

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU



FACULTE DU GENIE ELECTRIQUE ET D' INFORMATIQUE
DEPARTEMENT D' ELECTRONIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Sciences et Technologies**

Filière : **Génie électrique**

Spécialité : **Télécommunication et réseaux**

Présenté par
Koceila LOUNICI

Thème

Conception et réalisation d'un Cryptosystème hybride pour la transmission sécurisée d'images

Mémoire soutenu publiquement le 14/07/ 2016.. devant le jury composé de :

M Rezki ZIANI

Professeur, UMMTO, Président

M Mourad LAHDIR

Maitre de conférences A, UMMTO, Encadreur

M Hamid HAMICHE

Maitre de conférences A, UMMTO, Examineur

M Youcef ATTAF

Doctorat, UMMTO, Examineur

Remerciements

Au terme de ce mémoire, je tiens à remercier tout naturellement en premier lieu DIEU le tout puissant qui m'a donné la force, le courage et la patience de bien mener ce travail.

En premier lieu, je tiens à exprimer ma reconnaissance à monsieur LAHDIR Mourad mon encadreur, pour m'avoir guidé et encouragé tout au long de l'accomplissement de ce travail, et pour son entière confiance, ses conseils, critiques et sa patience.

Mes remerciements s'adressent également à monsieur le président de jury et les membres de jury d'avoir accepté d'assister à ma soutenance et de juger ce modeste travail.

Mes sentiments vont également à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce travail.

Dédicace

La page de la dédicace est généralement celle qui est écrite en dernier et lue en premier, aussi je doute que ce mémoire puisse faire exception à la règle.

Je dédie ce mémoire à toute ma famille, mon père, ma mère, mes frères, et mes amis.

Liste des figures

Figure 1.1. Principe de l'algorithme symétrique.....	6
Figure 1.2. Schéma de Feistel	10
Figure 1.3. Cryptage et décryptage synchrone	13
Figure 1.4. Cryptage et décryptage asynchrone	14
Figure 1.5. Principe de l'algorithme asymétrique.....	16
Figure 1.6. Cryptage hybride.....	19
Figure 2.1. Principe de l'algorithme symétrique	23
Figure 2.2. Structure de données de AES (a) Forme matricielle de l'AES, (b) la clef étendue	24
Figure 2.3. AES : (a) Chiffrement (b) déchiffrement.....	26
Figure 2.4. Les tours de chiffrement de AES	27
Figure 2.5. Opérations aux niveaux d'octets	28
Figure 2.6. Construction des boîtes de AES S-Box et IS-Box	31
Figure 2.7. ShiftRows Transformation.....	32
Figure 2.8. MixColumns Transformation.....	34
Figure 2.9. AddRoundKey Transformation.....	35
Figure 2.10. La clé de l'AES étendue	36
Figure 2.11. Cryptosystème à clé publique	38
Figure 2.12. Cryptosystème à clé publique (authentification et confidentialité)	39
Figure 2.13. Cryptage à plusieurs blocs	44
Figure 2.14. Principe de fonctionnement de SHA-1	46
Figure 2.15. Calcul des Wt dérivés du bloc de 512bits	47
Figure 2.16. Fonction de ronde du SHA-1	48
Figure 3.1. Le cryptosystème hybride de transmission d'images. <<Bloc ÉMETTEUR>>	52
Figure 3.2. Le cryptosystème hybride de transmission d'images. <<Bloc Récepteur >>.....	53
Figure 3.3. Mode ECB	55
Figure 3.4. Mode CBC	56

Figure 3.5. Mode CFB.....	58
Figure3.6. Mode OFB	59
Figure3.7. Image originale (a) lena ,(e) médicale et (c) logo ; et leurs histogrammes respectifs en (b),(d) et (f).....	61
Figure3.8. Image Lena chiffrée en : (a) mode ECB, (e) mode CBC et déchiffrée en : (c) mode ECB, (g) mode CBC avec leurs histogrammes respectifs.....	62
Figure3.9. Image Lena chiffrée en : (a) mode CFB, (e) mode OFB et déchiffrée en : (c) mode CFB, (g) mode OFB avec leurs histogrammes respectifs.	63
Figure3.10. Image Médicale chiffrée en : (a) mode ECB, (e) mode CBC et déchiffrée en : (c) mode ECB, (g) mode CBC avec leurs histogrammes respectifs	65
Figure3.11. Image Logo chiffrée en : (a) mode ECB, (e) mode CBC et déchiffrée en : (c) mode ECB, (g) mode CBC avec leurs histogrammes respectifs.....	66
Figure3.12. Corrélacion des pixels adjacents.....	70
Figure3.13. Analyse de la sensibilité à la clé secrète. (a) image originale; (b) image.	71
Figure 3.14. (a)Image originale de Lena. (b)Image cryptée. (c)Copie d'une région de l'image cryptée et collage sur une autre zone. (d)Décryptage de l'image après modification	73
Figure4.1 Interface de Netbeans IDE.	75
Figure 4.2. Interface de client.....	76
Figure4.3. fenêtre de saisie la clé secrète de l'algorithme AES	77
Figure4.4 Fenêtre des paramètres du Client.	78
Figure 4.5 fenêtre de transmission le tout {image chiffrée+K'' la clé chiffrée+empreinte chiffrée}.....	79
Figure 4.6. Interface de Serveur.	79
Figure4.7. Fenêtre des paramètres du Serveur.	80
Figure4.8. Fenêtre de réception de tout {image chiffrée+K'' la clé chiffrée+empreinte chiffrée}.....	81
Figure4.9. A propos : affiche la fenêtre donnant les informations générales sur le logiciel.....	81
Figure4.10 envoie de la clé secrète en text clair.....	83
Figure4.11 interception de la clé secreta par un attaquant.	83
Figure4.12 envoie de la clé secrète en text cryptée.....	84
Figure4.13 interception de la clé secreta cryptée par un attaquant.....	85

Figure4.14. Procédure de chiffrement d'une image	86
Figure4.15. Procédure de déchiffrement d'une image	87
Figure4.16. Image originale.	88
Figure4.17. Chiffrement de l'image en mode CBC	88
Figure4.18. Transmission de tout {image chiffrée+K'' la clé chiffrée+empreinte chiffrée}.....	89
Figure4.19. la Réception des informations.....	89
Figure4.20 Déchiffrement de l'image et vérification d'intégrité.	90
Figure4.21 Image originale.	90
Figure4.22 Image cryptée.....	91
Figure4.23. Transmission de tout {image chiffrée+K'' la clé chiffrée+empreinte chiffrée}.....	91
Figure4.24 la Réception des informations.....	91
Figure4.25 Déchiffrement de l'image et vérification d'intégrité.	91
Figure4.26 Image cryptée par la clé :koceilalounici91.	92
Figure4.27 image déchiffrée avec la clé :koceilalounici99.....	92
Figure4.28 Mesure du temps de cryptage en milliseconde.	93

SOMMAIRE

Introduction générale	1
 Chapitre 1 : Généralités sur la cryptographie	
1.1. Introduction	3
1.2 La sécurité de l'information et la cryptographie.....	3
1.2.1. La cryptographie	4
1.2.2. Terminologie	5
1.2.3. Buts de la cryptographie.....	6
1.3. Chiffrement symétrique.....	6
1.3.1. Cryptosystème classique	7
1.3.1.1. Chiffrement par décalage (shift cipher)	8
1.3.1.2. Alphabets désordonnés ou codage par substitution.....	8
1.3.1.3. Chiffrement affine	8
1.3.1.4. Chiffrement de vigne.....	9
1.3.1.5. Chiffrement de HILL	9
1.3.1.6. Chiffrement par permutation.....	9
1.3.2. Cryptosystème moderne	10
1.3.2.1. Le schéma de Feistel	10
1.3.2.2. Les tables de substitutions et de permutation.....	12
1.3.2.3. Mode de chiffrement	13
1.3.2.3.1. Chiffrement par flot.....	13
1.3.2.3.2. Chiffrement par bloc	15
1.4. Chiffrement asymétrique.....	15
1.5. Cryptage Symétrique vs Asymétrique.....	18
1.6. Condensat et fonction de hachage	18
1.7. Chiffrement hybride	18
1.8. Cryptanalyse.....	19
1.8. Conclusion.....	21

Chapitre 2 : Les algorithmes de cryptages : AES et RSA

2.1. Introduction	22
2.2. La structure de AES	22
2.2.1. Les fonctions de transformation de L' AES	27
2.2.1.1 « Substitute Bytes Transformation »	27
2.2.1.2 « ShiftRows Transformation »	32
2.2.1.3 « MixColumns Transformation »	32
2.2.1.4 « AddRoundKey Transformation »	34
2.2.2 La clef de l' AES étendu	35
2.3. Algorithme asymétrique RSA	37
2.3.1. Cryptosystème a clé publique	37
2.3.2. L' algorithme RSA	39
2.3.3. Principe de fonctionnement de l' algorithme RSA	39
2.4. Principe de fonctionnement de l' SHA-1	45
2.5. Conclusion	49

Chapitre 3 : Conception du cryptosystème hybride

3.1. Introduction	50
3.2. Le cryptosystème de transmission	50
3.2.1. Bloc émetteur	50
3.2.2. Bloc récepteur	51
3.3. Mode de chiffrement	54
3.3.1. Le mode « ElectronicCodeBook »	55
3.3.2. Le mode « Cipher Block Chaining(CBC) »	56
3.3.3. Le mode CipherFeedBack (CFB)	57
3.3.4. Le mode « Output FeedbackOutput Feedback » (OFB)	59
3.4. Résultats expérimentaux	60
3.5. Performances et analyse de la sécurité du cryptosystème	64

3.5.1. Analyse statistique.....	64
3.5.1.1. Analyse d’histogramme.....	64
3.5.1.2. Analyse de l’entropie	67
3.5.1.3. Corrélation entre l’image originale et l’image chiffrée.....	67
3.5.1.4. Corrélation entre les pixels adjacents.....	68
3.5.1.5. Analyse de la sensibilité à la clé secrète	71
3.5.2. Analyse différentielle	72
3.5.3. Contrôle d’intégrité	73
3.6 Conclusion.....	74

Chapitre 4 : Application et Résultats

4.1. Introduction	75
4.1. Présentation de l’application	75
4.2.1. Outils et technologie à utiliser.....	75
4.2.2. L’interface de l’application	76
4.2.2.1. Client	76
4.2.2.1.1. Les menus et les sous menus.....	76
4.2.2.2. Serveur	82
4.3. Tests et Résultats.....	82
4.3.1. Capture des trames sur le réseau	82
4.4. Temps d’exécution	93
4.5. Conclusion.....	94

Conclusion générale	95
----------------------------------	-----------

Annexe

Références bibliographies

La sécurité de l'information est une question importante dans notre société de l'information [1] [2]. Avec l'augmentation croissante de l'Internet et les systèmes de réseautage, les parties ont tendance à partager ou à envoyer des données de plus en plus sur des canaux de communication. La sécurité de ces données est d'une importance primordiale dans de nombreuses applications telles que les renseignements militaires, les informations médicales et les informations gouvernementales. Les images numériques ceux qui représentent 70% de l'information transmise sur Internet [3] sont des éléments importants du l'échange en réseau.

Interactivité, l'ouverture et le partage du réseau facilitent la copie et la modification d'une image numérique, tout en offrant la commodité pour les utilisateurs, fournissent également l'occasion pour les attaquants de voler ou d'altérer des données secrètes [4]. Par exemple, en Juin 2013, Edward Snowden dévoile les documents de PRISM aux médias [5]. Le contenu principal c'est que depuis 2007, l'Agence nationale de sécurité américaine (NSA) collectionne des documents, des logs de connexion, photos, vidéos, e-mails pour acquérir des informations personnelles.

La transmission des images soulève donc un nombre important de problèmes. Nous citons, par exemple la confidentialité, l'authentification et l'intégrité des données.

Parmi les nombreuses méthodes de protection de la littérature, le chiffrement est l'une des méthodes les plus efficaces. Plusieurs techniques de chiffrement sont proposées pour répondre à ces exigences, Nous trouvons parmi elles, l'algorithme symétrique AES (Advanced Encryption Standard)[6] [7] [8] [9] et l'algorithme asymétrique RSA (Rivest, Shamir and Adleman)[7] [10] qui ont prouvés de nos jours leurs robustesses contre les différents types d'attaques. Mais si nous appliquons ces algorithmes séparément pour la transmission des images, ils ne peuvent assurer que la confidentialité, pour cette raison nous proposons un cryptosystème de chiffrement hybride basé sur les deux algorithmes AES, RSA et la fonction de hachage SHA-1 pour assurer les trois grands axes de sécurité : la confidentialité, l'authenticité et l'intégrité.

Pour assurer la confidentialité dans notre cryptosystème notre choix est tombé sur l'algorithme AES, car il consomme peu de mémoire, n'étant pas basé sur un schéma de Feistel, Sa complexité est moindre et est plus facile à implémenter et en plus de sa rapidité AES n'a

pas était cassé jusqu'à nos jours, l'RSA pour assurer l'authentification et les échanges sécurisés des clés et la fonction hachage SHA-1 [11] pour vérifier l'intégrité.

Ce travail a pour objectif la Conception et la réalisation d'un cryptosystème hybride pour la transmission sécurisée d'image. Une application a été réalisé pour l'implémentation notre cryptosystème en utilisant une architecture client/serveur.

Ce mémoire s'articule autour de quatre chapitres :

Le premier chapitre représente l'état de l'art de la cryptographie : nous aborderons plusieurs types de chiffrements et les différents types d'attaques.

Le deuxième chapitre se compose de trois parties : La première traite en détail l'Algorithme symétrique AES, la deuxième est consacrée pour l'étude détaillée de l'algorithme asymétrique RSA et la troisième traite la fonction de hachage SHA-1.

Dans Le troisième chapitre nous allons expliquer en détail les quatre modes de fonctionnement utilisés pour la transmission des images et détaillerons le cryptosystème proposé et sa robustesse.

Le dernier chapitre décrit l'application réalisée et les résultats fournis de cette dernière.

Nous finirons notre travail par une conclusion et un ensemble des perspectives ouvertes par ce thème.

1.1. Introduction :

La cryptographie est une science très ancienne qui date de 1900 ans avant J.-C. Des recherches indiquent qu'un scribe égyptien a employé des hiéroglyphes non conformes à la langue pour écrire un message. De ce temps-là et au long de l'histoire, la cryptographie a été utilisée exclusivement à des fins militaires. Aujourd'hui, les réseaux informatiques exigent une phase de cryptographie comme mécanisme fondamental afin d'assurer la confidentialité de l'information numérique. En se rapportant à des événements historiques de notre ère informatisée, nous pouvons retenir les suivants : la machine Énigma créée par Dr Arthur Scherbius en 1923 et qui a été utilisée largement dans la seconde guerre mondiale ; la théorie de l'information de Claude Shannon en 1949; l'algorithme lucifer développé par IBM en 1970; la théorie du système à clé publique par Whitfield Diffie et Martin Hellman en 1976 qui a marqué le début de la cryptographie moderne ; le premier standard pour le cryptage, l'algorithme DES en 1976; les résultats de la cryptographie quantique par Charles H. Bennett et Gilles Brassard en 1990, et le standard AES en 2000, qui a remplacé le DES.

1.2. La sécurité de l'information et la cryptographie :

Pour introduire la cryptographie, la compréhension des enjeux liés à la sécurité de l'information en général est nécessaire. La sécurité de l'information se manifeste de plusieurs façons selon la situation et l'exigence. Indépendamment de qui est en cause, à un degré ou un autre, toutes les parties d'une transaction doivent avoir confiance que certains objectifs associés à la sécurité de l'information ont été respectés.

Au cours des siècles, un ensemble de protocoles et de mécanismes a été créé pour faire face aux problèmes de sécurité de l'information lorsqu'elle est transmise dans des documents physiques. Souvent, les objectifs de sécurité de l'information ne peuvent pas uniquement être atteints seulement par des algorithmes mathématiques et des protocoles, mais nécessitent des techniques procédurales et respect des lois pour obtenir le résultat désiré. Par exemple, la confidentialité des lettres est assurée par des enveloppes scellées livrées par un service de courrier accepté. La sécurité physique des enveloppes pour nécessité pratique, des lois sont qui en font une infraction criminelle d'ouvrir le courrier pour lequel on n'est pas autorisé.

Parfois la sécurité n'est pas obtenue par l'information elle-même, mais à travers le document physique enregistré. Par exemple, la monnaie de papier nécessite des encres et des matériaux spéciaux pour prévenir la contrefaçon.

Conceptuellement, la façon dont l'information est enregistrée n'a pas changé considérablement au fil du temps. Considérant que l'information a été généralement stockées et transmises sur le papier, une grande partie réside maintenant sur supports magnétiques et est transmis par l'intermédiaire de différents systèmes de télécommunications, certains sans fil. Ce qui a changé de façon spectaculaire la possibilité de copier et de modifier des informations. On peut faire des milliers de copies identiques d'un élément d'information stockées électroniquement et chacun est identique à l'original. Avec des informations sur le papier, cela est beaucoup plus difficile. Ce qui est nécessaire donc pour une société où l'information est le plus souvent stockées et transmises sous forme électronique est un moyen d'assurer la sécurité de l'information qui sois indépendant de l'enregistrement sur support physique ou du transport et de telle sorte que les objectifs de sécurité de l'information reposent uniquement sur l'information numérique elle-même.

Atteindre la sécurité de l'information dans une société électronique nécessite un vaste éventail de compétences techniques et juridiques. Il n'y a cependant aucune garantie que tous les objectifs de sécurité de l'information estimés nécessaires peuvent être satisfaits de manière adéquate. Le moyen technique est assuré par la cryptographie.

1.2.1. La cryptographie :

Le mot cryptographie est un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Le verbe crypter est parfois utilisé mais on lui préférera le verbe *chiffrer*.

Le fait de coder un message de telle façon à le rendre secret s'appelle *chiffrement*. La méthode inverse, consistant à retrouver le message original, est appelée *déchiffrement*

Le chiffrement se fait généralement à l'aide d'une *clé de chiffrement*, le déchiffrement nécessite quant à lui une *clé de déchiffrement*. On distingue généralement deux types de clés:

- **Les clés symétriques**: il s'agit de clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.
- **Les clés asymétriques**: il s'agit de clés utilisées dans le cas du chiffrement asymétrique (aussi appelé *chiffrement à clé publique*). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

1.2.2. Terminologie:

- **Texte en clair** : c'est le message à protéger.
- **Texte chiffré** : c'est le résultat du chiffrement du texte en clair.
- **Chiffrement** : c'est la méthode ou l'algorithme utilisé pour transformer un texte en clair en texte chiffré.
- **Déchiffrement** : c'est la méthode ou l'algorithme utilisé pour transformer un texte chiffré en texte en clair.
- **Clé** : c'est le secret partagé utilisé pour chiffrer le texte en clair en texte chiffré et pour déchiffrer le texte chiffré en texte en clair. On peut parfaitement concevoir un algorithme qui n'utilise pas de clé, dans ce cas c'est l'algorithme lui-même qui constitue la clé, et son principe ne doit donc en aucun cas être dévoilé.
- **Cryptographie** : cette branche regroupe l'ensemble des méthodes qui permettent de chiffrer et de déchiffrer un texte en clair afin de le rendre incompréhensible pour quiconque n'est pas en possession de la clé à utiliser pour le déchiffrer.
- **Cryptanalyse** : c'est l'art de révéler les textes en clair qui ont fait l'objet d'un chiffrement sans connaître la clé utilisée pour chiffrer le texte en clair.
- **Cryptologie** : il s'agit de la science qui étudie les communications secrètes. Elle est composée de deux domaines d'étude complémentaires : la cryptographie et la cryptanalyse.
- **Décrypter** : c'est l'action de retrouver le texte en clair correspondant à un texte chiffré sans posséder la clé qui a servi au chiffrement. Ce mot ne devrait donc être employé que dans le contexte de la cryptanalyse.

- **Crypter** : en relisant la définition du mot décrypter, on peut se rendre compte que le mot crypter n'a pas de sens et que son usage devrait être oublié. Le mot cryptage n'a pas plus de sens non plus.
- **Coder, décoder** : c'est une méthode ou un algorithme permettant de modifier la mise en forme d'un message sans introduire d'élément secret.

1.2.3. Buts de la cryptographie :

La cryptographie permet de résoudre quatre problèmes différents :

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

1.3. Chiffrement symétrique :

Le **chiffrement symétrique** (aussi appelé *chiffrement à clé privée* ou *chiffrement à clé secrète*) consiste à utiliser la même clé pour le chiffrement et le déchiffrement.

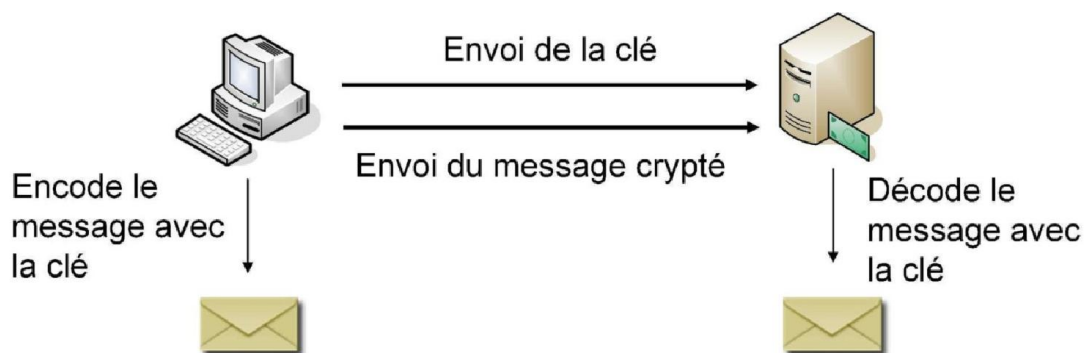


Figure 1.1. Principe de l'algorithme symétrique

Le chiffrement consiste à appliquer une opération (algorithme) sur les données à chiffrer à l'aide de la clé privée, afin de les rendre inintelligibles. Ainsi, le moindre algorithme (tel qu'un OU exclusif) peut rendre le système quasiment inviolable (la sécurité absolue n'existant pas).

Toutefois, dans les années 40, *Claude Shannon* démontra que pour être totalement sûr, les systèmes à clés privées doivent utiliser des clés d'une longueur au moins égale à celle du message à chiffrer. De plus le chiffrement symétrique impose d'avoir un canal sécurisé pour l'échange de la clé, ce qui dégrade sérieusement l'intérêt d'un tel système de chiffrement.

Le principal inconvénient d'un cryptosystème à clés secrètes provient de l'échange des clés. En effet, le chiffrement symétrique repose sur l'échange d'un secret (les clés). Ainsi, se pose le problème de la distribution des clés :

D'autre part, un utilisateur souhaitant communiquer avec plusieurs personnes en assurant de niveaux de confidentialité distincts doit utiliser autant de clés privées qu'il a d'interlocuteurs. Pour un groupe de N personnes utilisant un cryptosystème à clés secrètes, il est nécessaire de distribuer un nombre de clés égal à $N * (N-1) / 2$.

1.3.1. Cryptosystème classique [12] :

Les cryptosystèmes classiques ont été conçus avant la création des ordinateurs et qui ont donné les concepts et les bases pour l'évolution de plusieurs algorithmes symétriques encore utilisés de nos jours. Les cryptosystèmes classiques sont groupés en chiffrement mono alphabétique et poly alphabétique. Le chiffrement mono alphabétique est très primaire, il s'agit d'une substitution simple. Chaque lettre est remplacée par une autre lettre ou symbole conformément à un certain algorithme.

1.3.1.1. Chiffrement par décalage (shift cipher) :

Le chiffrement est effectué en remplaçant chaque lettre par la lettre située un certain nombre de places plus loin dans l'alphabet.

Donc par exemple si la clé était trois, le texte clair A serait remplacé par le texte chiffré D, la lettre B serait remplacé par E et ainsi de suite. Le mot en texte clair HELLO serait chiffré comme le texte chiffré KHOOR. Lorsque ce chiffre est utilisé avec la clé trois, il est souvent

appelé le chiffrement de César,

Les fonctions de Cryptage :

$$e_k(x) = x + K \bmod 26 \quad \text{I.1}$$

Et décryptage :

$$d_k(y) = y - K \bmod 26 \quad \text{I.2}$$

Sont définies en Z_{26} et K est la clé de décalage.

1.3.1.2. Alphabets désordonnés ou codage par substitution :

Ce genre de cryptosystème consiste à remplacer une lettre par une autre en utilisant un alphabet désordonné où K est l'ensemble des permutations aléatoires possibles des 26 lettres. Le chiffrement par décalage est un cas particulier du chiffrement par alphabets désordonnés, où il existe seulement 26 clés.

1.3.1.3. Chiffrement affine :

Il s'agit d'un algorithme qui utilise les fonctions affines pour le cryptage :

$$e_k(x) = ax + b \bmod 26 \quad \text{I.3}$$

et pour le décryptage :

$$d_k(y) = a^{-1}(y - b) \bmod 26 \quad \text{I.4}$$

La constante a est premier avec 26, et a^{-1} désigne l'inverse de a modulo 26. Nous remarquons également que si $a = 1$, nous avons le chiffrement par décalage.

1.3.1.4. Chiffrement de vigenère :

Le chiffrement de Vigenère utilise une clé qui définit le décalage pour chaque lettre du texte clair. Sa force réside dans l'utilisation non pas de 1, mais de 26 alphabets décalés $(Z_{26})^m$ où m est un nombre entier positif.

La fonction de cryptage est :

$$e_k(x_1, \dots, x_m) = (x_1 + k_1, \dots, x_m + k_m) \quad I.5$$

et la fonction de décryptage est :

$d_k(y_1, \dots, y_m) = (y_1 - k_1, \dots, y_m - k_m)$ sont faits avec les clés $K = (k_1, \dots, k_m)$.

Le chiffrement de Vigenère est considéré comme un chiffrement par flot, avec une période m .

1.3.1.5. Chiffrement de HILL :

Le chiffrement de Lester S. Hill est considéré polygamique . Dans un chiffrement polygamique, un groupe de n lettres est chiffré par un groupe de n symboles.

Le principe est de remplacer les lettres par leur rang dans l'alphabet, soit K une matrice inversible, alors :

$$e_k(x) = xK \text{ et } d_k(y) = yK^{-1} \quad I.6$$

En fait, l'idée est de prendre m combinaisons linéaires de m caractères de l'alphabet.

1.3.1.6. Chiffrement par permutation :

Le principe du chiffrement par permutation est de garder le même alphabet et de changer seulement l'ordre des lettres. Soit π toutes les permutations $\{1, \dots, m\}$

Pour une clé et m un entier positif, alors les fonctions de cryptage et décryptage sont décrites par :

$$e_\pi(x_1, \dots, x_m) = (x_{\pi(1)}, \dots, x_{\pi(m)}) \quad I.7$$

$$d_\pi(y_1, \dots, y_m) = (y_{\pi^{-1}(1)}, \dots, y_{\pi^{-1}(m)}) \quad I.8$$

Le chiffrement par permutation est un cas spécial du chiffrement de Hill pour une matrice de permutation $K_\pi = (k_{i,j})$.

1.3.2. Cryptosystème moderne :

Le chiffrement moderne utilise la puissance des ordinateurs modernes. Comme les données traitées par les ordinateurs sont uniquement sous forme numériques (bits), les procédés de substitutions et de transpositions sont toujours utilisées mais maintenant seulement sur deux éléments primaires (0 et 1). On constate donc que les idées et principes vues précédemment reste d'actualité, c'est pourquoi il était à notre avis intéressant de les mettre en valeur tout de suite, ce qui est généralement jamais fait à notre regret.

1.3.2.1. Le schéma de Feistel :

Le problème est que, si l'on sait depuis longtemps construire des fonctions qui ont l'air aléatoire, on ne savait pas avant les travaux de Feistel construire des bijections aléatoires. La solution apportée par Horst Feistel, ingénieur chez IBM qui dirigea l'équipe qui conçut l'algorithme de chiffrement DES, est très élégante : on suppose par exemple qu'on a une fonction f "presque aléatoire" qui prend comme argument un mot de n bits, et renvoie un mot de n bits (qui donne l'impression d'avoir été choisi au hasard). f n'est pas supposée bijective, c'est-à-dire que deux mots différents peuvent avoir la même image par f (ceci signifie encore que, si on dispose d'un mot de n bits chiffré par f , on ne peut pas retrouver le mot initial).

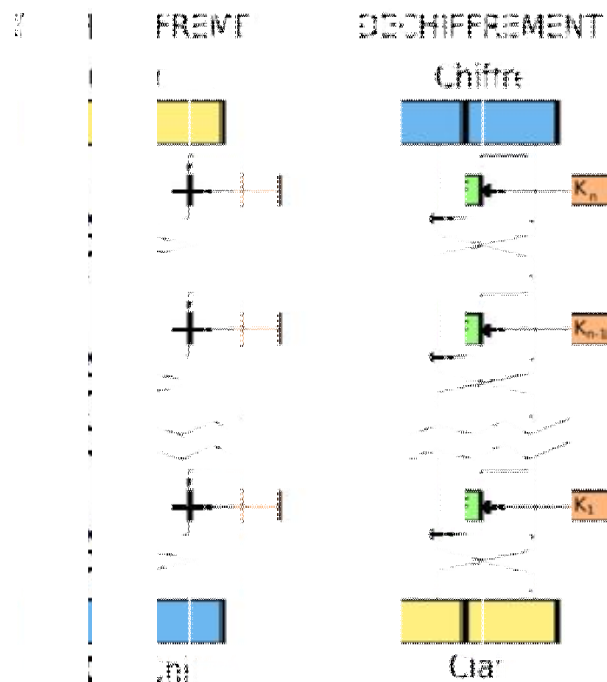


Figure 1.2. Schéma de Feistel

Une définition formelle d'un réseau de Feistel peut être donnée sous plusieurs formes. Nous reprenons ici la notation utilisée par Knudsen dans *Partial and Higher Order Differentials and Applications to DES*.

$$\begin{aligned} C_0^L &= P^L \text{ et } C_0^R = P^R \\ C_i &= (C_i^L, C_i^R) = (C_{i-1}^R, f(C_{i-1}^R, K_i) + C_{i-1}^L) \\ C^L &= C_r^L \text{ et } C^R = C_r^R \end{aligned} \quad \text{I.9}$$

- l'exposant représente la sous-partie du bloc considérée (L à gauche, R à droite).
- P correspond au bloc de texte clair, P_i^L correspond au bloc de gauche à l'entrée du tour i
- C correspond au bloc de texte chiffré, C_i^R correspond au bloc de droite à la sortie du tour i
- K_i est la clé du tour i , elle est calculée grâce à un key schedule de la clé principale
- r est le nombre de tours dans l'algorithme
- $+$ est une opération dans un groupe commutatif (souvent un XOR ou une addition)

➤ **Composition des Tours :** Chaque tour applique plusieurs transformations sur les données provenant du tour précédent :

- permutation des bits via des P-Boxes
- fonction non-linéaire (substitution) avec des S-Boxes
- mixage linéaire en utilisant la fonction XOR
- application de la clé du tour (intégrée dans une fonction ou via un XOR)

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

La plupart des algorithmes à clé secrète de la fin du XX^e s. était des schémas de Feistel. L'avènement de l'AES, au début du XXI^e siècle, qui n'en est plus un, marque la fin de la prédominance de tels algorithmes.

1.3.2.2 Les tables de substitutions et de permutation :

S-Box (*substitution box*), terme anglais désignant une table de substitution utilisée dans un algorithme de chiffrement symétrique. Une S-Box contribue à la « confusion » (terme employé par Claude Shannon) en rendant l'information originale inintelligible. Les S-Boxes permettent de casser la linéarité de la structure de chiffrement et leur nombre varie selon les algorithmes. DES compte par exemple huit tables de 16x4 éléments.

Une table de substitution prend en général une variable de m bits en entrée et produit une sortie de n bits, les entrées et les sorties n'ont pas forcément la même taille. Les tables sont souvent définies à l'avance mais il arrive parfois qu'elles soient générées par l'algorithme (par exemple dans Blowfish). Les valeurs présentes dans les S-Box doivent être choisies de manière à éviter les attaques, par divers moyens comme l'utilisation de fonctions courbes. Dans le cas de DES, il a été prouvé que les tables avaient été conçues de manière à résister à la cryptanalyse différentielle (technique qui ne sera publiée que bien des années plus tard).

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

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

1.3.2.3 Mode de chiffrement :

1.3.2.3.1 Chiffrement par flot:

Le chiffrement par flot (stream cipher) crypte séparément un caractère individuel (une suite de bits), en utilisant une transformation qui varie au fur et à mesure du temps. L'idée principale du chiffrement par flot est de produire un flux de clés (key stream) $z=z_1, \dots, z_n$ et d'utiliser z , conjointement avec le texte clair $x=x_1, x_2, \dots, x_n$ pour générer le texte chiffré $y=y_1, y_2, \dots, y_n$.

Le chiffrement par flot est mieux implanté en hardware parce qu'il demande des circuits peu

complexes et avec de petites zones mémoire (buffer). Il est ainsi conseillé pour les environnements bruités avec des ressources limitées. Ceci est le cas de la téléphonie GSM (Global System for Mobile Communications) qui se sert du chiffrement par flot A5/1/2.

➤ Chiffrement de Vernam (One-time pad)

Malgré sa simplicité, le chiffrement de Vernam est la seule méthode sûre de façon inconditionnelle. Celui-ci est impossible à décrypter même avec une puissance de calcul infinie. La fonction de cryptage de Vernam est $y_i = x_i \oplus z_i$, où $\oplus$ est un « ou exclusif » (XOR), et de décryptage $x_i = y_i \oplus z_i$. Le chiffrement de Vernam est appelé one-time pad, quand le flux de clés z est généré aléatoirement et indépendamment.

➤ Chiffrement synchrone

Un chiffrement par flot est considéré synchrone si le flux de clés z est produit indépendamment du texte clair x et du texte chiffré y (figure 1.3). Le chiffrement synchrone peut être décrit par les équations suivantes :

$$\sigma_{i+1} = f(\sigma_i, K), \quad z_i = g(\sigma_i, K), \quad y_i = h(z_i, x_i) \quad \text{I.10}$$

Avec σ_0 le vecteur d'initialisation, f la fonction d'état, g une fonction GNPA pour produire le key stream z , et h la fonction de sortie qui produit le texte chiffré y .

Les éléments du procédé de déchiffrement sont identiques, il suffit de remplacer $y_i = h(z_i, x_i)$ par $x_i = h^{-1}(z_i, y_i)$.

Un exemple classique de chiffrement synchrone, les modes OFB et CTR des chiffrements par bloc. Le chiffrement synchrone est appelé chiffrement par flot additif quand la fonction h utilise un « ou exclusif » pour le chiffrement $h = (z_i \oplus x_i)$.

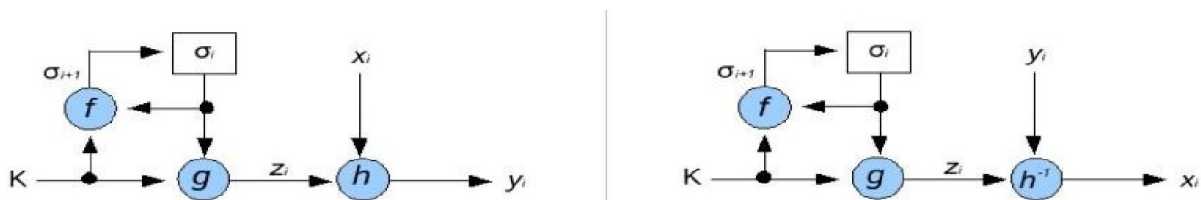


Figure 1.3. Cryptage et déchiffrement synchrone

Dans un chiffrement synchrone, l'émetteur et le récepteur doivent être synchronisés. Si un bit est perdu ou ajouté dans le texte chiffré pendant la transmission, alors il ne peut pas être décrypté. Ce fait demande une resynchronisation qui peut être acquise par une réinitialisation

ou par insertion de marques synchronisantes dans le texte chiffré.

La non-propagation d'erreur: si y_i est modifié lors de la transmission, il ne perturbera pas le déchiffrement des textes chiffrés suivants :

Attaques actives: la perte de synchronisation (première propriété) sera immédiatement détectée par le récepteur. Par contre, la deuxième propriété peut passer inaperçue auprès du récepteur. Il est donc important d'introduire des mécanismes garantissant l'authenticité et l'intégrité des données.

➤ Chiffrement asynchrone

La figure 1.4 illustre un système de chiffrement asynchrone (ou auto-synchronisant).

Les équations I.12 montrent que z est produit à partir de la clé K et d'un nombre fixe de caractères du flot chiffré y . Soit les équations suivantes :

$$\sigma_i = (y_{i-t}, y_{i-t+1}, \dots, y_{i-1}), \quad z_i = g(\sigma_i, K), \quad y_i = h(z_i, x_i) \quad \text{I.11}$$

avec $\sigma_0 = (y_{-t}, y_{-t+1}, \dots, y_{-1})$ l'état initial, K la clé, g la fonction qui produit le flot de clés, et h la fonction de sortie.

Actuellement, le chiffrement auto-synchronisant le plus employé est celui basé sur le chiffrement par bloc en mode CFB en utilisant un bit à la fois.

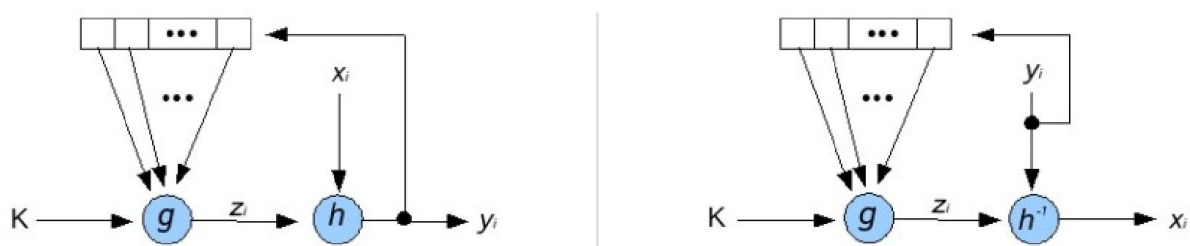


Figure 1.4. *Cryptage et décryptage asynchrone*

Auto-synchronisation: si des bits sont perdus ou ajoutés dans y_i , le procédé se resynchronise au bout de t textes chiffrés. En effet, le déchiffrement dépend uniquement des t précédents y .

Propagation d'erreur: si y_i est modifié, le déchiffrement des t suivants y sont corrompus.

Diffusion des propriétés statistiques: Du fait que le x_i influe la suite des textes chiffrés suivants, les propriétés statistiques sont dispersées. Cela rend plus difficile les attaques basées sur une analyse statistique utilisant les redondances du texte clair.

Attaques actives: la modification d'un chiffre se répercute sur les t chiffres suivants, elle a moins de chances de passer inaperçue auprès du récepteur. Par contre, l'ajout ou la suppression de chiffres seront moins détectés que pour les procédés synchrones.

1.3.2.3.2. Chiffrement par bloc:

Contrairement au chiffrement par flot qui utilise des transformations qui varient, le chiffrement par bloc utilise des transformations fixes sur tous les blocs de taille également fixe. En fait, le chiffrement par bloc est un cas spécial de chiffrement par flot où $z_i = K \forall i \geq 1$.

Nous revenons aux théories de Shannon pour citer les propriétés de confusion et diffusion qui sont très liées au chiffrement par bloc. La confusion est l'acte de rendre la corrélation entre la clé de chiffrement et le texte chiffré la plus complexe possible. La substitution ou le remplacement d'un symbole par un autre en utilisant une boîte-S (S-BOX) est un mécanisme pour accomplir la confusion. La diffusion est l'acte de dissiper la redondance des statistiques du texte clair dans les statistiques du texte chiffré. Elle est associée à la dépendance des bits d'entrée et de sortie. L'effet avalanche, la transposition et le réarrangement de l'ordre des symboles sont des techniques de diffusion.

La plupart des chiffrements symétriques par bloc (DES, 3DES, TEA (Tiny Encryption Algorithm) et IDEA (International Data Encryption Algorithm) sont construits sur l'approche du réseau de Feistel.

Les algorithmes de chiffrement par blocs peuvent être utilisés suivant différents modes, dont les deux principaux sont le mode ECB (Electronic Code Book) et le mode CBC (Cipher Block Chaining).

1.4. Chiffrement asymétrique :

Le chiffrement à clé publique, ou chiffrement asymétrique, a été proposé par Diffie et Hellman, en 1976. Dans un tel schéma, la clé de chiffrement est différente de celle de déchiffrement. N'importe qui peut utiliser la clé de chiffrement, ou clé publique, pour chiffrer

un message, mais seul celui qui possède la clé de déchiffrement, ou clé privée, peut déchiffrer le message chiffré résultant.

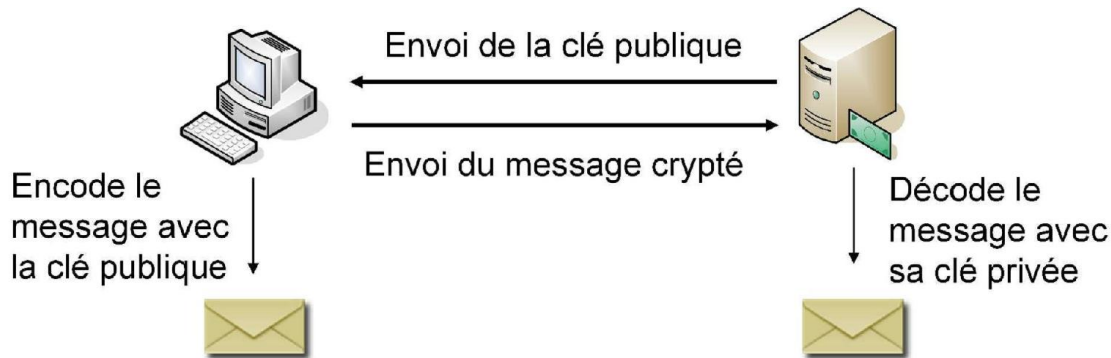


Figure 1.5. Principe de l'algorithme asymétrique

De plus, la clé de déchiffrement K^d ne peut pas être calculée (du moins dans un temps raisonnable) à partir de la clé de chiffrement K^e . Ce type de schéma repose directement sur l'existence de fonctions à sens unique. Une fonction est dite à sens unique quand il est facile de calculer K^e en connaissant K^d mais très difficile de calculer K^d connaissant K^e . Parfois, des fonctions à sens unique qui possèdent en plus une trappe sont utilisées. Une fonction à sens unique est dite à trappes quand il est très difficile de calculer K^d à partir de K^e , sauf si on connaît une information supplémentaire.

Lorsque Alice l'émetteur et Bob le destinataire, veulent communiquer de façon sécurisée, Bob choisit une paire de clés de chiffrement et de déchiffrement (K^e, K^d) . Il envoie la clé publique K^e à Alice, par l'intermédiaire d'un canal qui n'est pas forcément sécurisé. Alice transforme le message M en texte chiffré $C = E(K^e, M)$, où E représente une fonction de chiffrement, et envoie ce texte chiffré C à Bob. De son côté, Bob reçoit le texte chiffré C et calcule $M = D(K^d, C)$ où D est une fonction de déchiffrement et K^d est la clé privée connue uniquement de Bob. Ainsi, Bob récupère le message initial M .

Un exemple de chiffrement à clé publique est le schéma RSA, proposé par Rivest, Shamir et Adleman, en 1978. Ce schéma est encore très largement utilisé (sites web commerciaux, par exemple). Il repose sur la difficulté de factoriser des grands nombres et s'appuie donc sur la

théorie des nombres.

La génération des clés publique et privée peut être résumée par les étapes suivantes :

- Bob choisit deux grands nombres premiers (de longueur 1024 ou 2048 bits, en général) p et q et calcule $n = pq$.
- Bob choisit aussi, de façon aléatoire, un entier $K^e < n$ qui est premier avec $(p - 1)(q - 1)$.
- Il calcule K^d tel que $K^e K^d = 1 \bmod (p - 1)(q - 1)$.
 - Bob publie la clé publique, formée de (K^e, n) .
 - Pour envoyer un message à Bob, Alice calcule $C = M^{K^e}$ et envoie C à

Bob, et pour déchiffrer C , Bob calcule $M = C^{K^d} \bmod n$.

Dans ce schéma, la fonction de chiffrement est une fonction à sens unique et possède une trappe. En effet, toute personne connaissant la clé publique (K^e, n) et la factorisation de n peut calculer K^d .

1.5. Cryptage Symétrique vs Asymétrique [13] :

Cryptage symétrique	Cryptage asymétrique
-Chiffrement à clé privé (utilisation une clé pour crypter qui fonctionne aussi pour décrypter)	-Chiffrement à clé publique (utilisation deux clés une pour crypter clé publique et autre pour décrypter clé privé)
-Très facile	-Difficile par rapport au cryptage symétrique
-Très rapide	-Plus lent
-les clés de chiffrement symétrique doivent être conservées en toute sécurité - vous devez vous assurer que chaque personne qui a besoin de la clé, il obtient sans aucun risque de le sortir.	-les clés publiques qu'ils utilisent sont sans danger pour être publiées n'importe où parce que pour obtenir la clé privée à partir d'une clé publique peut prendre des centaines d'années de travail.

Tableau1.1 *Cryptage Symétrique vs Asymétrique*

1.6. Condensat et fonction de hachage :

Une fonction de hachage (parfois appelée *fonction de condensation*) est une fonction permettant d'obtenir un condensé (appelé aussi *condensat* ou *haché* ou en anglais *message digest*) d'un texte, c'est-à-dire une suite de caractères assez courte représentant le texte qu'il condense. La fonction de hachage doit être telle qu'elle associe un et un seul haché à un texte en clair (cela signifie que la moindre modification du document entraîne la modification de son haché). D'autre part, il doit s'agir d'une fonction à sens unique (*one-way function*) afin qu'il soit impossible de retrouver le message original à partir du condensé. S'il existe un moyen de retrouver le message en clair à partir du haché, la fonction de hachage est dite « à brèche secrète ».

Ainsi, le haché représente en quelque sorte l'*empreinte digitale* (en anglais *finger print*) du document.

Les algorithmes de hachage les plus utilisés actuellement sont :

- **MD5** (*MD* signifiant *Message Digest*). Développé par Rivest en 1991, MD5 crée une empreinte digitale de 128 bits à partir d'un texte de taille arbitraire en le traitant par blocs de 512 bits. Il est courant de voir des documents en téléchargement sur Internet accompagnés d'un fichier MD5, il s'agit du condensé du document permettant de vérifier l'intégrité de ce dernier)
- **SHA** (pour *Secure Hash Algorithm*, pouvant être traduit par *Algorithme de hachage sécurisé*) crée des empreintes d'une longueur de 160 bits
SHA-1 est une version améliorée de SHA datant de 1994 et produisant une empreinte de 160 bits à partir d'un message d'une longueur maximale de 2^{64} bits en le traitant par blocs de 512 bits.

1.7. Chiffrement hybride :

Le concept de chiffrement hybride fait appel aux deux techniques, symétrique et asymétrique [14] , comme présenté sur la figure 1.6, Il a été mis en œuvre par Zimmermann pour le PGP (Pretty Good Privacy) en 1991.

L'idée d'un système hybride est d'utiliser la rapidité de l'algorithme symétrique et la sécurité

de l'asymétrique. Une clé secrète K de 128bits est générée automatiquement pour la session. Le message m est chiffré avec cette clé K en utilisant un chiffreur symétrique, $m' = e_K(m)$. La clé K est alors chiffrée avec un chiffreur asymétrique en utilisant la clé publique du destinataire B , $K' = e_{K_B^{Pub}}(K)$. Ensuite, le message entier $M = m' + K'$ (message chiffré Symétriquement et clé asymétriquement) est envoyé au destinataire. De l'autre côté, B utilise sa clé privée k_B^{Pri} pour décrypter la clé K' et ensuite déchiffrer le message. Un exemple d'un système hybride est le protocole SSL (Secure Socket Layer) développé par les sociétés Netscape et RSA Security, cette dernière est responsable de l'algorithme RSA.

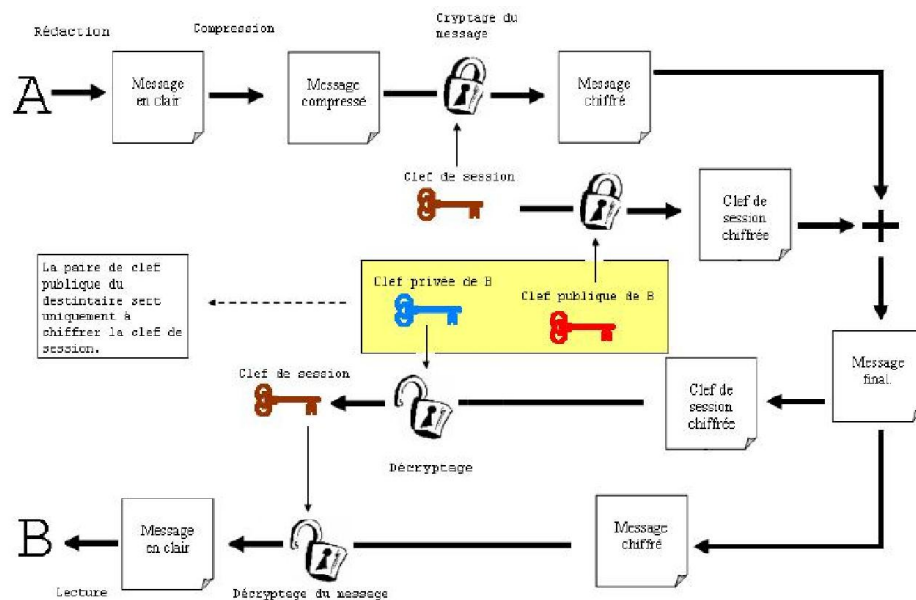


Figure 1.6. Chiffrement hybride

1.8. Cryptanalyse [12] :

La cryptanalyse ou l'attaque regroupe tous les moyens de déchiffrer un texte crypté sans avoir connaissance de la clé. Les procédés de cryptanalyse pour le chiffrement symétrique sont très nombreux et la plupart des attaques sont spécifiquement adaptées aux techniques de chiffrement. Les différents types d'attaques sont en fonction des données supposées connues par les attaquants, parmi elles on distingue :

➤ **Cryptanalyse statistique :**

- **L'attaque à texte chiffré seulement (Ciphertext-only attack) :** L'attaquant a connaissance du texte chiffré de plusieurs messages.
- **L'attaque à texte clair connu (Known-plaintext attack) :** Le cryptanalyste a accès à plusieurs textes chiffrés ainsi qu'aux textes clairs correspondants.
- **L'attaque à texte clair choisi (Chosen-plaintext attack) :** L'attaquant a accès à l'algorithme de chiffrement. Il l'utilise pour générer des couples (X_i, Y_i) de son choix. La différence principale par rapport l'attaque à texte clair connu est que le cryptanalyste peut choisir le texte à chiffrer.
- **L'attaque à texte chiffré choisi (Adaptive-plaintext attack) :** Le cryptanalyste a accès à l'algorithme de décryptage. Il peut choisir les textes à déchiffrer sans connaître la clé.
- **L'attaque par force brute (Brute-force attack) ou l'attaque exhaustive :** L'attaquant essaie toutes combinaisons de clés possibles jusqu'à l'obtention du texte clair.
- **L'attaque par canaux auxiliaires :** Toutes les façons d'analyser les propriétés inattendues d'un algorithme sont prises en compte pour réussir à casser le cryptosystème. Dans les algorithmes de chiffrement implémentés en hardware, par exemple, la consommation électrique pour chaque type de calcul du chiffrement peut être utile pour déduire certaines informations de la clé.
- **L'attaque algébrique :** Les attaques algébriques sont des attaques à clair connu qui exploitent des relations algébriques entre les bits du clair, ceux du chiffré et ceux de la clé secrète. La connaissance de plusieurs couples clairs-chiffrés fournit donc un système d'équations dont les inconnues sont les bits de la clé secrète. Ces derniers peuvent alors être retrouvés en résolvant le système, ce qui est possible s'il est de degré faible, de petite taille ou qu'il possède une structure particulière.

➤ **Cryptanalyse différentielle**

Elle a été proposée par Eli Biham et Adi Shamir en 1991. Elle permet de trouver la clé en utilisant une quantité de textes clairs. L'idée est de fournir comme entrée des textes clairs avec de légères différences (un bit par exemple). Ensuite, on analyse statistiquement le comportement des sorties selon les entrées pour retrouver la clé. En regardant comment les différences en entrée affectent les sorties, on peut établir des règles statistiques. Il existe plusieurs variantes des cryptanalyses différentielles, nous distinguons : différentielle tronquée, différentielle d'ordre supérieur et différentielles impossibles.

➤ **Cryptanalyse linéaire**

Elle a été inventée par Mitsuru Matsui en 1993 . Elle nécessite une quantité n de couples (texte clair, texte chiffré), tous chiffrés avec la même clé. Le principe est que le même message soit chiffré plusieurs fois avec des clés différentes pour construire une immense table (téraoctet) qui contient toutes les versions chiffrées de ce message. Lors d'une interception d'un message chiffré, on peut le retrouver dans la table et obtenir la clé qui avait été utilisée pour le cryptage. Cette attaque n'est bien sûr pas faisable car nous aurions besoin d'une table trop importante. Le génie d'Hellman a été de trouver un moyen pour réduire cette table, processus réalisable. Celui-ci consiste à faire une approximation linéaire de l'algorithme pour le simplifier

1.9. Conclusion :

Dans ce chapitre, nous avons présenté plusieurs techniques et quelques théories de la cryptographie qui vont nous permettre de comprendre cet axe de recherche. Nous avons évoqué les notions formelles de sécurité et leurs implications. Nous avons ensuite abordé les différents types de classifications des algorithmes de chiffrement et leurs contextes d'applications. Nous avons aussi observé que les clés ont un rôle important et le choix de leur longueur est cruciale pour rendre sûrs les cryptosystèmes. Ce chapitre a introduit aussi, les principaux algorithmes de cryptage symétrique, asymétrique, par flot et par bloc, et a présenté également les différentes formes d'attaques et leurs classifications.

2.1 Introduction :

Dans ce chapitre, nous détaillons les trois algorithmes AES, RSA et SHA-1 qui seront utilisés dans notre cryptosystème de transmission. « Advanced Encryption Standard » ou AES, aussi connu sous le nom de Rijndael, est un algorithme de chiffrement symétrique. Il remporta en octobre 2000 le concours AES, lancé en 1997 par le NIST et devient le nouveau standard de chiffrement pour les organisations du gouvernement des États Unis. Il a été également approuvé par la NSA (National Security Agency) pour les informations top secrètes. Il est issu d'un appel à candidatures international lancé en janvier 1997 et ayant reçu 15 propositions. Parmi ces 15 algorithmes, 5 furent choisis pour une évaluation plus poussée en avril 1999 : MARS, RC6, Rijndael, Serpent, et Twofish . Au bout de cette évaluation, ce fut finalement le candidat Rijndael, du nom de ses deux concepteurs Joan Daemen et Vincent Rijmen (tous les deux de nationalité belge) qui a été choisi. Ces deux experts en cryptographie étaient déjà les auteurs d'un autre algorithme: Square. AES est un sous-ensemble de Rijndael : il ne travaille qu'avec des blocs de 128 bits alors que Rijndael offre des tailles de blocs et de clés qui sont des multiples de 32 (compris entre 128 et 256 bits). Ce faisant, l'AES remplace le DES (choisi comme standard dans les années 1970) qui de nos jours devenait obsolète, car il utilisait des clés de 56 bits seulement. 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 un schéma de Feistel, sa complexité est moindre et il est plus facile à implémenter.

RSA (Rivest Shamir Adleman) est le premier système à clé publique solide à avoir été inventé, et le plus utilisé actuellement, est le système RSA. Publié en 1978 par Ron Rivest, Adi Shamir et Leonard Adleman. Le RSA est fondé sur deux principes mathématiques fondamentaux : la difficulté de factoriser des grands nombres, et l'arithmétique des congruences.

2.2 La structure d'AES [6] [7] [15] :

La figure 2.1 montre la structure de l'AES. La taille de bloc en clair est de 128 bits, soit 16 octets. La longueur de la clé peut être 16, 24 ou 32 octets (128, 192, ou 256 bits). L'algorithme est appelé AES-128, AES-192, ou AES-256, en fonction de la longueur de la clé. Ce bloc est copié dans une matrice d'état, qui est modifiée à chaque étape de chiffrement

ou de déchiffrement. Après la dernière étape, la matrice d'état est copiée dans une matrice de sortie. Ces opérations sont représentées sur la figure 2.2 a. De même, la clé est représentée comme une matrice carrée d'octets 4×4. La figure 2.2 b montre l'expansion de la clé de 128 bits. Les quatre premiers octets de la clé forment un mot.

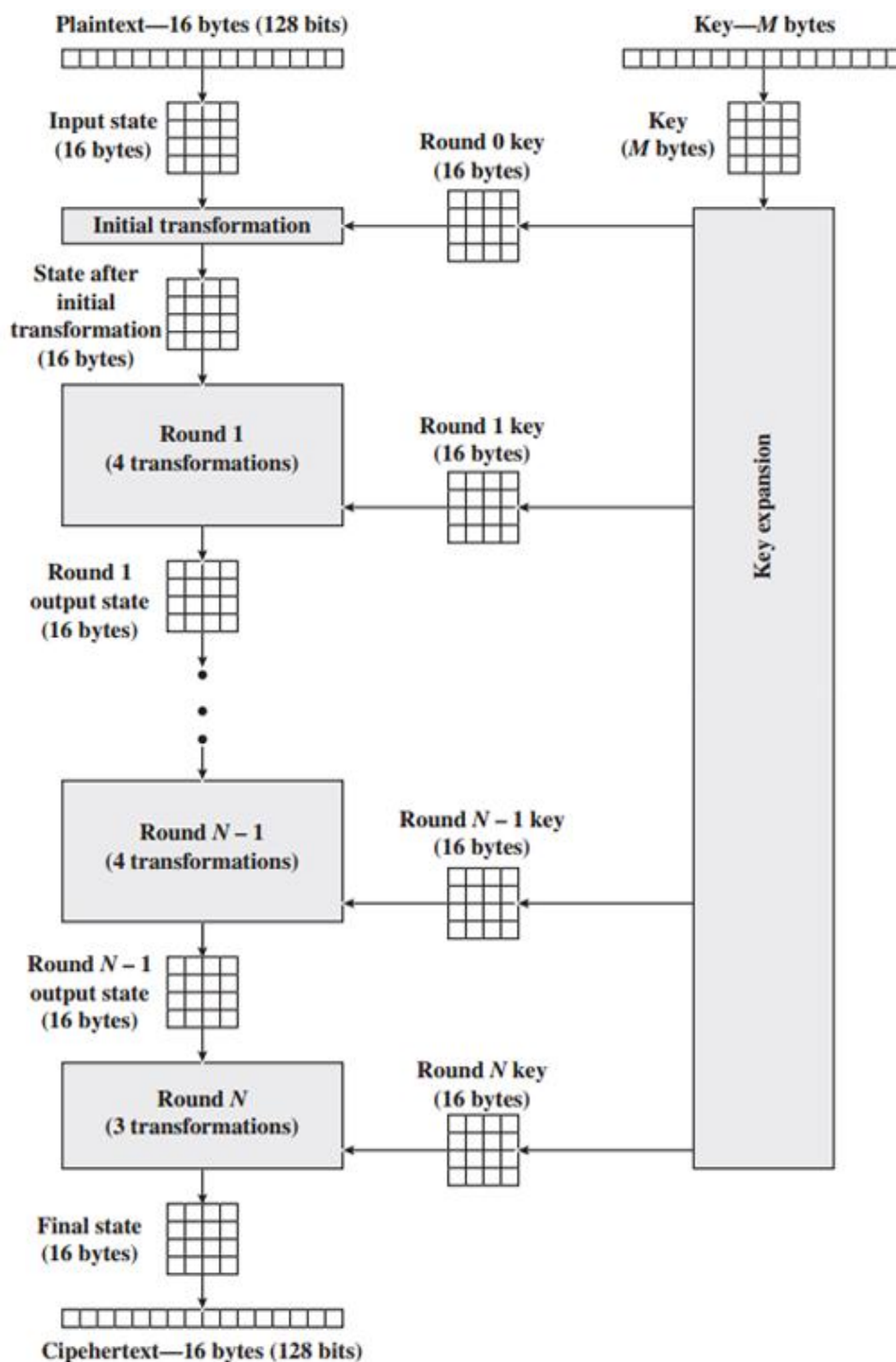


Figure 2.1. Principe de l'algorithme symétrique

Le chiffrement se compose de tours, où le nombre des tours dépend de la longueur de la clé: 10 tours pour une clé de 16 octets, 12 tours pour une clé de 24 octets, et 14 tours pour une clé de 32 octets (tableau 2.1).

Les premiers tours se composent de quatre fonctions: « SubBytes, ShiftRows, MixColumns et AddRoundKey » qui sont décrits par la suite. Le tour final ne contient que trois transformations, et il ya une seule transformation initiale (AddRoundKey) avant le premier tour. Chaque transformation prend un ou plusieurs matrices 4×4 en entrée et produit en sortie une matrice 4×4 . La figure 2.1 montre que le sortie de chaque tour est une matrice 4×4 , à la sortie de la ronde finale étant la partie chiffrée.

Nombre de tours	La longueur de la clé
10	16
12	24
14	32

Tableau 2.1. La longueur de clé en fonction du nombre des tours

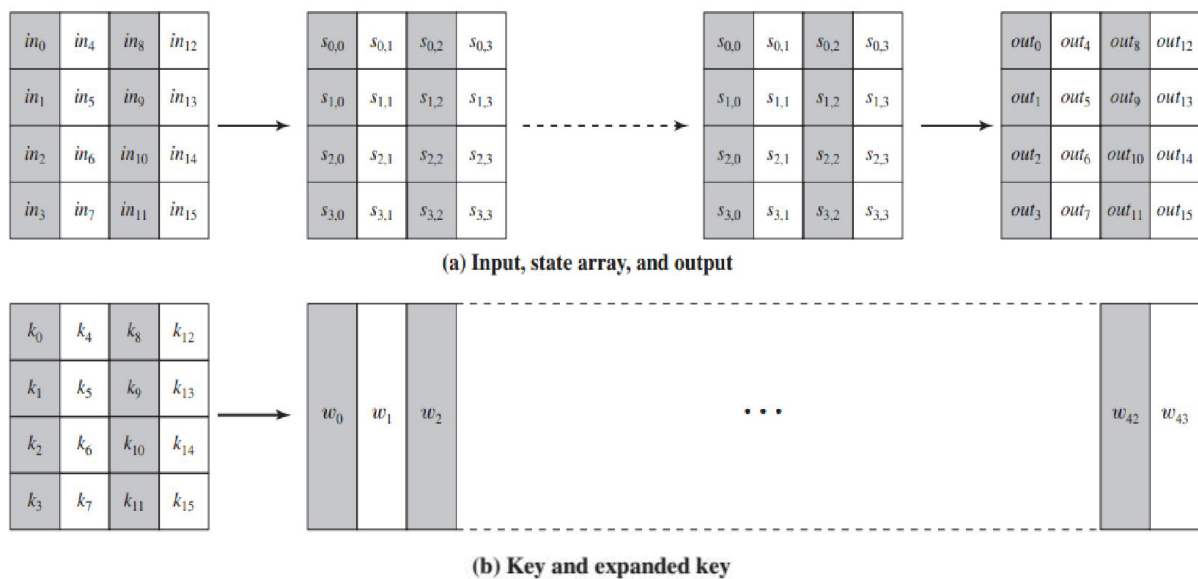


Figure 2.2. Structure de données d'AES (a) Forme matricielle de l'AES, (b) la clé étendue

La taille des clés (mots/octetets/bits)	4/16/128	6/24/192	8/32/256
La taille de texte clair en bloc (mots/octetets/bits)	4/16/128	4/16/128	4/16/128
Le nombre des tours	10	12	14
La taille de clé pour chaque tour(mots/octetets/bits)	4/16/128	4/16/128	4/16/128
La taille de clé étendu (mots/octetets/bits)	44/176	52/208	60/240

Tableau 2.2. *Longueur de la clé*

La figure 2.3 montre la structure de l'algorithme de chiffrement AES de façon plus détaillée. Avant d'entrer dans les détails, nous pouvons faire quelques remarques sur La structure de l'algorithme :

- Une caractéristique remarquable de cette structure est quelle n'est pas une structure de Feistel.
- La clé qui est fourni en entrée est élargie en une matrice de quarante-quatre Mots 32 bits, $w[i]$. Quatre mots distincts (128 bits) servent pour assurer le chiffrement pour chaque tour; elles sont indiquées dans la figure 2.3.
- Quatre phases différentes sont utilisées pour chaque tour : une de permutation et trois de substitution:
 - « Substitute bytes » utilise une boîte S pour effectuer une substitution octet par octet
 - « ShiftRows » Une simple permutation entre les octets « MixColumns » Une substitution qui rend l'utilisation de l'arithmétique autour $GF(2^8)$
 - « AddRoundKey » une simple XOR entre le bloc courant avec une partie de la clé étendue
- L'étape de « AddRoundKey » est la seule qui utilise la clé. Pour cette raison, le procédure de chiffrement commence et se termine par une étape de « AddRoundKey »
- Chaque étape est facilement réversible. Pour « Substitute bytes », « ShiftRows »,

« MixColumns » une fonction inverse est utilisée dans l'algorithme de déchiffrement. pour l'étape de « AddRoundKey », linverse est réalisé par XOR : $A \oplus B \oplus B = A$.

- L'algorithme de chiffrement n'est pas identique à l'algorithme de déchiffrement, ceci est une particularité de l'AES.
- La tour finale de chiffrement et le déchiffrement se compose seulement de trois étapes. Encore une fois, ceci est une particularité de la structure de l'AES.

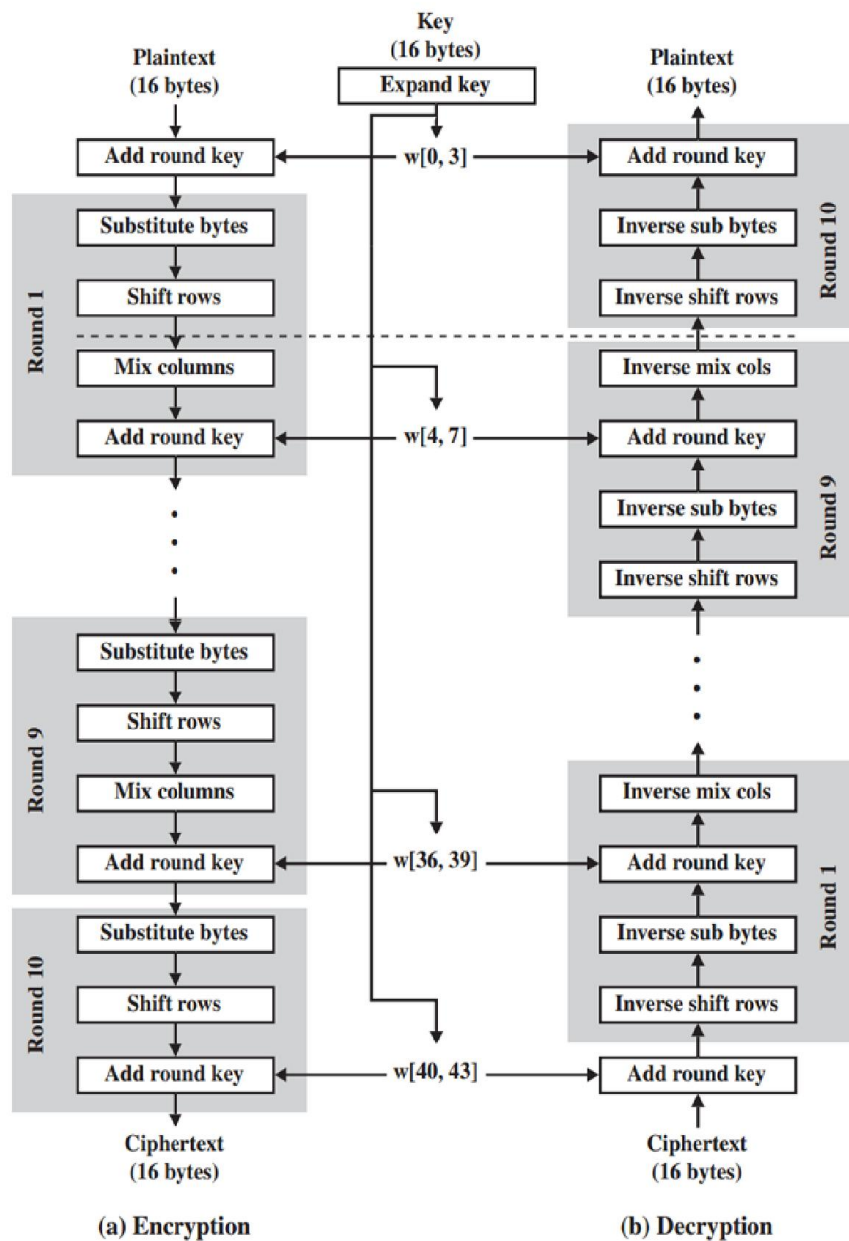


Figure 2.3. AES : (a) Chiffrement (b) déchiffrement

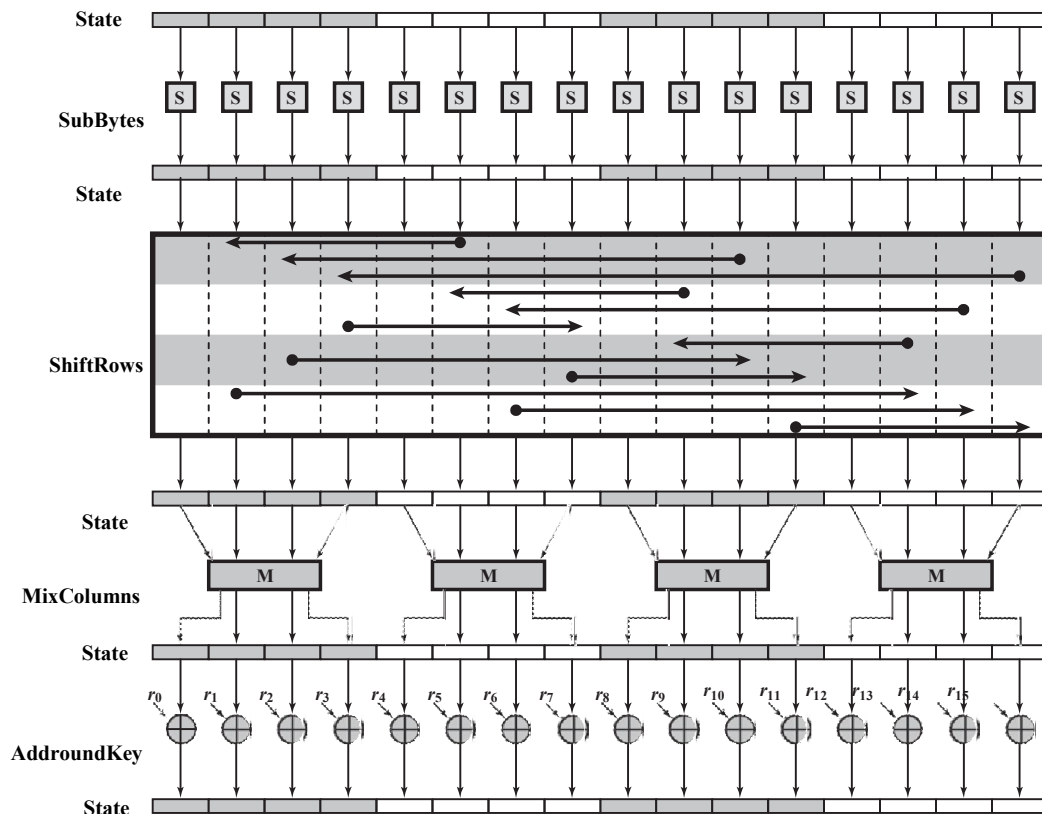


Figure 2.4. Les tours de chiffrement d'AES

2.2.1 Les fonctions de transformation de l'AES :

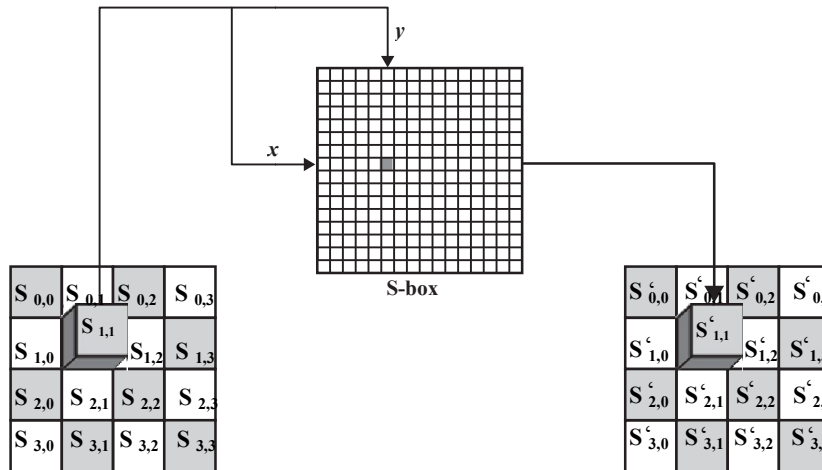
Nous passons maintenant à la discussion de chacun des quatre transformations utilisées en AES.

2.2.1.1 « Substitute Bytes Transformation »

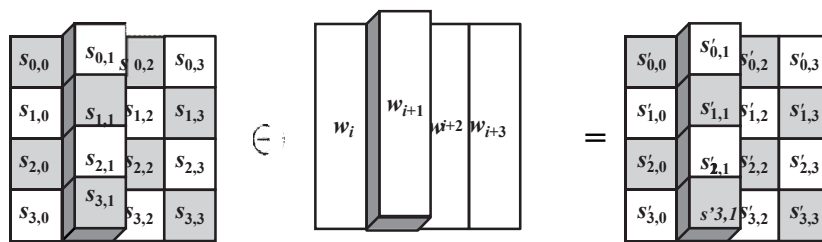
Nommée aussi « SubBytes », cette transformation est une simple consultation de table (figure 2.5 a). AES définit une matrice d'octets 16×16 , appelé une boîte-S (Tableau 2.3 a), qui contient l'ensemble des 256 valeurs possibles de 8 bits.

Chaque octet individuel est remplacé par un nouvel octet de la façon suivante: Les 4 bits les

plus à gauche de l'octet sont utilisés en tant que valeur de ligne et les 4 bits les plus à droite sont utilisés en tant que valeur de colonne. Ces valeurs de rangée et de colonne servent d'index dans la boîte-S pour sélectionner le nouvel octet. Par exemple, la valeur hexadécimale 95 est référencée ligne 9, colonne 5 dans S-box, qui contient la valeur 2A, la valeur 95 est remplacée par la valeur 2A



(a) Substitue byte transformation



(b) Add round key transformation

Figure 2.5. Opérations aux niveaux d'octets

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	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

(a) S-box

		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

(b) Inverse S-box

Tableau 2.3. S-box AES a) direct, b) inverses

Nous donnons ci-après un exemple de la transformation des « SubBytes » :

EA	04	65	85	→	87	F2	4D	97
83	45	5D	96		EC	6E	4C	90
5C	33	98	B0		4A	C3	46	E7
F0	2D	AD	C5		8C	D8	95	A6

La boîte S est construite de la façon suivante :

1- Initialisation de la boîte S avec les valeurs d'octets dans l'ordre croissant ligne par ligne .la première ligne contient {01} {02} {03}.....{0F}, la deuxième ligne contient {10} {11} {12}.....{1F} ...ect

Ainsi, la valeur de l'octet à la ligne x, et la colonne y est {yx}.

2- remplacer chaque octet dans S-Box par l'inverse multiplicatif dans le corps finier $GF(2^8)$, la valeur {00} ne change pas.

3- Considérons que chaque octet dans S-Box se compose de 8 bits ($b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0$), Appliquer la transformation suivante pour chaque bit de chaque octet :

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \quad \text{II.1}$$

Avec c_i est le bit numéro i de l'octet c de la valeur {63}, la sous forme matricielle de transformation est comme suit :

$$\begin{bmatrix} b'0 \\ b'1 \\ b'2 \\ b'3 \\ b'4 \\ b'5 \\ b'6 \\ b'7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \\ b4 \\ b5 \\ b6 \\ b7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \text{II.2}$$

« Inverse substitute byte transformation » est nommée InvSubBytes, utilise la Table 2.3 b pour trouver l'inverse

La boîte de S-inverse (figure 2.6 b) est construite en appliquant l'inverse de la transformation de l'équation (II.1), puis en prenant l'inverse multiplicatif autour de $GF(2^8)$.

La transformation inverse est

$$b'_i = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i \quad \text{Avec} \quad d_i = \{05\} \quad \text{II.3}$$

Nous pouvons illustrer cette transformation comme suit :

$$\begin{bmatrix} b'0 \\ b'1 \\ b'2 \\ b'3 \\ b'4 \\ b'5 \\ b'6 \\ b'7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \\ b4 \\ b5 \\ b6 \\ b7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{II.4}$$

Nous résumons les étapes de construction de la boîte S-Box et son inverse dans la figure suivante :

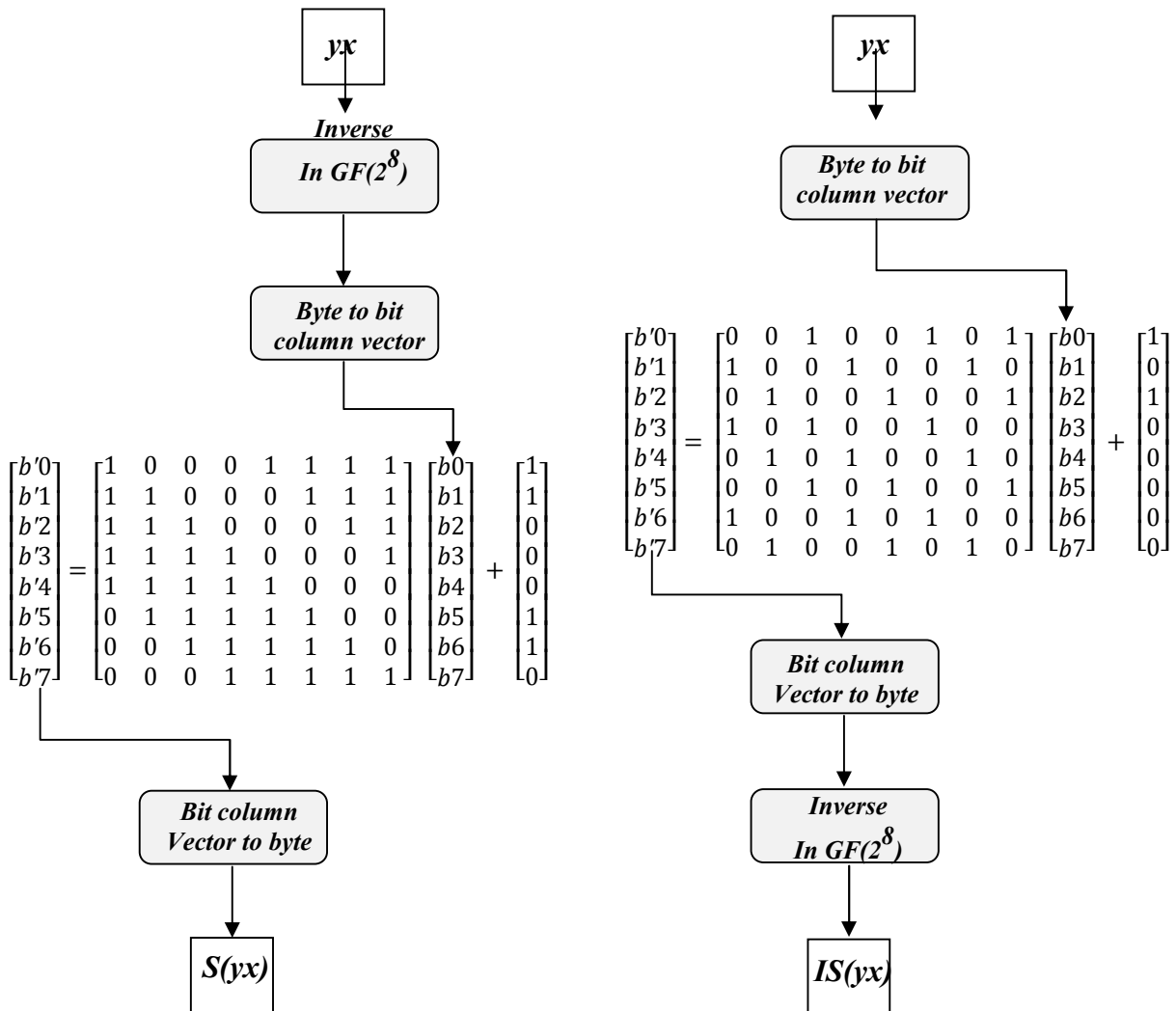


Figure 2.6. Construction des boîte de AES S-Box et IS-Box

2.2.1.2 « ShiftRows Transformation »

La transformation « ShiftRows » est représentée dans la Figure 2.7. La première ligne de la matrice d'état n'est pas modifiée. Pour la deuxième ligne, un décalage circulaire de gauche du premier octet est réalisé. Pour la troisième ligne, un décalage circulaire de gauche de deuxième octet est réalisé. Pour la quatrième ligne aussi un décalage circulaire de gauche de troisième octet est réalisé. Ce qui suit, est un exemple de « ShiftRows » :

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

« Inverse shift row transformation », appelé « ShiftRows », effectue les déplacements circulaires dans le sens inverse pour chacune des trois dernières lignes.

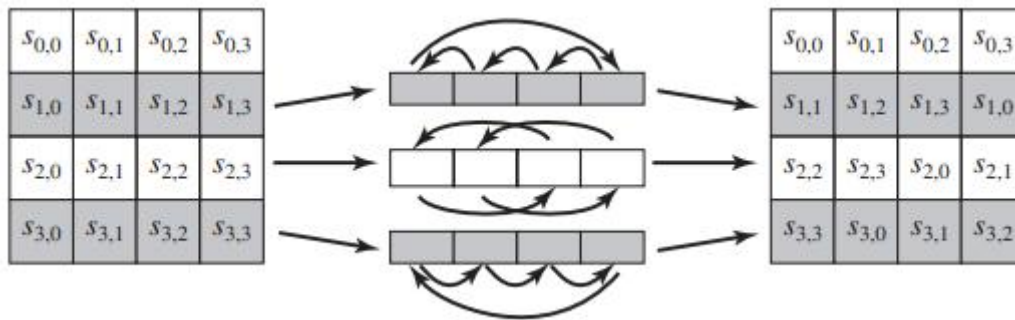


Figure 2.7 ShiftRows Transformation

2.2.1.3 « MixColumns Transformation »

Appelé « MixColumns », cette transformation agit sur chaque colonne individuellement. Chaque octet d'une colonne est remplacé par une nouvelle valeur qui est en fonction de l'ensemble des quatre octets de cette colonne. La transformation peut être définie par la matrice d'état (figure 2.8):

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \quad \text{II.5}$$

Le MixColumns sur une seule colonne de l'État peut être exprimé comme suit :

$$\begin{aligned}
 S'_{0,j} &= (2 \cdot S_{0,j}) \oplus (3 \cdot S_{1,j}) \oplus S_{2,j} \oplus S_{3,j} \\
 S'_{1,j} &= S_{0,j} \oplus (2 \cdot S_{1,j}) \oplus (3 \cdot S_{2,j}) \oplus S_{3,j} \\
 S'_{2,j} &= S_{0,j} \oplus S_{1,j} \oplus (2 \cdot S_{2,j}) \oplus (3 \cdot S_{3,j}) \\
 S'_{3,j} &= (3 \cdot S_{0,j}) \oplus S_{1,j} \oplus S_{2,j} \oplus (2 \cdot S_{3,j})
 \end{aligned}
 \tag{II.6}$$

« $\cdot$ » pour indiquer la multiplication sur le corps fini $GF(2^8)$. Ce qui suit est un exemple de MixColumns :

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

« Inverse mix column transformation », est défini par la multiplication matricielle suivante :

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}
 \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix}
 =
 \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}
 \tag{II.7}$$

Il n'est pas très clair que l'équation (II.5) est l'inverse de l'équation (II.7). Nous devons montrer ce qui est équivalent à :

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}
 \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}
 \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix}
 =
 \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}
 \tag{II.8}$$

On obtient :

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}
 \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}
 =
 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \tag{II.9}$$

Pour vérifier la première colonne de l'équation (II.9), nous avons besoin de montrer :

$$\begin{aligned}
 (\{0E\} \cdot \{02\}) \oplus \{0B\} & \oplus \{0D\} & \oplus (\{09\} \cdot \{03\}) &= \{01\} \\
 (\{09\} \cdot \{02\}) \oplus \{0E\} & \oplus \{0B\} & \oplus (\{0D\} \cdot \{03\}) &= \{00\} \\
 (\{0D\} \cdot \{02\}) \oplus \{09\} & \oplus \{0E\} & \oplus (\{0B\} \cdot \{03\}) &= \{00\} \\
 (\{0B\} \cdot \{02\}) \oplus \{0D\} & \oplus \{09\} & \oplus (\{0E\} \cdot \{03\}) &= \{00\}
 \end{aligned} \quad \text{II.10}$$

Les autres équations peuvent être vérifiées de façon similaire

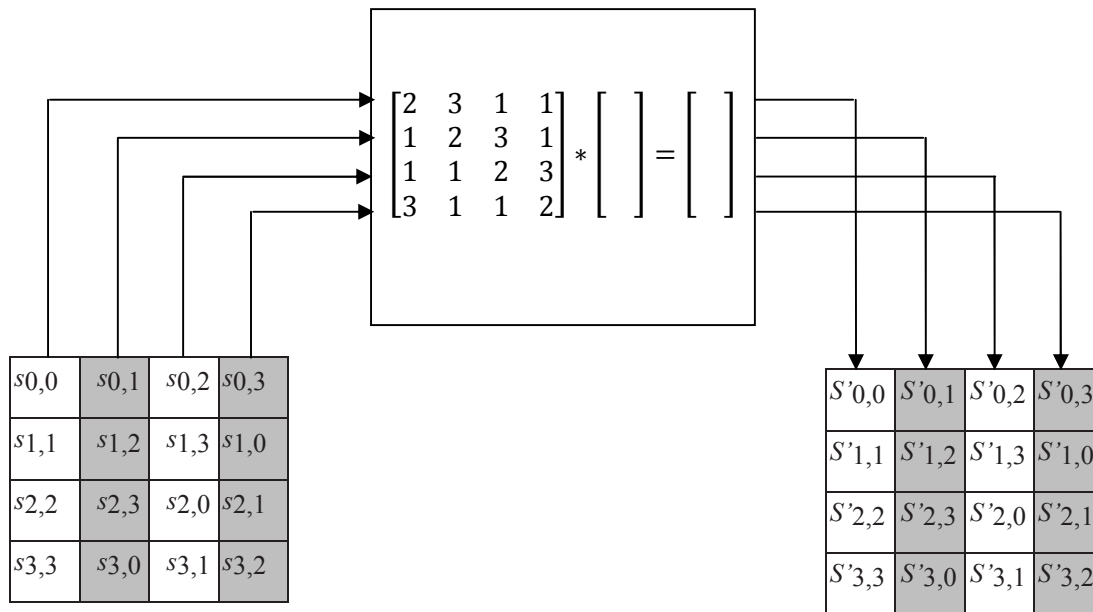


Figure 2.8 MixColumns Transformation

2.2.1.4 « AddRoundKey Transformation »

Appelé « AddRoundKey », les 128 bits de la matrice d'état de texte claire sont XORed avec 128 bits de la clé. Comme le montre la figure 2.9, Un exemple de AddRoundKey :

$$\begin{array}{|c|c|c|c|} \hline 47 & 40 & A3 & 4C \\ \hline 37 & D4 & 70 & 9F \\ \hline 94 & E4 & 3A & 42 \\ \hline ED & A5 & A6 & BC \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|} \hline AC & 19 & 28 & 57 \\ \hline 77 & FA & D1 & 5C \\ \hline 66 & DC & 29 & 00 \\ \hline F3 & 21 & 41 & 6A \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline EB & 59 & 8B & 1B \\ \hline 40 & 2E & A1 & C3 \\ \hline F2 & 38 & 13 & 42 \\ \hline 1E & 84 & E7 & D6 \\ \hline \end{array}$$

La première matrice est la matrice d'état

La deuxième matrice est la matrice des clés inverse add round key transformation est identique à AddRoundKey.

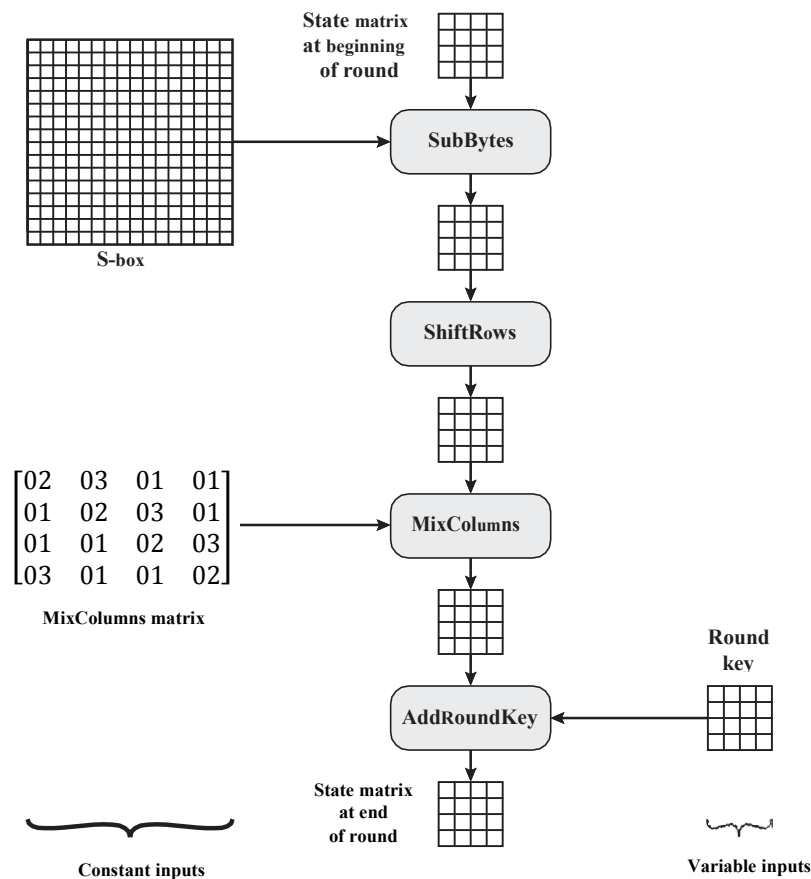


Figure 2.9 *AddRoundKey Transformation*

2.2.2 La clé de l'AES étendu :

L'algorithme d'extension des clés AES prend en entrée quatre mots (16 octets) et produit 44 mots (176 octets). Ceci est suffisant pour fournir les clés nécessaires pour 10 tours de l'algorithme AES

La Figure 2.10 illustre la génération de la clé étendue à l'aide de la fonction complexe g , la fonction g est constituée de sous-fonctions.

```

ExtdeClé (byte clé[16], mot w[44])
{
    mot temp

    Pour (i = 0; i < 4; i++) w[i] = (clé[4*i], clé[4*i+1],
                                     clé[4*i+2], clé[4*i+3]);

    Pour (i = 4; i < 44; i++)
    {
        temp = w[i - 1];
        Si (i mod 4 = 0)    temp = SubWord (RotWord (temp))
                                 $\oplus$  Rcon[i/4];
        w[i] = w[i-4]  $\oplus$  temp
    }
}

```

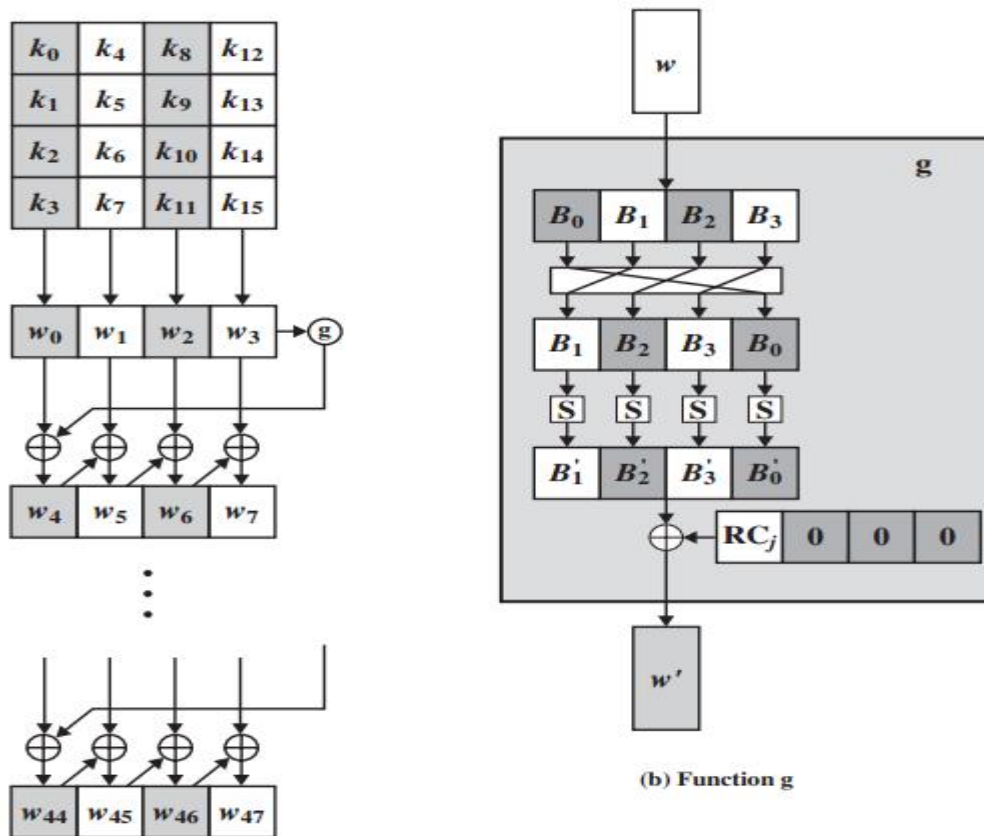


Figure 2.10 la clé de L'AES étendu

« RotWord » effectue un décalage d'un octet circulaire gauche sur un mot. Cela signifie que le mot d'entrée est $[B_0, B_1, B_2, B_3]$ transformé en $[B_1, B_2, B_3, B_0]$.

« SubWord » effectue une substitution sur chaque octet, en utilisant S-Box. Le résultat des étapes 1 et 2 est XORed avec la constante $Rcon[j]$.

Le constant est un mot donc les trois octets les plus à droite sont toujours à 0. Ainsi, leffet dun « OU exclusif » dun mot avec $Rcon$ est d'effectuer seulement une opération XOR sur loctet le plus à gauche de ce mot. La constante est différente pour chaque tour et est définie comme $Rcon[j] = (RC[j], 0, 0, 0)$

Avec $RC[1]=1$, $RC[j] = 2 \times RC[j-1]$, et avec la multiplication définie sur le domaine $GF(2^8)$.

La valeur de $RC[j]$ en hexadécimale est :

j	1	2	3	4	5	6	7	8	9	10
$RC[j]$	01	02	04	08	10	20	40	80	1B	36

Tableau 2.4. *La valeur de $RC[j]$*

2.3. Algorithme asymétrique RSA :

Le chiffrement RSA est un algorithme de cryptographie asymétrique, très utilisé et très efficace pour l'échange des clés des algorithmes symétriques comme l'AES. Cet algorithme a été décrit en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman.

2.3.1 Cryptosystème a clé publique :

Les algorithmes asymétriques sont basés sur une clé pour le chiffrement et une clé associée différent, pour le déchiffrement. Ces algorithmes ont la caractéristique importante suivante. Il est impossible de trouver la clé de déchiffrement malgré la connaissance de l'algorithme cryptographique et la clé de chiffrement.

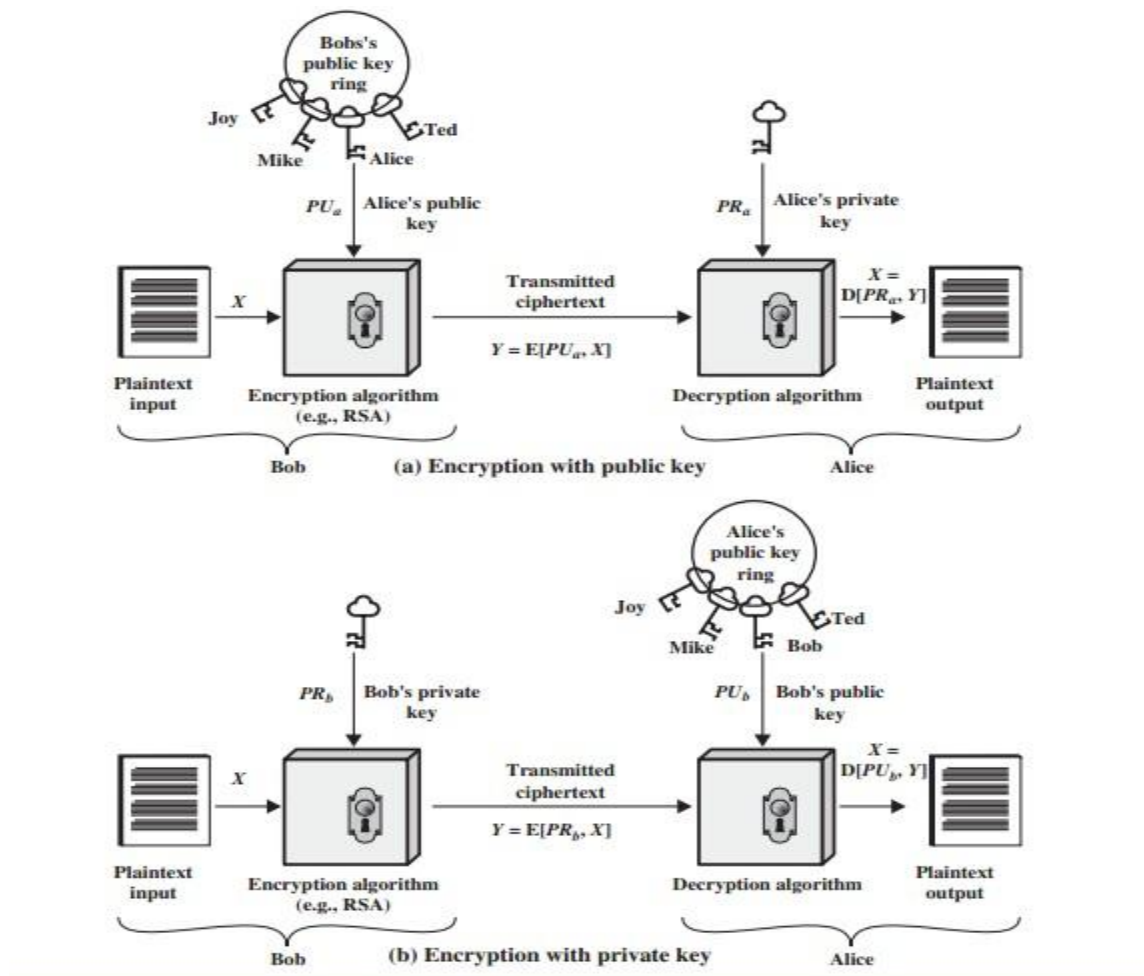
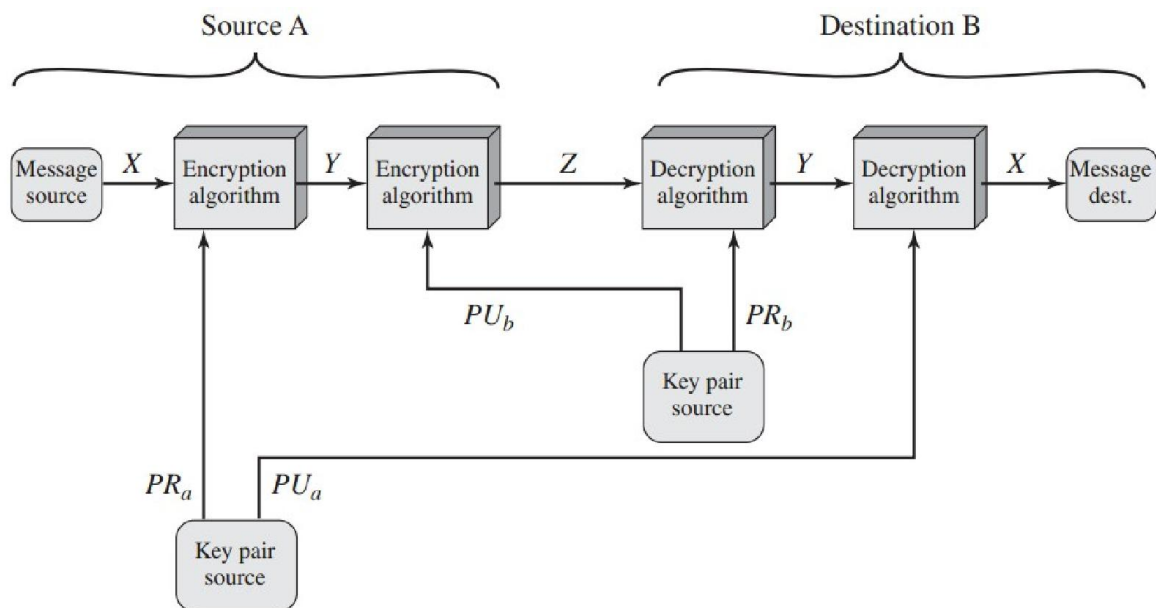


Figure 2.11. *Cryptosystème à clé publique*

D'après la figure 2.11, nous résumons les étapes d'un cryptosystème à clé asymétrique:

- 1- Chaque utilisateur génère une paire de clés à utiliser pour le chiffrement et le déchiffrement des messages.
- 2- Chaque utilisateur place sa clé publique dans un registre public ou autre fichier accessible.
- 3- Si Bob veut envoyer un message confidentiel à Alice, Bob chiffre le message utilisant la clé publique d'Alice.
- 4- Quand Alice reçoit le message, elle le déchiffre à l'aide de sa clé privée.

Il est, cependant, possible d'assurer à la fois la fonction d'authentification et de confidentialité



par une double utilisation du système à clé publique comme montre la figure 2.12:

Figure 2.12. *Cryptosystème à clé publique (authentification et confidentialité)*

Dans ce cas, nous commençons par chiffrer un message, en utilisant la clé privée de l'expéditeur. Ceci permet d'obtenir la signature numérique. Ensuite, nous chiffons à nouveau, en utilisant la clé publique de récepteur. Le texte chiffré final peut être déchiffré que par le destinataire qui possède la clé privée correspondante. Ainsi, la confidentialité est assurée.

2.3.2 L'algorithme RSA :

Le premier système à clé publique solide à avoir été inventé, et le plus utilisé actuellement, est le système RSA. Le RSA est fondé sur deux principes mathématiques fondamentaux : la difficulté de factoriser des grands nombres, et l'arithmétique des congruences.

2.3.3 Principe de fonctionnement de l'algorithme RSA :

Pour qu'Alice puisse échanger sa clé avec Bob, elle doit d'abord la calculer en mettant en œuvre des notions mathématiques remarquables par leur simplicité. L'algorithme RSA est ainsi défini par 03 phases :

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

➤ **Génération des clés :**

Cette phase peut se résumer en 03 étapes :

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

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

Et fait leur produit :

$$n = p \times q = 1643$$

2^{er} étape : Alice détermine la fonction d'Euler associée à n déjà calculé en utilisant la formule :

$$\phi(n) = (p-1) \times (q-1)$$

$$\phi(n) = 30 \times 52 = 1560$$

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

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

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

La clé publique est donc (11, 1643)

3^{er} étape : Cette dernière étape consiste à trouver la clé privée « d » qui correspond à la clé

publique choisie précédemment avec d compris entre 1 et $\phi(n)$, pour se faire, il faut résoudre l'équation suivante :

$$e \cdot d \equiv 1 \pmod{\phi(n)}$$

$$\text{Donc : } e \cdot d = k \phi(n) + 1$$

Selon notre exemple on aura :

$$e \cdot d = k \phi(n) + 1$$

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

Pour $k=6$ on aura : $11 \cdot d = k \cdot 1560 + 1$ on aura $d = 851$

Le couple (d, n) constitue la clé privée

La clé privée est donc $(851, 1643)$

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

En fin, Alice et Bob disposent toutes les clés indispensables au chiffrement et au déchiffrement des messages après la transmission ou la publication de sa clé publique (e, n) .

Maintenant, il faut qu'elle conserve sa clé privée (d, n) et qu'elle n'oublie jamais les nombres p et q .

➤ Chiffrement

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

Il va procéder au cryptage de la manière suivante :

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

M = ANEMONE

M = A N E M O N E

M = 01 14 05 13 15 14 05

- Il découpe son message numérisé en blocs de même longueur représentant chacun une taille égale ou inférieure à celle de n ce qui empêche la simple substitution. Dans notre exemple la taille de n est 3, ce qui donne des tranches m_i de 03 chiffres chacune, le message devient :

M = 001 140 513 151 405

M = m_1 m_2 m_3 m_4 m_5

Chaque bloc m_i est chiffré par l'équation :

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

$$C_1 = m_1^{11} \bmod 1643 = 001$$

$$C_2 = m_2^{11} \bmod 1643 = 109$$

$$C_2 = m_2^{11} \bmod 1643 = 109$$

.. . . .

.. . . .

$$C_5 = m_5^{11} \bmod 1643 = 374$$

$001^{11} \bmod 1643 =$	1
$140^{11} \bmod 1643 =$	109
$513^{11} \bmod 1643 =$	890
$151^{851} \bmod 1643 =$	1453
$405^{11} \bmod 1643 =$	374

Alors le message chiffré C sera:

$$C = \begin{matrix} c1 & c2 & c3 & c4 & c5 \\ 0001 & 0109 & 0890 & 1453 & 0374 \end{matrix}$$

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

➤ **Déchiffrement :**

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

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

Chacun des blocs c_i du message chiffré sera déchiffré par l'équation :

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

Ce qui lui donne :

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

M = 001 140 513 151 405

M= ANEMONE

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

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

➤ **Résumé :**

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

- 1- Génération de 2 nombres premiers p et q
- 2- Calcul de $n = p \times q$
- 3- Déterminer e tel que $3 < e < \phi(n)$ et $(e, \phi(n)) = 1$
- 4- Calculer d tel que $e \times d \equiv 1 \pmod{\phi(n)}$
- 5- Clé publique : (e, n)
- 6- Clé privée : (d, n)
- 7- p et q doivent rester secrets, voire supprimés
- 8- $C = M^e \pmod{n}$ et $M = C^d \pmod{n}$

Les deux figures suivantes représentent respectivement la manière de chiffrer un bloc de texte, et un exemple qui explique bien le fonctionnement de l'algorithme RSA en blocs.

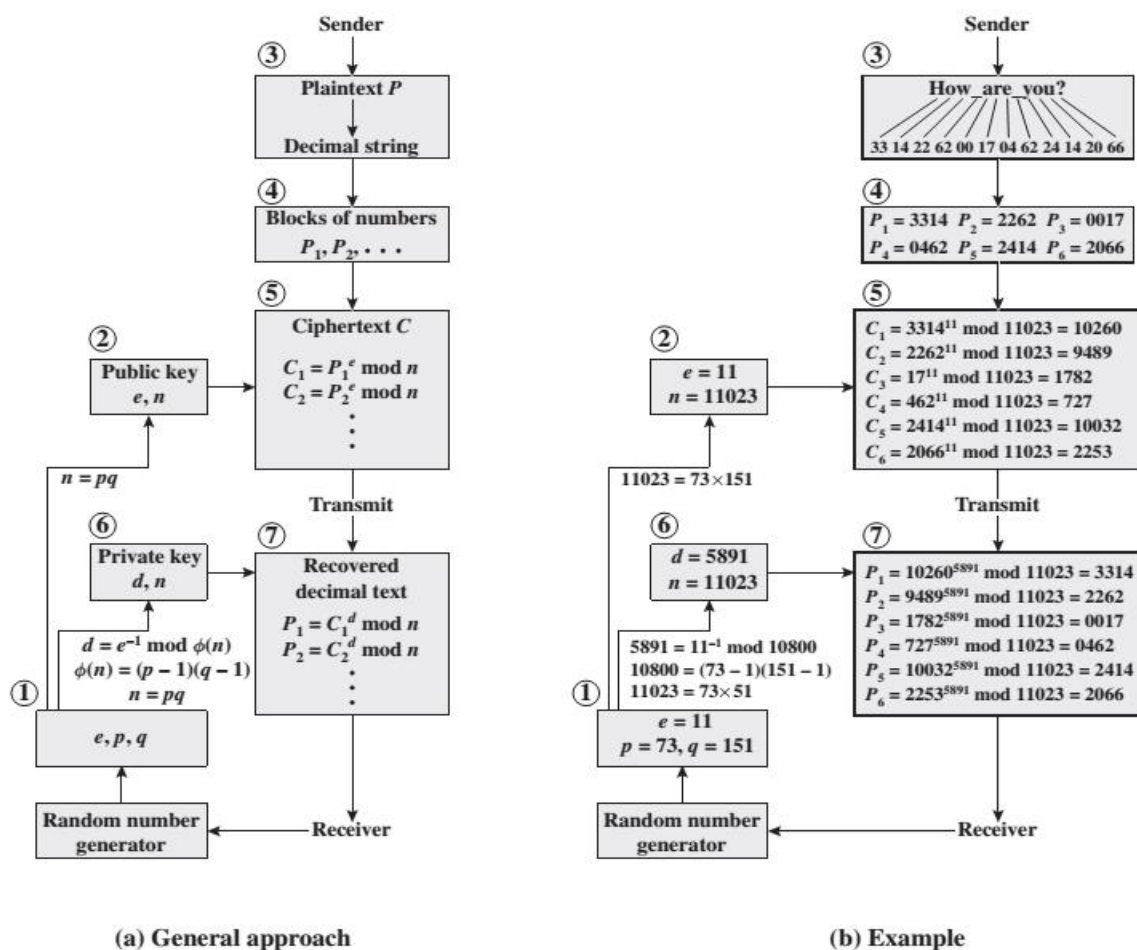


Figure 2.13. Cryptage à plusieurs blocs

2.4 Principe de fonctionnement de l' SHA-1 [16] :

SHA (Secure Hash Algorithm) a été conçu par NSA (Nation Security Agency) et publié par NIST (un organisme chargé des standards). En mai 1993, sa première version correspond à SHA-0. Deux ans plus tard (en 1995), suite à la découverte de faiblesses, la NSA publie une révision de son standard (SHA-0) pour étendre ses capacités en matière de sécurité : c'était le SHA-1.

La famille SHA se divise en trois catégories : SHA-1 , SHA-2 et SHA-3 . Jusqu'à 2005, il était l'algorithme généralement préféré pour le hachage, mais des rumeurs de cassage le font peu à peu évoluer vers des versions plus sophistiquées.

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

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

Avec :

- A=67452301 (en hexadécimal)
- B=EFCDA89
- C=98BADCFE
- D=10325476
- E=C3D2E1F0

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

- Elles travaillent sur 3*32 bits et fournissent un résultat sur 32 bits.
- Pour le 1^{er} groupe de 20 rounds : $0 \leq t \leq 19$, avec : t étant le numéro d'un round

- $f_1 = f(t, B, C, D) = (B \wedge C) \vee (\neg B \wedge C)$; ($\neg$: complément binaire)
- Pour le 2^{ème} groupe de 20 rounds : $20 \leq t \leq 39$, $f_2 = f(t, B, C, D) = B \oplus C \oplus D$
- Pour 3^{ème} groupe de 20 rounds : $40 \leq t \leq 59$,

$$F_3 = f(t, B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$$

- Pour le 4^{ème} groupe de 20 rounds : $60 \leq t \leq 79$, $F_4 = f(t, B, C, D) = B \oplus C \oplus D$

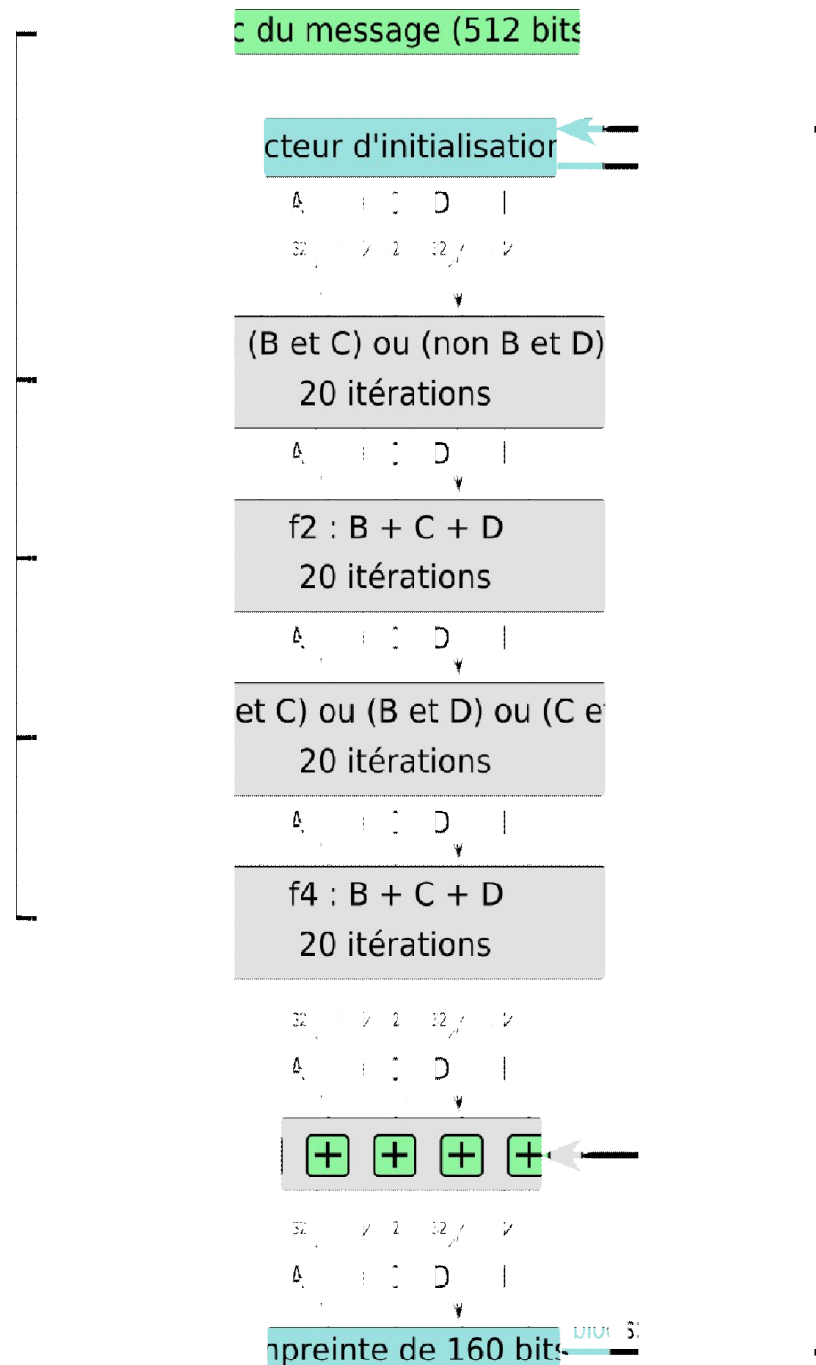


Figure 2.14.principe de fonctionnement de SHA-1

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

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

- Pour $0 \leq t \leq 15$: W_t prend les 16 premières valeurs du de 512 bits.
- Pour $16 \leq t \leq 79$: $W_t = w[t-3] \oplus w[t-8] \oplus w[t-14] \oplus w[t-16]$

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

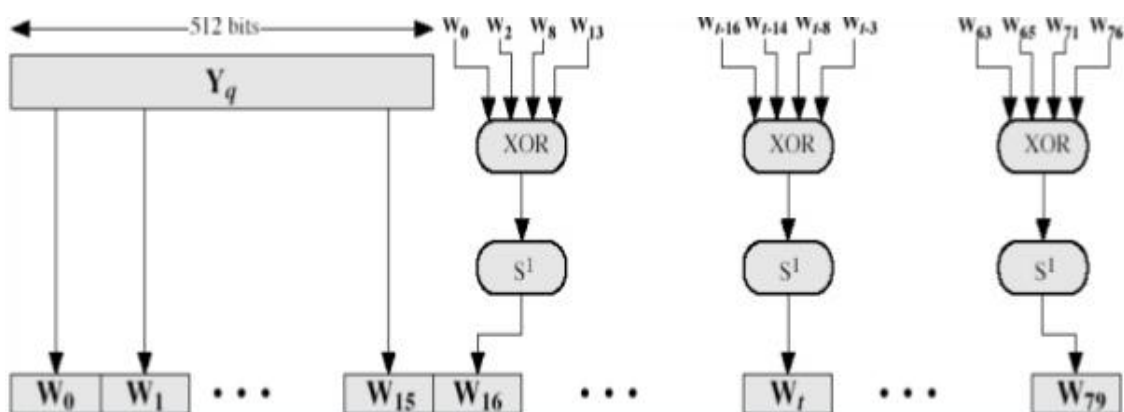


Figure 2.15 Calcul des W_t dérivés du bloc de 512bits

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

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

- Pour le 1^{er} groupe de 20 rounds : $0 \leq t \leq 19$: $K_t = 5A827999$
- Pour le 2^{eme} groupe de 20 rounds : $20 \leq t \leq 39$: $K_t = 6ED9EBA1$
- Pour le 3^{eme} groupe de 20 rounds : $40 \leq t \leq 59$: $K_t = 8F1BBCDC$
- Pour le 4^{eme} groupe de 20 rounds : $60 \leq t \leq 79$: $K_t = CA62C1D6$

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

$$(A, B, C, D, E) \leftarrow (E + f(t B, C, D) + (A \ll 5) + W_t + K_t), A, (B \ll 30), C, D)$$

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

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

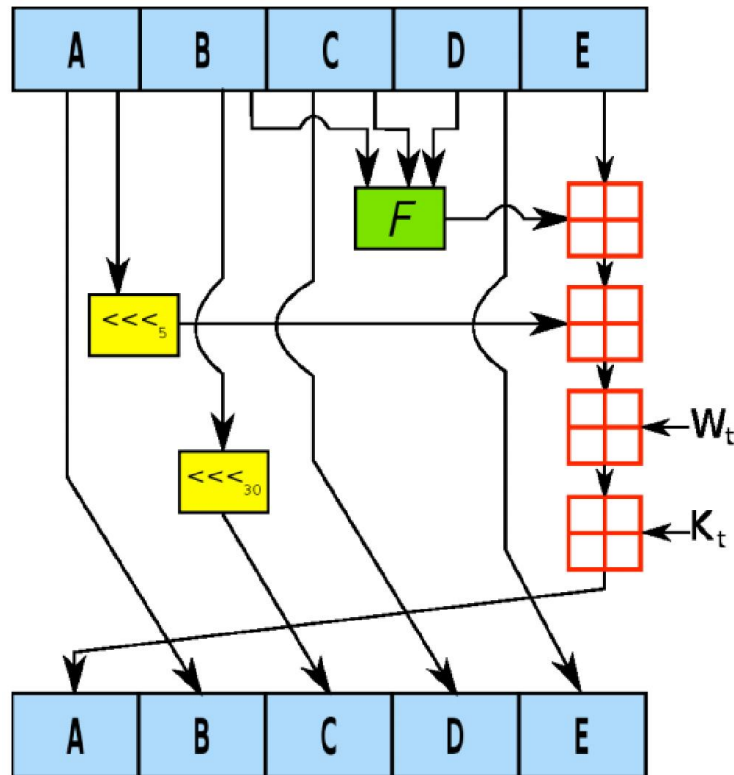


Figure 2.16 Fonction de ronde du SHA-1

Récapitulatif :

1. complétion.
2. découpage.
3. initialisation : initialiser 5 buffers de 32 bits (=160 bits) : A, B, C, D, E
4. calcule interactif : 4 rondes de 20 itérations chacune. Ces rondes ont une structure similaire mais utilisent des fonctions primitives différentes (f1, f2, f3, f4). Le SHA utilise des constantes additives K_t ($0 \leq t \leq 79$). Le résultat est utilisé pour initialiser les buffers du bloc suivant.

Chaque ronde comprend 20 étapes qui manipulent ainsi les 5 registres :

$$(A, B, C, D, E) \leftarrow (E + f(t, B, C, D) + (A \ll 5) + W_t + K_t), A, (B \ll 30), C, D)$$

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

5. le condensé final constitue le condensé attendu.

2.5 Conclusion :

Nous avons présenté dans ce chapitre, la structure détaillée des algorithmes AES, RSA et SHA-1. Nous avons aussi expliqué comment obtenir la clé étendue à partir d'une clé de longueur 128 bits pour l'algorithme AES. Et nous avons défini les étapes à suivre pour générer la paire de clés privée-publique avec l'algorithme RSA. L'utilisation de ces deux algorithmes est avantageuse. Ils sont très rapides et n'ont pas été cassés jusqu'à nos jours.

Le seul inconvénient de l'AES réside dans sa structure algébrique simple, et pour l'RSA, son utilisation avec des clés de chiffrement 1024 bits est peu sûre. De fait, Il est très important de comprendre le fonctionnement de ces deux derniers algorithmes, dans le prochain chapitre nous les utiliserons pour la transmission sécurisée des images. L'algorithme AES sera utilisée pour assurer la confidentialité des images, et l'algorithme RSA assurera l'échange des clés d'une façon sécurisée, il assure aussi la fonction d'authentification ainsi la fonction de hachage SHA-1 vérifie l'intégrité des données transmises.

3.1 Introduction

Dans la première partie de ce chapitre, nous allons présenter une méthode de chiffrement dite hybride qui associe les trois algorithmes étudiés au chapitre II que nous allons appliquer à la transmission d'images. La seconde partie sera consacrée à l'étude des performances, et à l'analyse de la sécurité du cryptosystème conçu.

3.2 le cryptosystème de transmission

Ci-dessous est décrit le cryptosystème hybride de transmission d'images avec ses deux blocs d'émission et de réception.

3.2.1 Bloc émetteur

Dans ce bloc, quatre opérations principales se déroulent dans l'ordre suivant :

1. Une clé K de taille 128 bits est générée par AES, celle-ci fera office de clé de session.
2. Une fois la clé générée, K est utilisé lors du chiffrement de l'image avec l'algorithme AES. nous choisissons un mode parmi les quatre mode pour la transmission d'image, nous représentons ci-dessous les quatre algorithmes de mode de chiffrement : ECB , CBC,CFB, OFB .
3. Afin de transmettre la clé de session d'une manier sécurisée, l'émetteur chiffre cette clé en utilisant l'algorithme à clé publique RSA de la manière suivant :
 - L'émetteur qui est un possession de ses clés privées $[n,d]$ et publique $[n,e]$ signe la clé de session en la chiffrant a l'aide de sa clé privé, une clé K' signée est alors obtenue. Cette clé K est chiffrée une deuxième fois en utilisant la publique du récepteur ce qui lui permet de générer la clé K'' va ensuit transmettre à travers le canal. Cette procédure de chiffrement assure l'authenticité de la clé et seul le récepteur peut alors déchiffre l'image envoyée.
4. la 4ème et dernier étape consiste alors à générer une empreinte numérique de l'image original en utilisant la fonction de hachage SHA-1, empreinte qu'on va signer en utilisant le RSA avec de la clé privée de l'émetteur.

Le tout {image chiffrée+K'' la clé chiffrée+empreinte chiffrée} est alors transmis à travers le canal public.

3.2.2 Bloc récepteur :

A la réception des fonctions inverses sont utilisées dans l'ordre suivant :

1. Le récepteur déchiffre la clé K' avec une fonction inverse de l'algorithme RSA en utilisant sa clé privée, et récupère K la clé de session toujours à l'aide du RSA en utilisant la clé publique de l'émetteur.
2. Il utilise alors K pour déchiffrer l'image reçue à l'aide d'une fonction inverse de l'algorithme de l'AES.
3. Dans le souci de vérifier l'intégrité des données reçues et authentifier leur expéditeur, le récepteur procède de la manière suivante :
 - Dans un premier temps il déchiffre l'empreinte reçue avec l'algorithme RSA en utilisant la clé publique de l'émetteur puisque cette dernière a été chiffrée avec la clé secrète de l'émetteur, empreinte qu'on va ensuite comparer avec l'empreinte numérique de l'image déchiffrée. la correspondance de deux empreintes implique que l'intégrité des données reçues est vérifiée, l'expéditeur de l'image authentifié, et une non répudiation de sa part assurée.

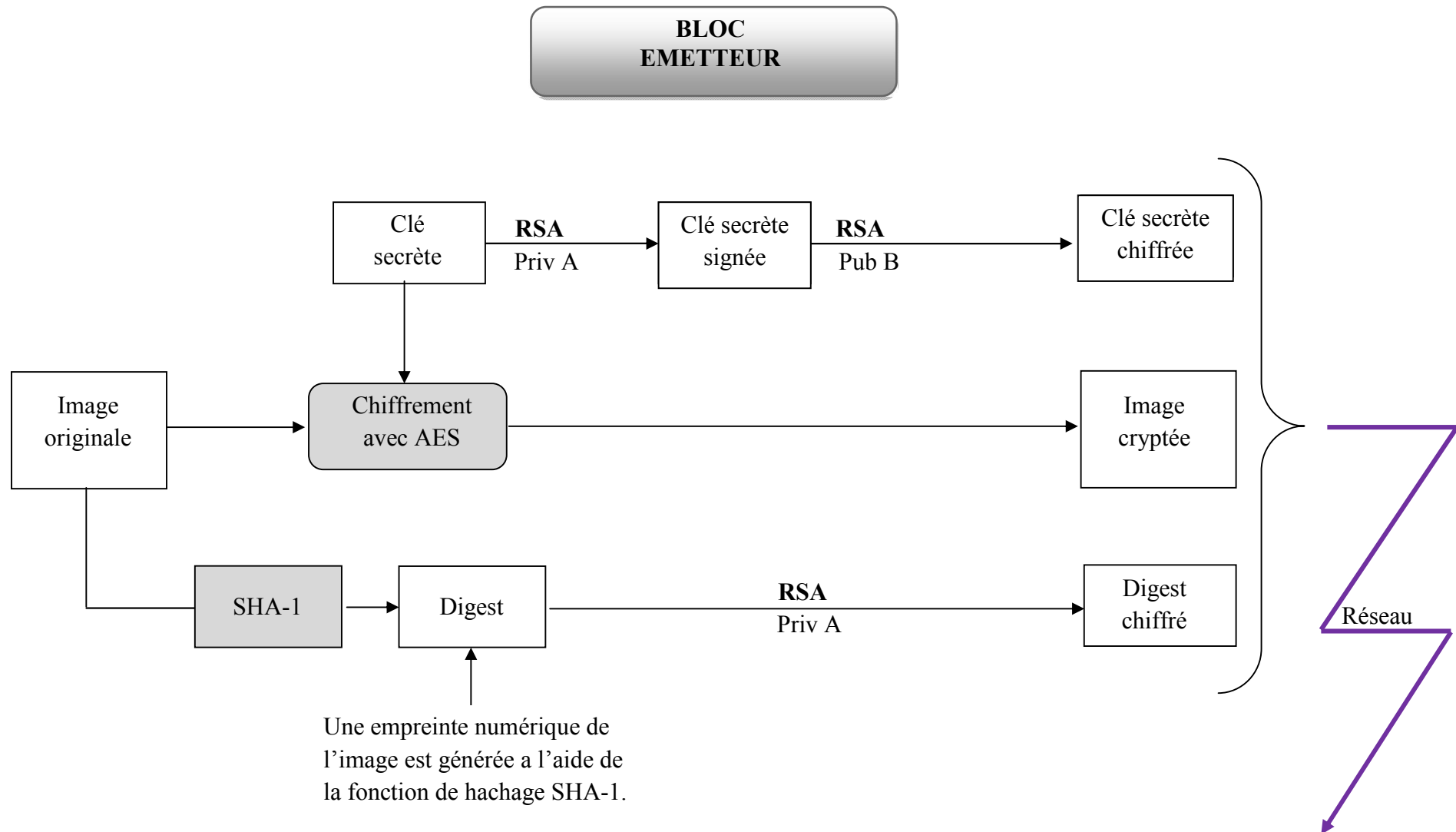


Figure3.1 le cryptosystème hybride de transmission d'images. <<Bloc EMETTEUR>>

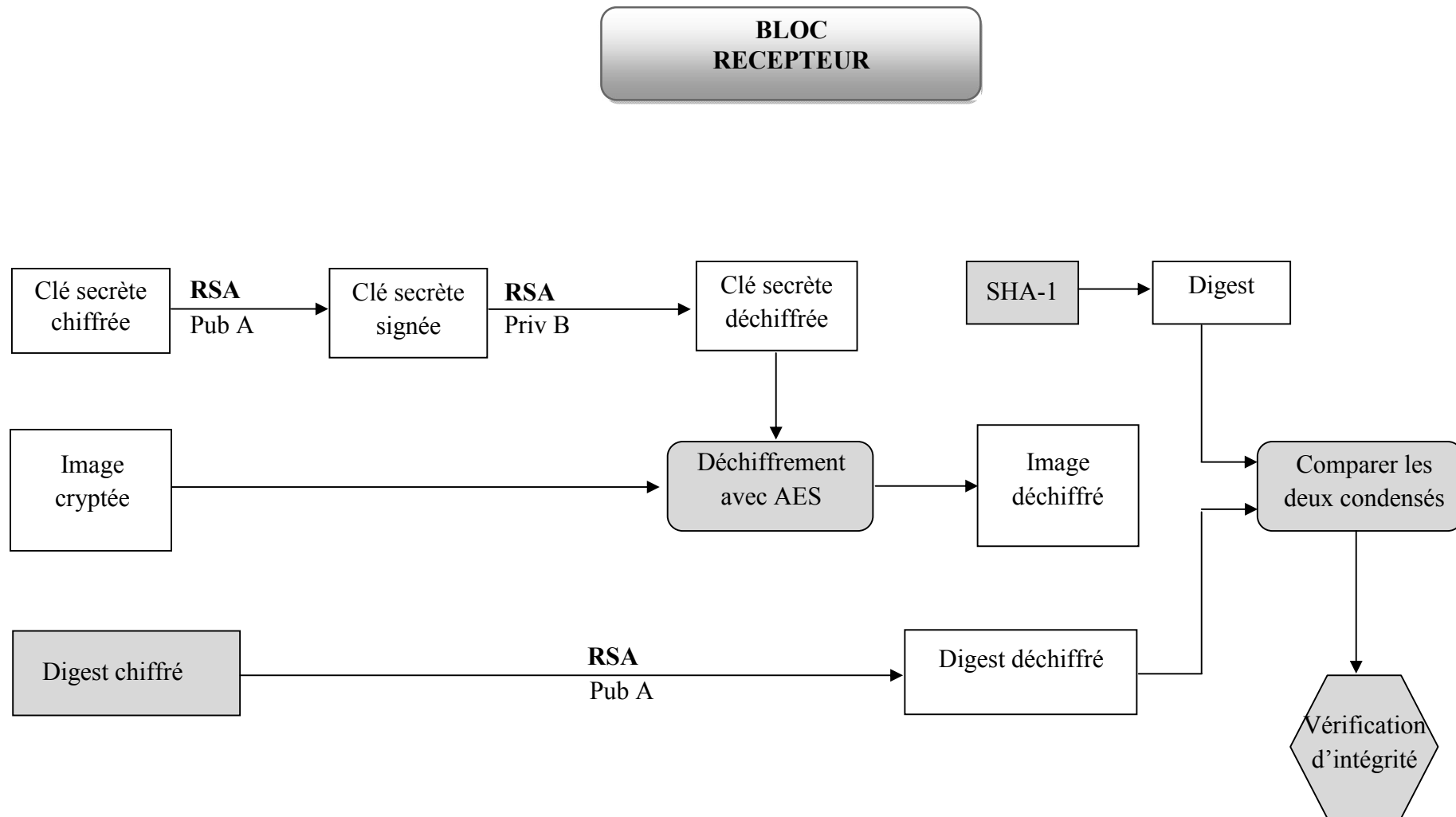


Figure3.2 le cryptosystème hybride de transmission d'images. <<Bloc Récepteur >>

3.3 Mode de chiffrement [7] [15] [17] :

Un algorithme de chiffrement par bloc prend des blocs de longueur fixe de bits b et une clé a pour produire des textes chiffrés de longueur b . Lorsque plusieurs blocs de texte en clair sont cryptés en utilisant la même clé, un certain nombre de problèmes de sécurité se pose. Un mode de fonctionnement est une technique visant à accroître l'effet d'un algorithme de cryptage ou l'adapte pour une application, dans notre cas c'est la transmission des images, Ces modes sont destinés à être utilisés avec n'importe quel algorithme de chiffrement symétrique, y compris AES.

Les modes sont résumés dans le tableau ci-dessous et décrites dans les sections suivantes :

Mode	Description
Le mode « ElectronicCodeBook » (ECB)	Il revient à chiffrer un bloc indépendamment des autres, cela permet entre autre de chiffrer suivant un ordre aléatoire (bases de données, etc.)
Le mode « CipherBlock Chaining » (CBC)	Il permet d'introduire une complexité supplémentaire dans le processus de chiffrement en créant une dépendance entre des blocs successifs. Il effectue un XOR entre un bloc de données en clair et un bloc de données cryptées. Quand au premier bloc il est XORé avec un vecteur appelé vecteur d'initialisation (IV) qui peut être un mot de passe par exemple, ce vecteur change à chaque session, et doit être transmis au destinataire.
Le mode « CipherFeedBack » (CFB)	Le message est ajouté par un XOR à la sortie du bloc chiffré. Le résultat sert d'entrée pour l'étape suivante. Il est utilisé pour le chiffrement par flux.
Le mode « OutputFeedback » (OFB)	Une variante du CFB. La différence ici, c'est que le flux entrant vers les étapes ultérieures est indépendant du message clair.

Tableau 3.1 *les modes de chiffrement*

3.3.1 Le mode « ElectronicCodeBook » :

Le mode le plus simple est le mode Electronic Code Book, dans lequel chaque bloc est chiffré avec la même clé. A la réception, les blocs chiffrés sont déchiffrés avec la même clé comme la figure 3.3 le montre. Le texte clair est divisé en blocs :

$$P_1, P_2, \dots, P_N$$

Les blocs chiffrés correspondants sont :

$$C_1, C_2, \dots, C_N$$

Nous pouvons définir le mode ECB comme suit :

ECB	$C_j = E(K, P_j) \quad J = 1, \dots, N$	$P_j = D(K, C_j) \quad J = 1, \dots, N$	III.1
-----	---	---	-------

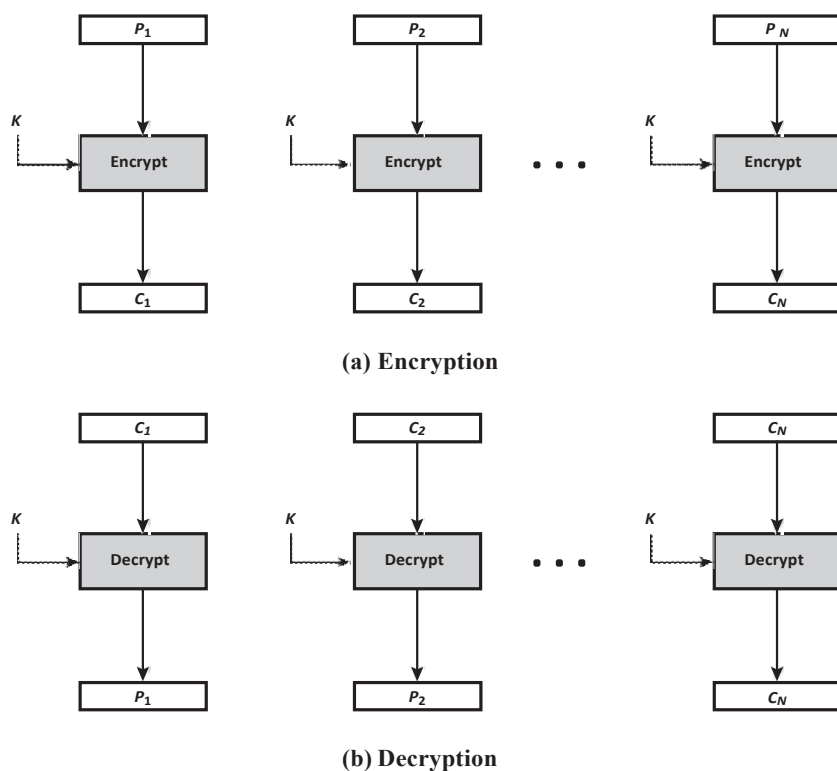


Figure 3.3 .Mode ECB

Le mode ECB est idéal pour la transmission des données courtes, tel qu'un chiffrement clé. Ainsi, si vous souhaitez transmettre une clé DES ou AES en toute sécurité, ECB est le mode

appropriés à utiliser.

La caractéristique la plus importante de ECB est que le même bloc de bits de texte en clair apparaît plus d'une fois dans le message, il produit toujours le même texte chiffré.

Pour de longs messages, le mode ECB est peu sécurisé. il peut être possible pour un cryptographe à exploiter la propriété de mode ECB que les mêmes blocs sont chiffrés de la même manière.

3.3.2 Le mode « Cipher Block Chaining(CBC) »

Pour résoudre le problème de chiffrement des blocs qui se chiffrent de la même manière avec le mode ECB, nous passons au mode CBC .Dans ce cryptosystème, l'entrée à l'algorithme de chiffrement est le « OU exclusif » du bloc actuel de texte en clair et le bloc de texte chiffré précédent; la même clé est utilisée pour chaque bloc.

Pour le déchiffrement, chaque bloc de chiffrement est déchiffré par l'algorithme de déchiffrement.

Le résultat est XORed avec le bloc de texte chiffré précédent pour produire le bloc de texte en clair. Pour bien illustrer le fonctionnement, nous pouvons écrire :

$$C_j = E(K, [C_{j-1} \oplus P_j])$$

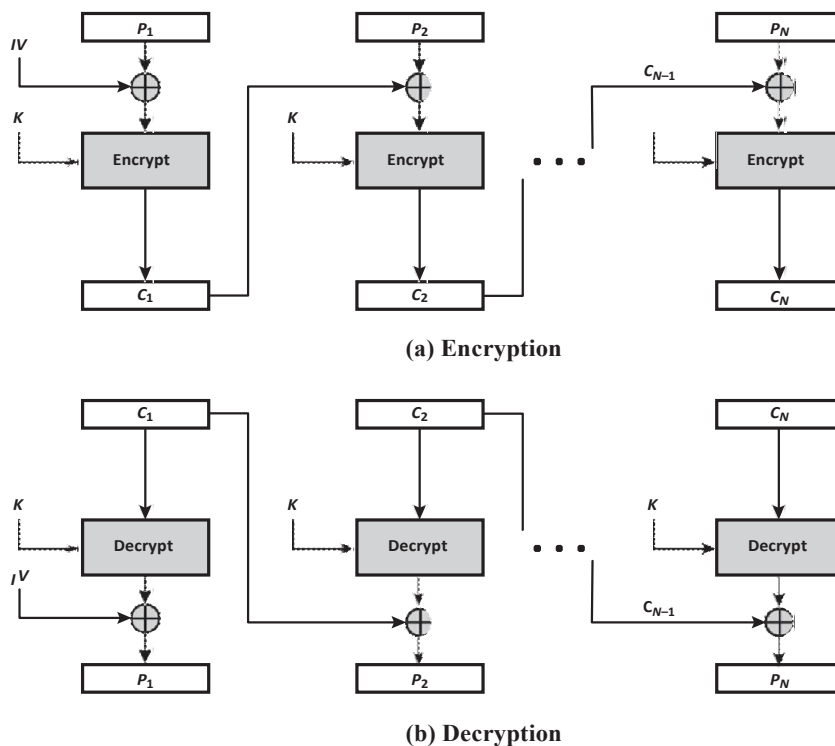


Figure3.4 Mode CBC

Après nous écrivons :

$$D(K, C_j) = D(K, E(K[C_{j-1} \oplus P_j]))$$

$$D(K, C_j) = C_{j-1} \oplus P_j$$

$$C_{j-1} \oplus D(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

Pour produire le premier bloc de texte chiffré, un vecteur d'initialisation (IV) est XORed Avec le premier bloc de texte en clair. Pour le déchiffrement, la IV est XORed avec la sortie de l'algorithme de déchiffrement pour récupérer le premier bloc de texte en clair, L'IV est un bloc de données qui est de la même taille que le bloc de chiffrement. Nous pouvons définir le mode CBC comme de suit :

CBC	$C_1 = E(K, [P_1 \oplus IV])$ $C_j = E(K, [P_j \oplus C_{j-1}]) \quad j = 2, \dots, N$	$P_1 = D(K, C_1) \oplus IV$ $P_j = D(K, C_j) \oplus C_{j-1} \quad j = 2, \dots, N$
-----	--	--

III.2

3.3.3 Le mode CipherFeedBack (CFB) :

Il est possible de convertir un chiffrement par bloc à un chiffrement de flux, en utilisant l'un des deux modes discutés dans cette section et la section suivante :CFB,OFB.

Un flux de chiffrement élimine la nécessité de diviser un message à un nombre entier de blocs. Il peut aussi fonctionner en temps réel. Ainsi, si un flux de caractères est transmis, chaque caractère peut être chiffré et immédiatement transmis.

La figure 3.5 décrit le cryptosystème de la CFB. Sur la figure, on suppose que l'unité de transmission est s bits; une valeur de S est 8. Ici le texte en clair est divisé en segments de S bits

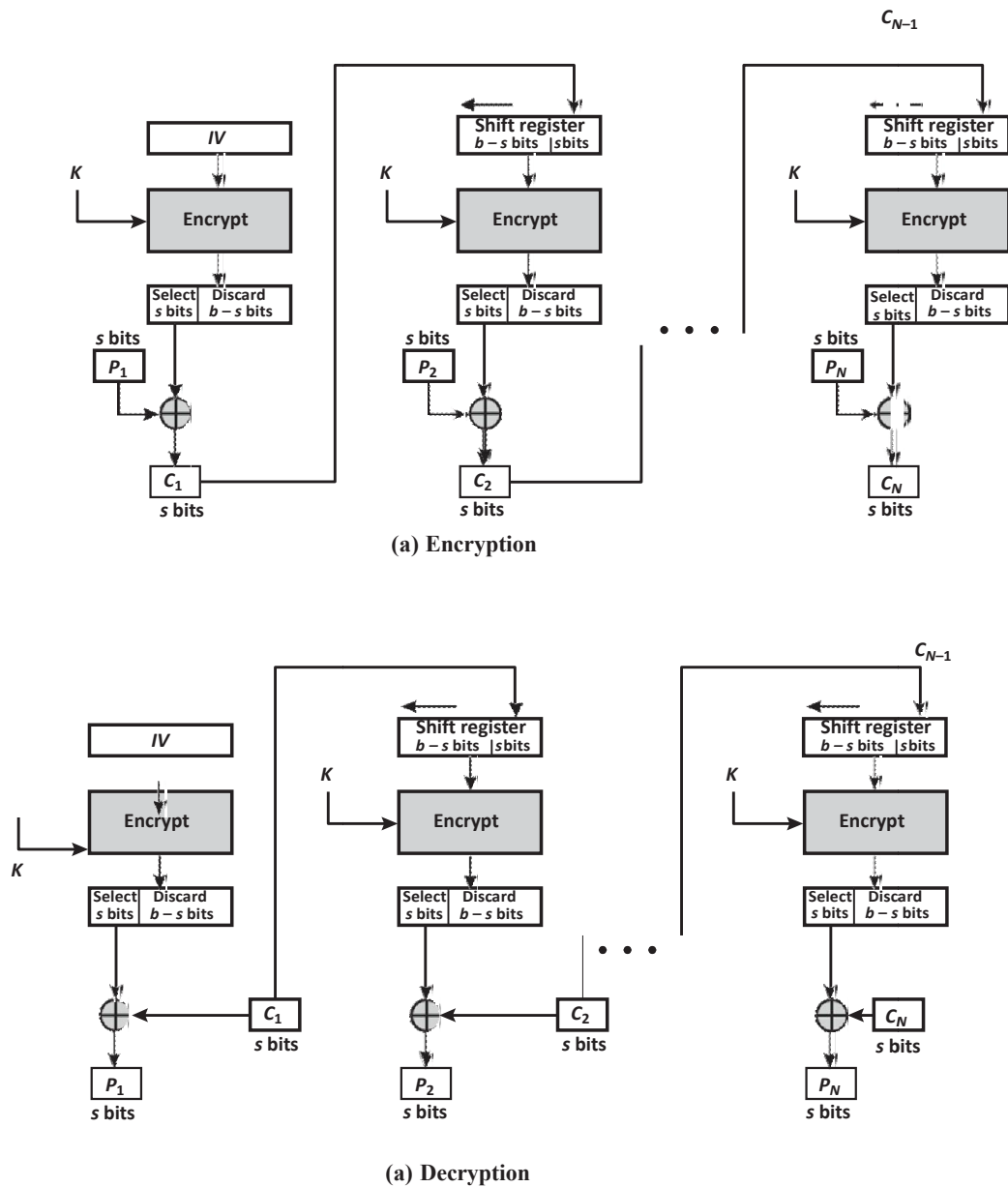


Figure 3.5. Mode CFB

A partir de la figure 3.5 nous pouvons résumer le fonctionnement du mode CFB comme suit :

CFB	$I_1 = IV$	$I_1 = IV$	III.3
	$I_j = LSB_{b-s}(I_{j-1}) // C_{j-1} \quad j = 2, \dots, N$	$I_j = LSB_{b-s}(I_{j-1}) // C_{j-1} \quad j = 1, \dots, N$	
	$I_j = E(K, I_j) \quad j = 1, \dots, N$	$O_j = E(K, I_j) \quad j = 1, \dots, N$	
	$C_j = P_j \oplus MSB_s(O_j) \quad j = 1, \dots, N$	$P_j = C_j \oplus MSB_s(O_j) \quad j = 1, \dots, N$	

3.3.4 Le mode « Output Feedback » (OFB)

La structure du mode OFB est similaire à celle de l'CFB, comme nous pouvons le constater à partir de la figure 3.6. C'est la sortie de la fonction de chiffrement qui est réinjecté au registre à décalage dans le mode OFB, alors que dans le mode CFB, le texte chiffré est celui qui est renvoyé au registre à décalage. L'autre différence est que le mode OFB opère sur des blocs complets de texte en clair et chiffré, et ne pas sur s-bit. Le chiffrement peut s'exprimer comme suit :

$$C_j = P_j \oplus E(K, [C_{j-1} \oplus P_{j-1}])$$

Et pour le déchiffrement on aura :

$$C_j = C_j \oplus E(K, [C_{j-1} \oplus P_{j-1}])$$

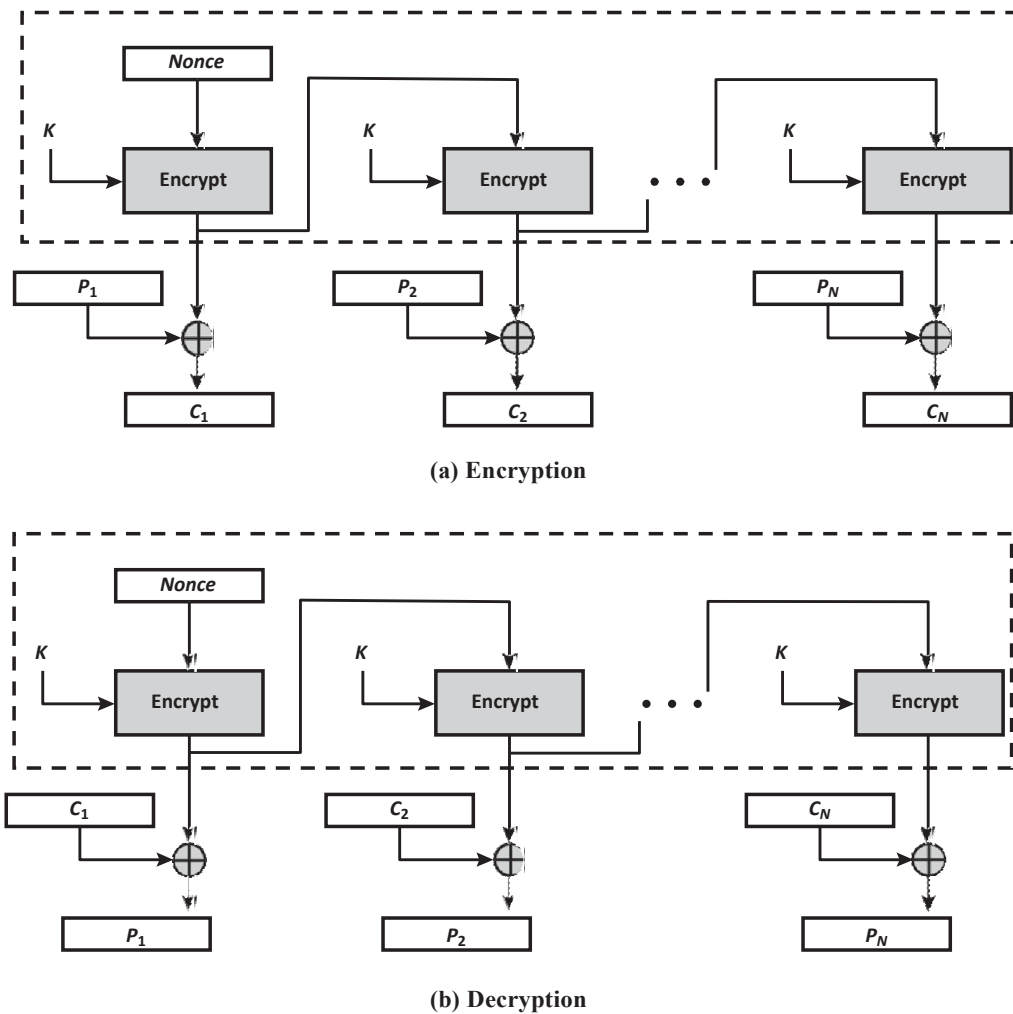


Figure 3.6. Mode OFB

Nous pouvons résumer le fonctionnement du mode OFB dans le tableau suivant :

OFB	$I_1 = \text{Nonce}$	$I_1 = \text{Nonce}$	
	$I_j = O_{j-1} \quad J = 2, \dots, N$	$I_j = O_{j-1} \quad J = 2, \dots, N$	
	$O_j = E(K, I_j) \quad J = 1, \dots, N$	$O_j = E(K, I_j) \quad J = 1, \dots, N$	III.4
	$C_j = P_j \oplus O_j \quad J = 1, \dots, N - 1$	$P_j = C_j \oplus O_j \quad J = 1, \dots, N - 1$	
	$C_N^* = P_N^* \oplus \text{MSB}_u(O_N)$	$P_N^* = C_N^* \oplus \text{MSB}_u(O_N)$	

Comme avec les modes CBC et CFB, le mode OFB nécessite un vecteur d'initialisation. Dans le cas du mode OFB, la IV doit être un nonce; autrement dit, la IV doit être unique pour chaque exécution de l'opération de chiffrement. Un avantage du mode OFB est que l'erreur ne se propage pas, par exemple, si une erreur de bit se produit dans le bloc C_1 , seule la valeur P_1 récupérée après le déchiffrement est affectée.

3.4 Résultats expérimentaux

Dans cette section, nous évaluons la performance de l'algorithme de chiffrement d'images (AES) avec une clé de 128 bits. Nous avons chiffré l'image « lena » de résolution $512 \times 512 \times 8\text{bit}$ pour les différents modes (ECB, CBC, CFB, OFB figure 3.8 et figure 3.9). L'image originale et l'histogramme sont représentés dans la figure 3.7.b. D'après les résultats, nous remarquons que l'image cryptée peut être déchiffrée sans distorsion et les histogrammes des images déchiffrées sont identiques à ceux des images originales. Nous utiliserons aussi les deux modes de chiffrement ECB et CBC pour le chiffrement des images avec peu de détails (figures 3.7.10 ,3.7.11), les figures 3.7.c et 3.7.e représentent les images utilisées.

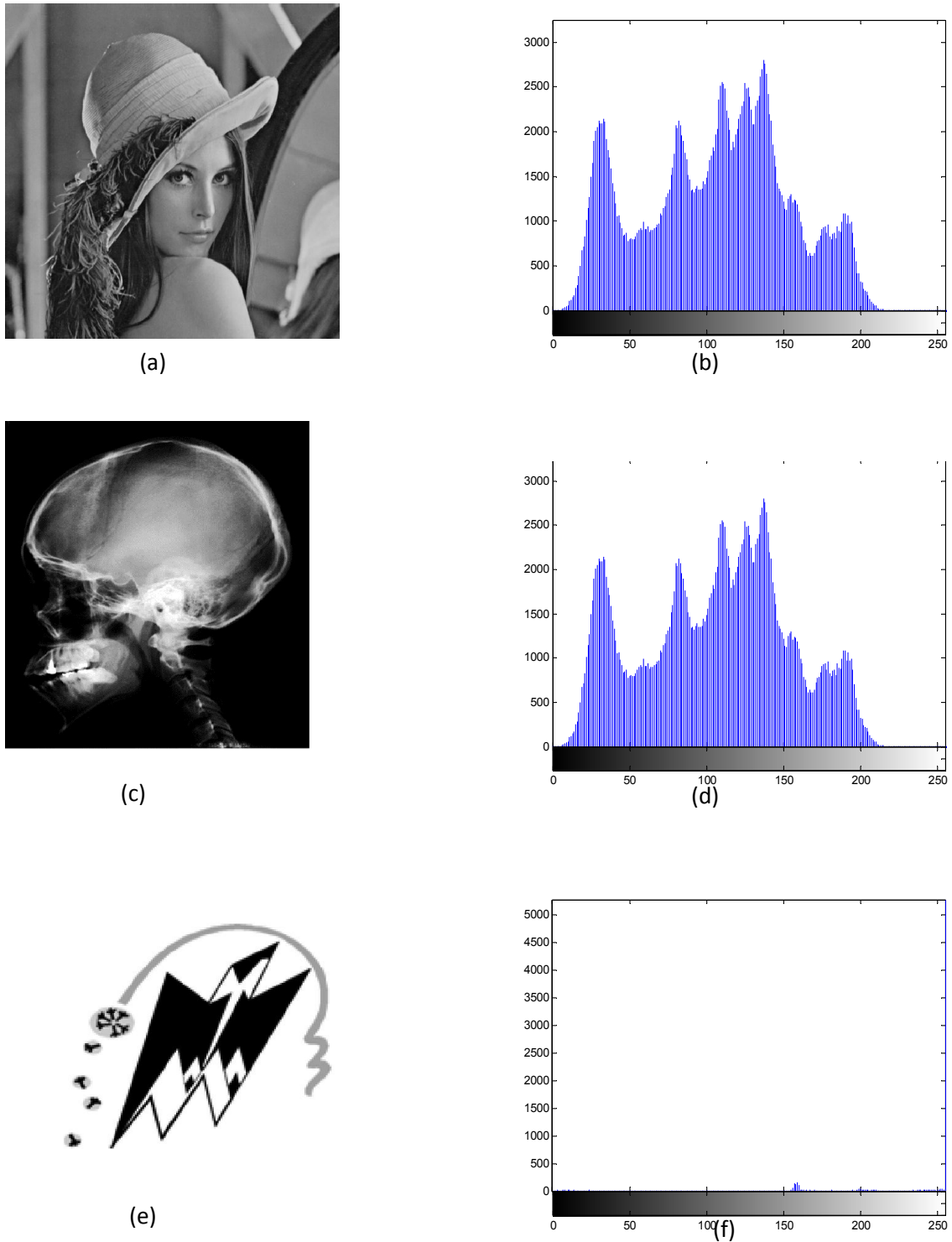


Figure 3.7 image originale (a) lena , (e) médicale et (c) logo ; et leurs histogrammes respectifs en (b), (d) et (f).

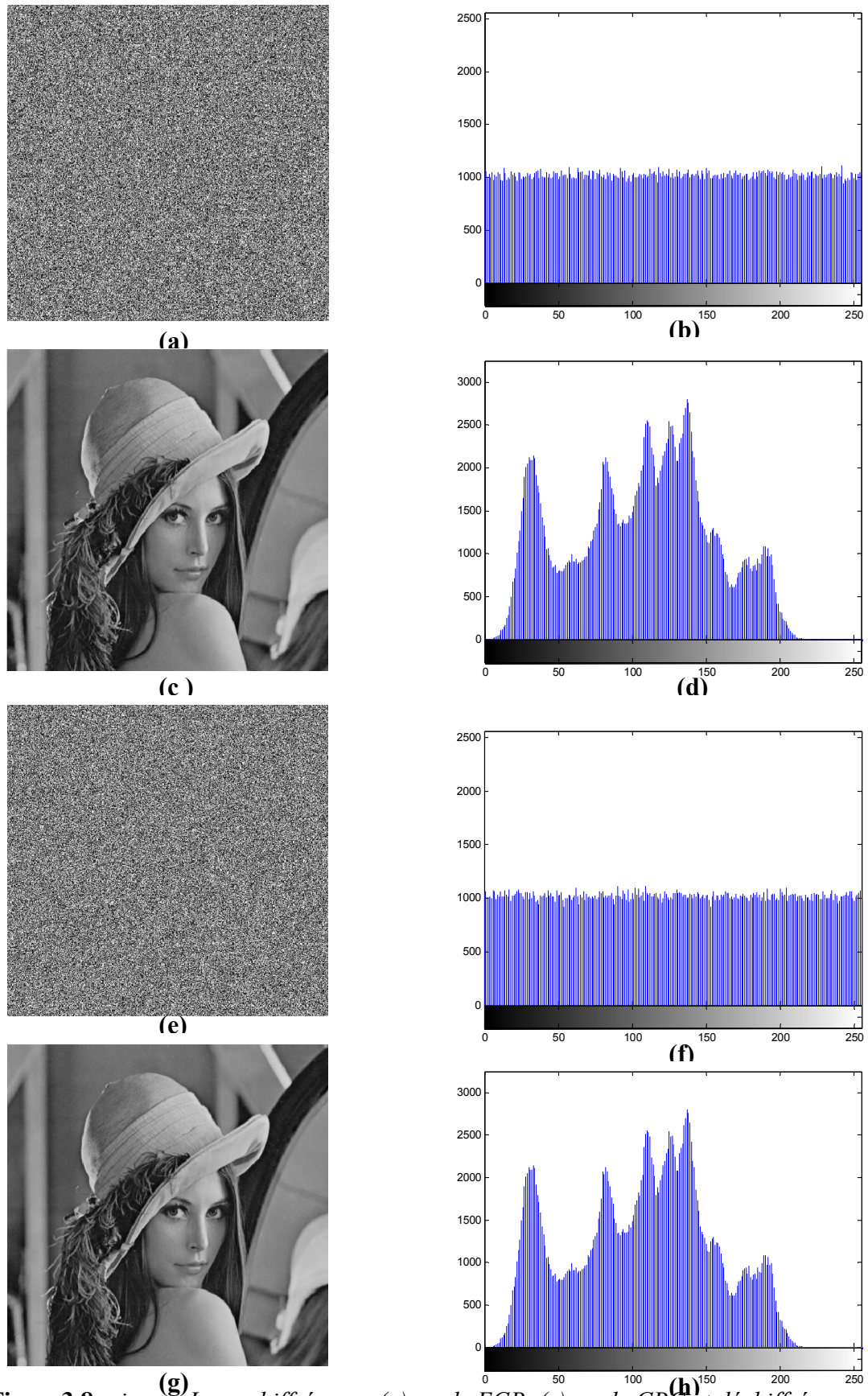


Figure 3.8. image Lena chiffrée en : (a) mode ECB, (e) mode CBC et déchiffrée en : (c) mode ECB, (g) mode CBC avec leurs histogrammes respectifs.

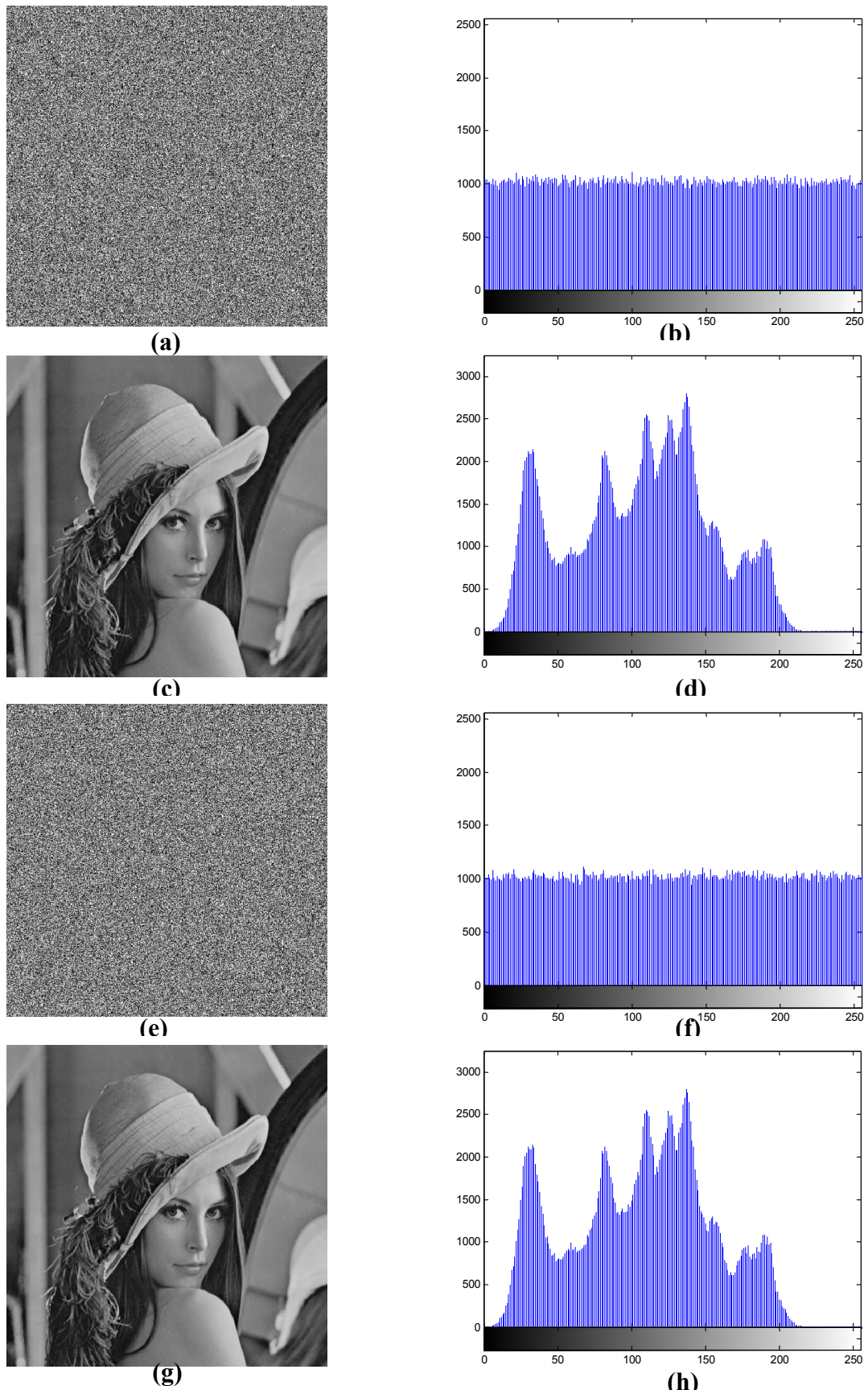


Figure 3.9. image Lena chiffrée en : (a) mode CFB , (e) mode OFB et déchiffrée en : (c) mode CFB , (g) mode OFB avec leurs histogrammes respectifs.

3.5 Performances et analyse de la sécurité du cryptosystème

Pour analyser la robustesse de notre cryptosystème de transmission, la simple inspection visuelle s'avère être insuffisante pour juger le chiffrement d'une image. L'analyse menée dans cette section a alors pour but d'évaluer le degré de chiffrement de notre image.

On commence par une analyse statistique qui va comporter les métriques suivantes :

- Analyse des histogrammes.
- Analyse de la corrélation entre l'image originale et l'image chiffrée.
- Analyse de la corrélation entre les pixels adjacents horizontaux et verticaux.
- Analyse de la sensibilité à la clé secrète
- Analyse différentielles
- Analyse de l'entropie

3.5.1 Analyse statistique :

3.5.1.1 Analyse d'histogramme :

Un histogramme d'image montre de la manière de distribution des pixels dans une image en traçant le nombre de pixels correspondant à chaque intensité de couleur. L'histogramme de l'image chiffrée doit être suffisamment uniforme pour résister à une attaque statistique, et sinon les attaquants peuvent déduire des informations utiles de l'image simple en analysant l'histogramme de l'image cryptée .nous pouvons voir que l'histogramme de l'image chiffrée devient assez uniforme et que nous pouvons facilement en tirer plusieurs remarques :

Avec l'image de la figure 3.7.e chiffrée en mode ECB (figure 3.11.a), nous remarquons clairement que les blocs homogènes dans l'image originale sont chiffrés avec la même manière.

L'histogramme de la figure 3.11.b confirme cette remarque. Cet inconvénient majeur du mode ECB représente une faille que les attaquants peuvent exploiter pour tirer des informations de l'image originale.

L'histogramme des autres modes dans la figure 3.8.f et la figure 3.9.b-f sont uniforme. Ce qui permet de résoudre le problème du mode ECB.

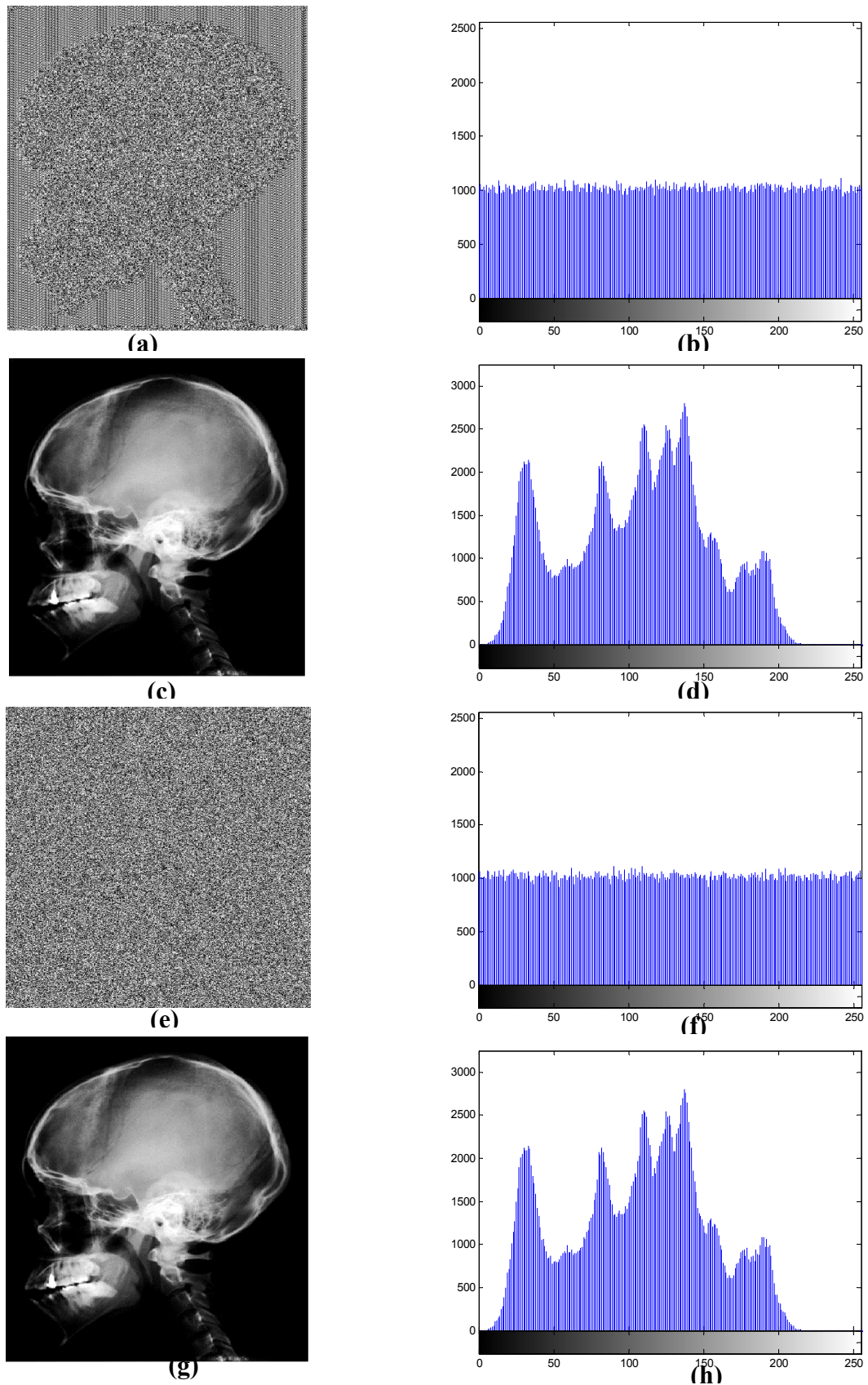


Figure3.10. image Médicale chiffrée en : (a) *mode ECB* ,(e) *mode CBC* et déchiffrée en : (c) *mode ECB* ,(g) *mode CBC* avec leurs histogrammes respectifs.

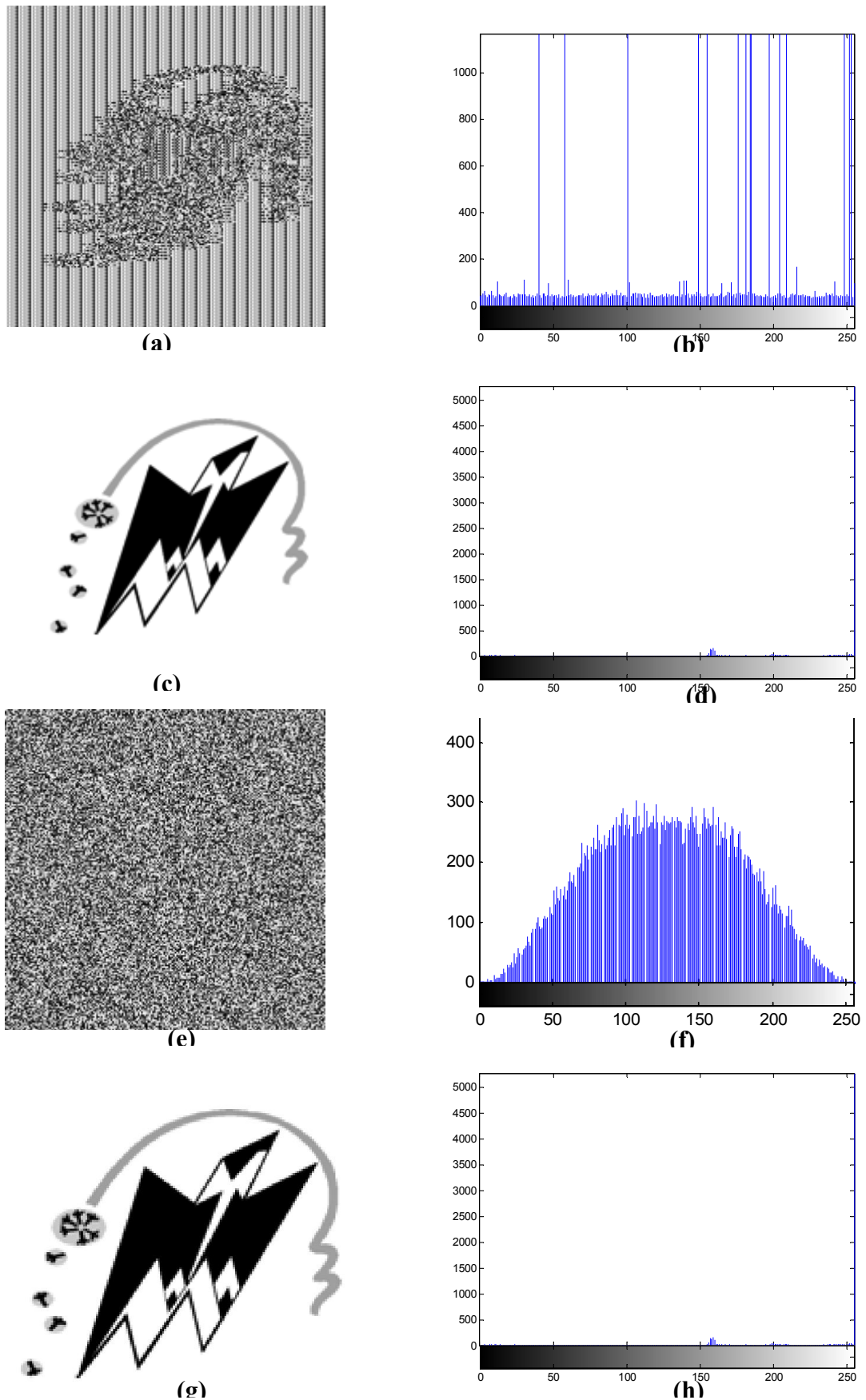


Figure3.11. image Logo chiffrée en : (a) mode ECB, (e) mode CBC et déchiffrée en : (c) mode ECB ,(g) mode CBC avec leurs histogrammes respectifs.

3.5.1.2 Analyse de l'entropie

L'entropie de l'information est une caractéristique importante de l'aléatoire. Sur la base de la théorie de Shannon [18], l'entropie d'une source s est définie comme suit,

$$H(m) = \sum_i p(m_i) \log_2 \frac{1}{p(m_i)} \quad \text{III.5}$$

Où $p(m_i)$ représente la probabilité du symbole (Valeur de pixel) m_i . Théoriquement, un véritable système aléatoire devrait générer 2^8 symboles avec une probabilité égale. Par conséquent, selon l'équation, l'entropie du système est $H(m) = 8$.

Pour l'image "Lena", l'entropie de l'image cryptée (mode CBC) représentée sur la figure 3.4b est 7,9993, qui est proche de 8 et démontre que l'image cryptée est proche d'une image aléatoire. Cela signifie que le système de chiffrement est sécurisé lors de l'attaque de l'entropie. En général plus l'entropie est grande, plus qu'il est difficile de cassé un cryptosystème.

	Lena	Médicale	Logo
Entropie	7.9993	7.9991	7.9959

Tableau 3.2 Résultats de l'analyse de l'entropie

3.5.1.3 Corrélation entre l'image originale et l'image chiffrée

En plus de l'analyse d'histogramme qui est juste un test visuel, nous avons également calculé et analysé la corrélation entre les diverses paire de l'image originale et l'image chiffrée. Les coefficients de corrélation sont calculés comme suit [19]:

$$CC = \frac{\frac{1}{(H,W)} \sum_i^H \sum_j^W (O_{i,j} - \bar{O}) (C_{i,j} - \bar{C})}{\left[\left(\frac{1}{(H,W)} \sum_i^H \sum_j^W (O_{i,j} - \bar{O})^2 \right) \left(\frac{1}{(H,W)} \sum_i^H \sum_j^W (C_{i,j} - \bar{C})^2 \right) \right]^{1/2}}$$

$$\text{Avec : } \bar{O} = \frac{1}{(H,W)} \sum_i^H \sum_j^W (O_{i,j}) \quad \text{et} \quad \bar{C} = \frac{1}{(H,W)} \sum_i^H \sum_j^W (C_{i,j}) \quad \text{III.6}$$

Ici O représente l'image originale, C l'image chiffrée, et $\bar{C}$ et $\bar{O}$ respectivement les valeurs moyennes des éléments des matrices C et O . H et W sont respectivement la hauteur et la largeur de l'image originale et chiffrée.

Les résultats obtenus sont illustrés dans le tableau suivant :

	Lena	Médicale	Logo
ECB	0.0011	-0.0201	0.1743
CBC	-0.0053	0.0008	-0.0028

Tableau 3.3 Coefficient de corrélation entre l'image originale et l'image chiffrée

Il est clair que les coefficients de corrélation de l'image originale et l'image chiffrée dans les deux modes ECB et CBC sont très petits. Par conséquent, il n'y a pas de corrélation entre l'image originale et l'image chiffrée, cette dernière prend alors les caractéristiques d'une image aléatoire.

3.5.1.4 Corrélation entre les pixels adjacents

Pour une image ordinaire, chaque pixel est fortement corrélé avec ses pixels adjacents dans la direction horizontale, verticale et diagonale. Un algorithme de chiffrement idéal devrait produire des images chiffrées dont la corrélation entre les pixels adjacents est négligeable.

Pour tester la corrélation entre les pixels adjacents horizontaux, verticaux ou diagonaux de l'image, on calcul les coefficients de corrélation à l'aide de l'équation suivante [20]:

$$CC_{XY} = \frac{\frac{1}{N} \sum_{i=1}^N (X_i - \bar{X})(Y_i - \bar{Y})}{\left(\frac{1}{N} \sum_{i=1}^N ((X_i - \bar{X})^2 (Y_i - \bar{Y})^2) \right)^{1/2}}$$

$$\text{Avec : } \bar{X} = \frac{1}{N} \sum_{i=1}^N X_i \quad \text{et } \bar{Y} = \frac{1}{N} \sum_{i=1}^N Y_i \quad \text{III.7}$$

Les résultats des coefficients de corrélation calculés des pixels adjacents horizontaux et verticaux pour l'image originale et son chiffré sont donnés les tableaux suivant :

		Horizontale	Verticale	Diagonale
Lena	Image originale	0.9732	0.9865	0.9603
	Image cryptée	0.0023	-0.0017	0.0010
Médicale	Image originale	0.9748	0.9828	0.9651
	Image cryptée	-0.0014	0.0012	0.0041
Logo	Image originale	0.9261	0.9600	0.8757
	Image cryptée	-0.0095	-0.0107	0.0063

Tableau 3.4. *Coefficients de corrélation des pixels adjacents*

Il est clair qu'à partir des résultats des tableaux 3.3 et 3.4 que la corrélation est très petite dans l'image chiffrée. Les distributions de corrélation des pixels adjacents de l'image "Lena" le long de directions horizontales, verticales et diagonales sont présentées dans la figure (3.12.a,c,e) respectivement, et la figure(3.12 b,d,f) montres les distributions correspondantes de l'image cryptée (Résultats pour 2000 paire de pixels sélectionnés d'une façon aléatoire) :

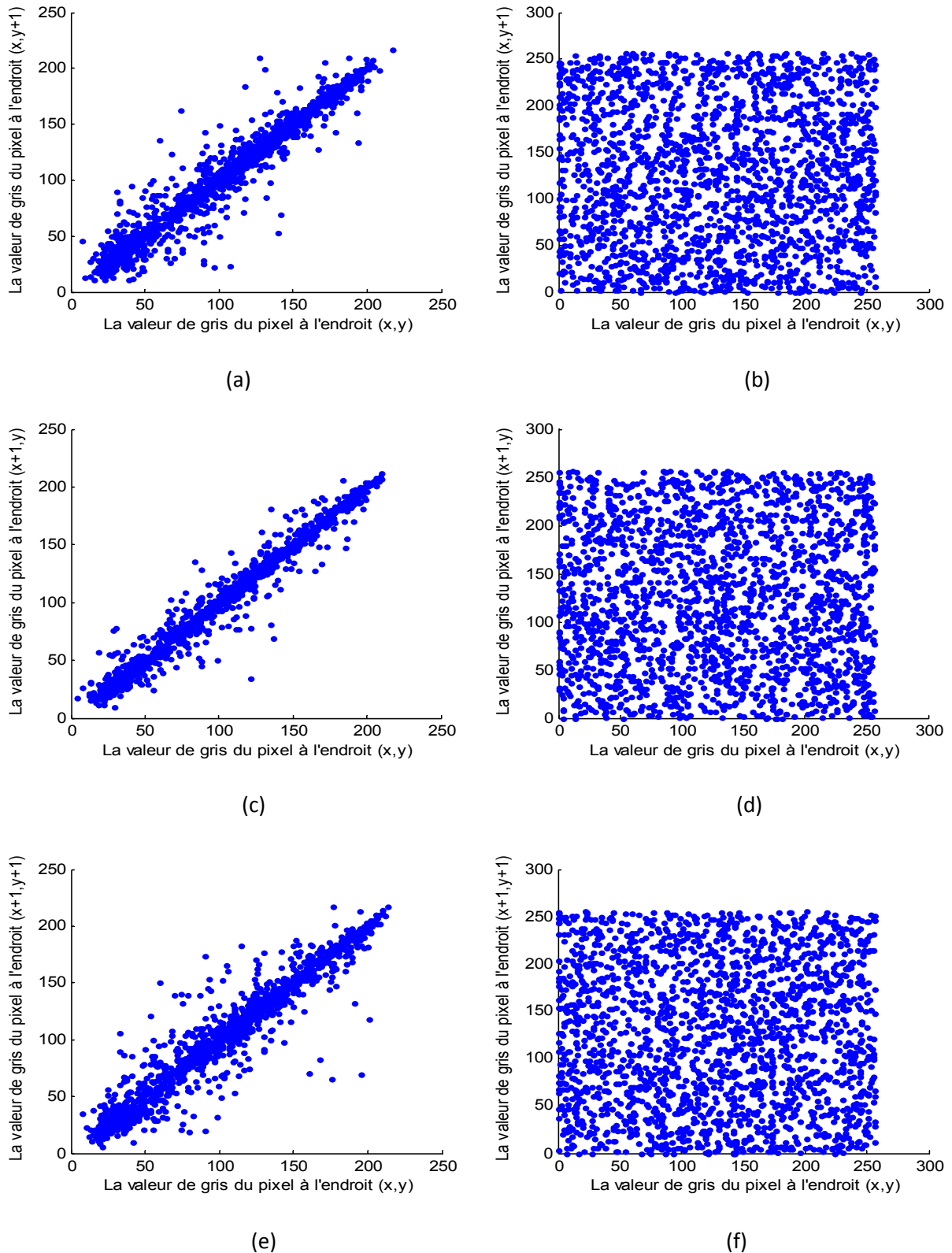


Figure3.12. *Corrélation des pixels adjacents. (a) la distribution de corrélation de l'image originale le long de la direction horizontale; (b) la distribution de corrélation de l'image cryptée dans la direction horizontale; (c) la distribution de corrélation de l'image originale le long de la direction verticale; (d) la distribution de corrélation de l'image cryptée dans la direction verticale; (e) la distribution de corrélation de l'image originale le long de la direction diagonale; (f) la distribution de corrélation de l'image cryptée selon la direction diagonale.*

3.5.1.5 Analyse de la sensibilité à la clé secrète

La sensibilité à la clé secrète est une caractéristique essentielle pour un bon cryptosystème qui garantit la sécurité de ce dernier contre toute attaque exhaustive.

On effectue deux testes pour observer la sensibilité à la clé secrète de notre cryptosystème :

1. **Test 1** : on utilise deux clés qui diffèrent d'un seul bit pour chiffrer la même image, les deux images chiffrées produites devraient être complètement indépendante entre elles ou en d'autres termes elles devraient posséder une corrélation négligeable.
2. **Test 2** : l'image chiffrée ne peut être déchiffrée correctement malgré la présence d'une légère différence (différence d'un seul bit) entre la clé de chiffrement et de déchiffrement.

- **Test 1 :**

Pour ce test, une image est chiffrée avec deux clés qui diffèrent d'un seul bit, on calcule ensuite le coefficient de corrélation entre les deux images chiffrées (Mode CBC) :

Corrélation entre les deux images chiffrées	
Lena	0.0006
Médicale	-0.0013
Logo	0.0031

Tableau 3.5 coefficient de corrélation entre les deux images chiffrées

- **Test 2 :**

Pour ce test une image est chiffrée avec une clé de chiffrement, on essaye ensuite cette même image avec une clé de déchiffrement qui diffère de la clé de chiffrement d'un seul bit. Les résultats sont sur la figure 3.13

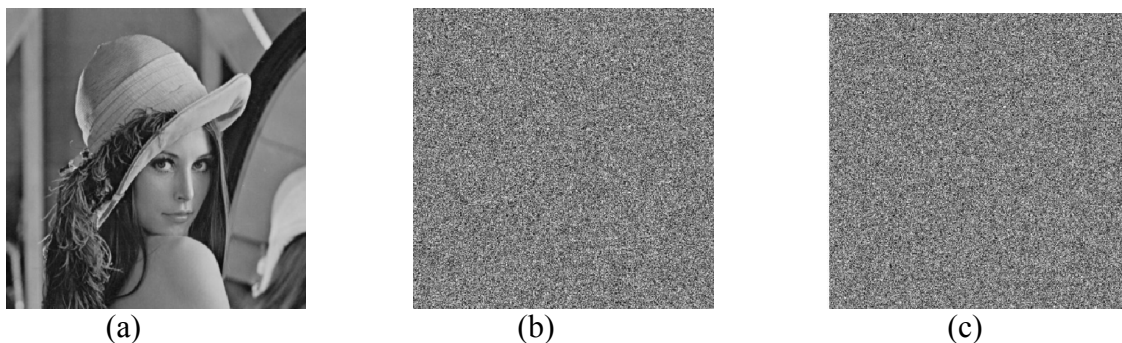


Figure 3.13 analyse de la sensibilité à la clé secrète. (a) image originale; (b) image chiffrée; (c) image déchiffrée avec une clé différente.

D'après la figure 3.13, on constate que l'algorithme n'a pas pu déchiffrer correctement l'image.

On calcule ensuite le coefficient de corrélation entre l'image originale et l'image déchiffrée. $CC = -0,0082$.

On constate à partir du coefficient calculé entre les deux images une forte dé-corrélation. On peut bien conclure alors que le cryptosystème conçu est très sensible à tout changement dans la clé secrète.

3.5.2 Analyse différentielle

Par ce test, on met en évidence la sensibilité de l'image chiffrée par rapport à l'image originale. Pour cela, on chiffre deux images (I1 et I2) qui diffèrent d'un seul pixel. Les deux images chiffrées obtenues sont notées (C1 et C2). Les deux paramètres NPCR (Number Of Pixels Change Rate) et UACI (Unified Average Changing Intensity), sont des mesures que nous allons utiliser pour quantifier la différence entre deux images.

NPCR représente le taux des pixels changés dans l'image chiffrée lors du changement d'un seul pixel dans l'image originale. Il est défini par cette formule [21]

$$\text{NPCR} = \frac{\sum_{i,j} D(i,j)}{W \times H} \times 100\% \quad \text{III.8}$$

Avec

$$D(i,j) = \begin{cases} 0 & C1(i,j) = C2(i,j) \\ 1 & C1(i,j) \neq C2(i,j) \end{cases}$$

UACI est défini par l'équation suivante :

$$\text{UACI} = \frac{1}{W \times H} \left[\sum_{i,j} \frac{|C1(i,j) - C2(i,j)|}{255} \right] \times 100\% \quad \text{III.9}$$

Le tableau montre les résultats obtenus avec un changement de la valeur d'un seul pixel à la position (5,5) des images originales

	Lena	Médicale	Logo
NPCR	0.9884	0.9962	0.9961
UACI	0.3318	0.4030	0.4873

Tableau 3.6. NPCR et UACI des images cryptées

Il est très clair que les résultats obtenus sont assez satisfaisants, et restent dans la gamme des valeurs espérées, c'est-à-dire que notre cryptosystème est extrêmement sensible au texte clair.

En somme, on peut dire que notre cryptosystème résiste bien à l'attaque différentielle.

3.5.2 Contrôle d'intégrité

Pour ce test, nous avons chiffré notre image avec le mode ECB, ensuite une petite région de l'image cryptée a été copiée et collée sur une autre zone de l'image. Après décryptage, il est possible de visualiser les images mais il n'est pas possible de garantir l'intégrité comme illustrée en figure 3.14

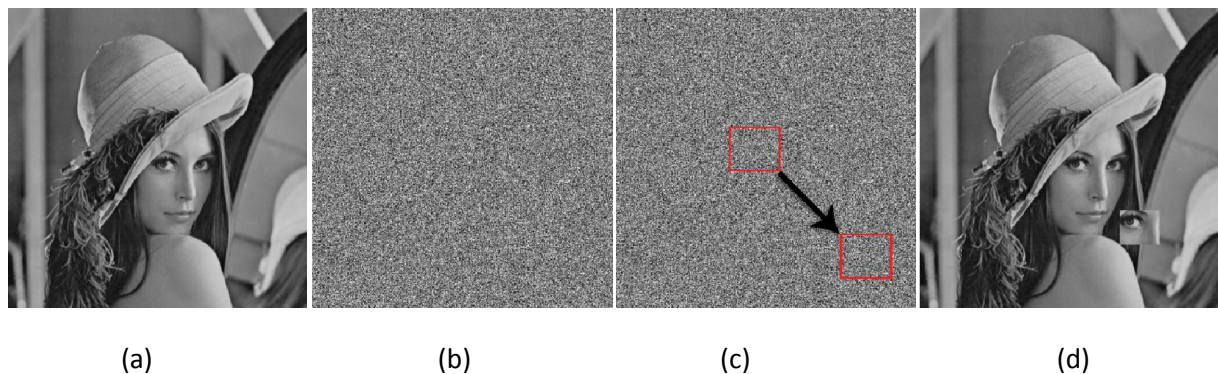


Figure 3.14. (a) Image originale de Lena. (b) Image cryptée. (c) Copie d'une région de l'image cryptée et collage sur une autre zone. (d) Décryptage de l'image après modification

Afin de résoudre ce problème d'intégrité, on a calculé l'empreinte de l'image originale et celle de l'image déchiffrée à l'aide de la fonction de hachage SHA-1. Le résultat est dans le tableau ci-dessous :

Empreinte de l'image originale	Empreinte l'image déchiffrée
7ae1cb9ab6	e7ab5dc73f
a6514d9181	8796dad48c
e499496ce4	561ed64b06
2a4977ec2c	7b4cc31344

Tableau3.7. Empreinte de l'image originale à l'émission et l'empreinte de l'image déchiffrée à la réception

Nous constatons que toute modification de l'image chiffrée dans le canal de transmission, induit un changement complet de l'empreinte numérique de l' image déchiffrée par rapport à l'empreinte de l' image originale. On conclue alors que le problème de l'intégrité n'est plus posé dans notre cryptosystème.

3.6 Conclusion :

Dans ce chapitre, nous avons détaillé les quatre modes de chiffrement, et nous avons présenté notre cryptosystème de transmission. Ce dernier est divisé en deux blocs : un bloc d'émission et un autre de réception. Ces deux derniers sont reliés à travers un canal de transmission. Dans le premier bloc, nous avons chiffré les images avec l'algorithme de AES et une clé définie par le client, puis nous avons chiffré cette clé avec l'algorithme RSA deux fois successives pour assurer une transmission sécurisée de cette dernière, et garantir l'authenticité de l'émetteur, ainsi la fonction SHA-1 vérifie l'intégrité des données transmises .

Dans le bloc de réception des fonctions inverses, sont appliquées à l'image chiffrée pour la déchiffrer. Après le déchiffrement, nous avons pu vérifier l'intégrité et assuré que l'image n'a pas changé dans le canal de transmission.

Les résultats obtenus pour l'application de notre cryptosystème hybride pour la transmission des images; Pour le plan de confidentialité, les quatre modes avec AES ont garanti cette fonction sauf avec le mode ECB qui a présenté un inconvénient pour les zones homogènes et les histogrammes ont confirmé cet inconvénient.

Nous sommes aussi satisfaits pour les résultats obtenus pour la sensibilité des clés, les images sont complètement décorréélées malgré que nous n'ayons changé qu'un seul bit de la clé de session !

4.1 Introduction

Dans ce chapitre, nous présentons notre application qui permet de chiffrer des images reparties selon le modèle client/serveur. Pour sécuriser le transfert des images sur un réseau local entre deux ordinateurs.

L'application de chiffrement a été développée par IDE Netbeans version 6. Notre application est basée sur le modèle client / serveur.

4.2 Présentation de l'application

4.2.1. Outils et technologie à utiliser

Un langage de programmation Java est utilisé pour développer l'application. un avantage majeur de l'utilisation de Java est sa portabilité où il peut être exécuté sur une plate-forme / système d'exploitation dont Java Runtime Environment (JRE) est installé [22]. Cela signifie que le programmeur n'a pas besoin de mettre les efforts sur le développement d'une version différente de logiciels pour chaque plate-forme. Netbeans IDE 6.7.1 sera utilisé pour développer une interface utilisateur graphique (GUI) [23] pour l'application et la fonctionnalité de l'application qui est le chiffrement et le déchiffrement d'un fichier image.

- Class utilisées : JCE, JCA, Socket, Bufferedimage.

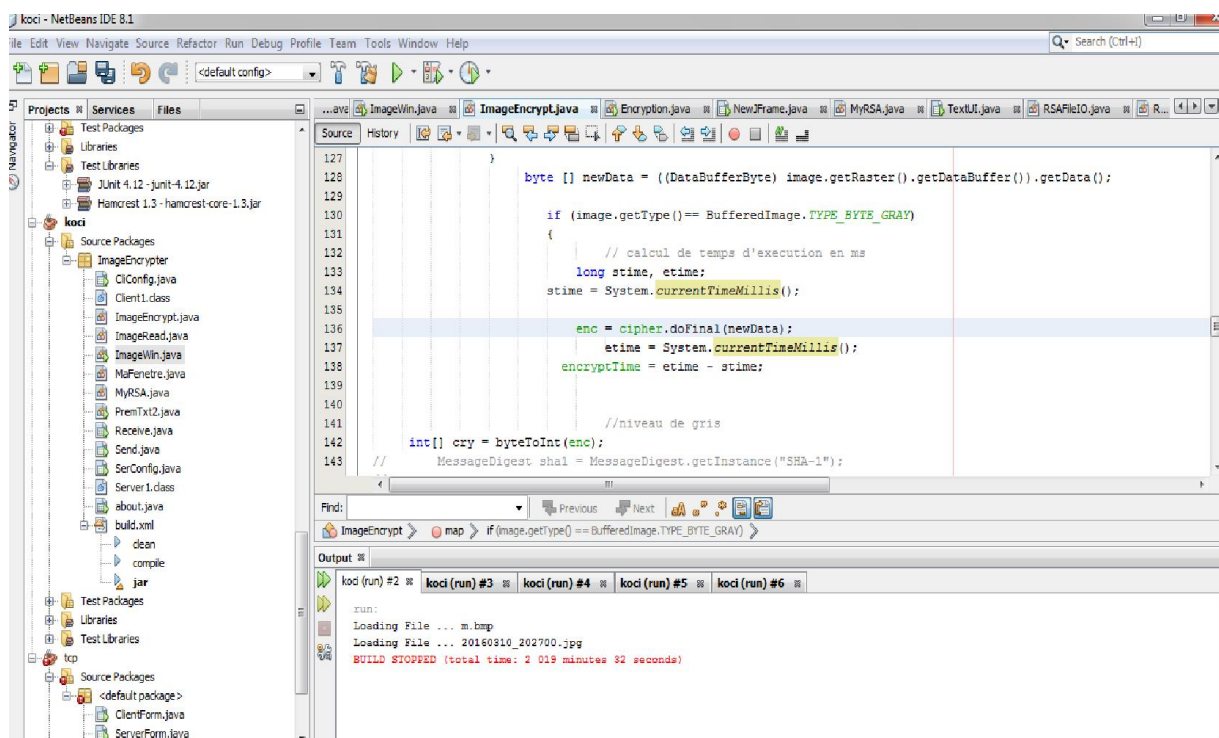


Figure 4.1 Interface de Netbeans IDE

4.2.2. L'interface de l'application

4.2.2.1. Client

La fenêtre d'interface de notre client est présentée ci-dessus par la figure :

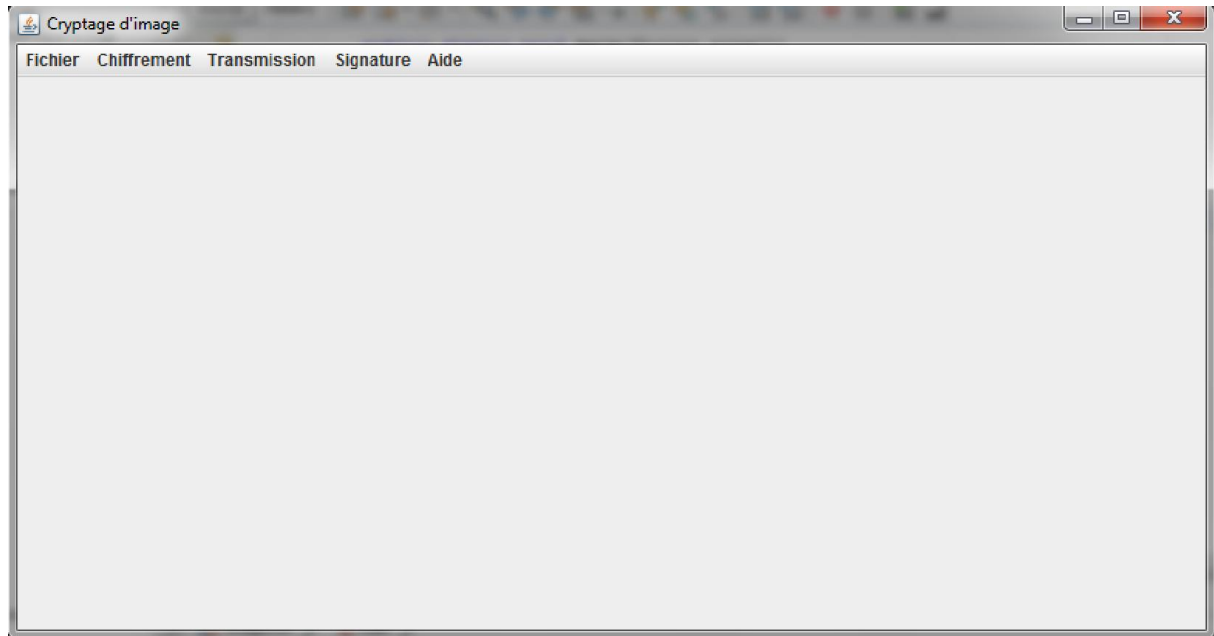
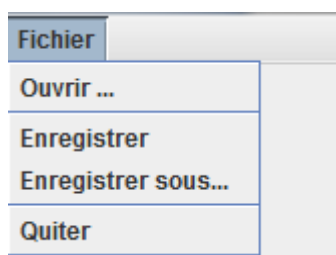


Figure 4.2. Interface de client

4.2.2.1.1 Les menus et les sous menus :

a- Le menu « Fichier » :

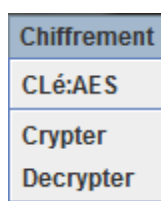


Ouvrir : permet l'ouverture d'une image (BMP, JPG, PNG...)

Enregistrer (Enregistrer sous) : permet de sauvegarder l'image courante.

Quitter : Permet de quitter l'application

b- Le menu « Chiffrement » :



Le menu « Chiffrement » est réservé au chiffrement/Déchiffrement de l'image

Clé :AES : permet de définir la clé AES de taille 128 bits (16 bytes).

Crypter : Permet de chiffrer l'image avec la clé secret AES en mode CBC

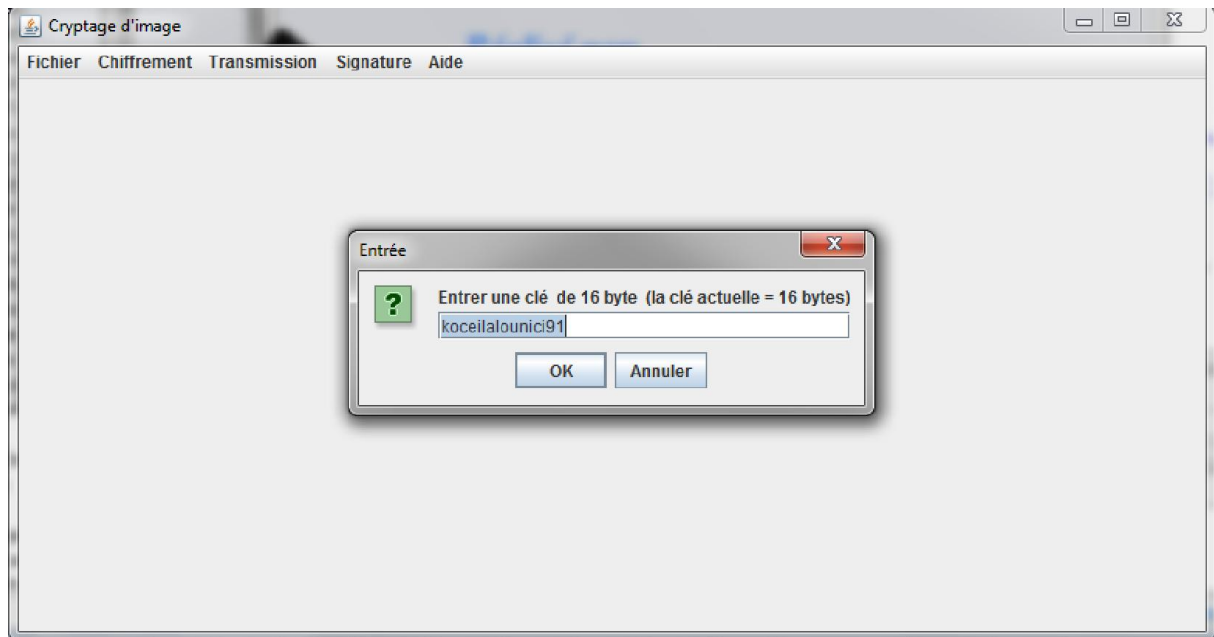
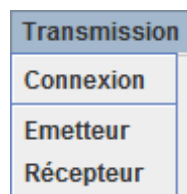


Figure4.3. *fenêtre de saisie la clé secrète de l’algorithme AES*

La fenêtre illustrée en Figure 4.3. Permet de définir la clé secrète de l’algorithme AES de taille de 128 bits nécessaire au chiffrement et au déchiffrement d’une image.

c- Le menu « transmission » :



Le menu « **Transmission** » est réservé à la connexion entre le client/serveur et à l’envoi des images cryptées au serveur

Connexion (client): permet d’établir une connexion entre le client/serveur et l’échange de clés publiques générées par RSA.

- La première étape : générer une paire de clé par l’algorithme RSA (clé publique et clé privée).
- La deuxième étape : l’échange de la clé publique au travers d’un canal non sécurisé.

Emetteur : Permet d’envoyer l’image cryptée et la clé secrète pour déchiffrement .

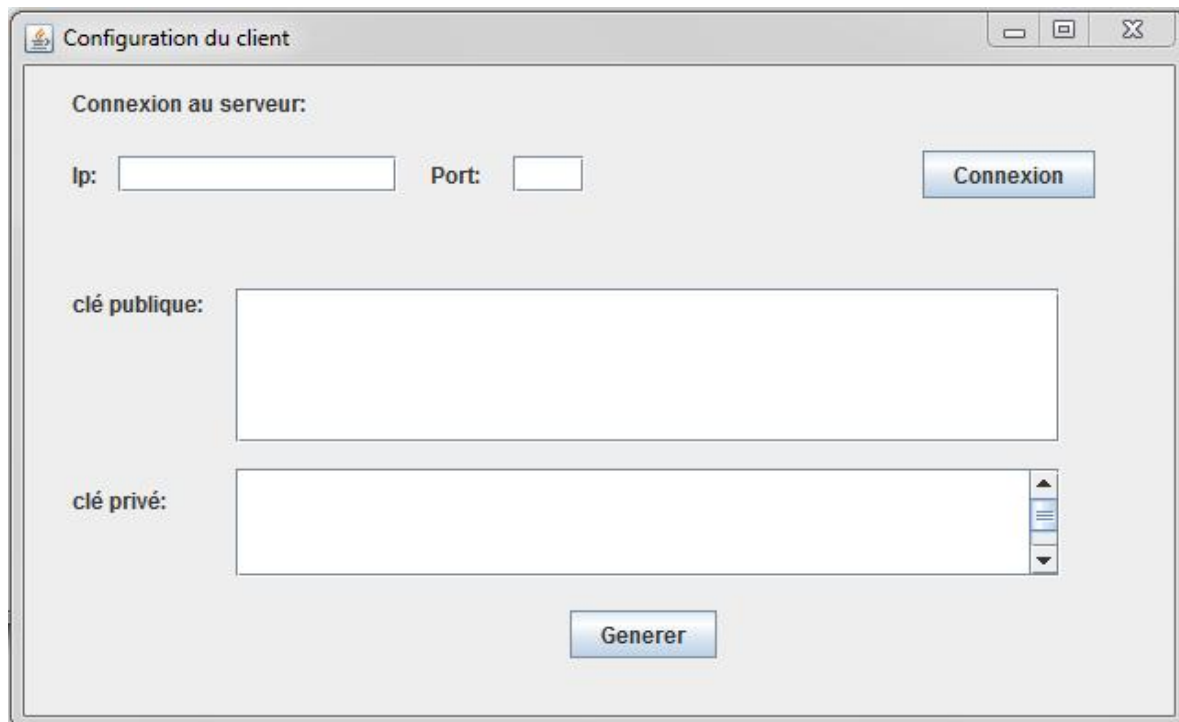


Figure4.4 *Fenêtre des paramètres du Client*

La fenêtre illustrée en figure4.4 permet au client de générer une paire de clé avec RSA (clé publique, clé privée) , et d'établir une connexion avec le serveur afin de partager la clé publique.

Les boutons **Générer** et **Connexion** permettent respectivement de :

- Générer une paire de clés RSA .
- échange de clés publique entre les deux parties en définissons l'adresse IP du et le port du serveur.

d- Le sous menu Emetteur :

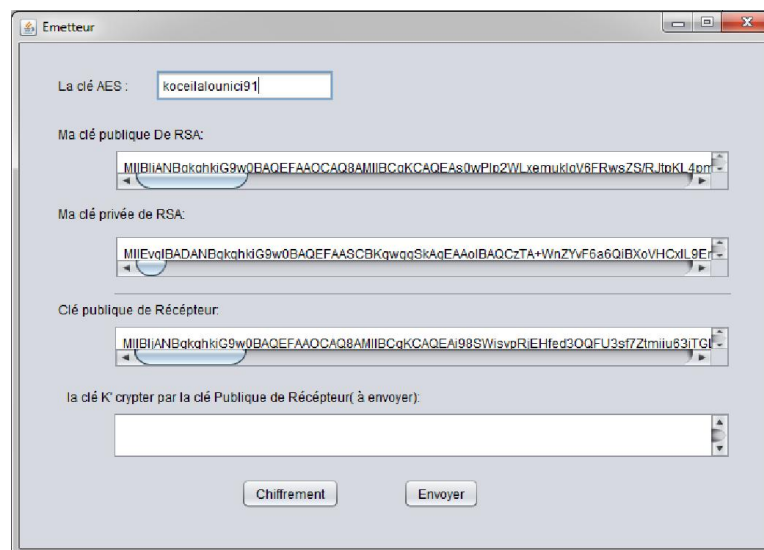


Figure 4.5 *fenêtre de transmission le tout {image chiffrée+K'' la clé chiffrée+empreinte chiffrée}*

- **Le bouton envoyer** : Transmission à travers le réseau le tout : image chiffrée+ K' la clé chiffrée + empreinte chiffrée de l'image cryptée.

4.2.2.2. Serveur :

La fenêtre d'interface de notre serveur est présentée ci-dessous :

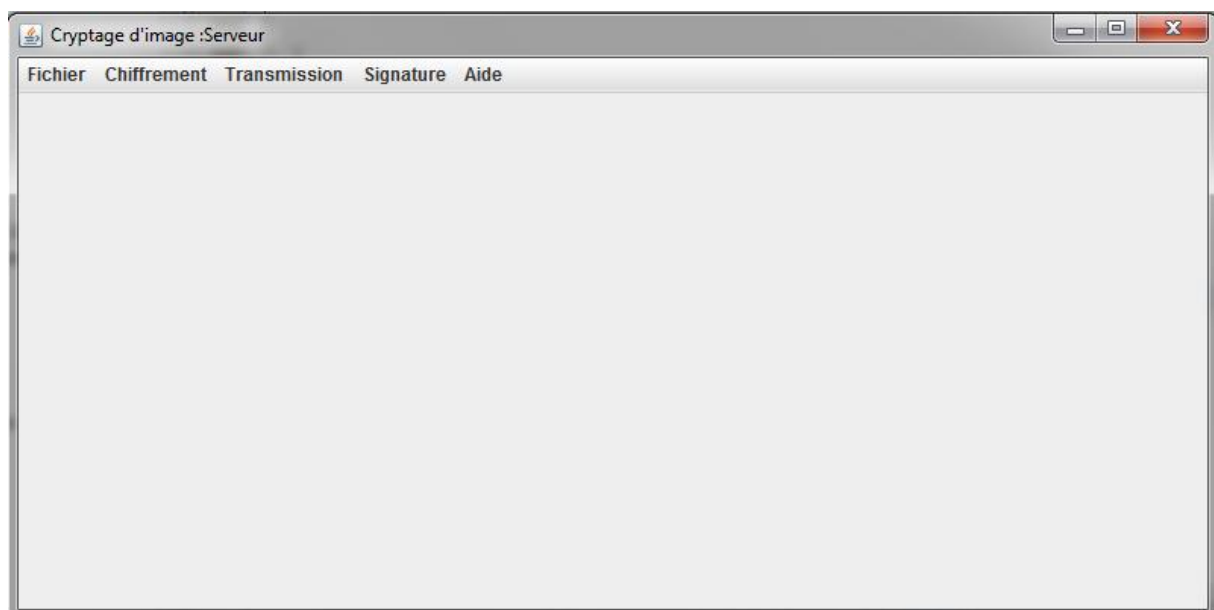
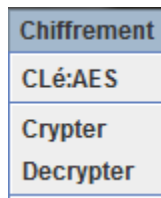


Figure 4.6. *Interface de Serveur*

a- Le menu « Déchiffrement » :



Le menu « **Déchiffrement** » est réservé au Déchiffrement de l'image

Clé : AES : permet de définir la clé AES de taille 128 bits (16 bytes).

Décrypter : Permet de déchiffrer l'image avec la clé secret AES en mode CBC .

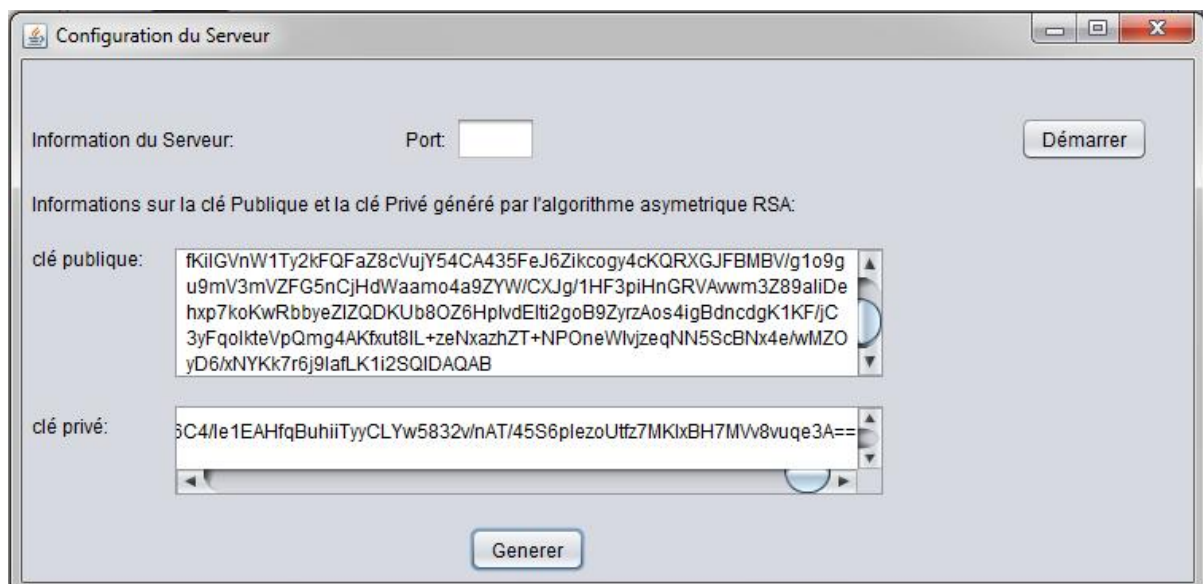


Figure4.7. Fenêtre des paramètres du Serveur

La fenêtre illustrée en figure4.7 permet au serveur de générer une paire de clés avec RSA (clé publique, clé privée) et lancer une écoute au port spécifié de la part de client afin d'établir une connexion avec le client et partage de la clé publique.

Les boutons **Générer** et **Démarrer** permettent respectivement de :

- Générer une paire de clés RSA (clé publique, clé privée).
- échange de clés publique entre les deux parties en définissons le port du connexion.

Le sous menu Récepteur :

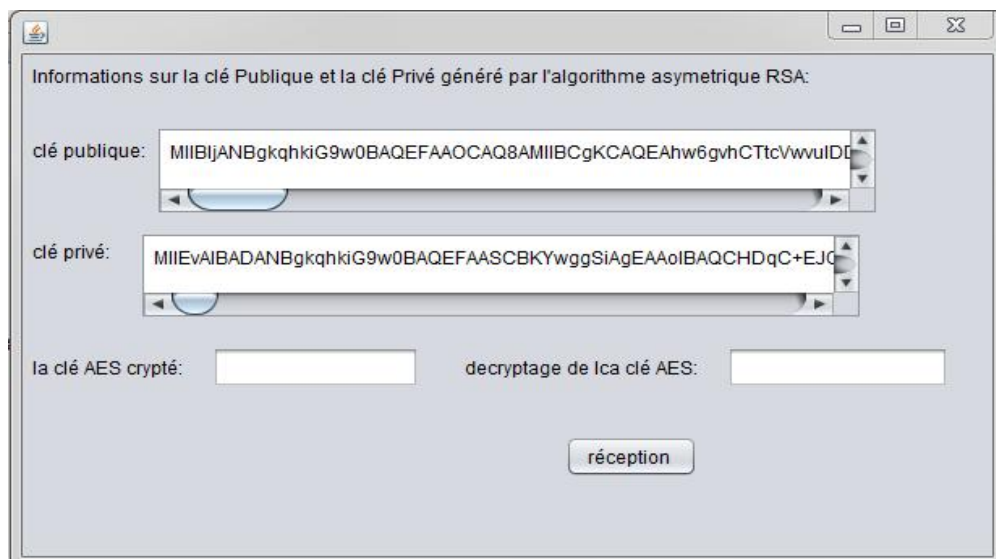


Figure4.8. Fenêtre de réception de tout {image chiffrée+K'' la clé chiffrée+empreinte chiffrée}

- Réception de l'image cryptée et la chargée en mémoire
- Déchiffrement de la Clé AES cryptée en utilisons la clé privée du serveur.
- Reconstitution et affiche de l'image originale.

Le menu aide:



Figure4.9. A propos : affiche la fenêtre donnant les informations générales sur le logiciel

4.3. Tests et Résultats :

Nous présentons dans cette section quelques tests et résultats pratiques , que nous allons comparer aux résultats théoriques obtenus au chapitre précédent .

4.3.1. Capture des trames sur le réseau

Pour capturer les trames sur le réseau en live, nous allons utiliser le logiciel Wireshark, Wireshark est un logiciel d'analyse réseau (sniffer) qui permet de visualiser l'ensemble des données transitant sur la machine qui l'exécute, et d'obtenir des informations sur les protocoles applicatifs utilisés. Les octets sont capturés en utilisant la bibliothèque réseau PCAP, puis regroupés en blocs d'informations et analysés par le logiciel.

Test01 : Transmission de la clé secrète a travers le réseau :

Le plus gros problème avec le cryptage à clé symétrique c'est l'envoi de la clé secrète a travers un réseau non sécurisé .

Dans cet exemple ,nous allons envoyer la clé secrète en text clair (Figure4.9) , et qu'elle est interceptée par un attaquant(figure4.10).

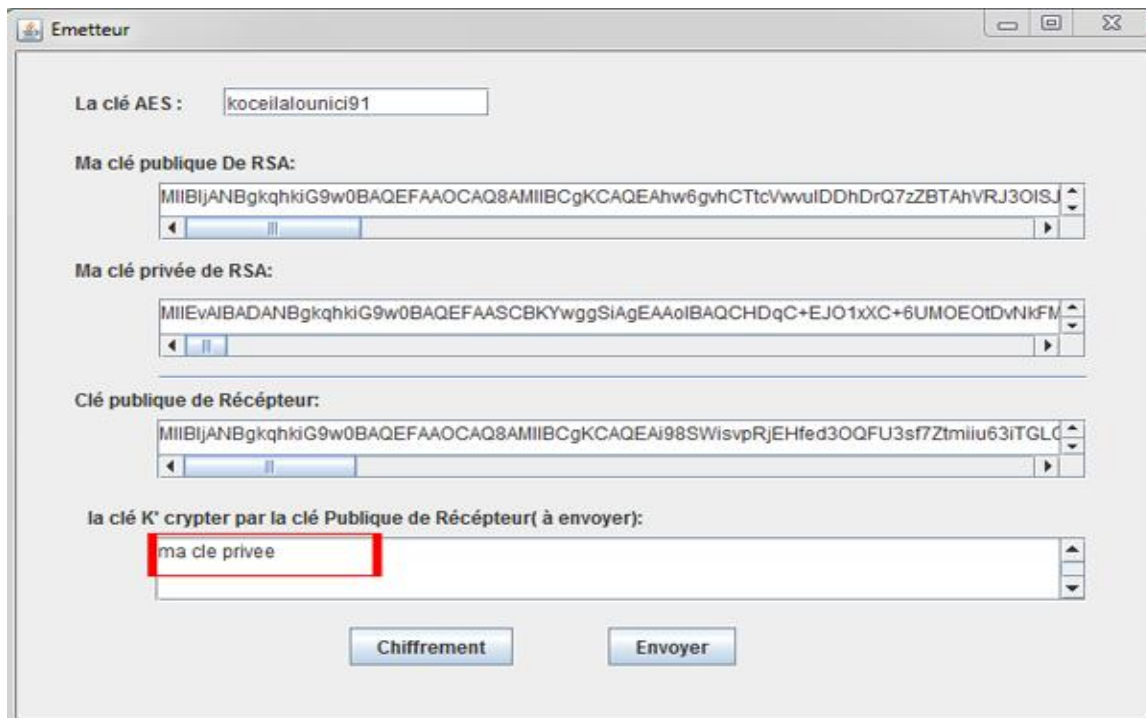


Figure4.10 envoi de la clé secrète en text clair

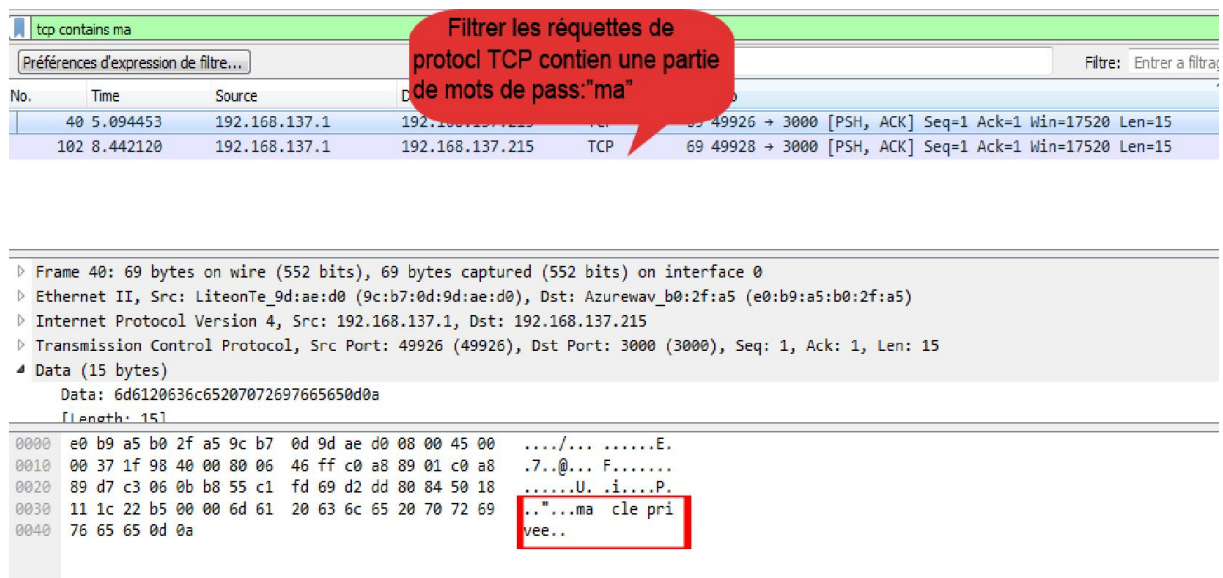


Figure4.11 interception de la clé secreta par un attaquant

Dans le cas d'envoi la clé de session (AES) cryptée par la clé publique du récepteur :

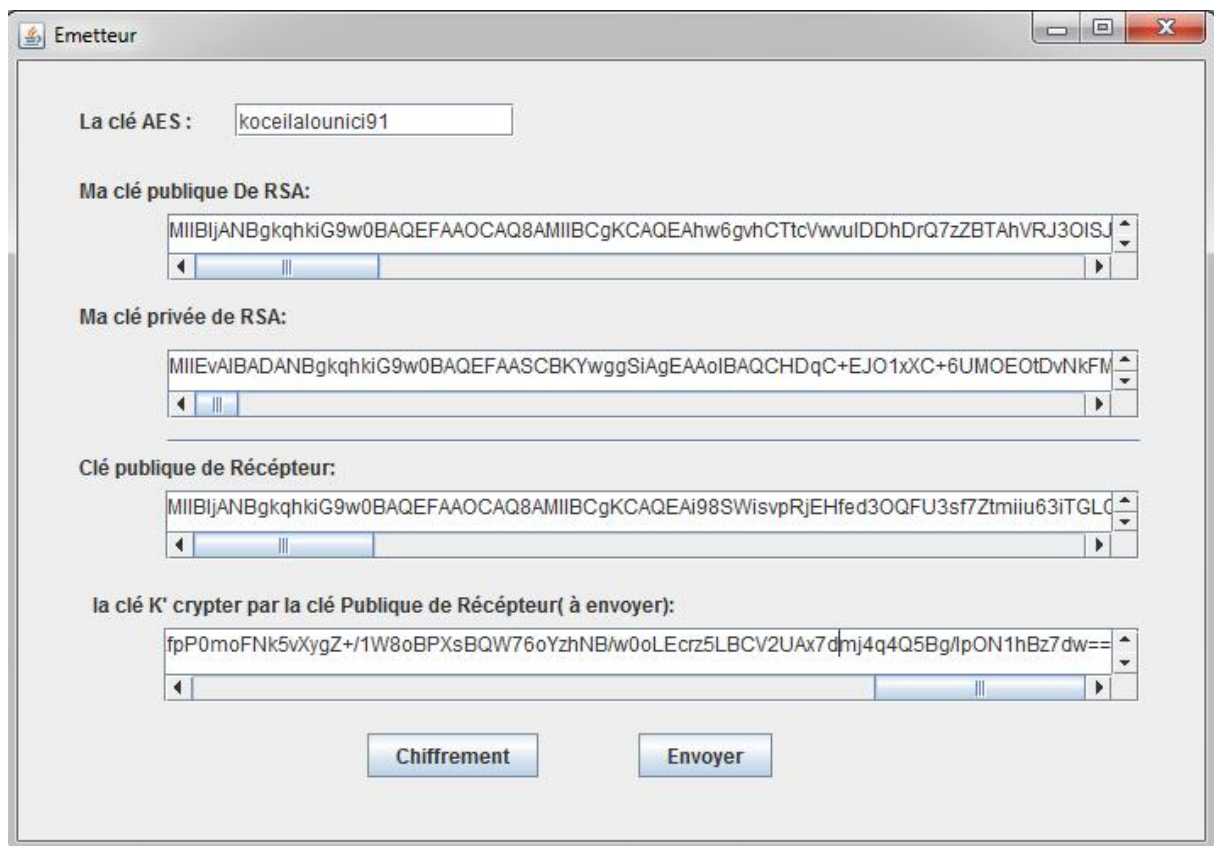


Figure4.12 envoie de la clé secrète en text cryptée

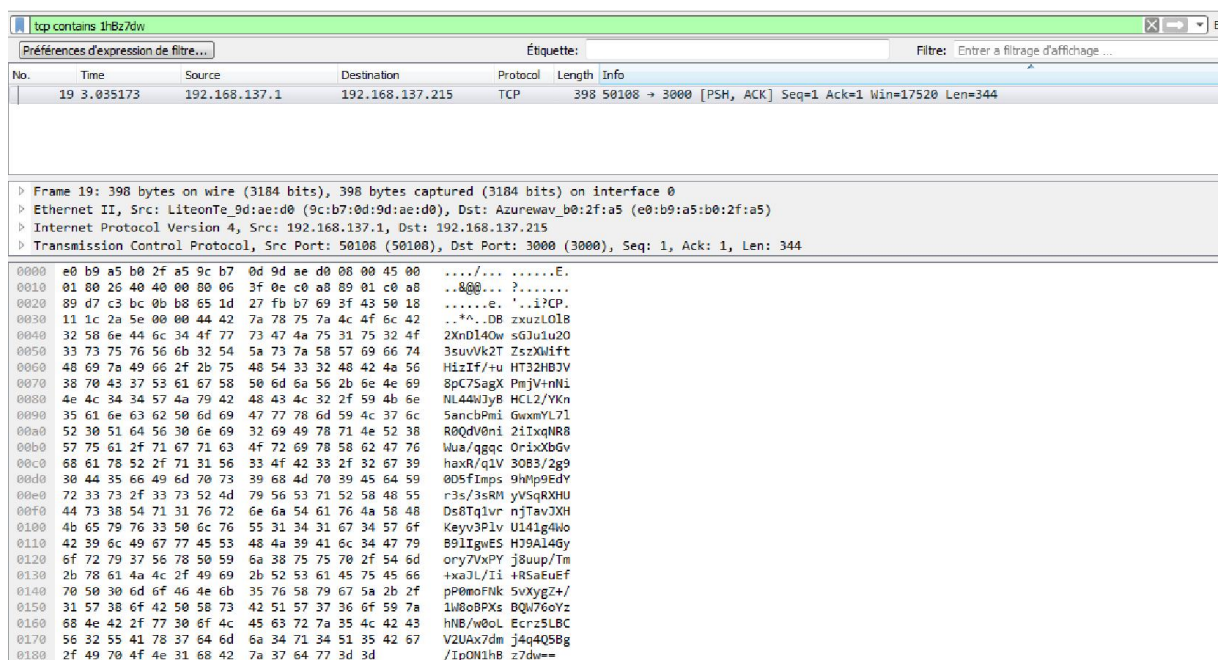


Figure4.13 interception de la clé secrète cryptée par un attaquant

On constate que le chiffrement symétrique pose quelques faiblesses importantes, malgré qu'il est un excellent choix pour le chiffrement de données, comme la distribution de la clé secrète. Ces faiblesses ont conduit à la création et la mise en œuvre du deuxième moyen de chiffrement asymétrique, c'est-à-dire lorsque on a crypté la clé de session par la clé publique de destinataire, l'attaquant ne pourra pas arriver de trouver la clé secrète.

Test02 : Chiffrement/déchiffrement d'une image :

Les figures représentent respectivement les organigrammes de chiffrement et de déchiffrement d'une image :

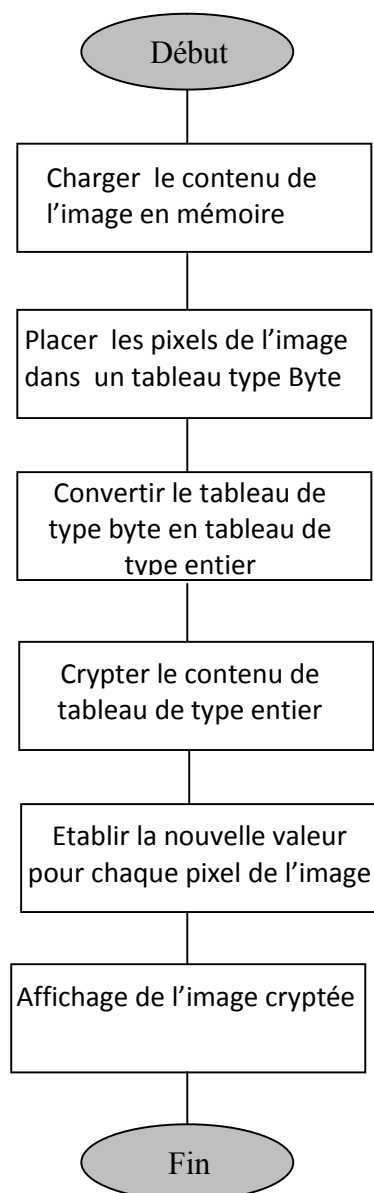


Figure4.14. *Procédure de chiffrement d'une image*

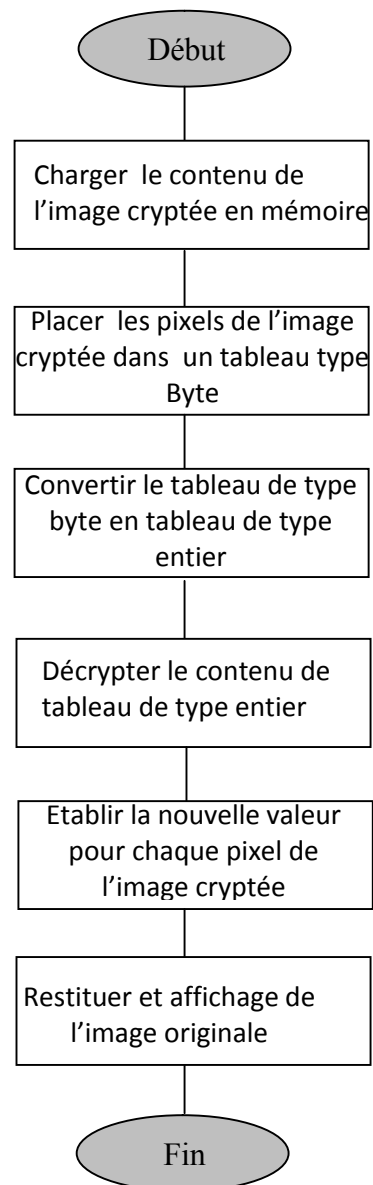


Figure4.15. *Procédure de déchiffrement d'une image*

➤ **Sur une image en niveau de gris :**

Dans cet exemple, on chiffre une image en niveau de gris avec les paramètres suivant :

Le mode :CBC

Le Ivector :00000000

La clé :koceilalounici91

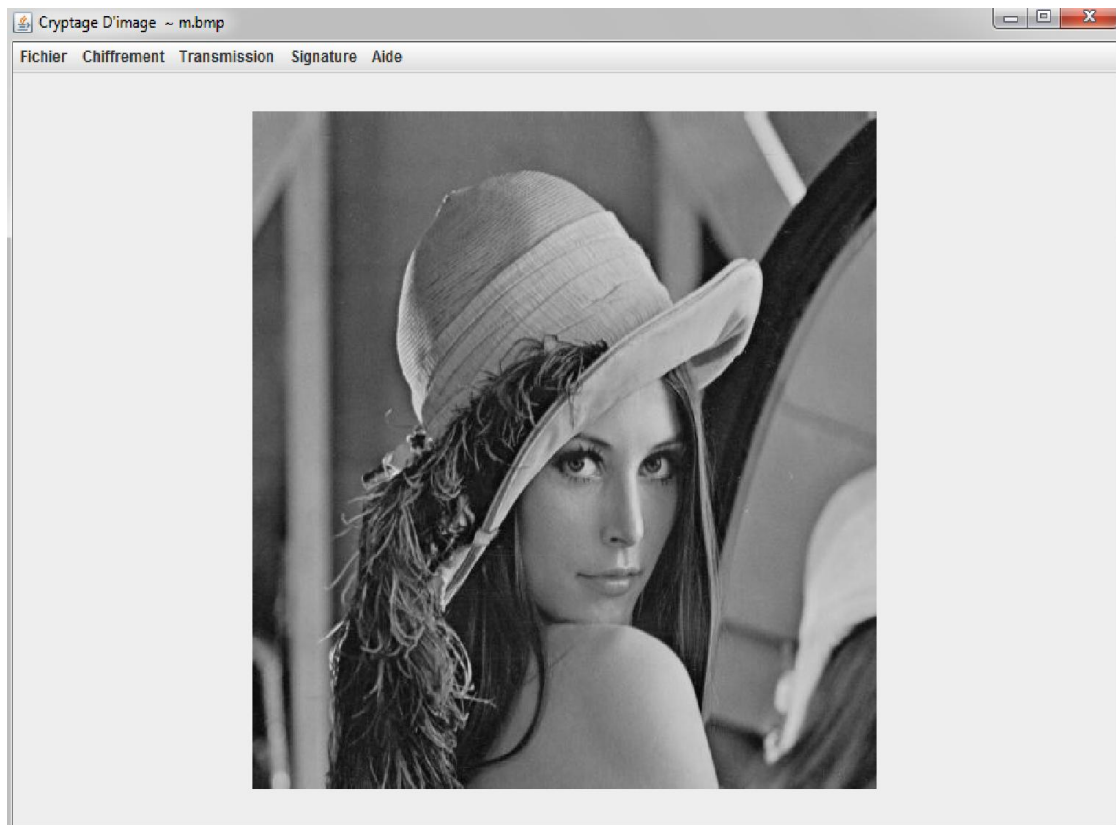


Figure4.16. *Image originale*



Figure4.17. *Chiffrement de l'image en mode CBC*

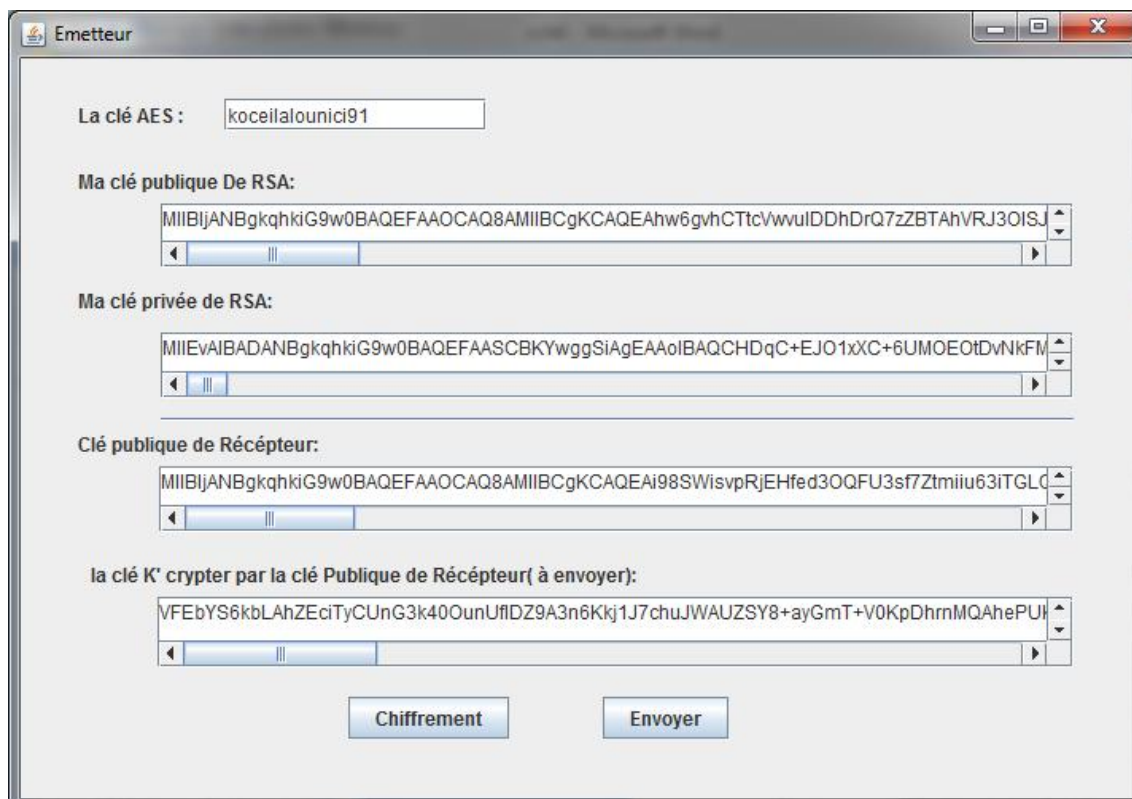


Figure4.18. *Transmission de tout {image chiffrée+K'' la clé chiffrée+empreinte chiffrée}*

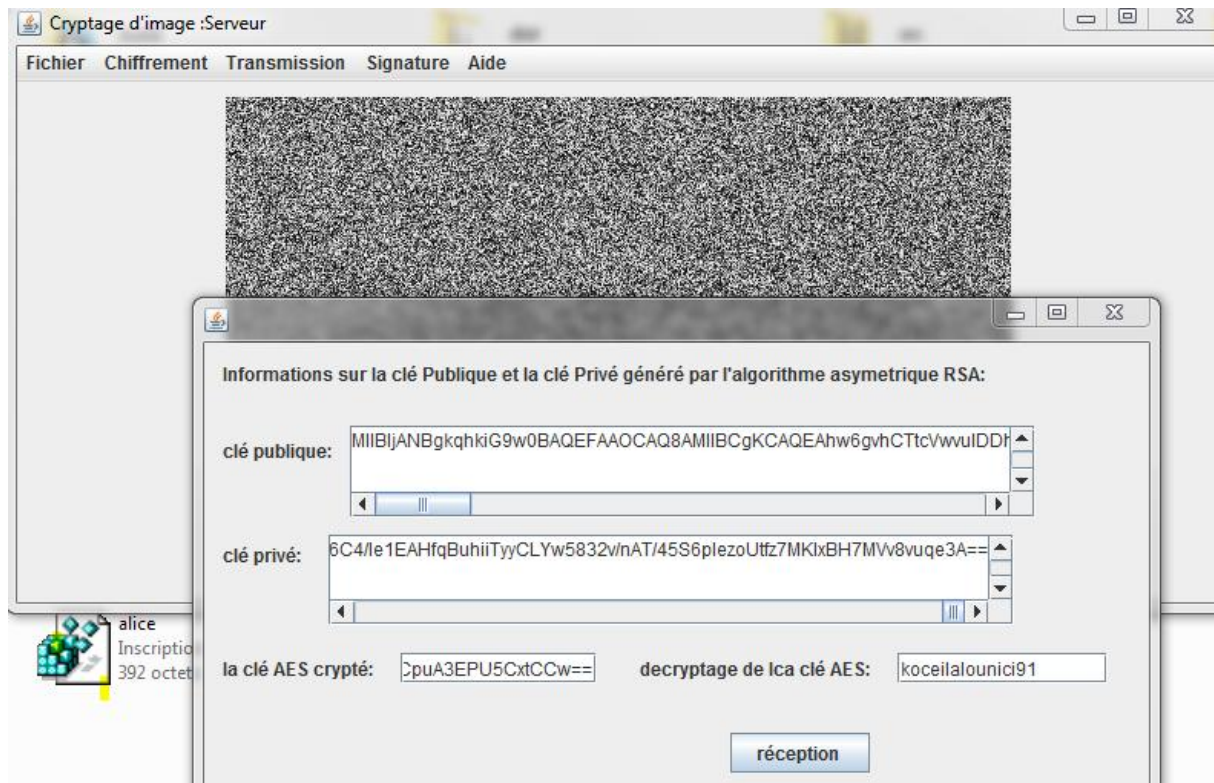


Figure4.19. la Réception des informations

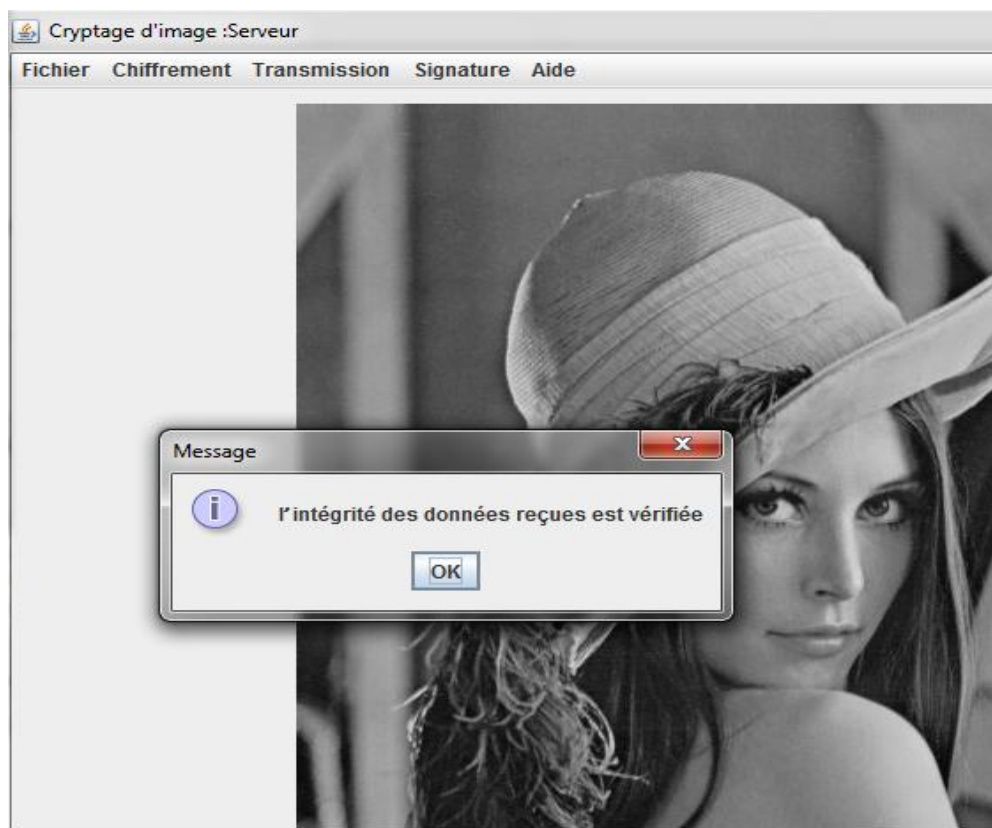


Figure4.20 Déchiffrement de l'image et vérification d'intégrité

➤ **Sur une image en couleur :**

Dans cet exemple, on chiffre une image en couleur sur 24bits avec les paramètres suivants :

Le mode :CBC

Le Ivector :00000000

La clé :koceilalounici91

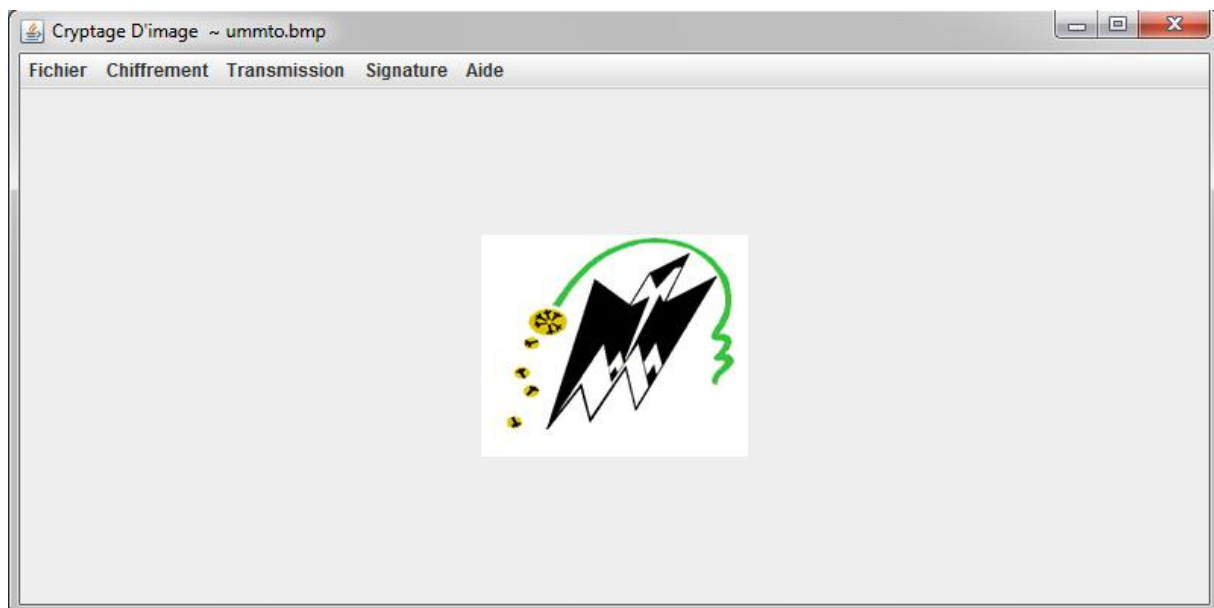


Figure4.21 *Image originale*

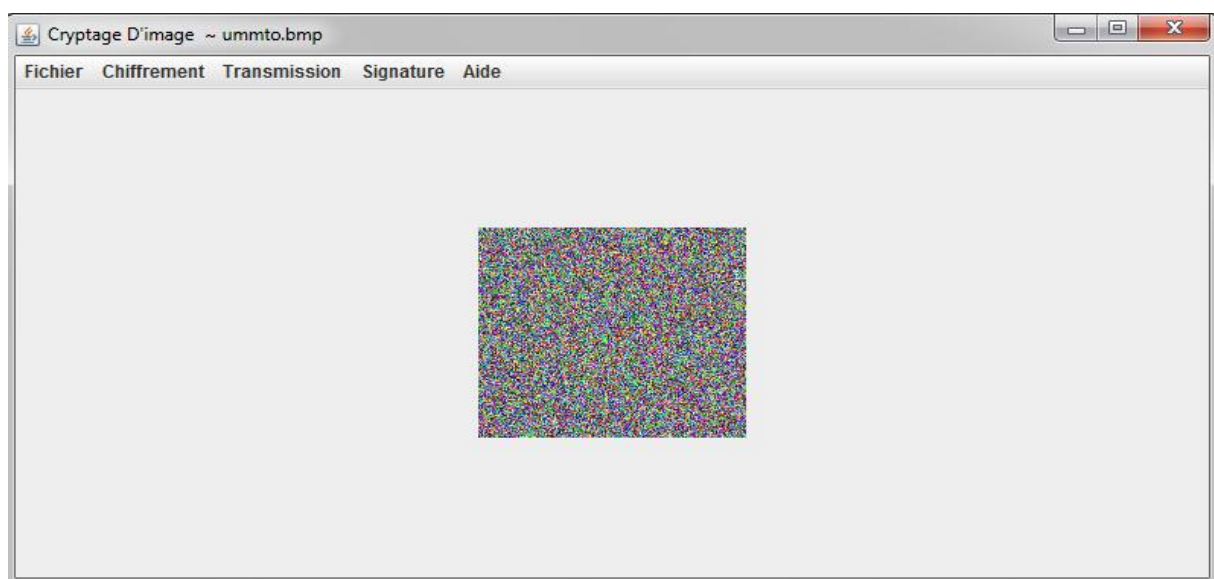


Figure4.22 *Image cryptée*

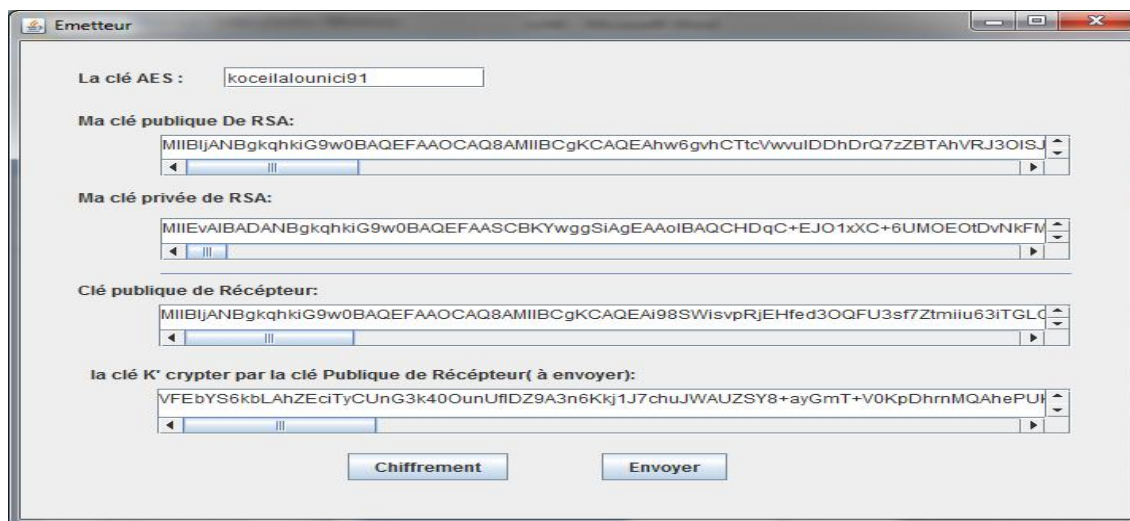


Figure4.23. Transmission de tout {image chiffrée+K'' la clé chiffrée+empreinte chiffrée}

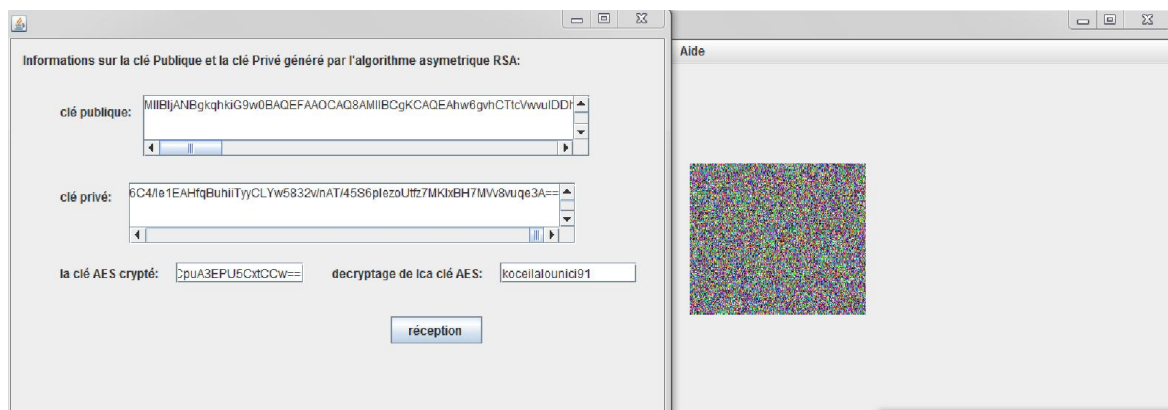


Figure4.24 la Réception des informations

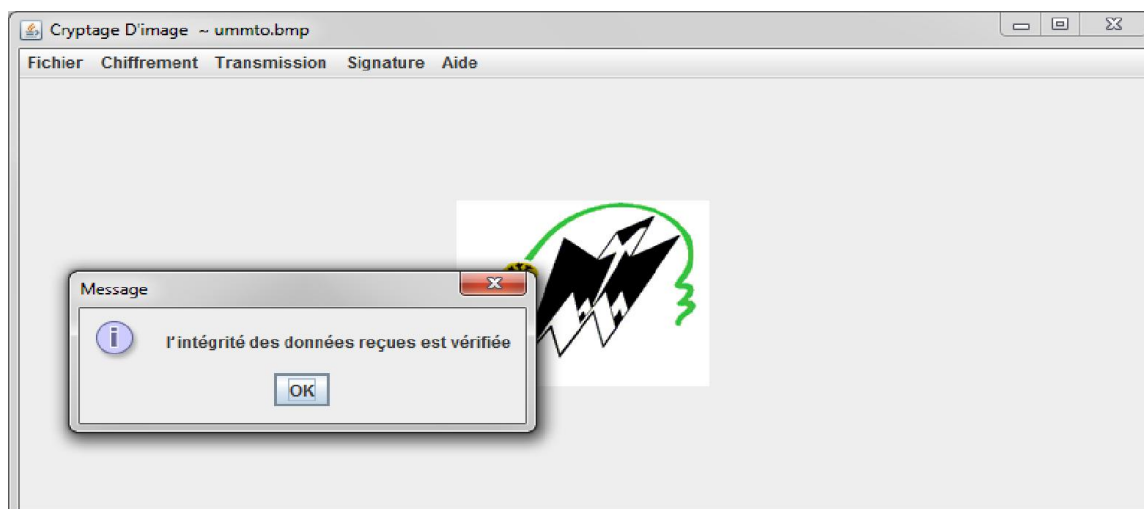


Figure4.25 Déchiffrement de l'image et vérification d'intégrité

Test03 : la clé de chiffrement/déchiffrement est différente :

Dans cet exemple, on essaye de déchiffrer l'image en niveau de gris de notre Test2 avec les paramètres suivants :

Clé de chiffrement :koceilalounici91 ,

Clé de déchiffrement :koceilalounici99

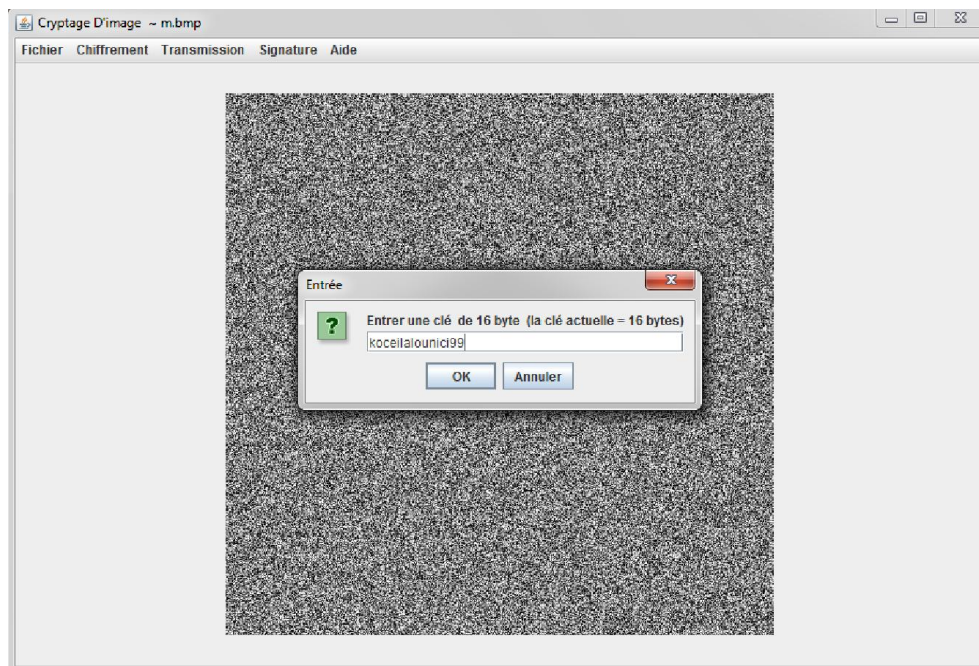


Figure4.26 Image cryptée par la clé :koceilalounici91



Figure4.27 image déchiffrée avec la clé :koceilalounici99

4.4. Temps d'exécution:

Les résultats de la figure 4.27 ont été évalués par notre application, Les expériences ont été menées à l'aide d'un processeur Intel® Core i5 avec une fréquence de 2.4 GHz. Dans cette figure, le temps d'exécution nécessaire pour crypter une image en niveau de gris (8bits) et en couleur (24bits) de différentes tailles est fourni.

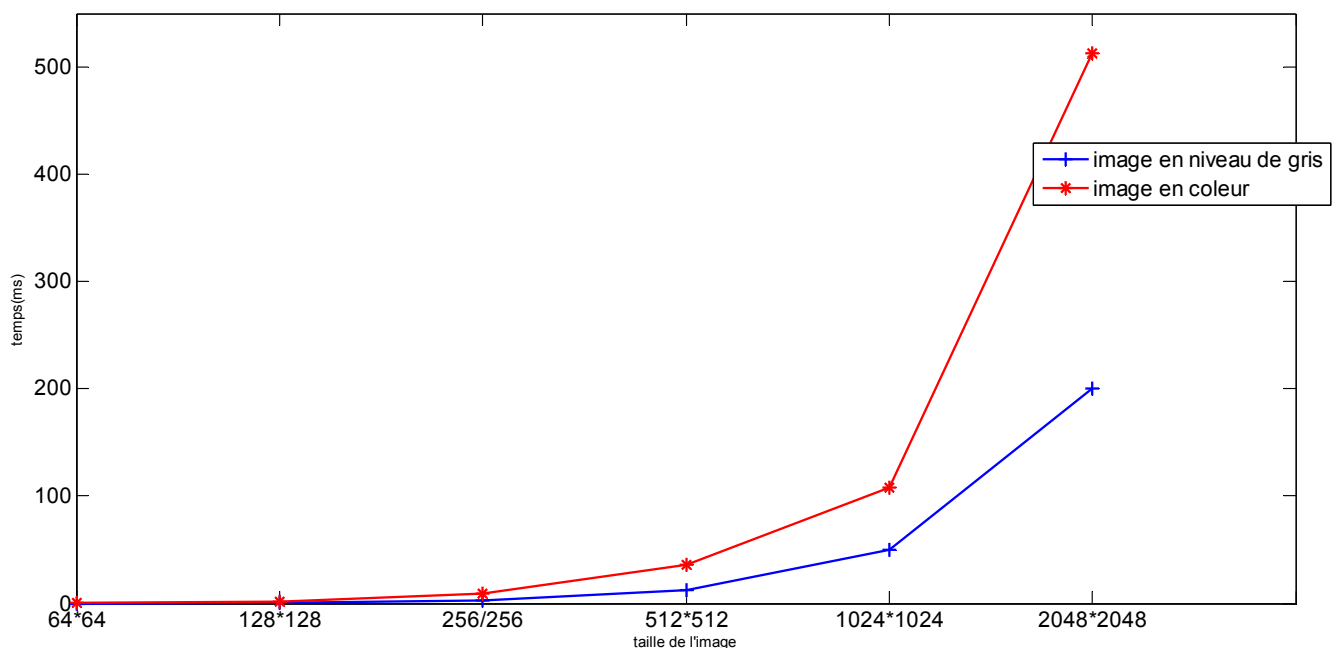


Figure 4.28 Mesure du temps de cryptage en milliseconde

A partir des résultats obtenus on peut conclure que l'algorithme AES elle est très rapide, Et comme il n'a pas des points faibles de sécurité connue jusqu'à présent, il peut être considérés comme un excellent algorithme de cryptage standard.

4.4. Conclusion :

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

➤ Ses points forts sont :

- Il repose sur le principe de chiffrement hybride
- Possibilité de chiffrer des différents formats d'images
- Gère les communications réseaux

➤ On a aussi relevé quelques lacunes :

- Ne gère pas plusieurs communications en même temps
- Les clés RSA et AES sont des tailles insuffisantes pour une sécurité optimale (1024 bits pour RSA et 128 pour AES)

Au cours de ce travail, nous avons étudié trois problématiques liées à la protection des images numériques : la transmission sécurisée, l'authentification et l'intégrité. Nous avons aussi développé et implémenté notre cryptosystème hybride selon une architecture Client/serveur pour faire face à ces problèmes sur la base des deux algorithmes AES (Advanced Encryption Standard), RSA (Rivest Shamir Adelman), et la fonction de hachage SHA-1.

Dans un premier temps, nous avons présenté plusieurs techniques et quelques théories de la cryptographie, qui nous a permet de comprendre cet axe de recherche. Nous avons aussi développé, en détails, les trois algorithmes AES, RSA et SHA-1, nécessaires dans notre cryptosystème de transmissions en donnant leurs avantages et inconvénients.

Dans la troisième partie, nous avons donné en détail notre cryptosystème hybride de transmission. Notre cryptosystème est composé de deux blocs : bloc d'émission et de réception, reliés entre eux par un canal de transmission. A l'émission, la clé de session est chiffrée deux fois par l'algorithme RSA pour assurer l'authentification et la transmission de la clé d'une façon sécurisée. Nous avons aussi chiffré les images avec la clé de session en utilisant l'AES afin de garantir la confidentialité. A la réception de ces images, des fonctions réversibles sont élaborés pour les déchiffrer et de garantir que l'image ne soit pas modifiée dans le canal de transmission.

Pour tester notre cryptosystème et analyser ces résultats, nous avons exploité plusieurs métriques d'évaluation du degré de chiffrement : analyse d'histogramme, calcul de la corrélation entre l'image originale et celle chiffrée, analyse de l'entropie, calcul de la corrélation entre les pixels adjacents, analyse de la sensibilité de la clé secrète et l'analyse différentielle.

A partir des résultats obtenus, nous avons remarqués que tous les histogrammes des images chiffrées sont uniformes pour les modes d'opérations (CBC , CFB ,OFB). Avec le mode de chiffrement ECB, l'histogramme présente des pics dû aux zones homogènes.

Nous avons aussi constaté que les pixels sont fortement décorrélés pour le mode de chiffrement CBC. Cela donne une distribution aléatoire d'information, les résultats de l'analyse de l'entropie confirment notre remarque.

Nous avons remarqué que tout changement de l'image chiffrée dans le canal de transmission, conduit au changement total de l'empreinte digitale à la réception. Nous pouvons donc dire que la fonction de hachage SHA-1 utilisée pour tester l'intégrité donne de très bons résultats.

Dans la dernière partie, nous avons réalisé une application pour l'implémentation de notre cryptosystème hybride de transmission sécurisée proposé selon une architecture client /serveur.

Les résultats expérimentaux relevés ont confirmé la bonne implémentation des algorithmes de chiffrement et de déchiffrement sur les deux applications client et serveur, ainsi que la transmission assurée par les sockets en mode connecté (TCP).

Les différents aspects que nous avons étudiés (cryptographie, traitement d'images, programmation sous MATLAB, programmation orientée objet sous java) nous ont été bénéfiques aussi bien sur le volet pédagogique en complétant notre formation mais aussi sur le volet de recherche en nous initiant à la lecture d'articles de recherches publiés récemment.

Comme perspectives, nous donnons les améliorations suivantes :

- Envisager de développer et d'implémenter d'autres techniques de chiffrement associant en particulier celles basées sur le chaos et les courbes elliptiques [24] [25].
- Améliorer le cryptosystème en introduisant une étape de codage et de compression de données pour optimiser le système de cryptage et de transmission [26] [27].

Ce travail offre un nouveau support à l'utilisateur, et ouvre un nouveau champ d'investigations dans le domaine de la technologie en générale et de la cryptographie en particulier, nous espérons donc que les promotions futures s'y inspireront.

A. Codes sous MATLAB :

Afficher l'histogramme

```
C=imread('m.bmp'); %charger l'image
subplot(2,2,1), imshow(C);
subplot(2,2,2), imhist(C);
```

L'Attaque de l'entropie

```
clear;
I=imread('lena.bmp'); %charger l'image
[h w]=size(I);
count(1:256) = 0.00;
for k=1:256
    for i=1:h
        for j=1:w
            if (I(i,j)==k-1)
                count(1,k)=count(1,k)+1;
            end
        end
    end
end
M=0;
for j=1:256
    p(j)=count(1,j)/(h*w);
    if (p(j) ~= 0)
        M=M+p(j)*log2(p(j));
    end
end
M = -M;
```

Calcul de la corrélation

```
I = imread('lena.bmp');
J = imread('cryptee.bmp');
R = corr2(I,J);
```

Corrélation entre les pixels adjacents

```
img=imread('lena.bmp');
%img=rgb2gray(img);
%horizontal
x1 =img(:,1:end-1);
y1 =img(:,2:end);
x1=x1(:);
y1=y1(:);
co_h=corr2(x1,y1);
co_h=sprintf('%.16f',co_h);
%Vertical
x2 = img(1:end-1,:);
y2 = img(2:end,:);
x2=x2(:);
y2=y2(:);
co_v=corr2(x2,y2);
co_v=sprintf('%.16f',co_v);
%diagonal
x3 = img(1:end-1,1:end-1);
y3 = img(2:end,2:end);
x3=x3(:);
y3=y3(:);
co_d=corr2(x3,y3);
co_d=sprintf('%.16f',co_d);
```

Analyse Différentielle :NPCR , UACI

```
C = imread('Ori.png');
D = imread('L1.png');
C = double(C);
D = double(D);
[X Y] = size(C);
sum = 0.00;
NPCR = 0.00;
UACI = 0.00;
for j = 1:X
    for k = 1:Y
        T = 0;
        if C(j,k) > D(j,k)
            T = C(j,k) - D(j,k);
            NPCR = NPCR+1;
        elseif C(j,k) < D(j,k)
            T = D(j,k) - C(j,k);
            NPCR = NPCR+1;
        end
        sum = sum+T;
    end
    UACI = UACI + sum/255.000;
    sum = 0;
end
MF = 100.00/double(X*Y);
UACI = UACI*MF;
```

Afficher la distribution des pixels adjacents

```
function []=distrub(ImageOriginal,ImageEncrypted)
%sur matlab command exécuter la fonction
%exemple:distrub('orig.bmp','crypt.bmp')
%image originale
I=imread(ImageOriginal);
A = im2double(I);
%image cryptée
I2=imread(ImageEncrypted);
A2 = im2double(I2);
%pixels adjacent de l'image originale
x1 = A(:,1:end-1,1);
y1 = A(:,2:end,1);
x2 = A(1:end-1,:,1);
y2 = A(2:end,:,1);
x3 = A(1:end-1,1:end-1,1);
y3 = A(2:end,2:end,1);
%pixels adjacent de l'image cryptée
x4 = A2(:,1:end-1,1);
y4 = A2(:,2:end,1);
x5 = A2(1:end-1,:,1);
y5 = A2(2:end,:,1);
x6 = A2(1:end-1,1:end-1,1);
y6 = A2(2:end,2:end,1);

subplot(3,2,1),grafik(x1,y1),title('Horizontal');
subplot(3,2,3),grafik(x2,y2),title('Vertical');
subplot(3,2,5),grafik(x3,y3),title('Diagonal');
subplot(3,2,2),grafik(x4,y4),title('Horizontal');
subplot(3,2,4),grafik(x5,y5),title('Vertical');
subplot(3,2,6),grafik(x6,y6),title('Diagonal');

end

function grafik(x,y)%2000 paires de pixels aléatoire.
randIndex = randperm(numel(x));
randIndex = randIndex(1:2000);
xRand = x(randIndex);
yRand = y(randIndex);
xRand = xRand * 256;
yRand = yRand * 256;
scatter(xRand,yRand, '.');
end
```

B. Introduction Générale Sur les images :

B.1. Les images numériques

Sont des clichés électroniques d'une scène ou numérisés à partir de documents tels que photographies, manuscrits, textes imprimés, et œuvres d'art.

L'image numérique est échantillonnée et mappée comme une grille de points ou éléments d'images (picture element ou pixel).

A chaque pixel correspond une valeur tonale (noir, blanc, niveaux de gris ou couleur), exprimée en code binaire (zéros et uns). Les chiffres binaires ("bits") de chaque pixel sont stockés dans une séquence par l'ordinateur, et souvent réduits à une représentation mathématique (compressés). Les bits sont alors réinterprétés et lus par l'ordinateur afin de délivrer une version analogique en vue d'être affichée ou imprimée

B.2. Définition de Pixels

Une image est constituée d'un ensemble de points appelés pixels (pixel est une abréviation de PICTURE ELEMENT) Le pixel représente ainsi le plus petit élément constitutif d'une image numérique. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions constituant l'image :

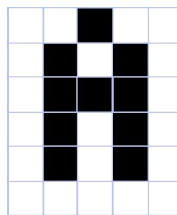


Figure 1 : *Les pixels d'une image*

Etant donné que l'écran effectue un balayage de gauche à droite et de haut en bas, on désigne généralement par les coordonnées $[0,0]$ le pixel situé en haut à gauche de l'image, cela signifie que les axes de l'image sont orientés de la façon suivante :

- L'axe X est orienté de gauche à droite.
- L'axe Y est orienté de haut en bas, contrairement aux notations conventionnelles en mathématiques, où l'axe Y est orienté vers le haut.

B.3. La taille des images

Ce sont les mesures horizontales et verticales d'une image exprimées en pixels. Les dimensions en pixels peuvent être déterminées en multipliant la largeur et la longueur de l'image par le dpi. Un appareil photo numérique possède également des dimensions en pixels, exprimés par le nombre de pixels horizontaux et verticaux définissant sa résolution (p.ex. 2048 par 3072). Pour calculer la résolution en dpi, divisez une des dimensions en pixels du document par la dimension en pouces correspondante.

$$\text{Taille de l'image} = \text{nb de pixels/longueur} \times \text{nb de pixels/largeur}$$

B.4. Représentation des couleurs

Nous l'avons vu une image apparaît comme une matrice où chaque case contient des nombres associés à une couleur. Usuellement on distingue 3 grands types de couleurs pour une image numérique :

- Le noir et blanc ;
- Les niveaux de gris ;
- La couleur.

Ces types sont généralement à choisir lors d'une numérisation par scanner ou lors de la configuration d'un appareil photographique.



Figure 2 : *Image originale*

ANNEXE

B.4.1. Image noir et blanc

Le noir et blanc est le plus simple. Le contenu de chaque case de la matrice est soit un 0 (noir) soit 1 (blanc). Le nombre de couleurs n'est que de 2 et le rendu de l'image le moins performant mais parfois suffisant dans le cadre par exemple de documents scripturaux.



Figure 3: *Image noire et blanc*

La taille du fichier obtenu, avec la même résolution (507x676) que celle de l'image initiale, est de 42,3 ko (au lieu de 980 ko !).

B.4.2. Niveaux de gris

Le codage dit en niveaux de gris permet d'obtenir plus de nuances que le simple noir et blanc.

Il offre des possibilités supplémentaires pour coder le niveau de l'intensité lumineuse. La couleur est codée souvent sur un octet soit 8 bits ce qui offre la possibilité d'obtenir 256 niveau de gris (0 pour le noir et 255 pour le blanc). On peut aussi le faire avec 16 niveaux de gris (4 bits).

000	008	016	024	032	040	048	056	064	072	080	088	096	104	112	120	128	136	144	152	160	168	176	184	192	200	208	216	224	232	240	248
000	008	016	024	032	040	048	056	064	072	080	088	096	104	112	120	128	136	144	152	160	168	176	184	192	200	208	216	224	232	240	248

Tableau 1 : *256 niveaux de gris*



Figure 4 : Image 'fleur' 256 niveaux de gris



Figure 5 : Image 'fleur' 16 niveaux de gris

De plus pour l'image codée sur 256 niveaux de gris, la valeur du pixel de coordonnées (54,51) (Situé en haut à gauche « dans le ciel ») est de 120.

Plus le niveau de gris est élevé, meilleur est la distinction des détails sur l'image. L'usage de ce codage est utilisé fréquemment pour la presse écrite ou l'envoi par messagerie électronique de fichier d'image de taille réduite avec une perte de lisibilité de l'image moindre.

ANNEXE

B.4.3. Image couleur

B.4.3.1. Principe :

La couleur d'un pixel est obtenue, comme le ferait un peintre, par le mélange de couleurs fondamentales. Il ne s'agit pas ici de décrire toutes les techniques utilisées. Nous allons décrire un des principes les plus couramment utilisé qui est celui de la synthèse additive.

B.4.3.2. Codage RVB

Le principe consiste à mélanger les 3 couleurs : rouge, vert et bleu (noté RVB ou RGB en anglais). A l'aide de ces 3 couleurs, on obtient toute une palette de nuances allant du noir au blanc. A chaque couleur est associé un octet (donc 256 niveaux de luminosité) de chacune des couleurs fondamentales.

Rouge	Vert	Bleu	Couleur
255	0	0	Rouge
0	255	0	Vert
0	0	255	Bleu
0	255	255	Cyan
255	0	255	Magenta
255	255	0	Jaune
0	0	0	Noir
128	128	128	Gris moyen
255	255	255	Blanc

Tableau 2 : *Principe codage de la couleur*

Un pixel 'couleur' est alors codé avec 3 octets et on a alors la possibilité d'obtenir 2²⁴ possibilités de couleurs soit de l'ordre de 16 millions de couleurs différentes.

On dit que les images obtenues sont en couleurs « vraies ». La qualité colorimétrique obtenue est celle d'une photographie argentique couleur.



Figure 6 : *Valeurs RGB d'un pixel*

Ainsi le pixel de coordonnées (54, 51) (en haut à gauche de l'image « dans le ciel ») a pour valeurs RVB respectivement (77, 118, 239).

Il existe d'autres formats de codage de la couleur (image à palette 256 couleurs, mode HAM...) pour lesquels on pourra se reporter utilement à leur description en ligne sur Internet.

B.5. Poids d'une image

Pour connaître le poids (en octets) d'une image, il est nécessaire de compter le nombre de pixels que contient l'image, cela revient à calculer le nombre de cases du tableau, soit la hauteur de celui-ci que multiplie sa largeur. Le poids de l'image est alors égal à son nombre de pixels que multiplie le poids de chacun de ces éléments.

B.6. Formats d'image

Concernant les images, il existe une grande variété de formats de fichiers, qui ont chacun des caractéristiques propres.

Consistent à la fois des bits comprenant les informations de l'image et d'en-tête concernant la lecture et l'interprétation du fichier. Les formats de fichiers varient en termes de résolution, profondeur de bit, capacités de couleurs et support pour la compression et les métadonnées. Le plus connu est le Jpeg. Il s'est largement démocratisé car il compresse les images pour les enregistrer.

ANNEXE

	Type (matriciel / vectoriel)	Compression des données	Nombre de couleurs supportées	Affichage progressif	Animatio n	Transparence
JPEG	matriciel	Oui, réglable (avec perte)	16 millions	Oui	Non	Non
JPEG2000	matriciel	Oui, avec ou sans perte	4 milliards	Oui	Oui	Oui
GIF	matriciel	Oui, Sans perte	256 maxi (palette)	Oui	Oui	Oui
PNG	matriciel	Oui, sans perte	Palettisé (256 couleurs ou moins) ou 16 millions	Oui	Non	Oui (couche Alpha)
TIFF	matriciel	Compression ou pas avec ou sans pertes	de monochrome à 16 millions	Non	Non	Oui (couche Alpha)
SVG	vectoriel	compression possible	16 millions	* ne s'applique pas *	Oui	Oui (par nature)

Tableau 2 : *Tableau comparatif des Formats d'image*

- [1] **K. Shikha**, and **K. Ishank**, “*Data Security Using RSA Algorithm In MATLAB*”, International Journal of Innovative Research and Development, Vol. 2 Issue 7, pp. 479-483, 2013.
- [2] **A. Kazeem Bolade**, and **P.Akinlolu Adediran**, “*Development of a GUI for Hybrid (DES-RSA) Data Encryption and Decryption for Transmission of Biomedical Data*”, International Journal of Science and Research, Vol. 3 Issue 11, pp. 1281-1248, 2014.
- [3] **P. Radhadevi**, and **P. Kalpana**, “*Secure image encryption using AES*”, International Journal of Research in Engineering and Technology, Vol. 1 Issue 2, pp. 115-117, 2012.
- [4] **X. Wei**, **L. Guo**, **Q. Zhang**, **J. Zhang**, and **S. Lian**, “*A novel color image encryption algorithm based on DNA sequence operation and hyper-chaotic system*”, The Journal of Systems and Software, Vol. 85 Issue 2, pp. 290-299, 2012.
- [5] **C. Guang hui**, **H. Kai**, **Z. Yizhi**, **Z. Jun**, and **Z. Xing**, “*Chaotic Image Encryption Based on Running-Key Related to Plaintext*”, The Scientific World Journal, Vol. 2014, pp. 9, 2014.
- [6] **J. daemen**, and **V. rijmen**, “*AES proposal: the rijndael block cipher*,” technical report, proton world int. l, katholieke universiteit leuven, esatcosic, belgium, 2002.
- [7] **W. stallings**, “*cryptography and network security principles and practice*”, pearson, united states of america, 2011, p. 900.
- [8] NIST: national institute of standards and technology, web site: www.nist.gov/aes/
- [9] **H. wen**, “*AES encryption algorithm analysis and security study*,” computer applications of petroleum 2008, vol 16 no.2.
- [10] **R. rivest**, **A. shamir**, and **I. adleman**, “*a method for obtaining digital signatures and public- key cryptosystems*,” communications of the ACM, 21:120-126, 1978.
- [11] **Z. Ameen Abduljabbar**, **H. Jin**, “*Robust scheme to protect authentication code of message/image documents in cloud computing*,” International Conference on Computing, Networking and Communications (ICNC), pp. 1-5, 2016.
- [12] **R. stinson**, “*cryptography: theory and practice, (discrete mathematics and its applications)*,” chapman & hall/ crc press, new york, november 2005.

- [13] **B. schneier**, “*applied cryptography*,” CRC press, united states of America, 1996.
- [14] **W. Mao**, “*Modern Cryptography: Theory and Practice*,” prentice hall ptr, juillet 2003.
- [15] **M.Boukhatem Mohammed Belkaid**, “*Application des techniques de cryptage pour la transmission sécurisée d’images MSG*” mémoire de magister en télédétection , université mouloud mmameri de tizi ouzou, mars 2015.
- [16] **h. Ramdane, H. Mokrane**, “*Conception et réalisation d’un cryptosystème hybride*” mémoire d’ingénieur en électronique , université mouloud mmameri de tizi ouzou, 2008.
- [17] **M. dworkin**, “*recommendation for block cipher modes of operation*,” NIST special publication 800-38 ,2001 edition.
- [18] **C. Shannon**, “*Communication theory of secrecy systems*,” Bell System Technical Journal, vol. 28, no. 4, pp. 656–715, 1949
- [19] <http://in.mathworks.com/help/images/ref/corr2.html>
- [20] **J. Fridrich, Int. J. Bifurcation Chaos**, vol. 8, no. 6, (1998).
- [21] **Wu, Y., Noonan, J. P., Agaian**, “*NPCR and UACI Randomness Tests for Image Encryption*”, Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications, pp. 31-38,2011.
- [22] **H. Schildt**, “*JAVA: A beginner’s Guide*” McGraw-Hill, 4 edition, 2007.
- [23] **D. Bishop**, “*NTRODUCTION TO CRYPTOGRAPHYWITH JAVA™ APPLET*” , Jones and Bartlett Publishers, 1 edition, 2003.
- [24] **H. hamiche, M. lahdir , M. tahanout and S. djennoune** , “*masking digital image using a novel technique based on a transmission chaotic system and spiht coding algorithm*” international journal of advanced computer science and applications (IJACSA), 3(12), 2012.
- [25] **J. Fridrich**, “*Symmetric ciphers based on two-dimensional chaotic maps*,” International Journal of Bifurcation and Chaos, vol. 8, no. 6, pp. 1259–1284, 1998.
- [26] **A. jose marconi and M. rodrigues**, “*transfert sécurisé d’images par combinaison de techniques de compression et cryptage*” thèse de doctorat de l’université de Montpellier ii, oct. 2006.

- [27] **c. oatrieux, H. maitre**, “*images médicales, sécurité et tatouage*,” annales des télécommunications, numéro spécial santé, vol. 58, pp. 782-800, 2003.