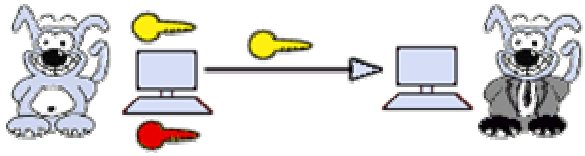



De cette façon un utilisateur se connectant à un site de commerce sécurisé via un navigateur Internet enverra des données chiffrées (code de Carte Bleue..) par SSL s'en même s'en préoccuper. Seuls l'apparition d'un petit cadenas s'affichant dans le navigateur et l'url commençant par https:// (le «s» signifiant secured) pourront permettre à un utilisateur averti de se rendre compte de l'intervention de SSL dans un tel échange.

L'utilisation de SSL est donc assez simple étant donné qu'elle se fait seule : son fonctionnement l'est aussi. La transaction par protocole SSL est en fait basé sur un échange de clef entre un client et un serveur. La transaction sécurisée se fait selon un schéma bien défini. Pour commencer, le client se connecte à un site de commerce (grâce à un navigateur utilisant SSL) en lui demandant de s'identifier (ce qui évitera de nombreux problèmes comme nous le verrons dans le chapitre concernant les attaques sur SSL). Le client envoie aussi une liste des cryptosystèmes qu'il connaît. Ensuite le serveur s'identifie en envoyant un certificat d'authentification contenant la clef publique du serveur ainsi que le nom du cryptosystème qui lui convient (souvent celui dont la longueur de clef est la plus longues).

**clef jaune : clef publique**

**clef rouge : clef privée**



A la réception, le client doit alors vérifier la validité du certificat envoyé par le marchant (serveur). Il choisi en suite une clef secrète (de manière aléatoire) et l'envoie au serveur sous forme cryptée grâce à la clef publique du serveur (c'est la que RSA intervient). Cette nouvelle clef est appelée clef de session. Le serveur est enfin capable de déchiffrer le reste des transactions par le biais de la clef de session. Le serveur ainsi que le client sont donc tous deux en possession d'une clef commune , permettant alors la confidentialité des données échangées. Une dernière authentification est possible, celle du client. Elle permettrait encore une plus grande sécurité, mais elle est en fait très rarement utilisée dans les utilisations courantes de SSL.

L'utilisation du système SSL ne cesse de s'accroître. En outre il semble important de rappeler que SSL est indépendant du protocole utilisé (couche supplémentaire), c'est-à-dire qui peut non seulement sécuriser des transactions effectuées sur le Web mais aussi des connexions par FTP, POP, TELNET, IMAP, SMTP, etc…

# 3. Sécurité et attaques de systèmes actuels

Bien évidemment, la qualité la plus importante que doit posséder un cryptosystème est la sécurité, c'est-à-dire qu'il doit pouvoir résister à quelconque attaque qui aurait pour but de le casser et dans extraire le texte en clair original. Bien évidemment la bête noire de la cryptographie est la cryptanalyse, mais l'on se rend vite compte que sans ces multiples attaques ennemies, la cryptographie n'aurait bien évidemment jamais autant progressé et serait restée au même stade que celle de César.

De nos jours, comme de tous temps, un système de chiffrement repose donc sur son inviolabilité. C'est pour cela que pour des systèmes aussi utilisés que RSA, DES ou SSL, la cryptanalyse de ces algorithmes est essentielle pour en trouver les failles. A l'aide de simulations (attaques), on effectue alors la cryptanalyse des algorithmes de chiffrement : on étudie sa sécurité en tentant de casser les fonctions cryptographiques qui le composent. Il est donc préférable pour un algorithme d'être diffusé (code source) pour que des cryptographes puissent mettre à l'épreuve sa sécurité.

Dans le cas des systèmes vus auparavant et bien d'autres, il semble évident que la cryptanalyse a bien aidée à renforcer leur sécurité, mais celle-ci n'en est pas pour autant devenue parfaite. En effet leur quasi invulnérabilité cède parfois devant certaines attaques.

En fait, les attaques sur un système de chiffrement ne touchent souvent pas les algorithmes de chiffrement en cause. Il peut s'agir la plupart du temps de l'exploitation d'une erreur de conception, d'une erreur de réalisation ou d'une erreur d'installation.

On peut distinguer deux types réels d'attaque. Premièrement les attaques passives ont pour but d'intercepter un message et d'exploiter les informations contenues dans celui-ci. Cette attaque ne modifie en rien les informations interceptées ou de manières imperceptibles. Le deuxième type d'attaques est les attaques actives qui visent à ralentir, dégrader ou même empêcher la communication, d'envoyer des informations parasites dans le but de saturer des

systèmes, de modifier les informations afin de tromper le destinataire ou de faire carrément disparaître ces informations.
Les types d'attaques employés le plus couramment sont nombreux et diffèrent selon le type du système (symétrique, asymétrique, fonction de hachage, etc…).
Nous vous présentons ici une liste des types d'attaques sur les algorithmes :

- **L'attaque en force (ou *Brute force attack, Exhaustive key search attack)***
  Le cryptographe essaie toutes les combinaisons de clefs possibles jusqu'à l'obtention du texte clair. Avec des ordinateurs de plus en plus performants et des méthodes de calculs distribués, l'attaque en force restera toujours un moyen de casser des systèmes de chiffrement.

- **L'attaque à l'aide de l'analyse statistique (ou *Statistical analysis attack)***
  Le cryptographe possède des informations sur les statistiques du message clair (fréquences des lettres ou des séquences de lettres). Les systèmes tels que ceux par substitution ne résistent pas à une telle attaque.

- **L'attaque à l'aide de textes chiffrés seulement (ou *Ciphertext-only attack)***
  Le cryptographe dispose de messages chiffrés par l'algorithme et fait des hypothèses sur le texte clair (présence d'expressions, de mots, le sens du message, format ASCII etc.). Il peut grâce à cela soit retrouvé les textes en clair, soit retrouver la clef.

- **L'attaque à l'aide de textes clairs (ou *Known-plaintext attack)***
  Le cryptographe dispose des messages ou parties de message clairs et de leur version chiffrée. Le but du cryptographe est alors de retrouver la clef. Ce type d'attaque est très répandu.

- **L'attaque à l'aide de textes clairs choisis (ou *Chosen-plaintext attack)***
  Le cryptographe dispose des messages clairs et de leur version chiffrée. Il a aussi la possibilité de tester des messages et d'obtenir le résultat chiffré. Les chiffrements asymétriques sont notamment vulnérables à cette attaque.

- **L'attaque d'une tierce personne (ou *Man-in-the-middle attack)***
  Cette attaque plus communément appelée l'attaque de «l'homme du milieu» intervient dans une transaction entre deux personnes(groupes…). Une troisième personne s'interpose de manière transparente entre les deux et termine la transaction normalement en captant les messages et en transmettant d'autres messages. Il peut donc ainsi intercepter et même modifier les messages envoyés sans que les deux entités s'en aperçoivent. Cette attaque peut être évitée avec les signatures digitales.

- **L'attaque à l'aide du temps d'opération (ou *Timing Attack)***
  Cette méthode est basée sur la mesure du temps nécessaire pour effectuer des chiffrements ou des déchiffrements. Ainsi cette étude permet de mieux cibler la longueur de la clef utilisée et a donc pour but de limiter grandement le domaine des clefs à explorer pour une cryptanalyse classique.

## 3.1 RSA

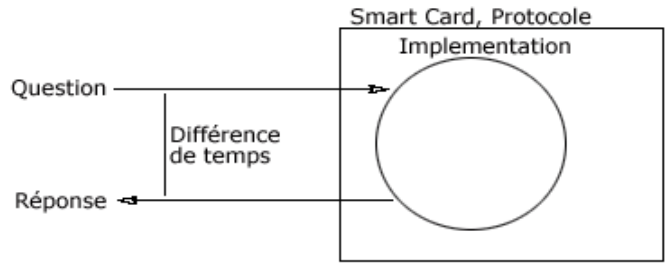
RSA est probablement une méthode de chiffrement assez sûre: il y a très certainement plus de risques liés à une mauvaise utilisation qu'à une attaque, mais les risques issus de cette dernière ne sont pourtant pas nuls. Nous rappelons que par exemple lors d'une attaque d'un système de type RSA, l'objectif d'un attaquant n'est pas forcément de retrouver la clef privée associée à la clef publique. Le plus souvent l'attaquant cherche seulement à retrouver le message clair à partir d'un message chiffré ou même seulement un bit ou une partie du message.

Bien que RSA soit un des cryptosystèmes les plus sûrs de nos jours, il n'en demeure malheureusement pas moins vulnérable à certaines attaques comme l'attaque de «l'homme du milieu» ou la Timing attack.

L'attaque de « l'homme du milieu » peut être employé de façon évidente pour casser cet algorithme. Supposons dans notre cas qu'Edgar soit un espion et qu'il veuille intercepter les conversations entre Alice et Bob. Il suffit alors à Edgar de choisir une clef privée b et une clef publique qui convient (a, n) qu'il envoie à Alice en lui faisant croire que ce sont les nouvelles clefs de Bob. Notre espion peut alors lorsqu'il intercepte les messages d'Alice, les déchiffrer et les comprendre. Il lui reste donc à s'assurer que Bob ne se rende compte de rien en re-chiffrant les messages d'Alice avec l'ancienne clef publique de Bob qui reçoit le message comme si aucune opération n'avait été effectuée dessus. S'il le veut il peut effectuer la même opération en sens inverse, pour récupérer les messages que Bob envoie à Alice. Pour contrer cette attaque, il est très important pour Alice et Bob de toujours s'assurer que personne ne tente de modifier leurs messages en effectuant plusieurs vérifications. Pour cela, les méthodes les plus utilisées sont celles d'authentification et de signature que nous expliquerons plus en détail dans un chapitre ultérieure. Ces méthodes permettent de s'assurer que le document reçu n'a pas été modifié en cours de route, et qu'il provient bien de l'expéditeur initial.

De plus on a vu que le principe de base de RSA était la décomposition d'un nombre n en produit de 2 nombres entiers p et q. Une attaque assez évidente serait de tenter de trouver la factorisation de n. Cependant les plus puissants algorithmes existants ne peuvent effectuer cette factorisation que pour des entiers inférieurs à 155 chiffres (soit une clef de 512 bits). Il est donc évident que la sécurité du RSA repose sur la difficulté de factoriser de grands entiers ; car il est simple, pour garantir une grande sécurité, de choisir de plus grandes clefs (par exemple de 1024 ou 2048 bits). Malheureusement on ne peut pas affirmer que cette simple protection suffise, car la constante amélioration des ordinateurs et des algorithmes de factorisation permettront peut être plus tôt que prévu de casser RSA.

Enfin, il existe un nouveau type d'attaque qui semble plus percutante et plus novatrice que les autres, la Timing attack. En effet comme nous l'avons vu, le fonctionnement de RSA est basé sur la fonction $y^x$ mod n servant à déchiffrer le texte y codé reçu. L'entier n est rendu publique et le texte y a pu être intercepté par un attaquant qui aura alors pour but de trouver la clef secrète x. Pour chaque message y que reçoit la victime, elle devra donc calculé $y^x$ mod n. Il est alors facile pour l'attaquant d'enregistrer les messages reçus par la cible et de mesurer le temps mis pour répondre à chaque message codé y.



L'attaquant peut ensuite recueillir les temps de calcul pour un grand nombre de messages et dresser un tableau associant un message avec le temps de décryptage associé (voir tableau ci-dessous).

| Y1 | Y2 | Y3 | … | Yi | … | Yn |
|---|---|---|---|---|---|---|
| 137ms | 154ms | 132ms | … | 170ms | … | 147ms |

Ainsi en dressant cette liste de mesure et en l'exploitant, l'on peut restreindre de beaucoup l'intervalle dans lequel on recherche la clef, car il est évident que pour des clefs d'une telle longueur, il faut explorer une immensité de cas différents. Cependant cette attaque n'est pas infaillible, car il est sur que RSA ne serait pas encore si utilisé de nos jours si une attaque avait d'aussi grandes chances de « casser » à chaque tentative ce système. La solution assez évidente pour parer ce genre d'attaque est de choisir un nouvel exposant x pour chaque opération, l'attaque ne fonctionne donc plus.
De nombreuses attaques maintenant assez répandues fonctionnent contre le cryptosystème RSA, mais la grande sécurité de ce système permet d'y remédier. C'est pour cela que comme nous l'avons dit précédemment les failles de RSA pourraient plutôt provenir de la manière de l'utiliser et non de ces failles de sécurités en elle-même. RSA en demeure un des systèmes les plus sûrs de nos jours.

## 3.2 DES

Le DES l'un des cryptosystèmes les plus connus à clefs privées, n'est tout autant que RSA pas invulnérables à toutes les attaques qu'il peut subir régulièrement. En effet de nombreuses attaques sont déjà arrivées par le passé à venir à bout des algorithmes qui composent le DES, malgré le fait qu'il était considéré comme l 'un des moyens de chiffrement les plus sûrs. Mais les attaques incessantes qu'il subissait, ont eu raison de DES et ont montré au grand jour ses faiblesses.
L'attaque qui fonctionne le mieux sur DES est l'attaque en force qui consiste, comme nous l'avons vu, à essayer d'appliquer toutes les combinaisons de clefs envisageables sur le texte chiffré, jusqu'à obtenir le texte en clair. Cette méthode semble assez « primaire », mais elle a pourtant permis en 1999 à l'aide d'un grand nombre d'ordinateurs travaillant simultanément de casser le DES en seulement 22 heures. Ceci dit l'attaque de type force brute ne se révèle pas tout le temps aussi fructueuse que cela, et il faut dans certains cas plusieurs mois voire plusieurs années pour passer toutes les clefs.
Les attaques par force brute ne sont pas les seuls à fonctionner contre DES. Les attaques à l'aide du texte clair fonctionnent également comme contre beaucoup de cryptosystèmes actuels. Cette attaque consiste en une « cryptanalyse linéaire », c'est à dire que l'attaquant essai de modéliser sous forme d'approximation linéaire les algorithmes conduisant aux textes chiffrés. Avec un grand nombre de textes clairs et de textes chiffrés, on peut obtenir certains bits de la clef. Cette attaque est actuellement l'une des plus performante, puisqu'elle ne nécessite que $2^{43}$ couples de textes (clairs et chiffrés). Grâce à une telle cryptananalyse une clef DES a été trouvée en seulement 50 jours avec l'aide d'une douzaine de machines.

## 3.3 SSL

Le protocole SSL comme beaucoup des cryptosystèmes modernes est la cible de nombreuses attaques qui ont pu tester régulièrement les points forts de sa haute sécurité. En effet, SSL a été configuré de sorte à contrer toutes ces attaques. Nous allons toutefois tenter de dresser la liste des différentes attaques qui pourraient être utilisées contre le protocole SSL. Nous retrouvons ici les fameuses « attaques à textes clairs » et attaques « de l'homme du milieu » vues précédemment mais aussi la moins connue attaque replay .
Les attaques à "texte clair" ont lieu lorsque l'attaquant a une idée du message chiffré qui est envoyé, qu'il connaît une ou plusieurs parties du message. Il va chercher à retrouver la ou les clefs utilisées pour chiffrer, ou un algorithme permettant de déchiffrer d'autres messages chiffrés avec ces mêmes clefs. L'attaquant peut générer une liste (table de hachage) dont les clefs sont les valeurs chiffrées du texte clair et dont les valeurs sont les clefs de session. Une fois que l'attaquant connaît la clef de session, il peut déchiffrer le message entier. SSL est assez vulnérable à ce genre d'attaques. Le moyen d'y parer est d'utiliser des clefs de session très grandes. D'abord le client devra générer une très grande clef et envoyer une partie en clair, une partie en cryptée au serveur. Le serveur concatènera alors les deux parties pour retrouver la clef d'origine. En utilisant une clef d'une longueur de 128bits l'utilisateur de SSL peut se prémunir de ce type d'attaque, car déchiffrer une telle clef est un processus beaucoup trop coûteux en temps et en moyens financiers.
L'attaque classique de l'homme du milieu est aussi utilisée pour tenter de casser un protocole SSL. Ce type d'attaque peut se présenter lorsque trois personnes sont dans une session de communication SSL : le client, le serveur et l'espion. L'espion se situe entre le client et le serveur et intercepte les communications entre les deux. Son but est de prendre la place du serveur aux yeux du client. Malheureusement pour l'espion, si le client prend bien soin d'effectuer la procédure d'identification du serveur spécifique à SSL, cette attaque a toutes les chances d'échouer. Pour ce faire, il doit vérifier le certificat du serveur en regardant la signature et en vérifiant si le nom du donneur de certificat est reconnu. Pour résumé, si l'homme du milieu fournit un faux certificat, la vérification de la signature le détectera. . Si la signature n'est pas authentifiée par le serveur, la signature passera mais alors c'est la vérification du nom du serveur qui détectera l'intrusion. Enfin, si la faux certificat n'est pas détecté, Edgar (qui est notre espion ne l'oublions pas) ne possède pas la clef privée nécessaire à coder correctement la réponse au client. Une attaque par cette méthode est donc rendue extrêmement difficile, voire impossible par le fonctionnement de SSL.

Enfin SSL peut être sujet à un troisième type d'attaque : l'attaque replay**.** Cette méthode d'attaque est simple. Un attaquant enregistre une session de communication entre un serveur et un client. Ensuite il se reconnecte au même serveur en réutilisant les messages qu'il vient d'enregistrer du client. Mais SSL se protège de ce genre d'attaque en attribuant à chaque connexion un id (une « nonce ») qu'il est impossible de connaître à l'avance (car elle est basée sur une série d'évènements aléatoires). Même avec des moyens matériels énormes il aurait beaucoup de difficulté, en enregistrant de nombreuses sessions, à trouver une nonce correcte.
Il paraît donc évident que SSL est difficilement attaquable. Le fait simple de choisir une très grande clef décourage un grand nombre d'attaques étant donné le temps et l'argent nécessaires pour la mener à bien.

# 4. Fonctions de hachage, signatures et certificats électroniques

## 4.1 Fonctions de hachage

Lors d'échanges de messages cryptés, il est important de pouvoir s'assurer que le message n'a pas été altéré ou modifié par un tiers pendant l'envoi. Les fonctions de hachage permettent alors de s'assurer de l'intégrité du message.

### 4.1.1 Principe

Une fonction de hachage calcule l'empreinte y (ou digest) d'un message x. Cette fonction F doit être une fonction à sens unique c'est-à-dire qu'il doit être facile de trouver y à partir de x, mais très difficile de trouver x à partir de y. Elle doit aussi être très sensible pour qu'une petite modification du message entraîne une grande modification de l'empreinte. En envoyant le message accompagné de son empreinte, le destinataire peut ainsi s'assurer de l'intégrité du message en recalculant le résumé à l'arrivée et en le comparant à celui reçu. Si les deux résumés sont différents, cela signifie que le fichier n'est plus le même que l'original : il a été altéré ou modifié par une tierce personne.
Les fonctions de hachage les plus répandus sont MD5 et SHA-1 qui sont basés tous les deux sur MD4, MD5 générant des empreintes de 128 bits et SHA-1 de 160 bits(seul MD5 sera décrit, ces deux fonctions ayant un fonctionnement similaire).
Nous verrons plus en avant qu'une fonction de hachage joue un rôle dans la signature électronique, méthode qui permet d'authentifier l'expéditeur.

### 4 .1.2 MD5

MD5 (« Message Digest ») est un des plus connus algorithme de hachage. C'est une version améliorée de MD4 tous deux conçus par Ron Rivest, un des créateurs de RSA. MD5 fabrique une empreinte d'une taille de 128 bits.

**Padding**

Soit un message m d'une longueur de n bits. MD5 manipulant des blocs de 512 bits, l'algorithme complète le message avec un 1 suivi d'autant de 0 que nécessaires jusqu'à ce que la longueur de message soit congrue à 448 modulo 512. L'opération de padding a toujours lieu même si la longueur du message est déjà congrue à 448 modulo 512.

**Ajout de la taille**

On ajoute à ce message la valeur de n, codée en binaire sur 64 bits. On obtient donc un message dont la longueur est un multiple de 512 bits.
Chaque bloc de 512 bits est décomposé en 16 blocs de 32 bits.

**Initialisation**

MD5 prend 4 tampons de 32 bits en entrée initialisés de la manière suivante (en hexadécimal) :

A=01234567
B=89abcdef
C=fedcba98
D=76543210

**Rondes**

MD5 est composé de quatre rondes qui exécutent chacune 16 opérations. Pour chaque ronde, une seule fonction prenant 3 arguments codés sur 32 bits et renvoyant une valeur sur 32 bits est utilisée pour les 16 opérations. Les 4 fonctions sont les suivantes :

F(X,Y,Z) = (X and Y) or (not(X) and Z)
G(X,Y,Z) = (X and Z) or (Y and not(Z))
H(X,Y,Z) = X xor Y xor Z
I(X,Y,Z) = Y xor (X or not(Z))

Pour chaque bloc de 512 bits, on effectue les opérations suivantes :

- on sauvegarde les valeurs des tampons (A, B, C et D) dans des registres AA, BB, CC et DD.
- On calcule les nouvelles valeurs pour A, B, C et D à partir de leurs anciennes valeurs, des bits du bloc qu'on étudie et une des quatre fonctions F, G, H ou I selon la ronde.
- On effectue A=AA+A, B=BB+B, C=CC+C, D=DD+D

**Ecriture de l'empreinte**

L'empreinte sur 128 bits est obtenue en mettant bout à bout les quatre tampons finaux A, B, C et D.

## 4.2 Signatures

### 4.2.1 Principe

Les fonctions de hachage permettent de s'assurer de l'intégrité d'un message mais un autre problème se pose : comment être certain que personne n'a usurpé l'identité de l'expéditeur pour vous envoyer un message ? Ou que l'expéditeur ne va pas nier vous l'avoir envoyé ?

C'est le rôle de la signature numérique, celle-ci fournissant donc les services d'intégrité des données, d'authentification de l'origine des données et de non-répudiation.

La façon la plus simple de signer un message est d'utiliser la cryptographie asymétrique pour le chiffrer en utilisant sa clef privée : seul le possesseur de cette clef peut générer la signature et toute personne ayant accès à la clef publique correspondante peut la vérifier. Mais cette méthode est très lente et en pratique elle n'est que peu utilisée.

La méthode réellement utilisée repose non pas sur le chiffrement du message lui-même mais sur l'empreinte (empreinte issue d'une fonction de hachage comme MD5 par exemple) de celui-ci. En effet, cette méthode est beaucoup plus rapide du fait de la quantité réduite des données à chiffrer.

Une signature numérique est plus sûre qu'une signature papier car la signature change à chaque message. Elle est de ce fait inimitable (sans la connaissance de la clef secrète bien entendue).

### 4.2.2 Standard DSS

Le schéma de signature que nous décrivons ci-après est particulièrement important puisqu'il constitue la première tentative d'élaboration d'un standard de jure d'authentification et de signature digitale. Ce schéma a été sélectionné par le National Institute of Standards and Technology pour les transactions gouvernementales américaines. Son adoption comme standard officiel au niveau des autorités fédérales (Federal Information Publication-Standards Publication 186 du 19 mai 1994) fait encore l'objet d'un débat d'opinion assez vif aux Etats-Unis.

DSS se base sur l'algorithme DSA que nous allons décrire ci-dessous.

**DSA (Digital Signature Algorythm)**

DSA se fonde sur le problème du logarithme discret, le seul qui paraisse aujourd'hui pouvoir rivaliser avec le système RSA, basé lui, sur la difficulté extrême de factoriser des grands nombres. Il utilise les paramètres suivants :

- p est un nombre premier de l'intervalle $]2^{L-1}, 2^{L}[$ où L est un nombre compris entre 512 et 1024 et est un multiple de 64.
- q est un diviseur premier de p – 1 compris d'une longueur de 160 bits ($2^{159}<q<2^{160}$).
- $g = h^{(p-1)/q} \pmod p$, où $1<h<p-1$ et $g>1$.
- x est un entier de l'intervalle ]0, q[ .
- $y = g^{x} \pmod p$
- k est un entier aléatoire de ]0, q[ .

Les entiers p, q et g sont publics. Chaque utilisateur a une clef secrète x et une clef publique y. Le nombre k doit être régénéré à chaque signature.
Si Alice souhaite envoyer le message M à Bob, alors Alice calcule r et s à partir des nombres p, q, g(publics) et de sa clef privée (x , k) de la manière suivante :

- $r = (g^{k} \bmod p) \bmod q$
- $s = (k^{-1} (SHA(M) + x^{r})) \bmod q$
  avec $k^{-1}$ l'unique entier de ]0,q[ tel que $k^{-1}k = 1$(modulo q).
  SHA (M) est la conversion en entier d'une chaîne de 160 bits empreinte du message M issu de l'algorithme SHA-1.

Bob reçoit alors les quantités M', r', s' et dispose de l'identité d'Alice et de sa clef publique y. Pour vérifier la signature du message, Bob effectue les opérations suivantes :

- si $0 < r' < q$ et /ou $0 < s' < q$ n'est (ne sont) pas vérifiées alors la signature n'est pas valide.
- Bob calcule ensuite :
  - $w = (s')^{-1} \bmod q$
  - $u_1 = ((SHA(M')) w) \bmod q$
  - $u_2 = ((r') w) \bmod q$
  - $v = ((g^{u1} y^{u2}) \bmod p) \bmod q$

Si v = r', alors la signature est vérifiée et Bob est (quasi) sûr que le message a bien été envoyé par Alice.
Dans le cas contraire, le message doit être considéré comme invalide et il y a 3 possibilités :

1. le message M a été modifié.
2. le message M a été signé de manière incorrecte par Alice.
3. le message a été signé par une autre personne (intrus).

### 4.2.3 Standard PKCS

Les PKCS (Public-Key Cryptography Standards) sont des spécifications issues des travaux d'une compagnie américaine, RSA Data Security, en coopération avec le Massachusetts Institute of Technology et un consortium de compagnies informatiques incluant Apple, Microsoft, DEC, Lotus, Sun Microsystems. Elles ont été développées dans le but d'accélérer le déploiement de la cryptographie à clef publique.
Les différents PKCS sont :

- PKCS#1 : RSA Cryptography Standard
- PKCS#2 : obsolète, inclus dans PKCS1
- PKCS#3 : Diffie-Hellman Key Agreement Standard
- PKCS#4 : obsolète, inclus dans PKCS1
- PKCS#5 : Password-Based Cryptography Standard
- PKCS#6 : Extended-Certificate Syntax Standard
- PKCS#7 : Cryptographic Message Syntax Standard
- PKCS#8 : Private-Key Information Syntax Standard
- PKCS#9 : Selected Attribute Types

- PKCS#10 : Certification Request Syntax Standard
- PKCS#11 : Cryptography Token Interface Standard
- PKCS#12 : Personal Information Exchange Syntax Standard
- PKCS#13 : Elliptic Curve Cryptography Standard
- PKCS#15 : Cryptography Token Information Format Standard.

Les documents PKCS#6 à PKCS#15 sont essentiellement des documents spécifiant des syntaxes standards pour différentes opérations liées à la cryptographie. Nous ne décrirons ici que la spécification PKCS#1 car elle est la seule à décrire un standard de signature.
Pour correspondre au standard PKCS#1, une signature numérique doit avoir été constituée de la manière suivante :

1. Une empreinte du message doit être fabriquée en utilisant l'un des algorithmes de compression MD2, MD4, MD5 ou SHA-1.
2. L'empreinte doit ensuite être chiffré avec la clef secrète RSA du signataire.

C'est l'ensemble formé du message et de l'empreinte chiffrée qui constitue la signature digitale.

## 4.3 Certificats

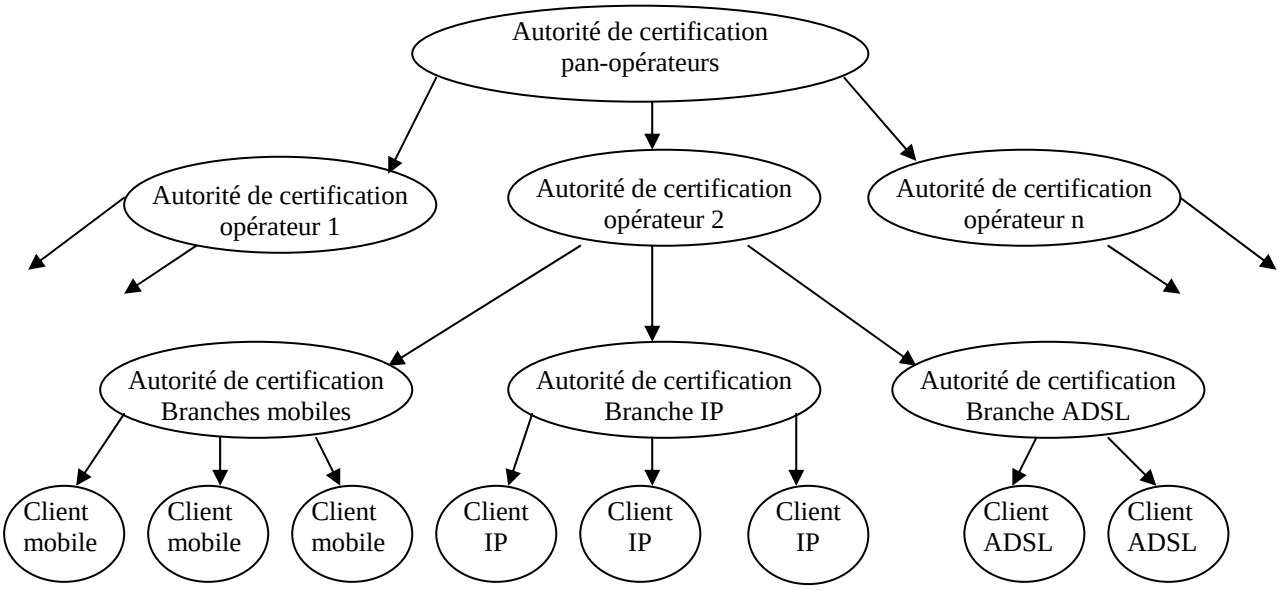
### 4.3.1 Principe

**Infrastructure à clef publique (ICP)**

L'infrastructure à clef publique définit un ensemble de services de sécurité afin de rendre les échanges électroniques fiables.
A l'intérieur de l'ICP, les éléments sont organisés hiérarchiquement dans le but de garantir un niveau de sécurité élevé. Elle est composée de :

- une autorité d'enregistrement (Registration Authority)
- une autorité de certification
- un système de distribution des clefs

Les ICP sont évolutives et interopérables c'est-à-dire qu'elles sont capables de suivre la croissance du nombre d'utilisateurs et doivent supporter l'ajout de nouvelles autorités de certification et l'établissement de certification croisée entre plusieurs autorités.



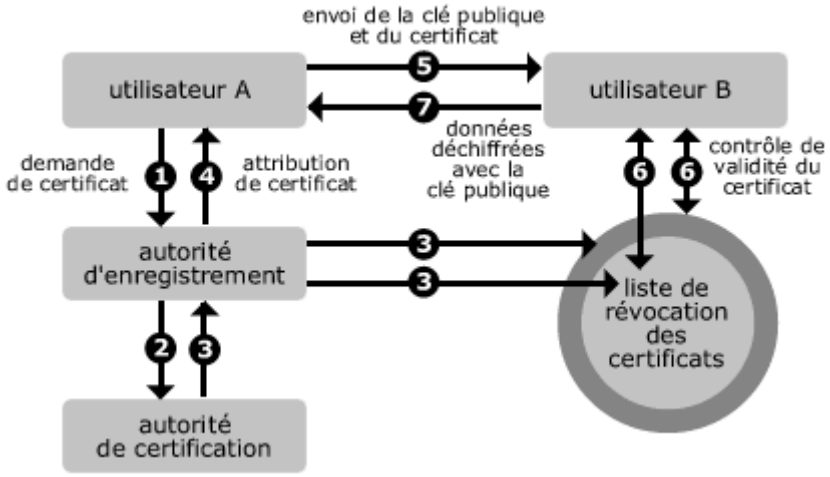

**Relation hiérarchique dans une ICP simple**

Le problème qui a amené à créer les certificats est l'opposé de celui qui a amené à créer les signatures numériques. Un certificat permet d'attester l'identité du destinataire de la même façon qu'un papier d'identité permet d'identifier une personne. En effet, les certificats numériques fonctionnent sur le même principe : ils sont émis par un organisme supérieur (l'Etat pour le papier d'identité) appelés autorité de certification.
Une autorité d'enregistrement est un organisme approuvé pour vérifier que les autres organismes sont bien ce qu'ils prétendent être. Pour cela elle applique des procédures d'identification conformes aux règles définies par l'autorité de certification.

**Obtenir un certificat numérique**

Pour obtenir un certificat d'une autorité d'enregistrement, il faut donc fournir à celle-ci des informations (documents officiels) prouvant votre identité ou celle de votre organisation ainsi que votre clef publique. L'autorité vérifie alors votre identité mais aussi que la clef fournie est bien issue d'un algorithme de cryptage à clef publique particulier.
Il existe trois classes de certificats correspondant à différents niveaux de sécurité :

- certificats de classe 1 : Le demandeur ne fournit qu'une adresse e-mail.
- certificats de classe 2 : Ils requièrent une preuve d'identité du demandeur.
- certificats de classe 3 : Ces certificats ne peuvent être délivrés que si le demandeur est présent physiquement.




**Accéder aux informations du certificat**

Une fois le certificat racine de l'autorité de certification intégré au navigateur Internet, le destinataire peut déchiffrer celui-ci et identifier son auteur en comparant les deux condensés(condensé issu de la signature et condensé calculé grâce à une fonction de hachage – le plus souvent MD5 ou SHA-1).

**Cycle de vie d'un certificat**

Pour des raisons de sécurité, un certificat est accordé pour une durée limitée. Il peut être également remis en question dans la période de validité pour différentes raisons :

- volonté du détenteur du certificat
- changement de situation du détenteur
- volonté de l'autorité de certification
- sécurité

Le certificat est alors suspendu ou révoqué, la suspension ou la révocation étant notifié dans un annuaire spécifique facilement accessible (en ligne).

### 4.3.2 Certificats X.509

Le format de certificats numériques X.509 de l'ISO (Organisation Internationale de Normalisation) est le plus répandu. Un certificat X.509 est composé de la signature de l'autorité de certification et d'informations.
Les informations contiennent :

- la version de la norme X.509 désigné par un nombre entier : 0 pour la version 1, 1 pour la version 2 et 2 pour la version 3.
- le numéro de série : chaque certificat a un numéro unique.
- l'algorithme de signature utilisé par l'autorité : MD5, DSA,etc…
- la période de validité du certificat
- le nom de l'autorité de certification
- le nom du sujet
- renseignements sur la certification de la clef publique :
  - algorithme utilisé
  - chaîne de bits représentant la clef publique
- informations optionnelles spécifiques à la version 3

La version 3 de X.509 est la version actuelle, mais toutes les versions sont utilisées. La version 3 donne la possibilité d'ajouter des extensions personnalisées (optionnelles) aux certificats.

```
Certificate:
  Data:
    Version: v3 (0x2)
    Serial Number: 3 (0x3)
    Signature Algorithm: PKCS #1 MD5 With RSA Encryption
    Issuer: OU=Ace Certificate Authority, O=Ace Industry, C=US
    Validity:
      Not Before: Fri Oct 17 18:36:25 1997
      Not After: Sun Oct 17 18:36:25 1999
    Subject: CN=Jane Doe, OU=Finance, O=Ace Industry, C=US
    Subject Public Key Info:
      Algorithm: PKCS #1 RSA Encryption
      Public Key:
        Modulus:
          00:ca:fa:79:98:8f:19:f8:d7:de:e4:49:80:48:e6:2a:2a:86: [...]
    Extensions:
          Identifier: Certificate Type
            Critical: no
            Certified Usage:
              SSL Client
          Identifier: Authority Key Identifier
            Critical: no
            Key Identifier:
              f2:f2:06:59:90:18:47:51:f5:89:33:5a:31:7a:e6:5c:fb:36:
              26:c9
    Signature:
      Algorithm: PKCS #1 MD5 With RSA Encryption
      Signature:
        6d:23:af:f3:d3:b6:7a:df:90:df:cd:7e:18:6c:01:69:8e:54:65:fc:06: [...]
```

**Exemple de certificat X.509 version 3**

# CONCLUSION

Nous avons vu un panel de méthodes de chiffrement de l'antiquité à nos jours, les attaques existantes sur les cryptosystèmes actuels les plus utilisées et les moyens inventés pour s'assurer de l'intégrité, de l'authentification de l'expéditeur et du destinataire d'un message.

Ainsi, la cryptographie est une science en perpétuelle évolution, la cryptanalyse aidant à trouver les failles d'un système pour toujours avancer. Cette évolution est importante car la cryptographie joue un grand rôle dans la sécurité internationale, tout étant aujourd'hui informatisé.

Pourtant, même si la cryptanalyse permet de faire avancer la cryptographie avec des méthodes de chiffrement et une technologie toujours plus poussées, elle représente aussi un danger à l'échelle internationale. En effet, qu'adviendrait-il demain si un mathématicien découvrait une vérité mathématique permettant de casser les algorithmes RSA et ElGamal par exemple ? Toute la sécurité informatique serait remise en question et ce que nous connaissons aujourd'hui tels que les sites de vente en ligne basées sur ces algorithmes ne fonctionneraient plus. Par conséquent, c'est non seulement la sécurité internationale qui serait touchée, mais aussi toute une économie.

C'est pourquoi tout est mis en œuvre pour assurer la sécurité de demain avec des perspectives telle que la cryptographie quantique, théoriquement incassable puisqu'elle serait une technique similaire au chiffre de Vernam (cryptosystème par flots) où la clé est aussi longue que le message.

# REFERENCES BIBLIOGRAPHIQUES


1. http://www.commentcamarche.net/crypto/crypto.php3
2. http://www.bibmath.net/crypto/
3. http://www.securiteinfo.com/
4. http://www.iro.umontreal.ca/~crepeau/CRYPTO/
5. http://fr.wikipedia.org/wiki/Cryptographie
6. http://rootshell.be/~virgil/TPE/
7. http://cui.unige.ch/tcs/cours/crypto/
8. http://www.uqtr.ca/~delisle/Crypto/
9. http://developpeur.journaldunet.com/tutoriel/sec/020905_ssl3.shtml