

Introduction à la Cryptographie

par Raymond ([homepage](#))

Date de publication : 8 janvier 2009

Dernière mise à jour : 22 février 2009



Depuis le début des civilisations, le besoin de dissimuler préoccupe l'humanité. La confidentialité apparaissait notamment nécessaire lors des luttes pour l'accès au pouvoir. Puis elle a été énormément développée pour les besoins militaires et diplomatiques.

Aujourd'hui, de plus en plus d'applications dites **civiles** nécessitent la sécurité des données transitant entre deux interlocuteurs via un vecteur d'information comme les réseaux de télécommunications actuels et futurs.

Ainsi les banques l'utilisent pour assurer la confidentialité des opérations avec leurs clients, les laboratoires de recherche s'en servent pour échanger des informations dans le cadre d'un projet d'étude commun, les chefs militaires pour donner leurs ordres de bataille, etc.

Le but de ce document, est de présenter les fondements et le fonctionnement de la cryptographie. Il intéressera le

lecteur qui connaît peu ou pas le domaine et qui souhaiterait comprendre le fonctionnement et les mécanismes mis en œuvre en cryptographie. Un minimum de culture mathématique sera parfois nécessaire lors de la lecture de certains paragraphes.

1 - Introduction.....	5
2 - Terminologie.....	6
3 - Cryptographie historique.....	7
3.1 - La technique grecque.....	7
3.2 - Le code de César.....	7
3.3 - Le code de Vigenère.....	7
3.4 - La machine Enigma.....	9
3.5 - Conclusion.....	9
4 - La stéganographie.....	11
4.1 - Le texte caché.....	11
4.2 - La stéganographie dans les fichiers images.....	11
4.3 - Un canal caché dans le protocole ICMP.....	12
4.4 - Conclusion.....	12
5 - La cryptographie moderne.....	13
5.1 - Buts de la cryptographie.....	13
5.2 - Alice, Bernard et les autres.....	13
5.3 - Quelques nombres pour fixer les idées.....	14
5.4 - Les différents modes de chiffrement.....	14
5.4.1 - Le mode ECB.....	15
5.4.2 - Le mode CBC.....	15
5.4.3 - Le mode CFB.....	16
5.4.4 - Le mode OFB.....	17
5.5 - Quel mode choisir ?.....	18
6 - Les algorithmes de calcul d'empreinte.....	20
6.1 - CHECKSUM.....	20
6.2 - CRC.....	21
6.3 - L'algorithme MD5.....	21
6.4 - L'algorithme SHA-1.....	22
6.5 - D'autres algorithmes.....	23
7 - Les algorithmes symétriques.....	25
7.1 - L'algorithme DES.....	25
7.2 - L'algorithme 3-DES.....	27
7.3 - L'algorithme AES.....	28
7.4 - L'algorithme IDEA.....	28
7.5 - L'algorithme du masque jetable.....	28
7.6 - D'autres algorithmes.....	28
8 - Les algorithmes asymétriques.....	30
8.1 - L'algorithme DH.....	31
8.2 - L'algorithme RSA.....	32
8.3 - D'autres algorithmes.....	34
9 - Les méthodes de génération de nombres aléatoires.....	35
9.1 - Les générateurs linéaires congruents.....	36
9.2 - Les générateurs à base de registre à décalage.....	37
9.3 - L'algorithme BBS.....	38
9.4 - D'autres algorithmes.....	39
10 - Quelques applications de la cryptographie.....	40
10.1 - PGP.....	40
10.2 - Notion de Certificat.....	41
10.3 - Notions de PKI.....	44
10.3.1 - Les rôles d'une PKI.....	44
10.3.2 - Les entités d'une PKI.....	45
10.3.3 - L'autorité de certification.....	45
10.4 - SSL.....	46
10.4.1 - Le protocole Handshake.....	46
10.4.2 - Le protocole Change cipher spec.....	46
10.4.3 - Le protocole Alert.....	46
10.4.4 - La phase de négociation du protocole Handshake.....	47
10.4.5 - Les algorithmes négociés par SSL.....	48

10.4.6 - Les ports utilisés par SSL.....	48
10.5 - OpenSSL.....	49
10.5.1 - Contenu de la bibliothèque.....	49
10.5.2 - L'interface ligne de commande.....	49
10.6 - Crypto++.....	49
11 - La cryptographie quantique.....	51
11.1 - Propriétés quantiques d'un photon.....	51
11.2 - Transmission de la clé.....	52
11.3 - Etat actuel de la cryptographie quantique.....	53
12 - Conclusion.....	55
13 - Annexes.....	56
13.1 - Quelques livres.....	56
13.2 - Quelques sites Internet.....	57
13.3 - Solution au courrier entre George Sand et Alfred de Musset.....	58

1 - Introduction

La cryptographie a évolué en trois périodes historiques :

- La cryptographie mécanique. Il s'agit de la cryptographie qui utilise des moyens mécaniques pour chiffrer un message. Cette cryptographie s'étend de l'antiquité jusqu'à la fin de la seconde guerre mondiale environ. De nos jours, elle n'a plus cours.
- La cryptographie mathématique. Il s'agit de la cryptographie qui utilise les mathématiques pour chiffrer un message. Cette cryptographie a commencé aux environs de la fin de la deuxième guerre mondiale et c'est celle que l'on utilise de nos jours.
- La cryptographie quantique. Il s'agit de la cryptographie dont les bases reposent sur la physique quantique. Nous sommes en train de la voir émerger de nos jours et nul doute qu'elle ne remplace dans les années qui viennent la cryptographie basée sur les mathématiques.

Cette présentation ne veut pas être une référence complète dans laquelle tous les détails très techniques sont explicités. Le but de ce document est de fournir suffisamment de pistes au lecteur qui voudrait approfondir certains détails.

Cette présentation n'est pas forcément complète non plus, certains points (les différentes méthodes de cryptanalyse par exemple) ont été volontairement laissés de côté.

N'hésitez pas à me contacter par messagerie privée si vous constatez une faute d'orthographe, une erreur dans mes explications, une omission dans les thèmes abordés ou bien un détail difficilement compréhensible, je vous en remercie d'avance et les futurs lecteurs aussi.

2 - Terminologie

Dans cette présentation je m'attacherai à utiliser toujours les mêmes termes. Ce paragraphe présente les termes utilisés ainsi que leur signification.

- Texte en clair : c'est le message à protéger.
- Texte chiffré : c'est le résultat du **chiffrement** du **texte en clair**.
- Chiffrement : c'est la méthode ou l'algorithme utilisé pour transformer un **texte en clair** en **texte chiffré**.
- Déchiffrement c'est la méthode ou l'algorithme utilisé pour transformer un **texte chiffré** en **texte en clair**.
- Clé : c'est le secret partagé utilisé pour **chiffrer** le **texte en clair** en **texte chiffré** et pour **déchiffrer** le **texte chiffré** en **texte en clair**. On peut parfaitement concevoir un algorithme qui n'utilise pas de **clé**, dans ce cas c'est l'algorithme lui-même qui constitue la **clé**, et son principe ne doit donc en aucun cas être dévoilé.
- Cryptographie : cette branche regroupe l'ensemble des méthodes qui permettent de **chiffrer** et de **déchiffrer** un **texte en clair** afin de le rendre incompréhensible pour quiconque n'est pas en possession de la **clé** à utiliser pour le **déchiffrer**.
- Cryptanalyse : c'est l'art de révéler les **textes en clair** qui ont fait l'objet d'un **chiffrement** sans connaître la **clé** utilisée pour **chiffrer** le **texte en clair**.
- Cryptologie : il s'agit de la science qui étudie les communications secrètes. Elle est composée de deux domaines d'étude complémentaires : la **cryptographie** et la **cryptanalyse**.
- Décrypter : c'est l'action de retrouver le **texte en clair** correspondant à un **texte chiffré** sans posséder la clé qui a servi au chiffrement. Ce mot ne devrait donc être employé que dans le contexte de la cryptanalyse.
- Crypter : en relisant la définition du mot **décrypter**, on peut se rendre compte que le mot **crypter** n'a pas de sens et que son usage devrait être oublié. Le mot **cryptage** n'a pas plus de sens non plus.
- Coder, décoder : c'est une méthode ou un algorithme permettant de modifier la mise en forme d'un message sans introduire d'élément secret. Le Morse est donc un code puisqu'il transforme des lettres en trait et points sans notion de secret. L'ASCII est lui aussi un code puisqu'il permet de transformer une lettre en valeur binaire.

3 - Cryptographie historique

Ce paragraphe présente plusieurs **algorithmes** ou méthodes de cryptographie à une époque où les mathématiques ne régnaient pas encore en maîtres sur ce domaine.

3.1 - La technique grecque

Une méthode de chiffrement datée entre le Xème et VIIème siècle avant Jésus Christ repose sur l'utilisation d'un bâton appelé **scytale** d'un diamètre fixé. Une lanière en cuir était enroulée en hélice autour de ce bâton et le **texte en clair** était alors écrit sur la lanière. Ensuite, la lanière était déroulée et pouvait être envoyée (sans le bâton) au destinataire du message.



Scytale grecque (source Wikipédia)

Pour déchiffrer le texte chiffré, il suffisait de d'utiliser un bâton possédant exactement le même diamètre que le précédent, d'y enrouler la lanière de cuir et le **texte en clair** pouvait alors être relu.

Le procédé utilisé par cette méthode est un chiffrement par **transposition**, c'est-à-dire que les lettres ne sont pas modifiées mais que seul l'ordre des lettres est changé.

3.2 - Le code de César

Le code de César est une méthode connue par tous les écoliers. Il suffit de décaler les lettres d'un certain nombre connu aussi bien par celui qui écrit le message que par celui qui le reçoit.

Par exemple si $n=4$, cela donne :

- $a=E, b=F, c=G, \dots w=A, x=B, y=C, z=D$.

Si le **texte en clair** à chiffrer est "**rendons a cesar ce qui est a cesar**", avec un décalage de quatre lettres, le texte chiffré est "**VIRHSRW E GIWEV GI UYM IWX E GIWEV**".

Ce procédé est un procédé de chiffrement par **substitution mono-alphabétique**. Il est très simple mais il ne résiste pas à une analyse basée sur la fréquence des caractères puisque chaque lettre est toujours remplacée par la même lettre. D'ailleurs dans l'exemple fourni, le mot **cesar** est chiffré deux fois de la même manière en **GIWEV** de même que le **a** est chiffré deux fois en **E**.

Un autre algorithme, nommé ROT13, basé sur le même fonctionnement a existé aux tous débuts de l'informatique. Le décalage était de treize lettres, et c'est donc le même algorithme qui était utilisé aussi bien pour le chiffrement que pour le déchiffrement (un premier décalage de treize lettres pour chiffrer suivi d'un autre décalage de treize lettres pour déchiffrer suffisait à redonner le **texte en clair**).

3.3 - Le code de Vigenère

Le code de Vigenère est présenté pour la première fois par le diplomate Blaise de Vigenère au courant du 16ème siècle.

Il reprend pour cela le principe du code de César mais il le complexifie en introduisant la notion de décalage variable en fonction d'une clé. Le code de Vigenère repose d'abord sur le carré de Vigenère. Il s'agit de l'alphabet qui est répété sur 26 lignes mais décalé d'une lettre à chaque fois.

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

Il nécessite ensuite un mot clé seulement connu par l'émetteur et le destinataire du message. C'est le premier algorithme à introduire la notion de clé. Ce mot clé est répété autant de fois que nécessaire afin d'avoir autant de lettres (ou plus) que le **texte en clair** à chiffrer. Ensuite l'alphabet utilisé pour chiffrer une lettre est celui correspondant à la lettre du mot clé.

Si le texte à chiffrer est "rendons a vigenere ce qui est a vigenere", avec le mot clé "RAYMOND", le texte chiffré est "IELPCAV R VGSSAHIE AQ EHL VSR M JVJVNCD S".

```
rendons a vigenere ce qui est a vigenere
RAYMONDRAYMONDRAYMONDRAYMONDRAYMONDRAYMOND
```

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

Alphabet à utiliser pour la lettre A de la clé

Alphabet à utiliser pour la lettre D de la clé

Alphabet à utiliser pour la lettre N de la clé

Alphabet à utiliser pour la lettre N de la clé

Alphabet à utiliser pour la lettre O de la clé

Alphabet à utiliser pour la lettre R de la clé

Alphabet à utiliser pour la lettre Y de la clé

Dans cet exemple on voit que les deux mots **vignere** ne sont plus codés de la même manière (**VGSSAHIE** et **JVJVNCD**) de même pour la lettre **a** qui est codée en **R** et **M**.

Ce code ne sera cassé qu'en 1854 (soit 200 ans plus tard) par le mathématicien Charles Babbage.

3.4 - La machine Enigma

La machine Enigma est née en 1918 dans l'entreprise de l'inventeur Arthur Scherbius en Allemagne. Elle a été utilisée par l'armée allemande durant la seconde guerre mondiale.

Il s'agit d'une machine à chiffrer et à déchiffrer mécanique qui allie à la fois les méthodes de substitution et de transposition.

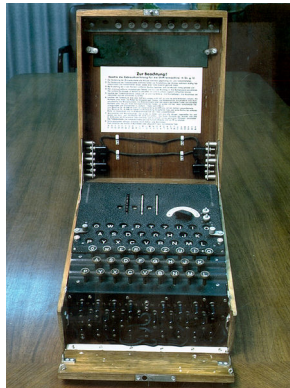


Photo d'une machine Enigma (source Wikipédia)

Cette machine se compose de :

- Un tableau de connexion qui permet d'échanger 6 paires de lettres 2 à 2
- 3 rotors choisis parmi les 6 rotors existants. Chaque rotor comprend 26 positions différentes. La position du premier rotor avance avec chaque lettre. Au bout de 26 lettres et donc 26 décalages, le premier rotor est revenu à sa position initiale et c'est le second rotor qui avance d'une lettre et ainsi de suite.
- 1 réflecteur qui permet de repasser la lettre en cours de chiffage dans les rotors et dans le tableau de connexion.

Le nombre de clés possibles est le suivant :

- Choix de 6 paires de lettres parmi 26 lettres = 100 391 791 500 possibilités
- Choix de 3 rotors parmi 6 = 120 possibilités
- Choix de la position initiale des 3 rotors parmi les 26 positions possibles = $26 * 26 * 26 = 17\,576$ possibilités
- Ce qui fait un nombre total de 211 738 335 288 480 000 clés différentes.

Le code utilisé par la machine Enigma sera lui aussi cassé durant la seconde guerre mondiale. Les polonais avant et au tout début de la seconde guerre mondiale auront commencé le travail de compréhension de l'algorithme. Les anglais à Bletchley Park avec l'aide d'Alan Turing finiront le travail.

3.5 - Conclusion

Bien sûr, il y a eu de nombreux autres codes utilisant des variations sur ces codes ou alors de nouveaux concepts. Mais la plupart des procédés cryptographiques reposent essentiellement sur deux moyens :

- La **transposition**, l'ordre des lettres du message est modifié dans le message.
- La **substitution**, les lettres sont remplacées par d'autres lettres selon une certaine règle.

En ce qui concerne la substitution, il faut noter qu'il y en a plusieurs :

- Substitution mono-alphabétique : chaque lettre du message est remplacée par une autre lettre de l'alphabet. Le code de César utilise cette technique.
- Substitution poly-alphabétique : chaque lettre du message est remplacée par une autre lettre d'un alphabet indiqué par une lettre de la clé. Le code de Vigenère utilise cette technique.
- Substitution homophonique : chaque lettre du message est remplacée par une des lettres de substitution possibles. Le nombre de lettres de substitution possibles est fonction de la fréquence de la lettre dans le langage concerné. Cette méthode de substitution a pour but d'empêcher le déchiffrement du message par une analyse fréquentielle.
- Substitution polygrammes : chaque groupe de caractères est remplacé par un autre groupe de caractères.

4 - La stéganographie

La stéganographie n'est pas vraiment une méthode cryptographique. Elle repose sur le fait que l'information à cacher est mélangée avec de l'information banale de manière à passer inaperçue.

4.1 - Le texte caché

Un grand classique de la stéganographie est cet échange de lettres entre George Sand et Alfred de Musset (cet échange fut-il réel ou bien fait il partie des légendes urbaines du Web, je ne sais pas, mais le contenu est tout de même beau) :

```
Je suis très émue de vous dire que j'ai
bien compris l'autre soir 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
là une preuve que je puisse être aimée
par vous. Je suis prête à vous montrer mon
affection toute désintéressée et sans cal-
cul, et si vous voulez me voir aussi
vous dévoiler sans artifice mon âme
toute nue, venez me faire une visite.
Nous causerons 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
en amitié, en un mot la meilleure preuve
dont vous puissiez rêver, puisque votre
âme est libre. Pensez que la solitude où j'ha-
bite est bien longue, bien dure et souvent
difficile. Ainsi en y songeant j'ai l'âme
grosse. Accourrez donc vite et venez me la
faire oublier par l'amour où je veux me
mettre.
```

La réponse d'Alfred de Musset :

```
Quand je mets à vos pieds un éternel hommage
Voulez-vous qu'un instant je change de visage ?
Vous avez capturé les sentiments d'un cœur
Que pour vous adorer forma le Créateur.
Je vous chéris, amour, et ma plume en délire
Couche sur le papier ce que je n'ose dire.
Avec soin, de mes vers lisez les premiers mots
Vous saurez quel remède apporter à mes maux.
```

Et enfin la réponse de George Sand :

```
Cette insigne faveur que votre cour réclame
Nuit à ma renommée et répugne mon âme.
```

Et vous, saurez-vous retrouver le message caché dans cet échange de lettres (la solution se trouve à la fin de ce document dans les annexes) ?

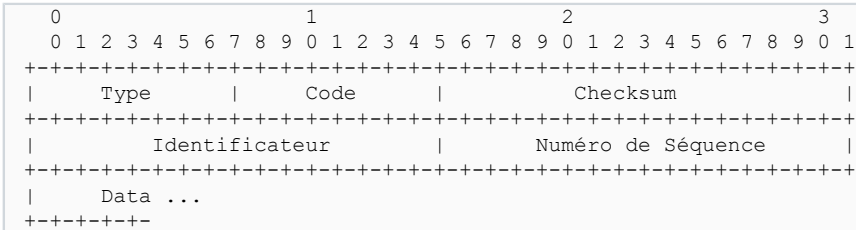
4.2 - La stéganographie dans les fichiers images

De la même manière que pour le texte, il est possible de cacher de l'information dans des fichiers images. En modifiant ou altérant quelques bits du fichier, il est ainsi possible de cacher un copyright ou alors un message de son choix sans que cela se voit. En effet, cette altération sera peu ou pas visible car l'œil humain n'est pas capable de discerner des petites aberrations sur une grande image à condition que le ratio taille du message / taille de l'image ne soit pas trop grand.

4.3 - Un canal caché dans le protocole ICMP

Le protocole ICMP est un protocole utilisé pour contrôler le fonctionnement d'un réseau IP. Une des applications la plus connue est l'utilitaire **ping** qui permet de connaître la présence ou non d'une machine sur le réseau.

Le format d'une trame ICMP est le suivant :



Pour les trames **ICMP echo request** et **ICMP echo reply** qui sont les trames générées et reçues par l'utilitaire **ping**, les valeurs des différents champs sont les suivantes :

- Type : 8 pour un message **echo request** et 0 pour un message **echo reply**.
- Code : toujours 0.
- Checksum : Ce champ est la somme de contrôle de la trame ICMP.
- Identificateur : Il s'agit d'un nombre utilisé pour aider à repérer les réponses.
- Numéro de séquence : Il s'agit d'un nombre utilisé pour aider à repérer les réponses.
- Data : des données libres et de longueur variable.

Avec un tel message, il est tout à fait possible d'imaginer que le champ **Data** de la trame du message ICMP soit utilisé pour transporter des données applicatives. Il serait ainsi possible d'imaginer un protocole complexe reposant sur des trames d'apparences anodines. Après tout, qui regarde le contenu du champ data des trames ICMP ?

Pour que ce protocole fonctionne, il suffit juste que la machine A puisse faire un **ping** de la machine B et que la machine B puisse faire un **ping** de la machine A.

Ce protocole fictif mais réalisable est ni plus ni moins que de la stéganographie.

4.4 - Conclusion

En conclusion, il faut retenir que la stéganographie est une méthode de cryptographie faible : elle repose uniquement sur le fait que personne ne remarquera le canal caché. Dès lors que ce canal est connu, il n'y a plus aucune protection.

5 - La cryptographie moderne

La cryptographie moderne repose maintenant uniquement sur les mathématiques. De plus, les règles de bases sont :

- L'algorithme utilisé n'est pas secret. Il peut être diffusé librement, cela ne doit avoir aucun impact sur la facilité ou non à déchiffrer le message.
- La clé de chiffrement utilisée est secrète.

Un protocole cryptographique basé sur la non-divulcation de l'algorithme mathématique utilisé n'est pas fiable. Tôt ou tard, l'algorithme utilisé sera connu et le protocole deviendra faible. Au contraire, diffuser l'algorithme mathématique utilisé permet à la communauté de valider et de tester la robustesse de cet algorithme.

5.1 - Buts de la cryptographie

Globalement, la cryptographie permet de résoudre quatre problèmes différents :

- La confidentialité. Le texte chiffré ne doit être lisible que par les destinataires légitimes. Il ne doit pas pouvoir être lu par un intrus.
- L'authentification. Le destinataire d'un message doit pouvoir s'assurer de son origine. Un intrus ne doit pas être capable de se faire passer pour quelqu'un d'autre.
- L'intégrité. Le destinataire d'un message doit pouvoir vérifier que celui-ci n'a pas été modifié en chemin. Un intrus ne doit pas être capable de faire passer un faux message pour légitime.
- La non répudiation. Un expéditeur ne doit pas pouvoir, par la suite, nier à tort avoir envoyé un message.

Il ne faut pas confondre cryptographie et codage. En effet, la cryptographie va s'attacher à cacher le sens d'un texte ; Le codage quant à lui s'attache à modifier le texte de façon à utiliser un support particulier. Ainsi par exemple, il existe le code Morse qui transforme les chiffres et lettres d'un texte en une succession de traits et de points afin de pouvoir utiliser le support télégraphique. Il existe aussi le code ASCII qui transforme les caractères en valeurs codées sur 8 bits (de 0 à 255) pour utiliser les supports informatiques. Il n'y a strictement rien de secret dans un codage.

5.2 - Alice, Bernard et les autres

Traditionnellement, pour illustrer les protocoles, on parle de communication entre des personnes fictives. Pour la cryptographie, par convention, ces personnes sont :

Français	Anglais	Rôle
Alice	Alice	Alice veut envoyer un message à Bernard
Bernard	Bernard	Bernard communique avec Alice
Christine	Carol	S'il faut une 3ème personne pour communiquer avec Alice et Bernard, c'est Christine.
David	Dave	S'il faut une 4ème personne pour communiquer avec Alice, Bernard et Christine, c'est David.
Estelle	Eve	Estelle est une espionne qui peut écouter les communications entre Alice et Bernard.
Martin	Mallory	Martin est un attaquant actif. Au contraire d'Estelle, il peut modifier les messages qu'il

		écoute, rejouer d'anciens messages, substituer ses propres messages, etc.
--	--	---

Il existe encore d'autres prénoms utilisés : Oscar, Trudy, Ivan, Gatien, Isaac, Nestor, etc. avec chacun un rôle identifié dans un protocole cryptographique.

Ces noms ont été utilisés pour la première fois par Ronald Rivest dans son article de 1978 dans lequel il présentait les principes de l'algorithme RSA.

5.3 - Quelques nombres pour fixer les idées

Comme on le verra plus loin, les méthodes cryptographiques utilisent des très grands nombres. Il est utile de comparer ces nombres par rapport à quelques mesures physiques de tous les jours. Attention toutefois, les nombres fournis dans le tableau suivant sont des approximations ou des valeurs couramment admises dans l'état actuel de nos connaissances.

Nombre de secondes dans un jour	86 400 secondes
Nombre de secondes dans une année	31 536 000 secondes
Nombre de secondes écoulées depuis la création de l'univers (13,7 milliard d'années)	432 043 200 000 000 000 secondes ou $432 * 10^{15}$ secondes
Durée de vie d'un proton	10^{30} années soient 10^{20} fois l'âge de l'univers
Masse du soleil	$2 * 10^{33}$ grammes
Nombre d'atomes dans un gramme de matière	$0,5 * 10^{24}$ atomes
Nombre d'atomes dans l'univers	10^{80} atomes

Le but de ces quelques nombres est de montrer l'inutilité des attaques dites **force brute**. En effet, un cryptanalyste en herbe pourrait se dire "**Ce message est chiffré avec l'algorithme AES-256, il y a 2^{256} clés possibles (soient environ 10^{76} clés), testons les toutes pour déchiffrer le message**". A raison de 100 milliards de tentatives par seconde (ce qui est énorme), il faudrait 10^{58} secondes pour tester toutes les clés. Ce temps de calcul nécessaire est beaucoup plus long que l'âge de l'univers.

5.4 - Les différents modes de chiffrement

Le mode de chiffrement correspond à la manière dont on va utiliser un algorithme de chiffrement donné. Ce mode de chiffrement consiste par exemple à rajouter de la contre-réaction entre l'entrée et la sortie de l'algorithme afin de lui rajouter des caractéristiques bien précises. Les différents modes de chiffrement utilisés sont les suivants :

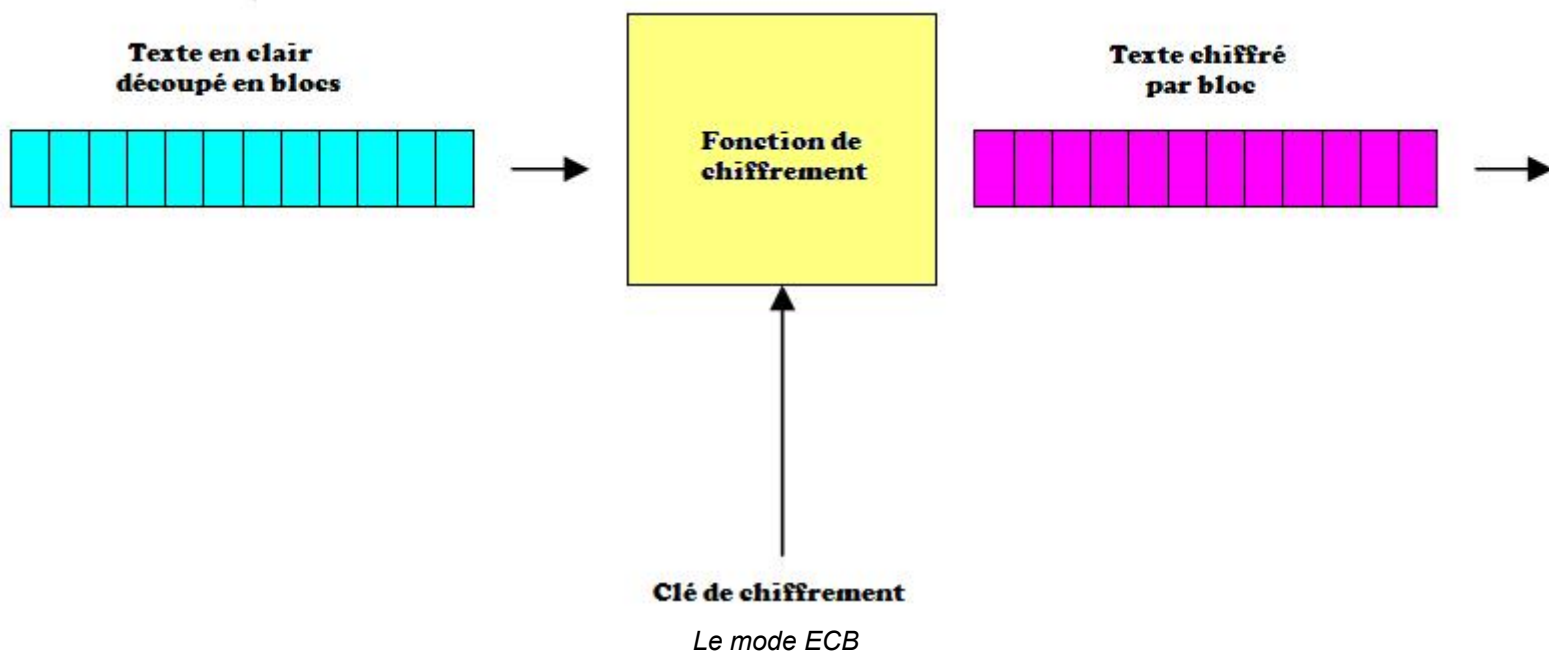
- Le mode ECB pour **Electronic Code Book**
- Le mode CBC pour **Cipher Block Chaining**
- Le mode **chiffrement en continu**
- Le mode CTAK pour **Cipher Text Auto Key**
- Le mode CFB pour **Cipher Feed Back**
- Le mode KAK pour **Key Auto Key**
- Le mode OFB pour **Output Feed Back**
- Le mode CTR pour **CounTeR**
- Le mode BC pour **Block Chaining**
- Le mode PCBC pour **Propagating Cipher Block Chaining**
- Le mode CBCC pour **Cipher Block Chaining with Checksum**

- Le mode OFB pour **Output Feed Back mode with a Non Linear Function**
- Le mode PBC pour **Plaintext Block Chaining**
- Le mode PFB pour **Plaintext Feed Back**
- Le mode CBCPD pour **Cipher Block Chaining of Plaintext Difference**
- Le mode CTS pour **Cipher Text Stealing**

Dans les paragraphes suivants nous allons regarder un peu plus en détail les quatre modes ECB, CBC, CFB et OFB. Les autres modes sont nommés à des fins d'exhaustivité mais sont rarement utilisés.

5.4.1 - Le mode ECB

Ce mode est le plus simple : un même bloc est toujours codé de la même manière. Il n'y a pas de rétroaction de l'entrée ou de la sortie sur la fonction de chiffrement.



Les avantages de ce mode sont les suivants :

- Le travail de chiffrement ou de déchiffrement peut être parallélisé. Plusieurs machines ou CPU peuvent travailler simultanément sur des parties différentes du message.
- Il permet un accès aléatoire dans le **texte chiffré**.
- Une erreur de transmission d'un bit affecte uniquement le décodage du bloc courant.

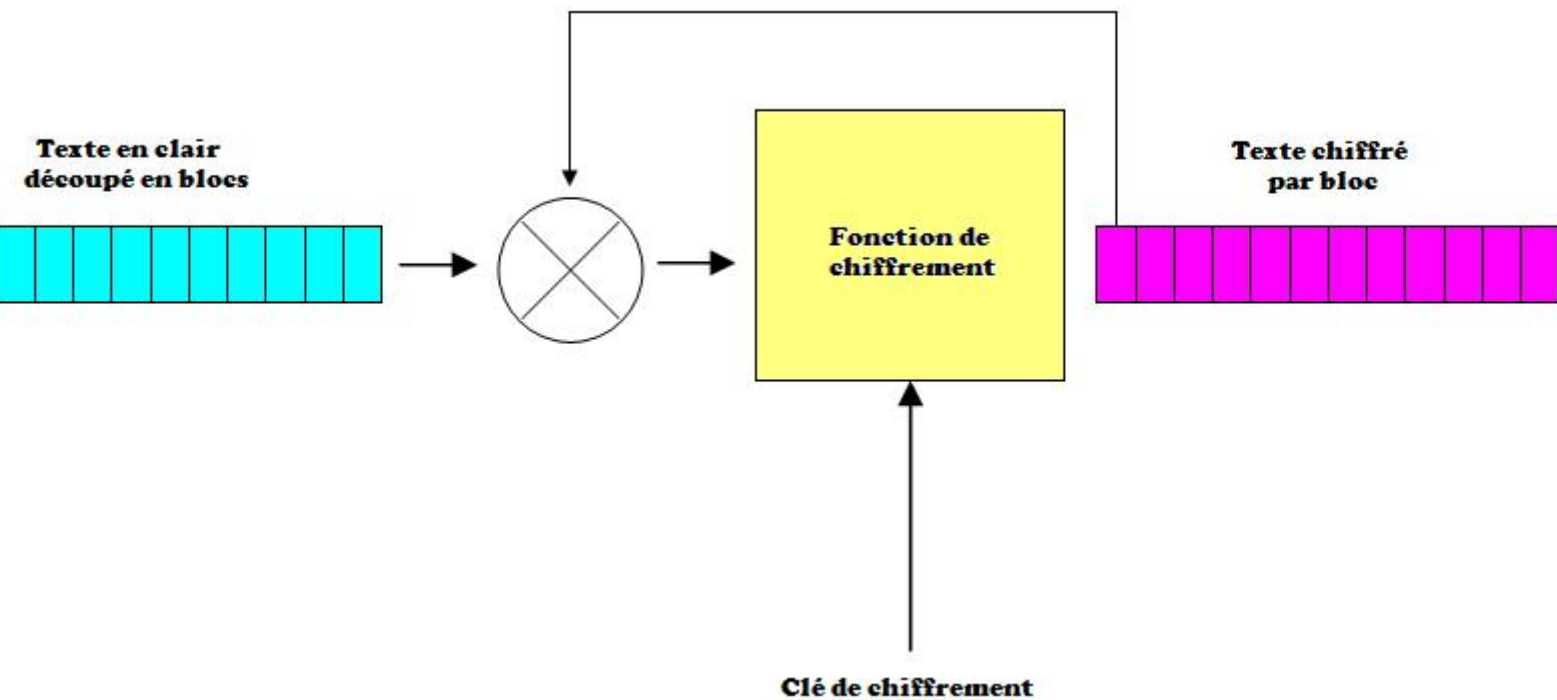
Par contre, ce mode a les désavantages suivants :

- Les répétitions du **texte en clair** ne sont pas masquées et se retrouvent sous la forme de répétitions de **textes chiffrés**.
- Des portions complètes du message peuvent être modifiées, répétées ou remplacées sans difficulté.
- La perte de synchronisation (perte ou ajout d'un bit) est irrécupérable.

5.4.2 - Le mode CBC

Dans ce mode de chiffrement, chaque bloc de texte en clair est d'abord combiné par un **ou exclusif** avec le dernier bloc du texte chiffré. La sortie de ce **ou exclusif** est ensuite appliquée à la fonction de chiffrement.

Ce mode de chiffrement dispose en plus d'un vecteur d'initialisation appelée IV (pour Initialisation Vector) qui permet d'initialiser le processus quand aucun bloc n'a encore été chiffré.



Le mode CBC

Les avantages de ce mode sont les suivants :

- Les répétitions de **texte en clair** sont masquées dans le **texte chiffré**.
- La valeur du vecteur d'initialisation IV n'a pas besoin d'être secrète.

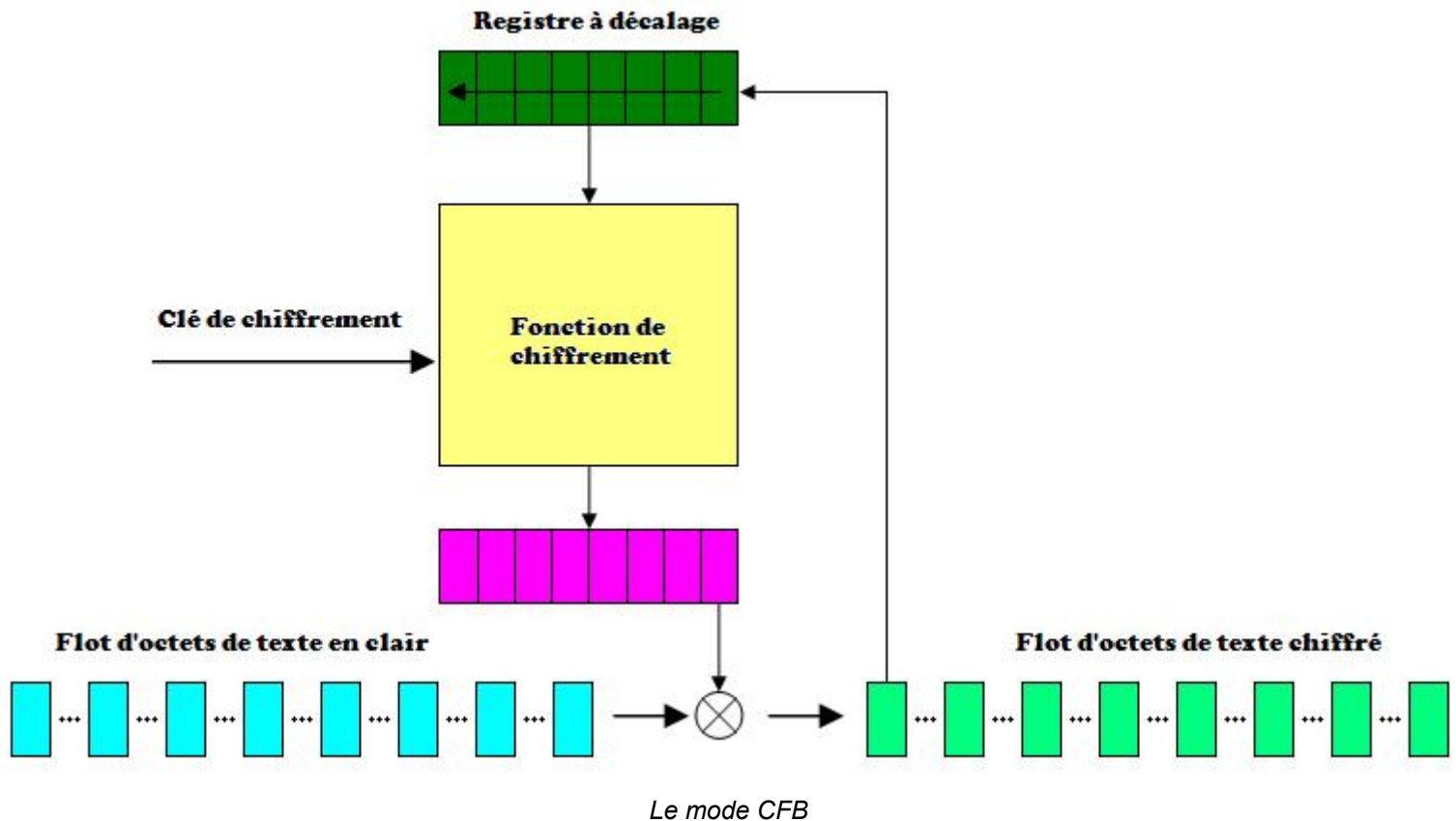
Par contre, ce mode a les désavantages suivants :

- Deux **textes en clair** commençant pareil auront le même début de texte chiffré.
- Une erreur de transmission d'un bit affecte uniquement le décodage du bloc courant ainsi que le décodage du même bit dans le bloc suivant.
- La perte de synchronisation (perte ou ajout d'un bit) est irrécupérable.

5.4.3 - Le mode CFB

Les modes ECB et CBC travaillent avec sur des blocs de **texte en clair** (64 bits par exemple). Ces modes ne sont pas utilisables lorsque le chiffrement ne peut débuter que lorsqu'un bloc est complet. Sur des applications réseau, cela peut poser des problèmes car les valeurs à chiffrer arrivent de manière asynchrone sous forme d'octets et doivent être transmises immédiatement (cas du protocole telnet par exemple).

Le registre à décalage est initialisé avec un vecteur d'initialisation. Le bloc complet est alors chiffré. L'octet de poids faible du texte chiffré est combiné par un **ou exclusif** avec l'octet de texte en clair. Le résultat de cette opération est alors transmis en même temps qu'il est injecté dans le registre à décalage.



Les avantages de ce mode sont les suivants :

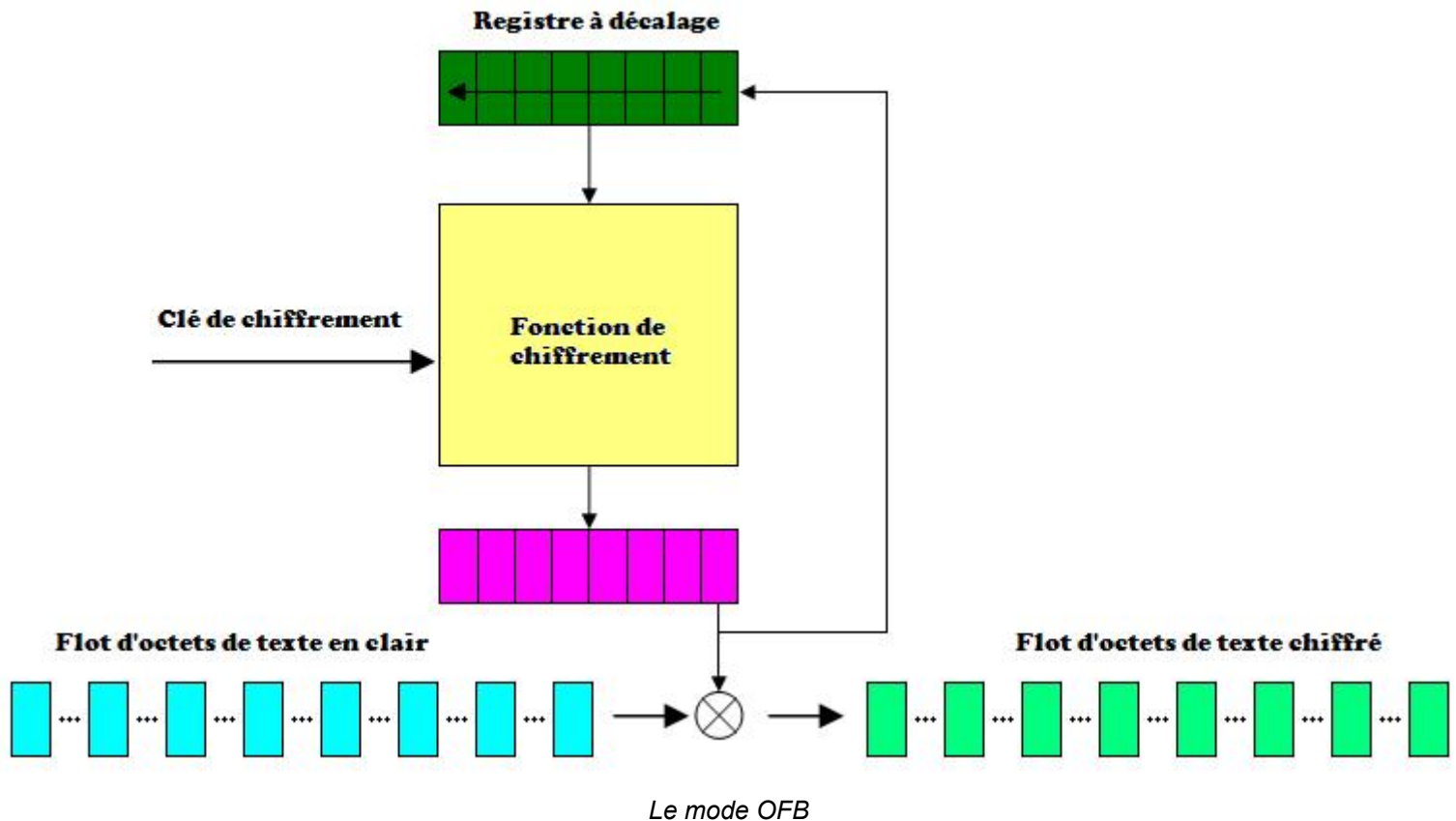
- Il est possible de chiffrer un flot de valeurs plus petites que la taille standard du bloc géré par l'algorithme.
- Les répétitions de **texte en clair** sont masquées dans le **texte chiffré**.
- La valeur du vecteur d'initialisation IV n'a pas besoin d'être secrète.
- La perte de synchronisation (perte ou ajout d'un bit) est récupérable.

Par contre, ce mode a les désavantages suivants :

- Une erreur de transmission d'un bit affecte uniquement le décodage du bloc courant ainsi que le décodage du même bit dans le bloc suivant.

5.4.4 - Le mode OFB

Le mode OFB ressemble au mode CFB. La seule différence est que l'octet injecté dans le registre à décalage est l'octet de poids faible du texte chiffré.



Les avantages de ce mode sont les suivants :

- Les répétitions de **texte en clair** sont masquées dans le **texte chiffré**.
- La valeur du vecteur d'initialisation IV n'a pas besoin d'être secrète.
- Ce mode n'amplifie pas les erreurs. Une erreur de transmission d'un bit affecte uniquement ce bit lors du décodage.

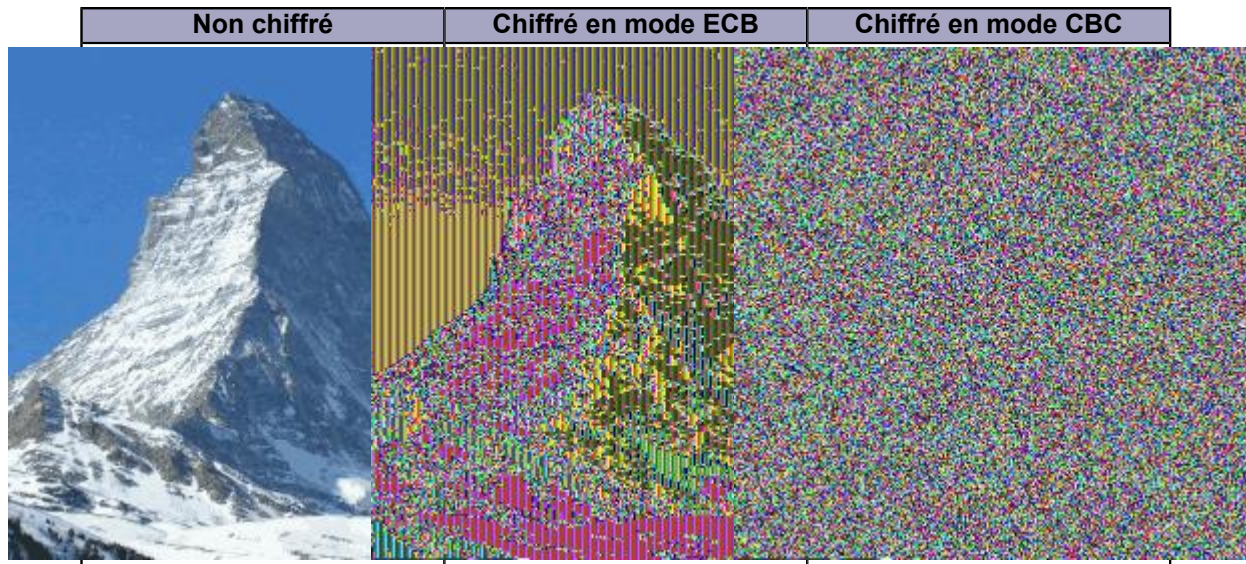
Par contre, ce mode a les désavantages suivants :

- La perte de synchronisation (perte ou ajout d'un bit) est irrécupérable.

5.5 - Quel mode choisir ?

Les quatre modes présentés ainsi que tous les autres ont chacun leurs avantages et leurs faiblesses. Il conviendra de choisir le mode en fonction de ce que l'on veut faire.

Pour finir, voici trois photos du mont Cervin dans les Alpes (source Wikipédia). La première photo est le **texte en clair**, la seconde est le **texte chiffré** de la photo utilisant un algorithme en mode ECB et enfin la dernière photo utilise le même algorithme en mode CBC.



Ces photos montrent clairement que le mode ECB n'est pas adapté au chiffrement de photographies. Ceci ne veut toutefois pas dire que le mode ECB ne doit pas être utilisé, il a d'autres domaines d'utilisation.

Après le parcours des différents modes de chiffrement utilisables, nous allons maintenant nous intéresser aux différents algorithmes de chiffrement. Il existe en tout quatre grandes familles d'algorithmes utilisées par la cryptographie. Ces familles sont :

- Les algorithmes de calcul d'empreinte
- Les algorithmes symétriques
- Les algorithmes asymétriques
- Les méthodes de génération de nombres aléatoires.

Chacune de ces familles est présentée dans les paragraphes suivants.

6 - Les algorithmes de calcul d'empreinte

Un algorithme de calcul d'empreinte est un algorithme qui transforme un **texte en clair** en un nombre (une empreinte numérique) qui soit représentatif de ce **texte en clair**. Le but de ces algorithmes est qu'une altération aussi minime soit elle du **texte en clair** puisse être détectée par le fait que la nouvelle empreinte est différente et même très différente de la précédente. On parle aussi de fonction de hachage.

En général, ce type d'algorithme fournit comme résultat un nombre dont la taille est fixe et ce quelle que soit la taille du **texte en clair** en entrée.

Pour ce type d'algorithme, on parlera aussi de la notion de collision. Une collision de **texte en clair** est le fait que deux **textes en clair** différents provoquent la même empreinte.

Les caractéristiques d'un algorithme d'empreinte utilisé en cryptographie sont les suivantes :

- Difficulté à retrouver le **texte en clair** à partir de l'empreinte.
- Difficulté à trouver un second **texte en clair** qui donne la même empreinte numérique qu'un autre **texte en clair** donné.
- Difficulté à trouver 2 **textes en clair** qui donnent la même empreinte numérique.

Il est évident que plus le nombre de collisions est faible, et donc plus le nombre généré en sortie est large, plus l'algorithme est fiable mais ce n'est pas le seul critère de fiabilité. Il faut aussi qu'il soit dur de générer des **textes en clair** différents pour une empreinte numérique donnée et il faut aussi que la plus petite variation dans le **texte en clair** se traduise par une grande variation dans l'empreinte générée.

Bien que les algorithmes checksum et CRC ne soient pas des algorithmes forts au sens cryptographique du terme, il m'a semblé intéressant de les faire figurer dans ce paragraphe pour des raisons historiques et aussi parce que leurs concepts sont ceux d'une fonction de calcul d'empreinte. Ces deux algorithmes ne sont pas suffisamment forts pour la cryptographie car le nombre d'empreintes possibles est vraiment trop petit (2^{16} pour l'algorithme CHECKSUM ou 2^{32} pour l'algorithme CRC-32 à comparer au 2^{128} du MD5 ou même 2^{160} du SHA-1).

Ce paragraphe s'intéressera ensuite aux deux familles principales d'algorithme de calcul d'empreinte utilisées en cryptographie : la famille des **MD** et la famille des **SHA**.

6.1 - CHECKSUM

La fonction de checksum peut être considérée comme une fonction de calcul d'empreinte. Son but est de vérifier qu'il n'y a pas d'erreur de transmission. En général, les fonctions de checksum sont plutôt utilisées dans les communications mais pas en cryptographie car la taille du nombre généré n'est pas suffisamment grande pour être qualifiée de fiable mathématiquement parlant.

On retrouve par exemple ce type d'algorithme dans l'en-tête des paquets IP.

```

En-tête IP tel que défini par le RFC 791
0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|Version| Taille|Type de Service|          longueur totale          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          Identification          |Flags|      Fragment Offset      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Durée de vie | Protocole |          Checksum d'en-tête          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          Adresse Source          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          Adresse Destination      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

[illegible]

Dans l'en-tête du paquet IP, on retrouve un champ nommé **checksum d'en-tête** qui est la somme de contrôle de l'en-tête. Si par hasard, un bit de l'en-tête venait à être modifié au cours du transfert, cette modification serait probablement détectée par le contrôle du **checksum d'en-tête** et le paquet serait ignoré.

L'algorithme utilisé est le suivant :

- Il s'agit du complément à un sur 16 bits de la somme des compléments à un de tous les octets de l'en-tête pris par paires (mots de 16 bits). Lorsque l'on calcule le checksum, la valeur du checksum de l'en-tête calculé est initialisée à 0.
- Lorsqu'un des champs de l'en-tête IP est modifié (le champ TTL par exemple lorsqu'il est décrémenté par un routeur), ce checksum est recalculé.

Cet algorithme, bien qu'élémentaire et rudimentaire, a montré son efficacité dans le cadre de cette utilisation.

6.2 - CRC

L'acronyme CRC provient de l'anglais **Cyclic Redundancy Check** ou encore en français, **Contrôle de Redondance Cyclique**.

En fait, le CRC est le reste de la division entière d'un nombre par un polynôme. Ce polynôme possède des caractéristiques mathématiques particulières.

Le but du CRC est de détecter une erreur de transmission et dans certains cas seulement, de corriger cette erreur de transmission (si l'erreur ne porte pas sur trop de bits bien sûr).

Une trame Ethernet est terminée par un CRC qui permet de détecter une erreur de transmission, le polynôme utilisé est un des nombreux CRC 32 existants appelé aussi **CRC-32-IEEE 802.3** dont la formule est la suivante :

- $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$

Un autre CRC est utilisé par exemple dans le cadre des protocoles X25 ou HDLC ou PPP, il s'agit d'un des CRC 16 nommé CRC-16-CCITT dont la formule est la suivante :

- $x^{16} + x^{12} + x^5 + 1$

La particularité de ce type d'algorithme est qu'il permet parfois de réparer une erreur de transmission par contre, ce type d'algorithme n'a pas sa place en cryptographie.

On se reportera à l'excellent [tutoriel sur le calcul des CRC](#) écrit par DVSoft.

6.3 - L'algorithme MD5

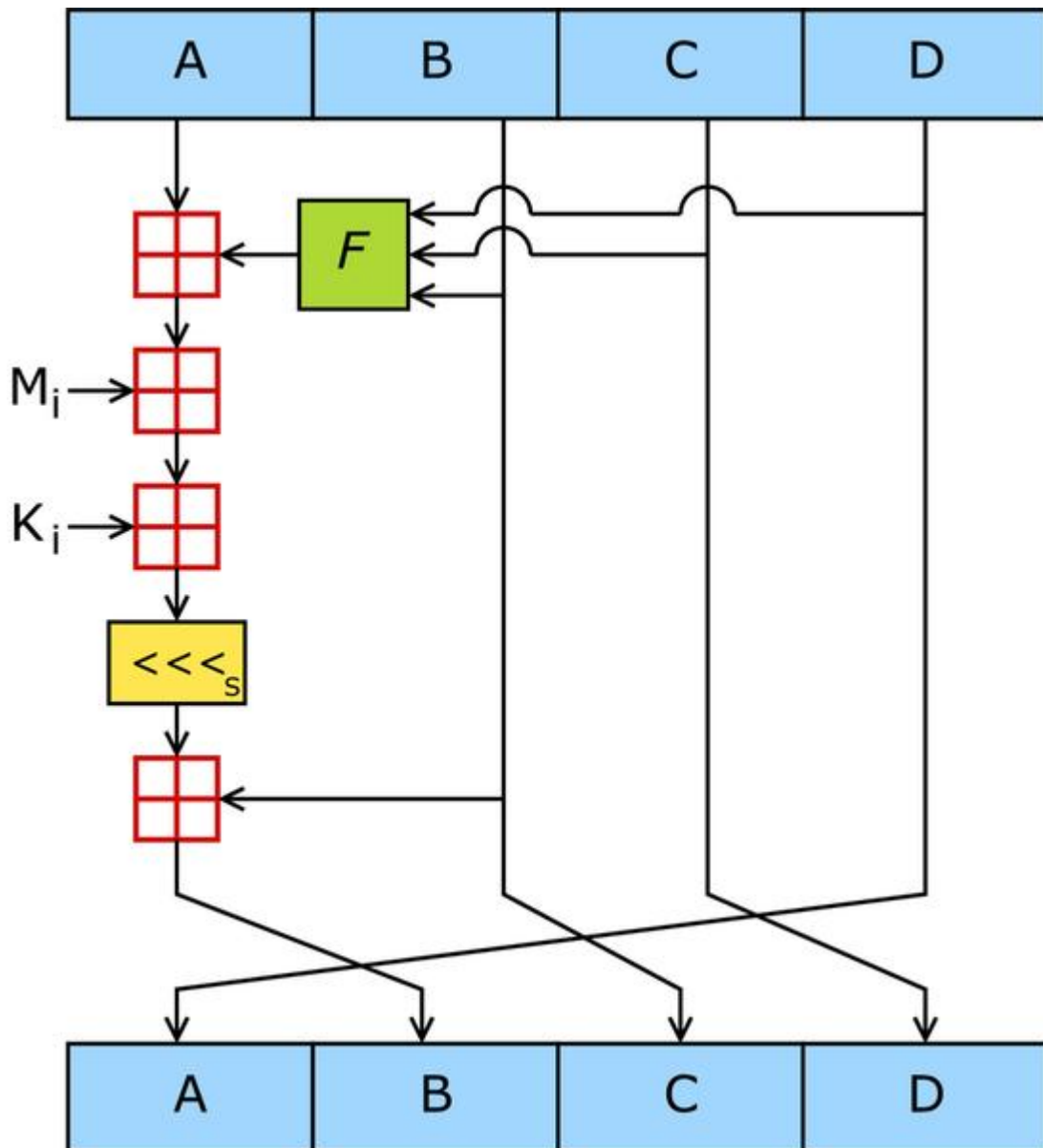
Les algorithmes MD2, MD4 et MD5 sont des algorithmes de calcul d'empreinte numérique inventés par Ronald Rivest. Le mot MD signifie **Message Digest**.

Le résultat généré est un nombre sur 128 bits (16 octets) à partir de messages de 512 bits (64 octets).

MD2 est la première version de cet algorithme inventée en 1989. Il a ensuite été supplanté par l'algorithme MD4. Finalement, l'algorithme MD4 a été abandonné suite à des faiblesses de conception mises en évidence par des attaques.

MD5 est l'évolution de MD4 faite par Ronald Rivest en 1991 afin de pallier les faiblesses identifiées. Cet algorithme est entièrement décrit dans la RFC 1321. Cette RFC comprend entre autres une implémentation de l'algorithme en C qui est directement compilable.

MD5 comprend 64 blocs de ce type, groupés en quatre tours de 16 opérations. F est une fonction non-linéaire, qui varie selon le tour. M_i symbolise un bloc de 32 bits provenant du message à hacher et K_i est une constante de 32 bits, différentes pour chaque opération.



L'algorithme MD5 (source Wikipédia)

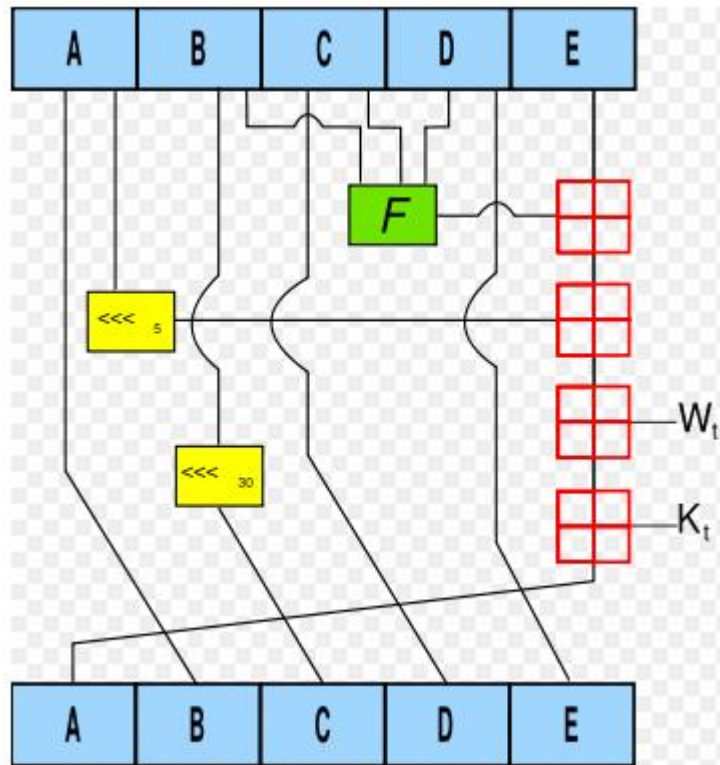
Au jour d'aujourd'hui, cet algorithme n'est plus considéré comme fiable et devrait être abandonné suites aux différentes attaques menées avec succès depuis 2004.

6.4 - L'algorithme SHA-1

Le SHA-0 est l'ancêtre des algorithmes SHA-1 et successeurs. Il a été inventé en 1993 et il a été rapidement abandonné pour des raisons de sécurité insuffisante. Il était soupçonné de contenir des faiblesses qui permettraient d'aboutir rapidement à des collisions. Le mot SHA signifie **Secure Hash Algorithm**.

Le SHA-0 a été modifié pour devenir le SHA-1 en 1995. Le résultat généré est un nombre sur 160 bits (20 octets) à partir de messages de 512 bits (64 octets).

Pour chaque bloc l'algorithme calcule 80 tours successifs et applique une série de transformations sur l'entrée. La première étape consiste à calculer 80 valeurs sur 32 bits. Les 16 premières valeurs sont obtenues directement à partir du bloc en entrée. Les 64 autres sont calculées successivement. Le SHA-1 les obtient grâce à une rotation (absente dans SHA-0) qui est appliquée sur le résultat d'un XOR. Il utilise pour cela 4 mots obtenus dans les itérations précédentes. On définit ensuite 5 variables qui sont initialisées avec des constantes (spécifiées par le standard). Le SHA-1 utilise encore 4 autres constantes dans ses calculs. Si un bloc de 512 bits a déjà été calculé auparavant, les variables sont initialisées avec les valeurs obtenues à la fin du calcul sur le bloc précédent.



Une itération de SHA-1 (source Wikipédia)

D'autres versions de cet algorithme (SHA-256, SHA-384 et SHA-512) existent aussi. Elles offrent respectivement des empreintes numériques de 256, 384 et 512 bits (soient 32, 48 et 64 octets).

6.5 - D'autres algorithmes

Il existe encore d'autres algorithmes de calcul d'empreintes parmi lesquels, on peut citer :

- Boognish
- Construction de Davies-Meyer
- Construction de Matyas-Meyer-Oseas
- Construction de Merkle-Damgård
- Construction de Miyaguchi-Preneel
- FFT-hash
- HAS-160
- Haval
- LM hash
- N-hash
- PANAMA
- RIPEMD, RIPEMD-128, RIPEMD-160, RIPEMD-256

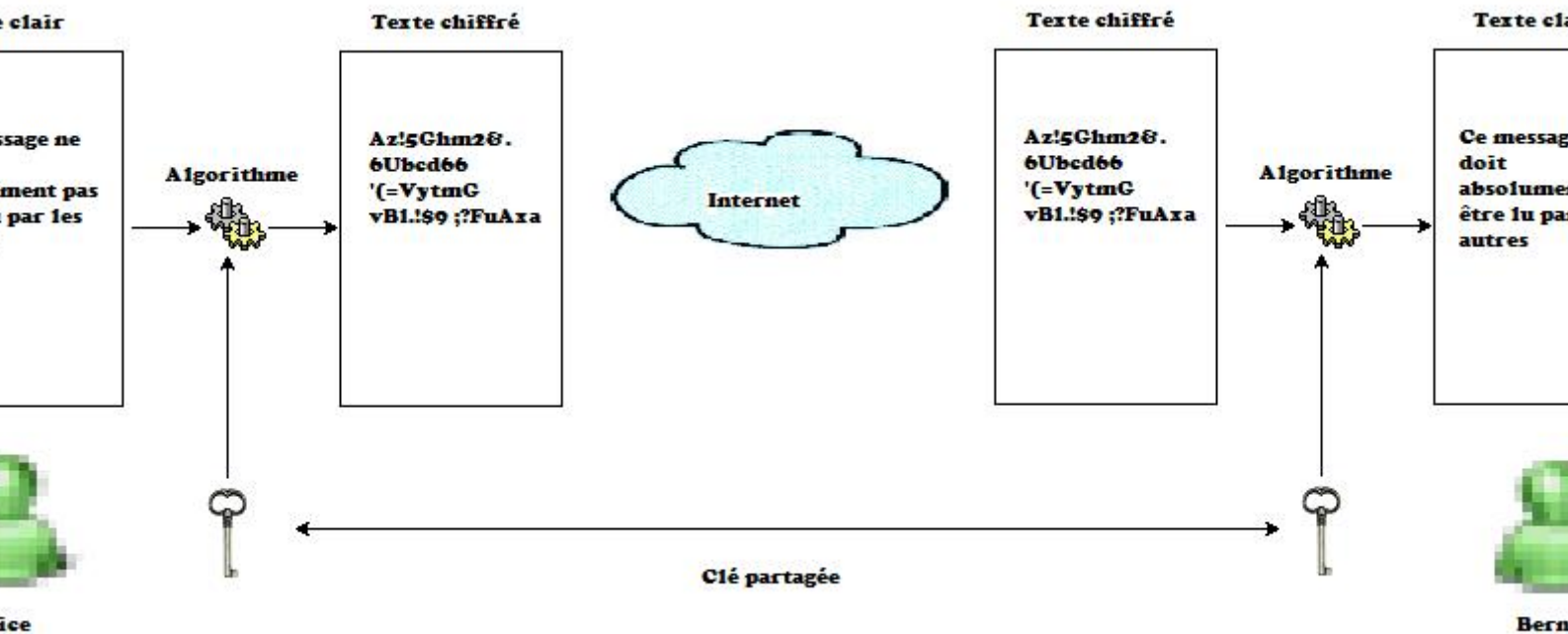
- Snefru
- StepRightUp
- Tiger
- VSH
- Whirlpool

7 - Les algorithmes symétriques

Un algorithme symétrique est un algorithme qui permet de transformer un **texte en clair** en **texte chiffré** en utilisant une clé et de retransformer le **texte chiffré** en **texte en clair** en utilisant la même clé.

Le secret de la communication est uniquement assuré par la clé qui est utilisée lors de la phase de **chiffrement** et de **déchiffrement**. L'algorithme utilisé ne fait pas partie du secret.

On parle d'algorithmes symétriques car c'est la même clé qui sert à la fois au chiffrement et au déchiffrement du message.



Principe de l'algorithme symétrique

Dans ce paragraphe nous allons plus particulièrement nous intéresser aux algorithmes DES, AES et IDEA.

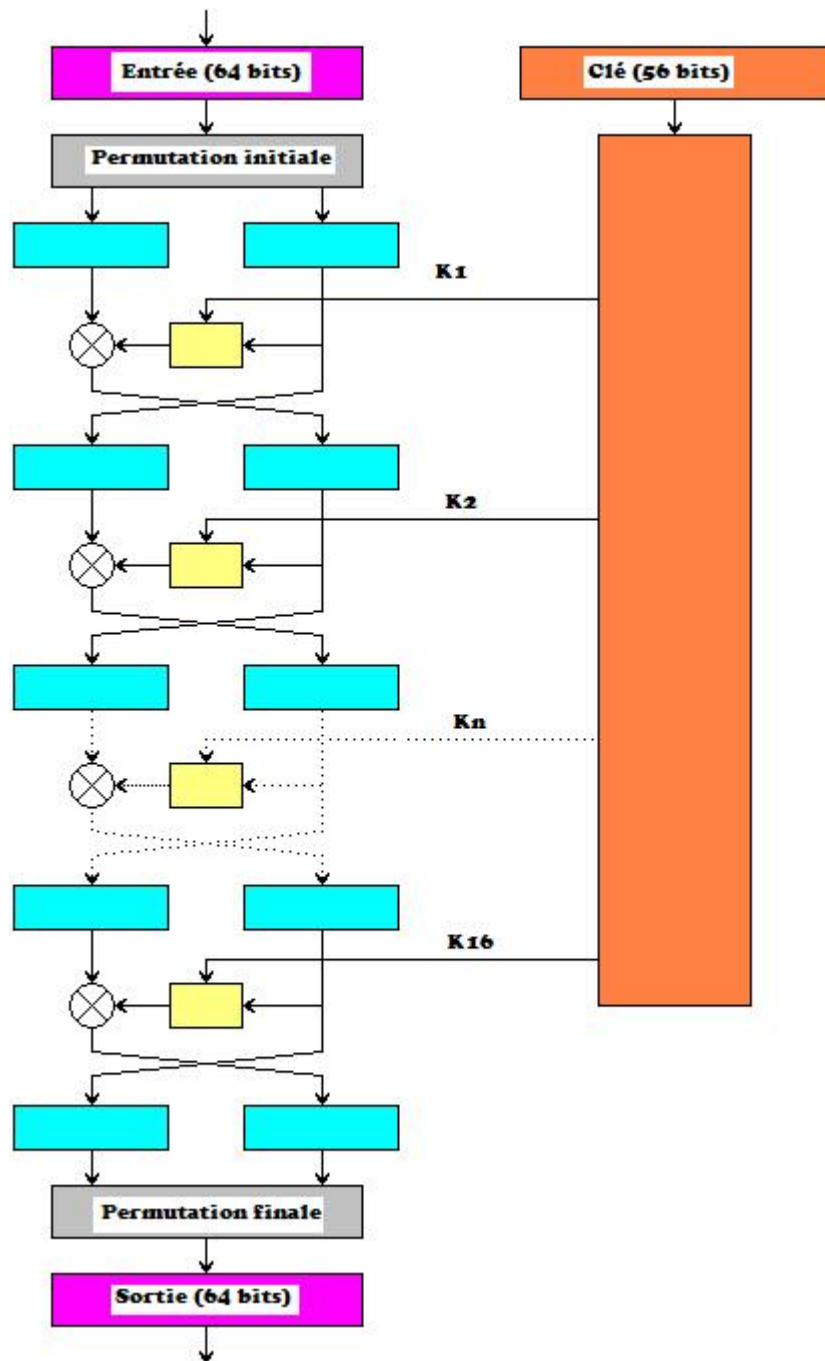
7.1 - L'algorithme DES

L'algorithme DES (Data Encryption Standard) est né dans les années 1970 sous l'impulsion du National Bureau of Standards américain. Celui-ci voulait promouvoir un système de chiffrement qui permettrait à différents interlocuteurs de communiquer en toute sécurité tout en utilisant un algorithme fiable et commun.

L'algorithme **Lucifer** est l'algorithme, conçu par IBM, qui fut retenu. La NSA (National Security Agency) imposa toutefois quelques modifications dans l'algorithme. Ces modifications imposées furent les causes de beaucoup de rumeurs.

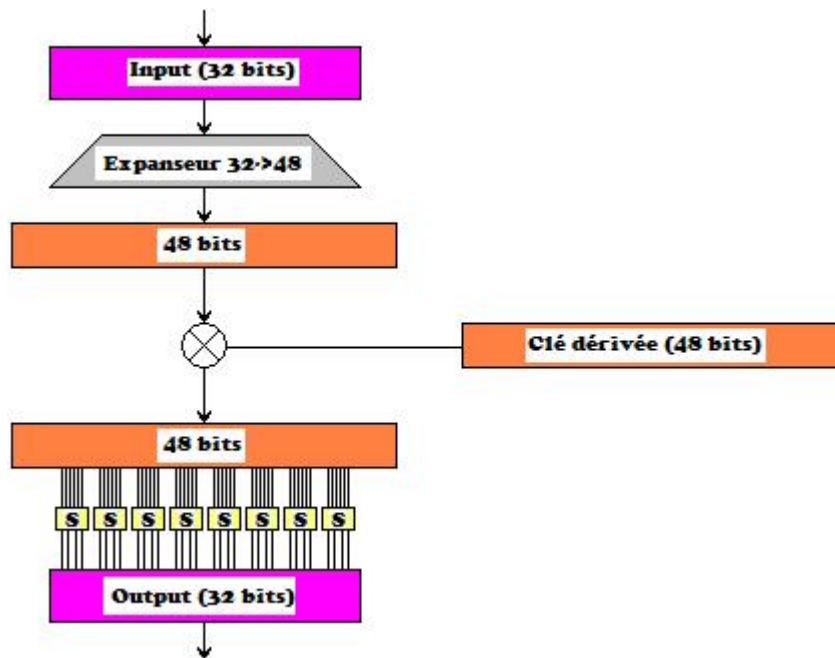
L'algorithme DES fut officiellement adopté le 23 novembre 1976, il est décrit dans le document FIPS PUB 46 modifié depuis par le **FIPS PUB 46-3**.

L'algorithme DES est un algorithme symétrique de chiffrement par bloc de 64 bits (soient 8 octets) fonctionnant avec une clé de 56 bits. Il fonctionne sur 16 rondes et lors de chacune des rondes, le bloc de 64 bits est découpé en 2 blocs de 32 bits.



Vue général du DES

Le bloc de droite qui subit les transformations à l'aide d'une des clés de 48 bits dérivées de la clé principale de 56 bits.



Opération sur le bloc de 32 bits de droite

Les 32 bits de droite sont d'abord **étendus** à 48 bits à l'aide d'une opération particulière. Ensuite, une opération de **ou exclusif** est effectuée entre ces 48 bits et les 48 bits d'une des clés dérivées de la clé principale. Et enfin, les 48 bits du résultat sont ensuite **condensés** sur 32 bits à l'aide des boîtes S1 à S8. Ces huit boîtes ont été imposées par la NSA et ont été la cause de toutes les rumeurs concernant cet algorithme. Chacune de ces boîtes transforme 6 bits en 4 bits selon un schéma encore une fois très particulier.

Comme on peut le voir, cet algorithme n'est pas très facile à transformer en programme compte tenu de toutes les permutations et compressions de bits. Par contre, c'est un algorithme très facile à implémenter en électronique. Il ne faut pas oublier qu'en 1970 on faisait plus de choses plus vite en électronique qu'en informatique.

Beaucoup d'applications encore de nos jours utilisent le DES. C'est par exemple l'algorithme qui a été utilisé pendant très longtemps (et encore maintenant pour certains) sur les ordinateurs de type Unix pour chiffrer les mots de passe des utilisateurs.

Cet algorithme ne devrait maintenant plus être utilisé du fait de son petit nombre de clés (2^{56}) qui paraissait énorme en 1970 mais qui est maintenant petit comparé à d'autres algorithmes et à la puissance de calcul disponible (le calcul distribué permet maintenant de trouver une clé en 24 heures). De plus, de nombreuses attaques cryptographiques ont permis de découvrir des petites faiblesses dans le DES.

Il faut noter aussi que le DES possède 4 clés faibles et 12 clé semi-faibles. La probabilité (relativement faible) de

tomber sur une de ces clés est de $\frac{4}{2^{56}}$.

7.2 - L'algorithme 3-DES

L'algorithme 3-DES utilise l'algorithme DES avec 2 ou 3 clé différentes. L'algorithme va d'abord chiffrer avec une clé, déchiffrer avec la deuxième clé est enfin chiffrer encore avec la troisième clé.

Que l'on utilise cet algorithme avec 2 clés différentes (112 bits) ou 3 clés différentes (168 bits), la clé effective est de 112 bits.

Pour les même raisons, le double DES n'existe pas car la clé effective n'est que de 57 bits soit 1 bit de plus que le simple DES.

7.3 - L'algorithmme AES

De même qu'en 1970 pour le DES avec le National Bureau of Standards américain, l'algorithmme AES est issu d'un appel d'offre lancé par le NIST américain (National Institute of Standards and Technology) en 1997.

C'est l'algorithmme Rijndael dont les deux concepteurs sont Joan Daemen et Vincent Rijmen qui est choisi comme nouveau standard pour remplacer le DES vieillissant.

L'algorithmme AES est un algorithmme symétrique de chiffrement par bloc de 128 bits (soient 16 octets) fonctionnant avec une clé de 128, 192 ou 256 bits.

7.4 - L'algorithmme IDEA

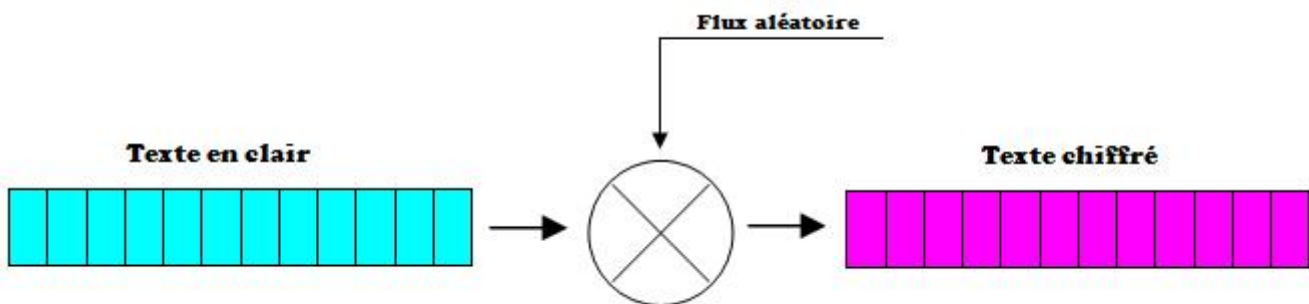
IDEA pour International Data Encryption Algorithm est un algorithmme de chiffrement conçu par Xuejia Lai et James Massey et présenté pour la première fois en 1991.

L'algorithmme IDEA est un algorithmme symétrique de chiffrement par bloc de 64 bits (soient 8 octets) fonctionnant avec une clé de 128 bits.

La particularité de cet algorithmme est que les opérations qu'il utilise en interne s'adaptent très facilement à une programmation informatique sur machine de 16 bits.

7.5 - L'algorithmme du masque jetable

L'algorithmme du masque jetable consiste à utiliser un flux de nombres aléatoires qui va être combiné par une opération **ou exclusif** sur le **texte en clair** pour générer le **texte chiffré**.



Algorithmme du masque jetable

Les contraintes imposées par ce système sont :

- Le flux aléatoire doit être vraiment aléatoire (voir le chapitre au sujet de la génération de nombres aléatoires).
- Le destinataire du message doit recevoir par un moyen fiable une copie du flux aléatoire utilisé pour chiffrer le **texte en clair** (CD-ROM convoyé par porteur spécial, etc.).
- Un flux aléatoire déjà utilisé ne doit jamais être réutilisé.

Si ces contraintes sont parfaitement respectées, le **texte chiffré** est indéchiffrable et inattaquable pour quiconque ne possède pas une copie du flux aléatoire.

7.6 - D'autres algorithmmes

Il existe encore d'autres algorithmmes symétriques parmi lesquels on peut citer :

- Blowfish
- Fox
- Frog
- Misty1
- MMB
- 3-Way
- RC2, RC4, RC5
- REDOC
- SHACAL

8 - Les algorithmes asymétriques

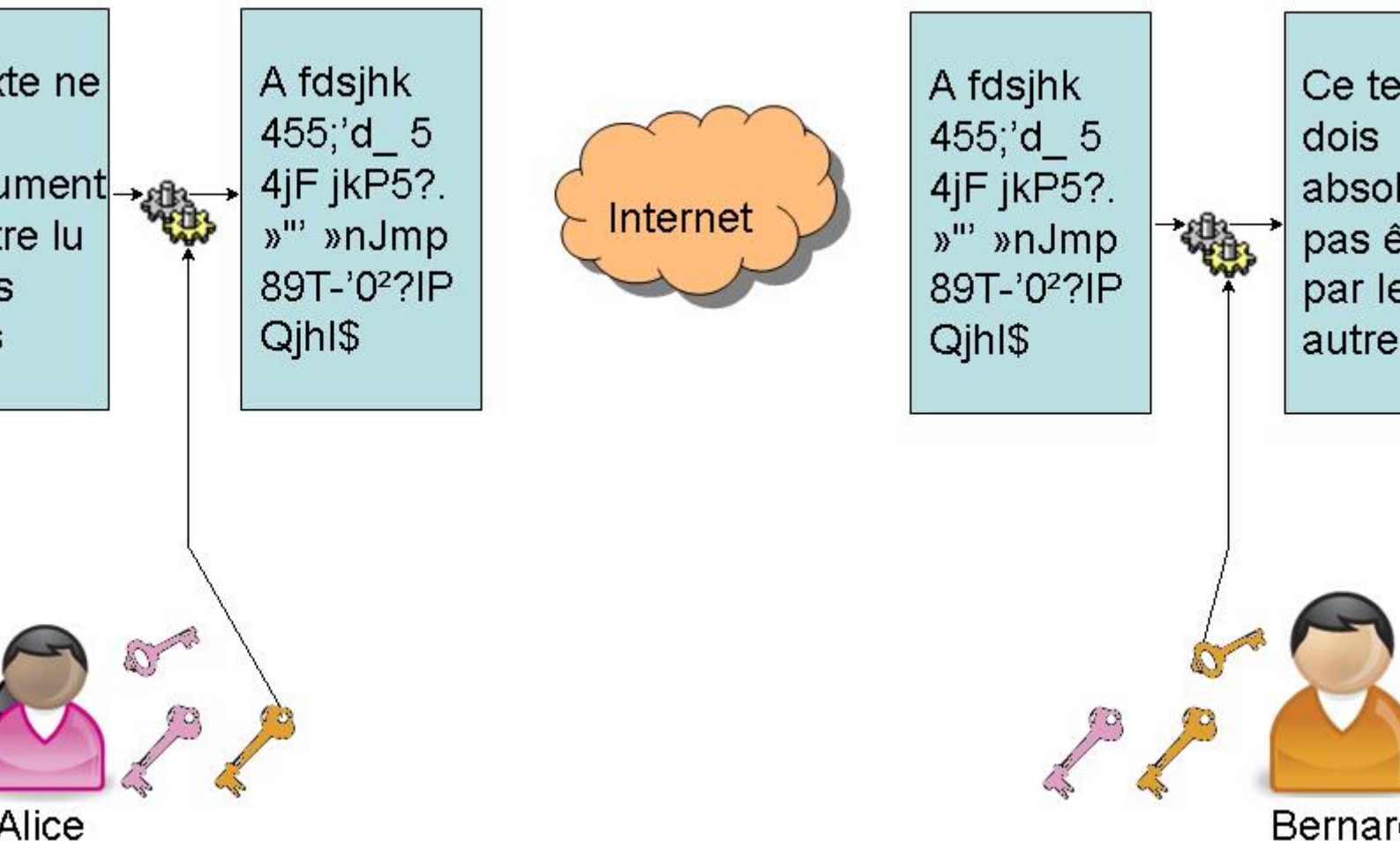
Les algorithmes symétriques vus dans le chapitre précédent sont tous fiables mais ils posent un problème, c'est celui de l'échange de la clé : **comment transmettre de manière fiable à mon interlocuteur la clé de chiffrement utilisée pour chiffrer le message que je lui envoie ?** Il y a bien sûr le téléphone, mais il y a aussi les écoutes téléphoniques.

Les algorithmes asymétriques ont été inventés pour pallier précisément le problème de transmission sécurisée de la clé.

On parle d'algorithmes asymétriques car ce n'est pas la même clé qui sert au chiffrement et au déchiffrement. Dans le cas de ces algorithmes, on parlera alors de **clé privée** et de **clé publique**. Ces deux clés, clé privée et clé publique, sont intimement liées par une fonction mathématique complexe.

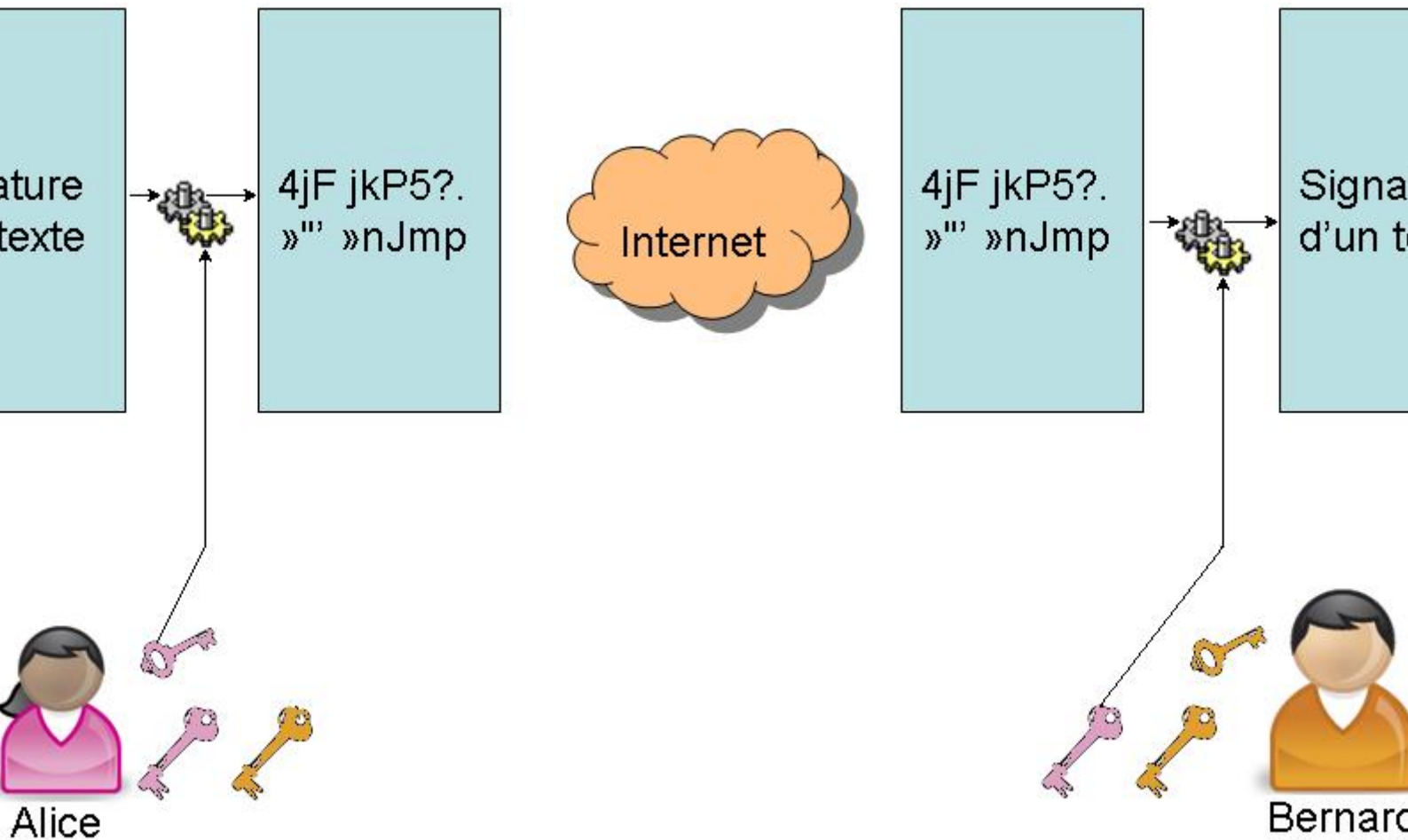
Les algorithmes asymétriques possèdent 2 modes de fonctionnement ;

- Le mode **chiffrement** dans lequel l'émetteur chiffre un fichier avec la clé publique du destinataire pour chiffrer. Le destinataire utilise sa clé privée pour déchiffrer le fichier. Dans ce mode, l'émetteur est sûr que seul le destinataire peut déchiffrer le fichier.



Chiffrement avec l'algorithme asymétrique

- Le mode **signature** dans lequel l'émetteur signe un fichier avec sa propre clé privée. Le destinataire utilise la clé publique de l'émetteur pour vérifier la signature du fichier. Dans ce mode, le destinataire est sûr que c'est bien l'émetteur qui a envoyé le fichier.



Signature avec l'algorithme asymétrique

Donc pour résumer :

- L'émetteur chiffre avec la clé publique du destinataire, le destinataire déchiffre avec sa clé privée.
- L'émetteur signe avec sa clé privée, le destinataire vérifie la signature avec la clé publique de l'émetteur.

8.1 - L'algorithme DH

Whitfield Diffie et Martin Hellman ont travaillé ensemble en 1974 sur ce problème. Comment deux personnes sur Internet peuvent-elles communiquer de manière sécurisée et donc pour cela s'échanger de manière fiable et sécurisée une clé de chiffrement permettant l'usage d'un des protocoles symétriques.

Ils ont tourné leurs recherches vers les fonctions non réversibles, c'est-à-dire les fonctions de la forme $y = f(x)$ où quand on connaît x , il est très facile de calculer y mais quand on connaît y , il est impossible de recalculer x .

Par exemple, la fonction $y = x * x$ est une fonction réversible. Connaissant x , il est facile de calculer y et connaissant y , il est facile de calculer x . La fonction réversible de $y = x * x$ est $x = \text{racine}(y)$.

Les fonctions qu'ils ont utilisées sont les fonctions de l'arithmétique modulo car elles ne sont pas réversibles.

Par exemple $y = x \text{ (modulo 7)}$ n'est pas une fonction réversible. En effet, pour $x = 11$, on obtient $y = 4$ mais pour $y = 4$, on obtient $x = 11$ ou $x = 18$ ou encore $x = 25$ et une infinité d'autres solutions encore.

Finalement, en 1976, ils découvrirent une fonction du type $y = Y^x \text{ (modulo P)}$ qui permet cet échange sécurisé.

Un exemple d'un échange sécurisé de clé est schématisé dans le tableau suivant. Alice veut communiquer avec Bernard et Estelle peut écouter les échanges.

Alice	Estelle	Bernard
Nous utiliserons la fonction $y = 11^x \text{ modulo } 13$	Ha ! Ils vont utiliser la fonction $y = 11^x \text{ modulo } 13$	OK, utilisons la fonction $y = 11^x \text{ modulo } 13$
Alice choisit un nombre secret $A = 5$	Estelle ne sait pas les nombres secrets A et B choisis par Alice et Bernard.	Bernard choisit un nombre secret $B = 8$
Alice calcule le nombre $A' = 11^A \text{ modulo } 13$. $A' = 7$ et envoie ce nombre $A' = 7$ à Bernard de manière quelconque	Ha ha ! Le nombre A' d'Alice est 7 et le nombre B' de Bernard est 9	Bernard calcule son nombre $B' = 11^B \text{ modulo } 13$. $B' = 9$ et envoie ce nombre $B' = 9$ de manière quelconque à Alice
Alice prend le nombre $B' = 9$ de Bernard et calcule le résultat de : $B'^A \text{ modulo } 13$ et obtient : 3	Estelle ne sait pas calculer ces nombres car elle ne dispose ni du nombre secret initial d'Alice A , ni du nombre secret initial de Bernard B .	Bernard prend le nombre $A' = 7$ d'Alice et calcule le résultat de : $A'^B \text{ modulo } 13$ et obtient : 3

En final, Alice et Bernard obtiennent un même nombre secret 3 qu'ils peuvent utiliser comme clé pour un algorithme symétrique choisi. Le calcul de cette clé s'est effectué en s'échangeant 2 valeurs presque quelconques. Estelle, qui pourtant a la possibilité d'écouter les échanges, ne sait rien de cette clé et n'a pas la possibilité de la calculer.

Bien sûr, dans la pratique, les nombres choisis seront plus grands mais le principe reste le même.

Avec cet algorithme, Whitfield Diffie et Martin Hellman ont prouvé que l'échange des clés de manière fiable et sécurisée était possible sans avoir besoin de se rencontrer.

Cet algorithme, bien que simple souffre d'un défaut. Il faut que Alice et Bernard se synchronisent afin de pouvoir s'échanger un minimum d'information : les paramètres de la fonction utilisée ainsi que les valeurs A' et B' . Cela signifie qu'Alice doit attendre que Bernard soit réveillé pour lui envoyer un message.

L'échange sécurisé des clés étant réglé, il restait le problème de synchronisation obligatoire à supprimer.

8.2 - L'algorithme RSA

Suites aux travaux réalisés par Whitfield Diffie et Martin Hellman, Ronald Rivest, Adi Shamir et Leonard Adleman ont eux aussi travaillé tous les trois sur les algorithmes asymétriques à partir de 1975.

L'algorithme qu'ils ont mis au point repose aussi sur une fonction à sens unique ou plutôt une fonction très difficilement réversible. Cette fonction difficilement réversible est la factorisation d'un nombre en produit de facteurs premiers. En effet, autant, il est très facile de choisir deux nombres (48 et 52 par exemple) et de calculer leur produit (2496), autant il est long et pénible d'extraire les facteurs premiers du nombre 2397 qui sont 47 et 51.

Le principe de l'algorithme RSA est donc basé sur la difficulté de factoriser un nombre surtout quand ce nombre est très grand et qu'il est le produit de deux nombres premiers très grands eux aussi.

L'algorithme imaginé est le suivant :

- Alice choisit au hasard 2 nombre premiers p et q (13 et 17 par exemple).
- Elle construit le produit de ces deux nombres N (221 dans notre exemple).
- Elle choisit un nombre e . Ce nombre ne doit pas être un diviseur de $(p-1) * (q-1)$ (5 par exemple).

- La clé publique d'Alice est composée de la paire de nombre **e** et **N** (5 et 221 dans notre exemple). Cette clé publique peut être publiée dans un annuaire.
- Alice calcule ensuite le nombre **d** donné par la formule **e * d = 1 modulo ((p-1) * (q-1))** (77 dans notre exemple). Ce nombre représente la clé privée d'Alice.
- Bernard veut envoyer le message **bonjour** à Alice. Pour cela, il convertit le mot en valeurs (nous prendrons les valeurs issues du code ASCII dans notre exemple). La conversion donne les valeurs **98, 111, 110, 106, 111, 117, 114**.
- La formule utilisée pour chiffrer les valeurs est la suivante :

$$c_i = m_i^e \pmod{N}$$

dans lequel **ci** représente la valeur chiffrée de **mi**, **e** et **N** représentant les paramètres publics de la clé d'Alice.

- Avec le **texte en clair 98, 111, 110, 106, 111, 117, 114**, et en appliquant la formule pour chacune des valeurs on obtient le **texte chiffré** suivant : **115, 76, 145, 123, 76, 104, 173**.
- Pour déchiffrer le message, Alice utilise la formule suivante :

$$m_i = c_i^d \pmod{N}$$

dans lequel **mi** représente la valeur en clair de **ci**, **d** et **N** représentant les paramètres publics et privés de la clé d'Alice.

- Avec le **texte chiffré 115, 76, 145, 123, 76, 104, 173**, et en appliquant la formule pour chacune des valeurs on obtient à nouveau le **texte en clair** d'origine : **98, 111, 110, 106, 111, 117, 114**.

Cet exemple montre que les paramètres publics **e** et **N** suffisent à chiffrer un message et que les paramètres privés et publics **d** et **N** suffisent pour déchiffrer le message. Bien sûr, dans la pratique, les nombres choisis seront plus grands mais le principe reste le même.

Alice peut donc diffuser librement sa clé publique (les paramètres **e** et **N**). Toute personne qui veut communiquer avec Alice peut récupérer la clé publique d'Alice et lui envoyer un message chiffré. Alice est la seule à pouvoir déchiffrer ce message puisqu'elle est la seule à détenir le paramètre **d** de sa clé privée.

De plus Alice et Bernard n'ont plus besoin de se synchroniser pour commencer l'échange puisque la clé publique est disponible dans un annuaire public et que n'importe qui peut aller la chercher à tout moment pour l'utiliser.

La force de l'algorithme repose sur la difficulté à factoriser **N**. Si la factorisation était facile à réaliser, Estelle pourrait recalculer les nombres **p** et **q** et, connaissant **e**, elle pourrait alors calculer **d** et donc elle pourrait déchiffrer le message destiné à Alice.

La taille des clés RSA devraient aujourd'hui être au minimum de 1024 bits (309 chiffres décimaux). En effet, en mai 2005, le nombre RSA-200, un nombre de 663 bits (200 chiffres décimaux) a été factorisé. Ce nombre RSA-200 est le suivant :

```
RSA-200 =
27997833911221327870829467638722601621070446786955428537560009929326128400107609
34567105295536085606182235191095136578863710595448200657677509858055761357909873
4950144178863178946295187237869221823983
=
35324619344027701212726049781984643686711974001976250236493034687761212536794232
00058547956528088349
*
79258699544783330333470858414800596877379758573642199607343303414557678728181521
35381409304740185467
```

Une autre utilisation de l'algorithme RSA concerne la signature. En effet, si Alice chiffre un document avec sa clé privée, tout le monde peut obtenir la clé publique d'Alice disponible dans un annuaire et déchiffrer ce texte. Le fait de pouvoir déchiffrer le texte avec la clé publique d'Alice prouve que c'est bien Alice qui l'a chiffré puisqu'elle est la seule à posséder la clé privée.

Pour résumer :

- L'émetteur **chiffre** un message avec la **clé publique** du destinataire de façon à ce que seul le destinataire puisse déchiffrer le message avec **sa clé privée**.
- L'émetteur **signe** un message avec **sa clé privée** de façon à ce que le destinataire puisse vérifier la signature du message avec la **clé publique de l'émetteur**.

8.3 - D'autres algorithmes

Il existe encore d'autres algorithmes asymétriques parmi lesquels, on peut citer :

- Cryptographie sur les courbes elliptiques
- Cryptographie sur les courbes hyper elliptiques
- Crypto système de Blum-Goldwasser
- Crypto système de ElGamal
- Crypto système de Goldwasser-Micali
- Algorithme de Guillou-Quisquater
- Crypto système de McEliece
- Crypto système de Paillier
- Crypto système de Rabin
- Algorithme LLL

9 - Les méthodes de génération de nombres aléatoires

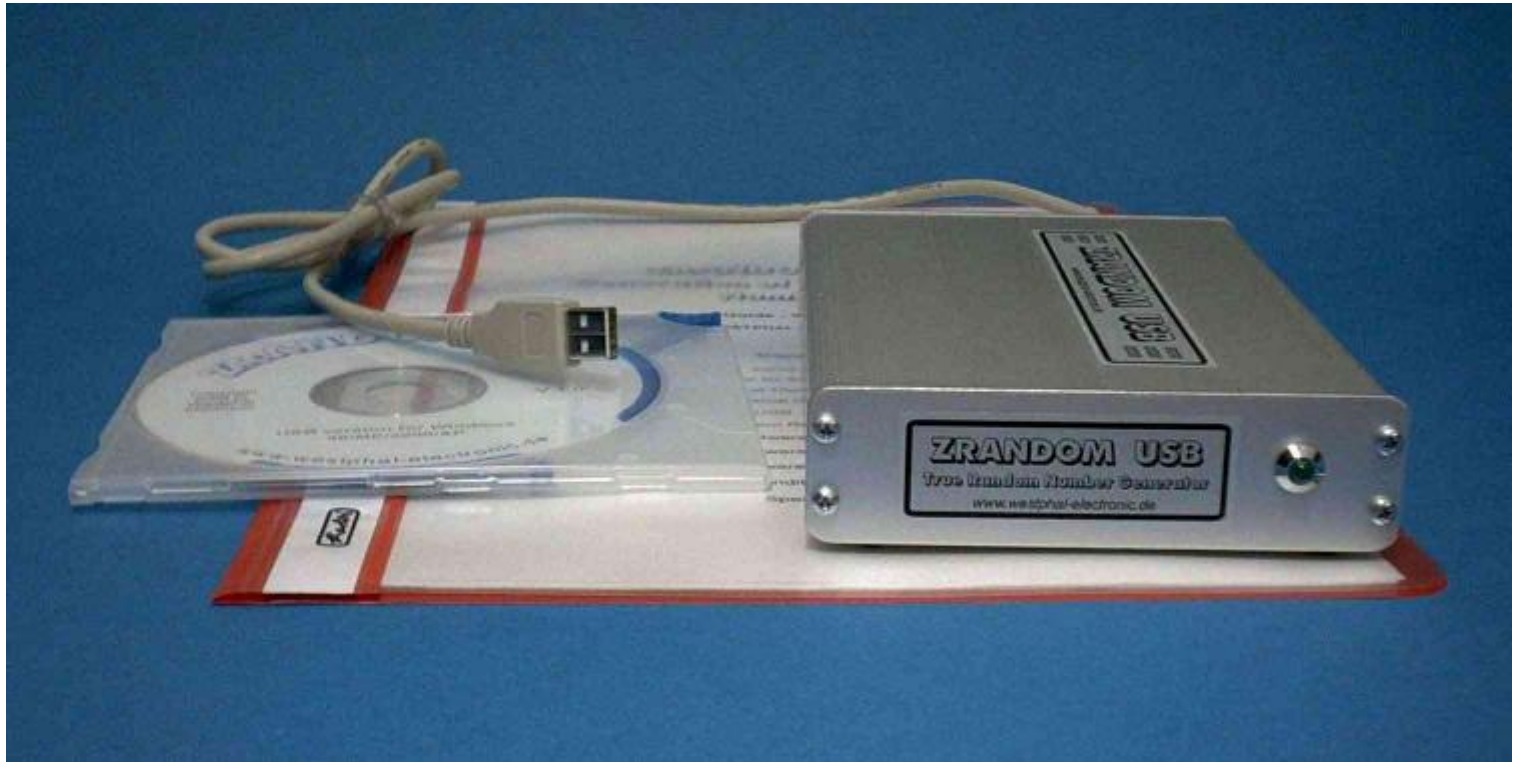
On l'a vu dans les chapitres précédents, les algorithmes cryptographiques demandent souvent des nombres aléatoires. Ces nombres aléatoires peuvent être utilisés pour générer une clé aléatoire ou encore pour générer une clé publique et une clé privée pour l'algorithme RSA. Dans le cas de l'algorithme RSA, il faut en plus que ces nombres aléatoires soient premiers mais il existe des algorithmes rapides pour déterminer la primalité d'un nombre.

Mais d'abord, quelles sont les caractéristiques d'une suite de nombres aléatoire ? Les mathématiciens aujourd'hui ne savent pas bien définir la notion d'aléatoire. Toutefois, on retient les caractéristiques suivantes :

- La répartition des nombres de la suite est équilibrée, c'est-à-dire que statistiquement, il y a autant de 0 que de 1.
- La suite de nombres ne possède pas de séquences plus ou moins longues de nombres qui se répètent.

Il existe plusieurs méthodes pour générer des nombres aléatoires :

- La méthode des dés ou le tirage à pile ou face. On lance un dé ou une pièce et on relève la valeur sortie. Cette méthode ne génère pas des nombres aléatoires mais plutôt des nombres imprévisibles. En effet, même si l'on connaît les conditions initiales (force du lancé, poids du dé, hauteur de la table, etc.), il est impossible de connaître le résultat car cela relève de la théorie du chaos. Les phénomènes chaotiques ne produisent pas des nombres aléatoires, ils produisent des nombres chaotiques.
- Les générateurs de nombres pseudo aléatoires. Il s'agit d'algorithmes qui génèrent des séquences de nombres qui ressemblent à des nombres aléatoires. Les nombres sont approximativement indépendants les uns des autres mais il est difficile de trouver cette relation. On utilise ces nombres pseudo aléatoires car il est d'une part difficile d'obtenir des vrais nombres aléatoires et d'autre part, ces algorithmes se prêtent bien à une réalisation informatique. Une remarque célèbre de John Von Neumann : **Quiconque considère des méthodes arithmétiques pour produire des nombres aléatoires est, bien sûr, en train de commettre un péché.** Des nombres pseudo aléatoires ne sont pas des nombres aléatoires mais ils suffisent la plupart du temps.
- Les générateurs physiques de nombres aléatoires. Ces générateurs reposent sur le caractère quantique et donc aléatoire de certains phénomènes physiques comme le bruit thermique d'une résistance ou alors la radio activité ambiante.



Générateur aléatoire électronique

Il est possible de trouver des générateurs de nombre aléatoires reposant sur un phénomène physique chez **Westphal Electronic**. Celui présenté ci-dessus coûte 595 €.

Dans ce paragraphe nous étudierons exclusivement les générateurs de nombres pseudo aléatoires.

Les principaux générateurs de nombres aléatoires peuvent être du type :

- Des générateurs linéaires congruents
- Des générateurs à base de registre à décalage
- D'autres algorithmes spécifiques

9.1 - Les générateurs linéaires congruents

Un générateur linéaire congruent est de la forme : $X_n = (a X_{n-1} + b) \bmod m$ dans lequel **a**, **b** et **m** sont les paramètres du générateur. Ce générateur a une période qui ne peut être plus grande que **m** et si **a**, **b** et **m** sont choisis correctement (par exemple **b** et **m** premiers entres eux), alors la période de répétition sera maximale et tous les nombres de la période seront utilisés.

Le générateur le plus connu est celui de la fonction rand() sur les machines IBM et d'autres machines 32 bits :

$X_{n+1} = (65539 \times X_n) \bmod 2^{31}$. Mais ce générateur souffre d'un grave défaut, les nombres générés sont prévisibles car il y a un lien entre X_n , X_{n+1} et X_{n+2} . Les 3 valeurs consécutives sont liées par la formule :

$$X_{n+2} = (6 \times X_{n+1} - 9 \times X_n) \bmod 2^{31}$$

Le tableau suivant présente quelques valeurs possibles pour les paramètres **a**, **b** et **m** :

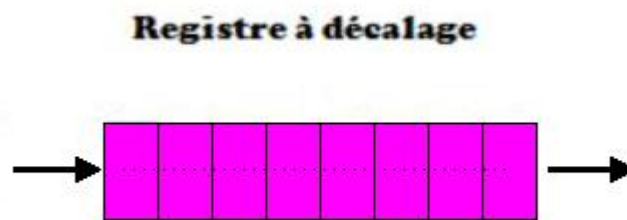
a	b	n
106	1 283	6 075
171	11 213	53 125
421	17 117	81 000
741	66 037	312 500
3 613	45 289	214 326
9 301	49 297	233 280

Ce type de générateur convient très bien pour des jeux mais certainement pas en cryptographie car l'espace des nombres obtenus est bien trop petit et en plus, les nombres obtenus sont trop prévisibles :

- Quand on connaît deux nombres, il est possible de retrouver les paramètres du générateur.
- Quand on connaît les paramètres de ce générateur et un nombre, le nombre suivant se déduit très facilement.

9.2 - Les générateurs à base de registre à décalage

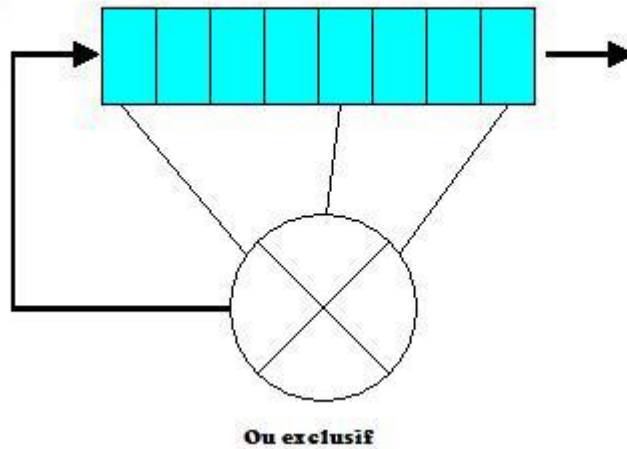
La notion de registre à décalage est issue de l'électronique. Dans ce composant, le registre contient un certain nombre de bits et à chaque impulsion d'horloge, tous les bits sont décalés d'une case vers la droite. Le bit le plus à droite est donc extrait du registre à décalage et un nouveau bit est inséré dans la partie gauche du registre.



Registre à décalage 8 bits

La forme la plus simple de ce type de générateur est le registre à décalage à rétroaction linéaire encore appelé RDRL. La fonction de rétroaction est composée d'un **ou exclusif** qui n'utilise que quelques bits du registre à décalage.

Registre à décalage



RDLR à 8 bits

Le choix des bits à utiliser dans la fonction de rétro action ne se fait pas au hasard. Pour que la fréquence soit maximale et que le générateur passe par tous les états du registre à décalage, il faut que le polynôme de rétroaction ($x^7 + x^3 + x^0$ dans l'exemple ci-dessus) soit un polynôme primitif modulo 2 (un polynôme est dit **primitif** si le pgcd de tous ses coefficients est 1).

La liste suivante présente quelques polynômes primitifs :

- $x^1 + x^0$
- $x^3 + x^1 + x^0$
- $x^7 + x^1 + x^0$
- $x^7 + x^3 + x^0$
- $x^{12} + x^6 + x^4 + x^0$
- $x^{14} + x^5 + x^3 + x^1 + x^0$

L'exemple présenté ci-dessus génère un flux de bits. Pour générer un flux d'octets, il suffit de lire 8 bits et de les réassembler dans un octet ou encore de coupler 8 générateurs en parallèle afin de générer les 8 bits simultanément.

9.3 - L'algorithme BBS

L'algorithme Blum Blum Shub (BBS) du nom de ces inventeurs **Lenore Blum**, **Manuel Blum** et **Michael Shub** est un générateur de nombres pseudo aléatoires utilisable en cryptographie. Il fut inventé en 1986 et il repose sur la formule :

$$x_{n+1} = (x_n)^2 \bmod M$$

Formule du BBS

Avec M produit de deux nombres premiers congruent à 3 modulo 4. La sortie de l'algorithme est alors le bit de poids faible de chaque itération.

Cet algorithme est relativement lent (1 bit à chaque itération seulement) mais il est montré comme **cryptographiquement sûr**.

9.4 - D'autres algorithmes

Il existe encore d'autres algorithmes de génération de nombres pseudo aléatoires parmi lesquels, on peut citer :

- Yarrow
- Fortuna
- ISSAC

10 - Quelques applications de la cryptographie

Après ce survol des principaux algorithmes utilisés en cryptographie, nous allons voir dans ce paragraphe leurs utilisations pratiques.

Après les deux paragraphes sur les algorithmes symétriques et asymétriques, on serait tenté de penser que vu la souplesse dans l'utilisation des algorithmes asymétriques avec des clés publiques et privées, on pourrait oublier les algorithmes symétriques pour lesquels l'échange de clés est un réel problème.

Il n'en est rien car les algorithmes asymétriques sont 100 à 1000 fois moins rapide que les algorithmes symétriques. C'est normal car les opérations de manipulation de bits et de **ou exclusif** sur des nombres de 64 bits sont plus rapides que l'élévation à la puissance et la division de nombres de 1000 bits et plus.

En général, les algorithmes applicatifs vont utiliser les deux types d'algorithmes. La première phase utilisera un **algorithme asymétrique** pour échanger une clé (appelée clé de session) et ensuite, on passera en mode **algorithme symétrique** en utilisant cette clé de session négociée afin de pouvoir travailler plus vite.

On voit donc que la résistance de ces algorithmes à la cryptanalyse n'est pas le produit de la résistance des deux algorithmes mais plutôt la valeur du plus faible des algorithmes utilisés. Il suffit que l'un des deux algorithmes soit faible (quelle qu'en soit la raison) pour que la session complète soit faible.

10.1 - PGP

PGP, pour **Pretty Good Privacy**, est un logiciel de chiffrement inventé en 1991 par Philip Zimmermann. Le but de ce logiciel est essentiellement de permettre l'échange de messages par messagerie électronique de manière totalement sécurisée. Ce logiciel aura causé beaucoup de soucis Philip Zimmermann pour 2 raisons :

- Un premier problème juridique avec la société **RSA Data Security Inc** car le logiciel utilise la cryptographie RSA sans licence d'utilisation.
- Un second problème concernait le fait que la cryptographie et donc la sécurité des échanges de son logiciel était trop forte par rapport aux lois d'utilisation et d'exportations américaines.

Finalement en 1996 le problème s'est résolu par un accord négocié entre **RSA Data Security Inc** Philip Zimmermann et par l'abandon des poursuites du gouvernement américain. Entre temps, le logiciel et ses sources avaient été diffusés sur Internet.

Le logiciel PGP est un algorithme hybride dans le sens où il utilise les deux techniques de cryptographie, la cryptographie symétrique et la cryptographie asymétrique.

Le logiciel PGP propose les fonctionnalités suivantes :

- Chaque utilisateur de PGP possède une bi-clé RSA asymétrique. Cette bi-clé est composée d'une clé publique et d'une clé privée.
- Le logiciel PGP permet de stocker la bi-clé avec une sécurité supplémentaire pour l'archivage de la clé privée, elle est protégée par une phrase que l'utilisateur doit entrer pour la déverrouiller la clé privée.
- Le logiciel permet d'importer les clés publiques des autres usagers avec qui l'utilisateur communique habituellement dans son porte clés.
- Le logiciel permet de chiffrer un message à destination d'un ou plusieurs des usagers dont les clés publiques sont dans le porte clés PGP.
- Le logiciel permet de déchiffrer les messages reçus en utilisant la clé privée de l'utilisateur (après déverrouillage).
- Le logiciel permet de signer un message et de l'envoyer. Ainsi, le destinataire peut être sûr de l'émetteur.
- Le logiciel permet de vérifier la signature d'un message.

Le gros problème de PGP concerne la vérification que la clé publique de Bernard est bien celle de Bernard et pas celle d'Estelle qui voudrait usurper l'identité de Bernard. Ce problème est partiellement et incomplètement résolu par la signature des clés.

Si Bernard fait signer sa clé par Christine et David et qu'Alice connaît Christine ou David, intuitivement elle pourra penser que la clé de Bernard est plus valide que si cette clé n'était signée par personne. Il y a transfert de la confiance.

PGP peut être librement téléchargé. Il existe aussi la version **GnuPG** qui respecte la licence **GNU General Public License**.

10.2 - Notion de Certificat

Avec PGP, Philip Zimmermann a résolu d'une certaine manière le problème de la confiance accordée à une clé publique. Cette confiance est transmise de proche en proche par la signature de la clé par un intermédiaire.

Un problème se pose si Alice et Bernard veulent s'échanger de manière sécurisée des messages et qu'ils n'ont pas d'intermédiaires capables de valider les clés publiques ou si la chaîne des intermédiaires est tellement longue que la confiance que l'on peut accorder à cette chaîne d'intermédiaires accorder est faible.

Ce problème est résolu par la notion de certificat. Alice et Bernard font tous les deux confiances à une autorité de certification et cette autorité valide les clés publiques d'Alice et Bernard. Encore une fois, le problème est de faire confiance à cette autorité de certification mais de toute façon, à un moment ou à un autre, il va falloir faire confiance à quelque chose ou à quelqu'un.

Un certificat peut être vu comme une enveloppe qui accompagne la clé publique. Cette enveloppe est signée par l'autorité de certification. Toute falsification de cette enveloppe pourra être détectée car lorsque le destinataire du certificat vérifiera la signature du certificat, il se rendra compte qu'elle n'est pas valide.

Une norme, la norme X.509, spécifie le contenu d'un certificat. on parlera alors de certificats X.509.

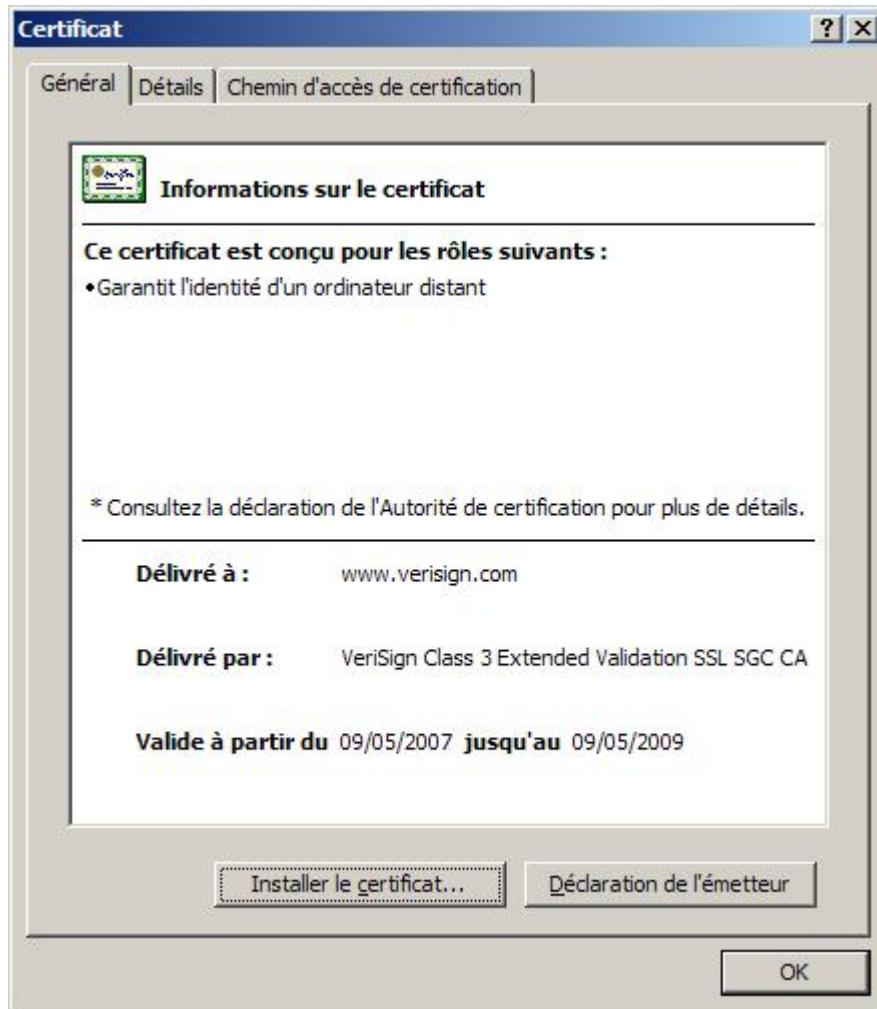
Voyons maintenant quels sont les éléments contenus dans un certificat X.509 :

- La version de la norme utilisée par le certificat. La norme X.509 a évolué et il en existe maintenant 3 versions différentes.
- Un numéro de série. Ce numéro de série est unique pour une autorité de certification donnée.
- Le nom de l'autorité de certification.
- La période de validité de ce certificat. Hors de cette période, un certificat est invalide.
- L'identification du possesseur de la clé publique.
- L'identification de l'algorithme de la clé publique ainsi que la clé publique de la personne.
- Des informations complémentaires optionnelles.
- L'identification de l'algorithme et la valeur de la signature du certificat.

Il existe quatre classes de certificat. Ces quatre classes sont définies en fonction du niveau de sécurité du processus utilisé lors de la génération du certificat.

- Classe 1 : L'adresse mail du demandeur suffit pour créer ce type de certificat.
- Classe 2 : Preuve d'identité requise (carte d'identité, extrait de Kbis pour les entreprises, etc.).
- Classe 3 : C'est un certificat de classe 2 mais la présence physique du demandeur est requise.
- Classe 4 : C'est un certificat de classe 3 mais celui-ci est implanté directement sur un support de type carte à puce.

Le certificat suivant est celui obtenu lorsque l'on se connecte sur le site de **VeriSign** :



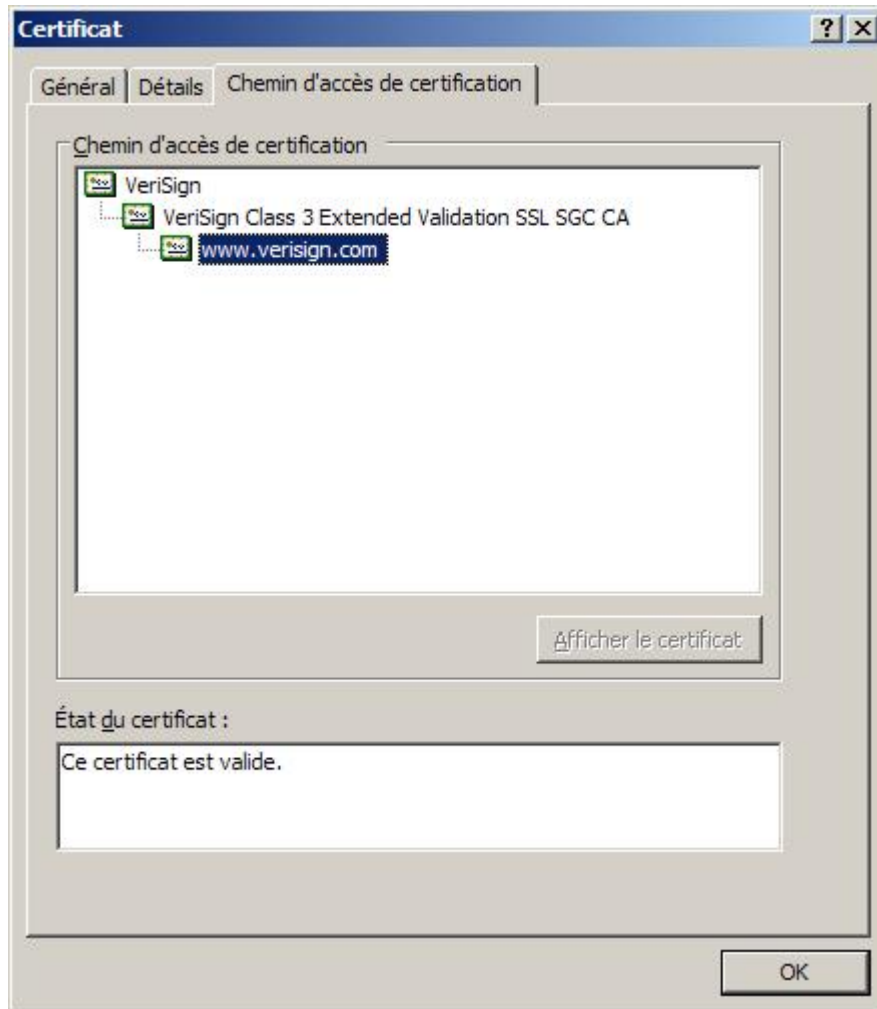
Certificat du site VeriSign

Les informations que l'on retrouve dans ce certificat sont :

- Version X.509 utilisée : 3
- Numéro de série du certificat : 6a 4a c3 1b 31 10 e6 eb 48 f0 fc 51 a3 9a 17 1f
- Algorithme de signature : Sha-1 et RSA
- Identification de l'autorité de certification : VeriSign Class 3 Extended Validation SSL SGC CA
 - CN = VeriSign Class 3 Extended Validation SSL SGC CA
 - OU = Terms of use at <https://www.verisign.com/rpa> (c)06
 - OU = VeriSign Trust Network
 - O = VeriSign, Inc., C=US
- Valide depuis le mercredi 9 mai 2007 01:00:00
- Valide jusqu'au samedi 9 mai 2009 00:59:59 (période de validité : 2 ans)
- Identification du propriétaire de la clé publique : www.verisign.com
 - CN = www.verisign.com
 - OU = Terms of use at www.verisign.com/rpa (c)06
 - OU = Production Security Services
 - O = VeriSign, Inc.
 - STREET = 487 East Middlefield Road
 - L = Mountain View
 - S = California
 - PostalCode = 94043
 - C = US
 - 1.3.6.1.4.1.311.60.2.1.2 = Delaware

- 1.3.6.1.4.1.311.60.2.1.3 = US
- Clé publique : Algorithme RSA, clé publique de 1024 bits, valeur de la clé publique : **30 81 89 02 81 81 00 c7 10 37 e5 eb fc ef d7 6b 81 00 a9 10 d1 91 43 70 11 68 23 f3 fd 69 de 4f d7 6d 31 ed ea 18 db f3 5d 02 6c e9 c8 e6 d4 1c 7c 5d e8 32 5b 58 01 8c 95 f9 66 5e 16 59 ba fd c2 90 16 67 56 32 27 46 c4 c9 28 48 1a 69 14 f1 f3 df 6d c5 d8 c1 2c 96 a1 6c 2f 13 93 51 69 40 76 ce d9 9f 01 57 6f 39 a1 8c 50 ac 67 cb 60 e7 8f 7b 40 75 a4 3e 14 e7 91 0c 92 58 83 a2 b5 87 37 33 ab fa 2d 39 fb 02 03 01 00 01**
- Quelques informations concernant le domaine d'utilisation possible de ce certificat
 - Contrainte de base : Type d'objet=Entité finale, Contrainte de longueur de chemin d'accès=Aucun(e)
 - Identificateur de la clé du sujet : f1 5a 89 93 55 47 4b ba 51 f5 4e e0 cb 16 55 f4 d7 cc 38 67
 - Utilisation de la clé : Signature numérique, Cryptage de la clé (a0)
 - Point de distribution de la liste de révocation des certificats : URL=<http://EVIntl-crl.verisign.com/EVIntl2006.crl>
 - Stratégie de certificat : Identificateur de stratégie=2.16.840.1.113733.1.7.23.6
 - Utilisation avancée de la clé : Authentification du serveur (1.3.6.1.5.5.7.3.1), Authentification du client (1.3.6.1.5.5.7.3.2), Utilisation de clé inconnue (2.16.840.1.113730.4.1), Utilisation de clé inconnue (1.3.6.1.4.1.311.10.3.3)
 - Identification de la clé de l'autorité : 4e 43 c8 1d 76 ef 37 53 7a 4f f2 58 6f 94 f3 38 e2 d5 bd df
 - Accès aux informations de l'autorité : Protocole d'état de certificat en ligne (1.3.6.1.5.5.7.48.1), Autorité de certification émettrice (1.3.6.1.5.5.7.48.2)
 - 1.3.6.1.5.5.7.1.12 : Type d'information inconnue.
- Algorithme d'empreinte numérique : SHA-1
- Empreinte numérique du certificat : c6 75 0e 8d a9 5f 5a 65 bb 04 6d 60 79 d4 fc 87 40 2e 03 81

Il est aussi possible de connaître le chemin de certification du certificat :



Chemin de certification du certificat de VeriSign

Le certificat de **VeriSign** est signé par le certificat **VeriSign Class 3 Public Primary Certification Authority - G5** lui-même signé par le certificat racine de confiance **Verisign**.

10.3 - Notions de PKI

Une PKI (en anglais **Public Key Infrastructure**) ou une IGC (en français **Infrastructure de Gestion des Clés**) est un ensemble de logiciels, machines et procédures permettant de gérer des certificats.

10.3.1 - Les rôles d'une PKI

Les services attendus d'une PKI sont les suivants :

- Création des certificats (enregistrement des utilisateurs ou des équipements informatiques, vérification des données d'identification, cohérence de ces informations).
- Gestion des certificats (publication, renouvellement, révocation).
- Gestion et publication de la liste des certificats révoqués (on parle aussi de listes de révocation).
- Identification et authentification des utilisateurs accédant à la PKI.
- Archivage, séquestre et recouvrement des certificats.

10.3.2 - Les entités d'une PKI

Une PKI définit cinq entités qui sont :

- L'autorité de certification dont le but est de signer les demandes de certificats et la liste des certificats révoqués. Cette autorité est la plus critique.
- L'autorité d'enregistrement dont le but est de générer les demandes de certificats après une éventuelle vérification d'identité du demandeur.
- L'autorité de dépôt dont le but est de publier les certificats ainsi que les listes de révocation.
- L'entité finale qui est l'utilisateur final ou la machine qui va utiliser le certificat.
- L'autorité de séquestre dont le rôle est de stocker de manière sécurisée les clés privées des certificats signés par l'autorité de certification. Ce rôle est optionnel et peut être rendu obligatoire par la législation de certains pays (dont la France) qui imposent de fournir aux autorités un moyen permettant de déchiffrer les données chiffrées par un certificat.

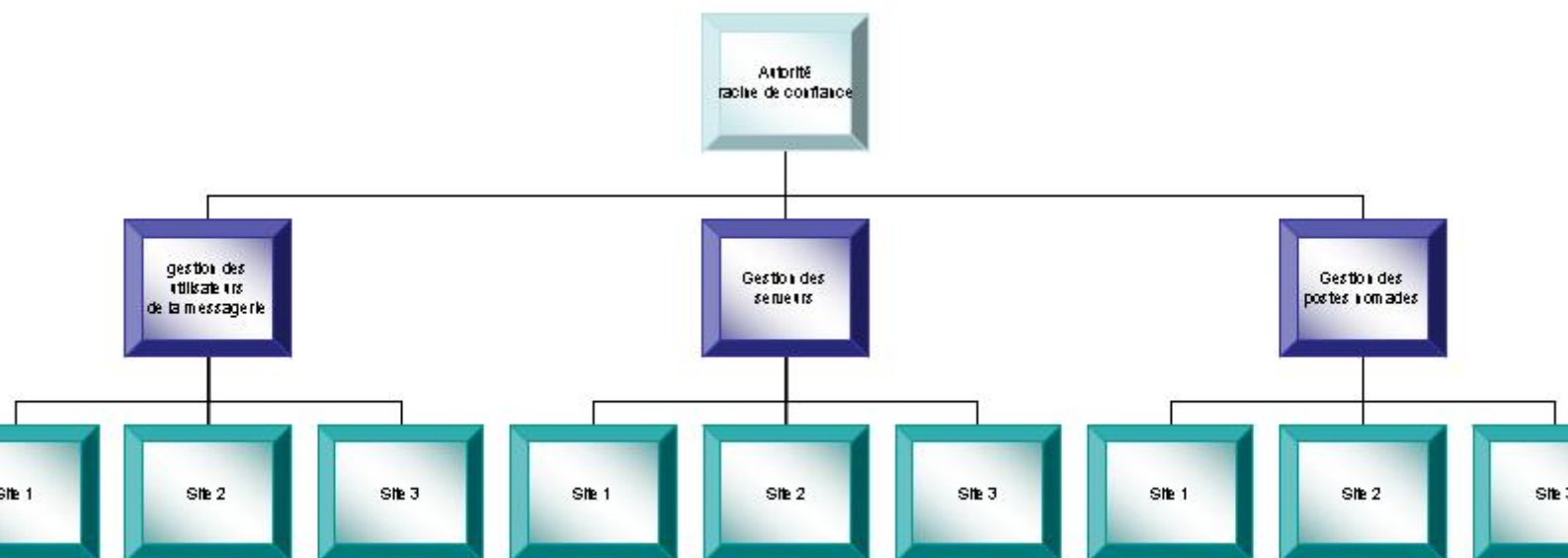
10.3.3 - L'autorité de certification

L'autorité de certificat est l'organisme ou l'entité qui signe les demandes de certificats émise par les usagers.

La signature d'un certificat par une autorité de certification permet d'apporter la crédibilité et la confiance que l'on a en cette autorité de certification sur un certificat.

Cette signature permet aussi de garantir que les informations transportées dans ce certificat ne sont pas modifiées ni altérées. Un usager ne peut pas modifier son certificat ou s'il le fait, cela sera immédiatement détecté car la signature du certificat ne correspondra plus avec celle générés par l'autorité de certification.

Il arrive souvent que l'autorité de certification soit hiérarchisée et déléguée. Il est courant dans les entreprises de trouver une architecture à plusieurs niveaux :



Exemple d'autorité de certification hiérarchisée

Dans cet exemple, on trouve une autorité de certification racine qui signe les certificats des autorités de certification de gestion des utilisateurs de la messagerie, de gestion des serveurs et de gestion des postes nomades. Chacune de ces trois entités signe à son tour un certificat pour chacun des trois sites de l'entreprise.

Au final, les certificats des usagers seront signés par une autorité de certification locale au site dont la mission est déléguée depuis l'autorité racine de confiance par les certificats intermédiaires.

10.4 - SSL

SSL (pour Secure Socket Layer) est un protocole réseau de chiffrement. La première version de ce protocole a été développée par Netscape pour assurer la sécurité des échanges sur Internet lors des phases sensibles (paiement sécurisé entres autres).

Les différentes versions de SSL sont les versions 1.0 et 2.0. La version actuelle est la version 3.0 et la version 3.1 correspond à la version TLS 1.0 standardisée par l'**IETF** (Internet Engineering Task Force). TLS (Transport Layer Security) est décrit dans la **RFC 2246**.

SSL est totalement transparent par rapport aux données transportée. Il ne peut s'appuyer que sur les protocoles TCP.

Le protocole SSL définit en fait trois sous protocoles qui sont :

- Le protocole **Handshake**
- Le protocole **Change cipher spec**
- Le protocole **Alert**

10.4.1 - Le protocole Handshake

Ce protocole permet de vérifier l'authentification obligatoire du serveur et l'authentification facultative du client. Elle permet de spécifier la clé de session qui sera utilisée lors des échanges sécurisés. Elle permet aussi de négocier les algorithmes utilisés lors de la phase de chiffrement.

Les messages possibles lors du protocole **Handshake** sont :

- hello_request (valeur 0)
- client_hello (valeur 1)
- server_hello (valeur 2)
- certificate (valeur 11)
- server_key_exchange (valeur 12)
- certificate_request (valeur 13)
- server_hello_done (valeur 14)
- certificate_verify (valeur 15)
- client_key_exchange (valeur 16)
- finished (valeur 20)

10.4.2 - Le protocole Change cipher spec

Ce protocole comprend en fait un seul message qui permet au protocole SSL d'utiliser les algorithmes négociés lors de la phase Handshake du protocole et de passer en mode chiffré.

Le seul message possible lors du protocole **Change cipher spec** est :

- change_cipher_spec(valeur 1)

10.4.3 - Le protocole Alert

Le protocole **Alert** est le protocole qui permet de remonter les messages d'erreur protocolaires.

Les messages possibles lors du protocole **Alert** peuvent avoir 2 niveaux : **Warning** ou **Fatal**. Les messages d'alerte sont :

- close_notify (valeur 0)
- unexpected_message (valeur 10)
- bad_record_mac (valeur 20)
- decryption_failed (valeur 21)
- record_overflow (valeur 22)
- decompression_failure (valeur 30)
- handshake_failure (valeur 40)
- bad_certificate (valeur 42)
- unsupported_certificate (valeur 43)
- certificate_revoked (valeur 44)
- certificate_expired (valeur 45)
- certificate_unknown (valeur 46)
- illegal_parameter (valeur 47)
- unknown_ca (valeur 48)
- access_denied (valeur 49)
- decode_error (valeur 50)
- decrypt_error (valeur 51)
- export_restriction (valeur 60)
- protocol_version (valeur 70)
- insufficient_security (valeur 71)
- internal_error (valeur 80)
- user_canceled (valeur 90)
- no_renegotiation (valeur 100)

10.4.4 - La phase de négociation du protocole Handshake

Les échanges protocolaires entre le client SSL et le serveur SSL sont synthétisés dans le tableau suivant :

Client	Serveur
ClientHello	
	ServerHello Certificate (optionnel) ServerKeyExchange (optionnel) CertificateRequest (optionnel) ServerHelloDone
Certificate (optionnel) ClientKeyExchange CertificateVerify (optionnel) ChangeCipherSpec Finished	
	ChangeCipherSpec Finished
Application Data	Application Data

Il existe aussi un protocole **Handshake** simplifié qui permet de reprendre le protocole sur erreur ou aussi de dupliquer une session SSL. L'échange protocolaire est alors :

Client	Serveur
ClientHello	
	ServerHello ChangeCipherSpec

	Finished
ChangeCipherSpec Finished	
Application Data	Application Data

On trouvera tous les détails de spécification du protocole dans la RFC 2246.

10.4-5 - Les algorithmes négociés par SSL

Les combinaisons d'algorithmes négociables par SSL sont les suivantes :

Algorithme asymétrique	Algorithme symétrique	Algorithme d'empreinte
RSA	Aucun	MD5
RSA	Aucun	SHA
RSA export	RC4 40 bits	MD5
RSA	RC4 128 bits	MD5
RSA	RC4 128 bits	SHA
RSA export	RC2 40 bits mode CBC	MD5
RSA	IDEA mode CBC	SHA
RSA export	DES 40 bits mode CBC	SHA
RSA	DES mode CBC	SHA
RSA	3DES EDE mode CBC	SHA
DH et DSS export	DES 40 bits mode CBC	SHA
DH et DSS	DES mode CBC	SHA
DH et DSS	3DES EDE mode CBC	SHA
DH et RSA export	DES 40 bits mode CBC	SHA
DH et RSA	DES mode CBC	SHA
DH et RSA	3DES EDE mode CBC	SHA
DHE et DSS export	DES 40 bits mode CBC	SHA
DHE et DSS	DES mode CBC	SHA
DHE et DSS	3DES EDE mode CBC	SHA
DHE et RSA export	DES 40 bits mode CBC	SHA
DHE et RSA	DES mode CBC	SHA
DHE et RSA	3DES EDE mode CBC	SHA
DH anonyme export	RC4 40 bits	MD5
DH anonyme	RC4 128 bits	MD5
DH anonyme export	DES 40 bits mode CBC	SHA
DH anonyme	DES mode CBC	SHA
DH anonyme	3DES EDE mode CBC	SHA

10.4.6 - Les ports utilisés par SSL

Les principaux protocoles transportés par SSL et les ports utilisés par défaut sont :

Protocole	Port par défaut	Port SSL par défaut
HTTP	80	443
SMTP	25	465
NNTP	119	563
POP3	110	995
IMAP	143	993
TELNET	23	992

10.5 - OpenSSL

OpenSSL est un projet collaboratif dont le but est d'offrir une implémentation d'algorithmes liés à la cryptographie.

Tous les algorithmes ne sont pas implémentés principalement pour des raisons de licences non compatibles avec la licence OpenSSL.

Le projet se compose de deux éléments principaux :

- Une bibliothèque écrite en C,
- Une interface ligne de commande.

A la date de rédaction de ce document, la dernière version OpenSSL est la version 0.9.8j du 7 janvier 2009.

10.5.1 - Contenu de la bibliothèque

Les fonctionnalités implémentées dans la bibliothèque OpenSSL sont :

- Algorithmes symétriques : blowfish, cast, des, idea, rc2, rc4, rc5,
- Algorithmes asymétriques : dsa, dh, rsa,
- Algorithme de hash : hmac, md2, md4, md5, mdc2, ripemd, sha,
- Gestion des certificats : x509, x509v3,
- Gestion des encodages : asn1, bio, evp, pem, pkcs#7, pkcs#12

Cette bibliothèque permet aux programmes qui le souhaitent d'utiliser des fonctions cryptographiques de manière simple et éprouvée à l'aide d'une **API très documentée**.

10.5.2 - L'interface ligne de commande

L'interface **ligne de commande** d'OpenSSL permet d'utiliser la bibliothèque OpenSSL et tous ses algorithmes. Il est possible par le biais de l'interface **ligne de commande** de créer un certificat X.509, de signer ce certificat, de chiffrer un fichier en utilisant un algorithme particulier et une clé donnée, de déchiffrer un fichier, de calculer le code de hachage d'un fichier, et beaucoup d'autres choses encore.

Une autre possibilité de cette interface **ligne de commande** est qu'il est possible de lancer un serveur SSL à l'écoute sur un port particulier ou un client SSL qui va se connecter sur un serveur SSL. Cette fonctionnalité est très utile pour le débogage des applications.

10.6 - Crypto++

Crypto++ est un autre projet collaboratif dont le but est d'offrir une implémentation d'algorithmes liés à la cryptographie.

Comme son nom l'indique, il s'agit d'une bibliothèque de classes C++. La dernière version de cette bibliothèque est la version 5.5.2 du 24 septembre 2007.

Les algorithmes implémentés par cette bibliothèque sont :

high speed stream ciphers	Panama, Salsa20, Sosemanuk
AES and AES candidates	AES (Rijndael), RC6, MARS, Twofish, Serpent, CAST-256
other block ciphers	IDEA, Triple-DES (DES-EDE2 and DES-EDE3), Camellia, RC5, Blowfish, TEA, XTEA, Skipjack, SHACAL-2
block cipher modes of operation	ECB, CBC, CBC ciphertext stealing (CTS), CFB, OFB, counter mode (CTR)
message authentication codes	VMAC, HMAC, CBC-MAC, DMAC, Two-Track-MAC
hash functions	SHA-1, SHA-2 (SHA-224, SHA-256, SHA-384, and SHA-512), Tiger, WHIRLPOOL, RIPEMD-128, RIPEMD-256, RIPEMD-160, RIPEMD-320
public-key cryptography	RSA, DSA, ElGamal, Nyberg-Rueppel (NR), Rabin, Rabin-Williams (RW), LUC, LUC-ELG, DLIES (variants of DH/AES), ESIGN
padding schemes for public-key systems	PKCS#1 v2.0, OAEP, PSS, PSSR, IEEE P1363 EMSA2 and EMSA5
key agreement schemes	Diffie-Hellman (DH), Unified Diffie-Hellman (DH2), Menezes-Qu-Vanstone (MQV), LUCDIF, XTR-DH
elliptic curve cryptography	ECDSA, ECNR, ECIES, ECDH, ECMQV
insecure or obsolescent algorithms retained for backwards compatibility and historical value	MD2, MD4, MD5, Panama Hash, DES, ARC4, SEAL 3.0, WAKE, WAKE-OFB, DESX (DES-XEX3), RC2, SAFER, 3-WAY, GOST, SHARK, CAST-128, Square

Les autres fonctionnalités de la bibliothèque sont :

- pseudo random number generators (PRNG): ANSI X9.17 appendix C, RandomPool
- password based key derivation functions: PBKDF1 and PBKDF2 from PKCS #5, PBKDF from PKCS #12 appendix B
- Shamir's secret sharing scheme and Rabin's information dispersal algorithm (IDA)
- fast multi-precision integer (bignum) and polynomial operations
- finite field arithmetics, including GF(p) and GF(2ⁿ)
- prime number generation and verification
- useful non-cryptographic algorithms
 - DEFLATE (RFC 1951) compression/decompression with gzip (RFC 1952) and zlib (RFC 1950) format support
 - hex, base-32, and base-64 coding/decoding
 - 32-bit CRC and Adler32 checksum
- class wrappers for these operating system features (optional) :
 - high resolution timers on Windows, Unix, and Mac OS
 - Berkeley and Windows style sockets
 - Windows named pipes
 - /dev/random, /dev/urandom, /dev/srandom
 - Microsoft's CryptGenRandom on Windows
- A high level interface for most of the above, using a filter/pipeline metaphor
- benchmarks and validation testing
- x86, x86-64 (x64), MMX, and SSE2 assembly code for the most commonly used algorithms, with run-time CPU feature detection and code selection
 - supports GCC-style and MSVC-style inline assembly, and MASM for x64
- certain versions are available in FIPS 140-2 validated form

11 - La cryptographie quantique

Jusqu'à présent, les différents algorithmes présentés reposaient en grande partie sur les mathématiques. Une nouvelle classe d'algorithmes est en train d'émerger : il s'agit de la cryptographie quantique.

Ces algorithmes ne sont plus basés sur des principes mathématiques mais plutôt sur les principes de la physique quantique.

La cryptographie quantique n'est pas vraiment un algorithme de chiffrement. Elle met en œuvre l'algorithme de cryptographie du masque jetable qui est le seul algorithme dont l'absence de faille est prouvée (voir ou revoir le paragraphe 7.5).

Cet algorithme est très sûr mais il nécessite une clé aléatoire partagée entre l'émetteur et le récepteur aussi longue que le message à échanger. L'échange de cette clé fait que cet algorithme est peu facile à utiliser.





11.1 - Propriétés quantiques d'un photon

La cryptographie quantique repose sur les propriétés quantiques du photon polarisé. La compréhension de ces propriétés est indispensable pour comprendre la cryptographie quantique.

- Un photon peut être polarisé selon un axe quelconque.
- Un photon polarisé selon un angle α passant au travers d'un filtre polarisant d'angle β possède une chance égale à $\cos^2(\beta - \alpha)$ de passer au travers de ce filtre polarisant. Donc :
 - Si le filtre est orienté précisément dans l'axe de polarisation du photon ($\beta = \alpha$), la probabilité que le photon traverse le filtre est $\cos^2(\beta - \alpha) = \cos^2(0) = 1$, c'est-à-dire qu'il traverse le filtre.
 - Si le filtre est orienté à 90° de l'axe de polarisation du photon ($\beta = \alpha + 90$), la probabilité que le photon traverse le filtre est $\cos^2(\beta - \alpha) = \cos^2(90) = 0$, c'est-à-dire qu'il est arrêté par le filtre.
 - Si le filtre est orienté à 45° de l'axe de polarisation du photon ($\beta = \alpha + 45$), la probabilité que le photon traverse le filtre est $\cos^2(\beta - \alpha) = \cos^2(45) = 1/2$, c'est-à-dire que le photon a une chance sur deux de passer le filtre.
- Ces propriétés sont encore du domaine dit de la **physique classique**. Les autres propriétés purement quantiques utilisées sont :
 - Quand la probabilité de passer au travers du filtre n'est ni 0 ni 1, le passage d'un photon individuel au travers du filtre polarisant est fondamentalement imprévisible et indéterministe.
 - On ne peut connaître l'axe de polarisation d'un photon qu'en le faisant passer au travers d'un filtre polarisant. Il n'existe pas de mesure directe permettant de mesurer l'angle de polarisation d'un photon.
 - On ne peut connaître l'axe de polarisation d'un photon qu'en utilisant un filtre polarisant dont l'axe de polarisation est exactement à 0° ou à 90° par rapport à l'axe de polarisation du photon.

La table de vérité ci-dessous présente les probabilités qu'un photon passe au travers d'un filtre polarisant suivant les différents angles du photon (colonne de gauche) et suivant les différents angles de filtre (ligne du haut) :

	Filtre polarisé à 0°	Filtre polarisé à 45°	Filtre polarisé à 90°	Filtre polarisé à 135°
Photon polarisé à 0°	1	1/2	0	1/2

				
Photon polarisé à 45°	1/2	1	1/2	0
				
Photon polarisé à 90°	0	1/2	1	1/2
				
Photon polarisé à 135°	1/2	0	1/2	1
				

11.2 - Transmission de la clé

La transmission de la clé consiste à transmettre une série de bits aléatoires prenant comme valeur 0 ou 1. Alice transmet chaque bit en choisissant aléatoirement un des deux modes de polarisation possibles :

- Le mode de polarisation **rectiligne** consiste à envoyer un photon polarisé à 0° pour un bit **0** et à 90° pour un bit **1**.
- Le mode de polarisation **diagonale** consiste à envoyer un photon polarisé à 45° pour un bit **0** et à 135° pour un bit **1**.

Alice émet chaque bit de la clé, photon par photon en notant pour chaque photon envoyé et donc pour chacun des bits le mode de polarisation choisi.

Bernard dispose d'un filtre polarisant pouvant être orienté au choix à 0° et 180° (ceci correspond au mode de polarisation **rectiligne**) ou à 45° et 135° (ceci correspond au mode de polarisation **diagonale**). Il positionne le filtre aléatoirement selon 1 des 2 modes possibles. Lors de l'arrivée d'un photon, il note le résultat, c'est à dire si le photon a passé ou non le filtre ainsi que le mode de polarisation qu'il avait choisi pour le filtre.

Pour chacun des photons reçus, il y a deux possibilités :

- Alice et Bernard ont par hasard choisi le même mode de polarisation. Dans ce cas, le photon reçu correspond au bit émis. Cela se produit en moyenne pour un photon sur deux.
- Alice et Bernard ont choisi un mode de polarisation différent et dans ce cas le photon reçu est interprété de mauvaise manière. Cela se produit en moyenne pour un photon sur deux.

Une fois que tous les bits sont transmis, Alice communique à Bernard, par un moyen conventionnel et non forcément fiable, le mode de polarisation employé pour chacun des bits.

Bernard peut donc alors connaître les bits pour lesquels le mode de polarisation a été le même. Il connaît donc alors de manière certaine N bits en moyenne pour 2xN bits transmis.

Par exemple :

Bits envoyés par Alice	1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 1 0 1
Polarisation choisie par Alice	+ x + x + x + + + + x x x + x x x + x + x + + x + x x x + + x +
Polarisation choisie par Bernard	+ x x x + x + x + + x x + + + + x x x + x x + x x + x x + x x + x + x

Valeurs lues par Bernard	1	1	0	0	1	0	1	0	1	1	1	0	1	0	0	1	0	0	1	1	0	1	0	0	0	1	0	1	0	0	0
Bits retenus	1	1	-	0	1	0	1	-	1	1	1	-	1	-	-	1	-	0	1	1	-	1	0	-	-	1	0	1	-	-	-

Jusqu'ici, ce protocole n'est qu'une manière (très compliquée) de communiquer N bits entre Alice et Bernard. L'avantage de cette méthode est que Bernard peut avoir la certitude absolue que les photons et donc la clé, n'ont pas été interceptés par Estelle. Un autre avantage non négligeable est qu'Estelle ne peut jamais être sûre d'avoir bien intercepté le message transmis par Alice.

Cela est possible car, dans le cas où l'on choisit une mauvaise orientation pour le filtre, la polarisation du photon reçu est parfaitement aléatoire et ne donne aucune information sur sa polarisation initiale.

Estelle est obligé, elle aussi, d'utiliser un filtre polarisant pour connaître la polarisation du photon qui code la valeur du bit. Pour passer inaperçue, Estelle doit réémettre le photon avec le même état de polarisation que le photon reçu. Mais si Estelle a choisi une mauvaise orientation du filtre pour recevoir le photon (cela arrive en moyenne une fois sur deux), elle va réémettre le photon avec une mauvaise polarisation.

Dans le cas où Estelle intercepte des photons sur la ligne, il peut arriver des cas où Bernard reçoit un bit différent du bit émis par Alice même s'ils ont choisi le même mode de polarisation.

Afin de tester la présence d'Estelle sur la ligne, Alice va, après avoir communiqué les modes de polarisation employés pour chaque photon, communiquer également la valeur de certains bits pour lesquels les orientations choisies par Alice et Bernard sont les mêmes.

Ces bits sont donc "sacrifiés" puisqu'ils sont communiqués par un canal de communication non sûr. Maintenant, si un seul de ces bits diffère entre Alice et Bernard, ils peuvent être sûrs qu'Estelle a tenté d'intercepter la clé et réagir en conséquence (abandon de la communication, choix d'un autre canal, etc.).

Ce concept d'échange de clé a été inventé en 1984 par les canadiens Charles Bennet et Gilles Brassard. Il faudra attendre 1989 pour voir la première démonstration physique mettant en œuvre ce concept.

11.3 - Etat actuel de la cryptographie quantique

Actuellement, même si les concepts sont parfaitement connus, la cryptographie quantique n'en est qu'à ses débuts du fait des difficultés technologiques à la mettre en œuvre.



Le système QKD (Quantum Key Distribution) de Toshiba utilisé dans le réseau à Vienne

Ce système a montré que des clefs quantiques pouvaient être envoyées le long d'une fibre optique de 20 kilomètres avec un débit supérieur à 1 Mbit/s - une performance qui pourrait permettre à des utilisateurs de communiquer avec une sécurité totale à travers les réseaux informatiques ([source Techno-Science](#) du 13 octobre 2008).

12 - Conclusion

Voilà, ce rapide tour d'horizon de la cryptographie est maintenant terminé. Pour conclure, je voudrais ajouter quelques recommandations issues de mon expérience personnelle :

- La cryptographie est une science des mathématiques complexe. Ne vous lancez pas dans la conception d'un algorithme **inviolable** si vous ne possédez pas de solides bases en mathématiques. Tous les algorithmes solides ont été inventés par des mathématiciens, la cryptographie est une science où l'amateurisme n'a plus sa place.
- Ne vous lancez pas dans l'implémentation d'un algorithme sans avoir de quoi le tester (un jeu de tests complet). En effet, il est parfois tentant de vouloir **améliorer** un algorithme et d'introduire alors un effet de bord qui fera qu'il ne fonctionne plus tout à fait pareil. Un nombre signé transformé en nombre non signé ou un entier 16 bits promu en entier 32 bits peuvent avoir de graves conséquences sur le fonctionnement de l'algorithme.
- Utilisez quand c'est possible les fonctionnalités de votre machine ou de votre système d'exploitation. La Crypto API de Windows ou encore OpenSSL sont des bibliothèques éprouvées aussi bien par les concepteurs que par les utilisateurs.
- Préférez l'utilisation des algorithmes publics dont les sources sont disponibles et qui ont été examinés et validés par des personnes extérieures à un algorithme privé et dont en fait vous ne savez rien si ce n'est que l'éditeur vous garantit qu'il est fiable.

Merci à [_Mac_](#) pour le travail effectué lors de la relecture de ce document.

13 - Annexes

13.1 - Quelques livres

<p>Cryptographie appliquée 2ème édition écrit par Bruce Schneier et traduit par Laurent Viennot. Une référence en termes d'algorithmes. Nécessite tout de même quelques notions en mathématiques et aussi en développement.</p>	
<p>Merveilleux nombres premiers écrit par Jean-Paul Delahaye. Ce livre est essentiellement axé sur les nombres premiers (normal vu le titre) mais possède un chapitre complet et détaillé sur la cryptographie asymétrique.</p>	
<p>Histoire des codes secrets écrit par Simon Singh. Ce livre se lit comme un roman et est abordable par le commun des mortels avec un niveau de connaissances mathématiques normal. Il retrace l'évolution de la cryptographie depuis l'antiquité jusqu'à nos jours.</p>	
<p>Cryptography in C and C++ écrit par Michael Welschenbach. Ce livre, comme : nom l'indique, s'intéresse à la cryptographie en C et en C++. Il explique la cryptographie et les mécanismes mathématiques sous jacents avec une approche spécifiquement dédiée aux développeurs.</p>	

13.2 - Quelques sites Internet

- Le portail **wikipédia** concernant la cryptologie.
- **Un site web** pour tester vos capacités de cryptanalyse. Il s'agit d'un site fournissant des énigmes cryptographiques. La résolution d'une énigme permet de passer au niveau suivant. Bien sûr, le degré de complexité augmente au fur et à mesure que l'on avance.
- **Un site** qui explique le fonctionnement de SSL.
- Le tutoriel **Introduction à la cryptologie** écrit par Denis Lapoire sur developpez.com.

13.3 - Solution au courrier entre George Sand et Alfred de Musset

Pour la première lettre, il faut lire une ligne sur 2, ce qui donne :

Je suis très émue de vous dire que j'ai
toujours une envie folle de me faire
baiser et je voudrais bien que ce soit
par vous. Je suis prête à vous montrer mon
cul, et si vous voulez me voir aussi
toute nue, venez me faire une visite.
Je vous prouverai que je suis la femme
la plus profonde comme la plus étroite
dont vous puissiez rêver, puisque votre
bite est bien longue, bien dure et souvent
grosse. Accourez donc vite et venez me la
mettre.

Pour la réponse, il faut lire uniquement le premier mot de chaque ligne :

Quand
Voulez-vous
Vous
Que
Je
Couche
Avec
Vous

Et pour la réponse finale, il faut aussi lire uniquement le premier mot de chaque ligne :

Cette
Nuit