REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mouloud Mammeri de Tizi-Ouzou
Faculté de Génie Electrique et d'Informatique
Département d'Electronique

Mémoire de fin d'études

En vue de l'obtention du diplôme d'ingénieur d'état en électronique
Option : contrôle

Thème

Conception et réalisation d'un cryptosystème hybride

Proposé et dirigé par :
Mr Lahdir M.

Réalisé par :
Halit Ramdane
Habachou Mokrane

Promotion: 2008

Remerciéments

Nous tenons avant tout à remercier le bon Dieu qui nous a donné la volonté et le courage pour réaliser ce modeste travail.

Nous remercions particulièrement notre promoteur Mr M. LAHDIR pour l'aide et l'assistance qu'il nous a apportées.

Nos remerciements vont aussi aux membres du jury qui nous ont fait l'honneur de juger notre travail et de l'évaluer.

Dédicaces

Je dédie ce modeste travail à :

Mes tantes et leurs fils,
Ma chère fiancée Fadhila,
Mes camarades de section,
Sans oublier mes chers amis et tous ceux que j'aime.

Ramdane

Je dédie ce travail à :

Ma chère mère, et a mon père,
Mes adorables frères,
Ma chère grand-mère,
Mes tantes et mes oncles,
Et à tous les gens qui m'aiment… les autres aussi.

Mokrane

Sommaire

Chapitre IV : Tests et résultats

Introduction Générale

Depuis l'apparition de l'écriture, l'homme a toujours ressenti le besoin de dissimuler des informations. Il s'est mis à imaginer des procédés qui lui permettaient de protéger ses secrets. L'un d'entre eux, datant du VIème siècle avant J.C., consistait à enrouler une bande de papier autour d'un cylindre, puis à écrire le message sur la bande. Une fois déroulée, elle était envoyée au destinataire qui pouvait déchiffrer le message en utilisant un cylindre identique.

Pendant de nombreuses années, la cryptographie était exclusivement réservée au domaine militaire et diplomatique. Aujourd'hui, elle est devenue une science à part entière. Au croisement des mathématiques, de l'informatique, et parfois même de la physique, elle permet ce dont les civilisations ont besoin depuis qu'elles existent : le maintien du secret.

En effet, dans la société de l'information, l'usage de la cryptologie s'est banalisé. Téléphones mobiles, cartes bleues, télévision, décodeurs, Internet, etc. On ne compte plus les objets de la vie courante qui incorporent des mécanismes de sécurité. Des algorithmes cryptographiques assurent par exemple que personne ne peut téléphoner à vos frais, intercepter un numéro de carte de paiement sur la Toile, etc.

Notre but principal est la mise en place d'un cryptosystème se basant sur le principe de la cryptographie hybride et qui a pour fonction d'assurer la confidentialité des messages et autres fichiers de tous formats. Pour ce faire, le mémoire sera divisé en quatre chapitres :

Le premier englobera les généralités ainsi que les notions fondamentales sur la cryptographie, qui sont requises pour la bonne compréhension des chapitres suivants.

Dans le deuxième nous exposeront les fonctionnalités que peut nous offrir un logiciel de cryptographie hybride, on l'occurrence PGP, pour ensuite concevoir notre propre application basée sur le même principe, en s'attachant à bien expliquer ses différentes composantes, selon une étude détaillée de chacune d'elles.

Les deux derniers chapitres présenteront l'application, ainsi que les résultats fournis par cette dernière.

En fin, nous terminerons par une conclusion générale.

Chapitre I

I. Terminologie [1]:

Une certaine confusion règne concernant les différents termes de la cryptographie, à cause en premier lieu de l'utilisation d'anglicismes (termes empruntés à l'Anglais), ainsi nous allons définir la terminologie qui va être utilisée tout au long de l'étude afin d'éviter toute ambiguïté:

Clé : Une clé est un paramètre utilisé en entrée d'une opération cryptographique (chiffrement, déchiffrement).

Chiffrer ou Chiffrement : Transformation à l'aide d'une clé de chiffrement d'un message en clair en un message chiffré (**Cryptogramme**), incompréhensible par des tiers n'ayant pas la connaissance de la clé (en anglais Encryption). On utilise aussi le mot « Crypter ».

Déchiffrer ou Déchiffrement : Transformation qui consiste à retrouver les informations claires, à partir des informations chiffrées en utilisant la clé de déchiffrement.

Décrypter : Retrouver le message clair correspondant à un message chiffré sans posséder la clé de déchiffrement.

Cryptosystème : Un cryptosystème est constitué d'un algorithme cryptographique ainsi que toutes les clés possibles et tous les protocoles qui le font fonctionner.

Ceci dit, nous pouvons à présent donner une définition précise pour :

La Cryptographie : Etymologiquement « écriture secrète », devenue par extension l'étude de cet art (donc aujourd'hui la science visant à créer des cryptogrammes, c'est-à-dire à chiffrer).
La Cryptanalyse : science analysant les cryptogrammes en vue de les décrypter.
La Cryptologie : C'est une science mathématique regroupant la cryptographie et la cryptanalyse.

Il faut savoir aussi qu'il sera souvent fait mention d'Alice et Bob ; En cryptographie, plus par tradition, on nomme "Alice" et "Bob" les deux interlocuteurs qui veulent s'échanger confidentiellement des messages

II. La Cryptographie :

La cryptographie s'attache à la protection des données sensibles en s'aidant souvent de secrets ou de clés. Elle est utilisée depuis l'Antiquité, mais certaines de ses méthodes les plus importantes, n'ont que quelques dizaines d'années d'existence.

II.A. Différents services de la cryptographie [7]:

La cryptographie permet de rendre un certain nombre de services de base en sécurité informatique :

- La **confidentialité** est l'assurance qu'un document ne sera pas lu par un tiers qui n'en a pas le droit lors de la transmission de ce document ou lorsqu'il est archivé. Les documents papiers qui doivent rester secrets sont généralement stockés dans des coffres et sont transportés sous plis cachetés. Pour les documents électroniques, on utilisera le chiffrement, la confidentialité représente le service le plus important en cryptographie.

- L'**authentification** est l'assurance de l'identité d'un objet, généralement une personne, mais cela peut aussi s'appliquer à un serveur, une application, … Dans la vie courante, la présentation de la carte nationale d'identité et la signature manuelle assurent un service d'authentification ;

- L'**intégrité** d'un objet (document, fichier, message …) est la garantie que cet objet n'a pas été modifié par une autre personne que son auteur. Sur une feuille de papier toute modification est visible d'un simple coup d'œil. Sur un document électronique (Courrier électronique, fichier Word, …) non sécurisé, cette détection est impossible ;

- Un quatrième service de sécurité est appelée « **non répudiation ».** Comme ce terme l'indique, le but est que l'émetteur d'un message ne puisse pas nier l'avoir envoyé et le récepteur l'avoir reçu. Les transactions commerciales ont absolument besoin de cette fonction. Le reçu que l'on signe au livreur, la lettre recommandée sont des mécanismes de non répudiation.
  Les certificats permettent d'assurer ce service. Dans la communauté enseignement-recherche, cette fonction n'est pas primordiale si ce n'est pour certains actes administratifs (votes, notations, transferts de crédits, …).

II.B. Notions fondamentales sur la cryptographie :

Le chiffrement et le déchiffrement représentent les deux principales fonctions de la cryptographie, car elles assurent avant tout la confidentialité des donnés électroniques en particulier et de l'information en général.

Cette opération consiste à appliquer une fonction mathématique (en fait c'est un ensemble de fonctions), cette fonction utilise une variable, la clé de chiffrement qui est une suite aléatoire de bits. Une fois le texte chiffré, il devient illisible. Pour obtenir la version lisible, il faut le déchiffrer, c'est à dire appliquer une autre fonction mathématique, compatible avec la première, en utilisant une autre variable, la clé de déchiffrement. Les deux fonctions mathématiques sont regroupées dans ce qu'on appel algorithme cryptographique.

La valeur de la clé de déchiffrement dépend de la valeur de la clé de chiffrement et seul le possesseur de la clé de déchiffrement peut déchiffrer le texte.

Concrètement, il y a deux grands principes de chiffrement possibles. Le chiffrement à clé secrète ou Symétrique et le chiffrement à clé publique ou Asymétrique. A cela on rajoute le chiffrement hybride qui représente un compromis entre les deux systèmes déjà cités, et le chiffrement quantique qui a fait récemment ses premiers pas en dehors des laboratoires.

II.B.1. Chiffrement symétrique :

Le chiffrement symétrique (aussi appelé chiffrement à clé privée ou chiffrement à clé secrète) consiste à utiliser la même clé pour le chiffrement et le déchiffrement. Un tel système impose d'avoir un canal sécurisé pour l'échange de la clé.

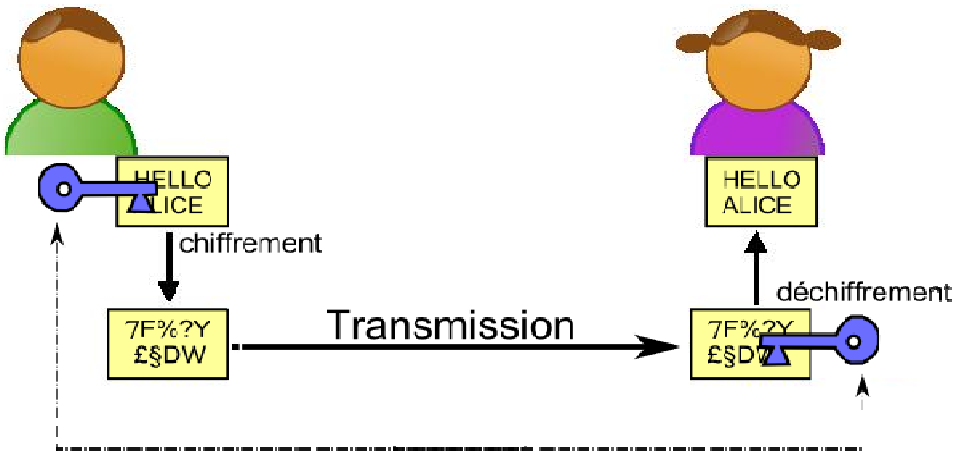


Fig.I.1. Procédure de chiffrement symétrique

Les premiers algorithmes utilisés pour le chiffrement d'une information étaient assez rudimentaires dans l'ensemble (à voir plus comme des conventions que de vrais algorithmes mathématiques). Ces anciennes méthodes ont constitué la première pierre de l'édifice qui nous mènera vers les méthodes de chiffrement dites modernes, car comme nous allons le voir ces dernières, au delà de leur complexité se basent sur des principes « artisanaux ».

II.B.1.a. Méthodes anciennes :

La confidentialité de l'algorithme de chiffrement était la pierre angulaire de ces systèmes pour éviter toute cryptanalyse rapide, déjà la notion de secret était bien présente. Ces principes on évolués au cours du temps jusqu'à fonder les bases de la cryptographie moderne.

De ces faits, on retiendra deux principes fondamentaux:

- **Le principe de Kerckhoffs :** A été énoncé par Auguste Kerckhoffs à la fin du 19^{éme} siècle dans un article en deux parties La cryptographie militaire. Ce principe exprime que la sécurité d'un cryptosystème ne doit reposer que sur le secret de la clé. Autrement dit, tous les autres paramètres doivent être supposés publiquement connus.

- **La Confusion et la Diffusion :** Dans le sens de la théorie de l'information de Claude Shannon, la confusion dans le texte est provoquée par la technique de substitution, ce qui correspond à une volonté de rendre la relation entre la clé de chiffrement et le texte chiffré la plus complexe possible, ou encore une augmentation du désordre. Quand à la diffusion elle s'accroit avec la technique de transposition, c'est à dire une dispersion de la redondance du texte en clair, en la répartissant dans le texte chiffré.

La substitution et la transposition continuent d'être utilisées dans les algorithmes symétriques les plus récents, voyons en détail ces deux techniques :

II.B.1.a.1. Chiffrement par substitution :

La substitution consiste à remplacer systématiquement une lettre par une autre (ou par un chiffre), par exemple une fonction de chiffrement qui décale de 1, 2, 3…, n positions les lettres constituant une phrase en respectant l'ordre de l'alphabet, ce type de substitution est dit mono-alphabétique.

Il existe un autre type de substitution, dit poly-alphabétique, qui introduit la notion de clés. La clé est choisie, de façon à avoir une taille supérieure à un caractère, et de l'appliquer sur tout le texte par tronçons successifs de la taille de la clé (c'est-à-dire, que chaque lettre du texte en clair, sera remplacée par une lettre différente en fonction de la lettre qui lui correspond dans la clé).

Exemple 1-- Cryptage de César :

Cette méthode de cryptage est considérée comme le plus ancien des algorithmes de chiffrement par substitution mono-alphabétique, dans la mesure où Jules César l'avait utilisé.

La technique est élémentaire : il suffit de remplacer chaque lettre du texte à chiffrer par la lettre qui se situe ***n*** places plus loin dans l'alphabet. Par exemple si n=3, on remplacera A par D, B par E, C par F etc. et on aura le tableau suivant :

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

Du tableau, chaque caractère du texte clair correspond un caractère du texte crypté, si on considère M_{CL} le message contenant le texte en clair, M_{CH} le message chiffré, C et D respectivement les fonctions de chiffrement et de déchiffrement, on aura :

$M_{CH} = C (K, M_{CL})$, avec K est la clé de chiffrement.

Si on prend : M_{CL}= Le décalage de César, K=3 (César) :

$M_{CH} = C (K, M_{CL}) = C (3,$ 'Le décalage de César')

M_{CH} = 'oh ghfdodjh gh fhvdu'

Comme ce chiffrement est symétrique, on utilise la même clé K pour le déchiffrement donc :

$M_{CL} = D (K, M_{CH})$.

D ne fera que décaler de K=3 crans vers la gauche les lettres du texte chiffré.

$M_{CL} = D (3,$ 'oh ghfdodjh gh fhvdu').
M_{CL} = 'Le décalage de César'

On peut aussi déchiffrer M_{CH} en utilisant la fonction D et la clé K=29.

Exemple 2--Cryptage de Vigenère :

L'un des premiers systèmes à introduire le concept de clé de chiffrement. Blaise de Vigenère (1523-1596) a repris un chiffre de César, où le décalage utilisé change de lettre en lettre. Pour cela, il a utilisé une table composée de 26 alphabets, écrits dans l'ordre, mais décalés de ligne en ligne d'un caractère (Mieux connue sous le nom de « table de Vigénere »). Puis, il a écrit en haut un alphabet complet, pour la clé, et à gauche, verticalement, un dernier alphabet, pour le texte à coder :

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

Fig.I.2. Table de Vigenère

Pour coder un message, on choisit une clé qui sera un mot de longueur arbitraire. On écrit ensuite cette clé sous le message à coder, en la répétant aussi souvent que nécessaire pour que sous chaque lettre du message à coder, on trouve une lettre de la clé. Pour coder, on regarde dans le tableau l'intersection de la ligne de la lettre à coder avec la colonne de la lettre de la clé.

Si on veut coder le texte "CRYPTOGRAPHIE DE VIGENERE" avec la clé "VIGENERE". On commence par écrire la clé sous le texte à coder :

C	R	Y	P	T	O	G	R	A	P	H	I	E		D	E		V	I	G	E	N	E	R	E
V	I	G	E	N	E	R	E	V	I	G	E	N		E	R		E	V	I	G	E	N	E	R

Pour coder la lettre C, la clé est donnée par la lettre V. On regarde dans le tableau l'intersection de la ligne donnée par le C, et de la colonne donnée par le V.

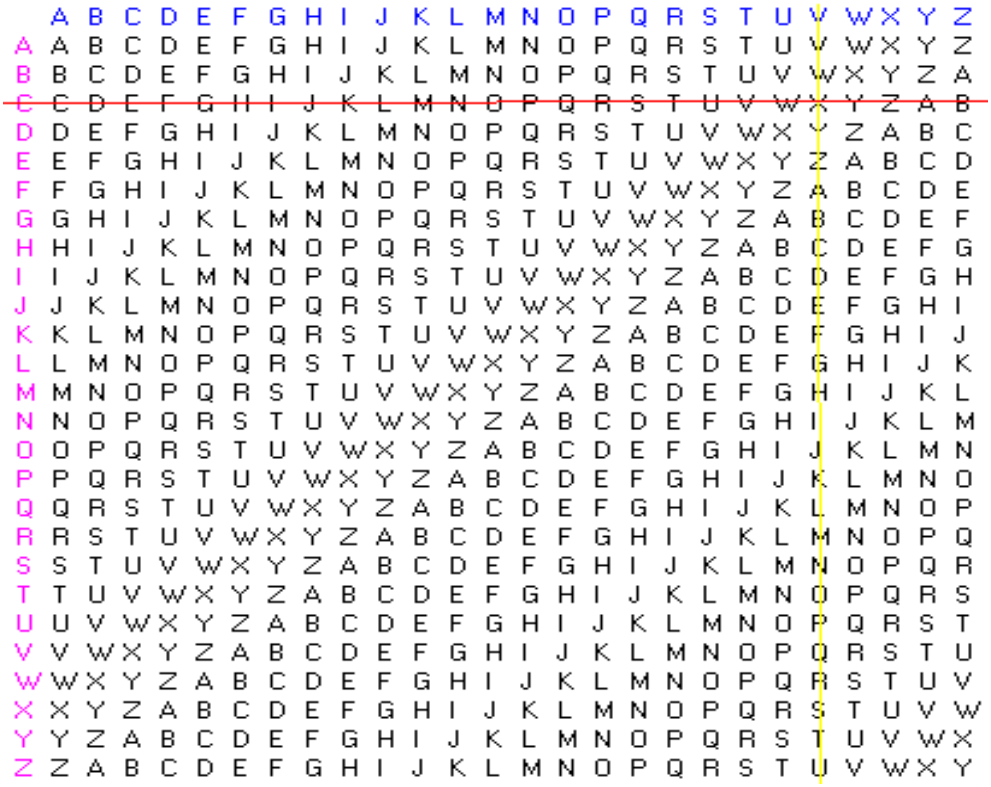


Fig.I.3. Substitution d'une lettre dans la Table de Vigenère

On trouve X. Puis on continue. On trouve : XZETGSXVVXNMR HV ZDOKRRVV

Dans cet exemple, comme la taille de la clé est supérieure à un caractère, le chiffrement est poly-alphabétique.

Exemple 3 : Le masque jetable (One Time Pad)

Le masque jetable est le seul algorithme de cryptage connu comme étant parfaitement sûr. C'est en fait un chiffre de Vigenère avec comme caractéristique une clé de chiffrement de même longueur que le message en clair. Le système du masque jetable fut inventé par **Gilbert Vernam** en 1917, puis fût perfectionné par le major **Joseph O. Mauborgne** en 1918, qui introduisit le concept de clé aléatoire.

Clair :	M	A	S	Q	U	E	J	E	T	A	B	L	E
Clef (Masque) :	X	C	A	A	T	E	L	P	R	V	G	Z	C
Chiffré :	J	C	S	Q	N	I	U	T	K	V	H	K	G

Pour chiffrer un texte de manière sûre avec le masque jetable, on se doit de respecter certaines conditions et pas des moindres:

- Choisir une clé aussi longue que le texte à chiffrer, et qui soit vraiment aléatoire, ce qui donne souvent des clés de tailles énormes et très difficile à acheminer.
- Une même clé ne doit pas servir à chiffrer plusieurs messages.

Avec l'avènement de l'informatique, on applique souvent le masque jetable au message en clair, en utilisant l'opérateur logique XOR, qui est simple et pratique.
Le masque jetable est couramment utilisé de nos jours par les gouvernements. En effet, ceux-ci peuvent communiquer les clefs de manière sûre via la valise diplomatique.

Exemple 4 : Le cryptage XOR :

Utilisé en masse dans les algorithmes cryptographiques modernes nous avons choisit de l'introduire dès maintenant.

Le cryptage XOR est un système de cryptage basique mais pas trop limité. Ainsi, il a beaucoup été utilisé dans les débuts de l'informatique et continue à l'être encore aujourd'hui, car il est facile à implémenter, dans toutes sortes de programmes.

Le XOR est un opérateur logique, qui correspond à un "OU exclusif", son principe est le suivant :

$M_{CH} = C(K, M_{CL}) = M_{CL}$ XOR $K = M_{CL} \oplus K$.

Comme l'algorithme est complètement symétrique, c'est-à-dire, la même opération est réappliquée au message final pour retrouver le message initial, on aura :

$M_{CL} = D(K, M_{CH}) = M_{CH}$ XOR $K = M_{CH} \oplus K$.

En informatique, chaque caractère du message à coder est représenté par un entier, c'est son code ASCII. Ce nombre est lui-même représenté en mémoire comme un nombre binaire à 8 chiffres (les bits). On choisit une clé que l'on place en dessous du message à coder, en la répétant autant de fois que nécessaire, comme dans le cryptage de Vigenère. Le message et la clé étant converti en binaire, on effectue un XOR, bit par bit. Le résultat en binaire peut être reconverti en caractères ASCII et donne alors le message codé.
Si on prend M_{CL} =' MESSAGE' et K = 'CLE' :

- Le mot MESSAGE converti en binaire :

Caractères	M	E	S	S	A	G	E
Code ASCII	77	63	83	83	65	71	69
Binaire	01001101	01000101	01010011	01010011	01000001	01000111	01000101

- Le mot CLE représenté en binaire :

La représentation binaire du mot CLE est : 01000011 - 01001100 – 01000101

- Le message chiffré en binaire sera :

Message en binaire	01001101	01000101	01010011	01010011	01000001	01000111	01000101
Clé en binaire	01000011	01001100	01000101	01000011	01001100	01000101	01000011
Message crypté en binaire	**00001110**	**00001001**	**00010110**	**00010000**	**00001101**	**00000010**	**00000110**

Exemple : Application du Masque jetable en utilisant l'operateur logique XOR

M = 'Le Petit Prince' = 4C 65 20 50 65 74 69 74 20 50 72 69 6E 63 65

k = 'OpkcUOZiDEpNrJV' = 4F 70 6B 63 55 4F 5A 69 64 65 70 4E 72 4A 56

C = M (Xor) k = 03 15 4B 33 30 3B 33 1D 44 35 20 27 1C 29 33

Essayons de cryptanalyser le texte chiffré avec différentes clés:

k1 = 4F 70 6B 74 42 5A 5D 79 64 77 6E 42 69 09 21

M1 = C (Xor) k1 = 'Le Grand Bleu !'

k2 = 4D 32 22 5D 34 52 40 69 21 4F 22 57 7D 5A 1D

M2 = C (Xor) k2 = 'N'insistez pas.'

Cet exemple montre qu'il n'existe pas une seule clé, donnant un seul texte déchiffré sensé. Il existe autant de clés utilisables avec C, qu'il existe de bouts de phrases tenant en 15 caractères.

II.B.1.a.2. Chiffrement par transposition :

La transposition, comme déjà définie, est un moyen de diffusion (Principe de Shannon), elle disperse la redondance du texte en clair en la répartissant dans le texte chiffré.
Le principe est simple : Les lettres se retrouvent mélangées (elles conservent leur identité, un E reste un E, mais changent de position)
Voyons un exemple de transposition dite en colonne simple : Une illustration simple (de ce qu'on appelle d'ailleurs la **transposition simple à tableau**, procédé le plus utilisé) consisterait à écrire un texte ligne par ligne dans un tableau à 10 colonnes puis à transmettre le message par colonne.
Prenons comme exemple le message :

"**CE MESSAGE EST UNE ILLUSTRATION SIMPLE DU PROCEDE DE TRANSPOSITION**"

Ecrivons ce message dans le tableau à 10 colonnes, en supprimant les espaces et la ponctuation (ici inexistante) :

C	**E**	**M**	**E**	**S**	**S**	**A**	**G**	**E**	**E**
S	**T**	**U**	**N**	**E**	**I**	**L**	**L**	**U**	**S**
T	**R**	**A**	**T**	**I**	**O**	**N**	**S**	**I**	**M**
P	**L**	**E**	**D**	**U**	**P**	**R**	**O**	**C**	**E**
D	**E**	**D**	**E**	**T**	**R**	**A**	**N**	**S**	**P**
O	**S**	**I**	**T**	**I**	**O**	**N**			

L'émetteur émet le message par colonne; cela donne la suite de caractère suivante :

CSTPDOETRLESMUAEDIENTDETSEIUTISIOPROALNRANGLSONEUICSESMEP

Il ne reste plus au destinataire qu'à déchiffrer : Il sait que le tableau à 10 colonnes. Il va donc reconstituer le tableau mais pour cela il faut d'abord qu'il trouve le nombre de lignes. Cela ne présente guère de difficulté à celui qui sait qu'il s'agit d'un tableau de 10 colonnes; il lui suffit de diviser le nombre de caractères par 10 : comme il a reçu 57 caractères, il en déduit qu'il y a 5 lignes de 10 caractères et 1 ligne incomplète de 7 caractères. Il va donc considérer 7 groupes de 6 caractères puis 3 groupes de 5 qu'il va ranger dans son tableau.
La connaissance de la langue lui permettra simplement ensuite de replacer les espaces et la ponctuation.

D'un point de vue pratique, il conviendrait d'ajouter le "conditionnement" du message permettant par exemple d'identifier l'émetteur ou d'identifier la ou les clefs de transposition car, fréquemment, plusieurs grilles sont en service en même temps.

II.B.1.a.3. Chiffrement par produit :

C’est la combinaison de la substitution et de la permutation. Le chiffrement par substitution ou par transposition ne fournit pas un haut niveau de sécurité, mais en combinant ces deux transformations, on peut obtenir un chiffrement robuste. La plupart des algorithmes de chiffrement par clé symétrique utilisent le chiffrement par produit.

II.B.1.b. Méthodes modernes :

De nos jour la cryptographie est indissociable ou presque de l’informatique. Le chiffrement moderne utilise la puissance des ordinateurs. Comme les données traitées par les ordinateurs sont uniquement sous forme numérique (bits), les procédés de substitution et de transposition sont toujours utilisés, mais maintenant seulement sur deux éléments primaires (0 et 1), en plus de ça le principe de Kirchhoff est souvent adopté par les cryptologues (Exception faite pour les militaires, qui penchent vers l’utilisation d’algorithmes secrets). On constate donc que les idées et les principes vus précédemment restent d’actualité. C’est pourquoi il était à notre avis intéressant de les mettre en valeur au préalable. Mais ce n’est pas tout, d’autres méthodes et principes, sont venus s’ajouter à ceux déjà existants, nous allons décrire les plus importants:

II.B.1.b.1. Le schéma de Feistel :

Si l'on sait depuis longtemps construire des fonctions qui ont l'air aléatoires, on ne savait pas avant les travaux de Feistel construire des bijections aléatoires. La solution apportée par Feistel est très élégante.
Un réseau de Feistel est subdivisé en plusieurs tours ou étages. Illustrés par la figure suivante :

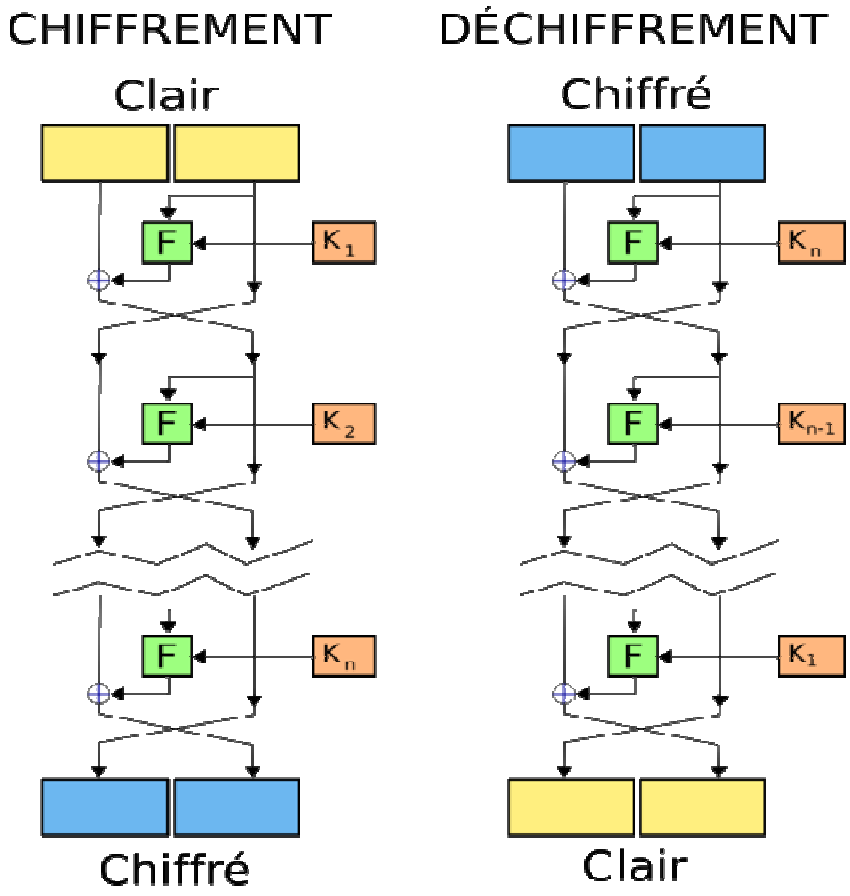


Fig.I.4. Schema de Feistel

Chaque tour applique plusieurs transformations sur les données provenant du tour précédent:

- Permutation des bits via des P-Boxes
- Substitution non-linéaire avec des S-Boxes
- Mixage linéaire en utilisant la fonction XOR
- Application de la clé du tour (intégrée dans une fonction ou via un XOR)

On utilise les termes de confusion et diffusion, comme déjà mentionner, pour décrire la propagation des informations dans la structure (termes utilisés par Claude Shannon). En effet, une modification d'un bit en entrée produira des variations très importantes dans les étages intermédiaires et en sortie.

La plupart des algorithmes à clé secrète de la fin du 20^{éme} siècle, étaient des schémas de Feistel (DES, Blowfish, Twofish, RC5). L'avènement de l'AES, qui n'en est plus un, marque la fin de la prédominance de tels algorithmes.

II.B.1.b.2. Les tables de substitution et de permutation :

- **S-box :**

  Les S-boxes, composantes des systèmes cryptographiques, sont des tables de substitution (ou boîte de substitution, fonction de substitution), elles contribuent à la « confusion ». Elles peuvent avoir plus d'entrées que de sorties, ou plus de sorties que d'entrées. Chacune d'elles rend difficile la table inversible.

- **P-box :**

  P-Box (permutation box), terme anglais désignant une table de permutation employée dans des algorithmes de chiffrement. Elle indique comment échanger les éléments d'une structure. Une P-Box contribue à la « diffusion » en mélangeant les données et en améliorant l'effet avalanche.

  Une P-Box peut se présenter sous plusieurs formes mais algorithmiquement, il s'agit en général d'un tableau à une dimension comme [1, 8, 5, 3, 4, 6, 7, 2]. Ce tableau signifie que le premier élément reste en place, que la deuxième sortie prend la valeur de la huitième entrée, que la troisième sortie prend la valeur de la cinquième, etc.

II.B.1.b.3. Modes de chiffrement :

La cryptographie à algorithmes symétriques fonctionne habituellement suivant deux procédés différents, le chiffrement par blocs et le chiffrement par flot (en continu).

II.B.1.b.3.1. Chiffrement par bloc :

Les algorithmes de chiffrement par bloc (block-cipher) sont les plus utilisés et permettent une meilleure sécurité. Les algorithmes concernés ont également plus de renommée, on citera l'AES (standard actuel), le DES (l'ancien standard), l'IDEA. La taille des blocs, et celle de la clé varient suivant l'algorithme et suivant le niveau de sécurité requis.

Le message est découpé en blocs de même taille (suivant les spécifications de l'algorithme utilisé : 64 bits, 128 bits, 196 bits, 256 bits...), qui sont ensuite chiffrés un par un suivant plusieurs modes d'opération et qui sont transmis ainsi.

Un mode d'opération est la manière de traiter les blocs de données claires et chiffrées au sein d'un algorithme de chiffrement par bloc, il est plus ou moins indépendant de l'algorithme choisi. Toutefois, tous les algorithmes ne permettent pas d'utiliser tout les modes possibles.

Quatre modes sont définis comme standards :

- Electronic CodeBook (ECB)
- Cipher Block Chaining (CBC)
- Cipher FeedBack (CFB)
- Output FeedBack (OFB)

- **Le mode Electronic CodeBook (ECB) [18]:**

C'est le plus simple à mettre en œuvre des modes d'opération. Il revient à chiffrer un bloc indépendamment des autres, cela permet entre autre de chiffrer suivant un ordre aléatoire (bases de données, etc.) mais en contrepartie, ce mode est très vulnérable aux attaques. Il est fortement déconseillé de l'utiliser, nous l'avons mentionné ici parce que par sa simplicité il permet une bonne compréhension de ce que sont les modes de chiffrement par bloc.

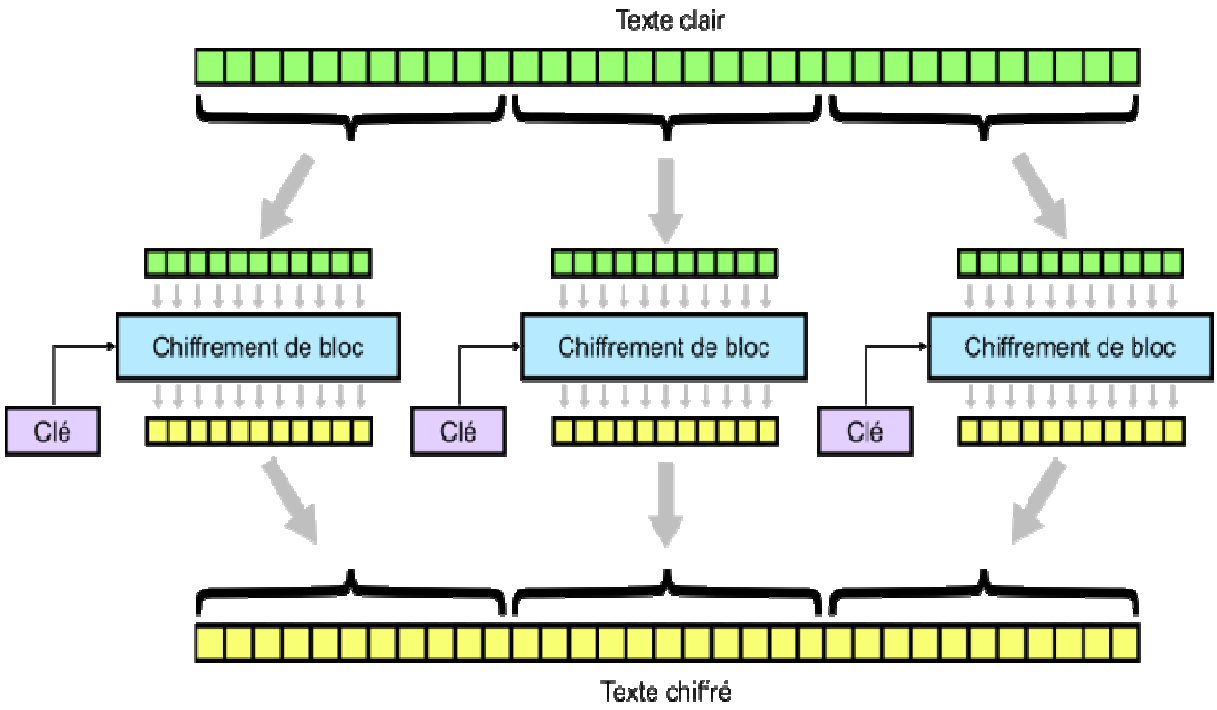


Fig.I.5. Mode ECB

- **Le mode Cipher Block Chaining (CBC) [18]:**

C'est le mode le plus courant. Il permet d'introduire une complexité supplémentaire dans le processus de chiffrement en créant une dépendance entre des blocs successifs. Il effectue un XOR entre un bloc de données en clair et un bloc de données cryptées. Quand au premier bloc il est XORé avec un vecteur appelé vecteur d'initialisation (Initialisation Vector,

IV) qui peut être un mot de passe par exemple, ce vecteur change à chaque session, et doit être transmis au destinataire.

Le schéma de base d'un tel mode est le suivant :

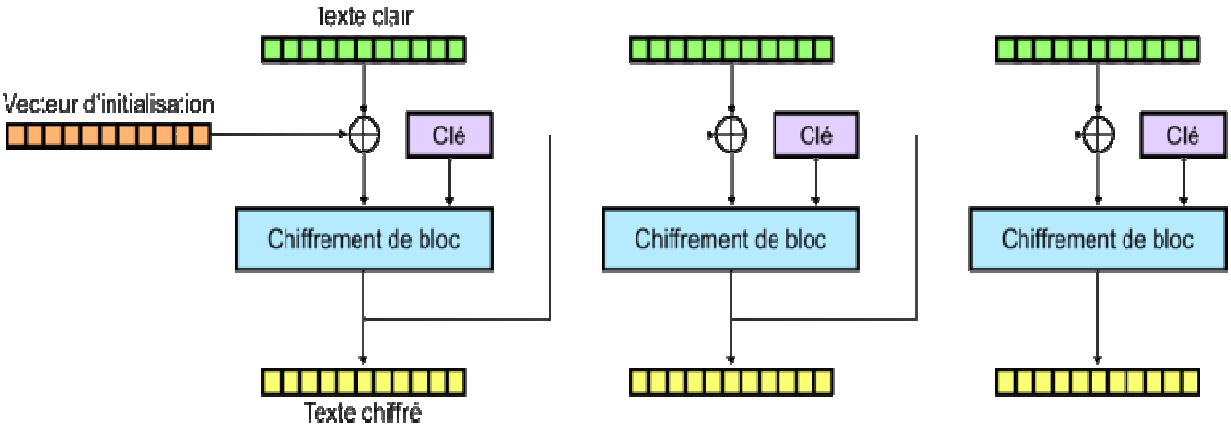


Fig.I.6. Mode CBC

- **Le mode Cipher FeedBack (CFB) [6]:**

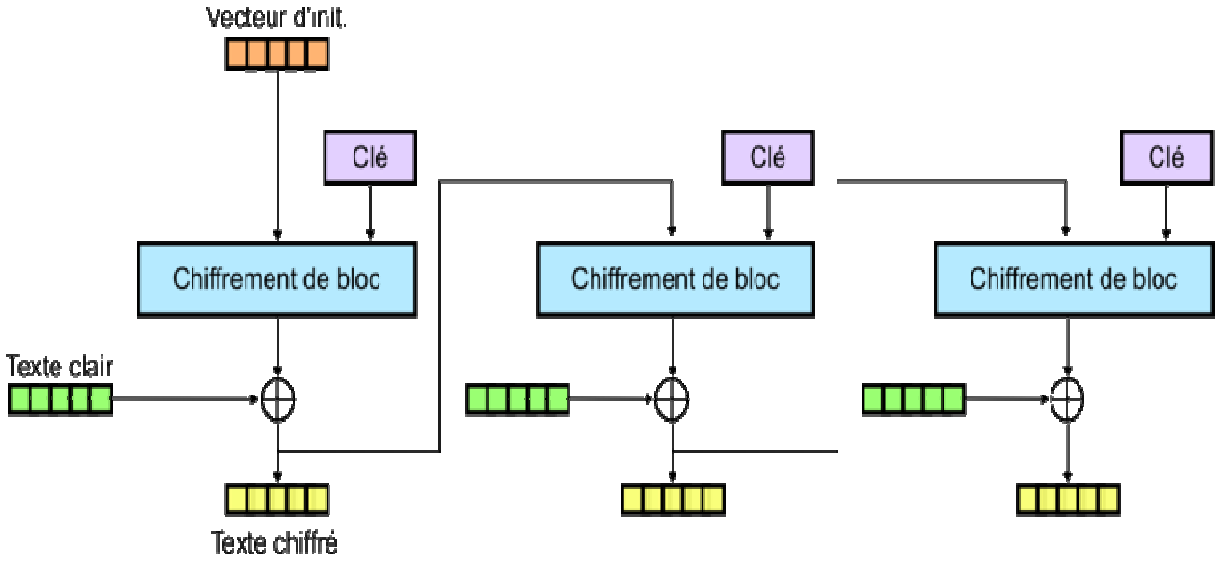


Fig.I.7. Mode CFB

Le message est ajouté par un XOR à la sortie du bloc chiffré. Le résultat sert d'entrée pour l'étape suivante. Il est utilisé pour le chiffrement par flux.

- **Le mode « Output Feedback » (OFB) [6] :**

Une variante du CFB. La différence ici, c'est que le flux entrant vers les étapes ultérieures est indépendant du message clair.

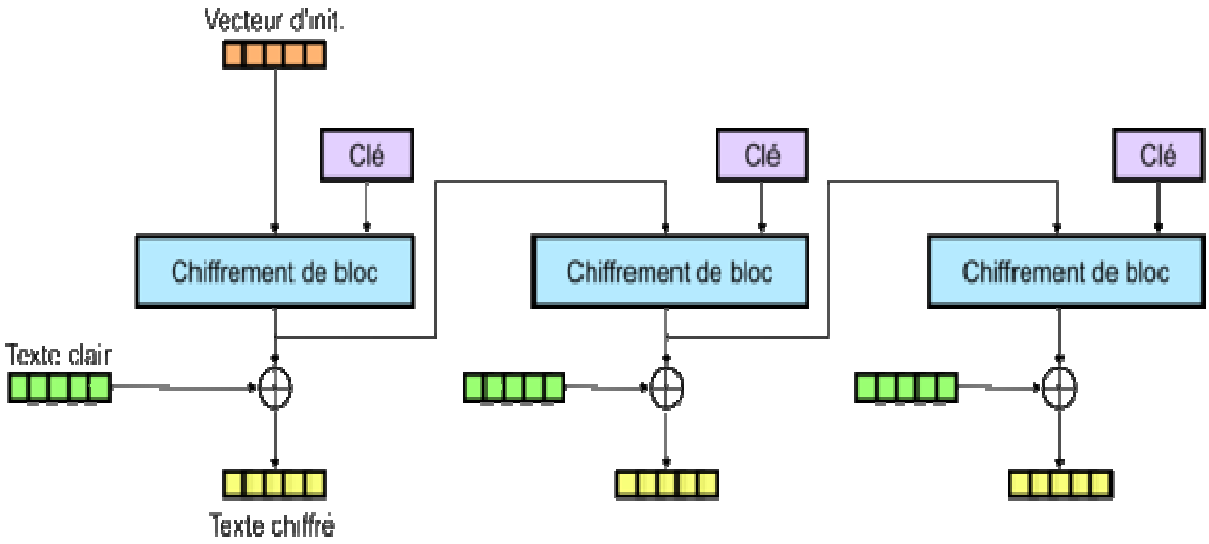


Fig.I.8. Mode OFB

II.B.1.b.3.2. Chiffrement par flot :

Les algorithmes de chiffrement par flot, appelés aussi Chiffrement de flux (stream ciphers), peuvent être définis comme étant des algorithmes de chiffrement par bloc, où le bloc a une dimension unitaire (1 bit, 1 octet, etc.) ou relativement petite. À noter qu'il n'y a aucun standard parmi les chiffrements par flot.

II.B.1.b.4. Avantages et inconvénients [6]:

Voici un petit récapitulatif des avantages et inconvénients des systèmes de chiffrement symétriques modernes.

Avantages	Inconvénients
• Rapidité • Facilité d'implantation sur hardware • Taille de la clé : Souvent de 128 bits (16 caractères mémorisables)	• Nombre de clés à gérer : pour un groupe de N personnes, il faut en tout **N(N-1)/2**. • Distribution des clés (authentification, confidentialité)

Tableau.I.1. Avantages et Inconvénients du chiffrement symétrique

II.B.2. Chiffrement asymétrique (chiffrement à clé publique) [8]:

C'est en 1976, que Whitfield Diffie et Martin Hellman proposaient pour la première fois une toute nouvelle façon de chiffrer, dont le but était de contourner l'écueil de la distribution des clés symétriques, la cryptographie asymétrique était née. Elle est fondée sur l'existence de fonctions mathématiques qui sont à la fois :

- A sens unique : c'est-à-dire qu'il est simple d'appliquer cette fonction à un message, mais extrêmement difficile de le retrouver à partir du moment où on l'a transformé.

- Et à brèche secrète : c'est-à-dire qu'on peut l'inverser (donc retrouver le message en question), uniquement en possédant une information particulière, en d'autres termes une clé.

Le premier algorithme de chiffrement à clé publique a été développé par R.Merckle et M.Hellman en 1977. Il fut vite rendu obsolète grâce aux travaux de Shamir, Zippel et Herlestman, de célèbres cryptanalistes.

En 1978, l'algorithme à clé publique de Rivest, Shamir, et Adelman (d'où son nom RSA) apparaît. Cet algorithme servait encore en 2002 à protéger les codes nucléaires des armées américaine et russe. En plus de ce dernier, on peut citer les algorithmes Diffie-Hellman et ElGamal

Toute la sécurité de ces algorithmes repose sur des problèmes purement mathématiques :

- RSA : Factorisation de grands entiers.
- ElGamal : Logarithme discret.
- Diffie-Hellman : Problème du sac à dos (knapsacks).

II.B.2.a. Principe [13]:

Bob souhaite envoyer des données chiffrées à Alice, ils procéderont ainsi :

1. Alice crée une paire de clés asymétriques : clé privée (qu'elle conserve précieusement), clé publique (qu'elle diffuse librement, notamment à Bob).

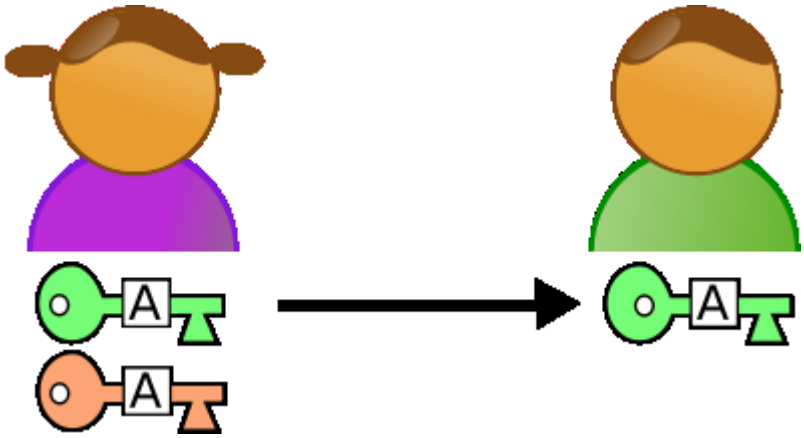


Fig.I.9. Echange de la clé publique

2. Bob chiffre son message avec la clé publique d'Alice.
3. Bob envoie le message chiffré à Alice.
4. Alice reçoit le message chiffré de Bob.
5. Enfin, Alice déchiffre le message avec sa propre clé privée.

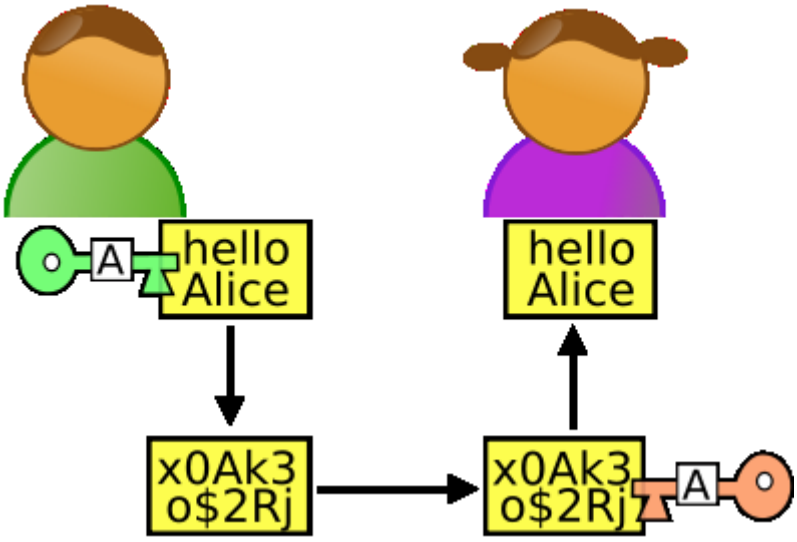


Fig.I.10. Procédure de chiffrement/déchiffrement

Afin de distribuer sa clé publique Alice peut utiliser des serveurs de distribution de clés publiques, ou plus communément par e-mail, SMS, etc.

II.B.2.b. Applications de la cryptographie asymétrique :

II.B.2.b.1. Mécanisme authentification [13]:

Un inconvénient majeur de l'utilisation des mécanismes de chiffrement asymétriques est le fait que la clé publique est distribuée à toutes les personnes : Bob, Oussama, ...

souhaitant s'échanger des données de façon confidentielle. De ce fait, lorsque la personne possédant la clé privée (Alice), déchiffre les données chiffrées, elle n'a aucun moyen de vérifier avec certitude la provenance de ces données (Bob, ou Oussama,...) : on parle de problème d'authentification. Afin de résoudre ce problème, on utilise un mécanisme d'authentification permettant de garantir la provenance des informations chiffrées. Ce mécanisme est aussi fondé sur le chiffrement asymétrique. On l'appel **Signature numérique** (ou signature électronique). Le procédé d'authentification par chiffrement asymétrique est le suivant :

Bob souhaite envoyer des données chiffrées à Alice en lui garantissant qu'il en est l'expéditeur.

1. Bob crée une paire de clés asymétriques : il conserve la clé privée et envoie la clé publique à Alice
2. Alice crée une paire de clés asymétriques : clé privée (qu'elle conserve), clé publique (qu'elle diffuse librement, notamment à Bob)
3. Bob effectue un condensat de son message "en clair" puis chiffre ce condensat avec sa propre clé privée
4. Bob chiffre son message avec la clé publique d'Alice.
5. Bob envoie le message chiffré accompagné du condensat chiffré.
6. Alice reçoit le message chiffré de Bob, accompagné du condensat.
7. Alice déchiffre le message avec sa propre clé privée. À ce stade le message est lisible mais elle ne peut pas être sûre que Bob en est l'expéditeur.
8. Alice déchiffre le condensat avec la clé publique de Bob.
9. Alice utilise la même fonction de hachage sur le texte en clair et compare avec le condensat déchiffré de Bob. Si les deux condensats correspondent, alors Alice peut avoir la certitude que Bob est l'expéditeur. Dans le cas contraire, on peut présager qu'une personne malveillante à tentée d'envoyer un message à Alice en se faisant passer pour Bob.

Cette méthode d'authentification utilise la spécificité des paires de clés asymétriques : si l'on chiffre un message en utilisant la clé publique, alors on peut déchiffrer le message en utilisant la clé privée ; l'inverse est aussi possible : si l'on chiffre en utilisant la clé privée alors on peut déchiffrer en utilisant la clé publique.

Ainsi donc si le condensat reçu par Alice a été chiffré par une clé privée, pour le déchiffrer, elle utilise la clé publique de l'expéditeur présumé. L'utilisation de la clé publique de Bob fait apparaitre le condensat envoyé par Bob.

Si au contraire, ce n'est pas Bob qui a envoyé le message, lorsque la personne malveillante a chiffré le condensat, elle a utilisée sa propre clé privée : pas celle de Bob ! Ainsi donc le déchiffrage avec la clé publique de Bob mènera à un texte erroné et lorsque Alice le comparera à son propre condensat, elle verra qu'ils ne correspondent pas : elle en déduira que Bob n'est pas l'expéditeur mais une autre personne.

- **Condensat et fonction de hachage :**

  Une fonction de hachage (parfois appelée fonction de condensation) est une fonction permettant d'obtenir un et un seul condensat (appelé aussi condensé ou haché ou en anglais message digest) d'un texte, c'est-à-dire une suite de caractères assez courte représentant le texte qu'il condense. D'autre part, il doit s'agir d'une fonction à sens unique (one-way function) afin qu'il soit impossible de retrouver le message original à partir du condensé.

  Ainsi, le haché représente en quelque sorte l'empreinte digitale (en anglais finger print) du document. Les fonctions de hachage les plus utilisés sont :

  MD5 (signifie Message Digest 5) : Développé par Rivest en 1991, MD5 crée une empreinte digitale de 128 bits à partir d'un texte de taille arbitraire en le traitant par blocs de 512 bits (Néanmoins ce dernier est de plus en plus délaissé, car ayant fait l'objet d'attaques fructueuses).

  SHA (pour Secure Hash Algorithm) : Pouvant être traduit par Algorithme de hachage sécurisé, crée des empreintes d'une longueur de 160 bits SHA-1 est une version améliorée de SHA datant de 1994 et produisant une empreinte de 160 bits à partir d'un message d'une longueur maximale de 264 bits en le traitant par blocs de 512 bits.

II.B.2.b.2. Certificats :

Dans un environnement cryptographique asymétrique, il est essentiel de s'assurer que la clé publique avec laquelle on chiffre les données est celle du destinataire concerné et non une contrefaçon.

Supposons maintenant qu'on doit échanger des informations avec des personnes qu'on ne connaît pas, comment savoir qu'on est en possession de la bonne clé ?

La solution est que comme dans la vie courante, on a recours à des certificats. Par exemple pour passer un examen, il faut prouver notre identité par une carte d'identité, passeport ou permis de conduire. Au préalable, ces document ont été certifiés authentiques par un organisme supérieur.

Un certificat électronique est une carte d'identité numérique dont l'objet est d'identifier une entité physique ou non-physique. Le certificat numérique ou électronique est un lien entre l'entité physique (les utilisateurs) et l'entité numérique (les clés publiques). L'autorité de certification fait foi de tiers de confiance et atteste du lien entre les deux entités.

Un certificat numérique contient des données similaires à celles d'un certificat physique. Il contient des informations associées à la clé publique d'une personne, aidant d'autres personnes à vérifier qu'une clé est authentique ou valide. Les certificats numériques permettent de contrecarrer les tentatives de substitution de la clé d'une personne par une autre.

Le standard le plus utilisé pour la création des certificats numériques est le X.509.

X.509 est un standard de cryptographie de l'Union internationale des télécommunications pour les infrastructures à clés publiques (PKI). X.509 établit entre autres les formats standards de certificats électroniques et un algorithme pour la validation de chemin de certification **[14]**.
La Structure d'un certificat est la suivante :

- Un numéro de série;
- L'identification de l'algorithme de signature;
- La désignation de l'autorité de certification émettrice du certificat;
- La période de validité au-delà de laquelle il sera suspendu ou révoqué;
- Les informations sur le titulaire de la clé publique (nom, adresse e-mail, etc.);
- L'identification de l'algorithme de chiffrement (RSA, El Gamal, etc.), et la valeur de la clé publique ;
- Des informations complémentaires optionnelles;
- L'identification de l'algorithme de signature et la valeur de la signature numérique.

Un certificat électronique est géré tout au long de son cycle de vie avec une infrastructure à clés publiques (PKI pour Public Key Infrastructure).

- **Infrastructure à clés publiques [15]:**

Une Infrastructure à clés publiques (ICP) ou Infrastructure de Gestion de Clefs (IGC) ou encore Public Key Infrastructure (PKI), est un ensemble de composants physiques (des ordinateurs, des équipements cryptographiques ou HSM, des cartes à puces), de procédures humaines (vérifications, validation) et de logiciels (système et application) en vue de gérer le cycle de vie des certificats numériques ou certificats électroniques.
Une infrastructure à clés publiques délivre un ensemble de services pour le compte de ses utilisateurs.

En résumé, ces services sont les suivants :

- Enregistrement des utilisateurs (ou équipement informatique),
- Génération de certificats,
- Renouvellement de certificats,
- Révocation de certificats,
- Publication des certificats,
- Publication des listes de révocation (comprenant la liste des certificats révoqués),
- Identification et authentification des utilisateurs (administrateurs ou utilisateurs qui accèdent à l'IGC),
- Archivage, séquestre et recouvrement des certificats (option).

II.B.2.b.3. Transmission sécurisée de la clé symétrique [13]:

Là, est le point qui nous intéresse le plus, et pour le quel est dédié cette étude. La cryptographie asymétrique répond à un besoin majeur de la cryptographie symétrique : le partage sécurisé d'une clé entre deux correspondants, afin de prévenir l'interception de cette clé par une personne tierce non autorisée, et donc la lecture des données chiffrées sans autorisation.

Les mécanismes de chiffrement symétrique étant moins coûteux en temps de calcul, ceux-ci sont privilégiés aux mécanismes de chiffrement asymétrique. Cependant toute utilisation de clé de chiffrement symétrique nécessite que les deux correspondants se partagent cette clé, c'est-à-dire la connaissent avant l'échange. Ceci peut être un problème si

la communication de cette clé s'effectue par l'intermédiaire d'un medium non sécurisé. Afin de pallier cet inconvénient, on utilise un mécanisme de chiffrement asymétrique pour la seule phase d'échange de la clé symétrique, et l'on utilise cette dernière pour tout le chiffrement des données. Ce compromis à pour nom « Cryptographie Hybride ».

II.B.2.c. Avantages et inconvénients [8]:

Avantages	Inconvénients
• Distributions des clés facilitées • Permet de signer des messages facilement • Nombre de clés à distribuer est réduit par rapport aux clés symétriques	• Lents par rapport aux systèmes symétriques (qui sont jusqu'à 1000 fois plus rapides). • Longueur des clés

Tableau.I.2. Avantages et Inconvénients du chiffrement asymétrique

II.B.3. Chiffrement hybride :

Le chiffrement hybride ou plus communément la cryptographie hybride fait appel aux deux grandes familles des systèmes cryptographiques : la cryptographie asymétrique et la cryptographie symétrique. Des logiciels comme PGP et GnuPG reposent sur ce concept qui permet de combiner les avantages des deux systèmes.

La cryptographie asymétrique est intrinsèquement lente de par les calculs complexes qui y sont associés alors que la cryptographie symétrique brille par sa rapidité. Toutefois, cette dernière souffre d'une grave lacune, on doit transmettre les clés de manière sécurisée (sur un canal authentifié). Pour pallier à ce défaut, on recourt à la cryptographie asymétrique qui travaille avec une paire de clés : la clé privée et la clé publique, afin de chiffrer la clé symétrique, et de pouvoir l'acheminer en toute sécurité. Ça, en plus d'assurer des mécanismes d'authentifications et de signatures aux messages envoyés.

Nous verrons en détails, à partir du 2^{éme} chapitre, le fonctionnement d'un système hybride à travers une étude fonctionnelle du logiciel PGP, ainsi que la mise en œuvre d'un logiciel s'appuyant sur le principe de la cryptographie hybride.

II.B.4. La cryptographie quantique [4]:

La cryptographie quantique ne s'appui pas forcement sur un algorithme de chiffrement à proprement parler, elle consiste à chiffrer une clé (suite de bits) en utilisant des photons envoyés par fibre optique (obligatoire). Toute tentative d'interception de la clé ou d'écoute du canal de transmission, avertit immédiatement l'émetteur et le récepteur, puisqu'elle modifie la polarisation des photons, et l'échange se verra immédiatement interrompu.

Cette méthode constitue donc un outil précieux pour des systèmes de cryptographie symétrique où les deux interlocuteurs doivent impérativement posséder la même clé et ce en toute confidentialité. Et là on pense forcement au Masque jetable, où il n'y aura plus de contrainte liée à la taille de la clé à transmettre, du moment qu'avec la fibre optique les débits de transfert plafonnent. Plus de détails sont exposés en annexes.

III. La Cryptanalyse [16] :

La cryptanalyse s'oppose, en quelque sorte, à la cryptographie. En effet, si déchiffrer consiste à retrouver le clair au moyen d'une clé, cryptanalyser (décrypter) c'est tenter de se passer de cette dernière.

Il y a quatre niveaux d'attaques qu'un cryptanalyste peut effectuer. On appelle attaque une tentative de cryptanalyse :

- **Attaque sur texte chiffré seul (ciphertext-only) :** Le cryptanalyste possède des exemplaires chiffrés des messages, il peut faire des hypothèses sur les messages originaux qu'il ne possède pas.

- **Attaque à texte clair connu (known-plaintext attack) :** Le cryptanalyste possède des messages ou des parties de messages en clair ainsi que les versions chiffrées.

- **Attaque à texte clair choisi (chosen-plaintext attack) :** Le cryptanalyste possède des messages en clair, il peut générer les versions chiffrées de ces messages avec l'algorithme que l'on peut dès lors considérer comme une boîte noire.

- **Attaque à texte chiffré choisi (chosen-ciphertext attack) :** Le cryptanalyste possède des messages chiffrés et demande la version en clair de certains de ces messages pour mener l'attaque.

Souvent, on ne s'attaque pas à une version complète de l'algorithme de chiffrement mais à une variante avec moins de tours. Cette analyse préliminaire, si elle permet de déceler des vulnérabilités, laisse entrevoir une attaque sur l'algorithme complet.

Les attaques fructueuses peuvent survenir pour diverses raisons :

- Exploitation des faiblesses structurelles de l'algorithme utilisé.
- Exploitation d'une erreur d'implémentation ou d'utilisation.
- Exploitation des clés faibles.

Ci-dessous, on ne citera que les attaques les plus intéressantes, et surtout les plus efficaces :

- **Cryptanalyse linéaire :**

La cryptanalyse linéaire, due à Mitsuru Matsui, consiste à faire une approximation linéaire de la structure interne de la méthode de chiffrement. Elle remonte à 1993 et s'avère être l'attaque la plus efficace sur DES.

- **Cryptanalyse différentielle :**

La cryptanalyse différentielle est une analyse statistique des changements dans la structure de la méthode de chiffrement après avoir légèrement modifié les entrées. Avec un très grand nombre de perturbations, il est possible d'extraire la clé. Elle est due à Eli Biham et Adi Shamir. Les attaques différentielles sont aussi possibles sur les fonctions de hachage, moyennant des modifications dans la conduite de l'attaque.

- **Cryptanalyse différentielle-linéaire** :

Introduite par Martin Hellman et Langford en 1994, la cryptanalyse différentielle-linéaire combine les deux principes. L'attaque différentielle produit une approximation linéaire de l'algorithme. Ce type de cryptanalyse a été amélioré par Eli Biham en 2002.

- **Attaque par paradoxe des anniversaires :**

Le paradoxe des anniversaires est un résultat probabiliste qui est utilisé dans les attaques contre les fonctions de hachage. Ce paradoxe permet de donner une borne supérieure de résistance aux collisions d'une telle fonction. Cette limite est de l'ordre de la racine de la taille de la sortie, ce qui signifie que, pour un algorithme comme MD5 (empreinte sur 128 bits), trouver une collision quelconque avec 50% de chance nécessite 264 hachages d'entrées distinctes.

Chapitre II

I. Introduction :

Ce chapitre se décompose en deux parties. Dans la première, nous exposeront les fonctionnalités que peuvent offrir les systèmes à base de chiffrement hybride, à travers la présentation du logiciel PGP.

Quand à la deuxième, elle englobera trois études détaillées des systèmes cryptographiques AES et RSA ainsi qu'une fonction de hachage SHA-1, qui seront utilisés dans le développement de notre application, dénommée LCH (Logiciel de Chiffrement Hybride).

II. Présentation d'un cryptosystème hybride : PGP

II.1. Historique de PGP [9]:

PGP (« Pretty Good Privacy », en français : « Plutôt bonne intimité ») est un logiciel cryptographique hybride, inventé par Philip Zimmermann, un analyste-informaticien américain, particulièrement bien adapté à l'utilisation sur Internet. La première version de PGP a vu le jour en 1991.

La version 2.0 de PGP inclut un algorithme développé en Suisse et connu sous le nom d'IDEA (International Data Encryption Algorithm), après que des défauts de sécurité aient été descellés sur la 1ère version.

En 1998, est proposé un standard de l'IETF (Internet Engineering Task Force) nommé OpenPGP et décrit dans la RFC 2440. Il décrit les formats des messages, signatures et clés, qui peuvent être envoyés par des programmes cryptographiques, en se basant sur PGP.

En 2008, PGP Corporation est toujours propriétaire de PGP, et la version actuelle PGP 9.8 n'a plus beaucoup de points communs avec PGP 1.0. D'un système de chiffrement en ligne de commande, PGP est devenu une offre logicielle complexe. Elle se compose toujours du logiciel mais il est désormais muni d'une interface graphique avancée, de l'emploi d'une vaste sélection d'algorithmes cryptographiques (RSA, DSA, DH, 3DES, IDEA, CAST-128, SHA-1) et de diverses possibilités telles que le chiffrement de disque dur, de mails via des plug-ins pour les différents clients de messagerie etc. Des bibliothèques de fonctions sont également

vendues et permettent d'incorporer les fonctionnalités de PGP au sein des développements logiciels.

II.2. Principe de Fonctionnement [8]:

Pour bien comprendre le principe de fonctionnement de PGP, on va se reporter sur le schéma de la *figure II.1*.

PGP utilise une combinaison des fonctionnalités de la cryptographie symétrique et asymétrique. Lorsqu'un utilisateur chiffre un texte avec PGP, les données sont d'abord compressées. Cette compression des données permet de réduire le temps de transmission par tout moyen de communication, d'économiser l'espace disque et surtout, de renforcer la sécurité cryptographique, car la plupart des cryptanalystes exploitent les modèles trouvés dans le texte en clair pour casser le chiffrement. La compression réduit ces modèles dans le texte en clair, améliorant par conséquent considérablement la résistance à la cryptanalyse.

Ensuite, l'opération de chiffrement se fait principalement en deux étapes :

- PGP crée une clé secrète IDEA de manière aléatoire, et chiffre les données avec cette clé (cette clé est aussi appelée « clé de session »)
- PGP crypte la clé secrète IDEA et la transmet au moyen de la clé RSA publique du destinataire.

Remarque [4]:

PGP permet en premier lieu de sécuriser l'envoi d'e-Mail, pour ce faire il doit, souvent, être associé a un client de messagerie électronique, la plupart de ces derniers n'acceptent que l'utilisation de blocs ASCII. Or, PGP fournit des données binaires. Ainsi, tout ou une partie du message consistera en une suite arbitraire d'octets. Par conséquent PGP doit coder les données binaires en caractères ASCII imprimables. Pour ce faire il utilise l'algorithme Radix-64 qui fait correspondre 3 octet à 4 caractères imprimables. Puis PGP procèdera également à la segmentation des messages. Car la taille des mails est souvent limitée (p. ex. à une longueur de message maximale de 50.000 octets). Ainsi, les messages plus longs doivent être découpés en

segments. PGP va automatiquement subdiviser un message trop grand. En réception, il faudra enlever tous les en-têtes du mail et rassembler tous les blocs en un seul.

L'opération de décryptage se fait également en deux étapes :

- PGP déchiffre la clé secrète IDEA au moyen de la clé RSA privée.
- PGP déchiffre les données avec la clé secrète IDEA précédemment obtenue.

Cette méthode de chiffrement associe la facilité d'utilisation du cryptage de clef publique à la vitesse du cryptage symétrique. Le chiffrement conventionnel est environ 1000 fois plus rapide que les algorithmes de chiffrement à clé publique. Par contre, le chiffrement à clé publique résous le problème de la distribution des clés. Utilisées conjointement, ces deux méthodes améliorent la performance et la gestion des clefs, sans pour autant compromettre la sécurité.

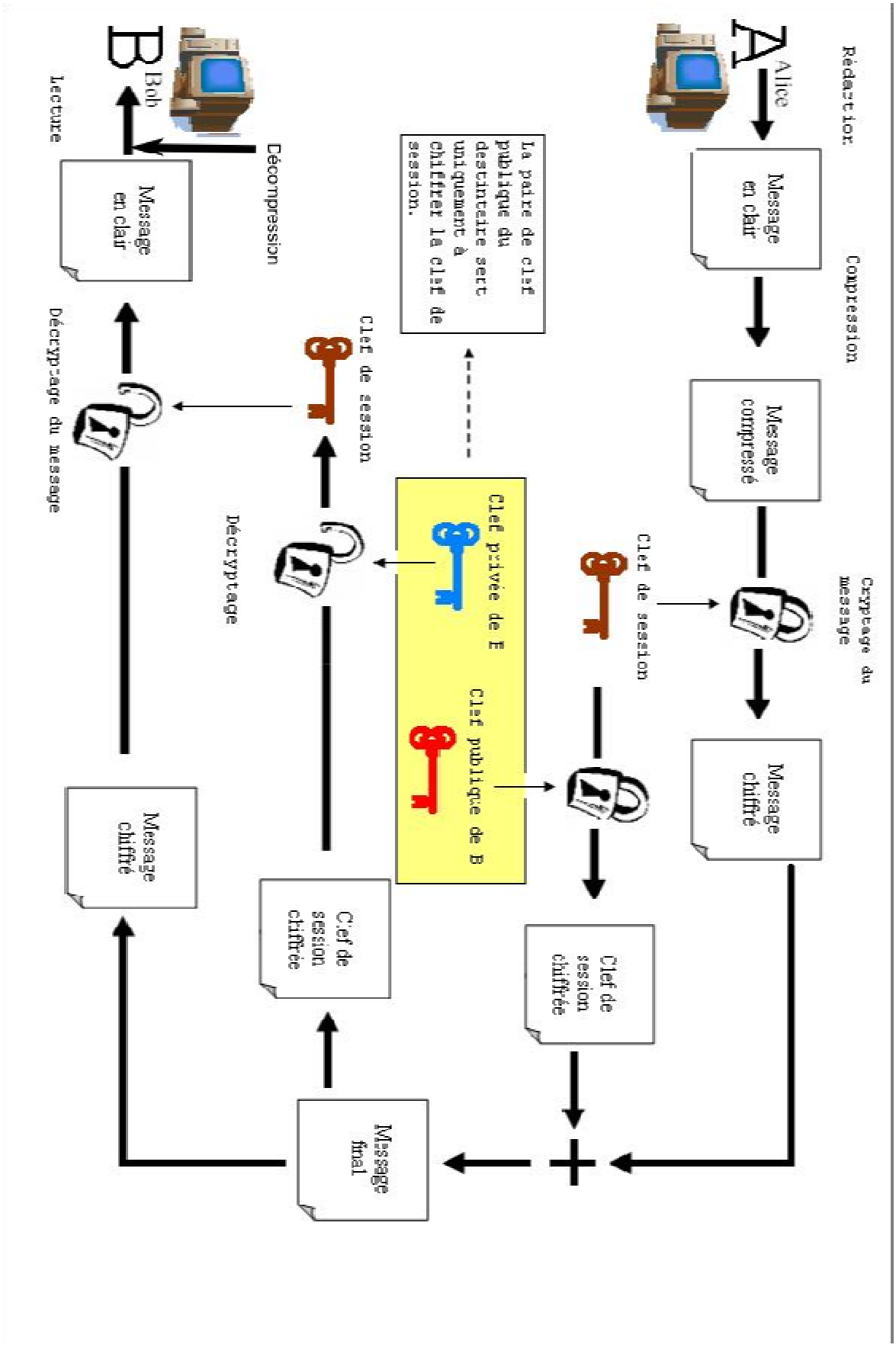


Fig.II.1. Synoptique de fonctionnement du PGP

II.3. Avantages du PGP [9] :

- La rapidité : le message est chiffré par un cryptage symétrique. La clé privée est chiffrée de façon asymétrique. Toutefois le volume de données que représente cette clé est négligeable par rapport au volume de données que représente le message. Par conséquent, le temps de chiffrage global est proche de celui d'un système symétrique.
- Une plus haute sécurité qu'un système à clef symétrique. Dans un système à clé privée standard, le canal d'échange de la clé est le point faible du système. Dans PGP en revanche, la clé utilisée pour chiffrer le message est acheminée en la chiffrant avec un système cryptographique asymétrique. Et pour plus de sécurité, cette clé est nouvelle pour chaque message. Ce qui implique que pour effectuer une attaque il est nécessaire de casser au choix :
    - Autant de clés privées que de messages.
    - Le système de clés RSA.

Ce qui rend finalement PGP plus sur qu'un système à clé privée classique.

II.4. Fonctionnalités du PGP :

II.4.1. La signature numérique des données [9]:

Les signatures numériques sont le seul moyen de s'assurer de l'authenticité des données reçus, ainsi que de leur intégrité. PGP hérite des nombreuses fonctionnalités qu'offrent les systèmes cryptographiques asymétriques puisqu'il en intègre un, dans son processus de chiffrement.

En matière de signature des données, PGP utilise ce qu'on appel un scellement de données. Il applique une fonction de hachage au texte en clair à signer. Puis le condensé obtenu, de taille fixe, est signé (fonctionnalité assurée par le système à clé publique) avec la clef privée de l'expéditeur. Le sceau ainsi obtenu est joint au texte en clair.

En expédiant le message accompagné de son sceau, on garantir l'intégrité d'un message, c'est-à-dire que le destinataire peut vérifier que le message n'a pas été altéré (intentionnellement ou de manière fortuite) durant le transfert.

A la réception du message, le destinataire procédera de la sorte :

1. Applique la fonction de hachage au texte en clair,
2. Utilise la clef publique de l'expéditeur pour retrouver la valeur du condensé joint au texte en clair
3. Compare les deux condensés.

Si le message a été falsifié durant la communication, alors les deux condensés ne correspondront pas.

La méthode de création des signatures numériques est illustrée dans la figure suivante :

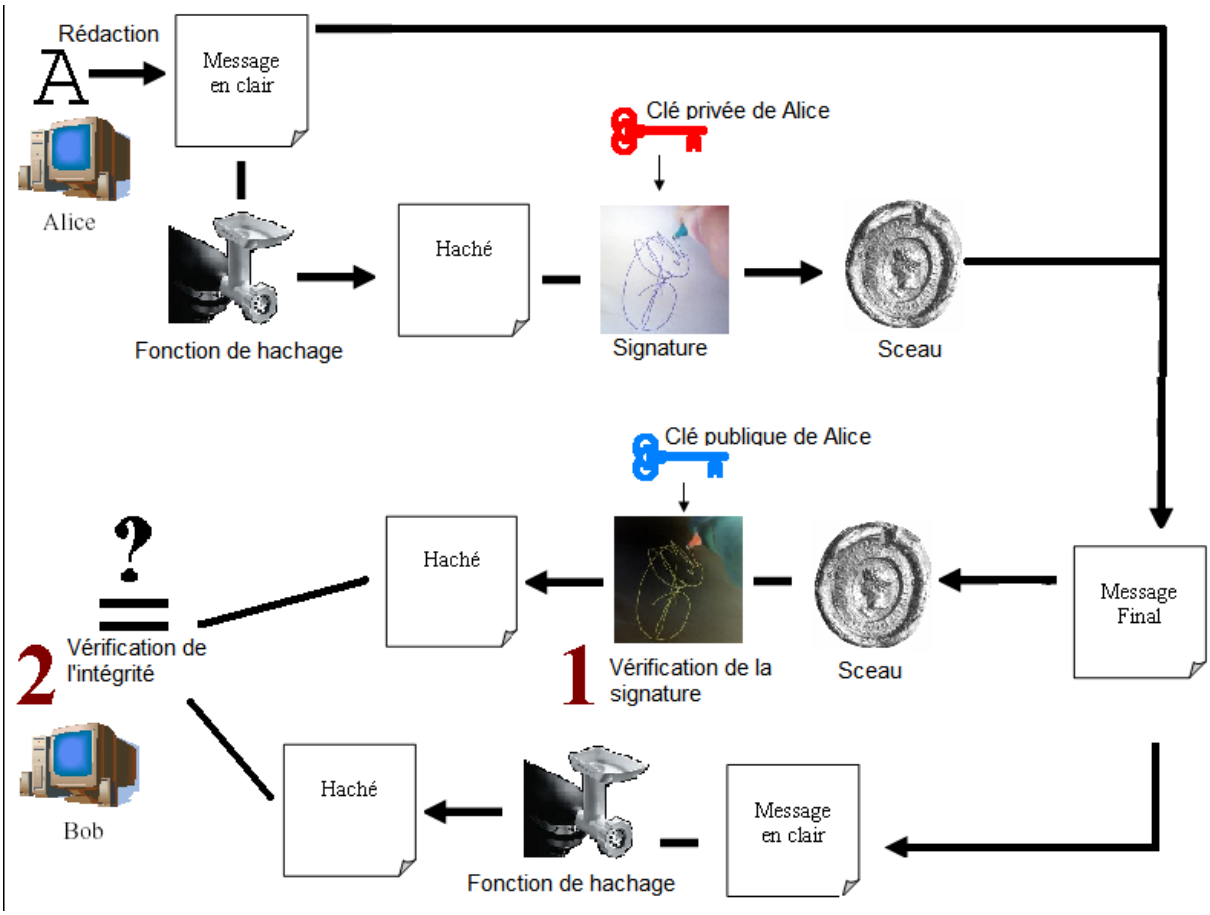


Fig.II.2. Signature de données (PGP)

II.4.2. Gestion des clés [4]:

Pour une sécurité optimale, il est nécessaire de générer, de façon aléatoire, une clé de session (clés symétriques) différente pour chaque opération de chiffrement.

Et puisque on est totalement libres de disposer de plusieurs paires de clés publique/privée. Il est donc obligatoire de conserver la liste de ses propres paires de clés mais également la liste des clés publiques de ses correspondants.

II.4.2.1. Clés de session :

On a besoin d'une clé de session pour chaque message. Les tailles varieront selon l'algorithme utilisé : DES 56 bits, CAST ou IDEA 128 bits, Triple-DES 168 bits.

Ces clés doivent êtres les plus aléatoires possibles. Pour introduire la notion aléatoire de la clé, PGP utilise des données issues des utilisations précédentes et de la dynamique de frappe de l'utilisateur (temps pris pour taper une séquence de lettres, le temps de pression sur une touche ou encore le temps entre deux pressions successives sur une même touche).

II.4.2.2. Trousseaux de clés :

Chaque utilisateur PGP dispose d'une paire de trousseaux :

- Le trousseau de clés publiques contient toutes les clés publiques des autres utilisateurs de PGP connus de cet utilisateur, classées par identifiant de clé.
- Le trousseau de clés privées contient les paires de clés publiques/privées de cet utilisateur, classées par identifiant de clé.

Remarque :

La clé privée n'est pas conservée en tant que telle mais chiffrée (avec CAST ou IDEA ou 3DES) au moyen d'un condensé (SHA-1) obtenu à partir d'une passphrase selon la méthode suivante :

1. L'utilisateur sélectionne une passphrase, un simple mot de passe n'étant plus suffisant pour garantir une bonne sécurité (la différence entre une passphrase et un mot de passe classique, est que ce dernier n'admet ni les caractères spéciaux ni les espaces, au contraire, bien évidement, de la passphrase).

2. Quand le système génère une nouvelle paire de clés RSA, il demande à l'utilisateur sa passphrase. En utilisant le SHA-1, il génère un condensé de 160 bits à partir de cette passphrase et la passphrase est effacée.

3. Le système chiffre la clé privée avec CAST (par exemple). Le condensé est ensuite supprimé et la clé privée chiffrée est stockée dans l'anneau des clés privée. C'est la raison pour laquelle PGP demande la passphrase lorsque l'on désire obtenir sa clé privée.

Les trousseaux de clés interviennent de la manière suivante lors de l'envoi/réception d'un message :

Dans un premier temps, on procède à la signature du message :

- Obtention de la clé privée dans le trousseau de clés privées
- Utilisation de la passphrase pour déchiffrer la clé
- Construction du composant signature

Ensuite, on procèdera au chiffrement du message :

- Génération d'une clé de session et chiffrement du message
- Obtention de la clé publique du correspondant dans le trousseau de clés publiques
- Scellement de la clé de session

Lors de la réception, le déchiffrement du message, se fera dans l'ordre suivant:

- Récupération de la clé privée dans l'anneau de clés privées (obtention de l'ID de la clé dans le composant de la clé de session)
- Utilisation de la passphrase pour déchiffrer la clé
- Récupération de la clé de session et déchiffrement du message

Authentification du message :

- Récupération de la clé publique de l'expéditeur dans l'anneau de clés publiques (utilisant l'ID de la clé dans le composant de la signature)
- Récupération du condensé transmis
- Calcul du condensé du message reçu et comparaison.

II.4.3. Les certificats :

Comme mentionné dans le premier chapitre, les systèmes à clé asymétrique, sont sensibles à l'attaque dite de « l'homme au milieu »,

Pour éviter cela, on crée des certificats numériques dont le but est d'apporter la preuve de la validité d'une clé et de son appartenance à un propriétaire donné. En règle générale, une autorité appelée « autorité de certification » est seule habilitée à produire des certificats et les signer à l'aide de sa clé privée.

Dans PGP le concept est différent, ce dernier adopte, pour les certifications numériques, un protocole qui ne se base pas sur une autorité de certification centralisée, mais sur un réseau de confiance appelé « La Toile de Confiance » de PGP. En d'autres termes, chaque utilisateur est une autorité de certification dans PGP.

Il est possible à chaque personne de signer avec sa clef privée un certificat donnée qu'il juge digne de confidence. Contrairement à un système centralisé, un certificat PGP pourra contenir une multitude de signatures numériques.

II.4.3.1. Format d'un certificat PGP [8]:

Le format d'un certificat PGP comprend entre autres les informations suivantes :

- **Le numéro de version de PGP** : identifie la version de PGP utilisée pour créer la clef associée au certificat.
- **La clef publique du détenteur du certificat** : partie publique de votre paire de clefs associée à l'algorithme de la clef, qu'il soit RSA, DH (Diffie-Hellman) ou DSA (Algorithme de signature numérique).
- **Les informations du détenteur du certificat** : il s'agit des informations portant sur l'« identité » de l'utilisateur, telles que son nom, son ID utilisateur, sa photographie, etc.
- **La signature numérique du détenteur du certificat** : également appelée auto-signature, il s'agit de la signature effectuée avec la clef privée correspondant à la clef publique associée au certificat (authenticité/integrité).
- **La période de validité du certificat** : dates/ heures de début et d'expiration du certificat. Indique la date d'expiration du certificat.
- **L'algorithme de chiffrement symétrique** préféré pour la clef : indique l'algorithme de chiffrement que le détenteur du certificat préfère appliquer au cryptage des informations. Les algorithmes pris en charge sont CAST, IDEA ou DES triple
- **Les éventuelles signatures** effectuées par d'autres utilisateurs**.**

II.4.3.2. Les niveaux de confiance [4]:

Avec le système des certificats à signatures multiples, propre à PGP, se pose le problème de déterminer si les signatures apportées aux certificats sont dignes de confiance.

Pour pallier cet inconvénient, un système basé sur les niveaux de confiance et de validité est mis en place.

De base, il existe dans PGP trois niveaux de validité : valide, semi-valide et invalide Ainsi que trois niveaux de confiance : confiance totale, confiance moyenne, aucune confiance. Lorsque l'utilisateur génère sa paire de clés dans PGP, celle-ci se voit implicitement attribuer les niveaux maximums pour ces deux domaines.

En résumé, pour qu'une clé soit validée dans le système PGP de l'utilisateur, elle devra avoir reçu soit :

- La signature de ce dernier,
- La signature d'une clé à laquelle une confiance totale aura été accordée,
- La signature d'au moins deux clefs auxquelles une confiance moyenne aura été apportée.

Sur l'ordinateur de l'utilisateur, les indicateurs de confiance sont inclus dans le trousseau des clés publiques.
Voici en détails les modèles de fiabilité de PGP :

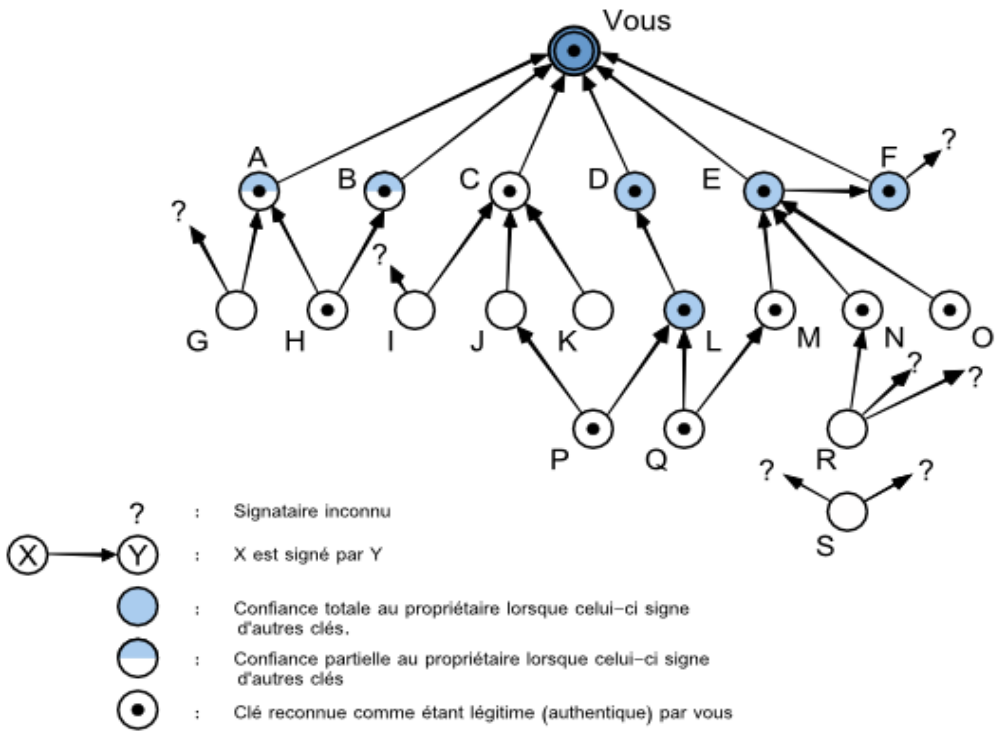


Fig.II.3. Toile de confiance de PGP

- Vous signez toutes les clés appartenant à des utilisateurs à qui vous faites confiance (ou partiellement confiance) à l'exception du nœud L. Ces signatures personnelles ne sont pas obligatoires (comme on le voit avec L), mais sont souhaitables.
- On le remarque avec le nœud E, qui est signé par F à qui vous faites confiance. Dans la majorité des cas, il sera préférable de "confirmer" la signature de F par une signature vous appartenant. Ainsi, il serait préférable que vous signiez la clé de L.

- On considère que deux utilisateurs à qui vous faites partiellement confiance sont suffisants pour certifier la clé d'un autre utilisateur. Ainsi, la clé de H est certifiée valide par la présence des signatures de A et de B, utilisateurs auxquels vous faites partiellement confiance.

- Il arrive que certaines clés soient certifiées, alors que vous n'avez pas confiance leur propriétaires. C'est le cas pour l'utilisateur N. Celui-ci est signé par E, sa clé est donc certifiée valide. Mais bien que vous ayez confiance en E, cette confiance n'est pas transitive. Pour l'utilisateur R, bien que signé par N dont la clé est validée, PGP ne le considère pas comme digne de confiance. Cette situation est tout à fait plausible : il peut arriver que vous souhaitiez envoyer des données chiffrées à un utilisateur quelconque en qui vous n'avez pas confiance, cependant, la clé utilisée pour chiffrer doit réellement être la clé de ce correspondant.

- Le nœud S est signé par deux utilisateurs inconnus (un tel cas peut avoir lieu si cette clé est obtenue à partir d'un serveur de clés.) PGP ne considère pas cette clé comme valide, même si elle provient d'un serveur certifié valide.

II.4.4. La révocation :

Un certificat possède une durée de validité. A l'issue de celle–ci, la clé de ce certificat n'est plus considérée comme valide. Il peut toutefois être nécessaire d'invalider une clé avant la fin de son certificat : si celle-ci a été cassée ou le mot de passe lié à cette clé a été perdu.

Dans X.509, révoquer sa signature sur un certificat consiste à enlever sa signature d'un certificat d'authenticité, c'est-à-dire à indiquer qu'on n'apporte plus son crédit à ce certificat.

La révocation PGP va plus loin en apportant la possibilité d'invalider entièrement un certificat (et plus seulement d'en enlever sa signature). Toutefois, une telle révocation n'est possible que pour :

- Le propriétaire de ce certificat
- Une personne considérée par ce propriétaire comme une autorité de certification, c'est à dire une personne à laquelle il a attribué un niveau de confiance totale.

III. Mise en œuvre du Logiciel de Chiffrement Hybride (LCH) :

III.A. Introduction :

Maintenant que la présentation de PGP est terminée, nous avons pu avoir une idée précise de ce qu'un système cryptographique hybride complet pouvait offrir comme fonctionnalités, un tel système offre aux données sensibles des niveaux de confidentialité, d'authentification, d'intégrité, très hauts. Nous nous sommes fixés comme but, la conception d'un logiciel se basant sur le concept de cryptographie hybride, assurant avant tout la confidentialité de messages et de fichiers avec formats quelconques.

III.B. Choix des algorithmes :

III.B.1. Cryptosystème symétrique :

Notre choix s'est porté sur l'AES (Advanced Encryption Standard). Au delà du fait que techniquement il présente des caractéristiques solides, avec notamment un très haut niveau de sécurité (résiste à toute forme de cryptanalyse), son étude est facilitée par la présence sur internet d'une documentation très riche et variée, expliquant d'une façon approfondie jusqu'aux moindres détails le fonctionnement de l'algorithme.

III.B.2. Cryptosystème asymétrique :

Là aussi, notre choix s'est porté sur un algorithme qui a une très bonne réputation, le RSA. C'est un algorithme qui est très largement répandu (d'ailleurs utilisé dans PGP dès ses débuts). Le principe de chiffrement repose intégralement sur la difficulté (à l'heure actuelle) de factoriser des grands nombres entiers (au moins 100 chiffres).

III.B.3. Fonction de Hachage :

Pour la génération de condensats, on utilisera le SHA-1 (Secure Hash Algorithm), qui est une fonction de hachage cryptographique conçue par la National Security Agency des États-Unis (NSA), et publiée par le gouvernement des États-Unis comme un standard fédéral de traitement de l'information. Elle produit un condensat de 160 bits **[11]**.

III.C. Etude détaillée des différents algorithmes choisis :

Nous détaillerons ci-dessous le fonctionnement de chaque algorithme choisis, mais juste avant nous allons présenter un tableau récapitulatif sur la tâche allouée à chaque algorithme au sein du LCH.

Systèmes	Tâches
AES	Chiffrement/Déchiffrement des données.
RSA	Chiffrement/Déchiffrement des clés symétriques (clés AES), Signatures.
SHA-1	Génération de condensés.

Tab.II.1. Récapitulatif sur les algorithmes choisis

III.C.1. AES (Advanced Encryption Standard) :

Le standard de chiffrement avancé (*Advanced Encryption Standard* ou *AES*) est un algorithme de chiffrement symétrique, choisi en octobre 2000 par le NIST (National Institute of Standards and Technology) afin être le nouveau standard de chiffrement pour les organisations du gouvernement des États-Unis, après que le DES, standard depuis les années 70, devenait obsolète **[17]**.

III.C.1.a. Caractéristiques :

L'AES est un algorithme de chiffrement symétrique, donc qui utilise la même clé pour chiffrer et déchiffrer les données, cette clé peut avoir au choix : 128, 192 ou 256 bits de longueur, il travail uniquement qu'avec des blocs de 128bits, il est compatible avec les modes de chiffrement par blocs : ECB et CBC.

III.C.1.b. Origines : La consécration de Rijndael

Il est issu d'un appel à candidatures international lancé en janvier 1997 et ayant reçu 15 propositions. Parmi ces 15 algorithmes, 5 furent choisis pour une évaluation plus poussée

en avril 1999 : MARS, RC6, Rijndael, Serpent, et Twofish. Au bout de cette évaluation, ce fut finalement le candidat Rijndael, du nom de ses deux concepteurs Joan Daemen et Vincent Rijmen (tous les deux de nationalité belge) qui a été choisi. AES est un sous-ensemble de Rijndael : Tandis que ce dernier opère sur des blocs de données de 128, 192 ou 256 bits, en utilisant des clés de chiffrement de 128, 192 ou 256 bits. L'AES s'est limité qu'à des blocs de 128 bits, pour des raisons de commodité dit-on **[17]**.

L'AES a été adopté par le NIST en 2001. De plus, son utilisation est très pratique car il consomme peu de mémoire et n'étant pas basé sur un schéma de Feistel, sa complexité est moindre et il est plus facile à implémenter.

III.C.1.c. Notations, structure des données [2]:

Avant d'entrer dans le vif du sujet, et dans un souci de conformité au standard, voici les conventions de notation issues de la spécification du NIST [FIPS-197]:

- **KeyExpansion :** Routine dont le but est de générer les clés de tour (Round Key) à partir de la clé de chiffrement.
    - **SubWord()** : Fonction utilisée par la routine d'expansion de la clé qui prend un mot en entrée et applique à ses 4 octets une substitution non linéaire.
    - **RotWord()** : Fonction utilisée par la routine d'expansion de la clé qui applique à un mot de 4 octets une permutation circulaire.

- **AddRoundKey()** : Transformation (pour le chiffrement et le déchiffrement) qui ajoute une clé de tour (Round Key) au bloc courant.

- **SubBytes()** : Transformation qui opère une substitution non linéaire de chaque octet du bloc en utilisant une table (S-box) lors du chiffrement.

- **InvSubBytes()** : Transformation du déchiffrement qui est l'inverse de la transformation SubBytes().

- **ShiftRows()** : Transformation qui applique des permutations circulaires aux trois dernières lignes du bloc lors du chiffrement.

- **InvShiftRows()** : Transformation du déchiffrement qui est l'inverse de la transformation ShiftRows().

- **MixColumns()** : Transformation qui permute les colonnes d'un bloc lors du chiffrement.

- **InvMixColumns()** : Transformation du déchiffrement qui est l'inverse de la transformation MixColumns().

- **K ou Key** : La clé de chiffrement.

- **Nk** : Nombre de mots de 32 bits dans la clé de chiffrement. Pour l'AES Nk= 4,6 ou 8.

- **State :** Matrice de taille 4xNb, servant à stocker le bloc d'entrée. On y applique les transformations intermédiaires.

- **Nb** : Nombre de colonnes de la matrice **state** (Nombre de mots de 32 bits). Pour l'AES Nb=4.
- **Nr** : Nombre de tours (rondes), en fonction de **Nb** et **Nk**.

	Nk	Nb	Nr
K = 128 bits	4	4	10
K = 192 bits	6	4	12
K = 256 bits	8	4	14

Tab.II.2. Relation entre Nr et (Nb,Nk)

- **Rcon[]** : Table constante.

- **GF(2^8)** : Corps fini à 256 éléments (Galois Fields ou Champs de Galois).
  Un corps est un anneau où tous les éléments non nuls sont inversibles
  (Exemples de corps : l'ensemble des réels, l'ensemble des complexes).

- « • » : Multiplication entre deux éléments de **GF(2^8)**

Soit une suite de 16 octets (128 bits) représentant un bloc d'entrée, telle que :

$$\{in_0, in_1, in_2, in_3, in_4, in_5, in_6, in_7, in_8, in_9, in_{10}, in_{11}, in_{12}, in_{13}, in_{14}, in_{15}\}$$

Ce bloc sera copié dans la matrice STATE au début du chiffrement et du déchiffrement, tout en respectant la convention illustrée par la figure suivante :

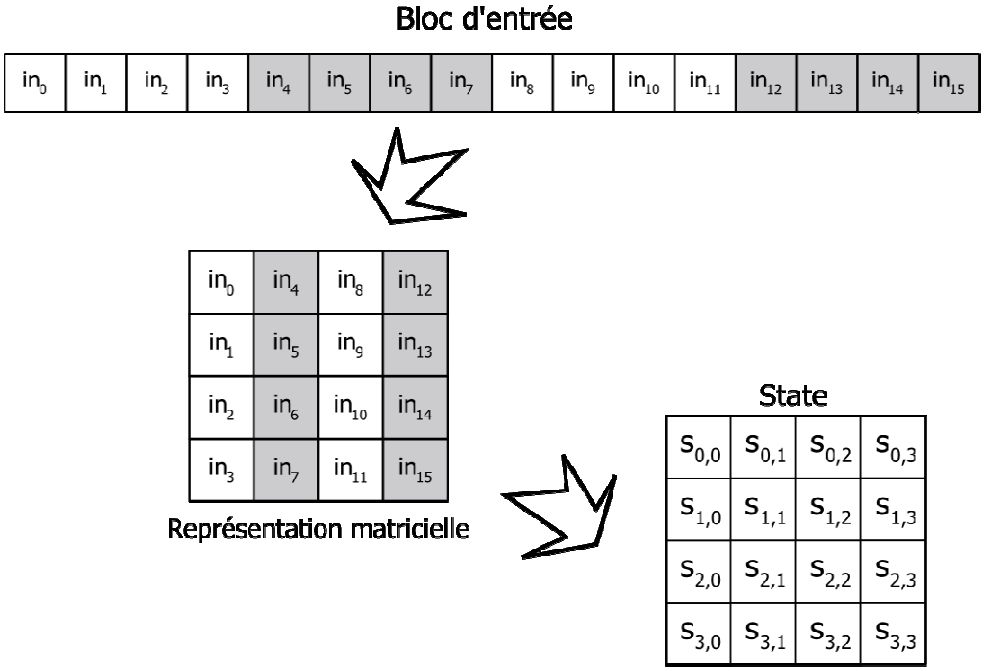


Fig.II.5. Convention de la mise en place de State

III.C.1.d. Préliminaires mathématiques [2]:

Tous les calculs établis dans l'AES, le sont dans le corps fini à 256 éléments **GF(2^8)**. L'algorithme de l'AES est orienté octets, c'est-à-dire qu'a bas niveau celui-ci traite les données par suite de 8 bits.

Tous les octets traités dans l'AES sont interprétés comme des éléments du corps fini à 256 éléments **GF(2^8)**. Où un octet *b*, composé des 8 bits b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0 (b_0 bit de poids le plus faible), est assimilé à un polynôme de degré ≤ 7 avec des coefficients dans {0, 1}. *b* est représenté comme suit : $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b$

Ces éléments peuvent-être additionnés et multipliés. Néanmoins, dans **GF(2^8)** ces opérations sont différentes de celles appliquées pour les nombres. Voyons leur mise en œuvre:

III.C.1.d.1. L'addition :

L'addition de deux éléments dans **GF(2^8)**, n'est autre que l'addition des coefficients des deux polynômes correspondants aux deux éléments. Cette addition correspond a l'opération XOR (notée ⊕).

Exemple :

$(x6 + x4 + x2 + x + 1) \oplus (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ (notation polynomiale);
$\{01010111\} \oplus \{10000011\} = \{11010100\}$ (notation binaire);
$\{57\} \oplus \{83\} = \{d4\}$ (notation hexadécimale).

III.C.1.d.2. La multiplication :

La multiplication dans **GF(2^8)** (notée « • ») correspond à la multiplication polynomiale conventionnelle, modulo un polynôme irréductible de degré 8. Un polynôme est dit irréductible sur un corps fini, ssi il n'admet pas de factorisation sur ce même corps. Pour l'AES ce polynôme irréductible est : $m(x) = x^8 + x^4 + x^3 + x + 1$ (0x11B en base hexadécimale).

Exemple : $\{57\} \bullet \{83\} = \{c1\}$, car :

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x +$$
$$x^6 + x^4 + x^2 + x + 1$$
$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

Et : $x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$ modulo $(x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$.

La réduction modulo $m(x)$ assure que le résultat de la multiplication soit un polynôme de degré < 8, afin de rester dans le domaine des octets.

Notons aussi que la multiplication par ***x*** (soit 00000010), revient à décaler l'octet d'un bit sur la gauche et à le réduire modulo $m(x)$.

III.C.1.d.3. Calcul de l'inverse :

Étant donné qu'on a une structure de corps, tout élément non nul est inversible. L'inverse d'un élément b(X) se trouve par l'algorithme de Bézout (l'algorithme d'Euclide étendu). A noté que l'inverse de l'élément 0x00 est lui-même.

III.C.1.e. Principe de Fonctionnement :

III.C.1.e.1. Chiffrement :

Voici l'organigramme représentant le chiffrement dans l'AES :

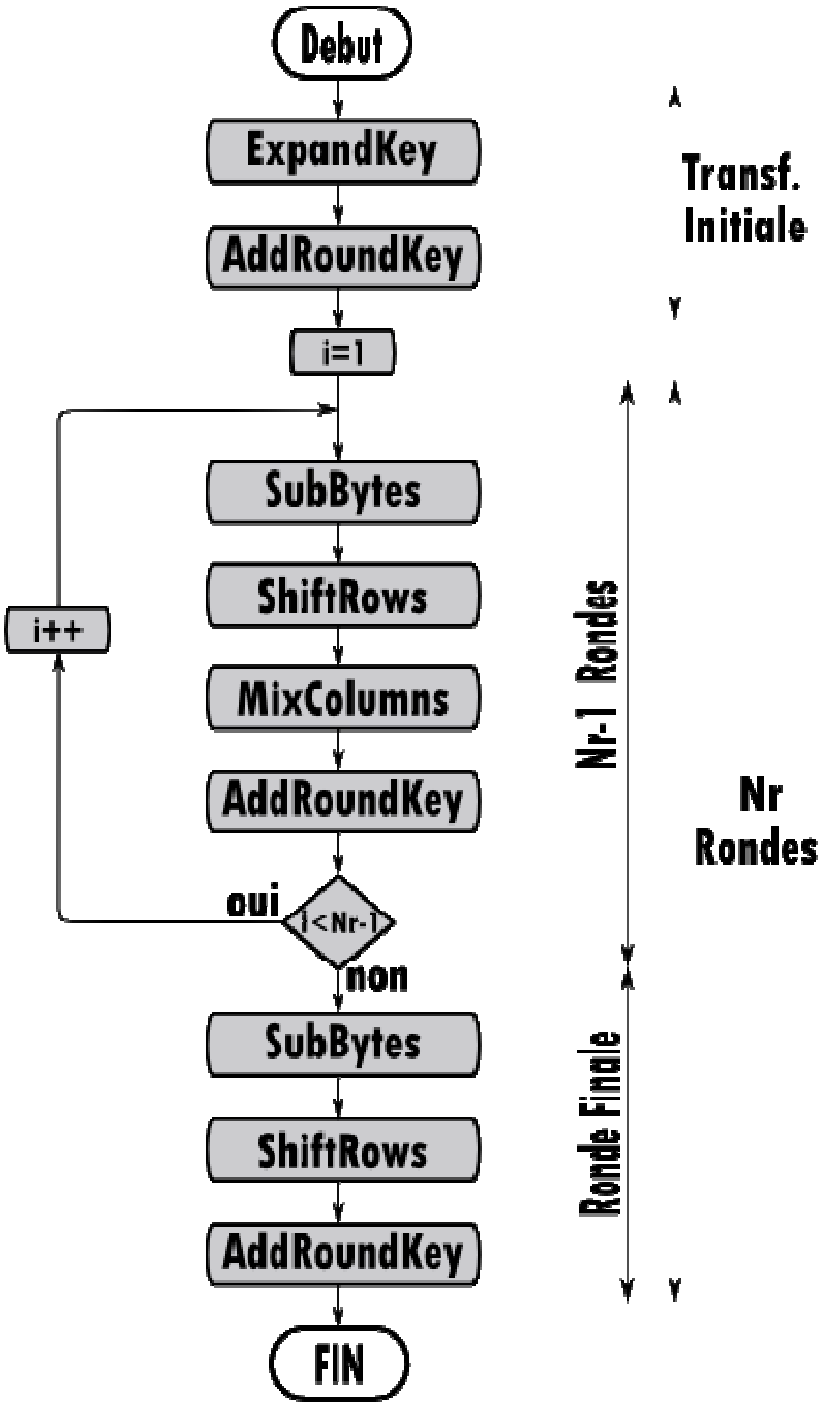


Fig.II.6. Organigramme de chiffrement

Il peut être interprété de la manière suivante :

1. On calcule la clé étendue (ExpandKey)
2. On effectue un AddRoundKey(State, Key). (Ronde initiale)
3. On effectue Nr-1 rondes :
    - SubBytes(State);
    - ShiftRows(State);
    - MixColumns(State);
    - AddRoundKey(State, Clé de tour);

4. On effectue une ronde finale :
    - SubBytes(State);
    - ShiftRows(State);
    - AddRoundKey(State, Clé de tour);

Le bloc chiffré est ensuite envoyé vers la sortie, puis réinitialisé avec la suite des données.
Voyons en détails le fonctionnement de chaque étape :

III.C.1.e.1.1. L'étape SubBytes :

Cette procédure est la seule transformation qui ne soit pas linéaire dans l'algorithme. C'est donc grâce à celle-ci que le système est résistant.
Dans cette étape, chaque élément de la matrice State est permuté selon une table de substitution inversible, à valeurs pré-calculées, notée S-Box (Table-S). Elle contribue à la « confusion » (Shannon). Dans l'AES elle est formée de 16x16 éléments codés sur 8 bits (un octet). Elle est illustrée dans la figure suivante :

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig.II.7. La table S-Box

La substitution se fait de la manière suivante :

La S-Box est indexée par les 4 bits de poids fort et les 4 bits de poids faible de l'octet. Telle que, le premier caractère hexadécimal de chaque élément de la matrice State (chaque octet) indique une ligne de la S-Box tandis que le deuxième indique une colonne. L'élément se trouvant à l'intersection ligne-colonne dans la matrice S-Box est donc celui qui doit être substitué à celui de la matrice State.

En gros cela se passe comme sur la figure suivante :

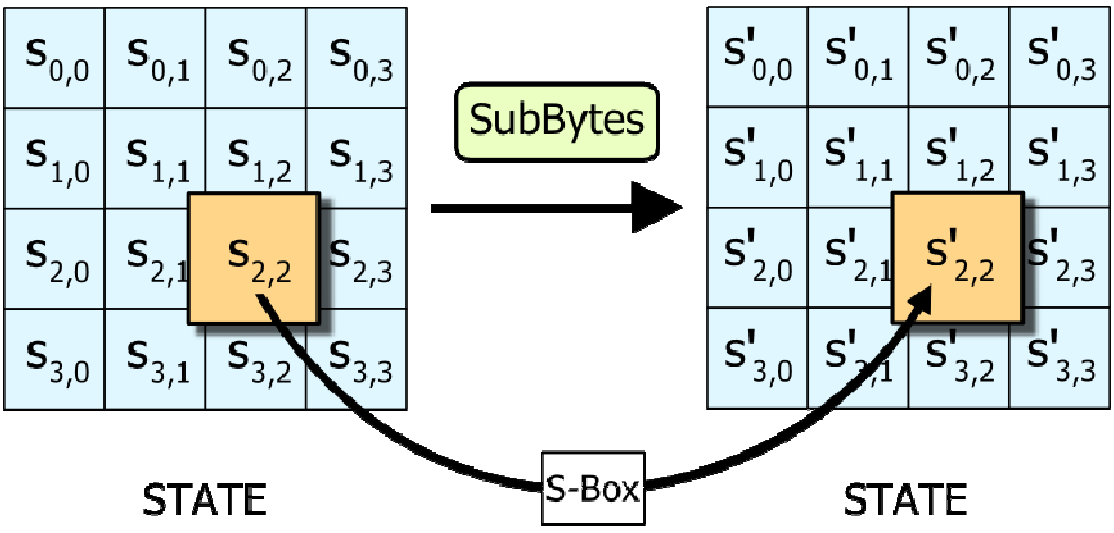


Fig.II.8. L'étapes SubBytes

- **Construction de la table [2]:**

Cette table est composée d'éléments pré-calculés, sa construction se fait selon la transformation suivante :

Pour chaque élément « a » de **GF(2^8)** on a : ***SBox[a] = $\mathcal{F}$(t(a))*** **[3]**

Avec :

1. **t** : a ⟶ a^{-1}, est une transformation inverse dans **GF(2^8).**
2. $\mathcal{F}$ une transformation affine inversible, définie par (forme matricielle simplifiée):

$$a' = \mathcal{F}(a) \Leftrightarrow \begin{bmatrix} a'0 \\ a'1 \\ a'2 \\ a'3 \\ a'4 \\ a'5 \\ a'6 \\ a'7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a0 \\ a1 \\ a2 \\ a3 \\ a4 \\ a5 \\ a6 \\ a7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \textbf{[2]}$$

III.C.1.e.1.2. L'étape Shiftrows :

L'étape Shiftrows opère sur les lignes de la matrice State, et effectue pour chaque ligne un décalage cyclique de n éléments vers la gauche. Le décalage n dépend de la ligne considérée de la manière suivante :

- Pour la 1ere ligne aucun decalage n'est opéré.
- Pour la 2eme ligne un decalage d'une case vers la gauche est opéré.
- Pour la 3eme ligne un decalage de 2 cases vers la gauche est opéré.
- Pour la 4eme ligne un decalage de 3 cases vers la gauche est opéré.

La figure suivante illustre cette transformation :

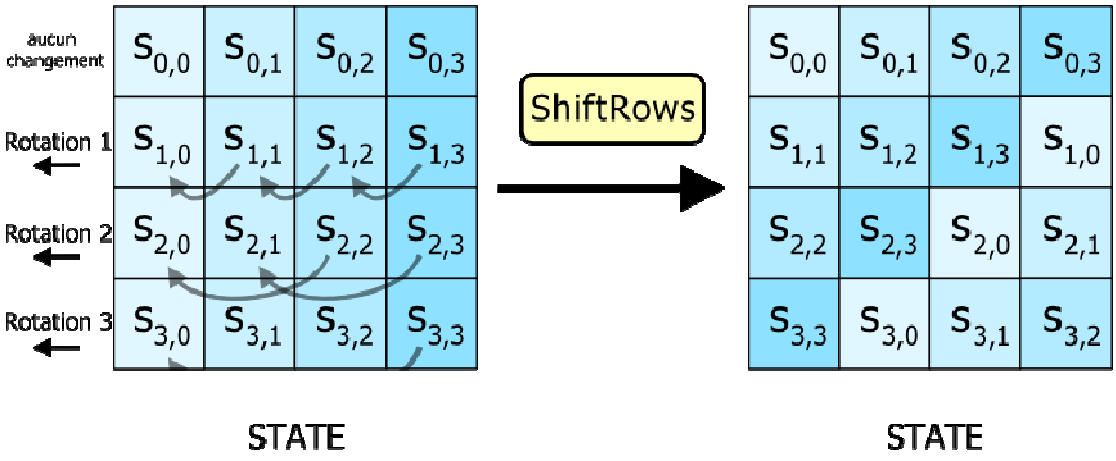


Fig.II.9. L'étapes ShiftRows

III.C.1.e.1.3. L'étape MixColumns [2]:

Cette transformation consiste à effectuer pour chaque colonne une multiplication (dans **GF(2^8)**) par un polynôme noté *c(x)* fixé, suivi d'une réduction modulo le polynôme x^4+1.

$$c(x) \bullet s(x) \bmod(x^4 + 1)$$

avec : $c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$

Cette opération est notée "⊗". Sa représentation matricielle s'écrit :

$$s'(x) = c(x) \otimes s(x) \quad \Leftrightarrow \quad \begin{bmatrix} S'(0,n) \\ S'(1,n) \\ S'(2,n) \\ S'(3,n) \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} S(0,n) \\ S(1,n) \\ S(2,n) \\ S(3,n) \end{bmatrix}; \quad 0 \leq n < Nb$$

La figure suivante illustre cette étape.

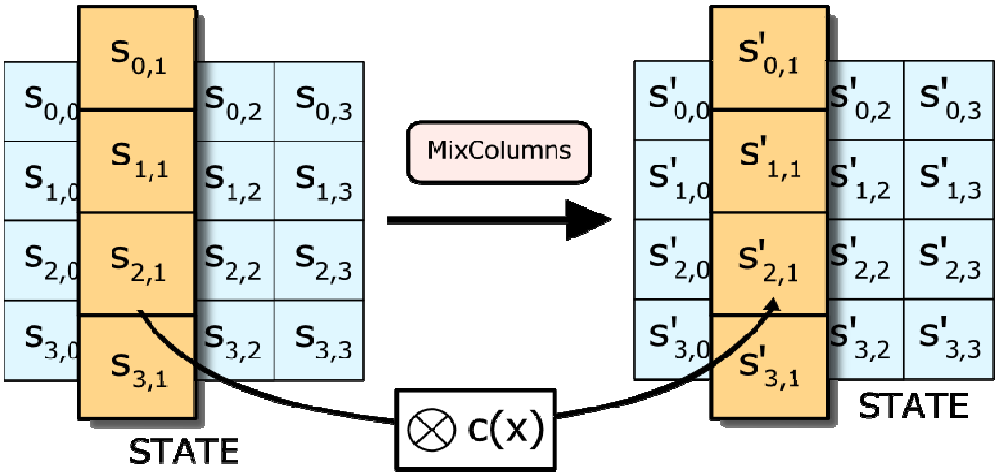


Fig.II.10. L'étapes MixColumns

- **Explication mathématique [2] :**

Tout comme un octet peut être représenté par un polynôme de degré 7, un mot de 32 bits (4 octets) peut l'être par un polynôme de degré 3 à coefficients dans **GF(2^8)**, où chaque coefficient représentera un octet du mot, la forme générale du polynôme sera:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

La différence ici, par rapport aux octets qu'on pouvait représentés en tant qu'éléments de **GF(2^8)** est que : c'est les coefficients eux même qui sont éléments de **GF(2^8)**.

Pour expliquer la multiplication dans ce cas de figure (intervenant dans l'étape MixColmuns ainsi que dans son inverse), soit un deuxième polynôme, à coefficient dans **GF(2^8)** :

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

La multiplication de a(x) et b(x) se fait en deux étapes :

D'abord, la multiplication polynomiale dans **GF(2^8)**, nous donnera un polynôme d'ordre 6 (avec coefficients dans **GF(2^8)**). Ce dernier ne représentera plus un mot de 4 octets. La seconde étape consiste en une réduction modulo un polynôme de degré 4, qui vaut pour l'AES X^4+1 (Afin de rester dans l'espace des mots de 32 bits).

Pour résumer : c(x) = a(x) ⊗ b(x) = a(x) • b(x) mod (x4 + 1).

Supposons b(x) constant, alors les opérations décrites précédemment peuvent êtres mises sous forme matricielle de la sorte:

$$c(x) = b(x) \otimes a(x) \quad \Leftrightarrow \quad \begin{bmatrix} c0 \\ c1 \\ c2 \\ c3 \end{bmatrix} = \begin{bmatrix} b0 & b3 & b2 & b1 \\ b1 & b0 & b3 & b2 \\ b2 & b1 & b0 & b3 \\ b3 & b2 & b1 & b0 \end{bmatrix} \cdot \begin{bmatrix} a0 \\ a1 \\ a2 \\ a3 \end{bmatrix}$$

X^4+1 n'étant pas irréductible sur **GF(2^8),** $c(x)$ n'est pas forcement inversible. Pour palier à cet inconvénient, l'algorithme de l'AES utilise un polynôme inversible fixé à la place de $a(x)$, qui garantit l'inversibilité de $c(x)$, il est donné par:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

Et son inverse par : $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$

III.C.1.e.1.4. AddRoundKey [2]:

Lors de l'étape AddRoundKey, la matrice State est modifiée en l'additionnant (au sens de l'addition terme à terme dans **GF(2^8)**) avec une clé de ronde.

Cette étape est illustrée par la figure suivante :

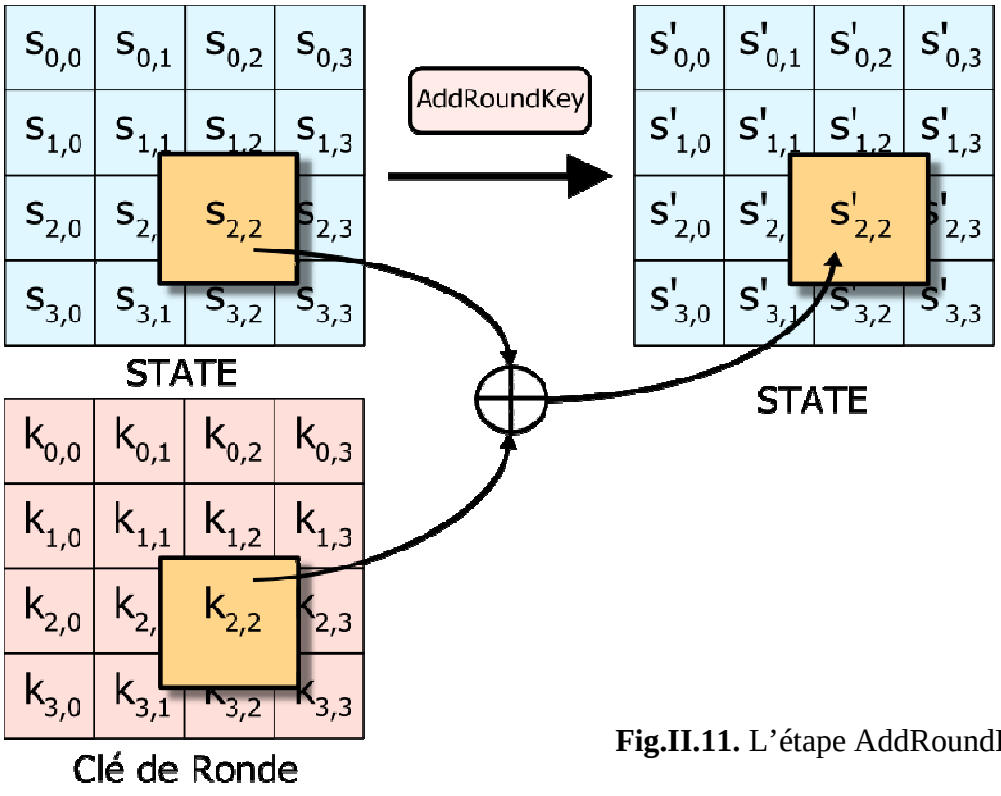


Fig.II.11. L'étape AddRoundKey

III.C.1.e.1.5. ExpandKey (ou KeyExpansion) [4]:

L'AES prend la clé principale de chiffrement K, et lui applique une routine d'extension (diversification) afin de générer un total de *Nb* (*Nr* + 1) mots de 32 bits chacun.
Le résultat de l'extension de la clé est représenté sous forme d'un tableau linéaire de mots de 32 bits, noté *Wi*, avec : $0 \leq i < Nb(Nr + 1)$ (Pour rappel Nb est fixé à 4 dans l'AES).

Longueur de la clé ***(Nk mots)***	**Nombre de tours (rondes)** ***(Nr)***	**Taille de la clé étendue Wi** ***(Nb(Nr+1) mots)***
4	10	44
6	12	52
8	14	60

Tab.II.3. Taille de *Wi* suivant la taille de la clé

La transformation initiale requière un jeu initial de 4 mots et chacun des Nr tours requière également 4 mots de clé. Comme l'illustre la figure suivante (avec une clé de 192 bits):

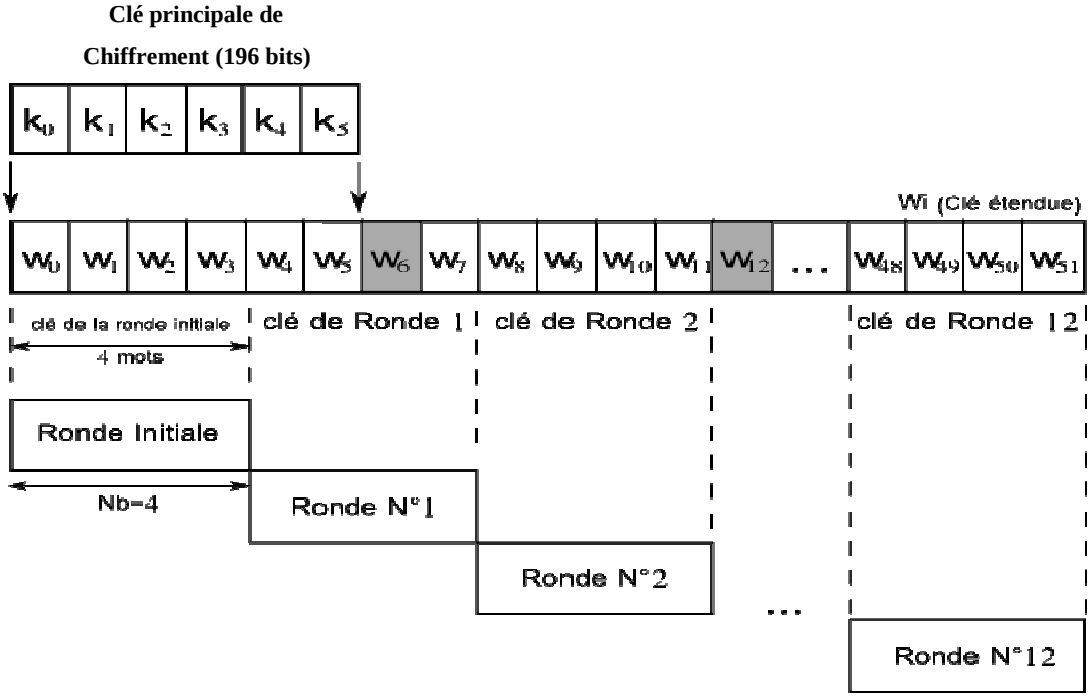


Fig.II.12. Application des clés de ronde sur chaque ronde

La Diversification de la clé se fait selon les étapes suivantes:

- Les Nk (4, 6 ou 8) premiers mots [w_0,…, w_{Nk-1}] contiennent la clé.
- Les mots suivants sont calculés de deux manières différentes :

1. Pour les mots situés sur une position qui est un multiple de Nk, On applique les transformations illustrées ci-dessous:

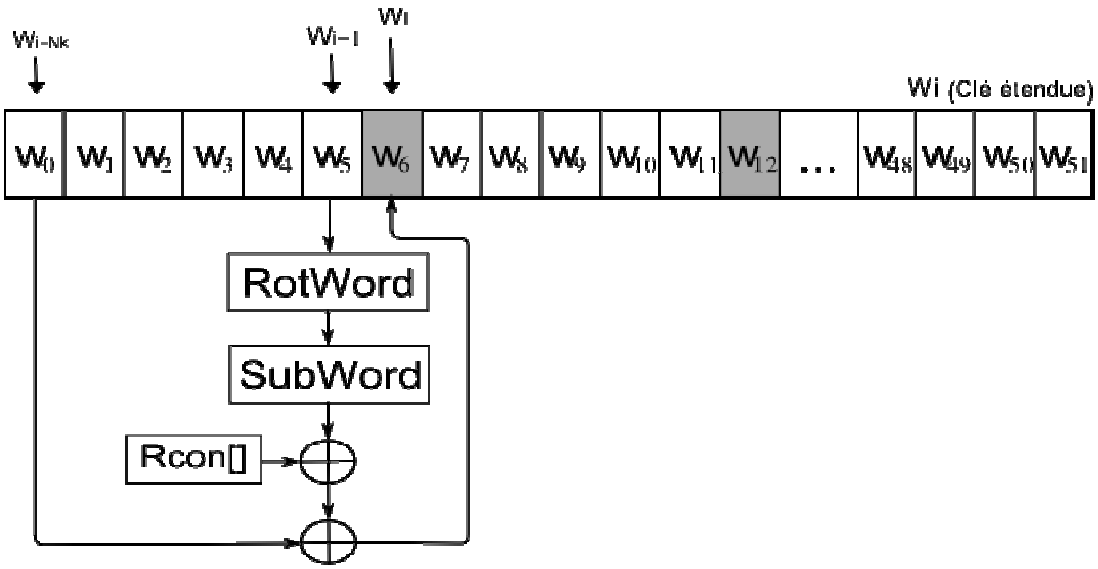


Fig.II.13. Calcul des éléments Wi se trouvant a une position multiple de Nk

La routine **RotWord()** prend en entrée un mot de 4 octets [a_0,a_1,a_2,a_3] et lui applique une permutation circulaire afin d'avoir [a_1,a_2,a_3, a_0].

La routine **SubWord()** prend aussi en entrée un mot de 4 octets et substitue à chaque octet sa valeur correspondante de la S-box.

Rcon[] est un vecteur constant, contenant un mot de 4 octets, défini par :

$$Rcon[i/Nk] = Rcon[j] = [x^{j-1}, 0, 0, 0]$$

Avec : x = 0x02

i : indice de Wi

Les puissances de **x** sont calculées dans GF(2^8) :

j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
$\mathbf{x^{j-1}}$	01	02	04	08	10	20	40	80	1b	36	6c	d8	ab	4d	9a

Tab.II.4. Puissances de *x* (0x02) dans **GF(2^8)**

2. Pour tout les autres mots :

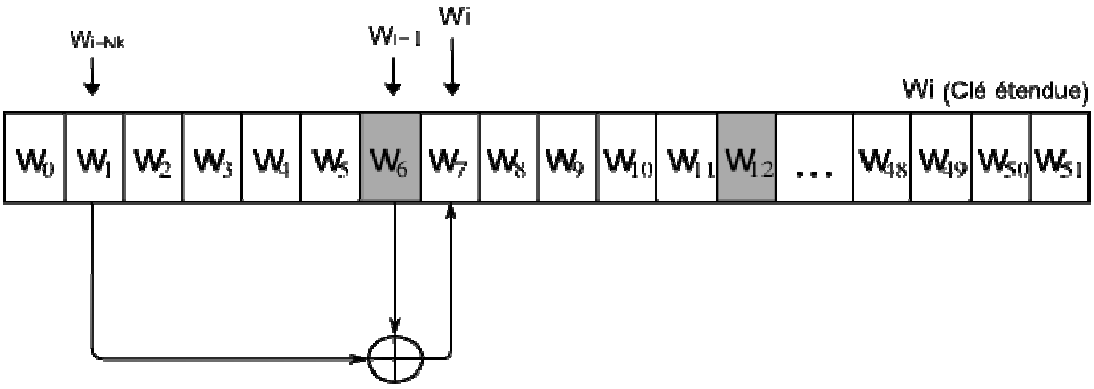


Fig.II.14. Calcul des éléments Wi commun

Le mot concerné est calculé en faisant un XOR entre le mot qui le précède et le mot qui a pour rang Nk position avant lui.

$$\mathbf{Wi = W_{i-1} \oplus W_{i-Nk}}$$

Remarque :

Il est important de signaler que l'Expansion dans le cas d'une clé à 256 bits est différente des deux autres cas (128 et 196 bits). En effet, dans ce cas où Nk = 8, si le rang i-4 est un multiple de Nk, la fonction « SubWord() » est appliquée à W_{i-1} avant le XOR avec W_{i-4}.

III.C.1.e.2. Déchiffrement :

La routine de chiffrement peut être inversée et réordonnée pour produire un algorithme de déchiffrement utilisant les transformations InvSubBytes, InvShiftRows, InvMixColumns, et AddRoundKey. **[3]**

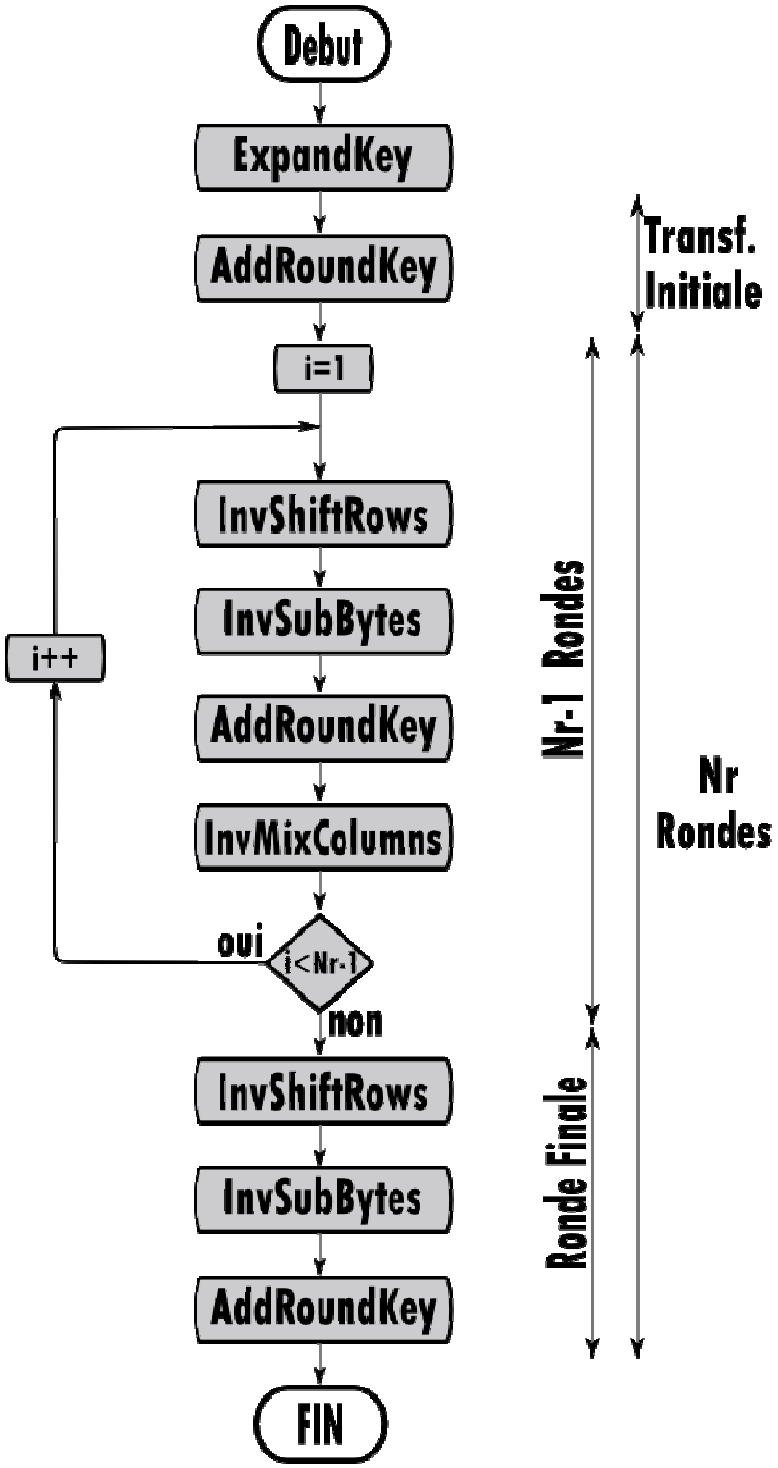


Fig.II.15. Organigramme de déchiffrement

Lors du déchiffrement la séquence des transformations diffère de celle du chiffrement, l'extension de la clé se fait de la même manière appart que les clés de rondes sont distribuées dans l'ordre inverse par rapport au chiffrement, comme illustré dans la figure suivante (avec une clé de 128 bits):

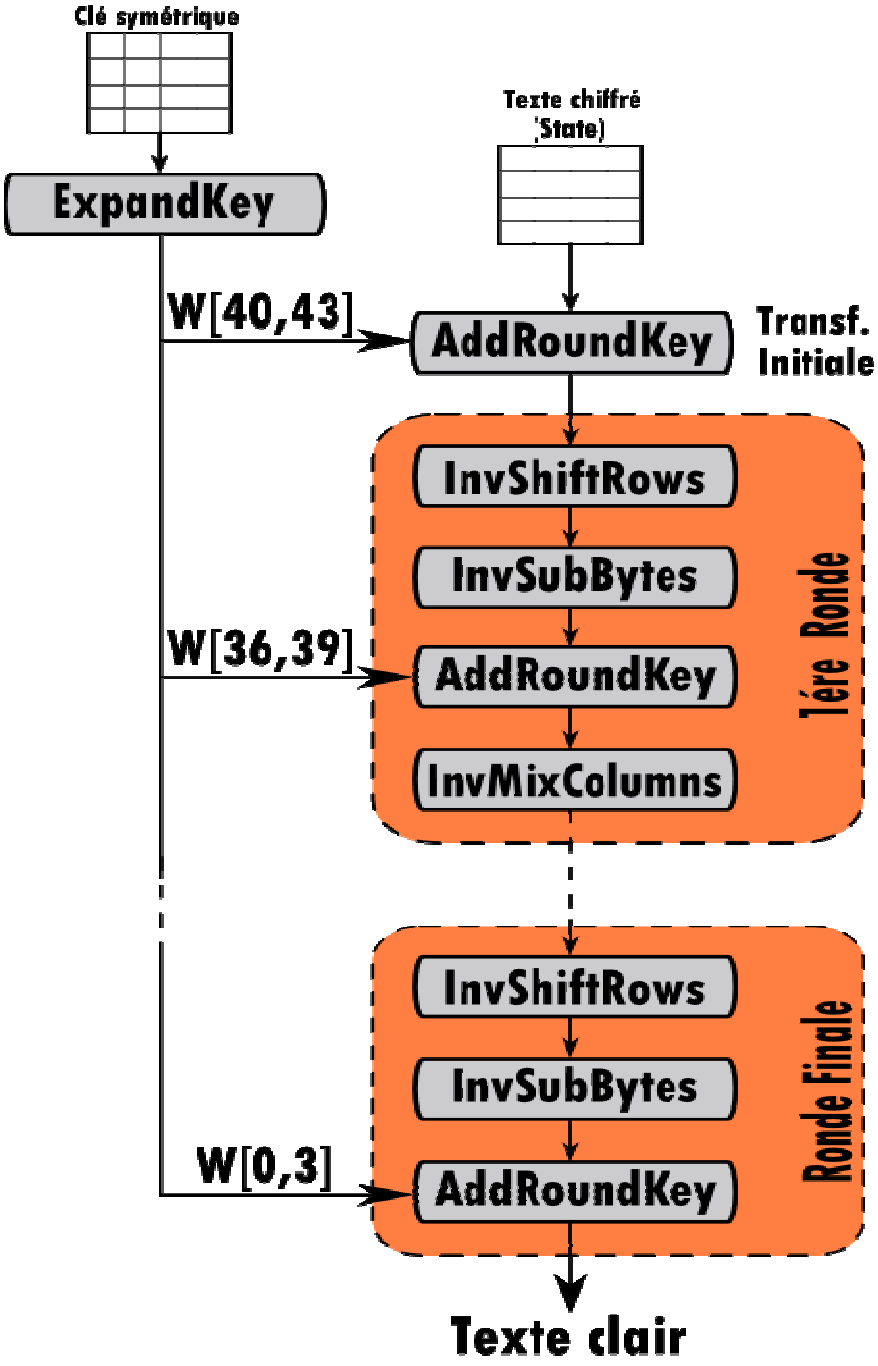


Fig.II.16. Ordre d'application des clés de ronde pendant le déchiffrement

Remarque [3]:

Certaines propriétés du Rijndael permettent d'implémenter une routine de déchiffrement équivalente qui respecte la séquence des transformations du chiffrement, avec toute fois un traitement spécifique de la clé (une routine est ajoutée a la fin de « KeyExpansion »).

III.C.1.e.2.1. Opération InvSubBytes [2]:

L'opération inverse de SubBytes est notée InvSubBytes et consiste à effectuer la même manipulation que SubBytes mais à partir de la Boite-S inverse notée InvSBox.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Fig.II.17. InvSBox

InvSBox est generée selon la transformation suivante :

$$InvSBox[a] = t^{-1}(\mathcal{F}^{-1}(a)) \ , \ \forall a \in GF(2^8)$$

Comme la fonction ***t*** est son propre inverse, on aura finalement :

$$InvSBox[a] = t(\mathcal{F}^{-1}(a)) \ , \ \forall a \in GF(2^8)$$

La fonction affine inverse $\mathcal{F}^{-1}$ est définie par (forme matricielle simplifiée):

$$a = \mathcal{F}^{-1}(a') \Leftrightarrow \begin{bmatrix} a0 \\ a1 \\ a2 \\ a3 \\ a4 \\ a5 \\ a6 \\ a7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a'0 \\ a'1 \\ a'2 \\ a'3 \\ a'4 \\ a'5 \\ a'6 \\ a'7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

III.C.1.e.2.2.Opération InvShiftrows :

L'opération inverse de Shiftrows est notée InvShiftrows et consiste à effectuer un décalage cyclique vers la droite de n éléments selon la même règle que pour l'opération shiftrows.

III.C.1.e.2.3. Opération InvMixColumns [2]:

L'opération inverse de MixColumns est notée InvMixColumns. Elle consiste à effectuer les mêmes opérations que dans MixColumns, mais à partir d'une multiplication, modulo x^4+1, par le polynôme :

$$d(x) = c^{-1}(x) = \{0b\}\, x^3 + \{0d\}\, x^2 + \{09\}\, x + \{0e\}$$

Comme déjà décrit pour l'étape ShiftRows, l'opération: s'(x) = s(x) ⊗ d(x), peut-être mise sous forme matricielle :

$$s'(x) = d(x) \otimes s(x) \Leftrightarrow \begin{bmatrix} S'(0,n) \\ S'(1,n) \\ S'(2,n) \\ S'(3,n) \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \bullet \begin{bmatrix} S(0,n) \\ S(1,n) \\ S(2,n) \\ S(3,n) \end{bmatrix}; \ 0 \leq n < Nb$$

III.C.2. RSA (Rivest Shamir Adleman) et SHA-1 (Secure Hash Algorithm):

Le premier système à clé publique solide à avoir été inventé, et le plus utilisé actuellement, est le système RSA. Publié en 1978 par Ron **R**ivest, Adi **S**hamir et Leonard **A**dleman. Le RSA est fondé sur deux principes mathématiques fondamentaux : la difficulté de factoriser des grands nombres, et l'arithmétique des congruences **[5]**.

III.C.2.1. Principe de fonctionnement de l'RSA [4] [5]:

Pour qu'Alice puisse échanger sa clé avec Bob, elle doit d'abord la calculer en mettant en œuvre des notions mathématiques remarquables par leur simplicité, d'ou le fonctionnement du cryptosystème RSA peut se résumer en 03 phases :

- Génération des clés (effectué par la destinataire Alice)
- Chiffrement (effectué par l'expéditeur Bob)
- Déchiffrement (effectué par la destinataire Alice)

III.C.2.1.a. Génération des clés :

Cette phase peut se résumer en 03 étapes :

1er étape : Alice choisit au hasard deux grands nombres entiers, naturels, premiers, p et q, ont environ 100 chiffres chacun ou plus pour rendre la factorisation hors de la portée. Dans notre exemple simplifié elle choisit :

$$p = 31 \text{ et } q = 53$$

Et fait leur produit :

$$n = p*q = 1643$$

2er étape : Alice détermine la fonction d'Euler « phi » associée à n déjà calculé en utilisant la formule :

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

$$\text{Phi}(n) = 30 * 52 = 1560$$

Une fois que la fonction d'Euler déterminée, Alice choisie au hasard sa clé publique « e », cette clé est un nombre premier compris entre 1 et phi(n) et premier relativement à Phi(n) c'est-à-dire le pgcd (e, phi(n))=1. Alice fait : e = 11

D'où le couple (e, n) constitue la clé publique.

$$\begin{cases} n : \text{c'est le module} \\ e : \text{c'est l'exposant} \end{cases}$$

La clé publique est donc (11, 1643)

3er étape : Cette dernière étape consiste à trouver la clé privée « d » qui correspond à la clé publique choisie précédemment avec d compris entre 1 et phi(n), pour se faire, if faut résoudre l'équation suivante :

$$e.d \mod \text{Phi}(n) = 1$$

$$\text{c.à.d : } e.d \equiv 1[\text{Phi}(n)]$$

$$\text{Donc : } e.d = k\,\text{Phi}(n) + 1$$

Selon notre exemple on aura :

$$e.d = k\ Phi(n) + 1$$

$$11.\ d = k\ .\ 1560 + 1$$

Pour k= 6 on aura : 11.d = k. 1560 + 1 on aura d = 851

Le couple (d, n) constitue la clé privée

La clé privée est donc (851, 1643)

$$\begin{cases} n : c'est\ le\ module \\ d : c'est\ l'exposant \end{cases}$$

En fin, Alice et Bob disposent toutes les clés indispensables au chiffrement et au déchiffrement des messages après la transmission ou la publication de sa clé publique (e, n). Maintenant il faut qu'elle conserve sa clé privée (d, n) et qu'elle n'oublie jamais les nombres p et q.

III.C.2.1.b. Chiffrement :

Bob veut donc transmettre le message M «ANEMONE» à Alice. Il cherche dans l'annuaire la clé de chiffrement qu'Alice a déjà publiée. Il sait maintenant qu'il doit utiliser le système RSA avec les deux entiers n et e (dans notre exemple n = 1643 et e = 11).
Il va procéder son cryptage de la manière suivante :

- Il transforme en nombres son message en remplaçant par exemple chaque lettre par son rang dans l'alphabet : a = 01, b = 02 ……..z = 26. il résulte :

M = ANEMONE

M = A N E M O N E

M = 01 14 05 13 15 14 05

- Il découpe son message numérisé en blocs de même longueur représentant chacun une taille égale ou inférieure à celle de n ce qui empêche la simple substitution. Dans notre

exemple la taille de n est 3, ce qui donne des tranches m_i de 03 chiffres chacune, le message devient :

$$M = 001\ \ 140\ \ 513\ \ 151\ \ 405$$

$$M = \ m_1\ \ m_2\ \ m_3\ m_4\ \ m_5$$

- Chaque bloc m_i est chiffré par la formule :

$$\mathbf{C_i = M_i^e \bmod n}$$

Ce qui donne :

$$C_1 = m_1^{11} \bmod 1643 = 001$$
$$C_2 = m_2^{11} \bmod 1643 = 109$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$\cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot$$
$$C_5 = m_5^{11} \bmod 1643 = 374$$

$001^{11} \bmod 1643 =$	1
$140^{11} \bmod 1643 =$	109
$513^{11} \bmod 1643 =$	890
$151^{11} \bmod 1643 =$	1453
$405^{11} \bmod 1643 =$	374

Alors le message chiffré C sera:

$$C = \ c_1 \quad c_2 \quad c_3 \quad c_4 \quad c_5$$
$$C = 0001\ \ 0109\ \ 0890\ \ 1453\ \ 0374$$

Enfin, Bob a son message chiffré, il peut donc l'envoyer à Alice.

III.C.2.1.c. Déchiffrement :

Alice reçoit le message de Bob, à partir de p et q, qu'elle a gardés secrets elle calcule la clef d de déchiffrement (c'est sa clef privée). Celle-ci doit satisfaire l'équation :

$$e \cdot d \bmod ((p-1)(q-1))=1$$

Chacun des blocs c_i du message chiffré sera déchiffré par la formule

$$M_i = C_i^d \bmod n$$

Ce qui lui donne :

$1^{851} \bmod 1643 =$	1
$109^{851} \bmod 1643 =$	140
$890^{851} \bmod 1643 =$	513
$1453^{851} \bmod 1643 =$	151
$374^{851} \bmod 1643 =$	405

M = 001 140 513 151 405

M= ANEMONE

Lors du déchiffrement, sachant qu'il faut obtenir des blocs de 2 éléments (grâce au codage particulier de l'exemple). Finalement, Alice prend sa table de correspondance alphabétique pour restituer le message M, elle aura :

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

III.C.2.1.d. Résumé :

Le RSA est un algorithme de chiffrement asymétrique fait appel aux notions suivantes :

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

Les deux figures suivantes représentent respectivement la manière de chiffrement et de déchiffrement d'un message M :

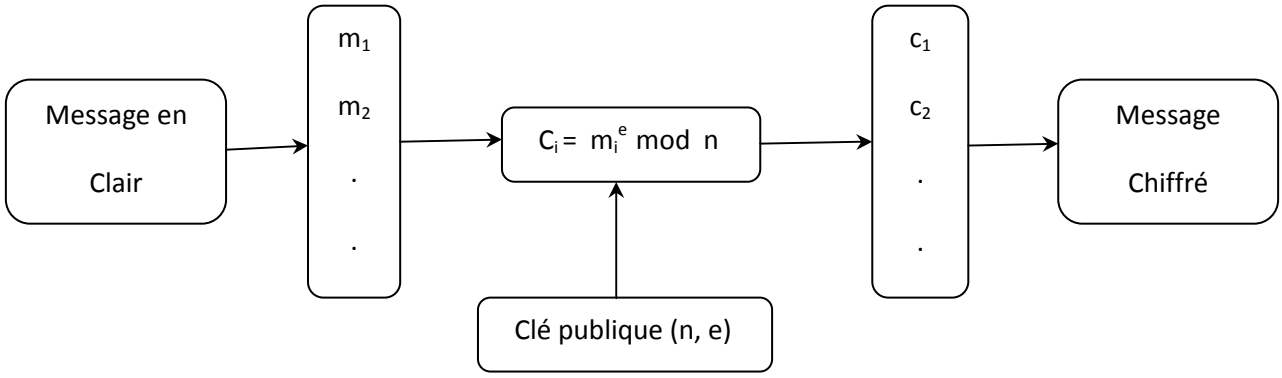


Fig.II.18. Chiffrement d'un message avec RSA

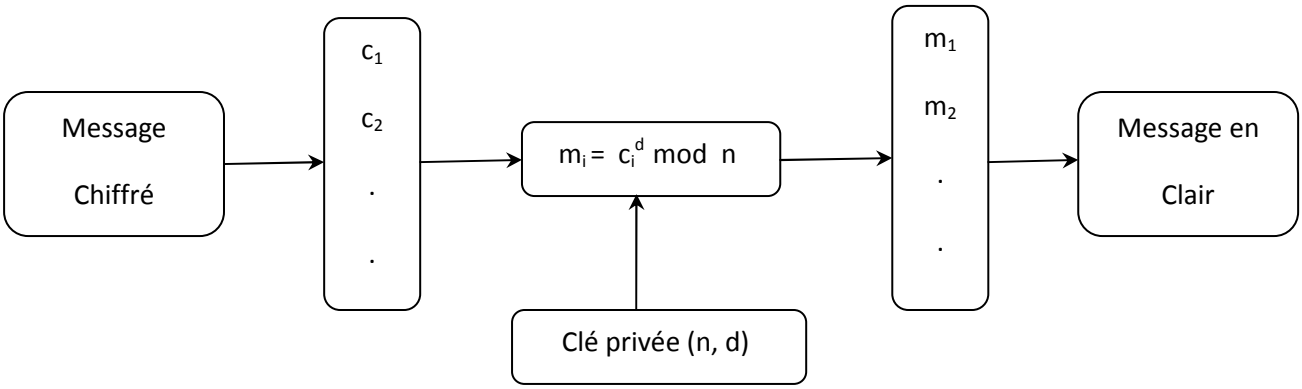


Fig.II.19. Déchiffrement d'un message avec RSA

III.C.2.2. Sécurité et conditions optimales d'utilisation du RSA [6]:

Il faut savoir qu'un mauvais choix des paramètres *p*, *q*, *d* et *e* peut rendre le système relativement vulnérable. Pour cela, les concepteurs du système ont proposé un certain nombre de règles :

- Ne jamais utiliser de valeur n trop petite : Il faut choisir $n = p.q$ de taille supérieure ou égale à 512 bits (environ 155 chiffres décimaux).
- N'utiliser que des clés fortes (p-1 et q-1 ont un grand facteur premier) : Il faut choisir, si possible, des nombres premiers *p* et *q*, tels que *p-1*, *p+1*, *q-1* et *q+1* possèdent de grands facteurs premiers et prendre *p* et *q* de taille sensiblement égale, mais pas trop proches en valeur absolue.
- Ne pas chiffrer de blocs trop courts,
- Ne pas utiliser de n communs à plusieurs clés,

La sécurité du système RSA se base sur le fait qu'il y a une impossibilité pratique de factoriser un grand nombre de quelques centaines de chiffres en un temps raisonnable : Selon R.S.A., factoriser un nombre à 200 chiffres demande 4 milliards d'années de calcul machine. Factoriser un nombre de 500 chiffres demande 10^{25} ans. La robustesse du R.S.A. apparaît donc liée à la difficulté de la factorisation avec les méthodes actuelles.

Pour « casser » un message chiffré par la méthode RSA, sachant qu'on ne sait pas calculer les racines e-ièmes modulo *n* de celui-ci, on s'attaquera à la méthode de détermination des clés. En l'examinant, on devine facilement comment forcer le codage : le choix des clés s'effectuant à partir de la formule $e.d = K.phi(n)+1$, on essaie de déterminer la clé secrète (*d*) à partir de la clé publique en résolvant l'équation. Pour la résoudre, il suffit d'avoir phi(n). Pour obtenir phi(n), il faut le décomposer en ses facteurs premiers *p* et *q*.

Cependant, il n'a jamais été démontré que le code ne pouvait être cassé sans la connaissance de *p* et *q*.

III.C.2.3. Principe de fonctionnement de l'SHA-1 :

SHA (Secure Hash Algorithm) a été conçu par NSA (Nation Security Agency) et publié par NIST (un organisme chargé des standards). En mai 1993, sa première version correspond à SHA-0. Deux en plus tard (en 1995), suite à la découverte de faiblesses, la NSA

publie une révision de son standard (SHA-0) pour étendre ses capacités en matière de sécurité: c'était le SHA-1.

La famille SHA se divise en deux catégories : SHA-1 et SHA-2. Jusqu'à 2005, il était l'algorithme généralement préféré pour le hachage, mais des rumeurs de cassage le font peu à peu évoluer vers des versions plus sophistiquées.

Le SHA-1 travaille sur des nombres binaires dont le message est soumis à un remplissage et il se voit découpé en blocs de 512 bits. Pour chaque bloc de 512 bits, SHA-1 calcule 80 itérations appelées aussi rondes. Elles sont groupées en quatre parties principales avec 20 transformations dans chacune d'entre elles. D'autre part un vecteur d'initialisation sur 160 bits fournie 05 mots de 32bits A, B, C, D, E en attaquant directement la première ronde. Leurs valeurs initiales pour le premier hachage sont définies par la spécification :

VI = (A, B, C, D, E) sur 160 bits : avec A, B, C, D, E sont des mots de 32 bits.

Avec :

- A = 67452301 (en hexadécimal)
- B = EFCDAB89 ……
- C = 98BADCFE ……
- D = 10325476 ……
- E = C3D2E1F0 …….

Ces transformations sont sous forme de petites boites noires, telles que 20 boites appartenant à un même groupe utilisent toutes la même fonction, cette fonction prend 3 mots de 32 bits (B, C, D) en entrée et fournit un mot de 32 bits en sortie. Par contre, les fonctions diffèrent entre les quatre groupes. Ces fonctions primitives sont non linéaires et définies comme suit :

- Elles travaillent sur 3*32 bits et fournissent un résultat sur 32 bits.
- Pour le 1er groupe de 20 rounds : $0 \leq t \leq 19$, Avec : t étant le numéro d'un round
- $f_1 = f(t, B, C, D) = (B \wedge C) \vee (\neg B \wedge C)$; ($\neg$: complément binaire)
- Pour le 2eme groupe de 20 rounds : $20 \leq t \leq 39$, $f_2 = f(t, B, C, D) = B \oplus C \oplus D$
- Pour le 3eme groupe de 20 rounds : $40 \leq t \leq 59$,

  $f_3 = f(t, B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
- Pour le 4eme groupe de 20 rounds : $60 \leq t \leq 79$, $f_4 = f(t, B, C, D) = B \oplus C \oplus D$

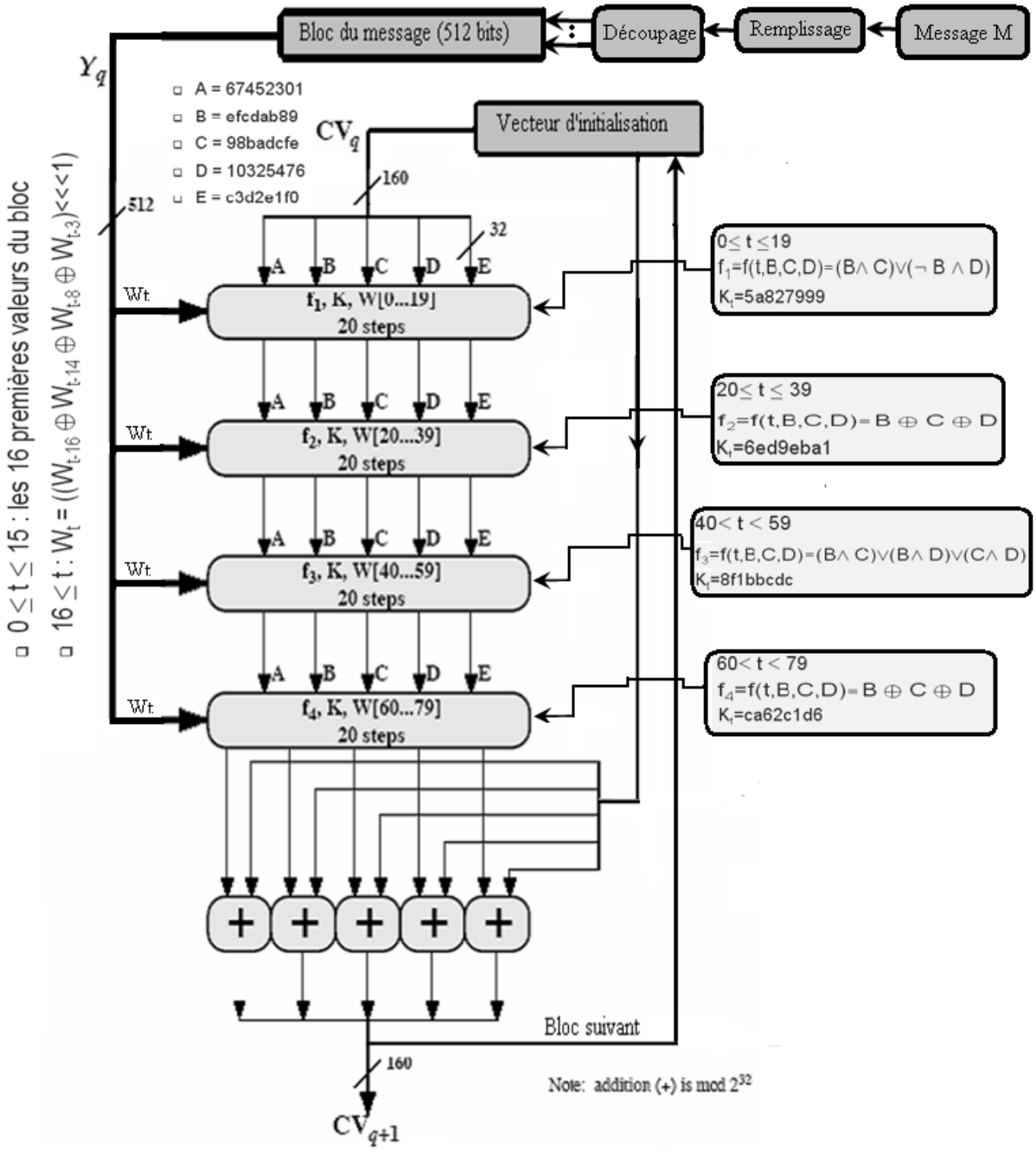


Fig.II.20. Principe de fonctionnement de SHA-1

SHA-1 n'a ainsi pas une structure semblable selon le numéro du tour. Mais ce n'est pas tout, les boites(les rounds) ont également besoin du bloc de 512 bits à hacher. Pour cela, on le découpe en 16 mots de 32 bits (w[1], w[2],..., w[16]) et une fonction se charge de générer encore 64 autres mots de 32 bits afin d'en avoir 80, donc un pour chaque ronde.

Pour créer ces mots supplémentaires, on applique un ou-exclusif entre quatre mots déjà présents dans la liste : w[t] = w[t-3] XOR w[t-8] XOR w[t-14] XOR w[t-16] c'est-à-dire :

- Pour $0 \leq t \leq 15$: Wt prend les 16 premières valeurs du de 512 bits.
- Pour $16 \leq t \leq 19$: Wt = w[t] = w[t-3] ⊕ w[t-8] ⊕ w[t-14] ⊕ w[t-16]

C'est ici qu'intervient la différence entre SHA-0 et SHA-1. Dans SHA-1, ce résultat est soumis à une rotation circulaire vers la gauche de 5 bits. Cela permet de casser une structure trop linéaire qui fut fatale pour SHA-0.

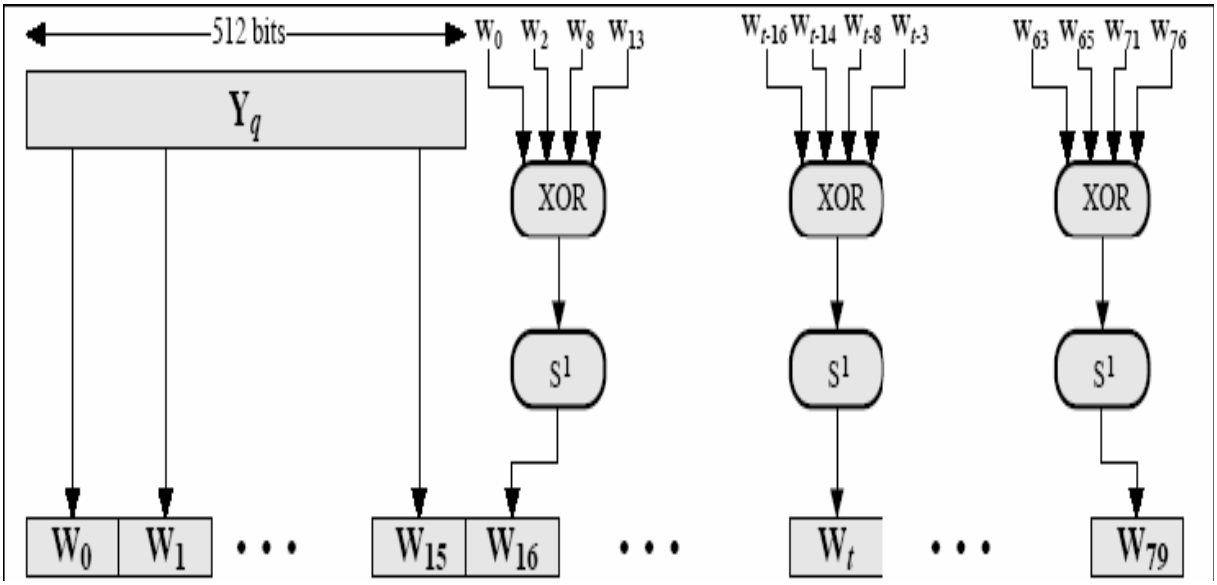


Fig.II.21. Calcul des Wt dérivés du bloc de 512 bits

À ce stade, nous avons donc 80 "clés" qui peuvent être envoyées vers les boites noires respectives qui prennent plusieurs variables de 32 bits en entrée :

A chaque ronde de SHA-1, ces 5 variables sont modifiées et le résultat est transmis à la ronde suivante. On effectue en premier une rotation circulaire vers la gauche de 5 bits sur la variable A, on calcule f(B,C,D) en utilisant les fonctions évoquées précédemment. On récupère alors E, ainsi que la "clé" correspondant à cette ronde ainsi qu'une constante Kt définie par le standard qui varie selon les tours (groupes), les valeurs de Kt en hexadécimal sont définies comme suit :

- Pour le 1er groupe de 20 rounds : $0 \leq t \leq 19$: Kt = 5A827999
- Pour le 2eme groupe de 20 rounds : $20 \leq t \leq 39$: Kt = 6ED9EBA1
- Pour le 3eme groupe de 20 rounds : $20 \leq t \leq 39$: Kt = 8F1BBCDC
- Pour le 4eme groupe de 20 rounds : $20 \leq t \leq 39$: Kt = CA62C1D6

On aura à chaque itération (ronde) les valeurs suivantes de (A, B, C, D, E) :

(A, B, C, D, E) ← (E + f(t, B, C, D) + (A << 5) +Wt + Kt), A, (B << 30), C, D)

En fin, on additionne le tout modulo 2^{32} car nous travaillons avec des registres sur 32 bits et on stocke temporairement cette valeur. Les variables sont ensuite écrasées en prenant bien soin de les permuter lors de l'attribution et d'appliquer quelques rotations intermédiaires sur certaines d'entre elles.

À la fin de la dernière ronde, on additionne les valeurs courantes de A,B,C,D et E avec les valeurs de départ provenant du bloc précédent. En concaténant ces cinq variables de 32 bits, on obtient l'empreinte de 160 bits. Si d'autres blocs doivent être hachés, cette empreinte intermédiaire est passée au bloc suivant en tant que vecteur d'initialisation.

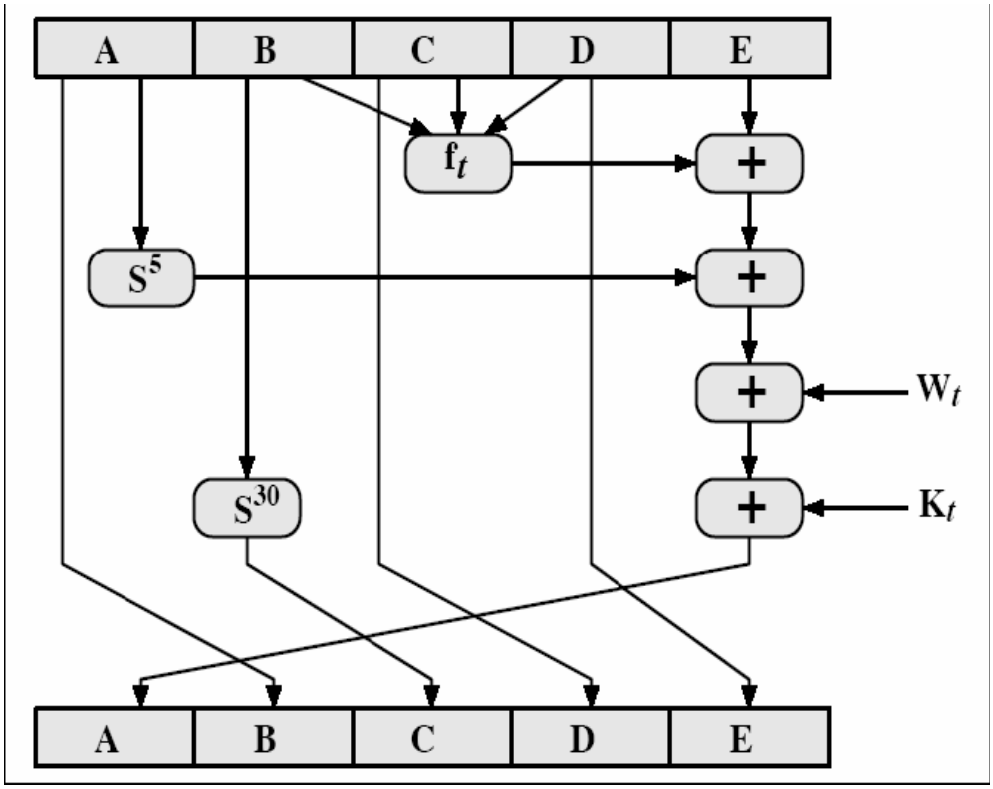


Fig.II.22. Fonction de ronde du SHA-1

Récapitulatif :

1. Complétion.

2. Découpage

3. Initialisation : Initialiser 5 buffers de 32 bits (= 160 bits) : A, B, C, D, E

4. Calcul itératif : 4 rondes de 20 itérations chacune. Ces rondes ont une structure similaire mais utilisent des fonctions primitives différentes (f1, f2, f3, f4). Le SHA utilise des constantes additives Kt ($0 \leq t \leq 79$). Le résultat est utilisé pour initialiser les buffers du bloc suivant.
Chaque ronde comprend 20 étapes qui manipulent ainsi les 5 registres :

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

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

5. Le condensé final constitue le condensé attendu.

Chapitre III

1. Introduction :

Pour donner des ordres à un ordinateur, il faut employer un langage, malheureusement ces machines ne comprennent pas le langage humain. Il est donc nécessaire d'écrire des suites d'ordres appelés programmes à l'aide d'un langage conventionnel dénommé langage de programmation.

Dans ce chapitre, une méthodologie récente de construction d'un logiciel est employée, il s'agit de la programmation orienté objet. Grâce à des environnements visuels, il est possible d'aborder avec profit ce type puissant, l'environnement RAD Borland Delphi _7 (Rapid Application Developpement) qui est une extension orienté objet du pascal est notre type d'exploration de cette technique sous Windows .

2. Synoptique de chiffrement de données :

Le type de données que le logiciel L.C.H (Logiciel de chiffrement hybride) traite peut être :

- Soit un texte
- Soit des fichiers
- Soit des images bitmaps en niveau de gris

Le principe de chiffrement retenu repose sur une suite d'étapes décrites dans la figure III.1 . En effet, celles-ci suivent un chemin bien définit afin d'être cryptées.

- Pour chaque type de donnée (Texte, Fichier, Image) , un algorithme de chiffrement doit être choisit :
    1. Algorithme de chiffrement à clé secrète (AES-128, 192, 256).
    2. Algorithme de chiffrement à clé publique (RSA-128 bits, 256, 512, 768, 1024).
    3. Combinaison entre les deux algorithmes (Hybride AES/RSA).

- S'il s'agit d'un chiffrement à clé secrète (AES), on choisit :
    1. Une clé de chiffrement (Générée par le programme à partir d'une phrase de passe).
    2. Un mode de chiffrement (ECB, CBC).

- S'il s'agit d'un chiffrement à clé publique (RSA), on a:
    1. Dans le cas ou la donnée est un texte, le programme donne deux possibilités de chiffrement :
        a. Chiffrement sans signature : accompagné d'un choix de paire de clés de chiffrement (Générée par le programme).
        b. Chiffrement avec signature (RSA-256 bits, 768, 1024) : accompagné de deux choix :
            - Choix de deux paires de clés : L'une pour le chiffrement et l'autre pour la signature.
            - Choix de la fonction de hachage (MD5 ou SHA-1).
    2. Dans le cas ou la donnée est un fichier on une image bmp : le chiffrement s'effectue après la génération de la paire de clés.
- Si nous choisissons le chiffrement Hybride (AES/RSA), alors les différentes étapes décrites devront être suivies.

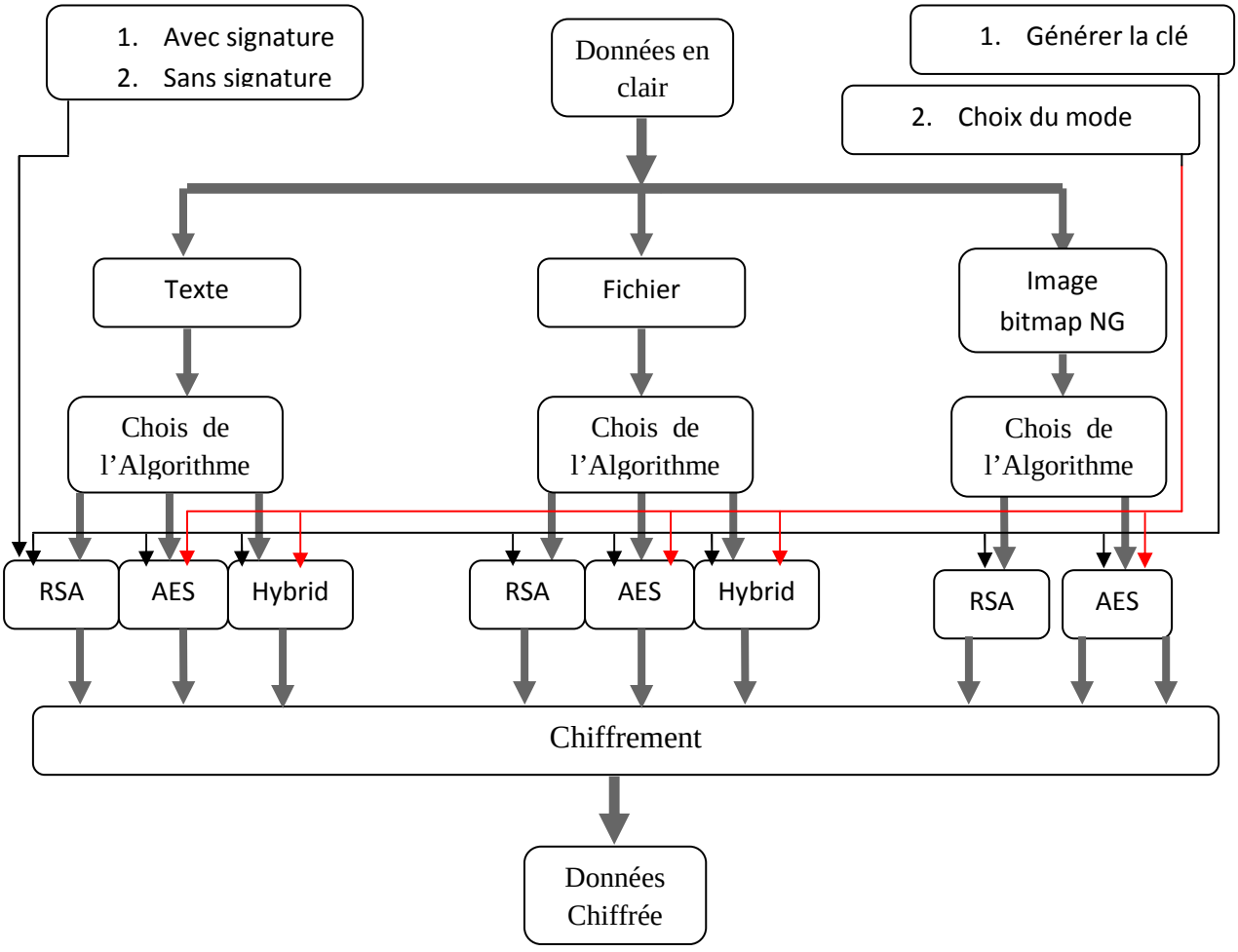


Fig.III.1. Procédure de chiffrement d'une donnée, adoptée par le logiciel L.C.H

3. Présentation de l'interface L.C.H :

Le logiciel réalisé L.C.H (Logiciel de chiffrement hybride) est à base d'une application MDI (Multiple document interface), constitué d'une fenêtre principale appelée « Le parent » et de fenêtres enfantes. Il reprend l'interface classique des applications développées sous Windows. La fenêtre principale comprend les composants suivants :

- Une barre de menu.
- Une barre d'outils.
- Une barre d'état : affiche des informations sur l'application ou le processus en cours d'exécution, la définition de l'élément sur lequel le curseur se pointe.

La figure suivante montre une telle fenêtre et les composants qu'elle intègre :

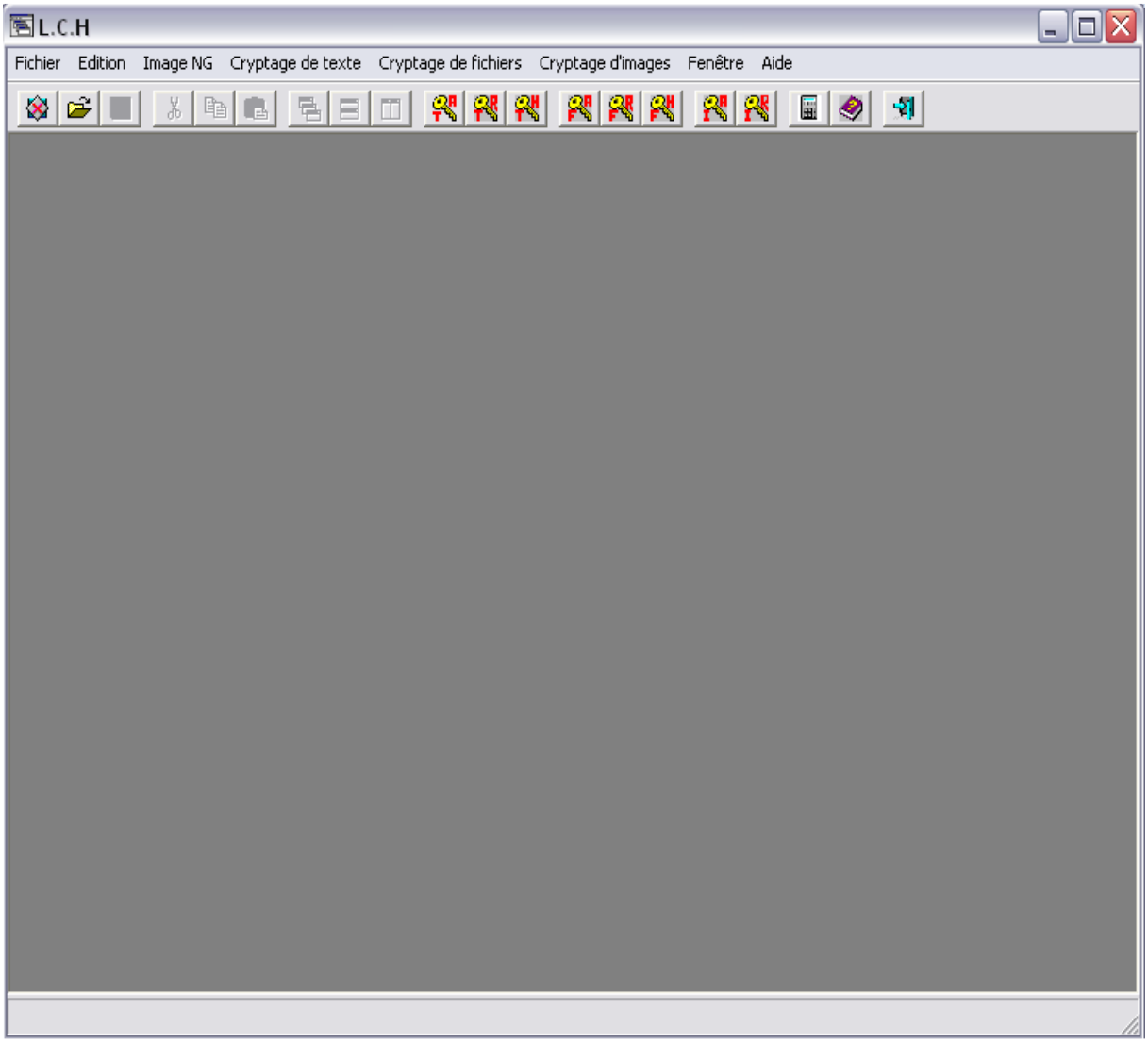
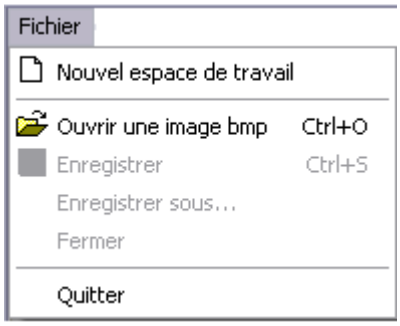


Fig.III.2. Fenêtre principale du logiciel L.C.H

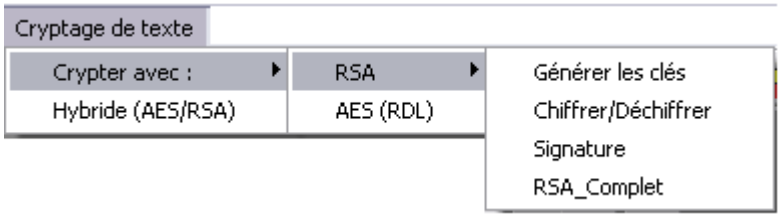
3.1. Les menus et les sous menus :

a. Le menu ‘Fichier’ :



- Nouvel espace de travail : permet de créer un nouvel espace pour le cryptage d’images.
- Ouvrir une image bmp: permet l’ouverture d’une image bitmap en niveau de gris.
- Enregistrer (Enregistrer sous) : permet de sauvegarder l’image courante.
- Fermer : Permet la fermeture de l’image active.
- Quitter: Permet de quitter l’application

b. Le menu ‘Cryptage de texte’ :



Le menu ‘**Cryptage de texte**’ est réservé au Chiffrement/Déchiffrement de texte.

On remarque une suite de sous menus permettant le choix de la méthode de chiffrement que nous souhaitons utiliser afin d’ouvrir des fenêtres dédiées au :

- Chiffrement hybride.
- Chiffrement à clé publique (RSA) avec ou sans signature.
- Chiffrement à clé secrète (AES).

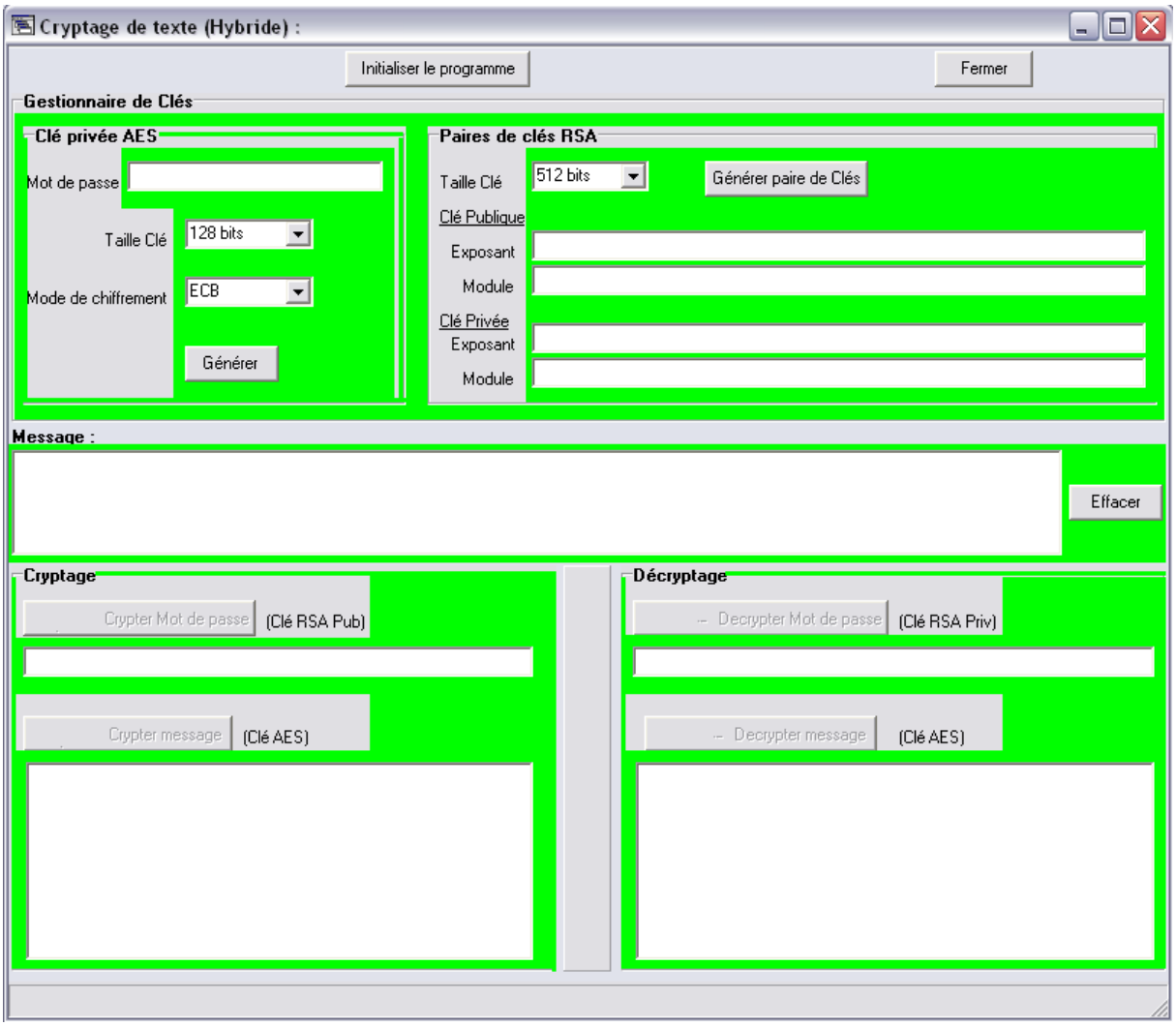


Fig.III.3. Chiffrement Hybride (AES/RSA) d'un texte

Cela permet de :

1) Chiffrer un texte avec une clé secrète.
2) Chiffrer la clé secrète avec la clé publique (supposant du destinataire).

Les sous menu RSA « Générer, Chiffrer/Déchiffrer, Signature », permet indépendamment de chiffrer/Déchiffrer, signer un texte avec une paire de clés (publique/privée) dont la fenêtre correspondante est illustrée dans la figure suivante :

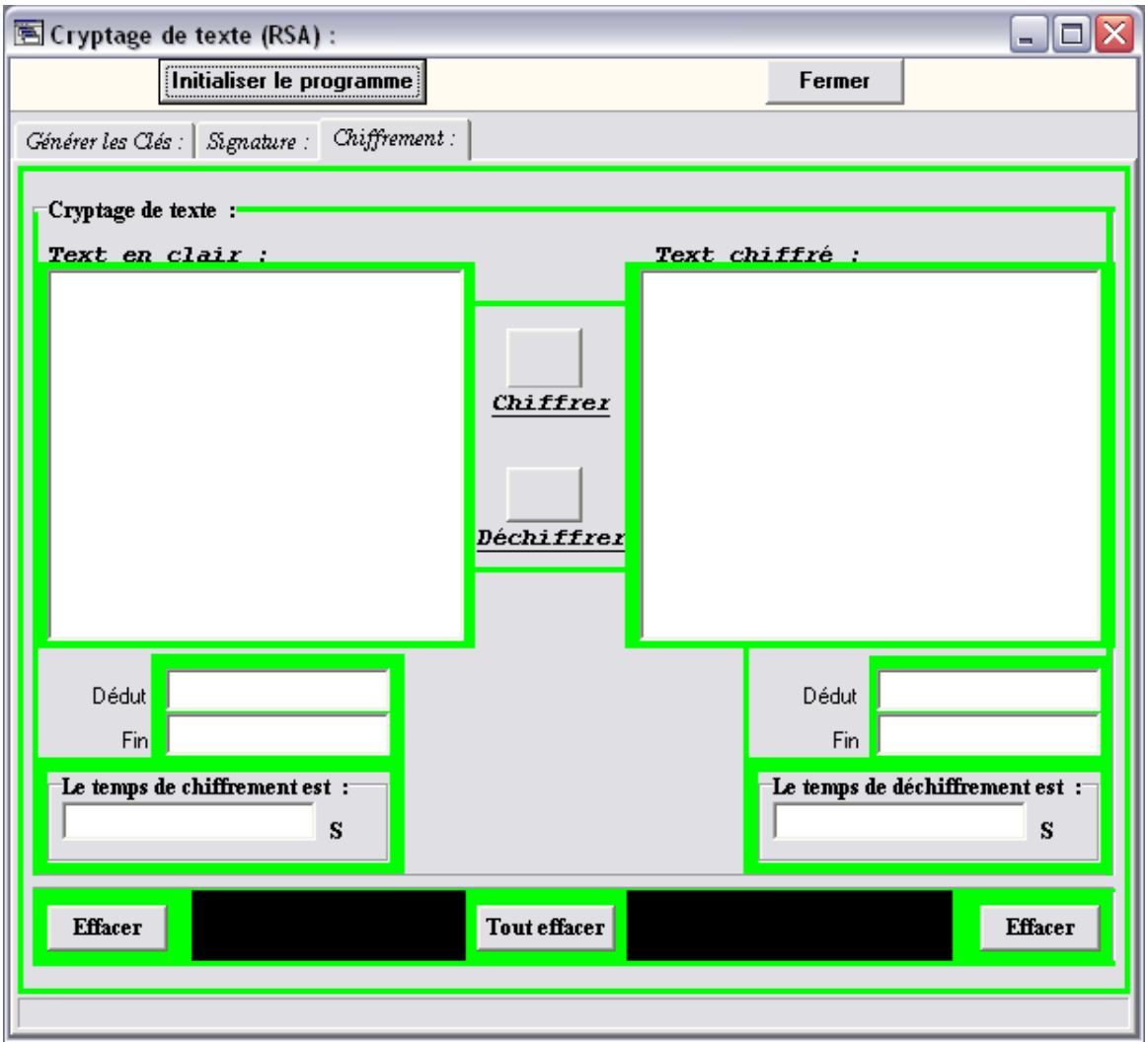


Fig.III.4. Chiffrement/déchiffrement d'un texte avec RSA

Les sous menu « RSA_Complet » permet de chiffrer/Déchiffrer et signer un message avec deux paires de clés (publiques/privées) dont la fenêtre correspondante est illustrée dans le figure suivante:

Fig.III.5. Chiffrement/déchiffrement, signature d'un texte avec AES

Le menu AES (RDL) permet de chiffrer/Déchiffrer un message avec une clé secrète générée à partir d'une phrase de passe dont la fenêtre correspondante est illustrée dans le figure III.6 :

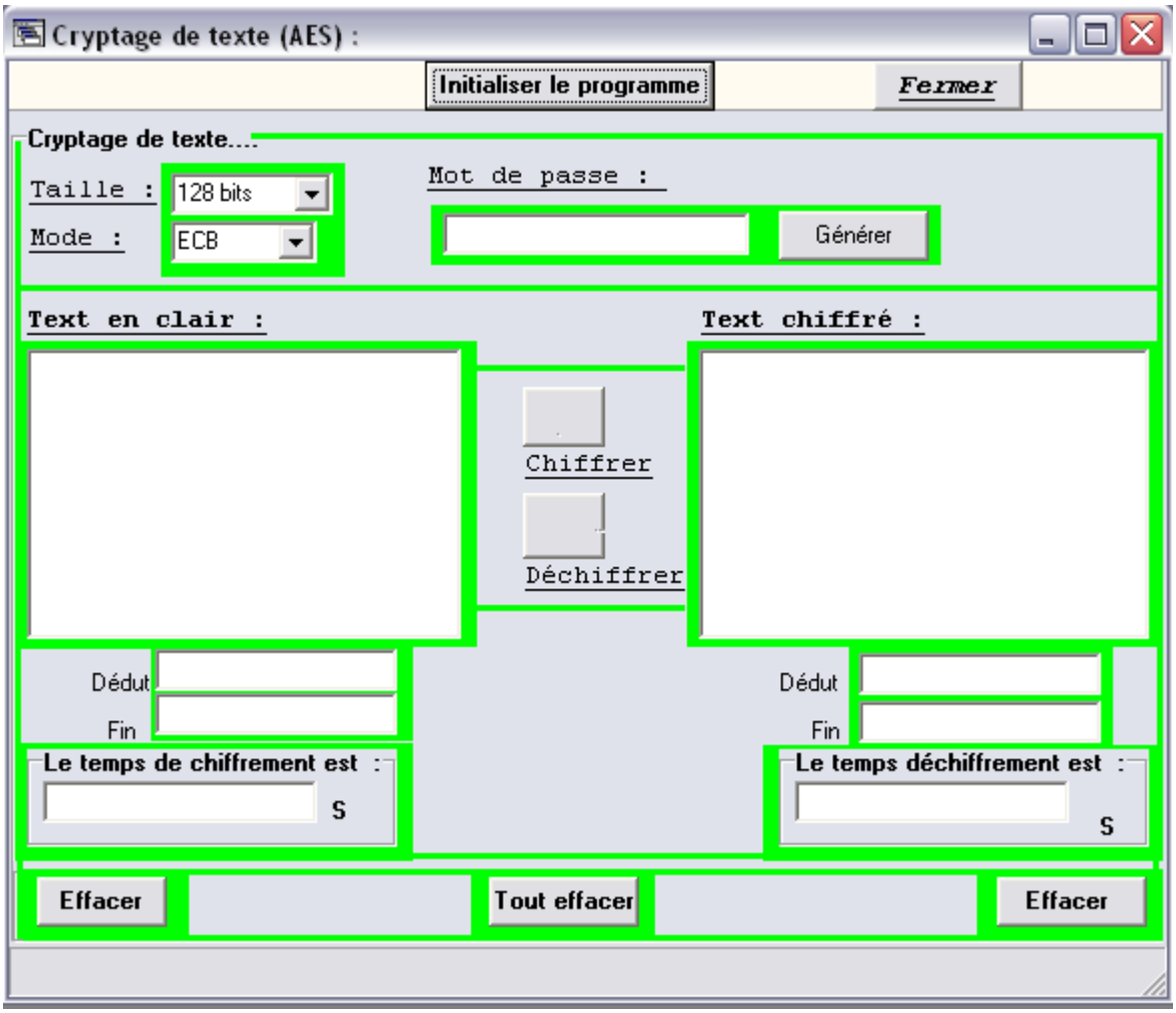
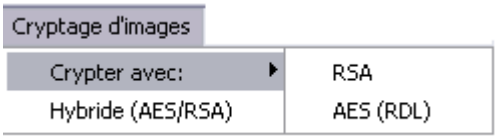


Fig.III.6. Chiffrement/déchiffrement d'un texte avec AES

c. Le menu 'Cryptage de fichiers' :



Le menu '**Cryptage de fichiers'** est réservé au Chiffrement/Déchiffrement de fichiers.

Comme pour le chiffrement d'un texte, ce menu et ses sous menus RSA, AES (RDL) permettent le choix de la méthode de chiffrement que nous souhaitons utiliser afin d'ouvrir des fenêtres dédiées au :

- Chiffrement hybride.
- Chiffrement à clé publique (RSA).
- Chiffrement à clé secrète (AES).

Voir la figure suivante :

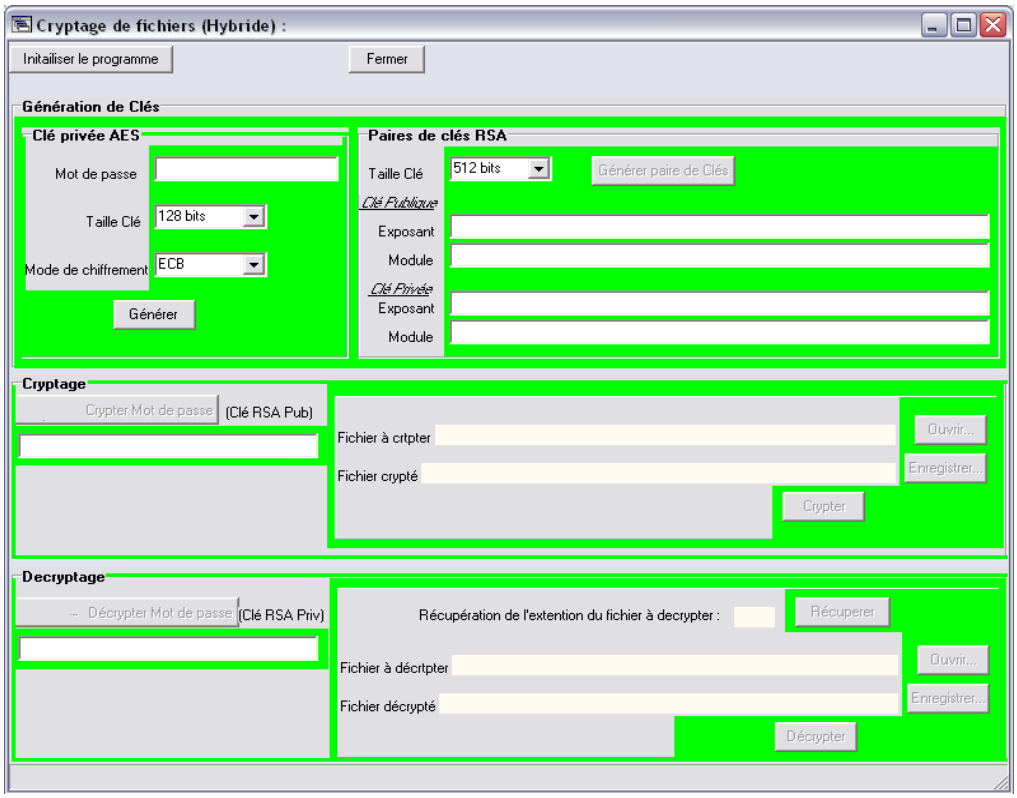


Fig.III.7. Chiffrement Hybride (AES/RSA) d'un fichier.

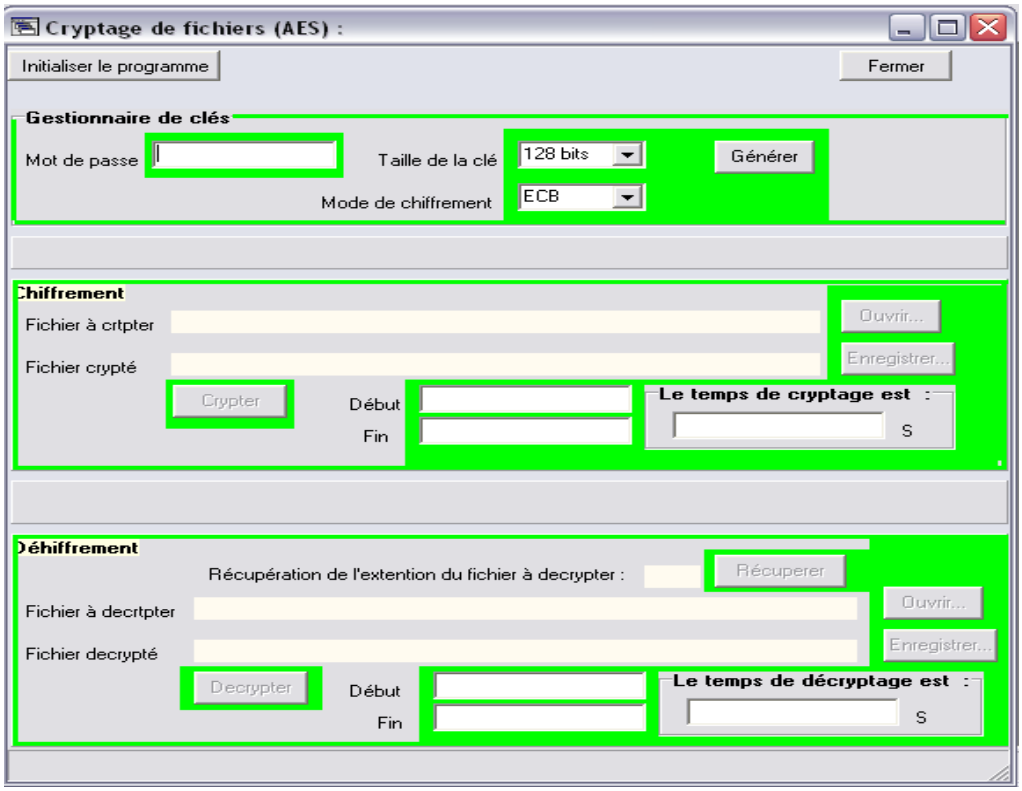


Fig.III.8. Chiffrement/déchiffrement d'un fichier avec AES

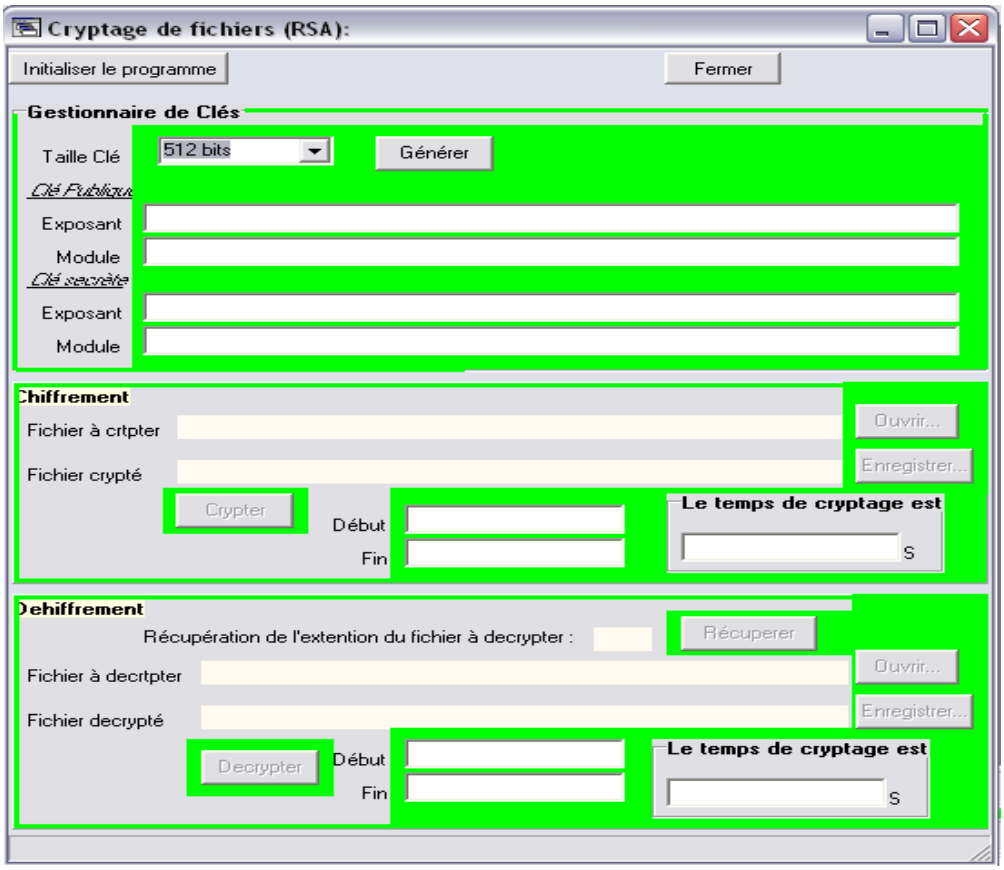


Fig.III.9. Chiffrement/déchiffrement d'un fichier avec RSA

d. Le menu 'Cryptage d'images' :

Voir la figure suivante :

Le menu '**Cryptage d'images'** est destiné au Chiffrement/Déchiffrement d'images bitmaps au niveau de gris.

Les sous menus RSA, AES (RDL) permettent le chois d'une méthode :

- Chiffrement à clé publique.
- Chiffrement à clé secrète.

Comme pour toute fenêtre illustrée précédemment, ces deux sous menus affichent deux volets dans la fenêtre principale pour l'ajustement des paramètres de chiffrement. Voir la figure suivante :

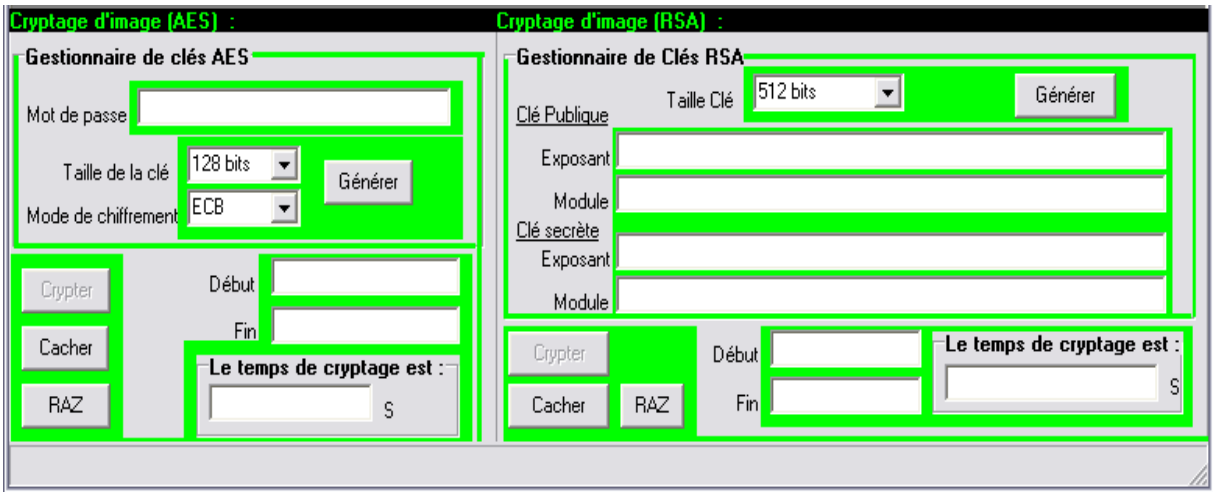
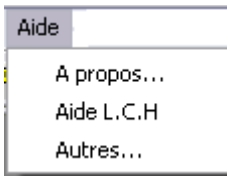


Fig.III.10. Paramètres de chiffrement d'une image

e. Le menu Aide :



- propos: Affiche une fenêtre contient les informations sur le logiciel.
- Aide L.C.H : lance un fichier d'aide sous forme d'un document HTML.
- Autres : Donne des informations supplémentaires.

Fig.III.11. Fenêtre A propos du logiciel L.C.H

3.2. Barre d'outils (Barre des boutons de contrôle) :

Les boutons de contrôle sont des raccourcis permettant un accès rapide aux différentes fenêtres de l'application ou l'ouverture d'un utilitaire système :

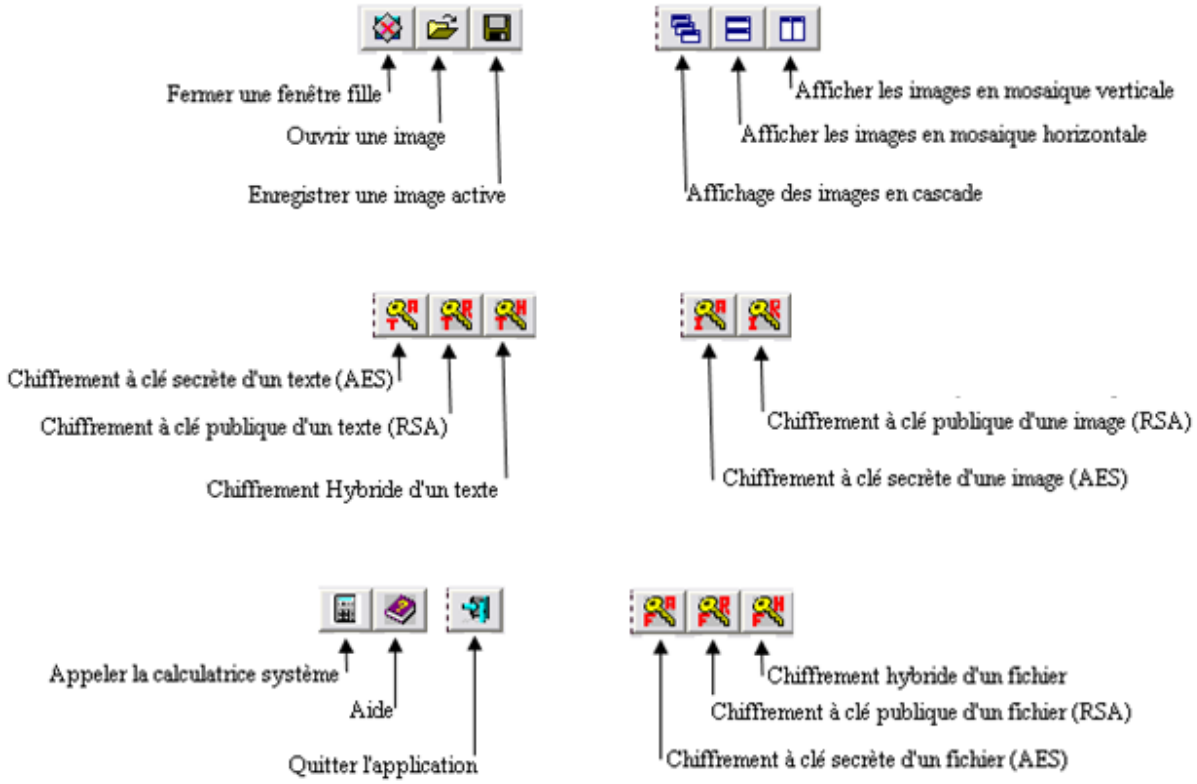


Fig.III.12. Boutons de contrôle.

Chapitre IV

1. Introduction :

Apres la mise en place du L.C.H, nous allons procéder aux tests du logiciel. Ces tests se feront sur du texte saisit sur le logiciel (à l'image d'un mail), sur des fichiers de format quelconque, ainsi que sur une image en niveau de gris.

Les tests on été effectués sur un ordinateur de bureau, avec la configuration suivante :

- µProcesseur Intel Pentium 4 à Double cœur, cadencé à 3,0 GHz.
- Une mémoire vive de 1 Go de type DDR2.

2. Sur un texte :

2. a. AES (128 bits):

Phrase de passe pour générer la clé de 128 bits: wxcvbn

Mode de chiffrement : ECB

- **Chiffrement :**

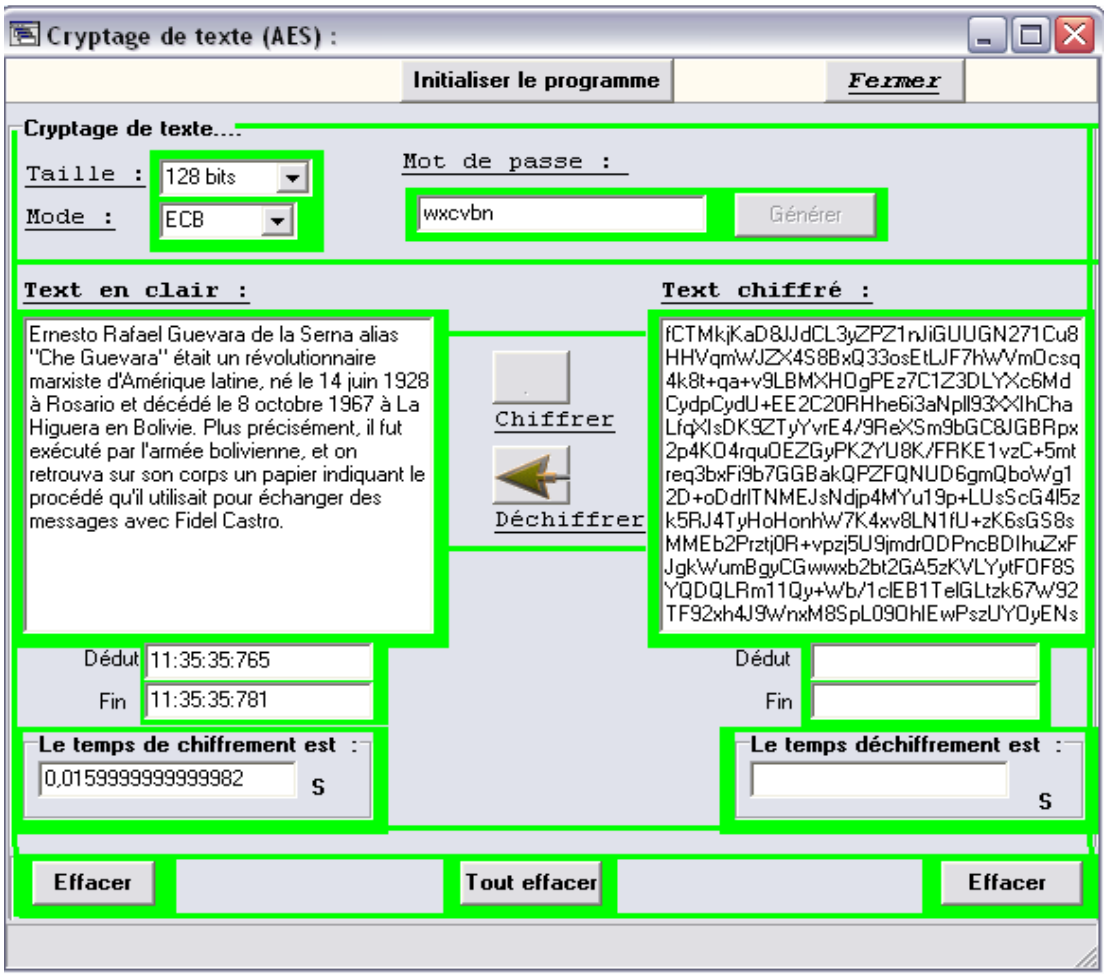


Fig.IV.1. Chiffrement d'un texte avec AES en mode ECB

- **Déchiffrement :**

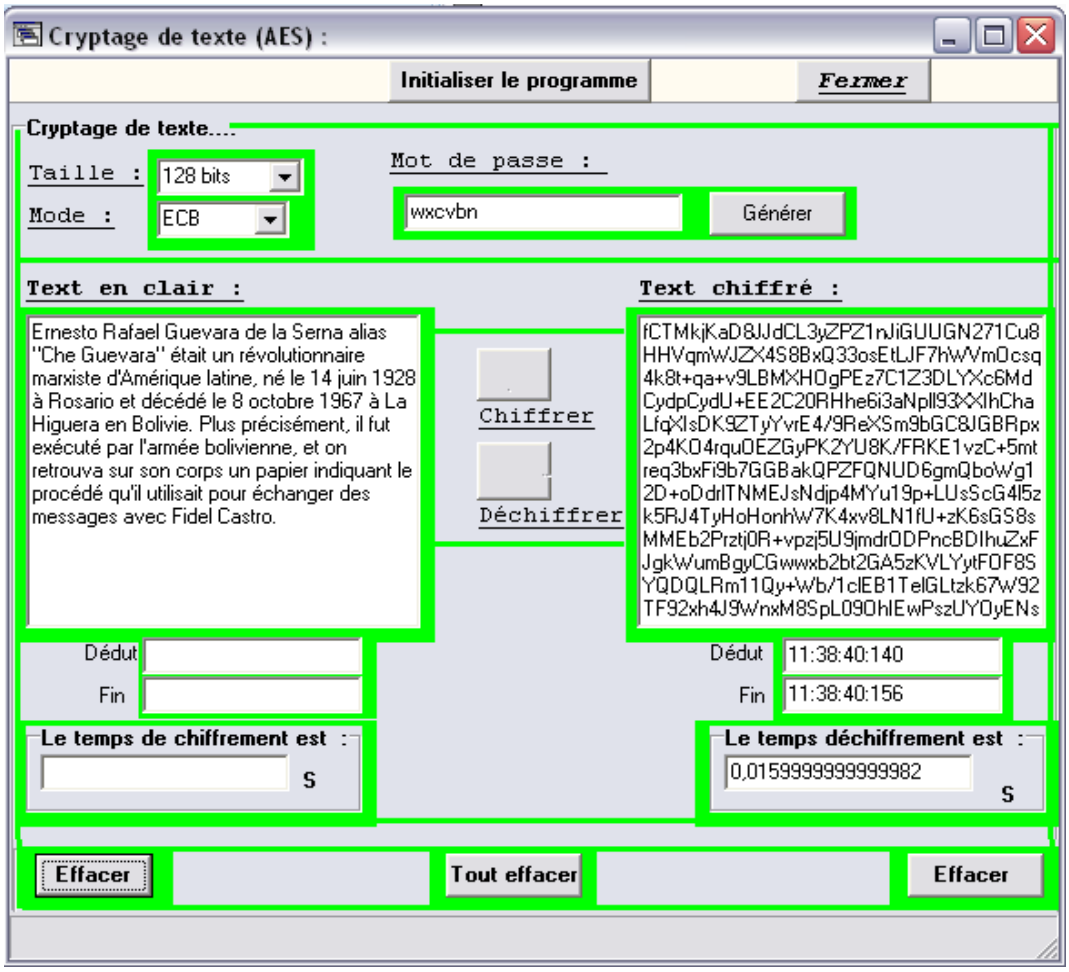


Fig.IV.2. Déchiffrement d'un texte avec AES en mode ECB

On retrouve bien le texte clair à partir du texte chiffré précédemment.

Nous avons relevé qu'avec le mode ECB, on obtient toujours le même chiffre à partir du même texte avec la même clé. Une attaque à texte clair connu est envisageable.

Pour améliorer le niveau de sécurité nous proposant :

1. Utiliser le mode CBC.
2. Générer une clé plus longue (192, 256 bits).

2. b. RSA (128 bits):

- **Génération de clés :**

Clé de chiffrement (clé publique):

Module (Hexa) : n = CB77D7D81DEAA5D1A0CA63C25905C5A1

Exposant (Hexa) : e = C527

Clé de déchiffrement (clé privée):

Module (Hexa) : n = CB77D7D81DEAA5D1A0CA63C25905C5A1

Exposant (Hexa) : d = 0D00A26D0196935BAA8C19C5D8161767

On peut également générer une paire de clés de (RSA-256, 512, 768, 1024 bits).

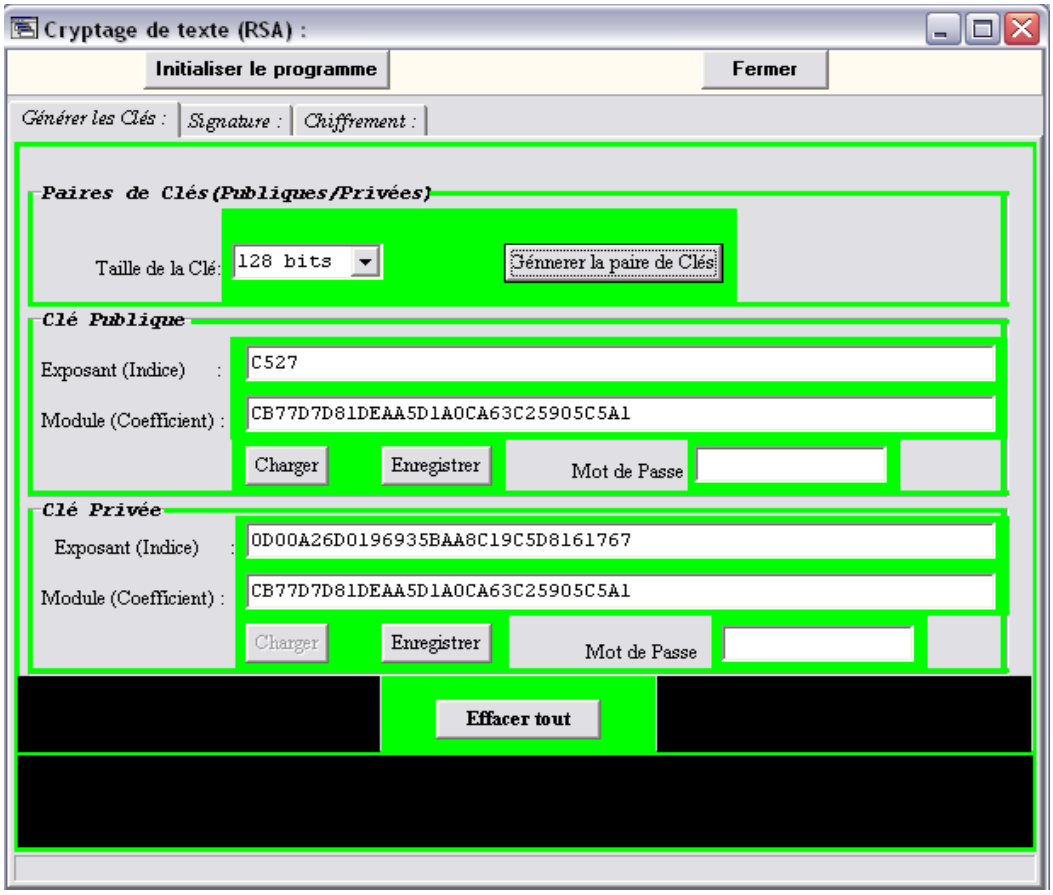


Fig.IV.3. Interface de gestion des clés RSA

Comme l'indique la figure ci-dessus, les clés (publique et privée) peuvent être sauvegardées sous forme de deux fichiers qui seront exploitables dans le chiffrement/déchiffrement et aussi dans la transmission de la clé.

- **Chiffrement :**

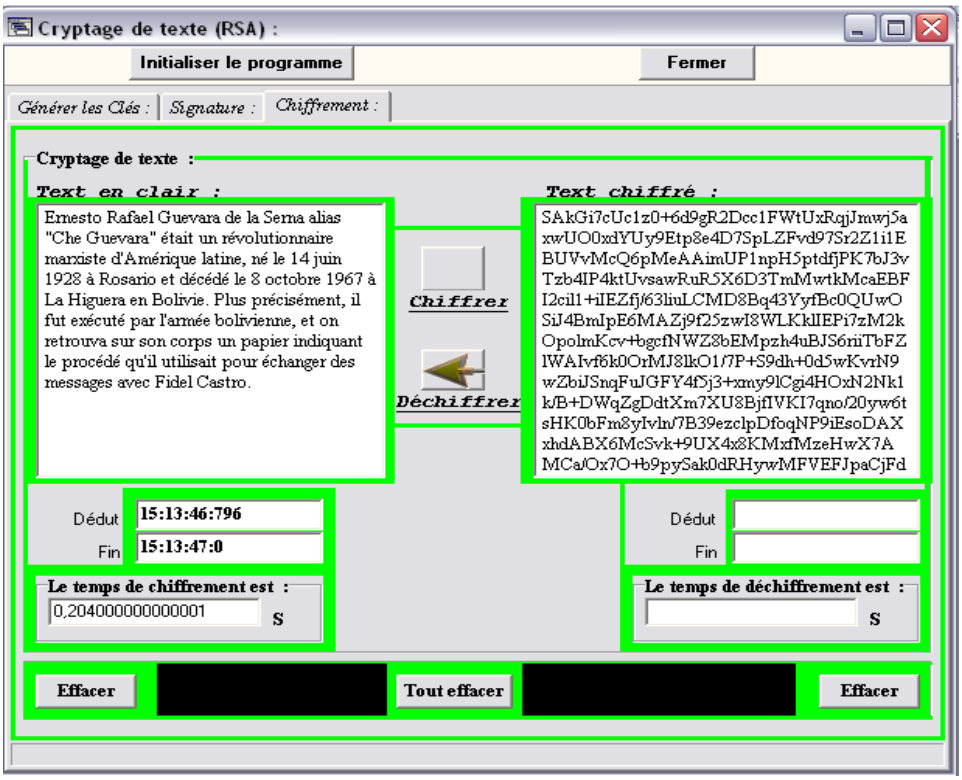


Fig.IV.4. Chiffrement RSA sur du texte

- **Signature (RSA 256 bits) :**

Clé de signature (clé privée):

Module (Hexa) :

n = 53EACFED951F134BFE29FA2F13F7F172E79151CEBC52A7F1C7AE71ADB2B6FE84

Exposant (Hexa) :

e = 558AAF4A4A757B67E87E691B13F7332A115639A2951EBFB68672C6324ED49C26

Clé de vérification (clé publique):

Module (Hexa) :

n = 53EACFED951F134BFE29FA2F13F7F172E79151CEBC52A7F1C7AE71ADB2B6FE84

Exposant (Hexa) : e = A51C

Méthode de hachage : SHA-1

On peut également Générer une paire de clés pour la signature (RSA-512, 768, 1024 bits)

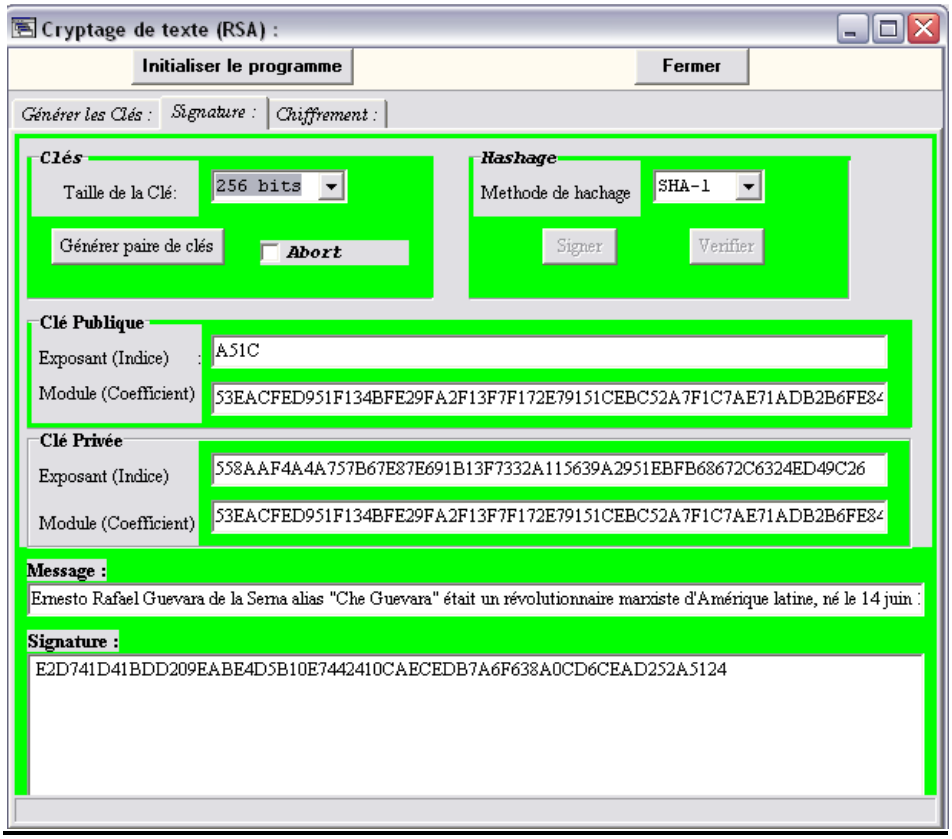


Fig.IV.5. Signature

- **Déchiffrement :**

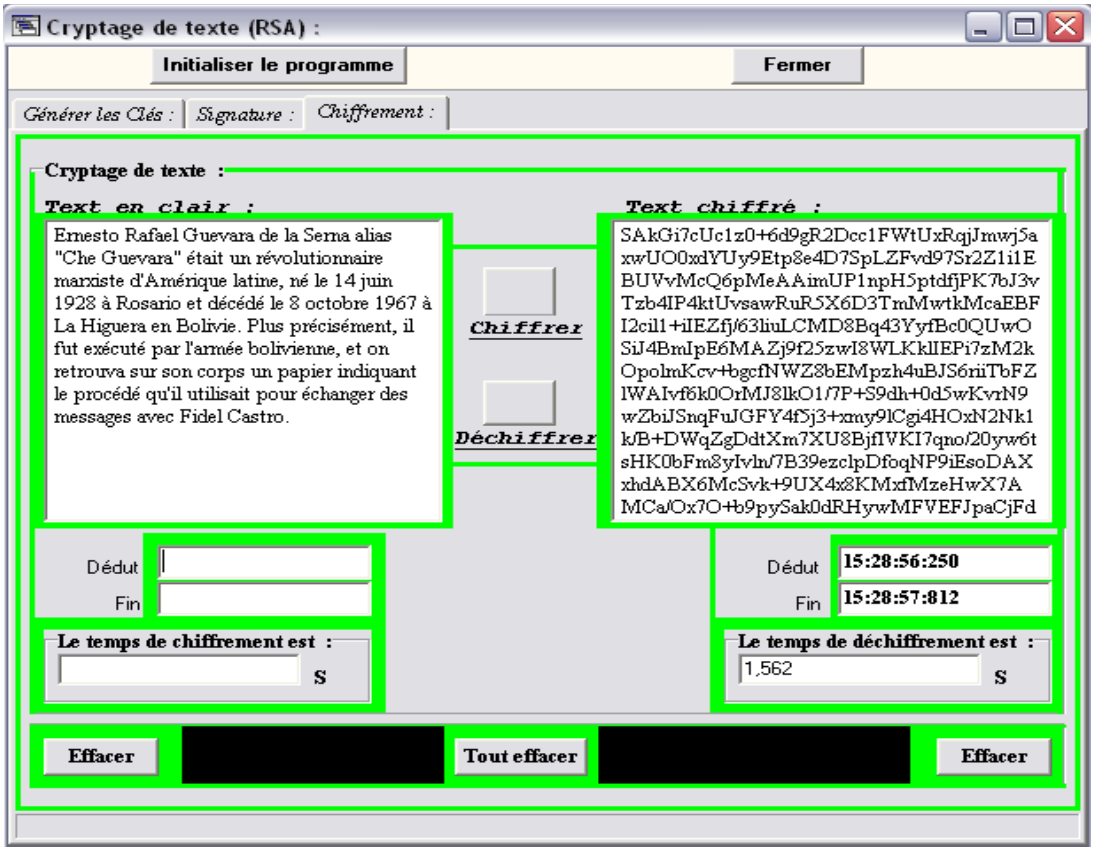


Fig.IV.6. Déchiffrement du texte avec l'RSA

- **RSA Complet (Chiffrement/Déchiffrement, Signature) :**

Cette partie englobe dans un même projet les fonctions de RSA (Chiffrement, signature, déchiffrement d'un texte en respectant toutes les phases de RSA précédemment citées (Génération de clés de 'Chiffrement/Déchiffrement' et de signature, choix de la fonction de hachage), cette étape est illustrée dans la figure ci-dessous :

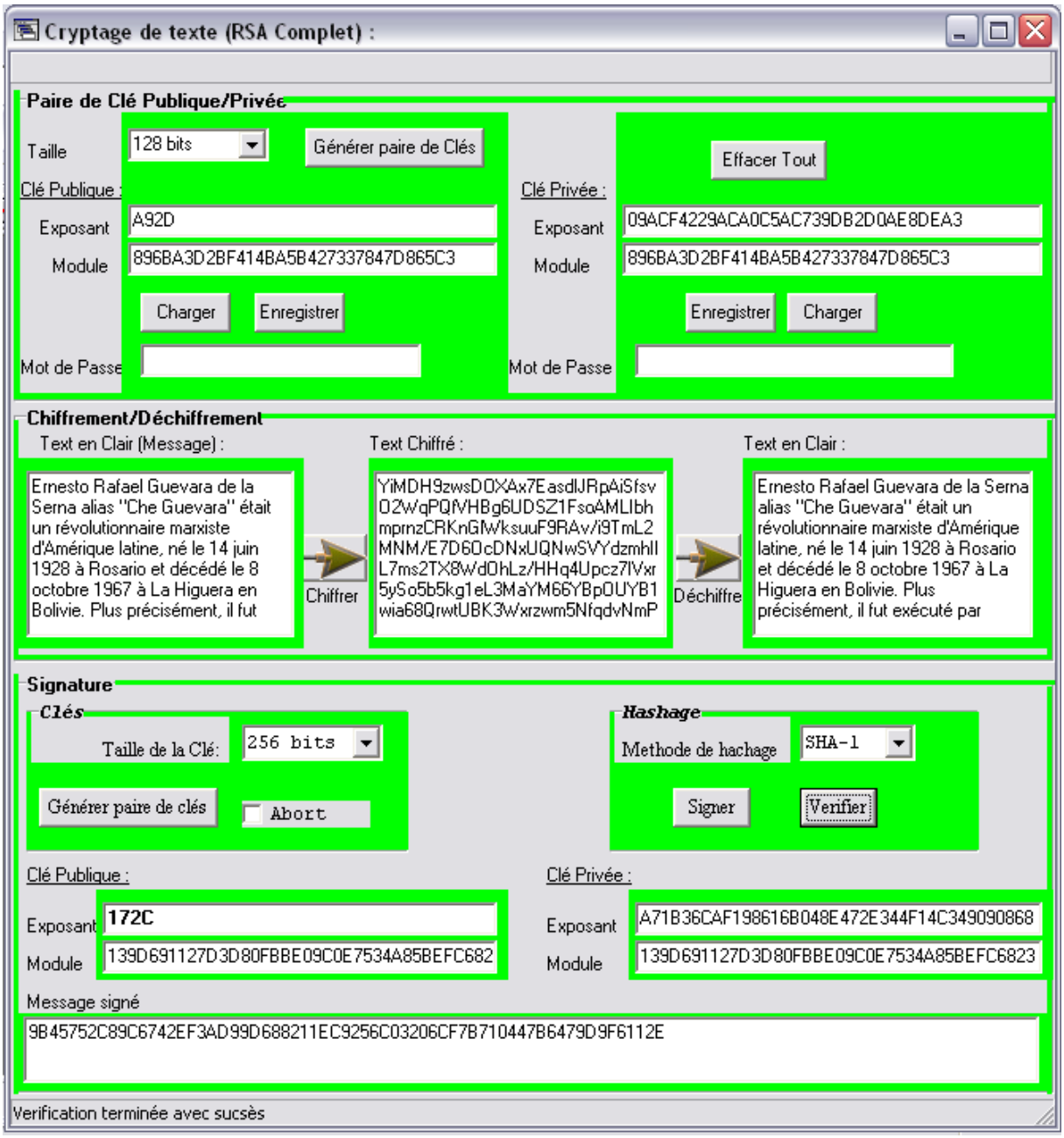


Fig.IV.7. Chiffrement/Déchiffrement, Signature d'un texte avec RSA

3. Sur un fichier :

Le logiciel L.C.H peut chiffrer/déchiffrer n'importe quel type de fichier (.txt, .doc, .bmp, …). Ces tests sont effectués sur un fichier BMP de taille de 65Ko environ, en respectant la procédure de chiffrement et de déchiffrement mentionnée dans le cas d'un texte :

3.a. L'AES (128 bits) :

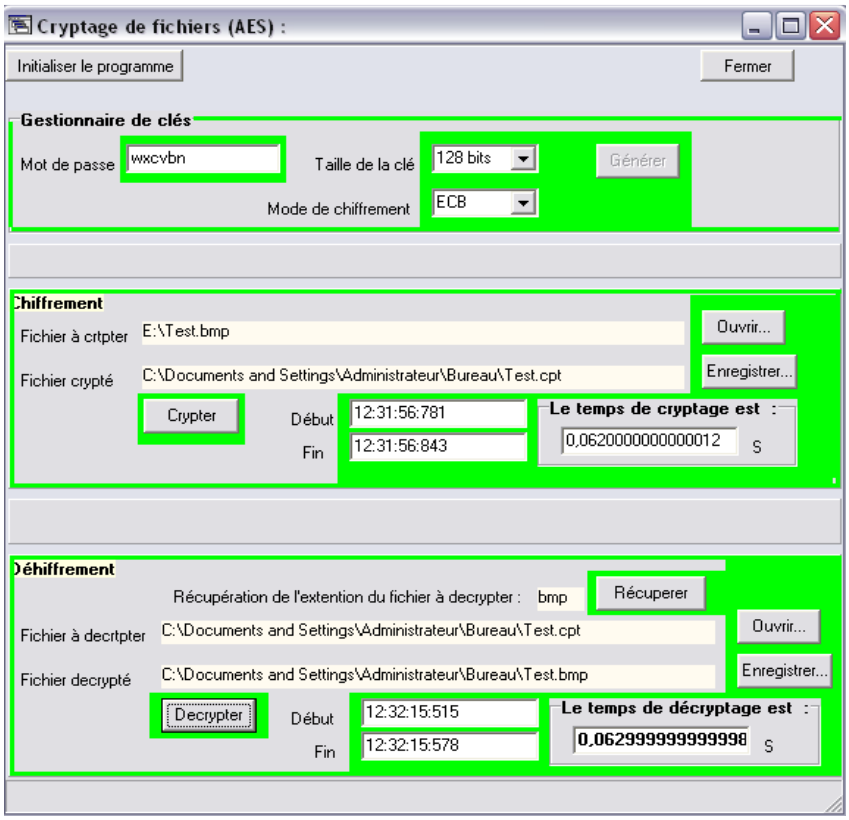


Fig.IV.8. Chiffrement/Déchiffrement du fichier avec AES

3.b. Le RSA (128 bits) :

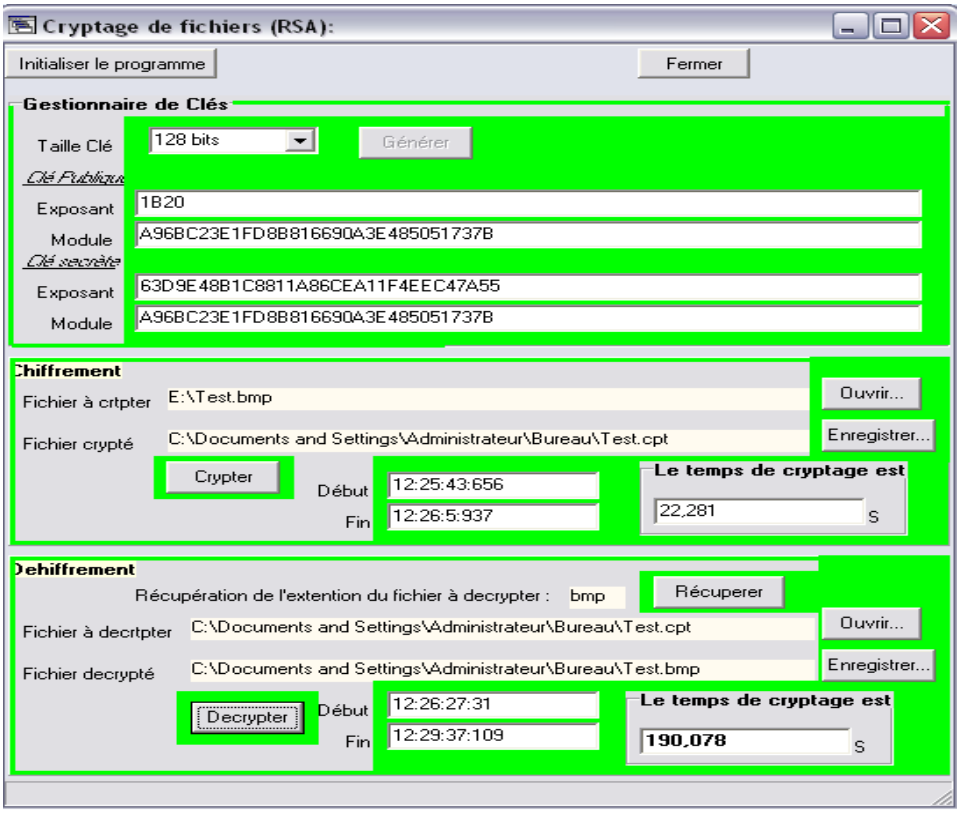


Fig.IV.9. Chiffrement/Déchiffrement du fichier avec RSA

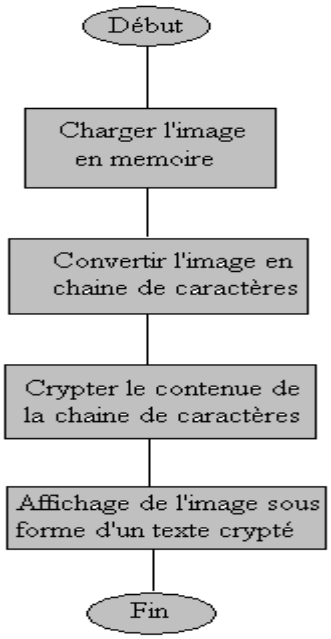
4. Sur une image :

Le mécanisme de chiffrement/déchiffrement d'une image adopté par le logiciel L.C.H repose sur le principe de chiffrement/déchiffrement d'un texte.

Les figures **Fig.IV.10.** et **Fig.IV.11.** représentent respectivement les organigrammes de chiffrement et de déchiffrement d'une image :

4. a. Principe de chiffrement/Déchiffrement d'une image :

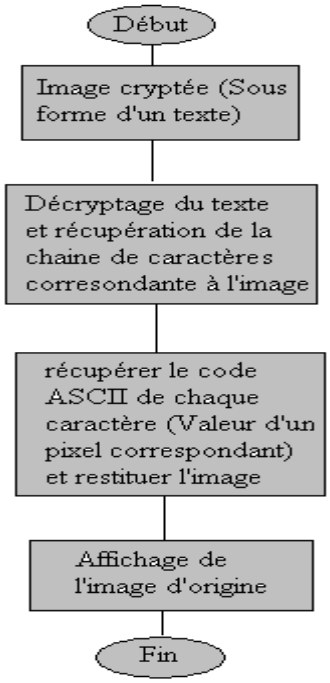
- **Chiffrement :**



- La 1ere étape : consiste à charger l'image en mémoire, pixel par pixel (valeurs 0…255).
- La 2ème étape : récupérer le caractère qui correspond à chaque pixel en formant une chaine de caractères.
- La 3ème étape : crypter la chaine de caractères obtenue précédemment.
- En fin : Affichage du résultat.

Fig.IV.10. Procédure de chiffrement d'une image

- **Déchiffrement :**



- La 1ere étape : déchiffrement du texte, à fin de récupérer la chaine de caractère qui correspond à l'image dont chaque caractère représente un pixel.
- Restituer le code ASCII de chaque caractère, puis construire l'image d'origine
- Affichage de l'image (décryptée).

Fig.IV.11. Procédure de déchiffrement d'une image

4.b. L'AES (128 bits) :

Phrase de passe pour générer la clé de 128 bits: wxcvbn

Mode de chiffrement : ECB

- **Chiffrement :**

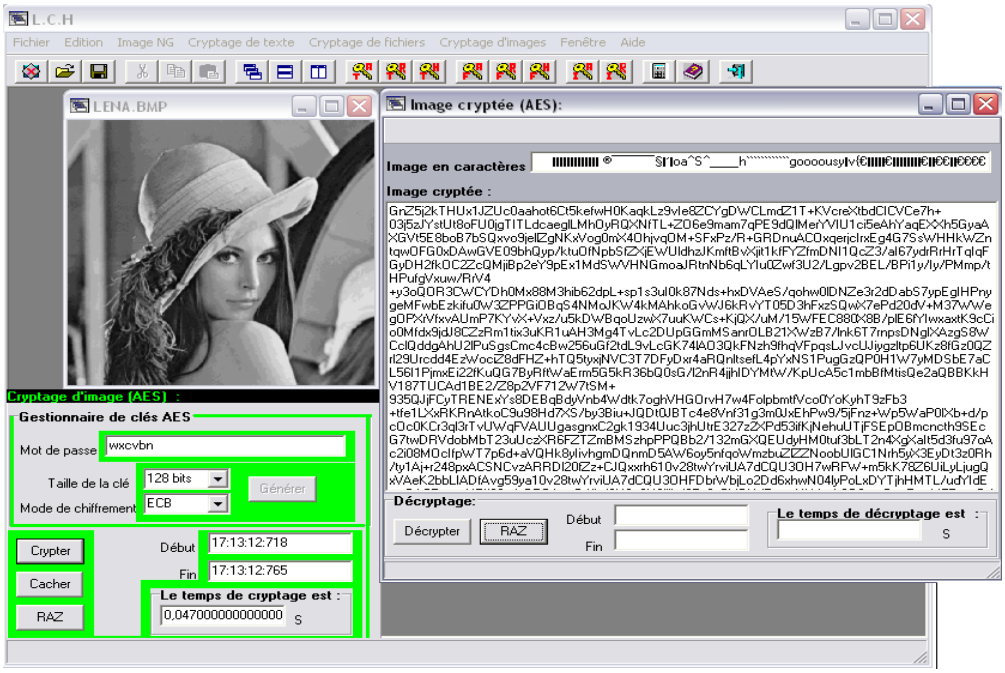


Fig.IV.12. Chiffrement de l'image avec AES

- **Déchiffrement :**

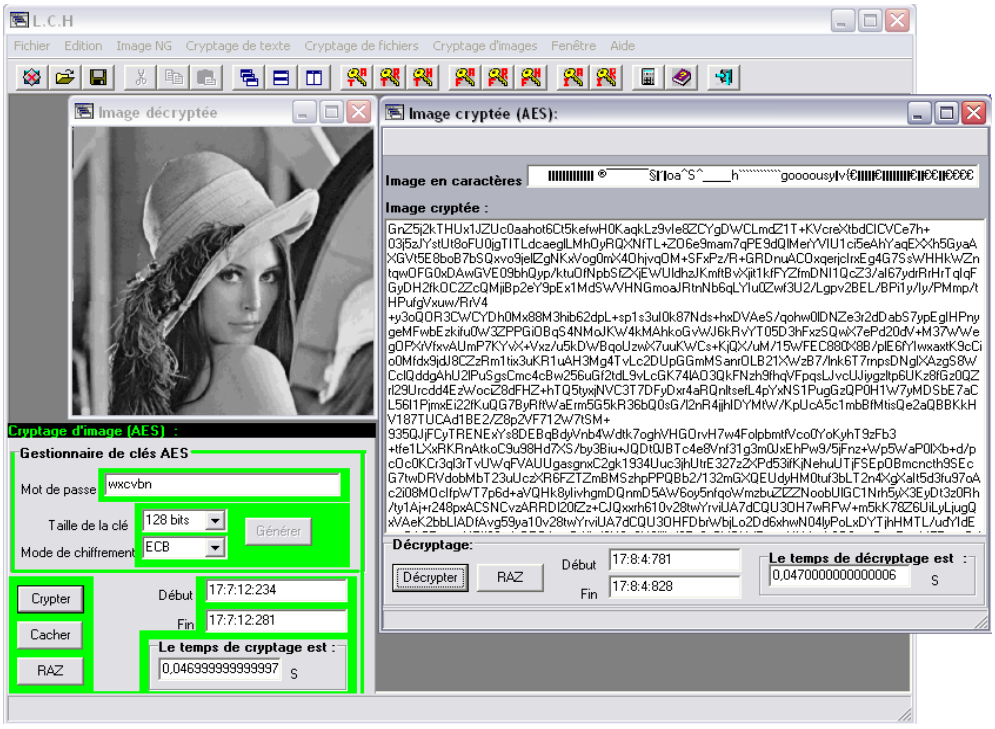


Fig.IV.12. Déchiffrement de l'image avec AES

4. c. Le RSA (128 bits):

Clé de chiffrement de (clé publique):

Module (Hexa) : n = FD4491C3FEC4796F182634D333FCF9D4

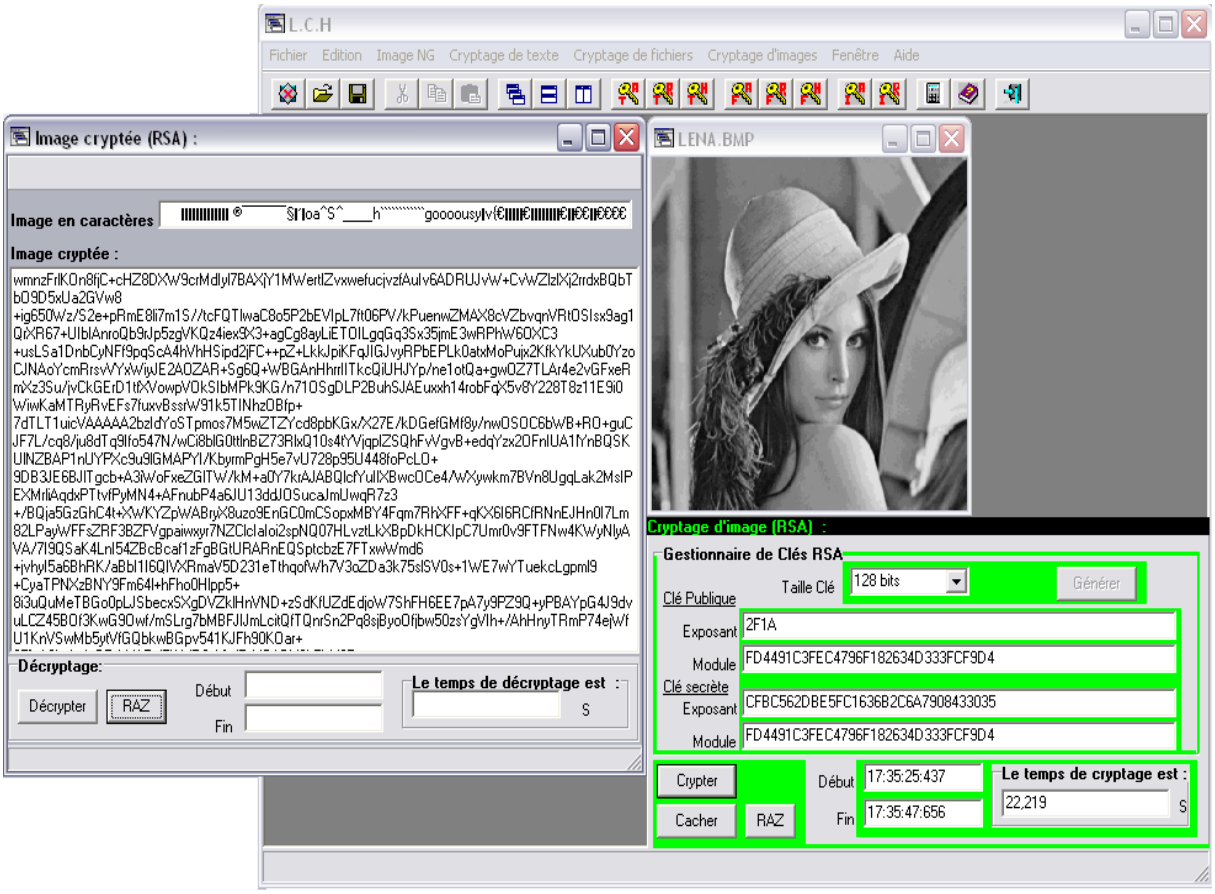
Exposant (Hexa) : e = 2F1A

Clé de déchiffrement (clé privée):

Module (Hexa) : n = FD4491C3FEC4796F182634D333FCF9D4

Exposant (Hexa) : d = CFBC562DBE5FC163682C6A7908433035

- **Chiffrement :**



FigIV.13. Chiffrement de l'image avec RSA

- **Déchiffrement :**

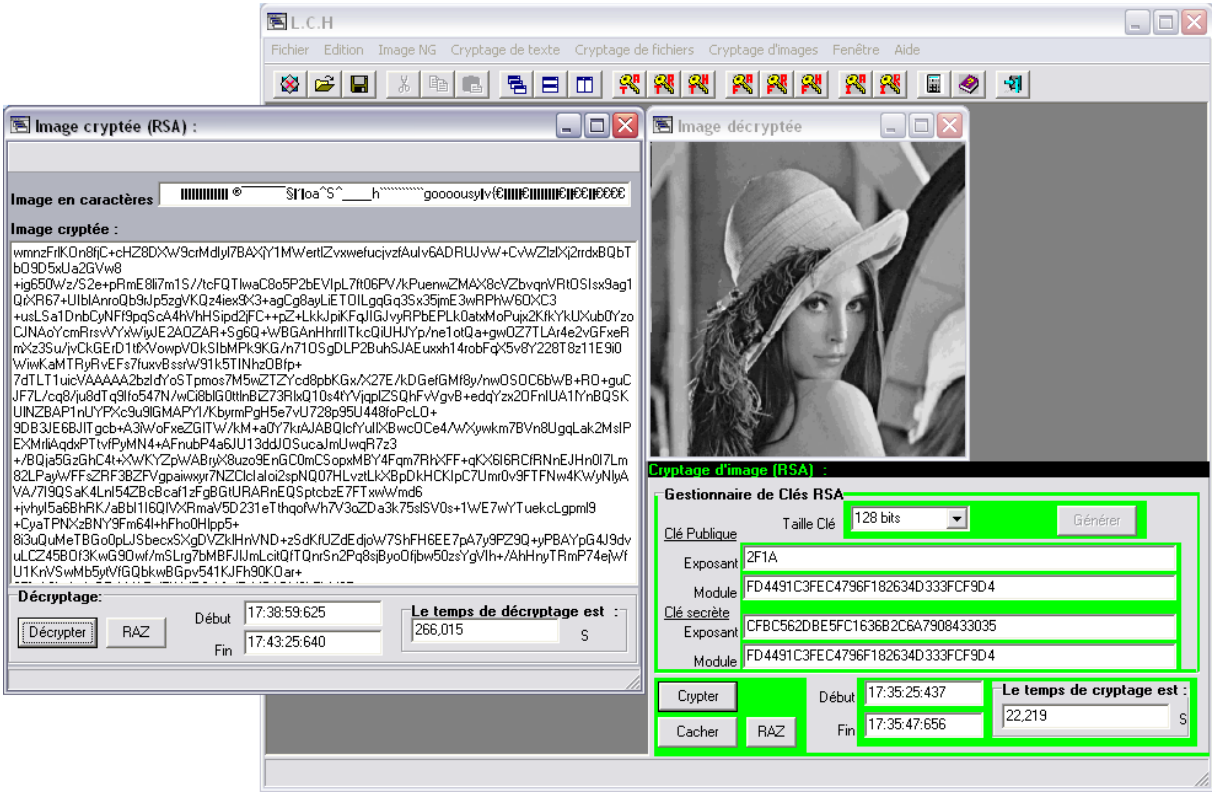


Fig.IV.14. Déchiffrement de l'image avec RSA

5. Tableau des résultats :

	Texte	Fichier (.bmp, 65 Ko)	Image BMP en NG, 65 Ko
AES	Chiff : 0,016 s Déchiff : 0,016 s	Chiff : 0,062 s Déchiff : 0,063 s	Chiff : 0,047 s Déchiff : 0,047 s
RSA	Chiff : 0,204 s Déchiff : 1,562 s	Chiff : 22,281 s Déchiff : 190,078 s	Chiff : 22,219 s Déchiff : 266,015 s

Le chiffrement AES est un chiffrement par bloc de 128 bits. Les clés peuvent être de 128 bits, 196 bits ou 256 bits. C'est un chiffrement de type réseau de permutations et substitutions dont le nombre de tours dépend de la longueur de la clé (Voir chapitre II). Le principal avantage de l'AES réside dans sa rapidité en chiffrement et en déchiffrement.

Le RSA est un algorithme très lent comparé aux algorithmes à clé secrète (dans notre cas l'AES). En pratique RSA n'est pas utilisé pour le chiffrement de données, mais il est destiné pour La signature, le chiffrement d'une clé de session d'un algorithme symétrique (Clé AES), c'est le principe d'un système hybride comme le PGP (Voir chapitre II).

6. Chiffrement hybride (AES/RSA) :

6. a. Sur un texte :

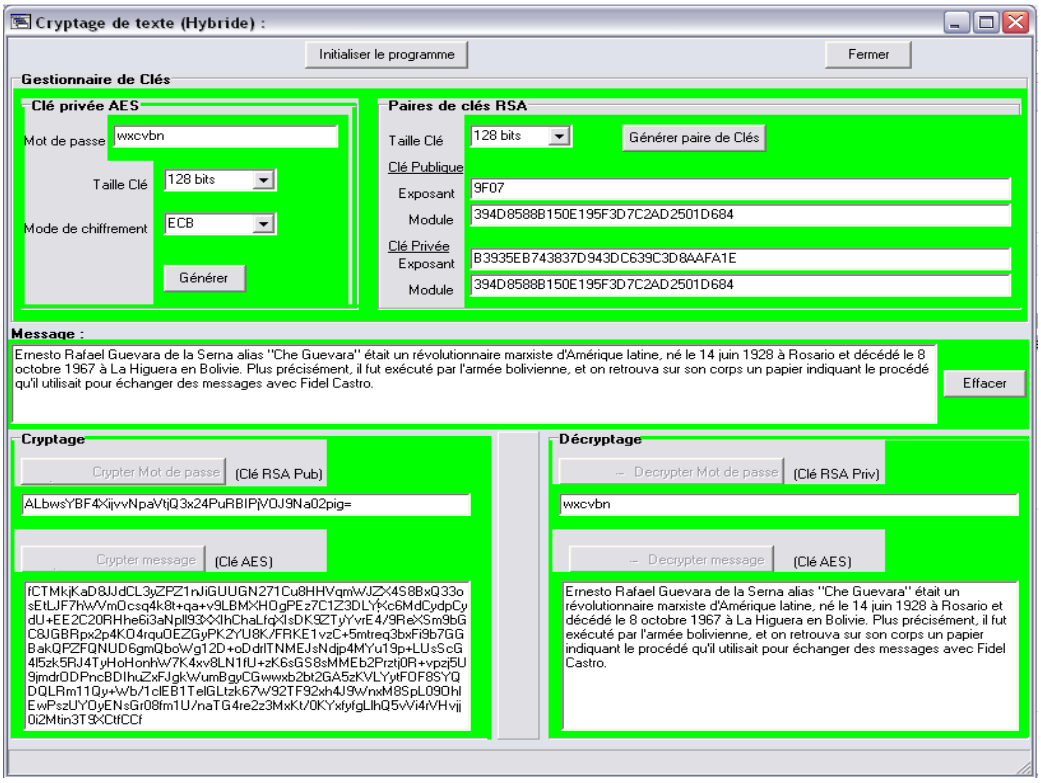


Fig.IV.15. Chiffrement Hybride sur un texte

6. b. Sur un fichier :

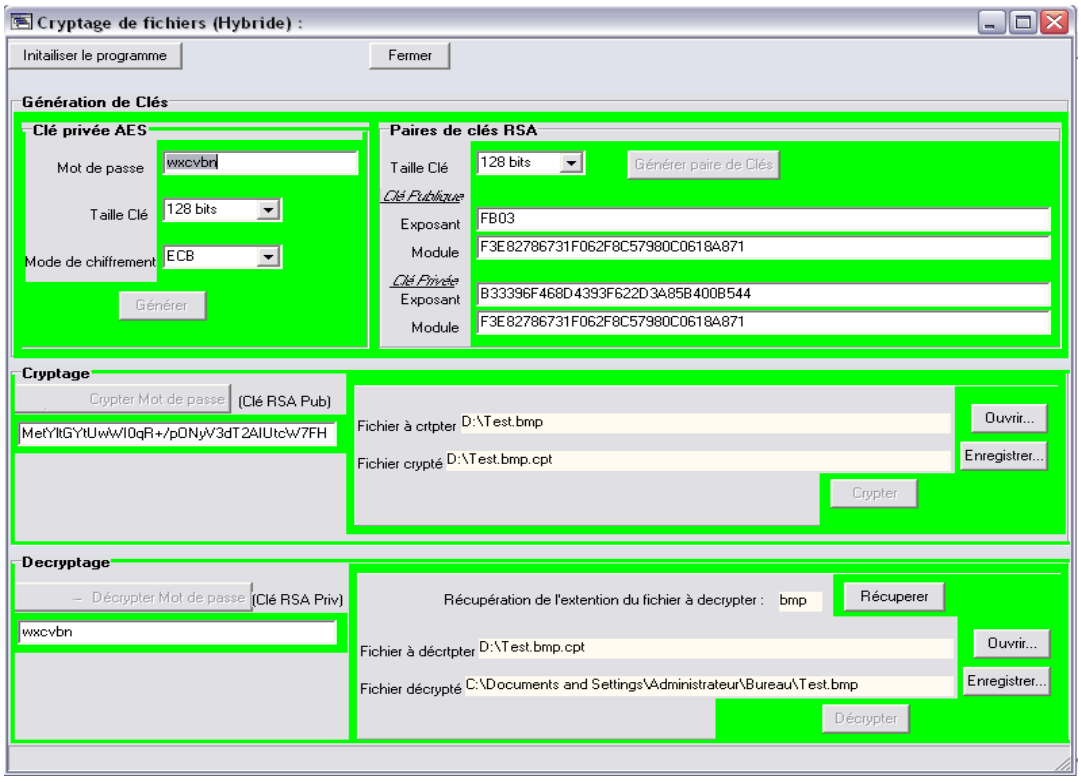


Fig.IV.16. Chiffrement Hybride sur un fichier

7. Avantages et inconvénients du L.C.H :

A travers les différents tests effectués, nous pouvons conclure que le logiciel fonctionne correctement en matière de chiffrement/déchiffrement et de signature et qu'il offre une sécurité acceptable à notre niveau.

Ses avantages sont les suivants :

- Il repose sur le principe de chiffrement hybride.
- Possibilité de chiffrer des fichiers de format quelconque.
- Utilisation de deux systèmes de chiffrement robustes, l'AES et l'RSA.
- Utilisation d'une fonction assurant la génération aléatoire de clés :
    - Génération automatique pour l'RSA.
    - Génération suivant une phrase de passe pour l'AES.
- Protection des paires de clés RSA par mot de passe.
- Le L.C.H a une orientation pédagogique.

On a aussi relevé quelques inconvénients :

- Ne gère pas les communications réseau.
- Les clés RSA sont de tailles insuffisantes pour une sécurité optimale (on conseille de nos jours des clés d'au moins 2048 bits).
- Interface pas très intuitive.

.

Conclusion Générale

Ce travail nous à permis de découvrir la cryptographie et de prendre conscience de l'importance de cette dernière dans les sociétés d'aujourd'hui, où tout s'immatérialise.

En traversant l'histoire de la cryptographie, depuis le chiffre de César jusqu'à l'AES en passant par l'RSA, nous avons constaté l'énorme évolution de cette science, en grande partie grâce à l'avènement de l'informatique. La cryptographie dite moderne, au delà de la complexité de ses procédures, se base sur des principes mathématiques simplistes. Néanmoins, elle procure des niveaux de sécurité de plus en plus élevés.

Un autre constat que nous avons relevé vient confirmer le principe de Kirchhoff. On a remarqué que les concepteurs d'algorithmes cryptographiques optent de plus en plus pour la publication du fonctionnement interne de leurs algorithmes, laissant le secret lié uniquement à la clé de chiffrement. Mieux encore, un logiciel libre donc au code source ouvert, en l'occurrence GnuPG, vient à être de plus en plus utilisé à la place de PGP, depuis que ce dernier ait changé de philosophie optant pour un système fermé.

Notre objectif principal était d'assurer la confidentialité des données en utilisant un chiffrement hybride, et il a été atteint. Nous avons choisi l'AES pour assurer la fonction de chiffrement des données elles mêmes et l'RSA pour assurer celle de l'acheminement sécurisé de la clé, ce choix n'était pas fortuit, ces algorithmes sont, à l'heure actuelle, parmi les meilleurs dans leurs catégories respectives.

Pendant nos tests on a bien relevé la lenteur de l'RSA par rapport à l'AES, ce qui justifie amplement la fonction allouée à chacun dans le L.C.H. Ceci dit, l'RSA offre plus de fonctionnalités intéressantes telles que la signature des données afin de vérifier leur intégrité.

Même si le L.C.H accomplis correctement sa tâche principale, il peut-être largement amélioré. D'abord, en matière de sécurité, en augmentant la taille des clés RSA. En suite, en fonctionnalités, en lui intégrant un module qui permettrait la transmission des messages chiffrés à travers un réseau (à l'image d'un logiciel de messagerie instantanée).

Annexes

La cryptographie quantique [6]

- **Origine :**

Dans l'histoire, beaucoup de codes étaient considérés comme inviolables mais ont été cassés (Vigenère, Enigma, ...). Les systèmes actuels à clés publiques sont-ils alors menacés ? Quoi qu'il en soit, on se tourne maintenant vers une nouvelle sorte de cryptographie, ne reposant pas sur l'aspect déterministe des mathématiques, mais sur l'aspect probabiliste de la physique, donc sur les lois-mêmes de la nature, et non pas sur l'ignorance de l'homme sur certains points. Ces travaux se basent sur les recherches de Stephen Wiesner dans les années 60, qui inventa le principe de cryptographie quantique.

- **Mécanisme :**

Le message est transmis par des photons polarisés. Pour simplifier, on supposera qu'il y a quatre sens de polarisation possibles : vertical |, oblique droit /, horizontal -, et oblique gauche \. Un photon polarisé passe à travers un filtre polarisant de même sens que lui, ne peut franchir un filtre de sens perpendiculaire à lui, et passe le filtre avec une probabilité 1/2 s'ils est décalé de 45° par rapport à lui. Ces situations sont regroupées dans le tableau suivant :

Polarisation du photon	**Polarisation du filtre**	**Passage du photon ?**
|	|	OUI
	/	1 chance sur 2
	-	NON
	\	1 chance sur 2
/	|	1 chance sur 2
	/	OUI
	-	1 chance sur 2
	\	NON
-	|	NON
	/	1 chance sur 2
	-	OUI
	\	1 chance sur 2
\	|	1 chance sur 2
	/	NON
	-	1 chance sur 2
	\	OUI

De plus, si un photon passe à travers le filtre, il ressort nécessairement dans la même orientation que lui.

Remarque : on peut coder le 1 et le 0 selon deux schémas :

	1	**0**
Schéma linéaire +	|	-
Schéma diagonal x	/	\

1) L'émetteur commence par transmettre une suite aléatoire, suffisamment longue, de bits (0 ou 1) avec des schémas (linéaires ou diagonals) aléatoires, en en prenant soigneusement note.

2) Le récepteur a le choix pour chaque bit de choisir un récepteur (filtre) diagonal ou linéaire. Il le fait donc au hasard, lui aussi en en prenant note, et par conséquent interprète mal de nombreuses polarisations (1 fois sur 2).

3) L'émetteur transmet alors au destinataire, par voie publique, le type de schéma utilisé pour chaque bit (mais pas la valeur de ces bits). Le destinataire lui communique en échange, toujours par voie publique, les fois où il a utilisé le récepteur adapté pour ces schémas. Les bits où ceci s'est produit constituent la clé.

4) L'émetteur et le destinataire connaissent donc tous les deux une clé aléatoire, avec laquelle ils peuvent coder et s'échanger des messages avec des méthodes traditionnelles de cryptographie.

Si la clé convenue est aussi longue que les messages qu'ils vont s'échanger, le code est totalement incassable. En effet, personne ne peut connaître la clef car si les photons étaient mesurés par un espion, lui aussi ne pourrait choisir le bon filtre à chaque fois pour mesurer leur polarisation. Il modifierait donc forcément la polarisation de certains photons et l'émetteur et le destinataire s'en rendraient facilement compte en adoptant dans la phase 3) un code correcteur d'erreur ajouté à la fin de leur transmission.

Corps fini [12]

En mathématiques et plus précisément dans la branche de la théorie de Galois, un corps fini est un corps (commutatif) dont le cardinal est fini. À isomorphisme près, un corps fini est entièrement déterminé par son cardinal qui est toujours de la forme p^n, une puissance d'un nombre premier. Ce nombre premier n'est autre que sa caractéristique et le corps se présente comme l'unique extension simple du corps premier Z/p.Z de dimension n.

Les applications sont essentiellement la théorie algébrique des nombres où les corps finis apparaissent comme une structure essentielle à la géométrie arithmétique.

Pour plus de détails, visitez le portail des mathématiques sur : http://fr.wikipedia.org

Congruences modulo n [10]

Définition
Soit n un entier strictement positif. On dit que deux nombres entiers a et b sont congrus modulo n si b - a est divisible par n. On note alors :

$$a \equiv b \pmod{n} \text{ ou } a = b \bmod n.$$

Exemples : $2 = 17 \bmod 5$; $12 \neq 10 \bmod 11$; $13 \equiv 23 \pmod{2}$

Théorème
a et b sont congrus modulo n si et seulement si a et b ont le même reste de la division par n.

Propriétés
- Compatibilité de la loi + avec la congruence modulo n :
  $a \equiv b \pmod{n}$ et $c \equiv d \pmod{n}$ ssi $a + c \equiv b + d \pmod{n}$.
- Compatibilité de la loi $\cdot$ avec la congruence modulo n :
  $a \equiv b \pmod{n}$ et $c \equiv d \pmod{n}$ ssi $a \cdot c \equiv b \cdot d \pmod{n}$.
- Pour tout entier strictement positif k,
  $a \equiv b \pmod{n}$ ssi $a^k \equiv b^k \pmod{n}$.

Définitions
Soit n un entier strictement positif. Soit a dans $\mathbb{Z}$
On appelle *classe de a modulo n* l'ensemble des entiers congrus à a modulo n.

On peut montrer que la relation de congruence est une relation d'équivalence sur $\mathbb{Z}$ et qu'elle définit une partition de $\mathbb{Z}$ en n classes distinctes, les classes de 0, 1, . . ., n-1. On désigne par
$\mathbb{Z}/n\mathbb{Z}$ l'ensemble quotient formé de ces n classes.

Les nombres premiers

Les nombres premiers sont les entiers naturels p n'ayant pour diviseurs que 1 et p, ou les entiers relatifs p n'ayant pour diviseurs que 1, - 1, p et - p.
Tout nombre entier se décompose sous la forme unique d'un produit de facteurs premiers.

Nous allons tenter de démontrer que le système RSA permet de retrouver le message original, après chiffrement et déchiffrement, c'est-à-dire x^ed = x [n] avec x le message sous forme de chiffres et n = pq avec p et q deux nombres premiers distincts.

Méthode utilisant une conjecture personnelle venue en premier :

Soit x un entier naturel quelconque
Soient p et q deux entiers naturels distincts quelconques très grands
On a x < p et x < q car p et q sont très grands

Si p est premier,
d'après le petit théorème de Fermat, x^(p-1) = 1 [p]

Si q est premier,
d'après le petit théorème de Fermat, x^(q-1) = 1 [q]

On a ainsi
(x^(p-1))^(q-1) = 1 [q] et (x^(q-1))^(p-1) = 1 [p] <=> x^(p-1)(q-1) = 1 [q] = 1 [p]

Nous pouvons en déduire que x^(p-1)(q-1) = 1 [pq]
car p et q sont premiers et que l'on utilise (p-1)(q-1) en exposant de x (aucun contre-exemple n'a été trouvé à ce jour)
Soit e et d des entiers naturels tels que ed = 1 [(p-1)(q-1)]
On a x*x^k(p-1)(q-1) = x^ed avec k un entier relatif tel que ed = k (p-1)(q-1) + 1

Par produit, x^ed = x*1 = x [pq]
Méthode utilisant le théorème d'Euler venue par la suite :

Si n est un entier supérieur ou égal à 2, phy(n) représente le nombre d'entiers compris entre 1 et n inclus, et premiers avec n.

Ainsi si n est premier, phy(n) = n-1

Nous admettons que si PGCD(m, n) = 1 avec m et n des entiers naturels, alors phy(m*n) = phy(m)*phy(n)

Si n = pq avec p et q premiers distincts, alors phy(n) = (p-1)(q-1)

Soit x, un entier naturel et n un entier naturel supérieur ou égal à 2, le théorème d'Euler affirme que x^phy(n) = 1 [n]

Dans le cas du RSA, ed = 1 [phy(n)] = k*phy(n) + 1 avec k un entier relatif
Donc x^ed = x*(x^phy(n))^k [n] = x*(1^k) [n] = x*1 [n] = x [n]

Après chiffrement, x devient x^e.
Après déchiffrement, x^e devient x^ed.
On retrouve bien x, puisque x^ed = x [n]

Opérations sur les mots binaires pour le SHA-1 [11]

Elles utilisent les conventions suivantes :

- Opérations binaires bit à bit
- Addition modulo 2^w, soit 2^{32}
    - L'opération *x + y* est définie comme suit. Soient deux mots *x* et *y* représentant les nombres entiers X et Y, tels que $0 \leq X \leq 2^{32}$ et $0 \leq Y \leq 2^{32}$, on a Z le résultat de l'addition modulo 2^{32} de X et de Y. $Z = (X + Y)$ modulo 2^{32}, $0 \leq Z \leq 2^{32}$. On convertit Z en un mot z, et on définit alors z = x + y.
- L'opération de rotation binaire par la gauche $ROTL^n(x)$, où x est un mot de 32 bits et $0 \leq n \leq 32$, est définie par : $ROTL^n(x) = (x << n) \wedge (x >> 32 - n)$.

Bibliographie

[1] : CRYPTOGRAPHIE

http://fr.wikipedia.org/wiki/Cryptographie

[2] : FEDERAL INFORMATION PROCESSING STANDARDS : PUBLICATION 197 (FIPS-197)

Publié le 26 Novembre 2001 par le National Institute of Standards and Technology

http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[3] : PROJET EN LANGAGE C++ IMPLEMENTATION DE L'ALGORITHME A.E.S

Par Sebastien Varrette (Sebastien.Varrette@imag.fr)

« www-id.imag.fr/~svarrett/enseignements/2005-2006/enonce_Projet_AES_ HTML/enonce_Projet_AES.html »

[4] : INTRODUCTION A LA CRYPTOGRAPHIE ET A LA SECURITE INFORMATIQUE (Version PDF).

Par Renaud Dumont

http://www.montefiore.ulg.ac.be/~dumont/pdf/crypto.pdf

[5] : CRYPTOGRAPHIE ET METHODES DE CRYPTAGE.

Par : Faci Salim et Djellid Anissa

Mémoire de fin d'études en Electronique option Communication.

Promotion 2001, Faculté du Génie Electrique et Informatique, UMMTO.

[6] : LA CRYPTOGRAPHIE.

Par : Lifchitz Renaud (MIAS 12) / Vuduy Anh-Dao

http://www.primenumbers.net/Renaud/fr/crypto/index.htm

[7] : SIGNATURE ET CHIFFREMENT.

Mémoire de DEA en Réseaux de télécommunications

Par : Charles Chebli.

Promotion 2003, Faculté de Génie, Université Libanaise.

[8] : CRYPTOGRAPHIE.
http://www.commentcamarche.net/crypto/crypto.php3

[9] : INITIATION A PGP (Version PDF)
Par : Bourgeois Morgan (morganbourgeois@gmail.com)
http://mbourgeois.developpez.com/articles/securite/pgp/

[10] : Http://www.gymnaseyverdon.vd.ch/branches/mathematique/nombres/index.htm

[11] : SHA-1
http://fr.wikipedia.org/wiki/Sha-1

[12] : CORPS (MATHEMATIQUES)
http://fr.wikipedia.org/wiki/Corps_(mathématiques)

[13] : CRYPTOGRAPHIE ASYMETRIQUE
http://fr.wikipedia.org/wiki/Cryptographie_asymétrique

[14] : X.509
http://fr.wikipedia.org/wiki/X.509

[15] : INFRASTRUCTURE A CLES PUBLIQUES
http://fr.wikipedia.org/wiki/Public_Key_Infrastructure

[16] : CRYPTANALYSE
http://fr.wikipedia.org/wiki/Cryptanalyse

[17] : STANDARD DE CHIFFREMENT AVANCE
http://fr.wikipedia.org/wiki/Standard_de_chiffrement_avancé

[18] : MODE D'OPERATION (CRYPTOGRAPHIE)
http://fr.wikipedia.org/wiki/ECB