COURS DE CRYPTOGRAPHIE

ISIG

2021 -2022

Notes rassemblées par Gabriel BOMBAMBO Boseko
Courriel : ***gbombambo@gmail.com*** / Twitter : ***@gbboseko*** / GPG-Key : ***7C4AB1AD*** /
Tél : ***+243 81 567 99 85 / +243 84 077 91 60 / +243 997 555 918***

CHAPITRE I : INTRODUCTION ET EXEMPLES HISTORIQUES

0. INTRODUCTION A LA CRYPTOGRAPHIE

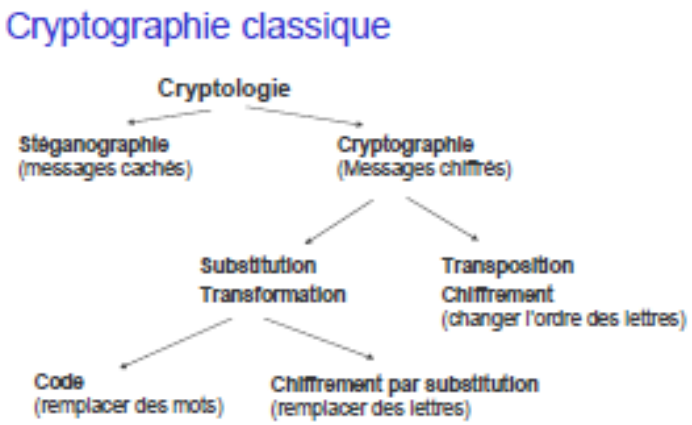
La cryptographie, ou **art de chiffrer**, coder les messages, est devenue aujourd'hui une science à part entière. Au croisement des **mathématiques**, de l'**informatique**, et parfois même de la **physique**, elle permet ce dont les civilisations ont besoin depuis qu'elles existent : le maintien du secret. Pour éviter une guerre, protéger un peuple, il est parfois nécessaire de cacher des choses...

La cryptographie étant un sujet très vaste, ce cours se focalisera essentiellement sur les méthodes de chiffrement dites *modernes*, c'est-à-dire celles étant apparues et utilisées après la Seconde Guerre mondiale. On passera en revue la saga du **DES** et de l'**AES**, en passant par le fameux **RSA**, le protocole le plus utilisé de nos jours. Ayant longtemps été l'apanage des militaires et des sociétés possédant de gros moyens financiers, la cryptographie s'est au fil du temps ouverte au grand public, et est donc un sujet digne d'intérêt. Toutes les méthodes de cryptographie seront présentées dans leur ordre chronologique d'apparition.

Notez cependant que ce cours ne s'intitule pas *cryptologie* ! L'amalgame est souvent fait entre cryptographie et cryptologie, mais la différence existe bel et bien. La cryptologie est la "science du secret", et regroupe **deux branches** : d'une part, la **cryptographie**, qui permet de **coder les messages**, et d'autre part, la **cryptanalyse**, qui permet de les **décoder.**

En effet, la **cryptologie** est une science mathématique qui comporte 2 branches : la **cryptographie** et la **cryptanalyse**.

La **cryptographie classique ou traditionnelle** est l'étude des méthodes permettant de transmettre des données de manière confidentielle. Afin de protéger un message, on lui applique une transformation qui le rend incompréhensible; c'est ce qu'on appelle le **chiffrement**, qui, à partir d'un texte clair, donne un **texte chiffré** ou **cryptogramme**. Inversement, le **déchiffrement** est l'action qui permet de reconstruire le texte en clair à partir du texte chiffré.



Chiffrement par substitution

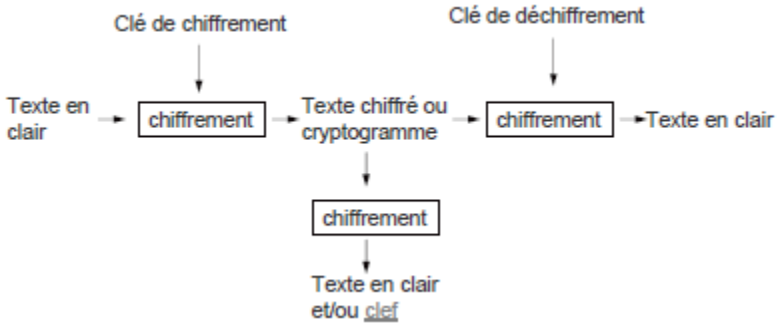
Les substitutions consistent à remplacer des symboles ou des groupes de symboles par d'autres symboles ou groupes de symboles dans le but de créer de la confusion.

Chiffrement par transposition

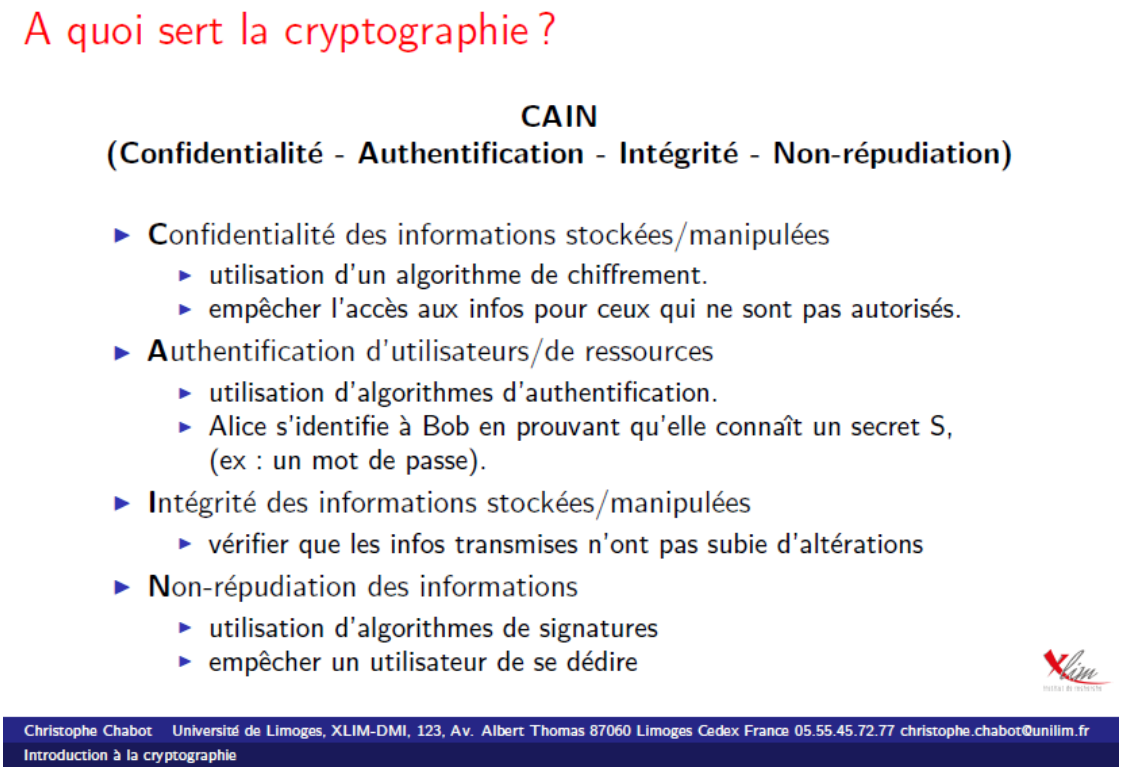
Les transpositions consistent à mélanger les symboles ou les groupes de symboles d'un message clair suivant des règles prédéfinies pour créer de la diffusion. Ces règles sont déterminées par la clé de chiffrement. Une suite de transpositions forme une permutation.

Dans la **cryptographie moderne**, les transformations en question sont des fonctions mathématiques, appelées **algorithmes cryptographiques**, qui dépendent d'un paramètre appelé **clef**.

La **cryptanalyse**, à l'inverse est l'étude des procédés cryptographiques dans le but de trouver des faiblesses et, en particulier, de pouvoir décrypter des textes chiffrés. Le **décryptement** est l'action consistant à retrouver le texte clair sans connaître la clef de déchiffrement.



Remarque: Deux éléments sont présents plusieurs fois dans ce cours. D'abord, l'utilisation des *bits*. A l'heure actuelle, la cryptographie étant quasi indissociable de l'informatique, il est souvent intéressant de travailler sur les nombres binaires. Ensuite, il sera souvent fait mention d'*Alice et Bob* : en cryptographie, plus par tradition qu'autre chose, on nomme "Alice" et "Bob" les deux interlocuteurs qui veulent s'échanger en secret des messages (sans doute pour désigner "entité _A_" et "entité _B_").



1. LE CHIFFRE DE CÉSAR

Jules César était un général, homme politique et écrivain romain, né à Rome le 12 juillet ou le 13 juillet 100 av. J.-C. et mort le 15 mars 44 av. J.-C. Il aurait été assassiné par une conspiration, son propre fils Brutus lui portant le coup de grâce. César s'est illustré lors de la guerre des Gaules, ce qui a donné des siècles plus tard son personnage dans la bande dessinée Astérix le Gaulois. Il utilisait une méthode de chiffrement qui porte aujourd'hui son nom.

Principe

Le chiffre de César est la méthode de cryptographie la plus ancienne communément admise par l'histoire. Il consiste en une **substitution mono-alphabétique** : chaque lettre est remplacée ("substitution") par une *seule* autre ("mono-alphabétique"), selon un certain décalage dans l'alphabet ou de façon arbitraire. D'après Suétone, César avait coutume d'utiliser un décalage de 3 lettres : **A** devient **D**, **B** devient **E**, **C** devient **F**, etc. Il écrivait donc son message normalement, puis remplaçait chaque lettre par celle qui lui correspondait :

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
-> décalage = 3																										
CODE	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

Exemple : d'après cette méthode, "VIVE LES MATHS" devient donc "YLYH OHV PDWKV" !

Notes rassemblées par Gabriel BOMBAMBO Boseko
Courriel : ***gbombambo@gmail.com*** / Twitter : ***@gbboseko*** / GPG-Key : ***7C4AB1AD*** /
Tél : ***+243 81 567 99 85 / +243 84 077 91 60 / +243 997 555 918***

Sécurité

Niveau sécurité, le chiffre de César n'est pas fiable du tout, et ce pour deux raisons :

- Il n'existe que 26 façons différentes de chiffrer un message : puisqu'on ne dispose que de 26 lettres, il n'y a que 26 décalages possibles. Dès lors, des **attaques exhaustives** (tester toutes les décalages un à un) ne demanderaient que très peu de temps.
- Le chiffre de César est très vulnérable à l'analyse des fréquences

Modulo

Soit $n > 2$ un entier fixé.

Définition 1.

On dit que a est congru à b modulo n, si n divise $b - a$. On note alors

$$a \equiv b \pmod{n}.$$

Pour nous $n = 26$. Ce qui fait que $28 \equiv 2 \pmod{26}$, car $28 - 2$ est bien divisible par 26. De même $85 = 3 \times 26 + 7$ donc $85 \equiv 7 \pmod{26}$.

On note $Z/26Z$ l'ensemble de tous les éléments de Z modulo 26. Cet ensemble peut par exemple être représenté par les 26 éléments $\{0, 1, 2, \ldots, 25\}$.

En effet, puisqu'on compte modulo 26 :

$0, 1, 2, \ldots, 25$, puis $26 \equiv 0, 27 \equiv 1, 28 \equiv 2, \ldots, 52 \equiv 0, 53 \equiv 1, \ldots$

et de même $-1 \equiv 25, -2 \equiv 24$,...

Plus généralement Z/nZ contient n éléments. Pour un entier $a \in Z$ quelconque, son représentant dans $\{0, 1, 2, \ldots, n-1\}$ s'obtient comme le reste k de la division euclidienne de a par n : $a = bn + k$.

De sorte que $a \equiv k \pmod{n}$ et $0 \leq 6\ k < n$.

De façon naturelle l'addition et la multiplication d'entiers se transposent dans Z/nZ.

Pour a, b ∈ Z/nZ, on associe a + b ∈ Z/nZ.

Chiffrer et déchiffrer

Le chiffrement de César est simplement une addition dans Z/26Z! Fixons un entier k qui est le décalage (par exemple k = 3 dans l'exemple de César ci-dessus) et définissons ***la fonction de chiffrement de César de décalage k*** qui va de l'ensemble Z/26Z dans lui-même :

$$C_k : \begin{cases} \mathbb{Z}/26\mathbb{Z} & \longrightarrow & \mathbb{Z}/26\mathbb{Z} \\ x & \longmapsto & x+k \end{cases}$$

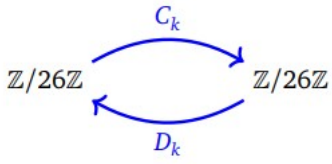
Par exemple, pour k = 3 : C_3 (0) = 3, C_3 (1) = 4. . . Pour déchiffrer, rien de plus simple ! Il suffit d'aller dans l'autre sens, c'est-à-dire ici de soustraire. ***La fonction de déchiffrement de César de décalage k*** est :

$$D_k : \begin{cases} \mathbb{Z}/26\mathbb{Z} & \longrightarrow & \mathbb{Z}/26\mathbb{Z} \\ x & \longmapsto & x-k \end{cases}$$

En effet, si 1 a été chiffré en 4, par la fonction C_3 alors D_3 (4) = 4 − 3 = 1. On retrouve le nombre original. Mathématiquement, D_k est la bijection réciproque de C_k , ce qui implique que pour tout x ∈ Z/26Z :

$$D_k\big(C_k(x)\big) = x$$

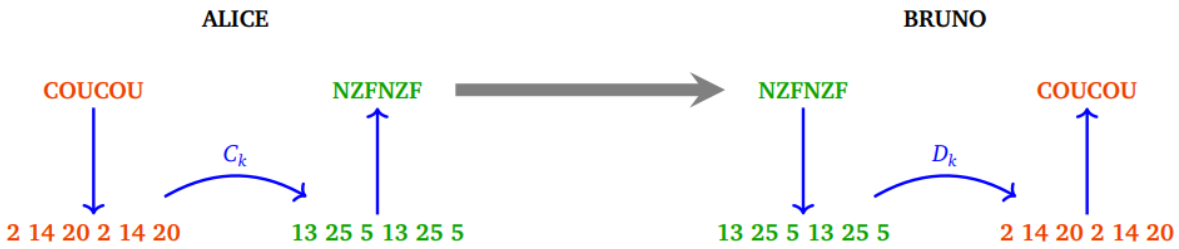
En d'autres termes, si x est un nombre, on applique la fonction de chiffrement pour obtenir le nombre crypté y = C_k (x); ensuite la fonction de déchiffrement fait bien ce que l'on attend d'elle D_k (y) = x, on retrouve le nombre original x.



Une autre façon de voir la fonction de déchiffrement est de remarquer que D_k (x) = C_{-k} (x). Par exemple C_{-3} (x) = x + (−3) ≡ x + 23 (mod 26).

Notes rassemblées par Gabriel BOMBAMBO Boseko
Courriel : ***gbombambo@gmail.com*** / Twitter : ***@gbboseko*** / GPG-Key : ***7C4AB1AD*** /
Tél : ***+243 81 567 99 85 / +243 84 077 91 60 / +243 997 555 918***

Voici le principe du chiffrement : Alice veut envoyer des messages secrets à Bruno. Ils se sont d'abord mis d'accord sur une clé secrète k, par exemple k = 11. Alice veut envoyer le message "COUCOU" à Bruno. Elle transforme "COUCOU" en "2 14 20 2 14 20". Elle applique la fonction de chiffrement $C_{11}(x) = x + 11$ à chacun des nombres : "13 25 5 13 25 5" ce qui correspond au mot crypté "NZFNZF". Elle transmet le mot crypté à Bruno, qui selon le même principe applique la fonction de déchiffrement $D_{11}(x) = x - 11$.



Espace des clés et attaque

Combien existe-t-il de possibilités de chiffrement par la méthode de César ? Il y a 26 fonctions Ck différentes, k = 0,1, . . . ,25. Encore une fois, k appartient à Z/26Z, car par exemple les fonctions C29 et C3 sont identiques. Le décalage ***k s'appelle la clé de chiffrement***, c'est l'information nécessaire pour crypter le message. Il y a donc 26 clés différentes et l'espace des clés est Z/26Z.

Il est clair que ce chiffrement de César est d'une sécurité très faible. Si Alice envoie un message secret à Bruno et que Chloé intercepte ce message, il sera facile pour Chloé de le décrypter même si elle ne connaît pas la clé secrète k. L'attaque la plus simple pour Chloé est de tester ce que donne chacune des 26 combinaisons possibles et de reconnaître parmi ces combinaisons laquelle donne un message compréhensible.

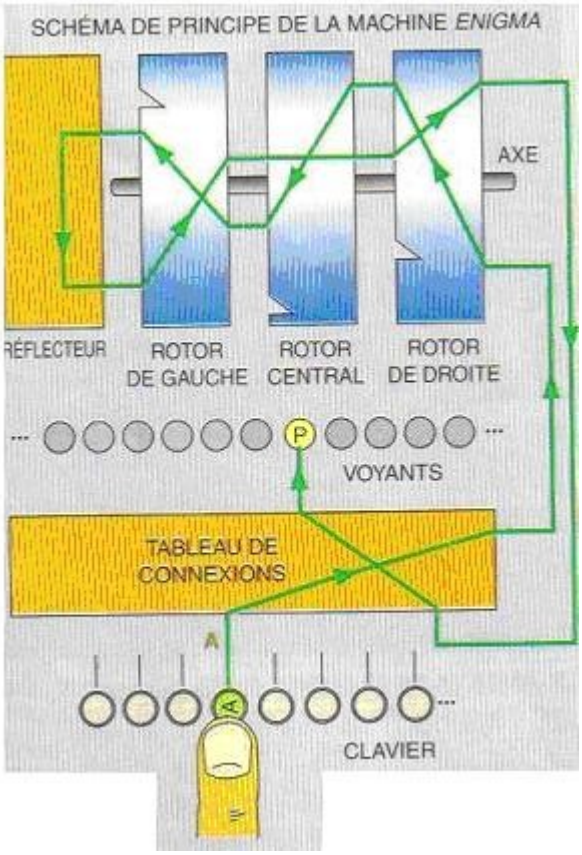
2. *LA MACHINE ENIGMA*

> Eté 1940. La guerre semble avoir choisi son camp. La Pologne, la France ont capitulé. La Grande-Bretagne résiste, mais elle dépend, pour la moitié à peu près de son approvisionnement en matières premières, des importations maritimes. Or, dans les mers, les sous-marins allemands, les redoutables U-Boot, font régner la terreur, coulant de nombreux navires. Ils attaquent de nuit, en meutes. Pour leur coordination tactique, ils échangent de nombreux messages radios, avec le commandement à terre. Ces messages sont cryptés à l'aide d'une remarquable machine, l'Enigma. (bibmath.net)

La machine allemande Enigma a joué un grand rôle pendant la guerre de l'Atlantique, et son déchiffrement par les Alliés leur a assuré bon nombre de victoires (notamment parce que les Allemands ne se doutaient pas que leurs messages étaient déchiffrés).

Enigma ressemble à une machine à écrire : on frappe le clair sur un clavier, et des petites lampes s'allument pour éclairer les lettres résultant du chiffrement.

Le principe de chiffrement qu'utilise Enigma est à la fois simple et astucieux. Simple, car il ne s'agit ni plus ni moins d'une substitution de lettres : par exemple, A devient Q, P devient N, etc. Et astucieux, parce que **la substitution change d'une lettre à une autre** : si la lettre A correspond à Q la première fois qu'on la saisit, elle pourrait correspondre à M, K, H, ou tout autre lettre différente de Q à la fois suivante (ce principe est possible grâce à un système de rotors).
De plus, un autre avantage non négligeable que possède Enigma est la réversibilité : si on tape le message clair, on obtient le message code, et avec le message codé, on obtient le message clair.
L'inconvénient majeur est que jamais la lettre A ne sera codée par A....



Notes rassemblées par Gabriel BOMBAMBO Boseko
Courriel : ***gbombambo@gmail.com*** / Twitter : ***@gbboseko*** / GPG-Key : ***7C4AB1AD*** /
Tél : ***+243 81 567 99 85 / +243 84 077 91 60 / +243 997 555 918***

3. *LE CHIFFRE DU CHE*

Ernesto Rafael Guevara de la Serna alias "Che Guevara" était un révolutionnaire marxiste d'Amérique latine, né le 14 juin 1928 à Rosario et décédé le 8 octobre 1967 à La Higuera en Bolivie. Plus précisément, il fut exécuté par l'armée bolivienne, et on retrouva sur son corps un papier indiquant le procédé qu'il utilisait pour échanger des messages avec Fidel Castro.

Première phase

Le Che utilise une substitution de lettres (par des chiffres cependant) telle que :

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
CODE	6	38	32	4	8	30	36	34	39	31	78	72	70	76	9	79	71	58	2	0	52	50	56	54	1	59

BONJOUR devient 38 9 76 31 9 52 58

Deuxième phase

Les chiffres du message sont découpés en blocs de 5 chiffres.

BONJOUR devient 38976 31952 58

Troisème phase

Le Che additionne en modulo 10* le message obtenu avec un nombre de 5 chiffres (**même taille que le message donc, c'est important**) connu de lui et de Fidel Castro seulement. Ce nombre a été choisi au préalable aléatoirement et constitue la **clé secrète**.

Choisissons par exemple **25638** comme clé secrète et chiffrons les deux premiers blocs de BONJOUR ("clair" désigne les blocs obtenus en phase 2) :

#BLOC 1#

38976 [clair]

25638 [clé]

53504 [crypté]

#BLOC 2#

31952 [clair]

25638 [clé]

56580 [crypté]

L'addition en modulo 10 signifie qu'on ne prend pas en compte les retenues : 6 + 6 = 2, 3 + 7 = 0, 8 + 9 = 7, etc.

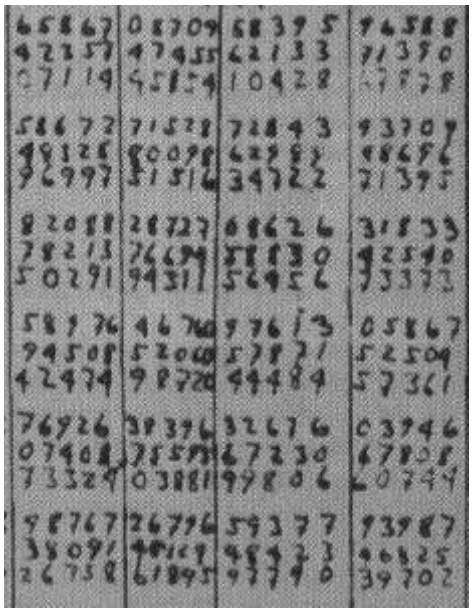
Et que fait-on du "58" isolé en fin de message ? On le groupe avec le bloc suivant, etc...

Quatrième phase

Envoi de tous les blocs chiffrés à Fidel Castro. Et destruction du brouillon ayant servi à faire les calculs, si possible ! Car quiconque mettait la main sur le papier, prenait par la même occasion connaissance de la clé, et le déchiffrement en devenait ridiculement facile : il suffisait en effet de soustraire au message chiffré la même clé.

Remarque

Le chiffre du Che est en fait un **système de Vernam**, puisque la taille de la clé est la même que celle du message.



Un brouillon utilisé par Che Guevara, on retrouve les additions telles que celles décrites plus haut.

4. Méthode utilisée par Vigenère (1586)

La clef définit le décalage pour chaque lettre du message

A : décalage de 0

B : décalage de 1

Z : décalage de 25

Le chiffrement du message "LA VIE EST BELLE" avec la clé "BONJOUR" donne le message chiffré suivant : « MO IRS YJU PRUZY »

Mathématiques

L'élément de base n'est plus une lettre mais un bloc, c'est-à-dire un regroupement de lettres. La fonction de chiffrement associe à un bloc de longueur k, un autre bloc de longueur k, ce qui donne en mathématisant les choses :

$$C_{n_1,n_2,\ldots,n_k} : \begin{cases} \mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z} \times \cdots \times \mathbb{Z}/26\mathbb{Z} & \longrightarrow & \mathbb{Z}/26\mathbb{Z} \times \mathbb{Z}/26\mathbb{Z} \times \cdots \times \mathbb{Z}/26\mathbb{Z} \\ (x_1, x_2, \ldots, x_k) & \longmapsto & (x_1 + n_1, x_2 + n_2, \ldots, x_k + n_k) \end{cases}$$

Chacune des composantes de cette fonction est un chiffrement de César. La fonction de déchiffrement est juste : $C_{-n_1,-n_2,\ldots,-n_k}$

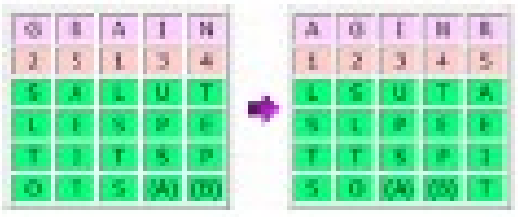
5. TRANSPOSITION

- Consiste à changer l'ordre des lettres
- Pour de très brefs messages : méthode est peu sûre car il y a peu de variantes
- Lorsque le nombre de lettres croît : impossible de retrouver le texte original sans connaître le procédé de brouillage.
- Par exemple, une phrase de 35 lettres peut être disposée de 35! = 1040 manières
- Nécessite un procédé rigoureux convenu auparavant entre les parties.

Pour de très brefs messages, comme un simple mot, cette méthode est peu sûre car il n'y a guère de variantes pour redistribuer une poignée de lettres. Par exemple un mot de trois lettres ne peut être tourné quand dans 6 (=3!) positions différentes. Ainsi col ne peut se transformer qu'en col, clo, ocl, olc, lco, loc.

Une transposition au hasard des lettres semble donc offrir un très haut niveau de sécurité, mais il y a un inconvénient: pour que la transposition soit efficace, l'ordonnancement des lettres doit suivre un système rigoureux sur lequel d'expéditeur et l'envoyeur se sont préalablement entendus.

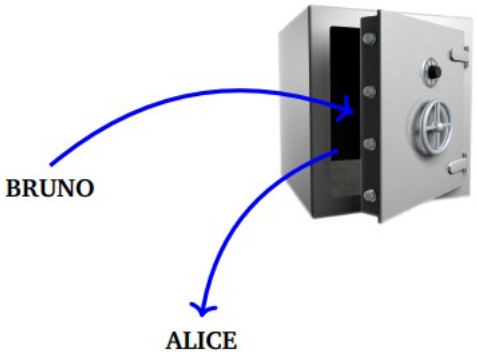
Une transposition rectangulaire consiste à écrire le message dans une grille rectangulaire, puis à arranger les colonnes de cette grille selon un mot de passe donné (le rang des lettres dans l'alphabet donne l'agencement des colonnes). Dans l'exemple ci- dessous, on a choisi comme clef GRAIN pour chiffrer 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.



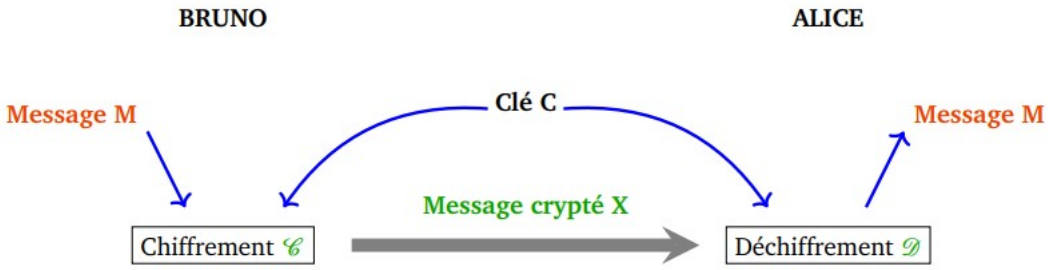
CHAPITRE II : LA CRYPTOGRAPHIE À CLÉ SECRÈTE

1. CHIFFREMENT ET SYSTÈME CRYPTOGRAPHIQUE

Petit retour en arrière. Les algorithmes étudiés dans le chapitre précédent étaient des chiffrements à clé secrète. De façon imagée, tout se passe comme si Bruno pouvaient déposer son message dans un coffre fort pour Alice, Alice et Bruno étant les seuls à posséder la clé du coffre.



En effet, jusqu'ici, les deux interlocuteurs se partageaient une même clé qui servait à chiffrer (et déchiffrer) les messages. Cela pose bien sûr un problème majeur : Alice et Bruno doivent d'abord se communiquer la clé.



Une fonction cryptographique, ou de **chiffrement**, est la donnée d'une transformation, en général *bijective* :

f : M --> C

où M est l'ensemble des messages en clair

et C est l'ensemble des messages chiffrés

Comme il s'agit d'une fonction bijective, l'opération permettant de passer de C à M est $\mathbf{f}^{-1}$, c'est le **déchiffrement**. Ce principe de chiffrement/déchiffrement est dit "**symétrique**"; car l'utilisation d'une bijection permet l'aller-retour très facilement.

L'emploi d'une bijection se justifie également parce qu'elle permet une correspondance **univoque** entre les éléments de l'ensemble. Autrement dit, à partir d'un message crypté, il n'est pas possible de tomber sur plusieurs possibilités de messages en clair.

Il est donc primordial, pour lire/écrire des messages cryptés :

- d'utiliser une fonction **facilement inversible**, en particulier pour un usage privé (il faut être capable de déchiffrer les messages sans avoir recours à des moyens techniques colossaux).

- de préserver cette fonction **secrète**, car si un "ennemi" se la procure, il lui suffira de l'inverser pour déchiffrer le message !

Ce dernier point implique un autre problème : l'utilisation d'une même fonction f pour chiffrer un message risque, à la longue, d'en dévoiler le secret. Ainsi, l'idéal serait de changer régulièrement de fonction de chiffrement.

On définit donc un système cryptographique, ou de chiffrement, ou encore un chiffre, comme étant une famille finie 'F' de fonctions 'f' cryptographiques, chacune étant déterminée par un paramètre 'K', appelé clé secrète:

$F = (f_K)$

On utilise ainsi la même "structure" de fonction de chiffrement, mais avec chaque fois un paramètre K différent.

2. *LE PRINCIPE DE KERCKHOFFS*

Plaçons-nous du point de vue du cryptanalyste, le spécialiste qui veut décoder le message. Idéalement, celui-ci doit trouver la clé 'K' et la transformation associée f_K. Cependant, la réalité est un peu différente. En vérité, si la fonction 'f' de chiffrement est utilisée à grande échelle, il est illusoire de le considérer comme secret. On suppose donc que le cryptanalyste connaît entièrement le système, mais qu'il ne sait pas quelle transformation f_K, c'est-à-dire qu'il ne connaît pas la clé secrète K !
C'est là le principe énoncé par Auguste Kerckhoffs à la fin du XIXe siècle, dans un article sur la cryptographie militaire :

La sécurité d'un cryptosystème ne doit reposer que sur le secret de la clé. Tous les autres paramètres sont supposés publiquement connus.

Ce principe est le fondement de la cryptographie à clé secrète.

3. LE CHIFFRE DE VERNAM

En 1917, Gilbert Vernam mit au point un algorithme de chiffrement -basé sur une clé secrète- **parfaitement sûr**, tellement sûr qu'il a longtemps protégé le fameux "téléphone rouge", qui reliait la Maison Blanche au Kremlin.

Utilisation :

- La clé est de la taille du message à envoyer
- Les lettres de cette clé sont choisies de façon totalement aléatoire
- La clé ne doit servir qu'une seule et unique fois

Illustration du principe par un exemple

Alice veut transmettre à **Bob** un message **M**. A l'aide de la clé secrète **K** (convenue avec B), elle va chiffrer M pour arriver à sa version cryptée **C**. Ecrivons :

C = K * M

où "*" est une loi de groupe. L'utilisation d'une loi de groupe se justifie car elle possède une propriété nécessaire : quels que soient deux nombres a et b, il existe un unique nombre x tel que a = b * x

(1) Alice choisit (par tradition et facilité) pour loi de groupe l'opérateur bit à bit "**OU EXCLUSIF**", noté **XOR**, décrit tel que :

1	XOR	1	=	0
1	XOR	0	=	1
0	XOR	0	=	0
0	XOR	1	=	1

(2) Alice va donc écrire son message sous forme binaire, puis générer une grande quantité de bits réellement aléatoires (par exemple, en tirant à pile ou face) qui vont constituer la clé. Elle pourra alors procéder au cryptage :

Soient M = 1000011 et K = 1101000,
Le message crypté C est donc : C = K XOR M = 1101000 XOR 1000011 = 0101011

(3) Alice transmet C à Bob par un canal quelconque, tel que la radio. Bob, pour obtenir le message original, va utiliser l'opération inverse de celle qui a permis le cryptage. Ici, comme XOR est son propre inverse :

$M = K (*)^{-1} C$
M = 1101000 XOR 0101011
M = 1000011, ce qui constitue bien le message de départ !

Sécurité "inconditionnelle"

Si la clé choisie est soumise aux conditions citées plus haut, l'utilisation d'une loi de groupe garantit une sécurité dite inconditionnelle. En effet, si on admet qu'un cryptanalyste intercepte le message crypté C, il ne pourra rien en déduire, si ce n'est la taille du message en clair M. **Il lui est impossbile d'établir une corrélation entre M et C sans connaître K**, car

étant donné qu'on utilise pour chiffrer une loi de groupe, il n'existe pour M et C qu'un seul nombre K tel que M = K * C !

Ce problème est comparable au suivant : dans un bateau, il y a trois poules et un canard ; quel est l'âge du capitaine ?
... Même si l'ennemi possédait des ordinateurs ultra puissants, il ne pourrait jamais en trouver la solution. C'est dans ce cas-là que le mot "sécurité inconditionnelle" prend son sens : une puissance de calcul infinie ne déchiffrerait pas le message.

Le système de Vernam

Il suit le principe détaillé précédemment. Soient :

- Un alphabet **E**, sur lequel s'écrivent les messages en clair et les messages chiffrés
- **m** le nombre de lettres de E (m = #E)

Un message M de n symboles $M = (x_1,x_2,...,x_n)$ se chiffre par la transformation f_K :

$f_K : M (x_1,x_2,...,x_n) \dashrightarrow C (y_1,y_2,...,y_n)$
où $y_i = x_i + k_i \bmod m$
avec $K = (k_i,..., k_n)$ la clé choisie aléatoirement dans E, et destinée à ne servir qu'une seule fois.

Pertinence de ce principe

Le système mis au point par Vernam, bien qu'apportant une sécurité optimale, est très peu d'usage dans le monde civil, et est surtout réservé à des organismes possédant des moyens importants. En effet, tout le monde n'est pas en mesure d'utiliser des clés de la manière décrite en début de chapitre :

- Générer des clés gigantesques (puisqu'au moins de la taille de l'ensemble des messages à envoyer) nécessite une grande puissance de calcul
- Transporter de telles clés, dont le secret doit être maintenu à tout prix, n'est pas chose aisée : tout le monde ne dispose pas de la valise diplomatique...

Il a donc fallu aux cryptographes trouver une alternative plus pratique et moins coûteuse, qui sera décrite dans le chapitre suivant, les schémas de Feistel.

Remarque : Le chiffre de Vernam est le chiffre de Vigenère avec comme caractéristique que la clef de chiffrement a la même longueur que le message clair

Exemple : Le déchiffrement du message " FVAWBTLIEBLVX" avec la clé "DECHIFFREMENT" donne le message clair suivant : « CRYPTOGRAPHIE »

4. SCHÉMAS DE FEISTEL, OU CHIFFREMENT PAR BLOCS

Plutôt que d'utiliser des clés immenses telles que pour un chiffre à usage unique, on utilise le plus souvent des algorithmes qui ont une clé secrète relativement petite (de 80 à 128 bits, mais qui utilisent cette clé d'une façon apparemment si complexe qu'il est impossible à un ennemi d'en trouver la valeur.

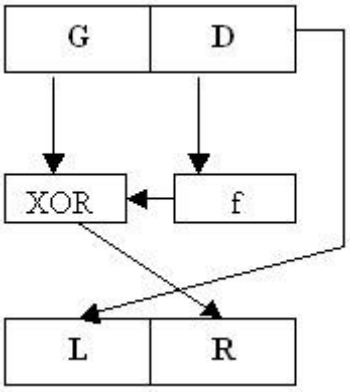
Dès lors, puisque la clé est si petite, n'est-il pas aisé d'essayer toutes les possibilités jusqu'à parvenir au décryptement ? En fait, aucun ordinateur ne pourrait y arriver en un temps réaliste. En effet, si la clé était de 128 bits, c'est-à-dire une succession de 128 "0" et "1", il y aurait 2^{128} ou 100000000000000000000000000000000000000 clés possibles !!

L'objectif est donc le suivant : élaborer à partir du message M une suite aléatoire de chiffres, ou du moins qui paraisse aléatoire, que seule la détention de la clé K permet de déchiffrer. Concrètement, il s'agit de construire une fonction bijective "**pseudo-aléatoire**" :

- Elle doit être une bijection, car elle doit faire correspondre à chaque chiffre du message en clair un chiffre du message codé ; et car elle doit permettre de pouvoir, à partir d'un chiffre C, de remonter de façon *univoque* au chiffre correspond du message en clair
- Elle doit être ou paraître aléatoire : en cryptographie, la perfection même est l'aléatoire, le message codé doit avoir l'air de découler directement du hasard, pour limiter les risques d'une attaque par analyse du texte chiffré, de ses redondances, etc.

La mise au point d'une fonction réunissant ces deux conditions posa problème aux cryptographes jusque dans les années 1950, lorsque Feistel montra qu'une fonction pseudo-aléatoire se transformait, par une méthode simple, en bijection. Actuellement, c'est la méthode de chiffrement à clé secrète la plus utilisée.

L'algorithme



Explication :

Soit une fonction f qui prend comme argument un mot de n bits.

L'algorithme de chiffrement va procéder en chiffrant des blocs de 2n bits, qu'on partage en 2 parties de n bits chacune : les parties gauche (G) et droite (D).

L'image du bloc (G,D) est le bloc (L,R), avec L=D et R = G XOR f(D).

Cette transformation est bijective, car si on a un couple (L,R), on retrouve bien (G,D) par D=L et G=R XOR f(L).

La partie droite n'a pas été transformée (juste envoyée à gauche). Il faut donc répéter le schéma de Feistel un certain nombre de fois (on parle de *tours*).

5. Applications des schémas de Feistel

Le DES

Le *Data Encryption Standard* (standard de chiffrement de données) a été publié en 1977, et fut ainsi le premier algorithme cryptographique à petite clé secrète (56 bits) à avoir été

rendu public. Le DES consiste en un réseau de Feistel de 16 tours : le message à chiffrer est découpé en blocs de 64 bits, chacun d'eux étant séparé en deux sous-blocs de 32 bits.

L'algorithme du DES sera le plus utilisé dans le monde jusqu'en 1998. A cette époque, une association de particuliers fit construire, pour moins de 250 000 $ (somme dérisoire pour un Etat ou une organisation mafieuse), un processeur capable de casser le DES. A l'heure actuelle, trois jours suffisent aux ordinateurs pour le percer, et ce grâce à des attaques exhaustives !

On aura bien tenté d'améliorer le DES, en doublant la taille de sa clé (on parle alors de TDES), mais cette version n'était pas assez rapide. Le NIST (National Institute of Standards and Technologies) lance donc un concours pour créer un successeur au DES, et ce sont les belges Joan Daemen et Vincent Rijmen qui seront retenus.

L' AES

L'Advanced Encryption Standard, ou Rijndael, est donc officialisé le 2 octobre 2000. Joan Daemen et Vincent Rijmen devaient respecter les conditions suivantes :

- Une large portabilité
- Un chiffrement rapide
- Un algorithme simple à comprendre et surtout, libre de droits

L'AES fait subir au message quatre transformations successives, agissant directement sur les octets :

- Une transformation non linéaire d'un octet
- Un décalage de lignes
- Un brouillage de colonnes
- Une addition de clés

6. *Conclusion sur la cryptographie à clé secrète*

Malgré toutes ses évolutions et ses mises en œuvre, la cryptographie à clé secrète est toujours entravée par un défaut : la condition sine qua non de son succès est et restera le secret de sa clé (principe de Kerckhoffs). Bien qu'ayant pu au fil du temps réduire sa taille, les cryptographes ont toujours été confrontés au problème de la transmission de cette clé... Mais le progrès ne s'arrête jamais ! Si le problème est de conserver le secret de la clé, pourquoi ne pas le contourner... en inventant un système qui la rend *publique* ?

Il existe des nombreux systèmes cryptographiques symétriques :

1. ***Chiffrement de flux*** : **RC4, RC5** (« Rivest's Code #4, #5 » 1987) dont une longueur de clé variable (de 1 à 256 octets).

Conçu en 1987 par Ronald Rivest, l'un des inventeurs de l'algorithme à clef publique RSA, RC4 est un générateur de bits pseudo-aléatoires, à clef secrète, considéré aujourd'hui comme insuffisamment sécurisé par les experts en cryptographie. Problème : il est toujours largement utilisé dans des protocoles comme WEP, WPA et surtout TLS, qui sécurise les échanges sur Internet, notamment en transformant HTTP en HTTPS.

En février 2015, l'Internet Engineering Task Force (IETF) recommandait de ne plus exploiter RC4 pour les échanges TLS. L'organisme, qui produit la plupart des standards Internet, notait alors qu'une des faiblesses du protocole, dévoilée en 2013, mettait en danger les communications par TLS, la faille étant exploitable avec des capacités de calcul accessibles. Les éditeurs de navigateur ont alors pris de premières mesures, limitant l'usage de RC4 aux serveurs ne proposant pas d'autres options lors de la phase de négociations entre serveur et client qui précède l'établissement d'une connexion sécurisée.

Google, Microsoft et Mozilla franchissent donc aujourd'hui un pas supplémentaire en proscrivant définitivement l'algorithme vieillissant. C'était le cas pour Chrome « en janvier ou février 2016 ». Même échéance pour Microsoft avec Edge – le navigateur de Windows 10 – et IE pour Windows 7, 8.1 et 10. Mozilla est plus précis, et va stopper le support de RC4 sur la version 44 de Firefox, apparue le 26 janvier . Le trafic sécurisé par RC4 est d'ores et déjà très faible. Selon Google, seulement 0,13 % des connexions HTTPS au sein de Chrome

(parmi les utilisateurs ayant accepté la collecte de ces statistiques) utilisent encore l'algorithme de 1987.

2. ***Chiffrement par blocs :***

- **DES** (« Data Encryption Standard » 1974), **triple-DES** (1985)

TDES ou **3DES**, une alternative au DES

En 1990 Eli Biham et Adi Shamir ont mis au point la cryptanalyse différentielle qui recherche des paires de texte en clair et des paires de texte chiffrées. Cette méthode marche jusqu'à un nombre de rondes inférieur à 15, or un nombre de 16 rondes sont présentes dans l'algorithme présenté ci-dessus.

D'autre part, même si une clé de 56 bits donne un nombre énorme de possibilités, de nombreux processeurs permettent de calculer plus de 106 clés par seconde, ainsi, utilisés parallèlement sur un très grand nombre de machines, il devient possible pour un grand organisme (un Etat par exemple) de trouver la bonne clé...

Une solution à court terme consiste à chaîner trois chiffrement DES à l'aide de deux clés de 56 bits (ce qui équivalait à une clé de 112 bits). Ce procédé est appelé Triple DES, noté TDES (parfois 3DES ou 3-DES).

- **IDEA** (« International Data Encryption Algorithm » , Suisse, 1992) contrôlé par une clé de 128 bits

L'International Data Encryption Algorithm (IDEA) a été proposé en 1992 pour remplacer le DES. Il utilise une clé de 128 bits, réalise un chiffrement par blocs de 64 bits, en opérant 8 rondes d'une même fonction.

La description de cette fonction est un peu compliquée; elle utilise seulement 3 opérations :

- ✓ Ou exclusif.
- ✓ Addition modulo 216.
- ✓ Multiplication modulo 216+1

IDEA est particulièrement adapté aux réalisations logicielles. Cet algorithme est connu essentiellement car il a longtemps constitué la partie "cryptographie à clé secrète" du célèbre logiciel PGP.

- **Blowfish** (Bruce Schneier 1993) une longueur de clé variant entre 32 bits et 448 bits

Twofish est légèrement plus lent que Rijndael mais plus rapide que les autres finalistes d'AES. Il surpasse Rijndael avec une clé de 256 bits. Twofish a été conçu pour être implanté dans des cartes à puce et d'autres systèmes embarqués. Sur un Pentium, une implémentation optimisée en assembleur permet de chiffrer un bloc de 128 bits en 18 coups d'horloge (16,1 coups d'horloge sur un Pentium Pro).

En 2005, aucune attaque n'a pu être appliquée sur la version complète de Twofish. La recherche exhaustive reste le seul moyen pour le casser. Il semble en tout cas plus résistant que ce qui avait été initialement annoncé durant le concours AES. De par sa complexité, la cryptanalyse de cet algorithme reste délicate.

Ses concepteurs ont eux-mêmes publié des attaques sur des versions à 6 et 7 tours. Une attaque sur 5 tours a une complexité de 251. Malgré ses atouts, il reste relativement peu utilisé et a été supplanté par le gagnant d'AES, Rijndael. Il n'en demeure pas moins une alternative séduisante à l'actuel AES si celui-ci devenait vulnérable.

- **AES** (« Advanced Encryption Standard » 1997 en cours de développement et qui correspond à une standardisation)
- **Serpent**

Tout comme les autres candidats pour AES, Serpent a une taille de bloc de 128 bits et supporte des clés de 128, 192 ou 256 bits, mais également d'autres longueurs inférieures (multiple de 8 bits). L'algorithme comporte 32 tours d'un réseau de substitution-permutation opérant sur quatre mots de 32 bits. Chaque tour utilise 32 copies de la même S-Box de 16x16 éléments, il y a 8 S-Boxes en tout qui sont utilisées chacune tous les 8 tours. Leur contenu provient d'une opération déterministe simple sur les S-Boxes de DES (les auteurs levaient ainsi une partie des soupçons sur des faiblesses volontairement insérées). Après

avoir opéré la substitution, une transformation linéaire (voir diagramme) modifie le bloc pour le tour suivant.

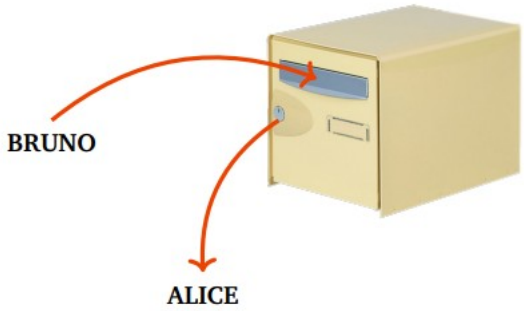
Celle-ci a fait l'objet d'une analyse poussée pour vérifier sa robustesse et améliorer l'effet avalanche.

Serpent a été conçu pour travailler en parallèle avec 32 tranches de 1 bit. Cela maximise le parallélisme, mais fait également appel à la cryptanalyse intensive dont DES a été l'objet.

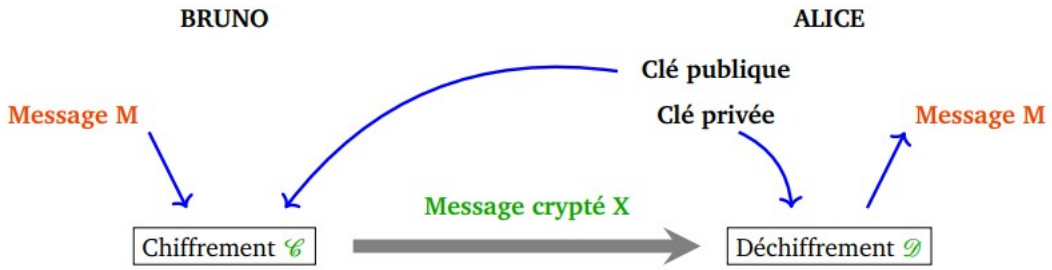
Serpent a été jugé plus prudent que Rijndael, le vainqueur d'AES, en termes de sécurité. Les concepteurs sont partis du principe que 16 tours suffisaient à repousser les attaques conventionnelles, mais pour contrer la cryptanalyse à venir, ils ont opté pour 32 tours

CHAPITRE III : LA CRYPTOGRAPHIE A CLE PUBLIQUE

Les fonctions à sens unique à trappe donnent naissance à des protocoles de chiffrement à clé publique. L'association «clé» et «publique» peut paraître incongrue, mais il signifie que le principe de chiffrement est accessible à tous mais que le déchiffrement nécessite une clé qu'il faut bien sûr garder secrète.



De façon imagée, si Bruno veut envoyer un message à Alice, il dépose son message dans la boîte aux lettres d'Alice, seule Alice pourra ouvrir sa boîte et consulter le message. Ici la clé publique est symbolisée par la boîte aux lettre, tout le monde peut y déposer un message, la clé qui ouvre la boîte aux lettres est la clé privée d'Alice, que Alice doit conserver à l'abri.



1. DIFFIE ET HELLMAN

Clé "publique" et fonctions à sens unique

C'est en 1976 que Whitfield Diffie et Martin Hellman, de l'Université Stanford, proposent un principe de chiffrement entièrement nouveau : la cryptographie à clé publique, ou *asymétrique*.

Expliquons leur procédé de façon imagée :

Alice doit recevoir un message de Bob, mais elle ne fait pas confiance au facteur qui pourrait ouvrir sa lettre. Comment peut-elle être sûre de recevoir ce message sans qu'il soit lu ?... Alice va d'abord envoyer à Bob un cadenas ouvert, dont elle seule possède la clé. Ensuite, Bob va placer son message dans une boîte, qu'il fermera à l'aide de ce cadenas, avant de l'envoyer à Alice. Le facteur ne pourra donc pas ouvrir la boîte, puisque seule Alice possède la clé !

Ainsi, un système cryptographie à clé publique est en fait basé sur **deux clés** :

- Une clé publique, pouvant être distribuée librement, *c'est le cadenas ouvert*
- Une clé secrète, connue uniquement du receveur, *c'est le cadenas fermé*

C'est la raison pour laquelle on parle de chiffrement asymétrique.

En résumé, on dispose d'une fonction P sur les entiers, qui possède un inverse S. On suppose qu'on peut fabriquer un tel couple (P,S), mais que connaissant uniquement P, il est impossible (ou au moins très difficile) de retrouver S. Autrement dit, il faut **déterminer mathématiquement des fonctions difficilement inversibles, ou "à sens unique".**

Trouver de telles fonctions semblait ardu aux mathématiciens. En effet, comment imaginer une fonction qui soit à sens unique pour tout le monde, excepté pour son créateur qui peut l'inverser grâce à la connaissance d'une information particulière (la clé) ? Ce sont Diffie et Hellman qui ont les premiers donné une réponse à cette question.

Le protocole de Diffie et Hellman

Parallèlement à leur principe de cryptographie à clé publique, Diffie et Hellman ont proposé un protocole d'échanges de clés totalement sécurisé, basé sur des fonctions difficiles à inverser.

(1) Alice et Bob se mettent d'accord publiquement sur un très grand nombre premier "p" et sur un nombre "n" inférieur à "p".

(2) Alice engendre une clé secrète "a" et Bob une clé secrète "b".

(3) Alice calcule l'élément public k_a et Bob l'élément public k_b :

$k_a = n^a \bmod p$

$k_b = n^b \bmod p$

(4) Alice transmet sa clé publique k_a à Bob, et Bob transmet sa clé publique k_b à Alice.

(5) Alice et Bob profitent ensuite de la commutativité de la fonction exponentielle pour établir leur secret commun :

$K_{Alice} = (k_b)^a = (n^b)^a \bmod p$

$K_{Bob} = (k_a)^b = (n^a)^b \bmod p$

$\Rightarrow K_{Alice} = K_{Bob} = n^{ab} \bmod p$

Sécurité du système

A priori, il n'y a pas moyen, à partir des informations transmises publiquement (p,n,n^a,n^b), de trouver n^{ab} sans caculer un logarithme modulo p, ou faire un quelconque calcul d'une complexité exagérée.
Ainsi, la sécurité du système est dite *calculatoire* et repose sur deux hypothèses :

- L'adversaire dispose d'une puissance de calcul limitée
- Avec cette contrainte de puissance et un temps limité, il n'est pas possible d'inverser la fonction exponentielle, ni de trouver n^{ab} à partir de p,n,n^a,n^b.

Remarque 1 : malgré tout cela, en 2001, des experts français ont réussi à inverser la fonction exponentielle modulaire pour un nombre p de 120 chiffres ! La sécurité d'un système dépend donc des progrès constants dans le domaine de la complexité algorithmique.

Remarque 2 : les fonctions à sens unique sont souvent issues de l'arithmétique modulaire, car elles se comportent de manière très irrégulière.

Les limites du système

Le schéma de Diffie-Hellman, bien qu'astucieux, reste un schéma de principe et souffre d'un inconvénient majeur : il n'assure pas les services de sécurité classiques que sont l'authentification mutuelle des deux intervenants, le contrôle de l'intégrité de la clé et l'anti-rejeu (vérifier qu'une information déjà transmise ne l'est pas à nouveau). L'ennemi peut donc facilement usurper l'identité d'Alice, en remplaçant son élément public par le sien.

2. Le RSA

Le principe

Le premier système à clé publique solide à avoir été inventé, et le plus utilisé actuellement, est le système RSA. Publié en 1977 par Ron **R**ivest, Adi **S**hamir et Leonard **A**dleman de l'Institut de technologie du Massachusetts (MIT), le RSA est fondé sur la difficulté de factoriser des grands nombres, et la fonction à sens unique utilisée est une fonction "puissance".

L'algorithme de chiffrement

Départ :

- Il est facile de fabriquer de grands nombres premiers p et q (+- 100 chiffres)
- Etant donné un nombre entier n = pq, il est très difficile de retrouver les facteurs p et q

(1) Création des clés

- La clé secrète : 2 grands nombres premiers p et q
- La clé publique : n = pq ; un entier e premier avec (p-1)(q-1)

(2) Chiffrement : le chiffrement d'un message M en un message codé C se fait suivant la transformation suivante :

$C = M^e \bmod n$

(3) Déchiffrement : il s'agit de calculer la fonction réciproque

$M = C^d \bmod n$

tel que $e.d = 1 \bmod [(p-1)(q-1)]$

La signature électronique

Après la confidentialité de la transmission d'un message subsiste un problème : son authenticité. Alice voudrait bien envoyer un message M à Bob de telle façon que celui-ci soit sûr qu'elle est réellement l'émettrice du message, et qu'un intrus ne tente pas de venir semer la confusion.
Le système RSA fournit une solution à ce problème :

Rappelons les données :

- Alice seule détient la clé secrète d et diffuse la clé publique (n,e)
- Alice va se servir de la clé publique pour chiffrer le message M

(1) Alice accompagne son message chiffré de sa **signature**, qui correspond à : M^d

(2) Bob va donc voir si l'égalité $(M^d)^e \bmod n = M$ est vérifiée. Si c'est le cas, Alice est bien l'émettrice du message.

Sécurité du système : primalité, factorisation

Comme pour les protocoles fondés sur le logarithme discret, la sécurité du système RSA est calculatoire : elle dépend essentiellement de la difficulté de factoriser un entier qui est le produit de deux grands nombres premiers. Si on sait factoriser n, il est facile de trouver d. Il est à noter qu'il est conseiller d'utiliser des clés de 1024 bits (+- 309 chiffres décimaux).

Le principal objet des attaques est l'*implémentation*, la *mise en œuvre* du RSA. C'est la manière de se servir du système qui peut poser problème, pas le système lui-même (par exemple, prendre une clé trop petite...).

Signalons enfin que le réel problème du RSA (et des autres systèmes à clé publique) n'est pas la sécurité, mais la lenteur. Tous les algorithmes à clé publique sont 100 à 1000 fois plus lents que les algorithmes à clé secrète, quelle que soit leur implémentation (logicielle ou matérielle) !

Exemple : chiffrer BONJOUR

1) Alice crée ses clés :

- La clé secrète : **p = 53 , q = 97** (Note : en réalité, p et q devraient comporter plus de 100 chiffres !)
- La clé publique : **e = 7 (premier avec 52*96), n = 53*97 = 5141**

2) Alice diffuse sa clé publique (par exemple, dans un annuaire).

3) Bob ayant trouvé le couple **(n,e)**, il sait qu'il doit l'utiliser pour chiffrer son message. Il va tout d'abord remplacer chaque lettre du mot BONJOUR par le nombre correspondant à sa position dans l'alphabet :
B = 2, O = 15, N = 14, J = 10, U = 21, R = 18

BONJOUR = 2 15 14 10 15 21 18

4) Ensuite, Bob découpe son message chiffré en blocs de même longueur représentant chacun un nombre plus petit que **n**. Cette opération est essentielle, car si on ne faisait pas des blocs assez longs (par exemple, si on laissait des blocs de 2 chiffres), on retomberait sur un simple chiffre de substitution que l'on pourrait attaquer par **l'analyse des fréquences** (Voir chiffre de César).

BONJOUR = 002 151 410 152 118

5) Bob chiffre chacun des blocs que l'on note **B** par la transformation $\mathbf{C = B^e \bmod n}$ (où C est le bloc chiffré) :

$C_1 = 2^7 \bmod 5141 = 128$
$C_2 = 151^7 \bmod 5141 = 800$

$C_3 = 410^7 \bmod 5141 = 3761$

$C_4 = 152^7 \bmod 5141 = 660$

$C_5 = 118^7 \bmod 5141 = 204$

On obtient donc le message chiffré C : 128 800 3761 660 204.

3. *CONCLUSION SUR LA CRYPTOGRAPHIE À CLÉ PUBLIQUE*

Après avoir vu les avantages indéniables de la cryptographie à clé publique (transmission de la clé, sécurité, authentification), on est en raison de se demander quel pourrait être l'avenir des systèmes à clé secrète.
Le RSA et ses comparses ont-ils relégué le DES et l'AES aux oubliettes ? Faut-il renier Vernam et Feistel ?

En effet, à l'heure actuelle, on utilise des **cryptosystèmes hybrides**, qui couplent les avantages des deux principes, alliant la souplesse de gestion des clés de la cryptographie asymétrique et les performances (vitesse) de la cryptographie symétrique.

Il existe deux types de cryptosystèmes hybrides :

- Soit, la cryptographie à clé publique sécurise le transprot d'une clé symétrique
- Soit, les entités émettrice et destinatrice se mettent publiquement d'accord sur un secret commun et l'utilisent ensuite pour chiffrer les données grâce à un algorithme symétrique classique.

CHAPITRE IV : FONCTIONS DE HACHAGE

1. *C'EST QUOI UN CODE DE HACHAGE ?*

Une fonction de hachage permet d'obtenir un condensé à partir d'une série de données. Ce condensé a une taille fixe qui dépend de l'algorithme utilisé.

Un code de hachage est généré par une "fonction de hachage". A partir d'un fichier on peut donc le passer à la "moulinette" d'une fonction de hachage et en sortir **une signature** : c'est le "hash-code".

L'une des utilisations des hash-codes est de pouvoir vérifier que le transfert d'un fichier s'est bien passé : avant d'envoyer le fichier, l'émetteur va calculer en local l'empreinte du fichier et va envoyer ces deux informations (le fichier et l'empreinte) vers le destinataire. Une fois les informations reçues par le destinataire, celui-ci va recalculer l'empreinte sur la base du fichier reçu et en utilisant le même algorithme de hachage et comparer le résultat de son calcul avec l'empreinte qui lui a été envoyée par l'émetteur : si les empreintes correspondent cela indique qu'il n'y a pas eu d'erreur durant le transfert.

- **Fonction de hachage** : fonction prenant en entrée un message de longueur variable et générant une empreinte de taille fixe.
- **Empreinte** : permet de s'assurer avec un certain niveau de vraisemblance qu'un message n'a pas été altéré accidentellement (bits de parité, codes détecteurs d'erreurs)
- **Fonction de hachage *à sens unique* (One-way hash function)** : permet de s'assurer qu'un message n'a pas été altéré *intentionnellement*. Donc :
- Calcul de l'empreinte : facile, algorithme connu de tous,
- Recherche d'un message d'empreinte donnée : très difficile

- Pour les fonctions *résistantes aux collisions* : recherche de deux messages de même empreinte : très difficile

- **Emetteur** :
  Calcule H(M)
  Chiffre H(M) avec sa clé privée d_E
  Envoie M et $(H(M))^{d}_{E}$

- **Destinataire** :
  Calcule H(M)
  Déchiffre avec clé publique émetteur
  Vérifie que :
  $((H(M))^{d}_{E})^{e}_{E}) = H(M)$

- **Pirate** (cherchant à substituer le message par un autre)
  Doit trouver M' tel que H(M')=H(M)

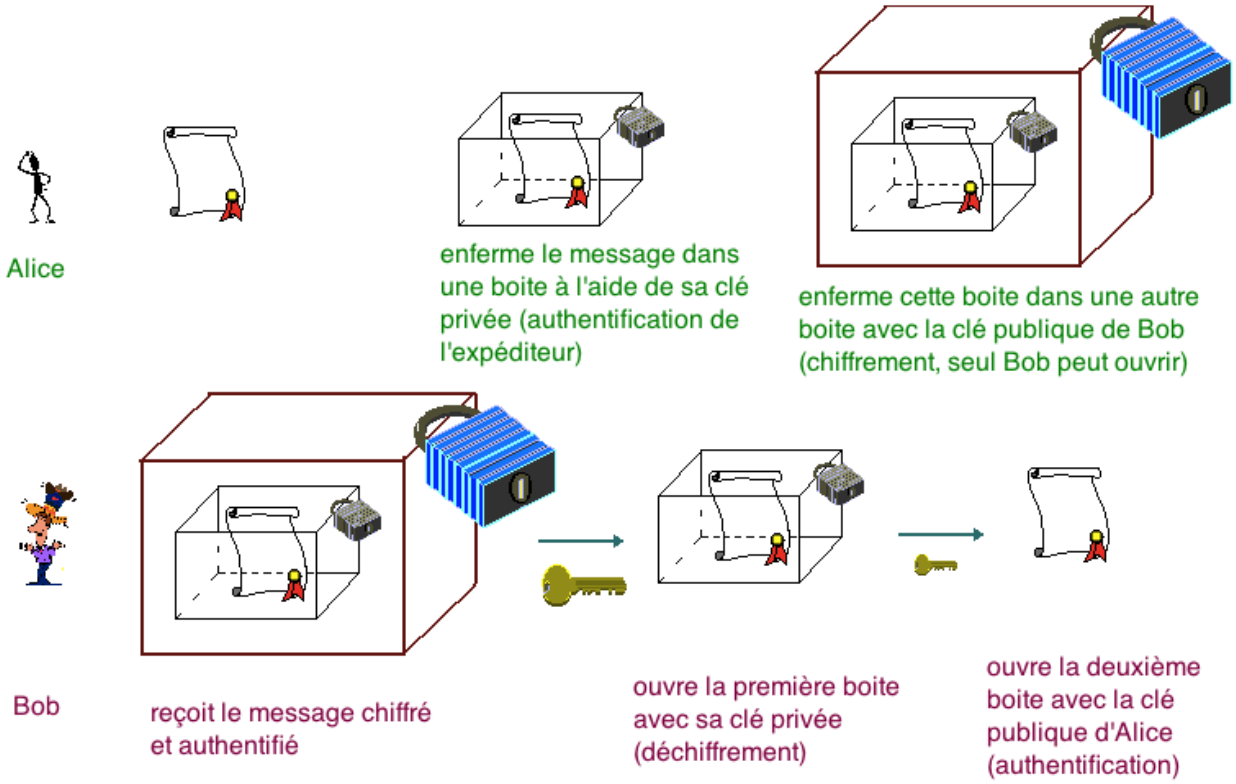


Figure (ci-dessus) : Signature électronique

2. *LES CODES DE HACHAGE SONT TRÈS UTILES*

Les codes de hachage sont utilisés dans une foule de protocoles, logiciels et systèmes. En fait, nous utilisons des codes hachage toute la journée (sans le savoir bien évidemment). Ils sont utilisés pour

- des structures de stockage de données ("*tables de hachage*")
- les signatures électroniques (ces fameux "certificats" SSL)
- dans les tunnels chiffrés SSL ou IPSEC (intégrité des communications)
- dans des protocoles d'authentification comme RADIUS (permet d'authentifier un client et un serveur RADIUS via un « secret partagé » mais aussi de « chiffrer » le mot de passe de connexion)
- pour la génération de clefs de chiffrement (le hashcode d'un mot de passe saisi devient la clef de chiffrement)
- dans la création de monnaie électronique comme les BitCoins (cf le BitCoin Mining)

  (Bitcoin est une technologie paire à paire fonctionnant sans autorité centrale. La gestion des transactions et la création de bitcoins est prise en charge collectivement par le réseau. **Bitcoin est *libre et ouvert*. Sa conception est publique, personne ne possède ni ne contrôle Bitcoin et tous peuvent s'y joindre**. Grâce à plusieurs de ses propriétés uniques, Bitcoin rend possible des usages prometteurs qui ne pourraient pas être couverts par les systèmes de paiement précédents.)

- le stockage des mots de passe (on ajoute un "sel" au mot de passe et on génère un hash-code qui est mis dans un fichier)
- pour accélérer et optimiser le transfert d'informations (synchronisation ou déduplication de données, etc.)
- dans les systèmes de génération de preuves (pour s'assurer que des données n'ont pas été modifiées depuis leur collecte initiale)
- les bases de réputation d'URL (pour savoir si une URL est "dangereuse" ou non, et ce sans que la liste des URLs soit accessible)

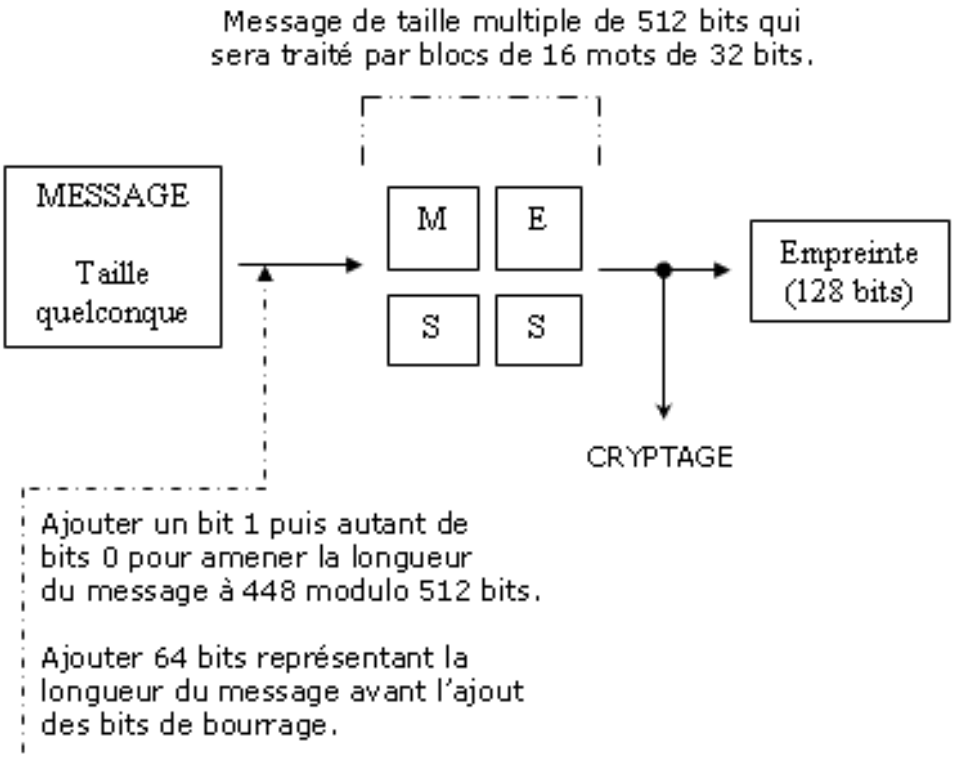
- les antivirus (pour optimiser les traitements, savoir qu'un fichier est sain car scanné très récemment)

3. *PROPRIÉTÉS D'UNE FONCTION DE HACHAGE*

- Réductrice : le résultat d'une fonction de hachage a **toujours le même format et la même taille**, et ce quelle que soit la taille ou la nature des données en entrée. Dans le cas de l'algorithme MD5 le résultat sera toujours une chaine de 128bits (ex: BF34 65CD E275 CE93).
- Injective : à partir d'une signature ou d'un hash-code, il n'est **pas possible de "revenir"** ou de "retrouver" les informations qui ont servi à générer le hash-code.
- **Absence de collision** : pour deux informations différentes en entrée de l'algorithme les hash-codes en sortie seront différent.
- **Cryptographique** : une fonction de hachage est dite "cryptographique" s'il n'est pas possible de prédire à l'avance la signature.

4. *TYPES D'ALGORITHMES DE HACHAGE*

a) **MD5** (Message Digest 5) est une fonction de hachage utilisée en standard avec RSA pour les signatures numériques. Elle génère une empreinte de 128 bits de la manière suivante :



b) **SHA** (*Secure Hash Algorithm*)

Le préfixe **SHA** (acronyme de *Secure Hash Algorithm*) est associé à plusieurs fonctions de hachage cryptographiques publiées par le NIST en tant que *Federal Information Processing Standard* (FIPS).

Les fonctions SHA-0, SHA-1 et SHA-2 ont été conçues par la NSA ; leurs spécifications sont décrites par les publications FIPS-180, dont la dernière version (fin 2012) est le FIPS-180-4[1].

SHA-0

SHA-0 s'appelait originellement SHA, et ses spécifications ont été publiées en 1993. Elle est inspirée des fonctions MD4 et MD5 de Ron Rivest. Le NIST recommande formellement de ne pas l'utiliser depuis 1996, pour des questions de sécurité. Elle est cependant restée un objet d'étude pour la communauté académique, en tant que prototype de SHA-1.

SHA-1

SHA-1 est une version légèrement modifiée de SHA-0 publiée en 1995, qui produit comme celle**i**-ci un haché de 160 bits. Il existe des attaques théoriques pour la recherche de collisions, de complexité nettement moindre que l'attaque générique des anniversaires. Elle

a été très utilisée dans les protocoles et applications de sécurité, mais, à cause de l'existence de ces attaques, tend à être remplacée par SHA-256.

SHA-2

SHA-2 est une famille de fonctions de hachage cryptographiques publiée en 2001 qui regroupe à l'origine SHA-224, SHA-256, SHA-384 et SHA-512. Ces fonctions produisent des hachés de taille différente (désignée par le suffixe, en bits). Le standard FIPS-180-4 (mars 2012) est augmenté de deux versions tronquées de SHA-512, SHA-512/256 (haché de 256 bits) et SHA-512/224 (haché de 224 bits). Elles utilisent des algorithmes très similaires, eux-mêmes largement inspirés de celui de SHA-1. L'un est à base de mots de 32 bits (et d'un découpage en blocs de 512 bits) pour SHA-256 et sa version tronquée SHA-224. L'autre est à base de mots de 64 bits (et d'un découpage en blocs de 1024 bits) pour SHA-512 et ses versions tronquées SHA-384, SHA-512/256 et SHA-512/224. Les attaques connues sur SHA-1 n'ont pu être transposées à SHA-2, même si la construction est proche.

SHA-3

SHA-3, originellement Keccak, est une nouvelle fonction de hachage cryptographique décrite en août 2015 par la publication FIPS-202[2]. Elle a été sélectionnée en octobre 2012 par le NIST à la suite d'un concours public lancé en 2007, ceci car les faiblesses découvertes sur MD-5 et SHA-1 laissent craindre une fragilité de SHA-2 qui est construite sur le même schéma. Elle possède des variantes qui peuvent produire des hachés de 224, 256, 384 et 512 bits. Elle est construite sur un principe tout à fait différent de celui des fonctions MD5, SHA-1 et SHA-2.

c) **RIPEMD-160** (*RACE Integrity Primitives Evaluation Message Digest)*

L'algorithme de hachage **RIPEMD-160**, pour *RACE Integrity Primitives Evaluation Message Digest*, est une fonction de hachage qui produit une signature de 160 bits. Elle a été développée en Europe par Hans Dobbertin, Antoon Bosselaers et Bart Preneel. Publiée en 1996, il s'agit d'une version améliorée du RIPEMD qui reprend certaines idées du MD4. D'autres versions de l'algorithme existent (128, 256 et 320 bits).

d) Whirlpool

Whirlpool est une fonction de hachage cryptographique conçue par Vincent Rijmen et Paulo Barreto pour le projet NESSIE. Elle a été nommée d'après la galaxie M51. La fonction utilise une architecture de type Miyaguchi-Preneel connue pour sa résistance à la cryptanalyse, cette structure produit des empreintes de 512 bits qui à l'heure actuelle sont assez rares (citons toutefois SHA-512).

En interne, l'algorithme travaille sur 512 bits grâce à une fonction similaire à celle de l'algorithme de chiffrement symétrique AES (auquel Vincent Rijmen a également participé et qui à l'origine s'appelle Rijndael). L'utilisation d'une version modifiée du bloc de chiffrage de AES (appelée W) garantit un système robuste et fiable.

e) Tiger

Tiger est une fonction de hachage cryptographique conçue par Ross Anderson et Eli Biham en 1996. Tiger fournit une empreinte sur 192 bits mais des versions sur 128 et 160 bits existent aussi. Ces versions raccourcies prennent simplement les premiers bits de la signature de 192 bits.

5. *QUEL ALGORITHME DE HACHAGE UTILISER ?*

Si dans le cadre d'un projet de développement logiciel ou de la configuration d'un système, vous devez choisir quel algorithme de hachage utilisé, voici quelques conseils :

- Eviter MD5 (Message Digest 5). Oui, MD5 est un "grand classique" mais il est à proscrire car cette fonction de hachage n'est plus considérée comme sûre. Il est donc préférable d'utiliser SHA1 (Secure Hash Algorithm 1) qui est très largement supporté.
- Pour ceux souhaitant un algorithme encore plus sécurisé ils pourront utiliser SHA-256 (qui est en fait l'un des algorithmes de la suite "SHA2"... donc le successeur de SHA1).
- D'autres algorithmes de hachage comme Tiger ou Whirlpool : leur utilisation reste rare et, sauf besoin spécifique, il est généralement préférable d'éviter de les utiliser.

Exemple d'application des fonctions d'hachage sur un texte : « **Merci pour l'essai »**

```
File: GBB.txt  (Merci pour l'essai)
File size: 18 bytes (18 bytes)
MD5 checksum: E29B1DFB0E3961345FDB138AB43B1D90
SHA1 checksum: B2625B25FE99A4D3C5E304F3A7F8F1CF6032FD53
SHA256 checksum: E5496999059BC02906904C14B3BEE335373E0CA2721F608896E7E9224C2237BF
SHA384 checksum: 5573F7E05F48EB33A685C2E64CB0C09808F79C57CE809E8DFAD34A5A612AD14B32B0FE7699A4E179614AF48B67E209A0
SHA512 checksum: 3DD4190B584CE587E91114485697F57CE2DB26E8D77C3BFA64469494C3B2090B2F566F407CD30DB8945AF6AB40C65AD7B05D96C5A7DC27801E6D0F9322C11F2F
```

HMAC (Hashed-Based Message Authentication Code)

L'algorithme HMAC est une alternative de l'algorithme MAC, consistant à protéger l'intégrité d'un message ainsi que l'authenticité de l'expéditeur, par l'utilisation d'un mécanisme de protection de l'intégrité qu'est la fonction de Hash. Le processus d'authentification de l'expéditeur suppose le partage d'une clé secrète partagée.

Le but de MAC est d'authentifier à la fois la source d'un message et son intégrité sans l'utilisation de mécanismes additionnels. HMAC possède deux paramètres fonctionnellement différents, un message en entrée et une clé secrète connue uniquement par l'expéditeur du message et son récipiendaire.

Des applications additionnelles des fonctions de hachage par clé (keyed-hash functions) comprennent leur utilisation dans des protocoles de challenge-response pour calculer les réponses.

Une fonction HMAC est utilisée par l'expéditeur d'un message pour produire une valeur, le Code d'Authentification de Message (MAC, Message Authentication Code) formé en condensant la clé secrète et le message en entrée.

Le code est typiquement envoyé au destinataire du message en même temps que le message. Le destinataire traite le code MAC du message reçu en utilisant la même clé et la fonction HMAC de la même façon que l'expéditeur. Il va comparer les résultats obtenus. Si les deux valeurs de hash sont identiques, le message a été correctement reçu (sans altération) et le destinataire est assure que l'expéditeur est un membre de la communauté d'utilisateurs partageant la même clé.

Il faut noter que HMAC traite la fonction de hash comme une boîte noire dont il ignore tout. La raison est de permettre de remplacer le module contenant la fonction de hash par n'importe quel autre algorithme qui serait plus sûr ou plus pertinent. Rappelons que la robustesse d'un algorithme MAC dépend de la « force cryptographique » de l'algorithme de hash embarqué.

La taille des blocs dépend de la fonction de hash implémentée. Il existe ainsi les implémentations courantes suivantes : HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, mentionnant la taille des blocs. Chaque implémentation utilise respectivement SHA-256, SHA-384 et SHA-512.

Notes rassemblées par Gabriel BOMBAMBO Boseko
Courriel : ***gbombambo@gmail.com*** / Twitter : ***@gbboseko*** / GPG-Key : ***7C4AB1AD*** /
Tél : ***+243 81 567 99 85 / +243 84 077 91 60 / +243 997 555 918***

CHAPITRE V : UTILISATIONS DE LA CRYPTOGRAPHIE

1. *LES CARTES BANCAIRES*

Les banques font partie des premiers utilisateurs de systèmes cryptographies. Les cartes bancaires possèdent trois niveaux de sécurité :

- Le code confidentiel : c'est la suite de chiffres à mémoriser et à saisir à l'abri des regards indiscrets.
- La signature RSA : permet de vérifier l'identité de la carte sans avoir besoin de secrets; en d'autres termes, cette vérification peut se faire sans connexion à un centre distant.
- L'authentification DES : apporte une preuve de légitimité de la carte, et se fait par connexion à un centre distant.

Paiement par carte bancaire (Est-ce qu'il s'agit d'une vraie carte ? Est-ce que le montant débité sera égal au montant crédité ? Est-ce que le code secret est bien protégé ?)

2. LES NAVIGATEURS WEB

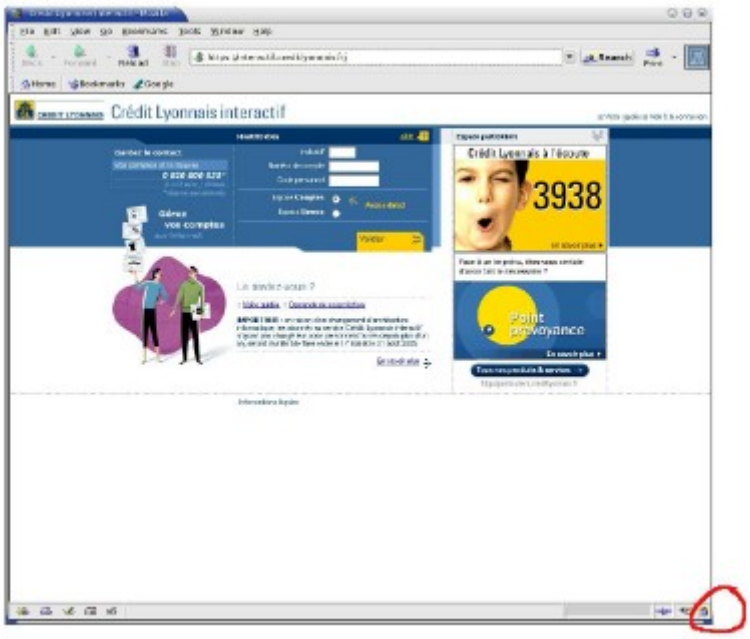
Les navigateurs, ou *broswers*, tels que Mozilla Firefox ou Internet Explorer, utilisent le protocole de sécurité SSL (*Secure Sockets Layers*), qui repose sur un procédé de cryptographie par clé publique : le RSA.

3. LE VOTE ELECTRONIQUE

Le résultat reflète le vote, chaque vote est confidentiel, on ne peut pas connaître des résultats partiels, seuls les électeurs peuvent voter et une seule foi

4. *INTERNET*

Confidentialité, anonymat, authentification (s'agit-il bien de ma banque ?)



5. *SIGNATURE ÉLECTRONIQUE*

Signature électronique (véritable, authentique, non-répudiation (je n'ai jamais signé ce texte ...)

6. *DÉCODEURS*

Décodeurs (Vérification de l'abonné, impossibilité de retransmettre les données décodées à une tierce personne, mise à jour de l'abonnement)

7. *PORTE MONNAIE ÉLECTRONIQUE*

Porte monnaie électronique (pas de création de fausse monnaie, pas de création de faux porte-monnaie)

Notes rassemblées par Gabriel BOMBAMBO Boseko
Courriel : ***gbombambo@gmail.com*** / Twitter : ***@gbboseko*** / GPG-Key : ***7C4AB1AD*** /
Tél : ***+243 81 567 99 85 / +243 84 077 91 60 / +243 997 555 918***

CHAPITRE VI : CRYPTOGRAPHIQUE QUANTIQUE

0. INTRODUCTION

La communication de données confidentielles par un canal de transmission classique (par exemple Internet) nécessite l'utilisation d'algorithmes de cryptographie classiques : algorithmes de chiffrement asymétrique tels que RSA, ou de chiffrement symétrique (Triple DES, AES). Dans le cas du chiffrement symétrique, les deux interlocuteurs doivent posséder *a priori* une clé secrète, c'est-à-dire qui ne soit connue que d'eux.

Se pose alors la question suivante : comment transmettre une clé de chiffrement entre deux interlocuteurs (1) *à distance*, (2) *à la demande*, et (3) *avec une sécurité démontrable* ? Actuellement, la technique se rapprochant au mieux de ces trois critères est une transmission physiquement sécurisée, de type valise diplomatique.

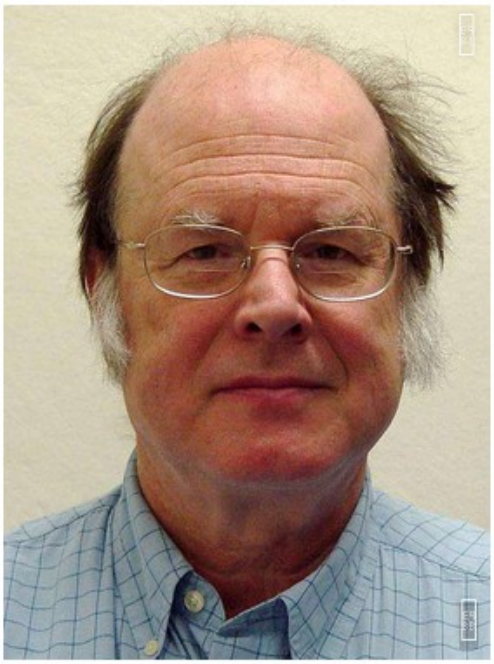
La cryptographie quantique cherche à répondre à ces trois critères en transmettant de l'information entre les deux interlocuteurs en utilisant des objets quantiques, et en utilisant les lois de la physique quantique et de la théorie de l'information pour détecter tout espionnage de cette information. S'il n'y a pas eu espionnage, une clé parfaitement secrète peut être extraite de la transmission, et celle-ci peut être utilisée dans tout algorithme de chiffrement symétrique afin de transmettre un message.

Paradoxalement, la cryptographie quantique n'est pas de la cryptographie, car elle n'est pas une méthode de chiffrement d'un message utilisant la mécanique quantique. On devrait plus correctement la nommer « distribution quantique de clés », comme c'est bien le cas en anglais. Il s'agit en effet d'un ensemble de protocoles permettant de distribuer une clé de chiffrement entre deux interlocuteurs distants, tout en assurant la sécurité de la transmission grâce aux lois de la physique quantique et de la théorie de l'information.

1. LES DIFFÉRENTS PROTOCOLES DE CRYPTOGRAPHIE QUANTIQUE

Il existe plusieurs protocoles de cryptographie quantique. On présente souvent celui développé par Bennet et Brassard en 1984, qui utilise la polarisation des photons. On s'y réfère comme le protocole BB84. Le protocole E91 a lui été imaginé par Artur Ekert en 1991.

Il utilise une paire de photons intriqués et donc repose sur l'effet EPR bien mis en évidence par les expériences d'Alain Aspect et ses collègues.



Charles H. Bennett (né en 1943) a travaillé sur les bases physiques de la théorie de l'information, notamment en rapport avec la physique quantique. Il a joué un rôle majeur dans l'élucidation de ces interconnexions, en particulier dans le domaine de l'informatique quantique, mais aussi dans celui des automates cellulaires. Il a découvert, avec Gilles Brassard, le concept de la cryptographie quantique et est l'un des pères fondateurs de la théorie moderne de l'information quantique. © 2011 ETH Zurich

2. APPLICATIONS DE LA CRYPTOGRAPHIE QUANTIQUE

La cryptographie quantique est sortie du domaine de la théorie depuis des années, ce n'est pas une curiosité de laboratoire car elle a déjà été mise en pratique, par exemple et pour la première fois en 2004 pour une importante transaction financière requérant une sécurité absolue et en 2007 lorsque l'entreprise suisse id Quantique a transmis les résultats des élections nationales à Genève.

Bien évidemment, la cryptographie quantique intéresse beaucoup les militaires. La Darpa (agence américaine sur la recherche militaire avancée) utilise ainsi depuis 2004 un réseau de distribution quantique des clefs. L'Union européenne n'est pas en reste car en réponse au programme d'espionnage Echelon, elle a été à l'origine du réseau Secoqc.

3. À QUOI VA SERVIR LE SATELLITE QUANTIQUE LANCÉ PAR LA CHINE

La Chine a effectué lundi 15 août le premier lancement mondial d'un satellite à communication quantique, a annoncé un média d'Etat, une percée technologique pour Pékin, qui ambitionne par ce biais d'édifier un système inviolable de communications cryptées.

Le lancement a été effectué à 19h30 dans le désert de Gobi (nord), a annoncé l'agence officielle Chine nouvelle, et intervient à l'heure où les Etats-Unis, le Japon et d'autres nations souhaitent elles aussi s'imposer dans cette technologie en plein essor.

La Chine a investi d'énormes ressources financières dans ce marathon technologique, l'un des nombreux investissements de Pékin dans la recherche scientifique de pointe, de l'exploitation minière des astéroïdes aux manipulations génétiques en passant par la conquête de Mars. Le programme d'Etat dédié à la recherche fondamentale (dont la physique quantique) est passé de 1,9 milliards à 101 milliards de dollars de 2005 à 2015.

Des communications indéchiffrables

Le satellite --nommé Mozi en l'honneur d'un philosophe chinois du 5e siècle avant J.-C.-- sera utilisé pour démontrer l'intérêt de la technologie quantique dans les communications longue distance.

A la différence des méthodes classiques de transmission sécurisée, le système utilise des photons (la particule élémentaire qui compose la lumière) pour envoyer les clés de chiffrement nécessaires au décodage de l'information. Les données contenues dans ces photons sont impossibles à intercepter: toute tentative d'espionnage provoquerait leur autodestruction, affirme Chine nouvelle.

Gregoir Robordy, co-fondateur d'une société spécialisée sur le chiffrement quantique, explique au *Wall Street Journal* que le chiffrement quantique rend l'information semblable à

une bulle de savon.: "Si quelqu'un essaye de l'intercepter quand elle est envoyée, en la touchant, il la fait exploser".

Comment est-ce possible? Grâce à "l'intrication quantique", un phénomène physique qui "lie" deux photons. Même si ceux-ci sont séparés par des kilomètres, si l'un change, l'autre changera aussi. L'idée consiste donc à envoyer un photon tout en gardant l'autre à proximité. Dans ces deux particules, le même mot de passe, appelé clé de chiffrement, permettra aux deux interlocuteurs de rendre lisibles les informations chiffrées qu'ils s'envoient.

Si le photon envoyé est intercepté, celui que vous avez gardé sera modifié et vous serez donc averti du piratage. Logiquement, vous n'utiliserez donc pas la clé de chiffrement stocké dans le photon.

Savoir viser juste

Si des scientifiques ont démontré l'efficacité de la technique pour transmettre des messages sur des distances relativement courtes, des obstacles techniques rendent jusqu'ici les communications longues distance hors d'atteinte.

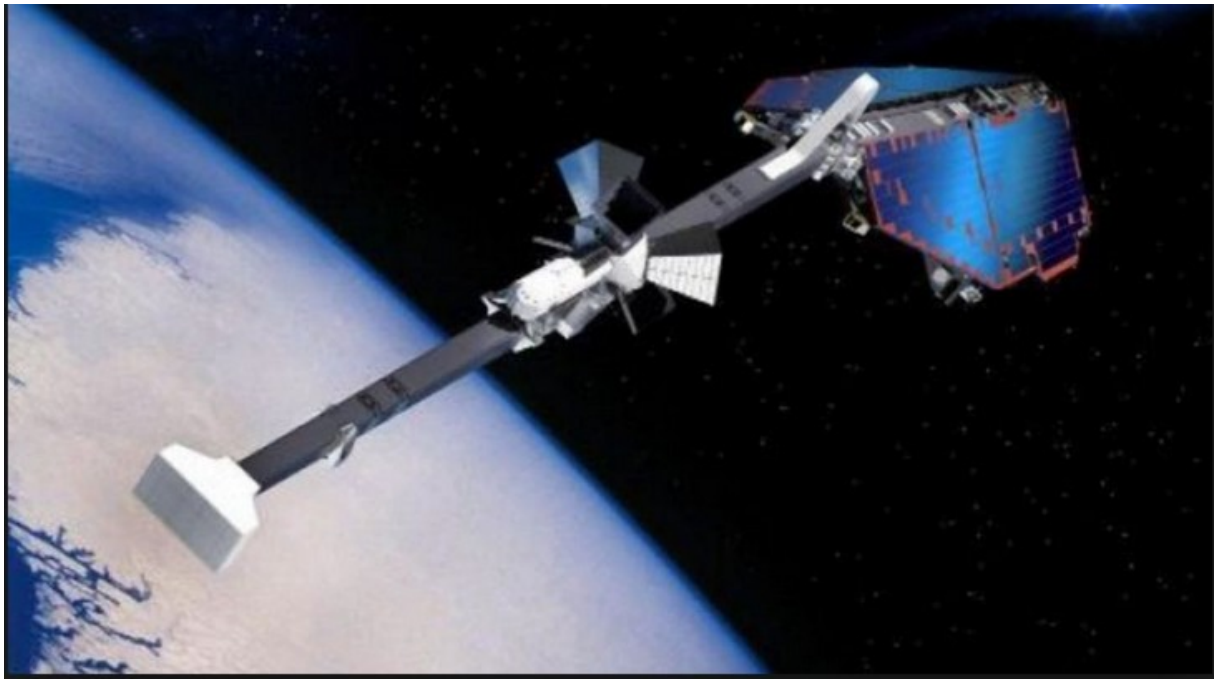
Le satellite tentera d'envoyer des données chiffrées entre Pékin et Urumqi, la capitale de la région du Xinjiang (nord-ouest), distantes de près de 2.500 km. L'opération nécessitera au satellite d'être orienté de façon extrêmement précise vers des stations réceptrices situées sur Terre, selon Chine nouvelle.

"Ce sera comme lancer une pièce de monnaie d'un avion volant à 100km d'altitude et espérer qu'elle vienne se ficher exactement dans la fente d'une tirelire-cochon en rotation", a expliqué Wang Jianyu, le responsable en chef du projet, cité par l'agence.

Développer cette nouvelle technologie est un objectif crucial pour Pékin, qui l'a incluse dans son nouveau plan quinquennal publié en mars. "Ce satellite (...) marque un tournant dans le rôle de la Chine. De celui de suiveuse en matière de développement de technologies de l'information classiques, à celui de leader menant les futurs accomplissements du secteur", s'est réjouit Pan Jianwei, le responsable en chef du satellite, cité par Chine nouvelle.

La Chine "peut espérer la création d'un réseau mondial de communications quantiques vers 2030", a-t-il déclaré. Les révélations de l'ex-consultant Edward Snowden sur les opérations d'espionnage de l'Agence de sécurité américaine (NSA) ont renforcé la quête chinoise de technologies hermétiques au piratage.

La Chine fait également partie du groupe de quelques pays travaillant à la création du premier "ordinateur quantique" au monde. Une telle machine, à l'aide des propriétés des particules subatomiques, pourrait effectuer des calculs à des vitesses fulgurantes, bien plus élevées qu'avec les technologies actuelles et pourrait notamment permettre de casser la plupart des communications chiffrées... à l'exception du chiffrement quantique



Premier *satellite* de communication *quantique* de l'histoire lancé par la Chine le 16 août 2016.

CHAPITRE VII : CERTIFICATS ET AUTHENTIFICATION

1. *CERTIFICAT IDENTIFIANT UNE PERSONNE OU UNE ENTITÉ*

Un *certificat* est un document électronique utilisé pour identifier un individu, un serveur, une entreprise ou toute autre entité et pour associer une clef publique à cette identité. Tout comme un permis de conduire, un passeport, ou tout autre moyen d'identification personnelle couramment utilisé, un certificat fournit généralement une preuve reconnue de l'identité de la personne. La cryptographie à clef publique utilise les certificats pour éviter les problèmes d'usurpation d'identité.

Pour obtenir un permis de conduire, il faut s'inscrire dans une agence gouvernementale, telle que le *Department of Motor Vehicles*, qui vérifie votre identité, votre capacité à conduire, votre adresse, et d'autres information avant de délivrer le permis. Pour obtenir une carte d'étudiant, il faut s'adresser à une université ou une école, qui contrôle certaines informations (comme le paiement des frais d'inscription) avant de délivrer la carte d'étudiant. Pour obtenir une carte de bibliothèque, il faut uniquement fournir votre nom et une attestation de logement à vos nom et adresse.

Les certificats fonctionnent sur les mêmes principes que ces différents documents d'identité. Les autorités de certification (AC ou CA) sont des entités qui valident les identités et émettent les certificats. Elles peuvent être des tierces-parties indépendantes ou des organisations possédant leur propre serveur d'émission de certificats (tel que *Red Hat Certificate System*). Les méthodes utilisées pour valider une identité varient selon les politiques d'émission d'une AC donnée — tout comme les méthodes de validation des autres formulaires d'identification varient selon les organismes d'émission et leurs domaines d'application. En général, avant d'émettre un certificat, une AC doit utiliser ses procédures de vérification publiées pour un type de certificat afin de s'assurer que l'entité demandant le certificat est bien celle qu'elle prétend être.

Le certificat émis par l'AC lie une clef publique particulière au nom de l'entité qu'il identifie (tel qu'un nom d'employé ou de serveur). Les certificats aident à prévenir l'utilisation de fausses clefs publiques pour l'usurpation d'identité. Seule la clef publique certifiée dans le

certificat fonctionnera avec la clef privée correspondante possédée par l'entité identifiée par le certificat.

En plus de la clef publique, un certificat contient toujours le nom de l'entité qu'il identifie, une date d'expiration, le nom de son AC émettrice, un numéro de série et d'éventuelles autres informations utiles. Plus important, un certificat contient toujours la signature numérique de l'AC émettrice. La signature numérique de l'AC émettrice permet au certificat de fonctionner comme une *lettre d'introduction* pour les utilisateurs qui connaissent l'AC et lui font confiance mais qui ne connaissent pas l'entité identifiée par le certificat.

CERTIFICATS

Le problème des certificats numériques est à l'opposé de celui de la signature électronique : 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. Cette fois, c'est donc du Destinataire que l'on veut être sûr, et non de l'Expéditeur.

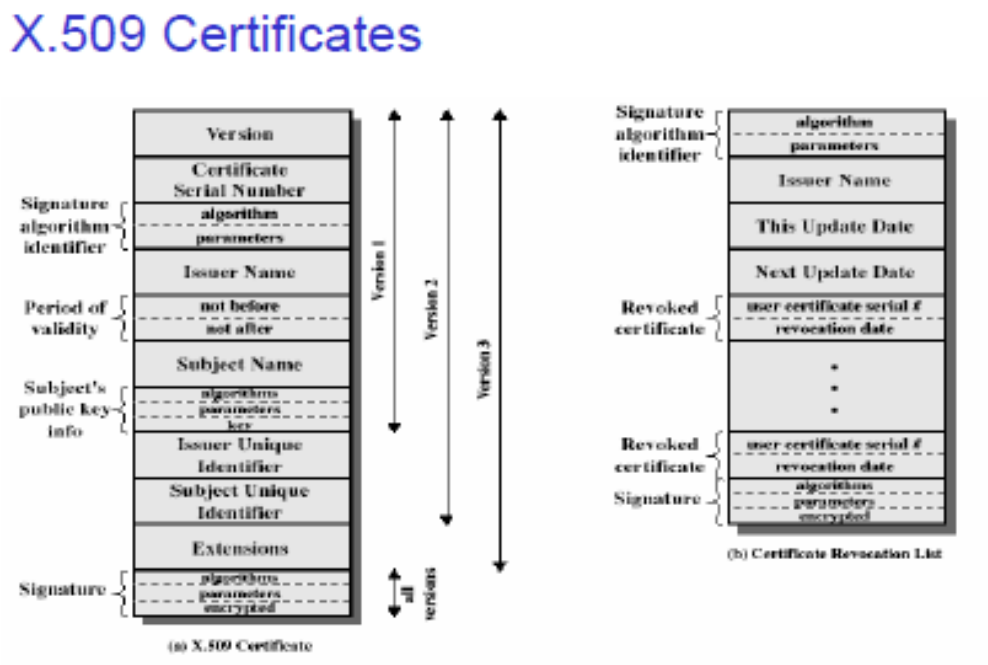
Comme dans la vie courante, on a recours à des certificats. Pour passer un examen, il vous faut prouver votre identité, ie fournir une carte d'identité, passeport ou permis de conduire. Un organisme supérieur (l'Etat) a signé ces certificats, s'assurant auparavant (par un acte de naissance,...) qu'il 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 calculée de la façon suivante : à partir des informations du certificat, l'organisme calcule un résumé en appliquant une fonction de hachage connue, comme MD5. Puis il signe ce résumé en lui appliquant sa clé secrète.

Lorsque Bob veut envoyer son message à Alice, il télécharge le certificat de celle-ci sur un serveur de certificat (on parle de PKI, Public Key Infrastructure). Il calcule le résumé du

certificat, puis applique la clé publique de l'organisme auteur du certificat à la signature électronique. Si cette quantité est égale au résumé, il est sûr qu'il a bien affaire à Alice.

- Une partie de la norme de service d'annuaire X.500
  - ✓ serveurs distribués maintenant une base de données d'information
- Définit le cadre pour des services d'authentification
- L'annuaire peut stocker des certificats de clés publiques et les clés publiques des utilisateurs correspondant
- Signé par une autorité de certification
- Utilisé fréquemment dans les protocoles d'authentification
- Utilise la crypto à clé publique et les signatures digitales
  - ✓ algorithmes non imposés, mais RSA recommandé



Obtention d'un certificat

- N'importe quel utilisateur ayant accès au CA peut obtenir un certificat de celui-ci
- 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

Hiérarchie de CA

Hypothèse :

- ✓ si les deux utilisateurs partagent un CA commun alors on suppose qu'ils connaissent sa clef publique
- ✓ chaque CA a des certificats pour les clients (vers l'avant) et le parent (vers l'arrière)
- ✓ chaque client fait confiance aux certificats parents

Solution : hiérarchie de CA

- ✓ employer les certificats liant les membres de la hiérarchie pour valider d'autre CA

Objectif : permettre la vérification de n'importe quel certificat d'un CA par des utilisateurs de tout autre CA dans la hiérarchie

Révocation de certificat

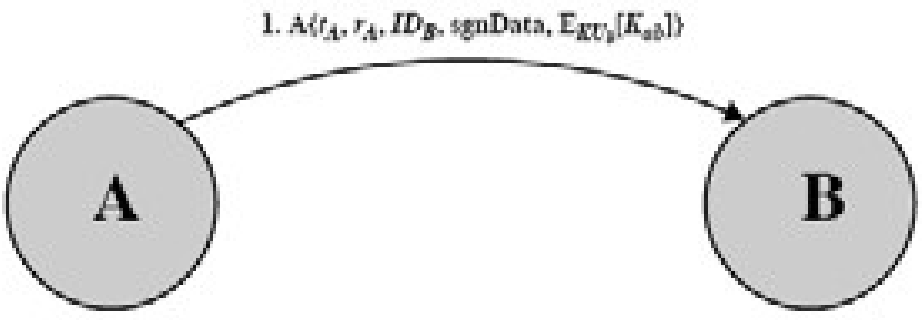
- Les certificats ont une période de validité
- On doit pouvoir le retirer avant l'échéance car, par exemple :
  - ✓ La clef privée de l'utilisateur est compromise
  - ✓ l'utilisateur n'est plus certifié par ce CA
  - ✓ le certificat du CA est compromis
- Les CA maintiennent la liste de certificats retirés
  - ✓ la liste de révocation de certificat (CRL)
- Les utilisateurs devraient pouvoir vérifier les certificats avec les CRLs

Authentication Procedures

- X.509 inclut trois procédures alternatives d'authentification :
  - ✓ Authentification à sens unique (one- way)
  - ✓ Authentification à 2 passages (two- way)
  - ✓ Authentification à 3 passages (three- way)
- Tous emploient des signatures à clés publiques

1. *One-Way Authentication*

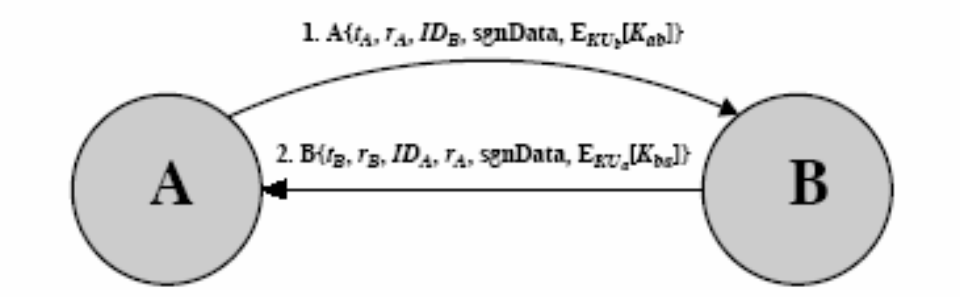
- 1 message (A->B) est utilisé pour établir
  - ✓ L'identité de A et l'origine du message
  - ✓ Le destinataire du message
  - ✓ L'intégrité du message
- Le message doit inclure l'horodateur (t_A), le nonce (r_A), l'identité de B (ID_B) et est signé par A (sgnData)



(a) One-way authentication

2. *Two-Way Authentication*

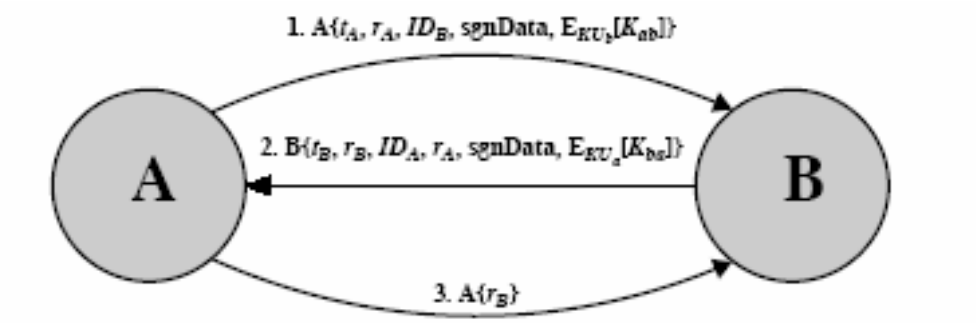
- 2 messages (A->b, B->a) – même principe +
  - ✓ l'identité de B et que cette réponse est de B
  - ✓ Le destinataire
  - ✓ intégrité et l'origine de réponse
- la réponse inclut le nonce de A, l'horodateur et le nonce de B



Permet donc aux deux parties de vérifier l'identité de l'autre

3. Three-Way Authentication

- 3 messages (A->b, B->a, A->b) qui permettent l'authentification sans horloges synchronisées
- la réponse de A à B contient la copie signée du nonce de B
- signifie que des horodateurs n'ont pas besoin d'être vérifiés ou comptés



1. L'AUTHENTIFICATION CONFIRME UNE IDENTITÉ

L'*authentification* est le processus de confirmation d'une identité. Dans le contexte d'interactions entre les réseaux, l'authentification comporte l'identification confiante d'une partie par une autre. L'authentification sur les réseaux pour avoir plusieurs formes. Les certificats sont un moyen d'authentification.

Les interactions entre les réseaux se font généralement entre un client, tel que le navigateur d'un ordinateur personnel, et un serveur, tel que le matériel et le logiciel hébergeant un site Web. L'*authentification cliente* se réfère à l'identification confiante d'un client par un serveur (c'est-à-dire, l'identification de la personne supposée utiliser le logiciel client). *L'authentification serveur* se réfère à l'identification confiante d'un serveur par le client (c'est-à-dire, l'identification de l'organisation supposée être responsable du serveur à une adresse réseau particulière).

Les authentifications cliente et serveur ne sont pas les seules formes d'authentification permises par les certificats. Par exemple, la signature numérique d'un courriel combinée au

certificat identifiant l'expéditeur fournissent une forte preuve que la personne identifiée par le certificat a bien envoyé le message. De même, une signature numérique dans un formulaire HTML combinée à un certificat identifiant le signataire peut fournir une preuve, après coup, que la personne identifiée par le certificat est d'accord avec le contenu du formulaire. En plus de l'authentification, la signature numérique assure, dans les deux cas, un degré de non répudiation (c'est-à-dire que la signature numérique rend difficile la négation ultérieure par le signataire des informations présentes dans le message électronique ou le formulaire).

L'authentification cliente est un élément essentiel de la sécurité réseau, sur les intranets ou les extranets. Les sections qui suivent présentes deux formes d'authentification cliente :

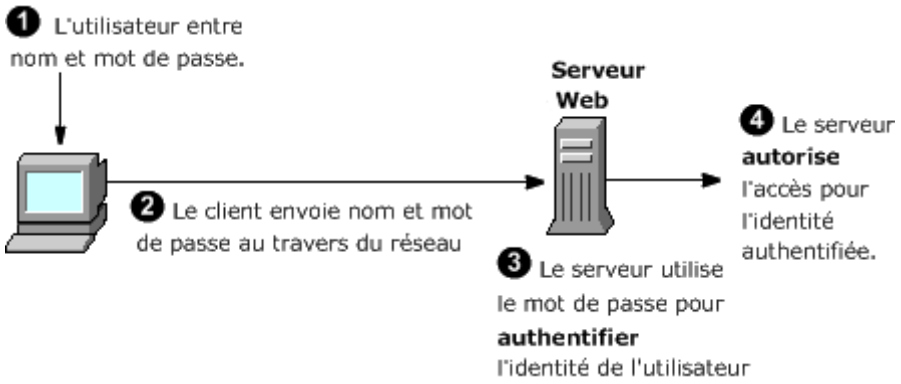
- **Authentification basée sur un mot de passe** : Presque tous les serveurs permettent l'authentification cliente à l'aide d'un nom, ou pseudonyme, et d'un mot de passe. Par exemple, un serveur pour demander à un utilisateur de fournir un nom et un mot de passe avant de donner des droits d'accès à certaines parties du serveur. Le serveur maintient une liste des noms et des mots de passe ; si un nom particulier est dans cette liste, et que l'utilisateur fournit le bon mot de passe, le serveur donne des droits d'accès.
- **Authentification basée sur un certificat** : L'authentification basée sur les certificats est une étape du protocole SSL. Le client signe numériquement des données générées aléatoirement et envoie à la fois ces données signées et le certificat sur le réseau. Le serveur utilise les techniques de cryptographie à clef publique pour valider la signature et confirmer la validité du certificat.

4. L'AUTHENTIFICATION PAR MOT DE PASSE

La figure ci-dessous montre les étapes basiques mise en œuvre dans l'authentification d'un client à l'aide d'un nom et d'un mot de passe. Cette figure suppose que :

- L'utilisateur a déjà décidé de faire confiance au serveur, sans authentification ou sur la base d'une authentification de serveur via SSL.
- L'utilisateur a demandé une ressource contrôlée par le serveur.

- Le serveur demande une authentification client avant de donner les droits d'accès aux ressources demandées.



Voici les étapes décrites dans la figure ci-dessus :

1. En réponse à une demande d'authentification de la part du serveur, le client affiche une boîte de dialogue demandant le nom de l'utilisateur et son mot de passe pour accéder à ce serveur. L'utilisateur doit fournir séparément un nom et un mot de passe pour chaque nouveau serveur qu'il désire utiliser pendant sa session de travail.
2. Le client envoie le nom et le mot de passe par le réseau, en clair ou par une connexion SSL chiffrée.
3. Le serveur vérifie le nom et le mot de passe dans sa base de données locale et, s'ils correspondent, il les accepte comme preuves authentifiant l'identité de l'utilisateur.
4. Le serveur détermine si l'utilisateur est autorisé à accéder aux ressources demandées, et si oui, autorise le client à y accéder.

Avec cet arrangement, l'utilisateur doit fournir un mot de passe pour chaque serveur, et l'administrateur doit conserver les noms et les mots de passe de chaque utilisateur, généralement sur des serveurs distincts.

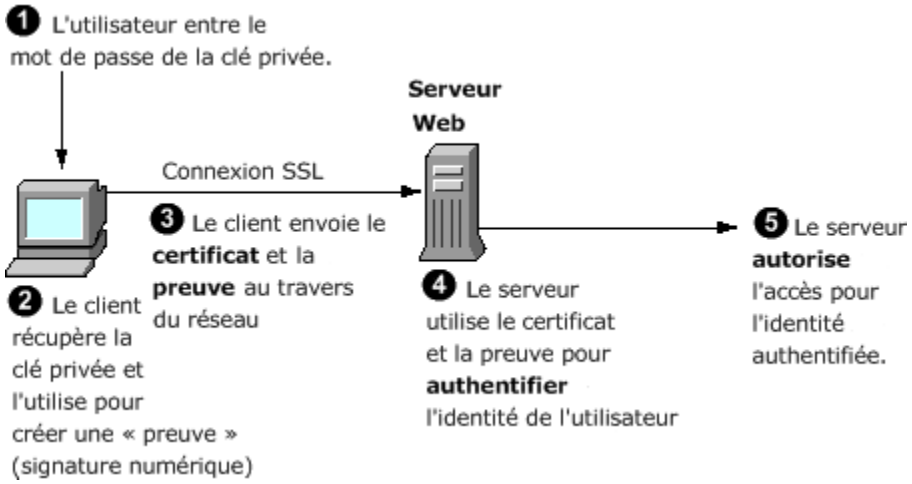
Une implémentation propre ne mémorise pas les mots de passe en texte simple. À la place, il concatène le mot de passe avec une valeur aléatoire propre à chaque utilisateur (également appelée « *salt* ») et mémorise la valeur hachée du résultat avec le « *salt* ». Ceci rend plus difficile des attaques de force brute.

Comme expliqué dans la section suivante, un des avantages de l'authentification par certificat est qu'elle peut être utilisée pour remplacer les trois premières étapes décrites à la figure 4 avec un mécanisme qui permet à l'utilisateur de fournir un seul mot de passe (qui n'est pas transmis à travers le réseau) et permet à l'administrateur de centraliser le contrôle de l'authentification des utilisateurs.

5. L'AUTHENTIFICATION PAR CERTIFICAT

La figure ci-dessous décrit le fonctionnement d'une authentification client à l'aide des certificats et du protocole SSL. Pour authentifier un utilisateur auprès d'un serveur, Le client signe numériquement des données générées aléatoirement et envoie à la fois ces données signées et le certificat sur le réseau. Pour les besoins de cette discussion, la signature numérique associée aux données signées peut être considérée comme une preuve fournie par le client au serveur. Le serveur authentifie l'identité de l'utilisateur en se basant sur la force de cette preuve.

Comme pour la figure du point 3 ci-dessus, la figure ci-dessous suppose que l'utilisateur a déjà décidé de faire confiance dans le serveur et qu'il a demandé une ressource, et que le serveur a demandé une authentification client lors du processus d'évaluation des droits à accéder à la ressource demandée.



Contrairement au processus décrit à la figure du point 3 ci-dessus, il y a nécessite d'utiliser SSL. Il est supposé également que le client possède un certificat valide qui peut être utilisé

pour l'identifier auprès du serveur. L'authentification par certificat est généralement considérée comme préférable à l'authentification par mot de passe car elle est basée sur ce que l'utilisateur a (la clef privée) aussi bien que sur ce que l'utilisateur sait (le mot de passe qui protège cette clef privée). Cependant, il est important de remarquer que ces deux affirmations ne sont vraies que si aucune personne non autorisée n'a accès à l'ordinateur de l'utilisateur ou a son mot de passe, si le mot de passe de la base de données des clefs privées du logiciel client a été défini, et si le logiciel est paramétré pour demander le mot de passe à intervalles raisonnablement fréquents.

Voici les étapes décrites dans la figure ci-dessus :

1. Le logiciel client, tel que le navigateur, maintient une base de données des clefs privées correspondantes aux clefs publiques publiées avec tous les certificats émis pour ce client. Le client demande le mot de passe de cette base de données la première fois qu'il a besoin d'y accéder lors d'une session donnée — par exemple, la première fois que l'utilisateur essaie d'accéder à un serveur SSL qui requiert une authentification par certificat. Après avoir renseigné une première fois ce mot de passe, l'utilisateur n'en a plus besoin pour la durée de la session, même en accédant à d'autres serveurs SSL.
2. Le client débloque la base de données des clefs privées, récupère la clef privée du certificat de l'utilisateur et utilise cette clef privée pour signer numériquement des données générées aléatoirement dans ce but en se basant sur des entrées du client et du serveur. Ces données et la signature numérique constituent une « preuve » de la validité de la clef privée. La signature numérique peut uniquement être créée avec la clef privée et peut être validée par la clef privée associée aux données signées, ce qui est réservé à la session SSL.
3. Le client envoie le certificat de l'utilisateur et la *preuve* (les données générées aléatoirement signées numériquement) par le réseau.
4. Le serveur utilise le certificat et la *preuve* pour authentifier l'identité de l'utilisateur (pour plus de détails sur ce fonctionnement.
5. À ce moment, le serveur peut éventuellement exécuter des tâches d'authentification supplémentaires, comme vérifier si le certificat présenté par le client est stocké dans l'entrée de l'utilisateur d'un annuaire LDAP. Le serveur continue, alors à évaluer si l'utilisateur identifié est autorisé ou non à accéder à la ressource demandée. Ce processus d'évaluation

peut employer une variété de mécanismes standards d'autorisation, en utilisant éventuellement des informations présentes dans un annuaire LDAP, des bases de données d'entreprises, etc. Si le résultat de l'évaluation est positif, le serveur autorise le client à accéder à la ressource demandée.

Comme on peut le voir en comparant les deux figures ci-dessus, les certificats remplacent la portion de l'authentification correspondant à l'interaction entre le client et le serveur. Plutôt que de demander à l'utilisateur d'envoyer des mots de passe par le réseau à longueur de journée, l'ouverture de session unique demande une seule fois à l'utilisateur de saisir son mot de passe de base de données de clefs privée, sans l'envoyer par le réseau. Pour la suite de la session, le client présente le certificat de l'utilisateur pour authentifier l'utilisateur auprès de chaque serveur auquel il se connecte. Les mécanismes d'autorisation existants basés sur l'authentification de l'identité du l'utilisateur ne sont pas concernés.

CHAPITRE VIII : LA STÉGANOGRAPHIE

DÉFINITION

Comme son nom l'indique, la stéganographie ne permet pas de chiffrer, mais de *cacher*. Alors qu'un texte codé attire irrémédiablement l'attention, quoi de plus ennuyeux qu'une image anodine, ou un texte apparemment sans intérêt ?

La lettre de George Sand

George Sand était le pseudonyme d'Amantine Aurore Lucile Dupin, écrivain français née à Paris le 1er juillet 1804 et morte à Nohant le 8 juin 1876. Elle s'adonnait à tous les genres littéraires : depuis les romans et les nouvelles jusqu'aux critiques et aux textes politiques, en passant par les pièces de théâtre. Parallèlement à ça, George Sand se passionnait pour la peinture et s'impliquait beaucoup dans la vie politique, notamment lors du gouvernement provisoire de 1848.

On a longtemps attribué à George Sand la lettre qui suit, destinée à Alfred de Musset (autre grand écrivain français). Cependant, il s'est rapidement avéré **qu'il s'agissait d'un canular** qui remonte au dernier quart du XIX^{e}siècle (Source : Les Amis de George Sand). Cela dit, les textes en eux-mêmes n'en restent pas moins de qualité et méritent tout de même le coup d'œil.

DE SAND À MUSSET

Vous l'aurez compris, l'astuce consiste à **lire une ligne sur deux**. Notez l'élégance manifeste du texte lorsqu'on le lit normalement : un canular, oui, mais un canular de qualité !

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 ou je
vis est bien long, bien dur et souvent bien>
insupportable. Mon chagrin est trop
gros. Accourrez bien vite et venez me le
faire oublier. À vous je veux me sou-
mettre entièrement.
Votre poupée

REFERENCES BIBLIOGRAPHIQUES

1. CRYPTAGE, Notes en ligne sur http://www.cryptage.org/sommaire.html

2. Sécurité des Systèmes d'information: la cryptographie appliquée, GNU Documentation License, 2007-2008

3. Introduction à la cryptographie, Christophe Chabot, Université de Limoges, XLIM-DMI,

4. Introduction à la cryptographie, Véronique Cortier, Ecole des Mines, 3e année

5. Exercices et problèmes de cryptographie, Damien Vergnaud, Dinod, 2e édition, 2015

6. https://fr.wikipedia.org/wiki/

7. http://www.futura-sciences.com/sciences/definitions/physique-cryptographie-quantique-10172/

8. http://www.huffingtonpost.fr/2016/08/16/satellite-quantique-chine_n_11539416.html