الجمهورية الجزائرية الديموقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Minstère de l'Enseignement Supérieur et de la Recherche Scientifique

CENTRE UNIVERSITAIRE DE MILA
INSTITUT DES SCIENCES ET DE LA TECHNOLOGIE

Réf. ***/12***

Mémoire de fin d'étude

Présenté pour l'obtention du diplôme de

Licence Académique

Domaine : **Mathématiques et Informatique**

Filière : **Informatique**

Thème

Implémentation d'un algorithme Cryptographique pour la sécurité des données

Présenté par :

1 - Bouraoui Nassim

2 - Ghichi Amine

Dirigé par :

- M^{elle} Deffas Zineb

Année universitaire 2011-2012

Remerciement

C'est avec l'aide de Dieu qu'a vu les jours ce présent travail.

Ensuite, il n'aurait pas pu être achevé sans le soutien, les conseils, les encouragements de certaines personnes auxquelles nous tenons ici à exprimer nos sincères remerciements.

En première lieu nous exprimons toute notre gratitude pour notre encadreur Mme Deffas Zineb pour ses précieux conseils, ses disponibilités, la confiance qu'elle nous a toujours témoigné et la sollicitude dont elle nous a entouré, et ce tout au long de l'élaboration du présent travail.

Nous n'oublions pas non plus nos enseignants qui tout au long du cycle d'étude au centre universitaire de Mila, nous ont transmis leur savoir.

Nous tenons enfin à remercier tous ceux qui ont collaborés de prés ou de loin à l'élaboration de ce travail. Qu'ils acceptent nos humbles remerciements.

Nassim & Amine

Dédicaces

Je dédis ce travail :

- ✓ *A mon père*
- ✓ *A ma mère*
- ✓ *A mes sœurs*
- ✓ *A mon binôme Amine et sa famille.*
- ✓ *A mes collègues d'étude surtout Radouane, Fouad, Zaki, Fayçal, Abderrahmen, Nouar, et mon ami Amir*
- ✓ *A mon cousin Abdessatar*
- ✓ *A toute ma famille*

Nassim

Dédicaces

Je dédie ce modeste travail ,

À Ma chère Mère, pour ses sacrifices depuis qu'elle m'a mis au monde, Mon Père, qui m'a toujours soutenu et aidé à affronter les difficultés,
À Tous mes Frères et sœurs,
À Tous ma famille
A mon binôme Nassim,
À Tous mes amis Radouane, Fouad, Faycal, Zaki, Abderrahmen, Nouar,et collègues de la promotion 2012,
À Tous ceux qui me sont chers,

AminE

Résumé

Le développement rapide des réseaux mondiaux et les immenses possibilités offertes par les transactions électroniques posent aujourd'hui de manière cruciale le problème de protection des informations échangées sur les réseaux informatiques mondiaux, d'où l'utilisation des techniques cryptographiques. Le cryptage et le décryptage sont devenus de plus en plus indispensables comme moyen de confidentialités et de protection de l'information contre tout système d'espionnage. L'objectif de notre travail est l'étude des différentes classes d'algorithmes cryptographiques ainsi que la réalisation d'un système cryptographique à clé secrète AES *(Advanced Encryption Standard)*, qui fait partie des systèmes symétriques (la même clé est utilisé pour crypter et décrypter) et qui exploite des méthodes cryptographiques traditionnelles.

Table des matières

Introduction générale **1**

Chapitre I: Etat de l'art sur la cryptographie

1. Introduction **2**

2. Les fondements de la cryptographie **2**

2.1 Historique 2

2.1.1 L'âge artisanal (part des origines) 2

2.1.2 L'âge technique 3

2.1.3 L'âge paradoxal (utilise les algorithmes de cryptage) 3

2.2 Définition de la cryptographie 4

2.3 Les fonctions de la cryptographie 4

2.4 A quoi sert la cryptographie ? 5

2.5 Termes de la cryptographie 6

3. Classification des algorithmes cryptographique **7**

3.1 La cryptographie classique 7

3.1.1 Chiffrement par substitution 8

3.1.2 Chiffrement par transposition 10

3.2 La cryptographie moderne 11

3.2.1 Chiffrement symétrique 11

3.2.2 Chiffrement asymétrique 19

3.2.3 Comparaison entre la cryptographie symétrique et asymétrique 21

3.3 Les algorithmes associés 21

3.3.1 Cryptographie hybride 21

3.3.2 Les fonctions de hachage 23

3.3.3 Signature numérique 24

3.3.4 Certificat numérique 25

4. La cryptanalyse **27**

4 .1 Définition 27

4 .2 Classification des attaques en fonction des données disponible 27

4.3 Les différentes attaques 28

a) L'analyse des fréquences 28

b) L’indice de coïncidence 28
c) L'attaque par mot probable 28
d) L’attaque par force brute 28
e) Attaque par paradoxe des anniversaires 28
g) Cryptanalyse linéaire 29
h) Autres attaques 29
4 .4 Réussite de l'attaque 29
5. Conclusion 30

Chapitre II: L'algorithme AES

1. Introduction 31
2. Historique de l’algorithme AES 31
3. Présentation de l’algorithme AES 32
3.1 Chiffrement 32
3.1.1 Mise en forme des entrées 32
3.1.2 Première étape – Round 0 33
3.1.3 Deuxième étape – Rounds 1-9 33
3.2 Troisième étape – Round 10 38
3.3 Déchiffrement 40
4. Conclusion 43

Chapitre III: Implémentation

1. Introduction 45
2. Pour quoi java ? 45
2.1 Présentation générale 45
2.2 Caractéristiques du langage java 46
3. Pour quoi NetBeans ? 48
4. Présentation général de NetBeans 48
4.1. Définition 48
4.2. Caractéristiques 49
4.3. Avantages de NetBeans 50
5. Description de l’application 51
5.1 Structure de l’application 51
5.1.1 Les classes 51
5.1.2 Les interfaces graphiques 51
5.2 L’interface graphique de l’application 52

5.2.1 Barre de Menu 52
5.2.2 Buttons raccourcis 53
6. Conclusion..... 59
Conclusion générale 60
Bibliographie..... 61

Liste des figures

Figure I.1: Schéma générale de cryptographie …… 4
Figure I.2: Classification les algorithmes de la cryptographie …….. 7
Figure I.3: La cryptographie symétrique ……..…….……..……. 11
Figure I.4: Chiffrement en mode ECB ……..……..……. 12
Figure I.5: Chiffrement en mode CBC ……..……..……. 13
Figure I.6: Chiffrement en mode CFB ……..……..……. 13
Figure I.7: Chiffrement en mode OFB ……..……..……. 14
Figure I.8: chiffrement par DES ……..…..…… 16
Figure I.9: L'algorithme TDES (192 bits) ……..…..……. 17
Figure I.10: chiffrement par IDEA ……...…..…..…… 18
Figure I.11: La cryptographie asymétrique ……..…...……. 19
Figure I.12 : Le chiffrement à clé publique RSA ……..……..……. 20
Figure I.13: shéma de hachage et signature numérique ……....…..……. 25
Figure I.14 : certificat numérique ……...…… 25
Figure II.1 Texte claire (State) ……...……. 33
Figure II.2 Clé(Key) ……...……. 33
Figure II.3 le tableau de texte après Add Round Key ……. 33
Figure II.4 Bytes substitution box ……... 34
Figure II.5 le texte après l'opération Sub Bytes …….. 34
Figure II.6 le texte après l'opération Shift Rows …… 35
Figure II.7 le texte après décalage ……...…….... 35
Figure II.8 la matrice de Mix Column …… 36
Figure II.9 le texte après Mix Column …… 36
Figure II.10 le texte après les champs de Galois ……. 37
Figure II.11 la clé après l'opération XOR ……... 37
Figure II.12 l'opération de Add Round Key …….. 38
Figure II.13 le résultat de Add Round Key ……. 38
Figure II.14 le texte après Round 10 …….…….. 38
Figure II.15 resultat finale de key schedule …… 40
Figure II.16 Inverse S-Box ……...…..…….. 40
Figure II.17 Inverse Shift Rows ……....……. 41

Figure II.18 La matrice constante inverse ……..……. 41
Figure II.19 Schéma général Chiffrage/Déchiffrage ……..……. 42
Figure III.1 Interface graphique principale ……. 51
Figure III.2 Barre de Menu ……. 52
Figure III.3 Buttons raccourcis ……..……. 53
Figure III.4 Texte Clair ……..……. 53
Figure III.5 Entrer la clé de cryptage……..……..……..……. 54
Figure III.6 Texte crypté ……. 54
Figure III.7 Entrer la clé de décryptage..……..…… 55
Figure III.8 Texte clair ……..……...…… 55
Figure III.9 Fenêtre ouvrir pour choisir l'image clair..…… 57
Figure III.10 Fenêtre après le choix de l'image……..……...…… 57
Figure III.11 Message d'entrer la clé de cryptage……...……..… 57
Figure III.12 Icône de fichier crypté..……...…… 57
Figure III.13 Fenêtre ouvrir pour choisir l'image crypté ……..……...… 58
Figure III.14 choisir l'image crypter et entrer la clé……..…….. 59
Figure III.15 L'image clair ……..……..… 59

Liste des tableaux

Tableau I.1: table de substitution …….. 8
Tableau I.2: table de transposition ……..…… 10
Tableau I.3 : Comparaison entre la cryptographie symétrique et asymétrique ..……. 21

Introduction générale

Depuis les temps historiques les plus reculés, les hommes ont utilisé diverses méthodes pour coder des messages et rendre ceux-ci inintelligibles à des lecteurs indésirables. C'est essentiellement dans les domaines militaires et diplomatiques que le cryptage de messages était utilisé. Aujourd'hui, le champ d'application de la cryptologie s'est élargi et a trouvé un regain d'actualité avec les problèmes posés par la sécurité des transactions bancaires, des transmissions de fichiers et de bases de données sous forme électronique.

La cryptographie (sécurité des données) est l'ensemble des processus de verrouillage visant à protéger l'accès à certaines données afin de les rendre incompréhensible aux personnes non autorisées, autrement dit garantir la confidentialité, l'intégrité de ces informations, ainsi que leur imputabilité. L'émetteur d'une information doit être certain de l'identité du destinataire et inversement.

La cryptographie est une science très ancienne, elle est moyenne de sauvegarde le caractère confidentiel des informations. Elle ne protège pas les communications en tant que telles mais plutôt leur contenu. Plusieurs algorithmes ont été proposé pour le cryptage, ces algorithmes sont classifié selon différents critères, par exemple en peut les classifier selon l'apparition (classiques et modernes), ou selon la nature de la clé (publique ou secrète).

Puisque cette science est très large, nous essayons d'implémenté un algorithme moderne à clé secrète ***AES (Advanced Encryption Standard)***.

Pour aboutir à ce but nous divisons notre travail en 3 chapitres:

- Le premier chapitre présente un état de l'art sur la cryptographie, et un panorama sur les algorithmes cryptographique existent avec des exemples, et termine par quelques concepts sur la cryptanalyse et les différentes techniques utilisent dans cette opération.
- Le deuxième chapitre explique en détaille l'algorithme que nous avons choisi (l'algorithme AES).
- Le dernier chapitre commence par la description du langage et l'environnement de développement utiliser pour l'implémentation. Puis présente l'interface graphique de notre application suivie par des exemples.

Chapitre I

Etat de l'art sur la cryptographie

1. Introduction

Dans ce chapitre, nous présentons un aperçu global sur la notion de la cryptographie. Nous commençons par l'historique, puis, nous donnons quelques définitions de la cryptographie et leurs fonctions. Par la suite, nous présentons une classification des algorithmes cryptographique, et enfin nous terminons ce chapitre par la cryptanalyse et les différentes attaques utiliser.

2. Les fondements de la cryptographie

2.1 Historique

Avant de décrire la cryptographie en soi, passons en revue son histoire, son apparition et son évolution au fil du temps ; les aspects techniques brièvement présentés dans cet élément.
La science de chiffrement n'est pas une science moderne mais l'histoire de cryptage est plus long est classé sur trois grand étapes :

2.1.1 L'âge artisanal (part des origines)

La cryptographie serait apparue pour la première fois il y a 4000 ans, au bord du nil : un scribe aurait tracé d'une façon particulière des hiéroglyphes sur la tombe de son maître. Le but n'était néanmoins pas de rendre le texte illisible, mais de le rendre plus solennel.
mais le cryptage développé avec Jules César utilisait, semble-t-il un mécanisme de confidentialité rudimentaire, où chaque Lettre d'un message était remplacée par celle située trois positions plus loin dans l'alphabet.
La méthode se généralise en opérant une permutation quelconque de l'alphabet, et prend le nom de substitution. Une autre méthode, dite de transposition, change l'ordre des lettres; elle a été mise en œuvre au Moyen Age notamment par un dispositif appelé "grille de Cardan". De façon générale, jusqu'au début du vingtième siècle, la cryptographie était affaire de substitutions et de transpositions.
On opérait d'ailleurs fréquemment des substitutions non seulement sur des lettres mais sur des mots, en s'aidant d'une sorte de dictionnaire à double entrée, nommé code ou répertoire [1].

2.1.2 L'âge technique

De nombreux dispositifs mécaniques furent mis en place afin de faciliter le chiffrement. La plupart de ces dispositifs se basaient sur des rotors (des disques où sont imprimées les lettres de l'alphabet).

C'est le cas de la machine *Enigma*, destinée initialement aux civils (inventée en 1919 par Arthur Scherbius et commercialisée en 1923), mais très vite utilisée par l'armée allemande à partir des années 30. La plupart des communications allemandes seront chiffrées via la machine Enigma, d'où l'intérêt pour les alliés de pouvoir déchiffrer leurs messages.

La machine Enigma effectue une substitution qui change à chaque lettre, son fonctionnement est basé sur un assemblage d'un tableau de connexion (qui ne fait que permuter quelques lettres, afin de rendre la cryptanalyse plus difficile), de plusieurs rotors qui effectuent les permutations des lettres, et d'un réflecteur qui renvoie le courant sur le panneau lumineux où la lettre chiffrée s'allume. À chaque lettre tapée, le premier rotor avance d'une position ; lorsque le premier rotor a fait un tour complet, c'est le second qui tourne, et ainsi de suite. Grâce à la combinaison de ces dispositifs, on peut obtenir plus de 10^{17} clés possibles pour une machine à 5 rotors.

2.1.3 L'âge paradoxal (utilise les algorithmes de cryptage)

Après la guerre 40-45, il faudra attendre une trentaine d'années avant de nouvelles avancées dans le domaine de la cryptologie.

En 1971, un cryptographe d'IBM, Horst Feistel met au point un algorithme de chiffrement par bloc, nommé *Lucifer* qui possède de nombreuses variantes. Deux ans plus tard, la NSA modifie *Lucifer* pour sortir le *Data Encryption Standard*, *DES*, en 1977 ; il fut encore amélioré par la suite et longuement utilisé et reste utilisé encore aujourd'hui (bien qu'il soit moins répandu), notamment avec le *Triple DES*, qui consiste simplement à opérer trois DES consécutifs avec deux ou trois clés.

La même année, le chiffrement à clé publique (ou asymétrique) est présenté pour la première fois dans une publication de W. Diffie et M. Hellman . Ce concept est mis en pratique avec le chiffrement RSA, inventé par Ron Rivest, Adi Shamir et Leonard Adleman et présenté en 1978 dans une publication . Grâce aux algorithmes à clé publique, le problème de distribution des clés est résolu via l'utilisation de deux clés : une pour chiffrer, rendue publique, et une pour déchiffrer, gardée secrète par la personne censée déchiffrer le message [2].

2.2 Définition de la cryptographie

En informatique, la littérature fournit un tas de définitions du mot cryptographie. Ces définitions, dans leur diversité, offrent des points de vues à la fois différents et complémentaires. Dans ce qui suite, nous décrivons quelques unes.

- la cryptographie est un moyen de sauvegarder le caractère confidentiel des informations. Elle ne protège pas les communications en tant que telles mais plutôt leur contenu pour protéger l'accès à certaines données afin de les rendre incompréhensible aux personnes non autorisées [3].
- La cryptographie est l'ensemble des techniques qui permet de rendre un message inintelligible. L'action de coder le message initial (plaintext) en un message inintelligible, appelé « cryptogramme » (cipher text), se nomme « chiffrement » (ou cryptage). L'opération inverse est le « déchiffrement » (ou décryptage) [4].
- La *cryptographie* est la science qui utilise les mathématiques pour le cryptage et le décryptage de données. Elle vous permet ainsi de stocker des informations confidentielles ou de les transmettre sur des réseaux non sécurisés (tels que l'Internet), afin qu'aucune personne autre que le destinataire ne puisse les lire [5].

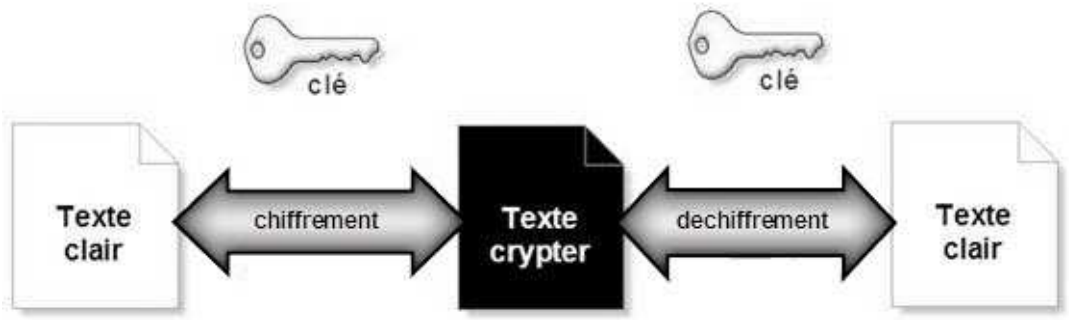


Figure I.1: schéma générale de cryptographie

2.3 Les fonctions de la cryptographie

La cryptographie est traditionnellement utilisée pour dissimuler des messages aux yeux de certains utilisateurs. Cette utilisation a aujourd'hui un intérêt d'autant plus grand que les communications via internet circulent dans des infrastructures dont on ne peut garantir la confidentialité. Désormais, la cryptographie sert non seulement à préserver la confidentialité des données mais aussi à garantir leur intégrité et leur authenticité .

2.3.1 l'authenticité:

Garantit l'identité d'une entité donnée ou l'origine d'une communication ou d'un fichier. Lorsqu'il s'agit d'un fichier et que l'entité qui l'a créé est la seule à avoir pu apporter la garantie d'authenticité, on parle de non-répudiation.

2.3.2 non-répudiation:

Mécanisme pour enregistrer un acte ou un engagement d'une personne ou d'une entité de telle sorte que celle-ci ne puisse pas nier avoir accompli cet acte ou pris cet engagement.

2.3.3 l'intégrité:

Garantit que le contenu d'une communication ou d'un fichier n'a pas été modifié. Par exemple, on peut souhaiter vérifier qu'aucun Changement du contenu d'un disque dur n'a eu lieu: des produits commerciaux, mettant en jeu des méthodes cryptologiques, sont disponibles à cet effet.

2.3.4 La confidentialité:

Garantit que le contenu d'une communication ou d'un fichier n'est pas accessible auxtiers.

Des services de confidentialité sont offerts dans de nombreux contextes

- en téléphonie mobile, pour protéger les communications dans la partie "aérienne";
- en télévision à péage pour réserver la réception des données aux abonnés;
- dans les navigateurs, par l'intermédiaire du protocole SSL (Secure Socket Layer), dont l'activation est souvent indiquée par un cadenas fermé représenté En bas de la fenêtre [1].

2.4 A quoi sert la cryptographie ?

L'application la plus évidente de la cryptographie est la protection de la confidentialité d'une information, qu'elle soit stockée localement sur une machine, ou transmise sur un réseau.

Le besoin de confidentialité n'est pas l'apanage des militaires ou de certain gros industriels. Tout les individus, toutes les organisations ont, à des degrés divers, un tel besoin :

- Confidentialité des transactions bancaires.
- Protection de secrets industriels ou commerciaux.

- Protection des sessions de télé-travail.
- Protection du secret médical.
- Protection des systèmes informatique contre les intrusions.
- Protection de la confidentialité de communications dans le cadre d'une association, d'un Parti politique, d'un syndicat…
- Protection de la vie privée.
- jeux

2.5 Termes de la cryptographie

- **Le chiffrement :** est la démarche effectuée afin de rendre le message clair illisible, chiffré. On utilise parfois le terme « cryptage », qui est un anglicisme ; nous éviterons donc de l'utiliser.
- **Le déchiffrement :** est la démarche inverse du chiffrement, qui retrouve le message clair à partir du message chiffré en ayant connaissance de la clé, du secret ou de l'algorithme utilisé (contrairement à la cryptanalyse). Le mot décryptage peut aussi être utilisé.
- **Cryptologie :** Regroupe cryptographie et cryptanalyse.
- **La cryptanalyse :** vise à retrouver le message clair à partir du message chiffré sans avoir connaissance de la clé ou du secret.
- **La stéganographie:** est une discipline semblable à la cryptographie mais qui consiste à cacher un message (dans une image. . .) et non pas à la rendre inintelligible.
- **La clé de chiffrement:** est une donnée (mot, suite d'opération, nombre. . .) utilisée pendant le chiffrement afin de rendre le déchiffrement plus difficile sans la connaissance de celle-ci. Il existe plusieurs types de clés : certaines qui doivent être gardées secrètes (clés privées) et d'autres qui peuvent être diffusées (clés publiques).
- **Les données claires:** sont les données dans leur forme initiale, non chiffrées. Ces données peuvent être du texte (un message) ou d'autres données informatiques (un fichier,. . .).
- **Les données chiffrées :** sont les données qui sont chiffrées via un certain algorithme de chiffrement.
- **Un cryptogramme :** est un message chiffré dans le but d'être déchiffré par des amateurs de cryptographie (on en retrouve par exemple dans les journaux). Ils sont la plupart du temps assez simples.

- **La clé :** La clé c'est une valeur utilisée dans un algorithme de cryptographie, afin de générer un texte chiffré.
- **L'algorithme de cryptage :** Un algorithme de cryptographie ou un chiffrement est une fonction mathématique utilisée lors du processus de cryptage et de décryptage.

3. Classification des algorithmes cryptographique

On peut effectuer une classification des algorithmes cryptographiques selon déférents critères, par exemple en peut les classifier selon l'apparition (algorithmes cryptographiques classiques et modernes), ou selon la nature de la clé (publique ou secrète); dans ce chapitre nous essayons de les classifier en prenant en compte tout ces critères.

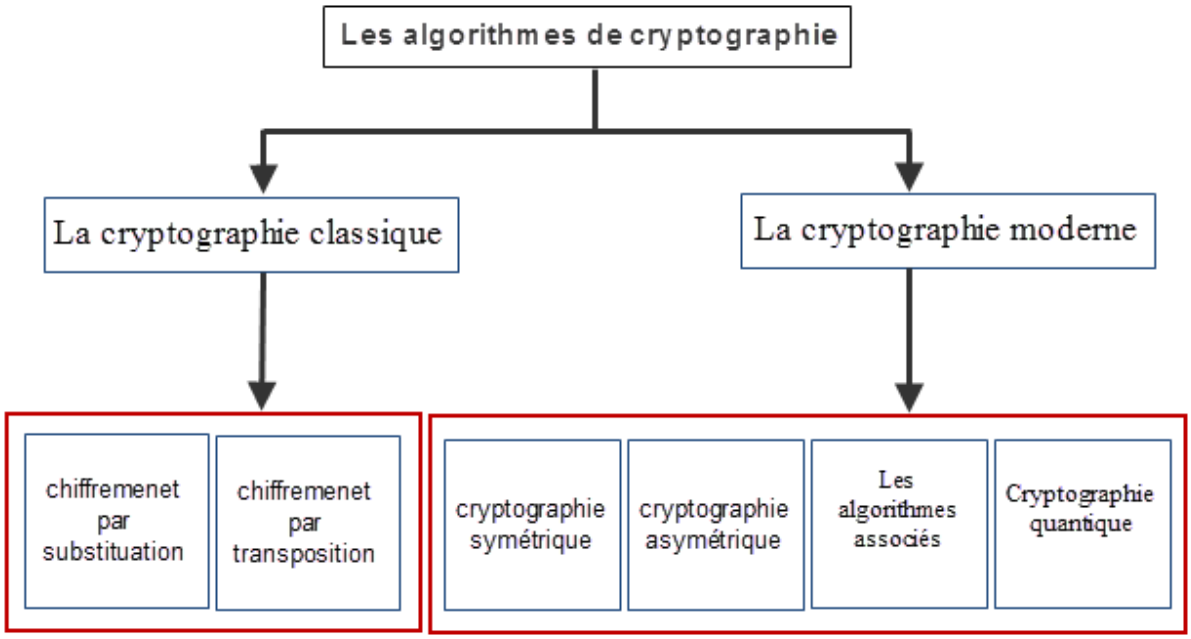


Figure I.2: Classification les algorithmes de la cryptographie

3.1 La cryptographie classique

La cryptographie classique décrit la période avant les ordinateurs. Elle traite des systèmes reposant sur les lettres et les caractères d'une langue naturelle. Les principaux outils utilisés remplacent des caractères par des autres et les transposent dans des ordres différents. Les meilleurs systèmes (de cette classe d'algorithmes) répètent ces deux opérations de base plusieurs fois. Cela suppose que les procédures (de chiffrement ou déchiffrement) soient gardées secrètes ; et sans cela le système est complètement inefficace (n'importe qui peut déchiffrer le message codé).

3.1.1 Chiffrement par substitution

3.1.1.1 Substitution mono alphabétique : « cryptage de césar »

Chaque caractère du texte en clair est remplacé par un caractère correspondant dans le texte chiffré. Les exemples les plus célèbres sont les algorithmes de César et Rot13. Ils sont encore utilisés aujourd'hui pour cacher le sens de certains messages (par exemple la solution de certains jeux dans des journaux), mais bien sûr elles sont très peu sûrs. En effet avec ce principe, les lettres les plus fréquentes dans le texte en clair restent les plus fréquentes dans le texte chiffré, il ne cache donc pas les fréquences d'apparition des caractères. C'est une faiblesse importante puisque des techniques statistiques peuvent être utilisés pour associer aux lettres les plus fréquentes, les algorithmes à base de substitutions mono-alphabétique sont facilement cassés par les spécialistes.

Pour l'instant, nous avons vu les chiffrements à un alphabet, ou mono alphabétiques, et constaté comme l'utilisation d'un ordre alphabétique et non alphabétique a un effet sur la complexité.

Exemple :

chaque texte clair A est remplacé par le texte chiffré B, chaque texte claire B est remplacé par le texte chiffré C, etc... Utiliser un seul alphabet pour substituer des lettres dans un message ne suffit pas à stopper un cryptanalyste.

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

On a le mot "CRYPTAGE" qui sera "DSZQUBHF" aprés la substitution .

Text clair	C	R	Y	P	T	A	G	E
Text crypté	D	S	Z	Q	U	B	H	F

Tableau I.1: table de substitution

3.1.1.2 Substitution poly alphabétique : « cryptage de vigenère »

Le chiffrement de vigenère ressemble beaucoup au chiffrement de césar, à la différece près qu'il utilise une clef plus longue afin de pallier le principal problème du chiffrement de césar. Le fait qu'une lettre puisse être codée d'une seul façon. Pour cela on utilise un mot clef au lieu d'un simple caractère.

Pour crypter on choisit une clef (mot ou phrase). chaque lettre du texte clair on fait correspondre une lettre de clef (la clef etant répétée autant de fois que nécessaire).

La lettre de chiffré cela prise dans la colonne corespondante à la lettre du texte clair , et dans la ligne corespondante à la lettre de la clef . en posant C le texte codé , T le texte et K la clé, on peut traduire ceci par la formule : C=T+K[mod26]

Il consiste à coder un texte avec un mot en ajoutant à chacune de ses lettres la lettre d'un autre mot appelé clé. La clé est ajoutée indéfiniment en vis-à vis avec le texte à chiffrer.

Pour déchiffrer le message, il suffit de faire l'opération inverse, on prend la ligne correspondante à la lettre de clé, et on la suit jusqu'à rencontrer le caractère codée, la lettre décodée est alors la première cette colonne, ce qui se traduit par la formule : T=C-K[mod26]

Exemple: le texte " texte chiffré " avec la clé « BIEN » sera codé de la manière suivante

texte Clair:

T	E	x	t	e	c	h	i	f	f	r	e
20	05	24	20	05	03	08	09	06	06	18	05

Clé:

B	I	E	N
02	09	05	14

Texte crypté :

t+B	e+I	x+E	t+N	e+B	c+I	h+E	i+N	f+B	f+I	r+E	e+N
20+02	05+09	24+05	20+14	05+02	03+09	08+05	09+14	06+02	06+09	18+05	05+14

V	N	C	X	G	L	M	W	H	O	W	S
22	14	29=3	24	7	12	13	23	8	15	23	19
		[26]									

''texte chiffré'' en écrit donc: **VNCXGLMWHOWS**

3.1.2 Chiffrement par transposition

Jusqu'ici, tous les chiffrements que nous avons abordés sont des systèmes substitution dans les quels les lettres du texte clair sont remplacées par celles du texte chiffré.

Changer les positions des lettres du texte clair est une autre technique de chiffrement On l'appelle transposition, beaucoup de journaux présentent des puzzles par transposition.

Un simple chiffrement par transposition de FIVE AM déplace chaque lettre d'une position vers la gauche. FIVE AM devient ainsi IVEA MF.

Bien que les lettres ayant été déplacées, les lettres du texte chiffré soient toutes identiques à celles du texte clair, il n'y a ni remplacement, ni substitution de lettre.

Pour illustrer cette technique de chiffrent, voyons une transposition plus complexe :

LA CRYPTOGRAPHIE CLASSIQUE. (Voir la table I.2 ci-dessous)

Texte clair

LA CRYPTOGRAPHIE CLASSIQUE

Texte chiffré

LT HSAOISCGEIRRC QYALUPPAE

L	A	C	R	Y	P
T	O	G	R	A	P
H	I	E	C	L	A
S	S	I	Q	U	E

Tableau I.2: table de transposition.

Chaque lettre en texte clair est transférée vers une nouvelle position. La table I.2 ressemble à un traitement de texte à six colonnes, muni de retour à la ligne la méthode employée pour lire les lettres de la grille est le chiffrement.

Ce système ci lit les lettres de première colonne vers la bas, puis les lettres de la deuxième colonne vers la bas, etc. les lettres cryptées sont les mêmes que les lettres en texte clair, mais elles sont positionnées pour former un nouveau schéma.

Le destinataire prévu doit connaître deux choses : la longueur et la largeur de la grille et la manière dont les lettres sont à partir de la grille.

3.2 La cryptographie moderne

Les méthodes utilisées de nos jours sont plus complexes, cependant la philosophie reste la même. La différence fondamentale est que les méthodes modernes manipulent directement des bits contrairement aux anciennes méthodes qui opéraient sur des caractères alphabétiques. Ce n'est donc qu'un changement de taille (ou de représentation), puisque l'on utilise plus que deux éléments au lieu des 26 lettres de l'alphabet. La plupart des bons systèmes de cette catégorie combinent toujours des substitutions et des transpositions, et les règles sont connues de tous, c'est pourquoi on appelle cette classe : le chiffrement à usage général. La sécurité de ces méthodes reposent maintenant sur un nouveau concept clé.

3.2.1 Chiffrement symétrique

3.2.1.1 Principe

Ce type de chiffrement est aussi appelé chiffrement à clé privée, du fait que le chiffrement se fait via une clé qui doit rester secrète et qui ne doit être connue que par les personnes devant chiffrer et déchiffrer les messages.

La sécurité de la méthode de chiffrement réside donc dans la difficulté de trouver cette clé.

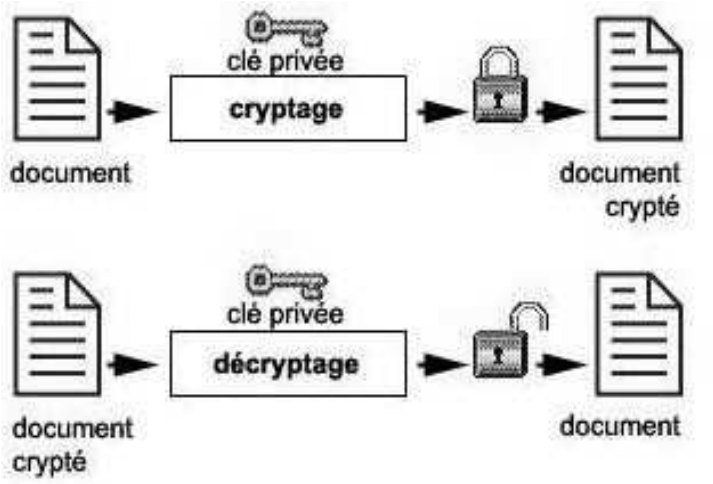


Figure I.3: La cryptographie symétrique

3.2.1.2 Modes de chiffrement

Il y a deux moyens de chiffrer les données avec les algorithmes à clé privée : le *chiffrement par bloc* et le *chiffrement par flot* aussi appelé *chiffrement par flux* :

a) Chiffrement par bloc

Dans ce monde de cryptage, le texte clair est fractionné en blocs de même langueur à l'aide d'une clef unique. Les algorithmes de chiffrement par blocs sont en général construit sur un modèle emploie une fonction F qui prend en paramètre une clef K et un message de n bits. F est répétée un certain nombre de fois, en parle de ronde. A chaque ronde, la clef K utilisée est changée et le message que l'on chiffre est le résultat de l'itération précédente.

1- Electronic codebook (ECB)

Il s'agit du mode le plus simple. Le message à chiffrer est subdivisé en plusieurs blocs qui sont chiffrés séparément les uns après les autres. Le gros défaut de cette méthode est que deux blocs avec le même contenu seront chiffrés de la même manière, on peut donc tirer des informations à partir du texte chiffré en cherchant les séquences identiques. On obtient dès lors un « dictionnaire de codes » avec les correspondances entre le clair et le chiffré d'où le terme *codebook*.

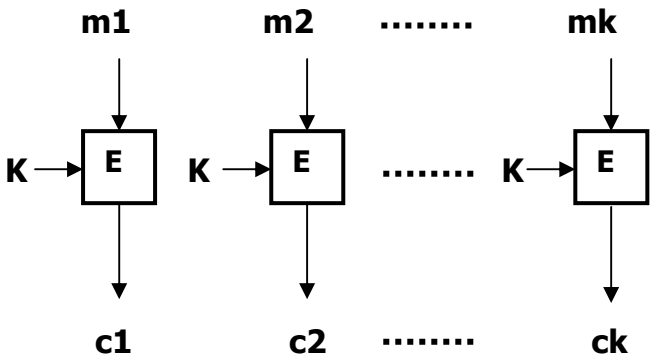


Figure I.4: Chiffrement en mode ECB.

2- Cipher Block Chaining (CBC)

Dans ce mode, on applique sur chaque block un 'OU exclusif' avec le chiffrement du bloc précédent avant qu'il soit lui-même chiffré. De plus, afin de rendre chaque message unique, un vecteur d'initialisation (IV) est utilisé.

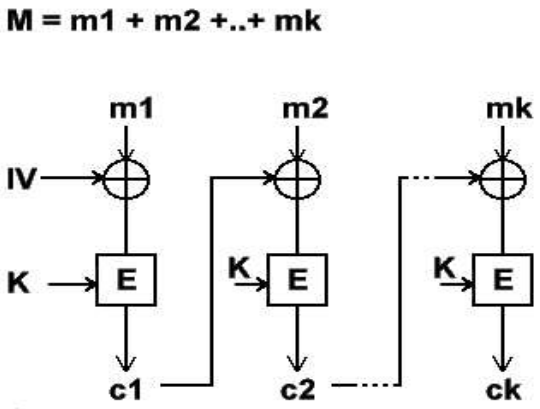


Figure I.5: Chiffrement en mode CBC.

3- Cipher Feedback (CFB)

Ce mode et les suivants agissent comme un chiffrement par flux. Ils génèrent un flux de clés qui est ensuite appliqué au document original.

Dans ce mode, le flux de clé est obtenu en chiffrant le précédent bloc chiffré. CFB est un chiffrement par flot. Son grand intérêt est qu'il ne nécessite que la fonction de chiffrement, ce qui le rend moins cher à câbler ou programmer pour les algorithmes ayant une fonction de chiffrement différente de la fonction de déchiffrement .

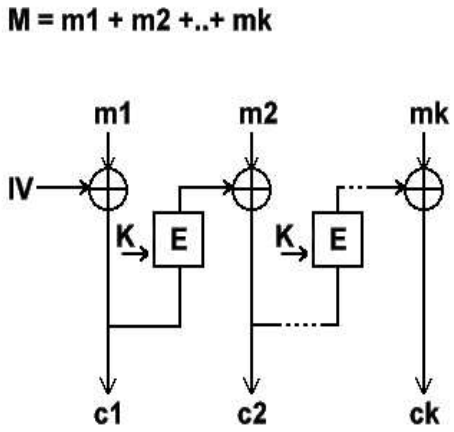


Figure I.6 : Chiffrement en mode CFB.

4- Output Feedback (OFB)

C'est un mode de chiffrement de flot qui possède les mêmes avantages que CFB. De plus, il est possible de le précalculer en chiffrant successivement le vecteur d'initialisation. Il n'est donc sûr que si la fonction de chiffrement alliée à la clé forment une bonne suite pseudo-

aléatoire. Il présente beaucoup de problèmes de sécurité et il est peu conseillé sauf dans le cas où sa longueur est égale à celle de l'algorithme utilisé.

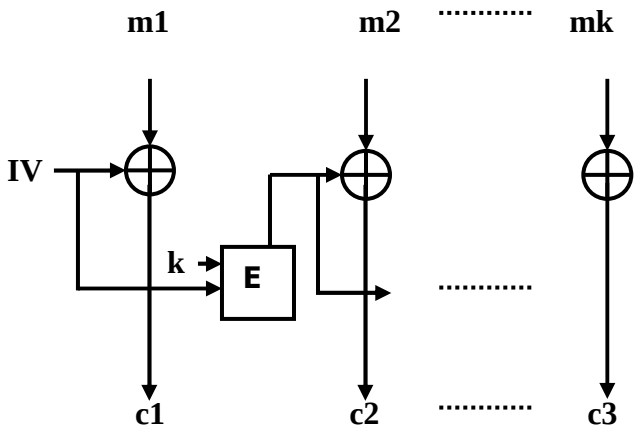


Figure I.7 : Chiffrement en mode OFB.

b) Chiffrement par flux

Le chiffrement par flux traite les données comme elles arrivent, élément par élément. Sur les systèmes informatiques et électroniques, un élément est composé d'un bit, chaque bit est donc traité séparément. Le chiffrement par flux le plus simple à comprendre est sûrement le chiffrement via le *ou exclusif*. Ce *ou exclusif*, ou *XOR*, est un opérateur logique défini dans l'algèbre de Boole, qui, pour deux valeurs données pouvant être soit vraies (1 en binaire) soit fausses (0 en binaire), renvoie *vrai* dans le cas ou une et une seule des valeurs données est vraie ; dans les autres cas, le *ou exclusif* nous renverra *faux*.

Cet opérateur est représenté par le symbole (+) et on peut le définir simplement comme suit, en considérant *A* et *B* comme des bits différents (valant 0 ou 1).

$$A (+) B = 1 \quad B (+) A = 1 \quad A (+) A = 0$$

Leurs avantages principaux sont :

- Du fait que la transformation (méthode de chiffrement) peut être changée à chaque symbole du texte clair et du fait qu'ils soient extrêmement rapides.
- De plus, ils sont utiles dans un environnement où les erreurs sont fréquentes car ils ont l'avantage de ne pas propager les erreurs (diffusion).
- Ils sont aussi utilisés lorsque l'information ne peut être traitée qu'avec de petites quantités de symboles à la fois, comme par exemple si l'équipement n'a pas de mémoire physique ou une mémoire tampon très limitée.

3.2.1.3 DES (Data Encryption Standard)

Au début des années 70, la NBS (National Bureau of Standards) a lancé un appel dans le Federal Register pour la création d'un algorithme de cryptage répondant aux critères suivants :

- ayant un haut niveau de sécurité lié à une clé
- compréhensible
- ne devant pas dépendre de la confidentialité de l'algorithme
- adaptable et économique
- efficace et exportable

Fin 1974, IBM propose Lucifer, qui grâce à la NSA (National Security Agency) est modifié pour donner le DES (Data Encryption Standard) en 1976. Le DES a finalement été approuvéen 1978 parle NBS DES, est le système de cryptage utilisé aujourd'hui en Etats-Unis pour certaines banques. Il est réctualisé tous les 5 ans.

C'est un système de chiffrement par blocs de 64 bits, dont le dernier octet (8 bits) sert de test de parité (pour vérifier l'intègrité des données). Il consiste à faire des combinaisons, des substitutions et des permutations entre le texte à chiffrer et la clé, en faisant en sorte que les opérations puissent se faire dans les deux sens (pour le décryptage). La clé est codée sur 16 bits et formée de 16 blocs de 4 bits, généralement notés k0 à k15. Etant donné que 8 bits de la clé sont réservés pour le test de la parité, « seulement » 56 bits servent réellement à chiffrer, ce qui représente tout de même 256 possibilités, c'est-à-dire 72*1015 clés possibles...

Les grandes lignes de l'algorithme sont les suivantes :

- fractionnement du texte en blocs de 64 bits
- permutation des blocs
- découpage des blocs en deux parties : gauche et droite
- Etapes de permutation et de substitution répétées 16 fois (appelées rondes)
- recollement des parties gauche et droites puis permutation initiale inverse.

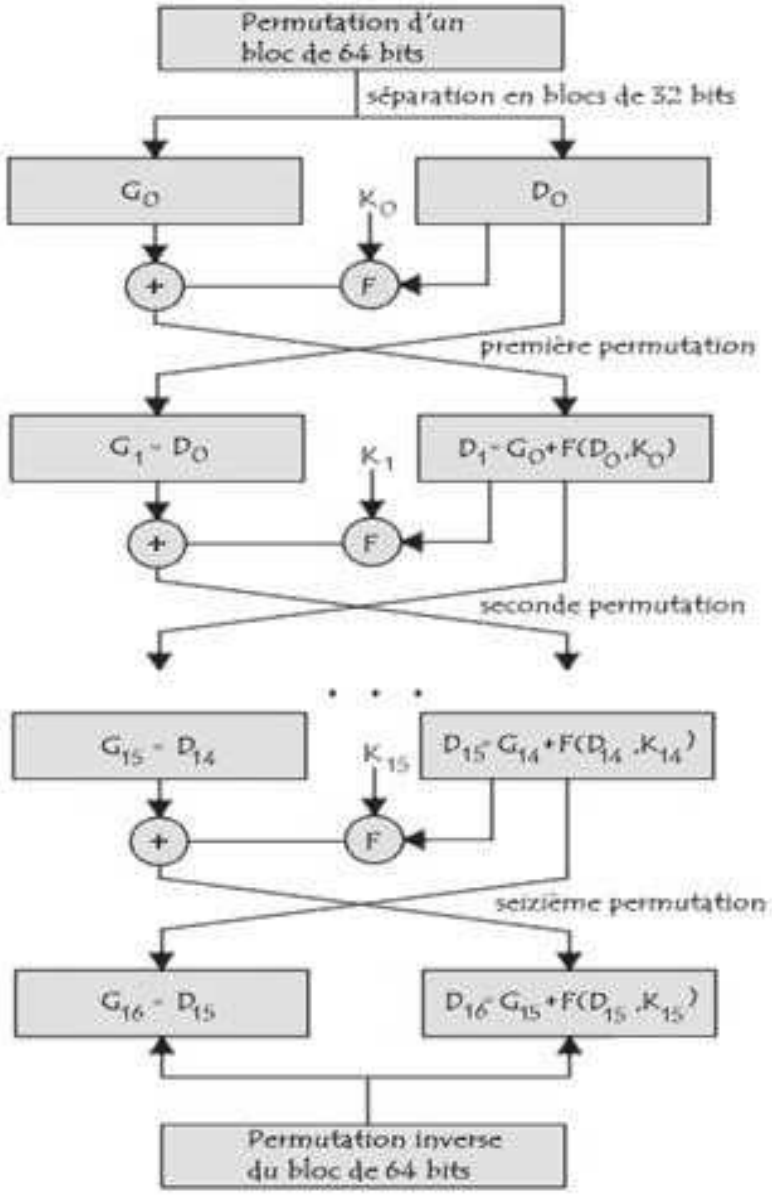


Figure I.8 : chiffrement par DES

3.2.1.4 Le Triple DES

Le Triple DES (aussi appelé 3DES) est un algorithme de chiffrement symétrique par bloc, enchaînant 3 applications successives de l'algorithme DES sur le même bloc de données de 64 bits, avec 2 ou 3 clés DES différentes.

Cette utilisation de trois chiffrements DES a été développée par Walter Tuchman (chef du projet DES chez IBM), il existe en effet d'autres manières d'employer trois fois DES mais elles ne sont pas forcément sûres. La version de Tuchman utilise un chiffrement, suivi d'un déchiffrement pour se conclure à nouveau par un chiffrement.

Le *Triple DES* est généralement utilisé avec seulement deux clés différentes. Le mode d'usage standard est de l'utiliser en mode EDE (*Encryption, Decryption, Encryption*, c'est-à-dire *Chiffrement, Déchiffrement, Chiffrement*) ce qui le rend compatible avec DES quand on utilise trois fois la même clé. Dans le cas d'une implémentation matérielle cela permet d'utiliser le même composant pour respecter le standard DES et le standard Triple DES.

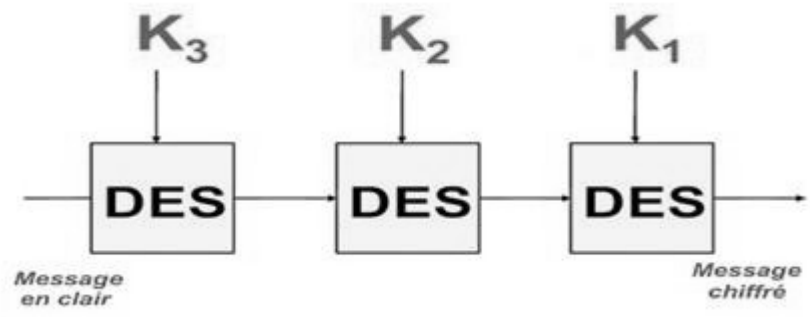


Figure I.9 : L'algorithme TDES (192 bits).

3.2.1.5 IDEA (International Data Encryption Algorithm)

Développé à Zurich en Suisse par Xuejia Lai et James Massey en 1992, le chiffrement par blocs IDEA (International Data Encryption Algorithm) utilise des blocs de 64 bits et est contrôlé par une clé de 128 bits. Malgré le fait que IDEA n'est pas un chiffrement Feistel, le déchiffrement est effectué avec la même fonction que celle utilisée dans le chiffrement. L'algorithme a innové dans son domaine en utilisant des opérations de trois groupes algébriques différents.

Le processus de chiffrement est le même que pour le déchiffrement, à moins d'une utilisation de différentes sous-clés, ce qui est rare. En gros, le processus se résume à huit étapes de chiffrement identiques, les rounds, suivies d'une transformation au bloc de sortie. Contrairement au DES et à Blowfish, IDEA n'utilise aucune S-Box.L'algorithme peut être utilisé avec les modes d'opération ECB, CBC, CFB et OFB. Sa vitesse est environ la même que le DES.

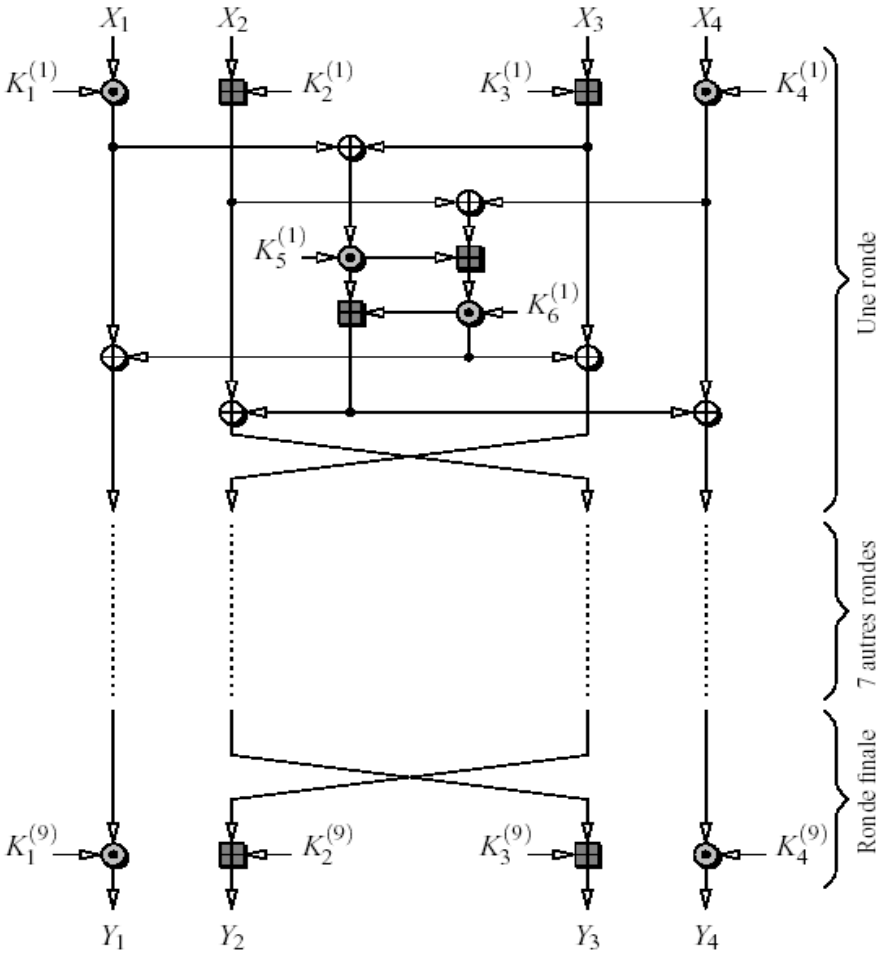


Figure I.10 : chiffrement par IDEA

3.2.1.6 AES (Advanced Encryption Standard)

AES est le sigle d'Advanced Encryption Standard. Il s'agit d'un algorithme de chiffrement symétrique, choisi en octobre 2000 par le NIST pour être le nouveau standard de chiffrement pour les organisations du gouvernement des Etats-Unis.

Ce faisant, l'AES remplace le DES (choisi comme standard dans les années 1970) qui de nos jours devenait obsolète. L'AES a été adopté par le NIST (National Institute of Standards and Technology) en 2001. De plus, son utilisation est très pratique car il consomme peu de mémoire et n'étant pas basé sur des schémas de Feistel, sa complexité est moindre et il est plus facile à implémenter.Utilise des clés de tailles 128, 192 et 256 bits.

Pour plus de détaille voir le chapitre 2.

3.2.2 Chiffrement asymétrique

3.2.2.1 Principe

Inventé en 1977 par Diffie et Hellman, le chiffrement asymétrique, aussi connu sous le nom de chiffrement à clé publique permet de résoudre le problème de communication des clés du chiffrement symétrique, et permet aussi l'authentification, en s'aidant des fonctions de hachage.

Chaque utilisateur possède alors deux clés : une clé publique qu'il distribue à tout le monde ainsi qu'une clé privée, qu'il garde secrète. La clé publique permettra de chiffrer un message, que l'utilisateur pourra déchiffrer grâce à sa clé privée. Ainsi, seule la personne en possession de la clé de déchiffrement (privée), le destinataire donc, pourra déchiffrer le message [6].

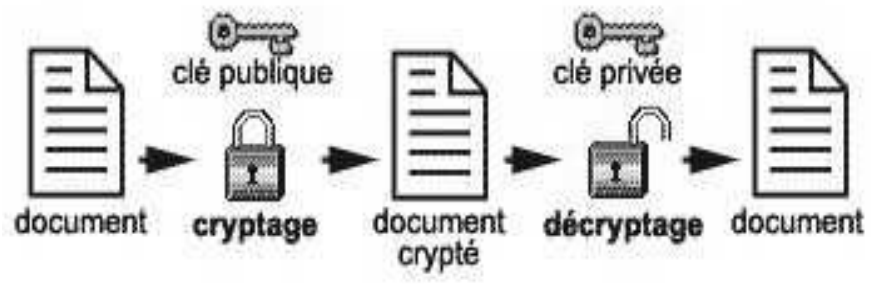


Figure I.11 : La cryptographie asymétrique

3.2.2.2 RSA (Rivest Shamir Adleman)

RSA est un algorithme de cryptographie asymétrique, très utilisé dans le commerce électronique, et plus généralement pour échanger des données confidentielles sur Internet. Cet algorithme a été décrit en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman. RSA a été breveté par le Massachusetts Institute of Technology (MIT) en 1983 aux Etats-Unis. Le brevet a expiré le 21 septembre 2000.

Cet algorithme est fondé sur l'utilisation d'une paire de clés composée d'une clé publique pour chiffrer (respectivement vérifier) et d'une clé privée pour déchiffrer (respectivement signer) des données confidentielles. La clé publique correspond à une clé qui est accessible par n'importe quelle personne souhaitant chiffrer des informations, la clé privée est quant à elle réservée à la personne ayant créé la paire de clés. Lorsque deux personnes, ou plus, souhaitent échanger des données confidentielles, une personne, nommée par convention *Alice* prend en charge la création de la paire de clés, envoie sa clé publique aux autres personnes *Bob*, *Carole*… qui peuvent alors chiffrer les données confidentielles à l'aide de

celle-ci puis envoyer les données chiffrées à la personne ayant créé la paire de clés, *Alice*. Cette dernière peut alors déchiffrer les données confidentielles à l'aide de sa clé privée.

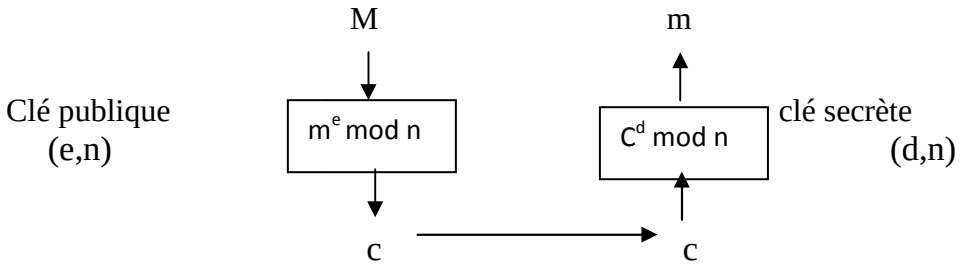


Figure I.12 : Le chiffrement à clé publique RSA.

3.2.2.3 ElGamal

L'algorithme ElGamal est un algorithme de cryptographie asymétrique basé sur les logarithmes discrets. Il a été créé par Taher Elgamal. Cet algorithme est utilisé par le logiciel libre GNU Privacy Guard, de récentes versions de PGP, et d'autres systèmes de chiffrement, et n'a jamais été sous la protection d'un brevet contrairement à RSA. Il peut être utilisé pour le chiffrement et la signature électronique. L'algorithme DSA du NIST est basé sur ElGamal [6].

L'algorithme fonctionne comme suit :

Il s'agit d'un système à clé publique dont la sécurité repose, comme le protocole de Diffie et Hellman, sur la difficulté de calculer le logarithme discret.

• Alice calcule $\mathbf{h = g^x}$ avec g, h$\in Z_p$ pour un grand nombre premier p, et divulgue sa clé publique (p, g, h). La valeur x est sa clé privée.

• Si Bob veut envoyer un message à Alice, il convertit d'abord son message sous la forme d'un nombre m$\in Z_p$.

• Bob génère un nombre entier k aléatoirement et calcule $\mathbf{c_1 = g^k}$ et $\mathbf{c_2 = m.h^k}$. Il envoie (c_1, c_2) à Alice.

• Alice peut reconstruire le message initial m en calculant $\mathbf{c_2 / c_1^x}$.

On remarque que :

$$\frac{c_2}{c_1^x} = \frac{m \cdot h^k}{g^{xk}} = \frac{m \cdot g^{xk}}{g^{xk}} = m$$

Il n'est pas obligatoire d'utiliser Z_p. Tout groupe cyclique convient.

3.2.3 Comparaison entre la cryptographie symétrique et asymétrique

Attribut	**Cryptographique symétrique**	**Cryptographique asymétrique**
Année d'existence	Des milliers	Moins de 50
Utilisation actuelle principale	Chiffrement des données en gros	Echange des clés, signature numérique
Standard actuel	DES, IDEA et Rijndeal	RSA, diffie-Hellman, DSA (la technologie ECC est une nouvelle venue prometteuse)
Vitesse de chiffrement/ déchiffrement	rapide	Lent
clés	Secrète partagée par au moins deux personnes	Privée : gardée cachée par un personne Publique : largement distribuée
Echange des clés.	Transfert risqué et difficile pour une clé secrète	Simple, moins risqué de remettre une clé publique Clé privée jamais partagée
Longueur de la clé	56 bits obsolètes 128 bits considérés comme sûr	1024suggérée (RSA) certains utilisateurs exigent 2048~172(courbe elliptique)
Confidentialité, Authentification, intégrité du message	Oui	Oui
Non-répudiation	Non besoin d'un tiers de confiance servant de témoin	Oui signatures numériques : pas besoin de tiers
Attaques	Oui	Oui

Tableau I.3 : Comparaison entre la cryptographie symétrique et asymétrique

3.3 Les algorithmes associés

3.3.1 Cryptographie hybride

La cryptographie asymétrique est intrinsèquement lente à cause des 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 ce défaut, on recourt à la cryptographie asymétrique qui travaille avec une paire de clés : la clé privée et la clé publique. La cryptographie hybride combine les deux systèmes afin de bénéficier des avantages (rapidité de la cryptographie symétrique pour le contenu du message) et utilisation de la cryptographie "lente" uniquement pour la clé.

La plupart des systèmes hybrides procèdent de la manière suivante. Une clé aléatoire est générée pour l'algorithme symétrique (3DES, IDEA, AES et bien d'autres encore), cette clé fait généralement entre 128 et 512 bits selon les algorithmes. L'algorithme de chiffrement symétrique est ensuite utilisé pour chiffrer le message. Dans le cas d'un chiffrement par blocs, on doit utiliser un mode d'opération comme par exemple CBC, cela permet de chiffrer un message de taille supérieure à celle d'un bloc. La clé aléatoire quant à elle, se voit chiffrée grâce à la clé publique du destinataire, c'est ici qu'intervient la cryptographie asymétrique (RSA ou Diffie-Hellman). Comme la clé est courte, ce chiffrage prend peu de temps. Chiffrer l'ensemble du message avec un algorithme asymétrique serait bien plus lourd, c'est pourquoi on préfère passer par un algorithme symétrique. Il suffit ensuite d'envoyer le message chiffré avec l'algorithme symétrique et accompagné de la clé chiffrée correspondante. Le destinataire déchiffre la clé symétrique avec sa clé privée et via un déchiffrement symétrique, retrouve le message.

➢ **PGP** (*Pretty Good Privacy*) est un cryptosystème inventé par Philip Zimmermann, un analyste informaticien. Philip Zimmermann a travaillé de 1984 à 1991 sur un programme permettant de faire fonctionner RSA sur des ordinateurs personnels (PGP). Il est très rapide et sûr ce qui le rend quasiment impossible à cryptanalyser.

PGP est une combinaison des meilleures fonctionnalités de la cryptographie asymétrique et symétrique. Il fonctionne suivant le principe suivant :

- **Compression :** Le texte à envoyer est compressé. Cette étape permet de réduire le temps de transmission des données, d'économiser l'espace de disque et améliore également la sécurité.
- **Chiffrement du message :** Une clé de session aléatoire est générée. Le message est chiffré par un algorithme symétrique à l'aide d'une clé de session. L'algorithme utilisé a varié au cours de temps : il s'agissait au début d'IDEA, puis de CAST.
- **Chiffrement de la clé de session :**La clé de session est chiffrée en utilisant la clé publique du destinataire et l'algorithme RSA.
- **Envoi et réception du message :** L'expéditeur envoie couple (message chiffré, clé) de session chiffrée au destinataire. Celui-ci récupère d'abord la clé de session, en utilisant sa clé privée, puis il déchiffre le message grâce à la clé de session.

3.3.2 Les fonctions de hachage

Les fonctions de hachage sont des fonctions à sens unique, et produisent à partir d'un texte d'une longueur quelconque, une *somme de contrôle* (aussi appelée *empreinte* ou *hash*) de longueur fixe, qu'on peut voir comme un résumé du texte (à partir duquel on ne peut pas retrouver le texte d'origine, ou du moins très difficilement).

Les fonctions de hachage doivent être résistantes aux collisions, c'est-à-dire qu'on ne peut pas trouver facilement deux messages différents ayant la même empreinte.

Fonctions de hachage usuelles :

- **MD4 et MD5 (Message Digest)** furent développées par Ron Rivest. MD5 produit des hachés de 128 bits en travaillant les données originales par blocs de 512 bits.
- **SHA-1 (Secure Hash Algorithm 1)**, comme MD5, est basé sur MD4. Il fonctionne également à partir de blocs de 512 bits de données et produit par contre des condensés de 160 bits en sortie. Il nécessite donc plus de ressources que MD5.
- **SHA-2 (Secure Hash Algorithm 2)** a été publié récemment et est destiné à remplacer SHA-1. Les différences principales résident dans les tailles de hachés possibles : 256, 384 ou 512 bits. Il sera bientôt la nouvelle référence en termes de fonction de hachage.
- **RIPEMD-160 (Ripe Message Digest)** est la dernière version de l'algorithme RIPEMD. La version précédente produisait des condensés de 128 bits mais présentait des failles de sécurité importantes. La version actuelle reste pour l'instant sure; elle produit comme son nom l'indique des condensés de 160 bits. Un dernier point la concernant est sa relative gourmandise en termes de ressources et en comparaison avec SHA-1 qui est son principal concurrent.
- **Tiger** : Tiger est une fonction de hachage cryptographique conçue par Ross Anderson et Eli Biham en 1996. Tiger fournit une empreinte sur 192 bits mais des versions sur 128 et 160 bits existent aussi. Ces versions raccourcies prennent simplement les premiers bits de la signature de 192 bits.

3.3.3 Signature numérique

La signature numérique (parfois appelée **signature électronique**) est un mécanisme permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la signature manuscrite d'un document papier. Un mécanisme de signature numérique doit présenter les propriétés suivantes :

- Il doit permettre au lecteur d'un document d'identifier la personne ou l'organisme qui a apposé sa signature.
- Il doit garantir que le document n'a pas été altéré entre l'instant où l'auteur l'a signé et le moment où le lecteur le consulte.

Pour cela, les conditions suivantes doivent être réunies :

- **Authentique**: L'identité du signataire doit pouvoir être retrouvée de manière certaine.
- **Infalsifiable**: La signature ne peut pas être falsifiée. Quelqu'un ne peut se faire passer pour un autre.
- **Non réutilisable**: La signature n'est pas réutilisable. Elle fait partie du document signé et ne peut être déplacée sur un autre document.
- **Inaltérable**: Un document signé est inaltérable. Une fois qu'il est signé, on ne peut plus le modifier.
- **Irrévocable**: La personne qui a signé ne peut le nier.

La signature électronique n'est devenue possible qu'avec la cryptographie asymétrique.
Elle se différencie de la signature écrite par le fait qu'elle n'est pas visuelle, mais correspond à une suite de nombres.

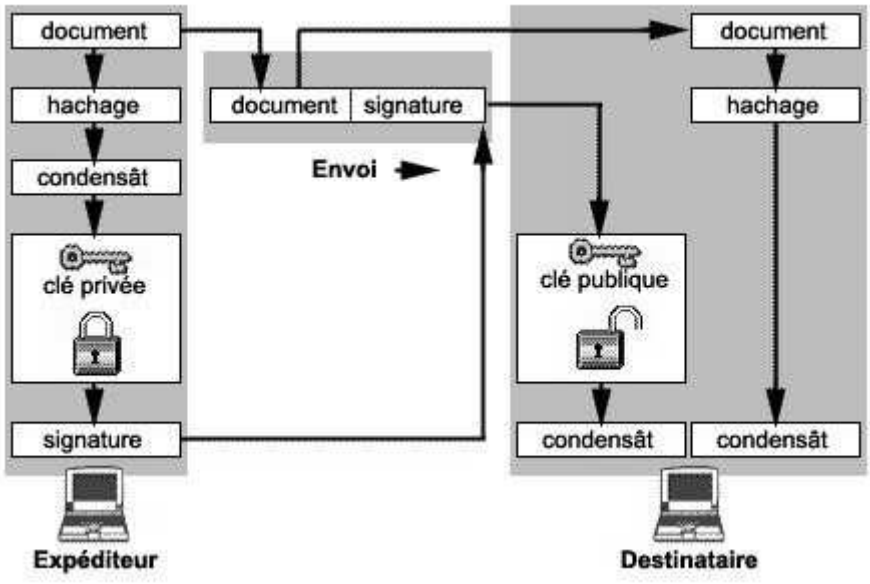


Figure I.13: shéma de hachage et signature numérique

3.3.4 Certificat numérique

C'est un certificat qui permet à une entreprise de **s'authentifier** de manière certaine, unique et sécurisée pour transmettre ou recevoir des informations numériques sensibles.

Un certificat numérique est délivré par un organisme **agréé** par le ministère chargé des finances. Cet organisme est un Prestataire de Service de Certification électronique (PSCe). Un certificat numérique autorise son possesseur à signer des engagements au nom de son entreprise dans les procédures pouvant être télétransmises : immatriculation de véhicules, déclaration de TVA, marchés publics, …

Il peut se présenter sous forme de clé USB ou de carte à puce. Il existe aussi des certificats numériques sous forme logicielle, mais ils ne sont pas autorisés pour le SIV (Système d'Immatriculation des Véhicules) [6].

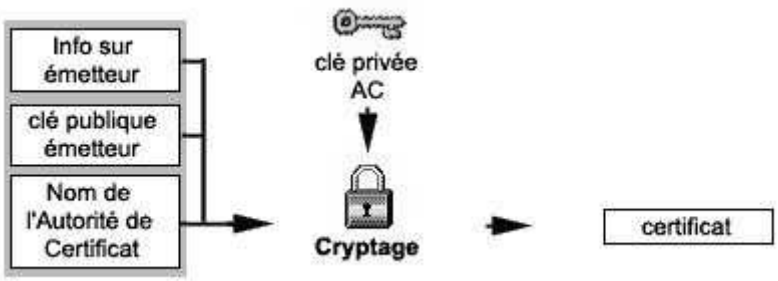


Figure I.14 : certificat numérique

3.4 Cryptographie quantique

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

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

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

Pourquoi utiliser le système de cryptographie quantique pour transmettre une clé, et non le message en lui-même ?

Pour deux raisons essentielles :

- Les bits d'informations communiqués par les mécanismes de la cryptographie quantique ne peuvent être *aléatoires*. Ceci ne convient pas pour un message, mais convient parfaitement bien à une clé secrète, qui doit être aléatoire.
- Même si le mécanisme de la cryptographie quantique garantit que l'espionnage de la communication sera toujours détecté, il est possible que des bits d'informations entrent en possession de l'espion avant que celui-ci ne soit détecté. Ceci est inacceptable pour un message, mais sans importance pour une clé aléatoire qui peut être simplement jetée en cas d'interception.

4. La cryptanalyse

4 .1 Définition

Si le but de la cryptographie est d'élaborer des méthodes de protection, le but de la cryptanalyse est au contraire de casser ces protections. On appelle attaque une tentative de cryptanalyse.

Un algorithme de chiffrement est considéré comme sûr si ce dernier résiste à des attaques dont le besoin en temps ou en espace de mémoire est raisonnable. En d'autres termes, ces derniers ne permettent pas de découvrir la clé secrète, ni le message en clair. Dans la plupart des cas, le chiffrement est d'autant plus sûr que la clé a une grande taille. Par conséquent, la taille de la clé est un paramètre essentiel de la sureté d'un algorithme de chiffrement. Dans la littérature de la cryptanalyse, il existe un vaste choix de méthode d'attaque pour retrouver la clé secrète. Ces attaques différent par le coût, le temps de calcul, la connaissance du matériel, l'algorithme de chiffrement utilisé.

4 .2 Classification des attaques en fonction des données disponible

Les techniques d'attaques peuvent être classifiées dans 4 catégories :

a) Attaque à texte chiffré connu : Cette famille d'attaques fait comme hypothèse de base que l'attaquant a uniquement les messages chiffrés.

b) Attaque à texte clair connu : Cette famille fait comme supposition qu'Eve a les textes clairs et les chiffrés de ces clairs.

c) Attaque à texte chiffré choisi : Cet ensemble d'attaques fait comme hypothèse que l'attaquant peut choisir les messages chiffrés et recevoir leur clair.

d) Attaque à texte clair choisi : Cet ensemble d'attaques suppose qu'Eve peut choisir les textes clairs et obtenir leur chiffré. Cette attaque est largement réalisable dans les algorithmes asymétriques où la clé de chiffrement est publique. Ainsi, l'attaquant peut chiffrer l'ensemble des messages qu'il choisit. Le but final étant de choisir les bons couples de clair/chiffré pour retrouver une information utile.

4.3 Les différentes attaques

a) L'analyse des fréquences

Exposée pour la première fois par Al-Kindi au IXe siècle, l'analyse fréquentielle permet de déchiffrer des chiffrements simples. Dans le cas d'une substitution, les caractères sont représentés par autre chose, mais leurs fréquences d'apparition ne changent pas ; on peut donc alors facilement retrouver le message initial, connaissant les fréquences normales d'apparition des caractères dans la langue du message.

b) L'indice de coïncidence

L'indice de coïncidence, inventé en 1920 par William F. Friedman, permet de calculer la probabilité de répétitions des lettres du message chiffré. Il est souvent couplé avec l'analyse fréquentielle. Cela permet de savoir le type de chiffrement d'un message (chiffrement mono-alphabétique ou poly-alphabétique) ainsi que la longueur probable de la clé.

c) L'attaque par mot probable

L'attaque par mot probable consiste à supposer l'existence d'un mot probable dans le message chiffré. Il est donc possible d'en déduire la clé du message si le mot choisi est correct. Ce type d'attaque a été mené contre la machine Enigma durant la Seconde Guerre mondiale.

d) L'attaque par force brute

L'attaque par force brute consiste à tester toutes les solutions possibles de mots de passe ou de clés. C'est le seul moyen de récupérer la clé dans les algorithmes les plus modernes et encore inviolés comme AES. Il est peu utilisé pour des mots de passe possédant un très grand nombre de caractères car le temps nécessaire devient alors trop important. De même plusieurs brevets rendent cette méthode inefficace, comme celui de Bell ou d'IBM.

e) 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 2^{64} hachages d'entrées distinctes.

f) Cryptanalyse différentielle

Découverte à la fin des années 1980, la cryptanalyse différentielle analyse comment les différences dans les messages clairs influencent les différences dans les messages chiffrés.
L'étude de ces différences renseigne alors l'attaquant de certains bits de la clé. Il faut donc alors être dans une situation d'attaque à texte clair choisi (l'attaque à texte clair connu est aussi possible, mais nécessite un plus grand nombre de textes connus). Nous n'allons pas plus développer cette attaque, qui serait bien trop longue à décrire en détail dans ce travail.

g) 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. Les algorithmes plus récents sont insensibles à cette attaque.

h) Autres attaques

Il existe encore d'autres types d'attaque non abordées ici. Le secteur de la cryptanalyse est en perpétuelle évolution, chacun peut trouver de nouvelles méthodes de cryptanalyse avec beaucoup de travail et de connaissances.
Les attaques présentées ici ne sont faisables que sur des algorithmes de chiffrements symétriques, les systèmes de chiffrements utilisant un lien entre la clé publique et la clé privée, on essayera plutôt d'exploiter ce lien pour trouver la clé privée. Dans le cas de RSA, il faut factoriser le nombre *e*, afin de trouver ses deux diviseurs, *p* et *q*, on pour alors retrouver facilement la clé privée *d*. La sécurité de cet algorithme réside alors dans la difficulté de factoriser un nombre qui est le produit de deux grands nombres premiers [7].

4 .4 Réussite de l'attaque

Il existe plusieurs découvertes qui permettent de dire qu'un algorithme de chiffrement est cassé. La liste suivante regroupe des exemples :

1. Si nous retrouvons la clé de chiffrement
2. Si nous retrouvons le message clair
3. Si nous retrouvons un sous-ensemble du message clair
4. Si nous retrouvons une information du message clair

Il existe un ensemble très vaste d'attaques. En effet, il existe autant de techniques d'attaque que d'algorithmes de chiffrement et certaines de ces méthodes sont généralisées pour une

famille d'algorithmes de chiffrement. Ni la liste exhaustive de ces attaques, ni de leur contremesure ne pourrait être donnée ici. Pour les lecteurs désirant apprendre des contremesures sur les attaques présentes dans ce document, est un bon choix. Toutefois, essayons de détailler quelques-unes de ces attaques pour voir d'où a débuté la cryptanalyse jusqu'à arriver à l'attaque utilisée dans ce travail avec l'aide de l'apprentissage automatique [8].

5. Conclusion

Dans ce chapitre, nous avons tenté de dresser un état de l’art sur la cryptographie, et son importance dans la protection des informations pendant le transfert. Ensuite, Nous avons abordé la classification des algorithmes cryptographique avec des exemples sur chaque classe d’algorithme .Enfin, nous avons vu quelques concepts sur la cryptanalyse et les différentes techniques utilisent dans cette opération.

Le chapitre suivant est consacré à la présentation détaillé de l’algorithme AES que nous avons choisi

Chapitre II

L'algorithme AES

1. Introduction

Après l'étude des différentes classes des algorithmes cryptographique dans le chapitre précédent. Nous prenons dans ce chapitre l'algorithme AES (l'Advanced Encryption Standard) en détail, qui est un standard de cryptage symétrique et largement adopté dans l'industrie du matériel cryptographique. Son principal avantage réside dans sa rapidité du chiffrement et du déchiffrement qui est nettement plus vite sur des cartes électroniques dédiées du point de vue hardware (cartes à puces, systèmes électroniques de communication).

2. Historique de l'algorithme AES

En 1997, les faiblesses de DES Data Encryption Standard deviennent réellement inquiétantes, à tel point que le NIST (*National Institute of Standards and Technology*) fait à nouveau un appel d'offre pour l'élaboration d'un nouveau système cryptographique **l'Advanced Encryption Standard** (**AES**). Le cahier des charges établi par le NIST comporte les points suivants :

- Grande sécurité.
- Large portabilité. Puisque cet algorithme doit remplacer DES, il est destiné à des cartes à puces, des processeurs peu performants, des processeurs spécialisés, etc.
- Rapidité.
- Lecture aisée de l'algorithme, puisque destiné à un usage publique.
- Le chiffrement se fait par blocs de 128 bits. Les clés comportent 128, 192 ou 256 bits.

Au 15 juin 1998, date de la fin des candidatures, 15 projets sont retenus . Certains sont l'oeuvre d'entreprises (IBM), d'autres regroupent des universitaires (CNRS), les derniers sont écrits par quelques personnes. Pendant deux ans, les algorithmes sont évalués par des experts. Ceci conduit à la désignation, en août 1999, de 5 finalistes :

- MARS (*IBM*). Complexe, rapide, haute marge de sécurité.
- RC6 (*RSA Laboratories*). Très simple, très rapide, faible marge de sécurité.
- Rijndael (*Joan Daemen, Vincent Rijmen*). Propre, rapide, bonne sécurité.
- Serpent (*Ross Anderson, Eli Biham, Lars Knudsen*). Propre, lent, très haute marge de sécurité.
- Twofish (*Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Niels Ferguson, Chris Hall*). Complexe, très rapide, haute marge de sécurité.

Le 2 octobre 2000, le NIST donne sa réponse. L'algorithme retenu est **Rijndael**, contraction des noms des deux inventeurs belges, Vincent **Rij**men et Joan **Dae**men. Il est retenu pour son bon compromis entre sécurité, performance, efficacité, facilité d'implémentation et flexibilité. Rijndael travaille par blocs de **128 bits** et il est symétrique. La taille de la clé est généralement de 128 bits avec les variantes 192 et 256 bits. Dans ce qui suit, on essaie de réaliser l'algorithme de ***« RIJNDAEL »*** qui représente le standard **AES** en considérant la norme de base qui est une clé de 128 bits.

3. Présentation de l'algorithme AES

L'algorithme AES se présente en deux temps, tout d'abord une procédure d'expansion de la clef, puis la fonction de chiffrement. La fonction de chiffrement se divise en trois :

- Une transformation initiale avec la clé.
- Une série de tours (round).
- Une transformation finale.

Le nombre de tours s'établit en fonction de la taille des blocs et de la clé :

- 9 tours si la taille des blocs et de la clé sont de 128 bits.
- 11 tours si la taille des blocs ou de la clé est de 192 bits.
- 13 tours si la taille des blocs ou de la clé est de 256 bits.

3.1 Chiffrement

3.1.1 Mise en forme des entrées

Comme tout système cryptographique symétrique, AES dispose de deux entrées, à savoir le texte clair à chiffrer (plaintext) ainsi que la clé de cryptage (key). Si la longueur de la clé est fixe (128 bits), la longueur du texte clair est variable d'une application à l'autre.

AES étant un algorithme de bloc, il doit commencer par découper le texte clair en blocs de 128 bits.

Ensuite, le bloc de texte clair en cours de chiffrage ainsi que la clé sont mis sous forme matricielle. Comme il s'agit de la version 128 bits de l'algorithme, les matrices obtenues sont carrées (4*4). Elles comptent 16 éléments et chacun de ces éléments contient 1 byte ($16 \cdot 8 = 128$ bits). Pour d'évidentes raisons de taille, les valeurs binaires seront notées sous forme hexadécimale. On considère les matrices suivantes, contenant des valeurs arbitraires

State

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

Figure II.1 Texte claire (State)

Cipher key

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

Figure II.2 Clé(Key)

3.1.2 Première étape – Round 0

La première étape consiste à combiner la matrice State (le bloc de texte clair) avec la clé. Cette opération s'appelle Add Round Key. Elle consiste à additionner modulo 2 chaque byte de la matrice State avec son homologue de la matrice Key (la clé originale). On obtient ainsi la nouvelle matrice State (elle désigne l'état actuel du bloc en cours de chiffrage). Donc notre matrice est transfère comme ca :

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

Figure II.3 le tableau de texte après Add Round Key

3.1.3 Deuxième étape – Rounds 1-9

Cette partie du processus de chiffrage dépend de la taille de la clé utilisée. Cette deuxième étape compte 9 itérations. Chacune de ces 9 itérations effectue successivement les quatre opérations détaillées ci-dessous :

3.1.3.1 Sub Bytes :

Cette opération consiste à remplacer chaque byte de la matrice State par une autre valeur. La substitution se fait à l'aide d'une table appelée S-Box et concrétise le principe de confusion de Shannon, ceci obscurcir la relation entre le texte en clair et le texte chiffré. Les bytes que cette table contient sont les bytes de remplacement. Les valeurs de la S-Box sont construites par deux fonctions dans deux champs finis différents (GF(28) et GF(2)).

hex	y															
	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

x

S-BOX

Figure II.4 Bytes substitution box

La substitution se fait de la manière suivante :

Pour chaque élément de cette matrice, procéder comme suit :

Le premier caractère hexadécimal de l'élément indique une ligne de la S-Box tandis que le deuxième indique une colonne.

Le byte se trouvant à l'intersection ligne-colonne dans la S-Box est celui qui doit être substitué à celui de la matrice State.

Ainsi, le premier byte de State (premier élément de la matrice, soit 19h) sera remplacé par le byte se trouvant à l'intersection de la ligne 1 et de la colonne 9 de la S-Box, c'est-à-dire d4h.

Le deuxième byte de State (3Dh) sera remplacé par le byte se trouvant à l'intersection de la ligne 3 et de la colonne D de la S-Box, c'est-à-dire 27h, et ainsi de suite...Après traitement des seize bytes, la matrice State devient :

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

Figure II.5 le texte après l'opération Sub Bytes

3.1.3.2 Shift Rows

Cette opération consiste à décaler des lignes dans la matrice State. En ce sens, elle garantit le principe de diffusion de Shannon. De faibles changements dans le texte clair impliquent de grands changements dans le texte chiffré. Les décalages ne modifient pas les valeurs des bytes, mais changent leur ordre. Les décalages se font comme suit :

- La première ligne n'est pas décalée.
- La deuxième ligne est décalée de 1 byte vers la gauche.
- La troisième ligne est décalée de 2 bytes vers la gauche.
- La quatrième ligne est décalée de 3 bytes vers la gauche.
- La matrice State issue de l'opération précédente (Sub Bytes) est la suivante:

d4	e0	b8	1e
bf	b4	41	27
11	98	5d	52
ae	f1	e5	30

Figure II.6 le texte après l'opération Shift Rows

Après les décalages, on obtient la nouvelle matrice State :

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
ae	f1	e5	30

Figure II.7 le texte après décalage

3.1.3.3 Mix Column

Cette opération participe également au principe de diffusion de Shannon. Elle consiste à multiplier une matrice constante avec la matrice State :

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

Figure II.8 la matrice de Mix Column

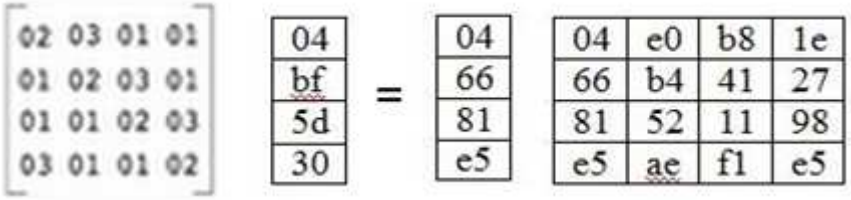


Figure II.9 le texte après Mix Column

La particularité de ce calcul est que les opérations d'addition et de multiplication se font dans le champ de Galois GF(28). Si les additions se réalisent par un simple XOR, les multiplications sont plus problématiques ... Multiplications dans le champ de Galois GF(28) :

Deux méthodes permettent de calculer les produits dans GF(28).
Considérons le premier produit, à savoir 02•D4. Il faut voir les deux bytes à multiplier comme deux polynômes du septième degré a(x), respectivement b(x) dont les coefficients sont donnés par les 8 bits composant le byte. Reprenons le calcul ci-dessus :
Result[1,1] = 02•D4 XOR 03•BF XOR 01•5D . 01•30
02 = 00000010 a(x)
D4 = 11010100 b(x)
a(x) = 0x7+0x6+0x5+0x4+0x3+0x2+1x1+0x0 = x
b(x) = 1x7+1x6+0x5+1x4+0x3+1x2+0x1+0x0 = x7+x6+x4+x2

Maintenant que a(x) et b(x) sont connus, il reste à les multiplier modulo le polynôme de degré 8 m(x) = x8+x4+x3+x+1. Ce polynôme est irréductible dans GF(28), il ne peut être divisé que par lui-même et 1 :

a(x)b(x) mod m(x) = x(x7+x6+x4+x2) mod x8+x4+x3+x+1

a(x)b(x) mod m(x) = x8+x7+x5+x3 mod x8+x4+x3+x+1

Le résultat de cette expression est le polynôme représentant le produit 02•D4 dans le champ de Galois GF(28).

Ce calcul ne sera mené à terme car l'objectif était juste de montrer le lien entre Rijndael et les champs de Galois.

Donc le résultat est :

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

Figure II.10 le texte après les champs de Galois.

3.1.3.4 Add Round Key

Lors du processus de chiffrage, Rijndael transforme la clé (matrice Key). A chaque itération (Round), une matrice Key différente est utilisée (RoundN Key). Ceci permet d'éliminer les attaques liées à la clé (Biham) en faisant disparaître la symétrie. Pour obtenir les 10 nouvelles clés nécessaires, Rijndael procède à une opération appelée Key Scheduling ou KeyExpansion.

Si la clé change à chaque Round, l'opération Add Round Key, elle, reste simple. Elle consiste, tout comme celle de la première étape, à additionner modulo 2 (XOR) la matrice State et la clé du Round en cours (RoundN Key).

Donc le RoundN Key :

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

Figure II.11 la clé après l'opération XOR

Faire l'opération suivant pour chaque matrice :

04		a0		a4
66	+	fa	=	9c
81		fe		7f
e5		17		f2

Figure II.12 l'opération de Add Round Key

Le résultat est :

a4	68	6b	02
9c	9f	5b	6a
7f	35	ea	50
f2	2b	43	49

Figure II.13 le résultat de Add Round Key

3.2 Troisième étape – Round 10

Cette étape est quasiment identique à l'un des neuf Rounds de la deuxième étape. La seule différence est que, dans ce dernier Round, l'opération Mix Columns n'est pas effectuée. Apres le 10 rounds la forme générale de la matrice composé comme ca :

39	02	dc	19
25	dc	11	6a
84	09	85	0b
1d	fb	97	32

Figure II.14 le texte après Round 10

Organisation des clés

Les différents rounds keys sont dérivées de la clé de cryptage en utilisant un procédé bien défini de **key schedule**.

Le nombre de round keys nécessaires pour chiffrer des informations dépend de la taille du bloc en entrée : par exemple, pour un bloc de 128 bits, il faudra 11 round keys : 1 pour l'étape initiale, 1 pour l'étape finale, et 9 pour les étapes standard (en effet, $N_r = 10$).

Ce qui suit est pris dans le cas de 128 bits. Les calculs dans le cas de plus de 6 colonnes dans la matrice de la clé de cryptage (donc 256 bits) sont légèrement différents, mais ne sont pas traités ici.

La clé de chiffrage est notée :

$$K = \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}$$

et on appelle W_i la i-ème colonne de K.

Le procédé de key schedule est alors le suivant :

$K = [W_0, W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8, W_9, W_{10}...]$ où $W_5 = W_4 + W_3$, $W_6 = W_5 + W_4$....

Clé de cryptage

Le calcul pour le cas de W_4, W_8, etc… est un peu différent :

$$W_i = W_{i-4} + SubByte\big(RotByte(W_{i-1})\big) + \begin{pmatrix} u \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

où SubByte correspond à la transformation de chaque élément (octet) de la colonne, RotByte correspond à une permutation circulaire des éléments d'une colonne verticalement vers le haut. De plus $u = x^{i/4}$ dans le corps GF(2^8).

Figure II.15 resultat finale de **key schedule**

3.3 Déchiffrement

Toutes les opérations réalisées lors du chiffrage sont réversibles, à condition d'avoir la clé bien entendu :

A procède à l'extension de la clé de la même manière que B.

Les additions modulo 2 (XOR) effectuées lors de l'opération Add Round Key sont réversibles ((A XOR B) XOR B = A).

L'opération Sub Bytes est inversée en utilisant la table S inverse (Inverse S-Box). Si par exemple la S-Box indique le byte ed (ligne 5, colonne 3), alors la Inverse S-Box restituera le byte 53 (ligne e, colonne d).

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

Figure II.16 Inverse S-Box

Les décalages de l'opération Shift Rows sont inversés, c'est-à-dire effectués vers la droite.

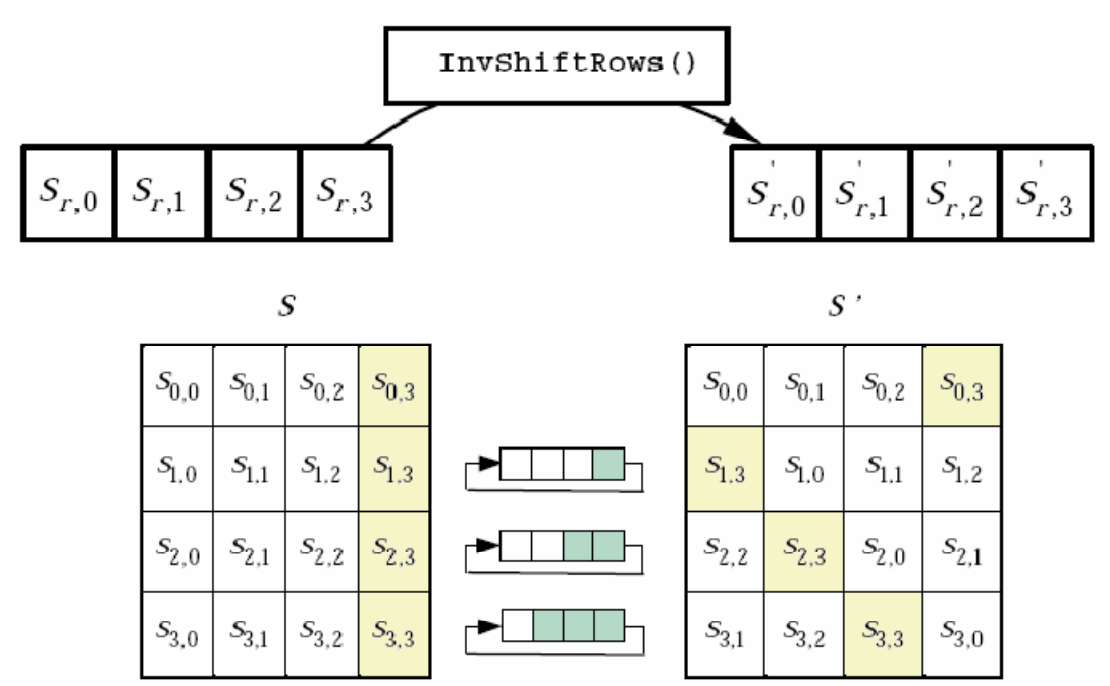


Figure II.17 Inverse Shift Rows

La multiplication matricielle de l'opération Mix Columns nécessite une inversion de la matrice. Une fois la matrice inverse obtenue, la manipulation est la même que pour l'opération Mix Columns.

La matrice constante inverse est de :

14	11	13	9
9	14	11	13
13	9	14	11
11	13	9	14

Figure II.18 La matrice constante inverse

A cause du calcul de la matrice inverse, le déchiffrage est environ 30% plus lent que le chiffrage.

Equivalent Inverse Cipher.

Version utilisant un ordre des transformations identique à celui du Cipher (chaque transformation étant remplacée par son inverse).

Propriétés algorithmiques :

1. Il est possible d'inverser l'ordre des transformation SubBytes() et ShiftRows(), et également pour InvSubBytes() et InvShiftRows().

2. Les transformations MixColumns() et InvMixColumns() sont linéaires vis-à-vis de la colonne d'entrée, c'est-à-dire :

InvMixColumns(state XOR RoundKey) =InvMixColumns(state) XOR InvMixColumns(RoundKey) [9] .

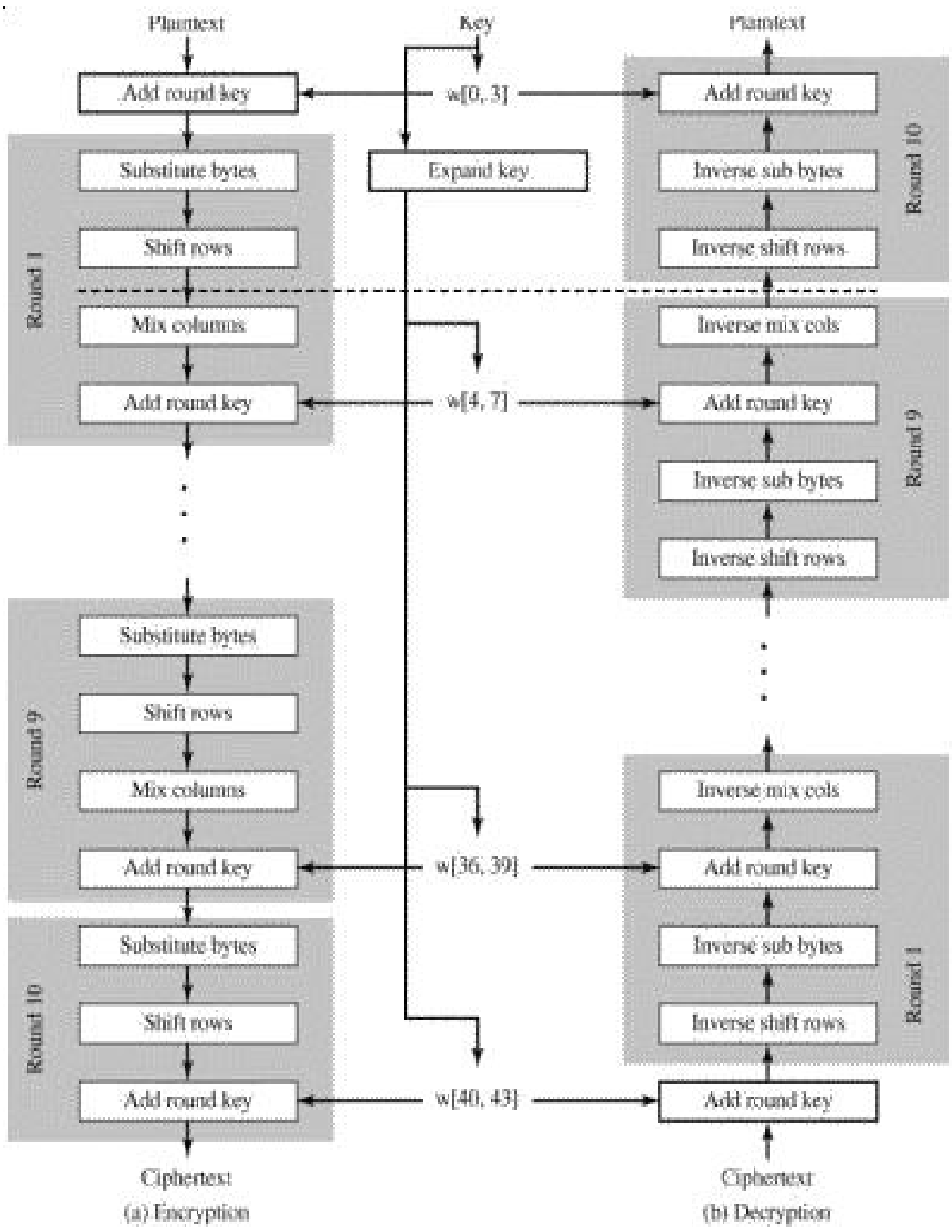


Figure II.19 *Schéma général Chiffrage/Déchiffrage*

4. Conclusion

Dans ce chapitre, nous avons représenté l'algorithme de la cryptographie AES d'une manière détaillé, nous signalons que l'avantage majeur de ce algorithme est l'exploitation des simples notions mathématique c'est pourquoi son utilisation impose un calcul simple et facile pour l'utilisateur autorisé, mais ce calcul devient très complexe dans le cas d'un ennemi qui applique une attaque exhaustive.

Chapitre III

Implémentation

1. Introduction

Dans le but de bien réaliser notre application qui intègre tous les algorithmes déjà vus dans le chapitre précédent, mais aussi afin de bien les implémenter on avait besoin d'utiliser un langage à la fois simple à manipuler, qu'il soit un langage orienté objet, qu'il puisse effectuer toutes les taches d'un langage de haut niveau (bureautique, graphique, multimédia, base de donné, environnement de développement, etc.). Notre choix porte sur le langage java.

2. Pour quoi java ?

Nous avons choisi Java comme langage de programmation pour réaliser notre système pour les raisons suivantes:

- ✓ Maitriser un nouveau langage de programmation et comprendre bien le concept orienté objet.
- ✓ Et les exceptions qui distingue ce langage par rapport les autres langages comme la facilité (leur syntaxe est une extension de langage C)…etc

2.1 Présentation générale

Apparu fin 1995 début 1996 et développé par Sun Microsystems Java s'est très rapidement taillé une place importante en particulier dans le domaine de l'internet et des applications client-serveur.

Destiné au départ à la programmation de centraux téléphoniques sous l'appellation de langage "oak", la société Sun a eu l'idée de le recentrer sur les applications de l'internet et des réseaux. C'est un langage en évolution permanente Java 2 est la version stabilisée de java fondée sur la version initiale 1.2.2 du JDK (Java Development Kit de Sun : http://java.sun.com).

Les objectifs de java sont d'être multi-plateformes et d'assurer la sécurité aussi bien pendant le développement que pendant l'utilisation d'un programme java. Il est en passe de détrôner le langage C++ dont il hérite partiellement la syntaxe mais non ses défauts (l'héritage multiple, les pointeurs, la surcharge des opérateurs). Comme C++ et Delphi, java est algorithmique et orienté objet à ce titre il peut effectuer comme ses compagnons, toutes les tâches d'un tel langage (bureautiques, graphiques, multimédias, bases de données, environnement de développement, etc...). Son point fort qui le démarque des autres est sa portabilité due (en théorie) à ses bibliothèques de classes indépendantes de la plate-forme, ce qui est le point essentiel de la programmation sur internet ou plusieurs machines dissemblables sont interconnectées. La réalisation multi-plateformes dépend en fait du système d'exploitation et

de sa capacité à posséder des outils de compilation et d'interprétation de la machine virtuelle Java.

Actuellement ceci est totalement réalisé d'une manière correcte sur les plates-formes Windows et Solaris, un peu moins bien sur les autres semble-t-il.
En bref on dit que java est le premier langage du troisième millénium actuel.

2.2 Caractéristiques du langage java

Java possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

- **Simplicité :**

Nous avons voulu créer un système qui puisse être programmé simplement, sans nécessiter un apprentissage ésotérique, et qui tire parti de l'expérience standard actuelle. En conséquence, même si nous pensions que C++ ne convenait pas, Java a été conçue de façon relativement proche de ce langage dans le dessein de faciliter la compréhension du système. De nombreuses fonctions compliquées, mal comprises, rarement utilisées, de C++, qui nous semblaient par expérience apporter plus d'inconvénients que d'avantages, ont été supprimées de Java.

- **Orienté objet :**

Pour rester simples, disons que la conception orientée objet est une technique de programmation qui se concentre sur les données (les objets) et sur les interfaces avec ces objets. Pour faire une analogie avec la menuiserie, on pourrait dire qu'un menuisier "orienté objet" s'intéresse essentiellement à la chaise (l'objet) qu'il fabrique et ensuite aux outils utilisés pour la fabriquer. Par opposition, le menuisier "non orienté objet" penserait d'abord à ses outils. Les facilités orientées objet de Java sont essentiellement celles de C++...

- **Fiabilité :**

Java a été conçu pour que les programmes qui l'utilisent soient fiables sous différents aspects. Sa conception encourage le programmeur à traquer préventivement les éventuels problèmes, à lancer des vérifications dynamiques en cours d'exécution et à éliminer les situations génératrices d'erreurs...

La seule et unique grosse différence entre C/C++ et Java réside dans le fait que ce dernier intègre un modèle de pointeur qui écarte les risques d'écrasement de la mémoire et d'endommagement des données.

- **Java assure la gestion de la mémoire :**

L'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au garbage collector qui restitue les zones de mémoire laissées libres suite à la destruction des objets.

- **Architecture neutre :**

Le compilateur génère un format de fichier objet dont l'architecture est neutre — le code compilé est exécutable sur de nombreux processeurs, à partir du moment où le système d'exécution de Java est présent. Pour ce faire, le compilateur Java génère des instructions en bytecodes (ou pseudo-code) qui n'ont de lien avec aucune architecture d'ordinateur particulière. Au contraire, ces instructions ont été conçues pour être à la fois faciles à interpréter, quelle que soit la machine, et faciles à traduire à la volée en code machine natif.

- **Portabilité :**(il est indépendant de toute plate-forme)

Il n'y a pas de compilation spécifique pour chaque plate forme. Le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine. Cette indépendance est assurée au niveau du code source grâce à Unicode et au niveau du byte code.

- **Interprété :**

La source est compilé en pseudo code ou byte code puis exécuté par un interpréteur Java : la Java Virtual Machine (JVM). Ce concept est à la base du slogan de Sun pour Java : WORA (Write Once, Run Anywhere : écrire une fois, exécuter partout). En effet, le byte code, s'il ne contient pas de code spécifique à une plate-forme particulière peut être exécuté et obtenir quasiment les mêmes résultats sur toutes les machines disposant d'une JVM.(on peut dit que java possède un semi compilateur et un semi interpréteur pour quoi on dit comme ça parce que la compilation extraire le programme en code machine et le cas de nous on obtient pas le code machine on obtient le code en octet aussi le semi interprétation parce que on n'interprète pas depuis le code sourc , on interprète depuis le code en octet).

- **Performances élevées :**

En règle générale, les performances des bytecodes interprétés sont tout à fait suffisantes ; il existe toute fois des situations dans lesquelles des performances plus élevées sont nécessaires. Les bytecodes peuvent être traduits à la volée (en cours d'exécution) en code machine pour l'unité centrale destinée à accueillir l'application.

- **Multithread :**

Il permet à nous l'utilisation de threads qui sont des unités d'exécution isolées. La JVM, elle même, utilise plusieurs threads. Les avantages du multithread sont une meilleure interréactivité et un meilleur comportement en temps réel.

- **Java, langage dynamique :**

Sur plusieurs points, Java est un langage plus dynamique que C ou C++. Il a été conçu pour s'adapter à un environnement en évolution constante. Les bibliothèques peuvent ajouter librement de nouvelles méthodes et variables sans pour autant affecter leurs clients. La recherche des informations de type exécution dans Java est simple [10].

3. Pour quoi NetBeans ?

On a distingué l'environnement de développement NetBeans parce que il est faciles à comprendre et à utiliser, très riche, et disponible gratuitement, pour compiler et exécuter les programmes Java, les environnements de développement intégrés peuvent nécessiter des ressources importantes et être lourds à utiliser pour de petits programmes. Comme solution intermédiaire, vous disposez des éditeurs de texte appelant le compilateur Java et exécutant les programmes Java. Lorsque vous aurez maîtrisé les techniques présentées.

4. Présentation général de NetBeans

4.1. Définition

NetBeans IDE est un environnement de développement intégrer (On peut y écrire, compiler et exécuter les programmes. Un IDE fournit aussi un utilitaire d'Aide qui décrit tous les éléments du langage et te permet de trouver et de corriger plus facilement les erreurs dans tes programmes.) Est à l'origine un EDI Java. NetBeans fut développé à l'origine par une équipe d'étudiants à Prague, racheté ensuite par Sun Microsystems. Quelque part en 2002, Sun a décidé de rendre NetBeans open-source. Mais NetBeans n'est pas uniquement un EDI Java. C'est également une plateforme, vous permettant d'écrire vos propres applications Swing. Sa conception est complètement modulaire : Tout est module, même la plateforme. Ce qui fait de

NetBeans une boite à outils facilement améliorable ou modifiable. La licence de NetBeans permet de l'utiliser gratuitement à des fins commerciales ou non. Elle permet de développer tous types d'applications basées sur la plateforme NetBeans. Les modules que vous pourriez écrire peuvent être open source comme ils peuvent être closed-source, Ils peuvent être gratuits, comme ils peuvent être payants.

4.2. Caractéristiques

- L'environnement de développement peut être étendu par un ensemble de plugins que la communauté des contributeurs développe.
- NetBeans est un moteur de graphique servant de base logicielle pour le développement de RAD (Rich Application Desktop). Il offre un framework RCP (Rich Client Programming) pour développer tout type d'application graphique en utilisant l'environnement graphique Swing de java.
- **Interopérabilité**

  Il est possible d'importer des projets Eclipse vers NetBeans avec NetBeans IDE 4.1 ou supérieur, il suffit d'installer 'Eclipse Projet Importer'.

- **Modularité**

  -NetBeans possède un module de facilitation et de hiérarchie classloader.

  -Un module dépendance influence runtime classpath.

  -Un module peut tourner sur d'autre module.

- **Coopération de modules**

  -Un module possède une fenêtre, un autre une action.

  -Windows peut avoir des associés contextes.

  -Sélection fenêtre définit le contexte mondial.

  -Eléments de l'interface utilisateur mise à jour en fonction de son classloader.

  -Le menu, la barre d'outils des éléments obtenus fusionnés par le cadre de NetBeans.

  -Windows Layout.

- **Il intègre**

  -Ruby Support.

  -Swing GUI développement.

4.3. Avantages de NetBeans

NetBeans fournit un environnement riche pour le règlement de problèmes et l'optimisation des applications.

- Intégration de :

  -Swing.

  -La plut part des outils java.

  -Window System.

  -Docking cadre extensions autour de Swing.

  -Module Système.

- Mise à jour automatique.
- Exécuter des morceaux de code bien spécifiques à un moment donné (en utilisant les points d'arrêt comme délimiteurs).
- Suspendre l'exécution lorsqu'une condition spécifiée est rencontrée (comme par exemple quand un itérateur atteint une certaine valeur).
- Suspendre l'exécution lors d'une exception, soit à la ligne de code qui a provoqué l'exception, ou dans l'exception elle-même.
- Pister la valeur d'une variable ou d'une expression.
- Pister l'objet référencé par une variable (fixed watched).
- Fixer le code à la volée et continuer la session de débogage.
- Suspendre les threads individuellement ou collectivement.
- Revenir au début d'une méthode appelée précédemment (pop a call) dans le call stack actuelle.
- Exécuter de multiples sessions de débogage en même temps. Par exemple, on pourrait recourir à cette capacité pour déboguer une application client/serveur.

5. Description de l'application

5.1 Structure de l'application

Notre application est regroupe en deux groupe:

5.1.1 Les classes

Est contient deux classes :

- **AESimg:** cette classe est une méthode pour crypter et décrypter l'image.
- **AEStxt:** cette classe est une méthode pour crypter et décrypter un texte.

5.1.2 Les interfaces graphiques

Notre logiciel possède une interface graphique qui facilite son utilisation, représentée par la figure suivant :

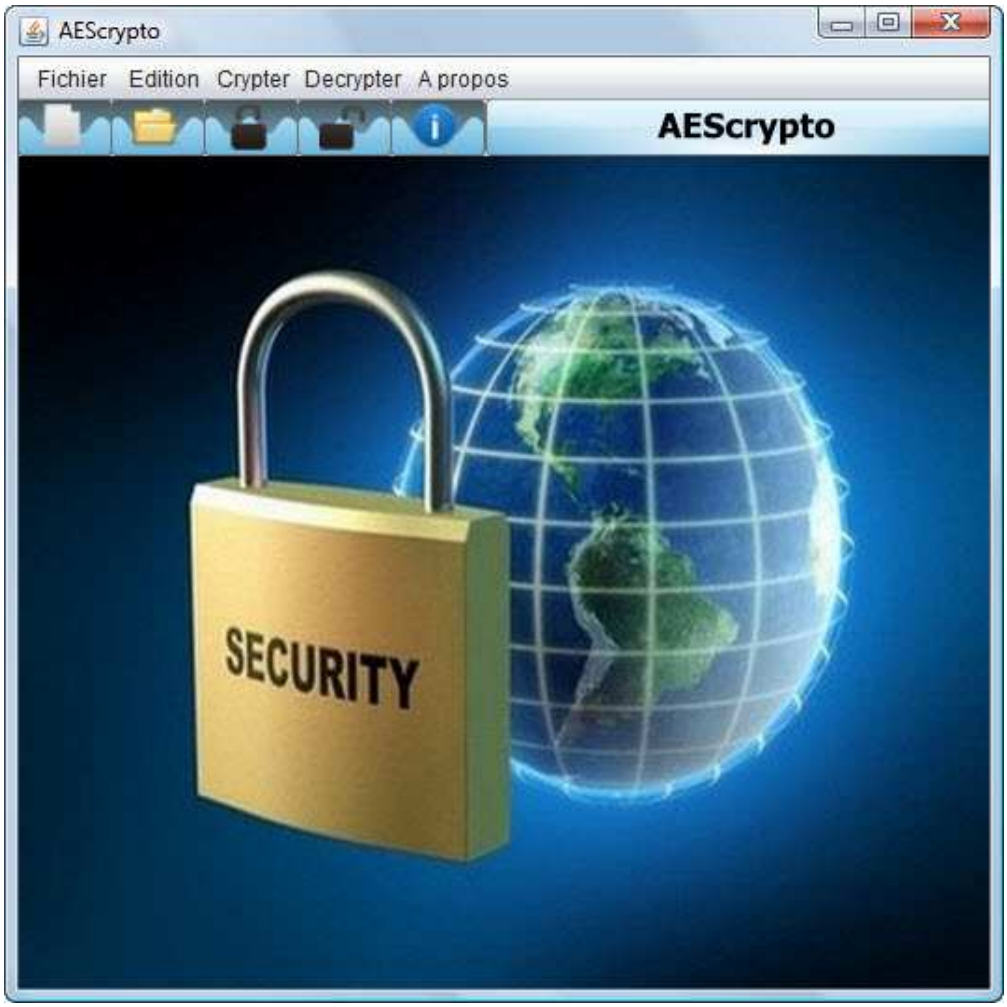


Figure III.1 Interface graphique principale

5.2 L'interface graphique de l'application

5.2.1 Barre de Menu

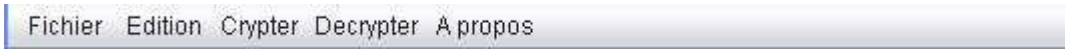


Figure III.2 Barre de Menu

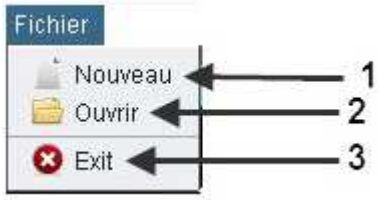
Cette bar est contient cinq menu comme suivant :

a) Menu Fichier

1- Nouveau : ouvrir nouvelle une zone de texte.

2- Ouvrir : choisir l'image pour crypter/décrypter.

3- Exit : fermer le logiciel.

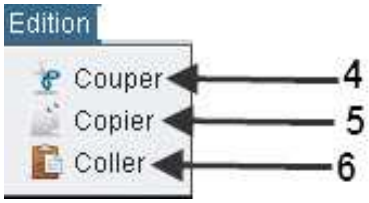


b) Menu Edition

4- Couper : couper le texte

5- Copier : copier le texte

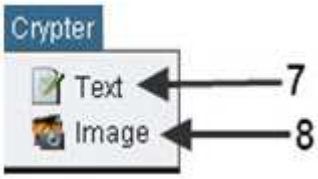
6- Coller : coller le texte



c) Menu Crypter

7- Text : cryptage un texte.

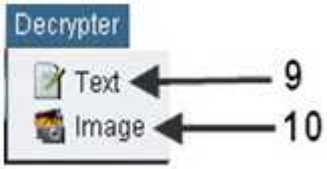
8- Image : cryptage un image.



d) Menu Decrypter

9- Text : décryptage un texte.

10- Image : décryptage un image.



e) Menu A propos

11- Info sur AEScrypto : information sur l'application.

5.2.2 Buttons raccourcis

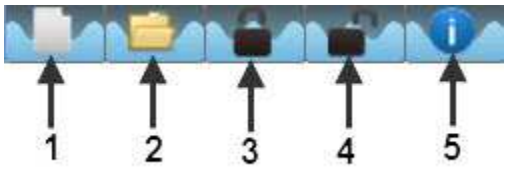


Figure III.3 Buttons raccourcis

1- Button nouveau: pour tapez un nouvelle texte dans l'éditeur de texte.

2- Button ouvrir: pour choisi l'image que nous allons crypter/décrypter.

3- Button crypter: pour entrez la clé de cryptage à le texte ou l'image.

4- Button decrypter: pour entrez la clé de décryptage à le texte ou l'image

5-Button a propos: information sur l'application

Exemple d'application sur un texte:

a) Le chiffrement :

1. Saisir le texte clair :

Figure III.4 Texte Clair

2. choisir le type de cryptage (Crypter : Text).
3. entrer la clé de cryptage (128 bits).

Figure III.5 Entrer la clé de cryptage

4. Le résultat de cryptage de texte est :

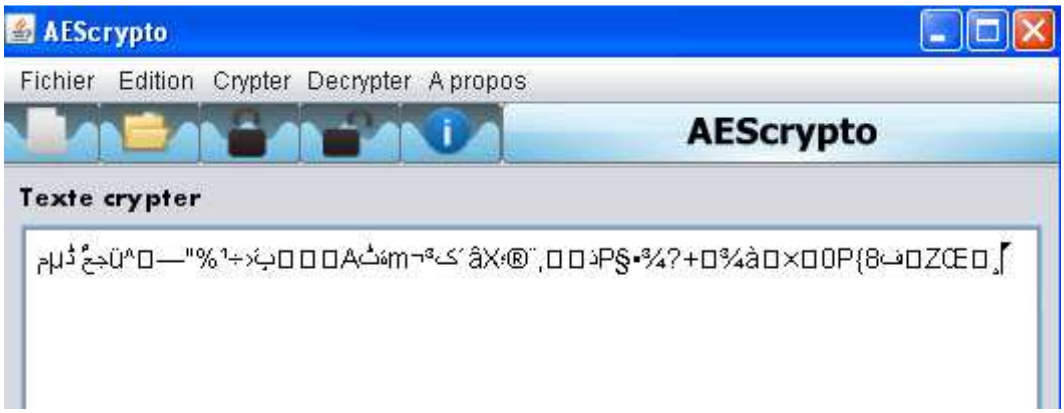


Figure III.6 Texte crypté

b) Le déchiffrement

1. Pour le décryptage nous suivons les même étapes précédent mais nous choisissons la command décrypter au lieu de crypter.

Figure III.7 Entrer la clé de décryptage

2. Le résultat de décryptage de texte est :

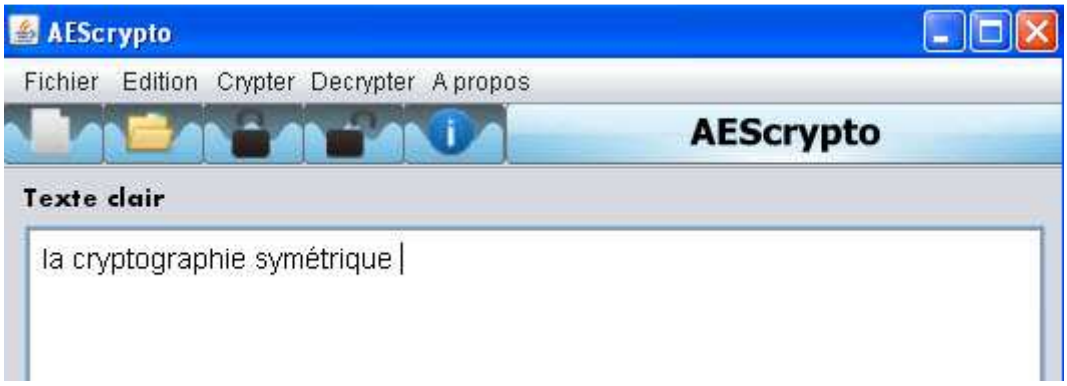


Figure III.8 Texte clair après le décryptage

Exemple d'application sur une image:

a) Le chiffrement

1. choisir l'image clair

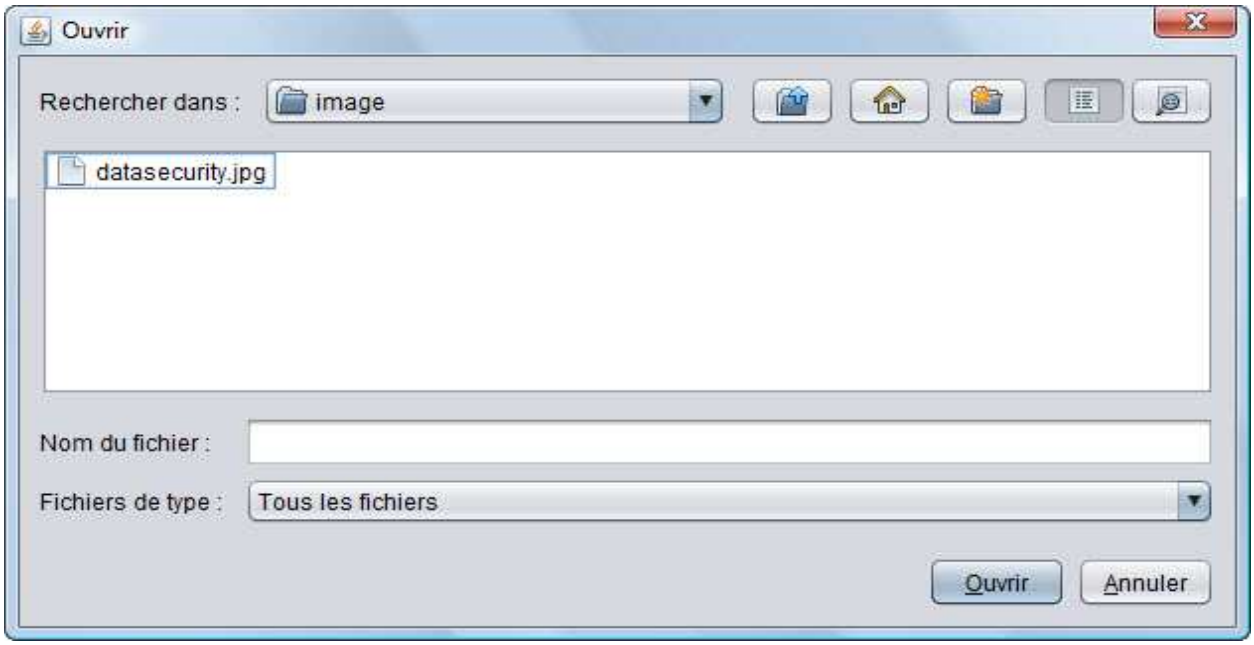


Figure III.9 Fenêtre ouvrir pour choisir l'image clair

2. Aprés le choix de l'image clair l'interface est formé comme suit

Figure III.10 Fenêtre après le choix de l'image

3. Cliquer sur sous menu crypter pour afficher le message d'entrer la clé

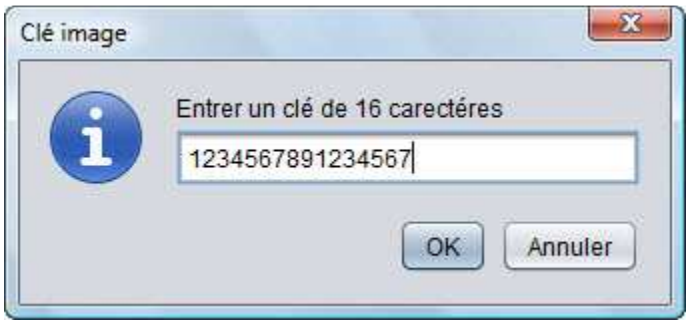


Figure III.11 Message d'entrer la clé de cryptage

4. Le résultat de cryptage de l'image est un fichier crypté

Figure III.12 Icône de fichier crypté

b) Le déchiffrement

1. Pour le décryptage nous suivons les même étapes précédent mais nous choisissons la command décrypter au lieu de crypter.

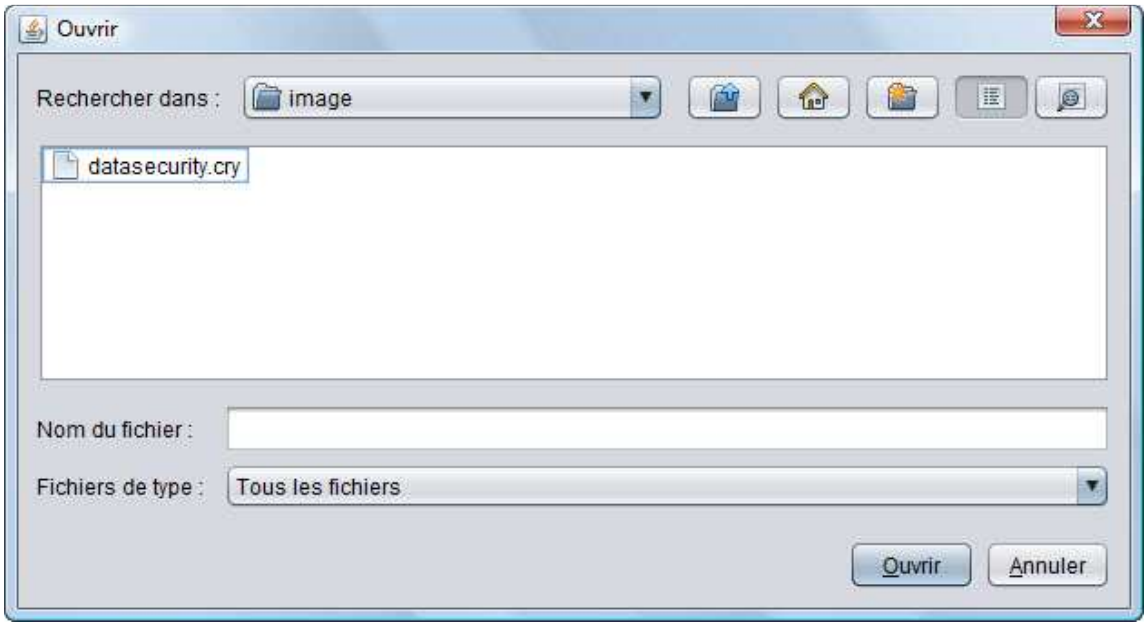


Figure III.13 Fenêtre ouvrir pour choisir l'image crypté

2. Après le choix de l'image crypté l'interface est formé comme suivant

Figure III.14 Choisir l'image crypter et entrer la clé

3. Le résultat de cryptage de l'image est une image clair

Figure III.15 L'image clair

6. Conclusion

Ce dernier chapitre était consacré pour la réalisation de l'algorithme AES. Nous avons commencé par la description du langage et l'environnement de développement choisi. Puis nous avons présenté l'interface graphique de notre application suivie par des exemples de cryptage et décryptage d'un texte et d'une image.

Conclusion générale

La cryptographie est une science très vaste, elle a un lien direct avec la sécurité des données et la communication. De nos jours il y a de plus en plus d'informations qui doivent rester secrètes ou confidentielles, par exemple les informations échanger par les banques, ou un mot de passe ne doivent pas divulgues et personne ne doit pouvoir les déduire. C'est pourquoi ce genre d'information est crypté. Le cryptage est donc, nécessaire pour que les données soient non intelligibles sauf à l'auditoire voulu..

Dans notre travail, nous avons élaboré un état de l'art sur tous les concepts en relation avec la cryptographie à partir des notions fondamentales jusqu'aux mécanismes avancées, suivi par un panorama sur les différentes classes des algorithmes cryptographiques existantes, et un aperçu global sur la cryptanalyse.

Enfin, nous avons présenté en détaille un standard de chiffrement symétrique qu'est l'algorithme AES, suivi par une description des classes d'objets utilisées pour le cryptage et décryptage des textes et des images en JAVA ,ainsi que l'interface graphique de l'application et les résultats obtenus.

Comme perspectives relatives au domaine que nous avons traité nous souhaitons étendre notre système pour :

- Le chiffrement et le déchiffrement des images animées.
- Le chiffrement et le déchiffrement des documents qui contiennent du texte et en même temps des images.

Bibliographie

[1] NGUYEN phong quang :Théorie et pratique de la cryptanalyse à clef publique,2000.

[2] Quentin Stiévenart : La cryptologie,1998.

[3] Adda ALI PACHA - Naima HADJ-SAID.La cryptographie et ses principaux systèmes de références, 2002.

[4] Bourgeois Morgan.Initiation à PGP : Gnu PG,2006.

[5] Hervé Schauer.Introduction à la cryptographie,1999-2001.

[6] www.commentcamarche.net.

[7] www.wikipedia.org.

[8] Liran Lerman.Cryptanalyse par analyse de consommation,2009-2010.

[9] Naima HADJ-SAID& Adda ALI-PACHA & Abderahmane BELGHORAF . Système Cryptographique AES: Advanced Encryption Standard,2002 .

[10] Cay S. Horstmann and Gary Cornell. Au cœur de Java™, 2008 .