République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU**G**ENIE **E**LECTRIQUE ET D'**I**NFORMATIQUE
DEPARTEMENT D'AUTOMATIQUE

Mémoire de Fin d'Etudes
De MASTER ACADEMIQUE

Domaine : **Sciences et Technologies**
Filière : **Génie électrique**
Spécialité :**Commande des systèmes**

Présenté par
Mohamed BOUTORA
Djouher BEN AMI

Thème

Conception, Etude et Réalisation d'un Cryptosystème Hybride de Transmission d'Images.

Mémoire soutenu publiquementle15 / 09 /2015 devant le jury composé de:

M Redouane KARA
MCA, UMMTO,Président

M Hamid HAMICHE
MCA, UMMTO,Encadreur

M Said DJENNOUNE
Professeur, UMMTO,Co-Encadreur

M Mourad LAHDIR
MCA, UMMTO, Examinateur

M Rezki HADDOUCHE
MAA, UMMTO, Examinateur

M^{elle} Sarah KASSIM
Doctorante, UMMTO, Invitée

Ce travail a été réalisé au sein du **L**aboratoire de **C**onception et **C**onduite des **S**ystèmes de **P**roduction (**L2CSP**).

Dédicaces

A YEMMAYI que DIEU ait son âme en PAIX.

A mes parents, mes frères, mes amis et toute ma famille.

A celle que j'ai aimée plus que tout au monde...

A Alice, Bob et Eve.

Sofiane

Dédicaces

A :

Mes **Parents** qui m'ont soutenu durant toute ma scolarité.

Mes **Sœurs** Kenza, Lylia, Malika

Mes **Amies** Thilelli, Djoudjou, Dihia, Lylia

Mes **Oncles** surtout **Dada Achour**

Exceptionnellement Mon **Binôme** Sofiane

Et à tous ceux qui me connaissent.

Djoudjou

Remerciements

Nous remercions notre encadreur ***M. Hamid Hamiche***, Maitre de conférences classe A à l'université Mouloud Mammeri de Tizi Ouzou pour ses conseils et ses encouragements.

Nous tenons à exprimer notre profonde gratitude et nos remerciements les plus sincères à notre Co-encadreur ***M. Saïd Djennoune***, professeur à l'université Mouloud Mammeri de Tizi Ouzou, pour ses conseils, sa disponibilité et ses critiques qui nous ont permis de mener à bien ce travail.

Nos grands remerciements vont aussi à l'égard de ***Melle. Sarah Kassim,*** Doctorante au sein du Laboratoire de Conception et de Conduite des Systèmes de Production (L2CSP) du Département Automatique pour sa disponibilité, ses conseils, sa patience et ses encouragements qui nous ont permis de surmonter bien des difficultés. Qu'elle trouve ici notre profonde gratitude.

Nous remercions également tous les membres du jury pour nous avoir honorés par leur présence, et pour avoir accepté d'évaluer ce mémoire.

Table des matières

Table des matières

Table des matières

Résumé

Le travail de mémoire réalisé s'inscrit dans le contexte de la transmission sécurisée d'images. Notre travail consiste à proposer un cryptosystème hybrideassociant trois algorithmes qui sont le DES, le RSA et SHA-1 qu'on va ensuite appliquer à la transmission d'images.

Dans une première partie,nous présentons les trois algorithmes utilisésqui sont : Le DES « Data Encryption Standard », un des algorithmes phares de la cryptographie à clé secrète il remplira le premier objectif de sécurité à savoir la confidentialité des données transmises. L'algorithme de chiffrement à clé publique RSA « Rivest Shamir Adleman » est utilisé pour authentifier l'expéditeur et assurer sa non répudiation. Le dernier objectif à savoir l'intégrité des données reçues est assuré par une fonction de hachage cryptographique SHA-1 de la famille « Secure Hash Algorithm » qui fournit une empreinte digitale sur 160 bits.

Nous consacrons une partie de ce mémoire afin de présenter la méthode hybride utilisée pour associer les trois algorithmes en présentant les blocs émetteur et récepteur du cryptosystème. Nous allons aussi faire subir à notre cryptosystème hybride une série de tests de sécurité à travers plusieurs métriques (analyses d'histogrammes, tests de corrélations, contrôle d'intégrité ...etc.). La qualité de ces résultats va mettre en évidence la robustesse du cryptosystème.

Dans une dernière partie nous envisageons une implémentation de notre cryptosystème autour de deux cartes Arduino-Uno. Les résultats pratiques sont discutés et comparés aux résultats théoriques.

Mots clés

Cryptographie symétrique et asymétrique, chiffrement et déchiffrement, algorithmes : DES RSA, SHA-1, fonction de hachage, cryptosystème hybride, blocs émetteurs et récepteur, transmission d'images, Arduino-Uno.

Liste des figures

CHAPITRE I

CHAPITRE II

CHAPITRE III

CHAPITRE IV

Liste des tableaux

CHAPITRE I

CHAPITREIII

Glossaire

A

AES: Advanced Encryption Standard.

C

CBC : Cipher Block Chaining

CFB : Cipher Feed-Back

D

DES : Data Encryption Standard

E

ECB: Electronic Code Book

EEPROM: Electrically-Erasable Programmable Read-Only Memory

EFF: l'Electronic Frontier Foundation

I

IDE: Integrated Devlopment Environment.

M

MAC : Message Authentification Code.

MAE: Mean Absolute Error.

MD5: Message Digest 5.

MDC: Modification Detection Code

MIT: Massachusetts Institute of Technology.

MSE: Mean Square Error.

N

NIST: National Institute of Standards and Technology.

NPCR: Number of Pixel Change Rate.

O

OFB: Output Feed-Back

P

PWM: Pulse Width Modulation.

R

RAM: Random Access Memory

RSA: Rivest Shamir Adleman

RC4: Rivest Cipher 4

S

SHA-1 : Secure Hash Algorithme-1.

Introduction générale

Depuis l'antiquité, l'homme n'a pas cessé de chercher les différents moyens afin de transmettre un message à son correspondant et pouvoir ainsi communiquer avec lui en toute sécurité. Tous ces efforts ont conduit à l'évolution des modes de communication et à leur développement dans le but d'atteindre une confidentialité totale dans la communication.

Une science est ainsi née « *La cryptologie* », étymologiquement *la science du caché* ou par extension *la science du secret*, qui est quasiment la base de tout mécanismes, outils et concepts de la sécurité des différentes communications. Elle est subdivisée en deux domaine distincts et complémentaires, *la cryptographie* qui est la branche qui s'intéresse aux méthodes de protection de messages ou documents, parallèlement à la conception de méthodes de protection, on trouve également *la cryptanalyse* qui consiste en l'étude des procédés et méthodes cryptographiques dans le but de trouver des faiblesses et en particulier de pouvoir déchiffrer des messages chiffrés sans posséder la clé de chiffrement.

La révolution numérique a engendré des moyens plus faciles pour le traitement, le stockage et la transmission d'information multimédia en particulier d'images numériques. Cependant, elle a aussi engendré des moyens de falsification, de contrefaçons et d'espionnage très avancés. Dans ces circonstances, il est devenu nécessaire de chiffrer ces images avant de les transmettre.

Le travail réalisé dans ce mémoire s'inscrit dans ce contexte. Le premier objectif du travail proposé est d'étudier quelques méthodes classiques de cryptage qui sont basées sur les algorithmes de chiffrement. Nous avons opté pour trois algorithmes les plus connus et les plus utilisés qui sont l'algorithme à clé secrète (symétrique) DES, l'algorithme à clé publique (asymétrique) RSA et l'algorithme à fonction de hachage SHA-1. L'autre objectif de notre travail est de proposer une méthode hybride associant les trois algorithmes RSA, DES et SHA-1. Les algorithmes hybrides de chiffrement et de déchiffrement proposés sont développés sous MATLAB en code script. L'efficacité de la méthode présentée est illustrée sur le cryptage d'images. Enfin, le dernier objectif de notre travail est de concevoir et de réaliser un système de transmission sécurisée utilisant la méthode de chiffrement hybride proposée et construit autour de deux cartes Arduino Uno. La première joue le rôle de l'émetteur et l'autre de récepteur.

Le mémoire est organisé autour de quatre chapitres :

Le premier chapitre est une introduction à la cryptographie. Il aborde entre autres les deux principaux schémas de chiffrement en cryptographie qui sont le chiffrement à clé secrète et le chiffrement à clé publique, ainsi que les fonctions de hachages cryptographiques, Il est conclu par un aperçu des différentes attaques (cryptanalyses) cryptographiques.

Le deuxième est une étude détaillée des algorithmes phares de chaque schéma, le DES pour le chiffrement à clé secrète, le RSA pour le chiffrement à clé publique, et SHA-1 pour la fonction de hachage.

Le troisième chapitre est composé de deux parties principales. Dans la première, nous développons l'algorithme du cryptosystème hybride proposé. Après avoir implémenté ce cryptosystème sous MATLAB, nous exposons les résultats de chiffrement et de déchiffrement appliqué sur une image. La deuxième quant à elle est consacrée aux tests de robustesse de notre algorithme à l'aide d'un ensemble de métriques d'évaluation du degré de chiffrement.

Dans le quatrième et dernier chapitre, le cryptosystème expérimental construit autour de deux cartes Arduini-Uno est exposé. Nous donnons les résultats obtenus dans le cas d'une transmission sécurisée d'image.

Enfin, notre travail est conclu par une conclusion générale et quelques perspectives ouvertes qui peuvent être envisagées comme suite à notre projet.

CHAPITRE I

INTRODUCTION A LA CRYPTOGRAPHIE

Chapitre I

Introduction à la cryptographie

I.1 Introduction

La cryptographie est par définition l'art de cacher l'information, elle désigne l'ensemble des techniques qui permettent de chiffrer messages, son objectif principale est de permettre à deux personnes **Alice** et **Bob** de communiquer à travers un canal peu sûr de telle sorte qu'un opposant **Eve** ne puisse pas comprendre ce qui est échangé.

Ce chapitre est alors une introduction à la cryptographie, il aborde après un Historique, différentes généralités sur la cryptographie, il introduit aussi les deux principaux schémas cryptographiques (symétrique et asymétrique). Enfin, un aperçu des différentes attaques (cryptanalyses) cryptographiques sera donné.

I.2 Histoire de la cryptographie: [1] [2] [3] [4]

Anciennement considérée comme un art, la cryptographie est désormais reconnue comme une science à part entière.

Les premières utilisations connues de la cryptographie remontent à l'Antiquité, où la plus ancienne trace de message chiffré a été retrouvée sur une table en argile sur les bords du Tigre en Irak. Au fil des années, les motivations militaires ont conduit les Hommes à développer de nouvelles méthodes de chiffrement plus robustes afin d'éviter que les tactiques ou plans de bataille ne tombent dans les mains de l'ennemi. Les Spartiates[1] ont ainsi inventé le premier dispositif militaire connu : *la scytale,* ou *bâton de Plutarque*. La scytale en elle-même est un bâton de bois, dont le diamètre est connu uniquement de l'émetteur et du destinataire du message.

Il a fallu attendre l'époque de *Jule César,* vers 50 avant J-C, pour voir apparaitre de véritables systèmes cryptographiques. Le plus célèbre d'entre eux est *le chiffre de César,* qui consistait simplement à décaler les lettres d'un message de trois positions vers la droite dans l'alphabet latin. Plus tard *Blaise de Vigenère* (1586) introduit un nouveau chiffrement dans lequel on ne se contente pas d'un seul décalage comme pour César mais de plusieurs. Une section est consacrée à l'étude de ces méthodes un peu plus loin dans ce travail.

En 1883, *Auguste Kerckhoffs* énonce un principe fondateur de la cryptographie moderne : « *les mécanismes de chiffrement et de déchiffrement doivent pouvoir être rendus publics, la confidentialité des messages doit être garantie uniquement par le secret d'une clé* ».

Le bond technologique suivant survient au XX^{ème} siècle lors des deux guerres mondiales. Avec les besoins militaires des différentes armées de protéger leurs communications ont permit de voir l'apparition de machines spécialement conçues pour le chiffrement et le déchiffrement, on peut citer par exemple, *Enigma*, *la C-36,* la *machine de Lorenz*. Dans la deuxième moitié du vingtième siècle, la cryptographie est devenue beaucoup plus mathématique et a été grandement facilité par l'apparition des premiers ordinateurs. Cette cryptographie moderne est initiée par le travail de *Claude Shannon*[2] en 1948 sur la *théorie mathématique de l'information*

Aujourd'hui, en plus de l'amélioration des méthodes classiques, de nouvelles techniques de chiffrement sont introduites, telles que la *cryptographie quantique* qui consiste à chiffrer une clé en utilisant des photons envoyés par fibre optique, et toute tentative d'interception de la clé modifie la polarisation des photons. Et la *cryptographie chaotique* qui se base sur des instabilités de natures inhabituelles des systèmes non linéaires, ce fut alors la découverte des signaux chaotiques, qui ont un comportement déterministe mais qui font penser à des allures pseudo-aléatoires. Le principe de la cryptographie chaotique est alors de noyer le message en clair dans un signal chaotique. Pour le chiffrement et le déchiffrement, on doit alors disposer au niveau de l'émetteur et du récepteur du même signal chaotique pour pouvoir récupérer le message chiffré.

---

[1] Les Spartiates : Habitants de Sparte, une célèbre cité de la Grèce antique.

[2] Claude Shannon : Mathématicien et ingénieur électricien américain.

I.3 Généralités sur la cryptographie :

I-3-1. Terminologie : [4] [5]

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

***Clé* :** Une clé est un ensemble de paramètres utilisés en entrée d'une opération cryptographique (chiffrement, déchiffrement).

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

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

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

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

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

La cryptographie : Etymologiquement « écriture secrète », devenue par extension l'étude de cet art (donc aujourd'hui la science visant à créer des cryptogrammes, c'est-à-dire à chiffrer).

La cryptanalyse : science analysant les cryptogrammes en vue de les décrypter.

La cryptologie : C'est une science mathématique regroupant la cryptographie et la cryptanalyse.

Plaintext : Terme anglais désignant le texte clair à chiffrer.

Ciphertext : Terme anglais désignant le texte chiffré.

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

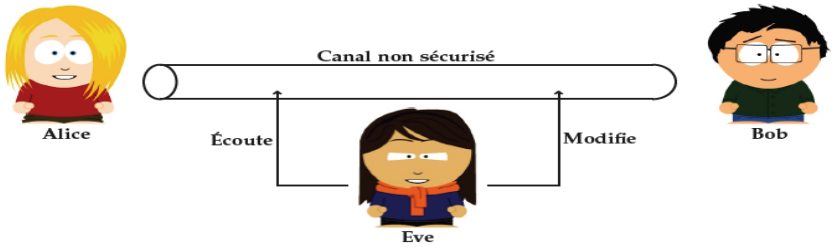


Figure I-1: Canal de communication non sécurisé : Alice et Bob s'échangent des messages via le canal écouté par Ève.

I-3-2.Objectifs de sécurité de la cryptographie : [1] [4] [5]

La cryptographie permet de rendre un certain nombre de services:

- ***La confidentialité*** : Est l'assurance qu'un document ne sera pas lu par une personne qui n'en a pas le droit lors de la transmission de ce document ou lorsqu'il est archivé. La confidentialité représente le service le plus important en cryptographie.
- ***L'intégrité*** : d'un (message, document, fichier…) est la garantie que cet objet n'a pas été modifié par une autre personne que son auteur. Pour certaines applications, l'intégrité d'une donnée est au moins aussi importante que sa confidentialité. Par exemple, lors d'une transaction bancaire, il est indispensable que la somme d'argent mise en jeu ne soit pas modifiée.
- ***L'authentification*** : Elle est l'assurance de l'identité d'un objet, généralement une personne, mais cela peut aussi s'appliquer à un serveur, une application,…. Dans la vie courante, la présentation d'une carte d'identité et la signature manuelle assurent un service d'authentification.
- ***La non répudiation*** : Le but de ce service est que l'émetteur d'un message ne puisse pas nier l'avoir envoyé et le récepteur l'avoir reçu. En cryptographie ce service est assuré par des certificats.

I-3-3. Techniques de chiffrement : [2]

Les premiers algorithmes utilisés pour le chiffrement d'une information ont été assez rudimentaires dans l'ensemble, comme nous l'avons vu dans le paragraphe consacré à l'Histoire de la cryptographie. Ces anciennes méthodes de chiffrement ont constitué la première pierre de l'édifice qui a mené vers les méthodes dites modernes. Comme nous allons le voir dans la section qui suit ces méthodes dites modernes, au delà de leur complexité se basent sur des principes artisanaux.

I.3.3.1 Méthodes anciennes : [2] [3] [4] [5]

- **Chiffrement par substitution :**

Le chiffrement par substitution consiste à remplacer systématiquement une lettre par une autre. Citons l'exemple d'une fonction qui décale de **n** positions les lettres constituant une phrase ce type de substitution est dit monoalphabétique.

Il existe un autre type de substitution, dit polyalphabétique qui introduit la notion de clés.

Exemples de chiffrement par substitution :

- **« Chiffrement de César » :**

Cette méthode de chiffrement est considérée comme la plus ancienne méthode de chiffrement par substitution monoalphabétique. Son principe est assez élémentaire, il consistait simplement à décaler les lettres d'un message de trois positions vers la droite dans l'alphabet latin. La lettre A est ainsi transformée en D. le B en E. etc.

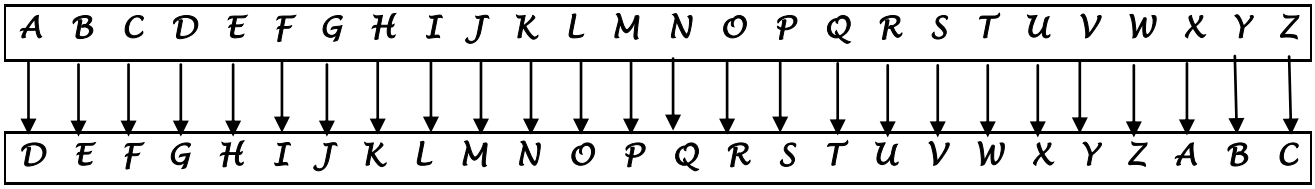


Figure I-2 : Chiffre de César : Les lettres du message en clair sont décalées de trois positions dans l'alphabet.

- **Chiffrement de Vigenère :**

C'est l'un des premiers systèmes à introduire le concept de clé de chiffrement. Blaise de Vigenère (1523-1596) a repris le chiffre de César, ou le décalage utilisé change de lettre en lettre. Pour cela il a utilisé une table composée de 26 alphabets, écrits dans l'ordre, mais décalés de ligne en ligne d'un caractère, mieux connue sous la table de Vigenère (voir figure 1- 3).

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

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

Figure I-3 : Table de Vigenère.

On veut coder le texte « MER TERRE CIEL » avec la clé « SEALS ».

1- On commence par écrire la clé sous le texte à coder :

M	E	R		T	E	R	R	E		C	I	E	L
S	E	A		L	S	S	E	A		L	S	S	E

2- Pour coder la lettre M, la clé est donnée par la lettre S. On regarde dans le tableau l'intersection de la colonne donnée M avec la ligne donnée par S.
On trouve E et on continu.
Le message chiffré est : *EIR EWJVE NAWP*

- **Chiffrement par transposition :**

La transposition est un moyen de diffusion (Principe de Shannon). Elle permet la dispersion des redondances ou du texte clair en la répartissant dans le texte chiffré.

Le principe est alors très simple : Les lettres se retrouvent mélangées, leur identité est conservée (un L reste un L), mais en changeant de position.

Ci-dessous un exemple de chiffrement par transposition dite en colonne simple, qui consiste à écrire un texte ligne par ligne dans un tableau 10 colonnes, puis transmettre le message par colonne.

Chiffrons le message suivant : « POURQUOI REMETTRE A DEMAIN CE QUE ON PEUX FAIRE A UNE MAIN ».

P	O	U	R	Q	U	O	I	R	E
M	E	T	T	R	E	A	D	E	M
A	I	N	C	E	Q	U	E	O	N
P	E	U	X	F	A	I	R	E	A
U	N	E	M	A	I	N			

L'émetteur émet le message par colonnes, cela donne la suite de caractère suivante.

PMAPU OEIEN UTNUE TCXM QREFA UEQAI OAUIN IDER REOE EMNA

Le déchiffrement par le destinataire se fait comme suit : Il sait que le tableau utilisé pour le chiffrement a 10 colonnes. Pour retrouver le nombre de ligne, ceci n'est pas difficile quand il sait que le nombre de colonne utilisée est 10. Alors il lui suffit de diviser les le nombre de caractère qu'il a reçu par 10. Comme il a reçu 47 caractères, il en déduit qu'il y a 4 lignes de 10 caractères, et une cinquième ligne incomplète qui compte 7 caractères.

Il va alors considérer 7 groupes de 5 caractères, puis 3 groupes de 4 caractères qu'il va ranger dans son tableau.

- **Chiffrement par produit :**

Le principe de ce mode de chiffrement consiste à utiliser des deux méthodes étudiées précédemment, a savoir la substitution et la transposition. La combinaison de ces deux méthode un chiffrement assez robuste qui fournit un bon niveau de sécurité, la plus part des algorithmes de chiffrement à clé symétrique comme nous allons le voir un peux plus loin utilisent le chiffrement par produit.

I.3.3.2 Méthodes modernes : [2] [3] [5]

- **Structure de chiffrement de Feistel[3] :**

Cette structure est pratiquement la base de tous les algorithmes de chiffrement modernes. La plupart des algorithmes de la fin du XX éme siècle, étaient des schémas de Feistel, (DES, Blowfish, RC5,…).

La structure de Feistel est assez simple et le chiffrement et déchiffrement sont similaires. Elle est basée sur des opérations de substitutions et de permutations avec une fonction principale qui change de clé à chaque round.

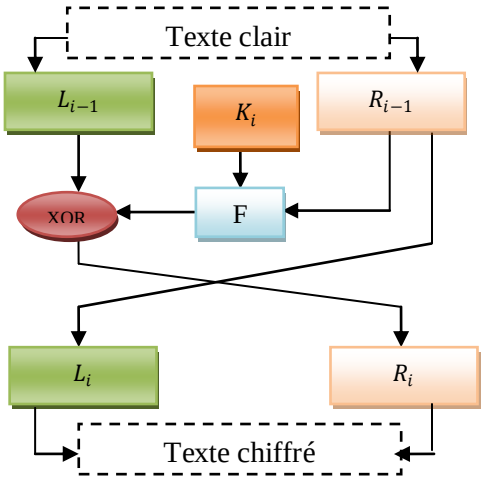


Figure I-4 : Structure de chiffrement de Feistel.

Le schéma ci-dessus peut être appliqué autant de fois sur le texte qu'on veut chiffrer, chaque répétition est appelée ***Round,*** les parties gauches L_i et droites R_i sont modifiées comme suit :

[3] Feistel : *Du nom du cryptologue Américain Horst Feistel ingénieur chez IBM. !*

$$\begin{cases} \mathbf{Li=R_{i-1}} \\ \mathbf{Ri=L_{i-1}\oplus F\ (R_{i-1},\ K)} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [1.1]$$

Chaque round applique :

1- une fonction F qui comporte des permutations via des P-BOX et des substitutions via des S-BOX.
2- Mixage linéaire via l'opération XOR.
3- Application de la clé de round qui est intégrée dans la fonction F via une opération XOR.

I.4 Chiffrement en cryptographie classique : [6]

L'objectif fondamental de la cryptographie est de permettre à deux personnes, Alice et Bob, de communiquer à travers un canal peu sûr de telle sorte qu'un opposant Eve, ne puisse pas comprendre ce qui est échangé. Le processus de chiffrement d'un message **M** afin d'obtenir le message chiffré **C**, est obtenu à partir d'une *fonction de chiffrement* $\boldsymbol{\varepsilon}$ telle que $\boldsymbol{C = \varepsilon(M)}$ **.** Le processus de déchiffrement est obtenu par la fonction de déchiffrement **D**. Il est donc requis que **:** $\boldsymbol{D(C) = D\big(\varepsilon(M)\big) = M}$

Un algorithme cryptographique est la définition des fonctions mathématiques utilisées pour le chiffrement et le déchiffrement. En pratique, les fonctions $\boldsymbol{\varepsilon}\ et\ \boldsymbol{D}$ sont paramétrées par des clés $\boldsymbol{K_e}\ et\ \boldsymbol{K_d}$ qui peuvent prendre l'une des valeurs d'un ensemble appelé espace des clés. On a donc la relation suivante :

$$\begin{cases} \varepsilon_{K_e}(M) = C. \\ D_{K_d}(C) = M. \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad [1.2]$$

Le type de relation qui unit les clés $\boldsymbol{K_e}\ et\ \boldsymbol{K_d}$ utilisées dans le chiffrement et déchiffrement permet de définir deux grandes catégories de systèmes cryptographiques classiques:

1. Les systèmes classiques à clé secrète d'une part étudiés dans la section **I-4-1.**
2. Les systèmes à clé publique d'autre part, qui seront exposés dans la section **1-4-2.**

I.4.1 Chiffrement à clé secrète : [2] [3] [4] [6].

I.4.1.1 Principe de fonctionnement :

Pour reprendre les notations vues dans l'équation [1.2] on a $\boldsymbol{K_e = K_d = K}$ autrement dit, Alice et Bob conviennent secrètement d'une clé privée K. ils conviennent également d'un algorithme cryptographique. La clé K est alors utilisée pour le chiffrement et le déchiffrement.

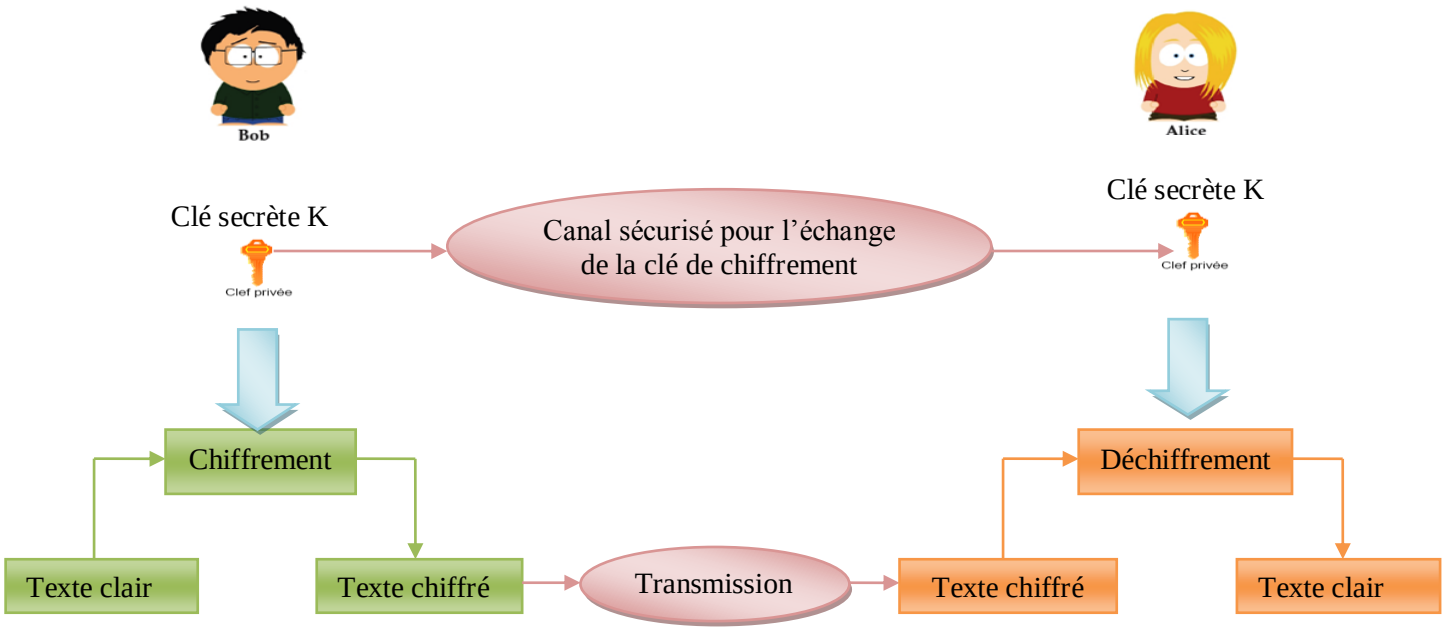


Figure I-5 : Principe d'un chiffrement à clé secrète (symétrique).

I.4.1.2 Procédés de chiffrement en cryptographie à clé secrète :

La cryptographie à clé secrète fonctionne habituellement suivant deux procédés différents :

1. **<u>à flot (en continu):</u>** Chaque nouveau bit est manipulé directement.
2. **<u>Par bloc :</u>** Chaque message est d'abord partitionné en blocs de tailles fixes, les fonctions s'appliquent alors sur chaque bloc.

Les deux sections qui suivent sont consacrées à l'étude de ces deux procédés :

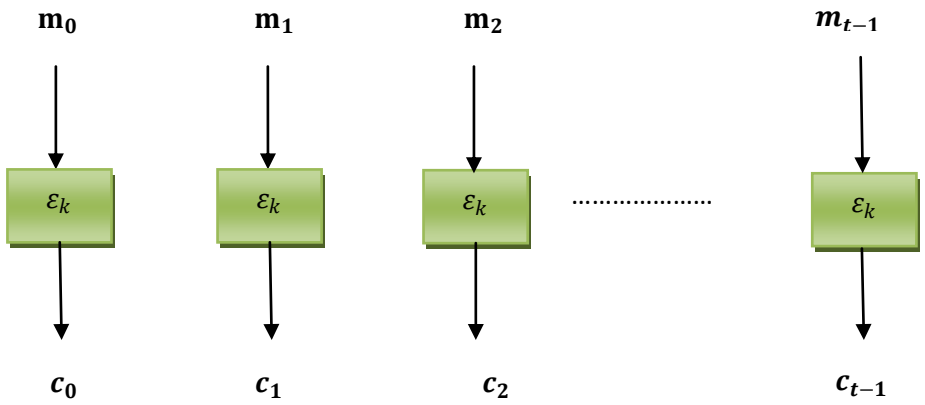
- **Le chiffrement par bloc : [1] [2] [3] [6]**

Les algorithmes de chiffrement par bloc constituent la principale méthode de chiffrement offerte par la cryptographie. Son principe de fonctionnement est comme suit :
Le message d'entrée est découpé en blocs de taille fixe de n bits suivant les spécifications de l'algorithme utilisé, qui sont tous traités simultanément par l'algorithme.

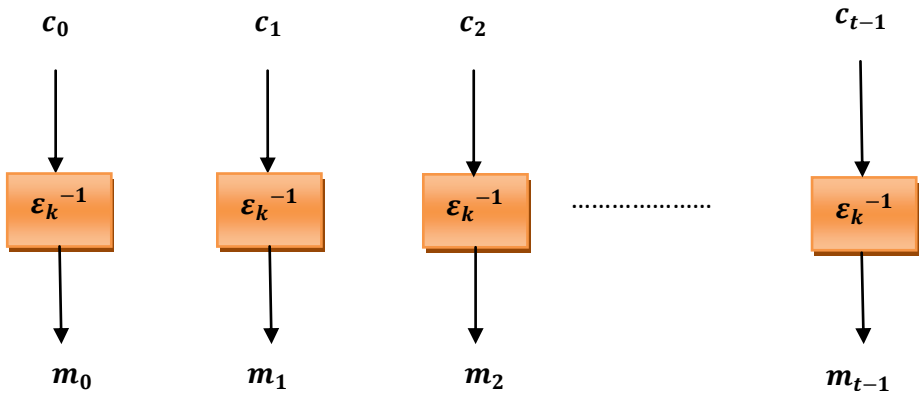
Dans les chiffrements par bloc, il est primordial de savoir gérer des messages de plus d'un bloc. Il faut alors lier les blocs de messages, les uns avec les autres dans un mode opératoire. On dénombre 4 modes principaux :

- **Le mode ECB (Electronic Code-Book) :**

Le mode ECB représente le fonctionnement le plus simple, les t blocs de messages $\{m_0, m_1, \ldots m_{t-1}\}$ clairs sont traités indépendamment par la fonction de chiffrement ε_k qui utilise la même clé K.



(a) Chiffrement en mode ECB.



(b) Déchiffrement en mode ECB.

Figure 1-6: Le mode ECB « Electronic Code-Book ».

Rarement utilisé sur les applications sensibles, en effet une simple substitution de m par $\varepsilon_k(m)$ a la forte propriété que les blocs de messages m_1 et m_2 égaux auront le même bloc de message chiffré. Ainsi, un attaquant pourra voir des répétions dans les blocs de messages chiffrés, et donc récupérer de l'information, ce qui voulait être évité.

Pour pallier ce point faible, un autre mode a été introduit, le **CBC** (Cipher-Block Chaining).

- **Le mode CBC (Cipher-Block Chaining) :**

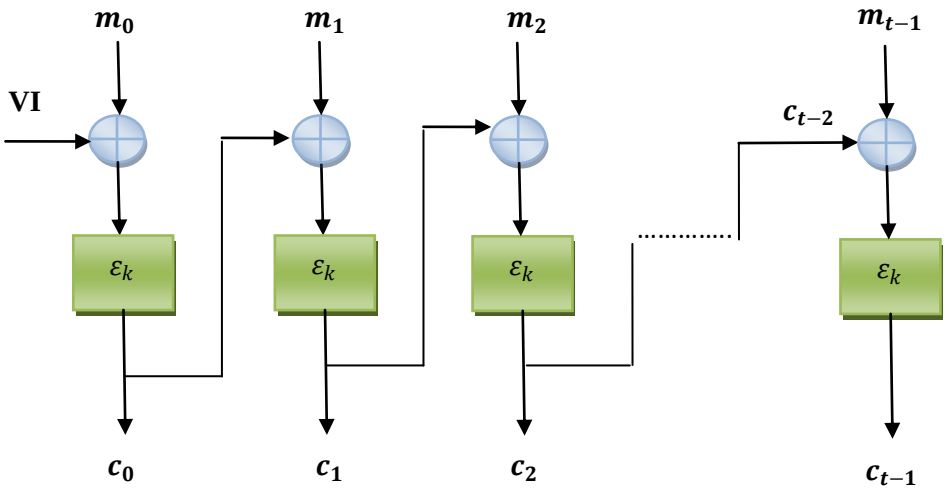
Ce mode permet de perturber le fonctionnement précédent. Les t blocs de messages m_0, m1,…. mt_{-1} sont traités en chaine par la permutation ε_k pour former la séquence de blocs chiffrés C_0, C_1,…, C_{t-1}.Il faut alors ajouter un **V**ecteur **I**nitial (**VI**) aléatoire, puis chaque bloc est d'abord modifié par XOR avec le bloc chiffré précédant avant d'être lui-même crypté par :

$$C_t = \varepsilon_k(m_t \oplus C_{t-1}) \quad \text{..... [1.3]}$$

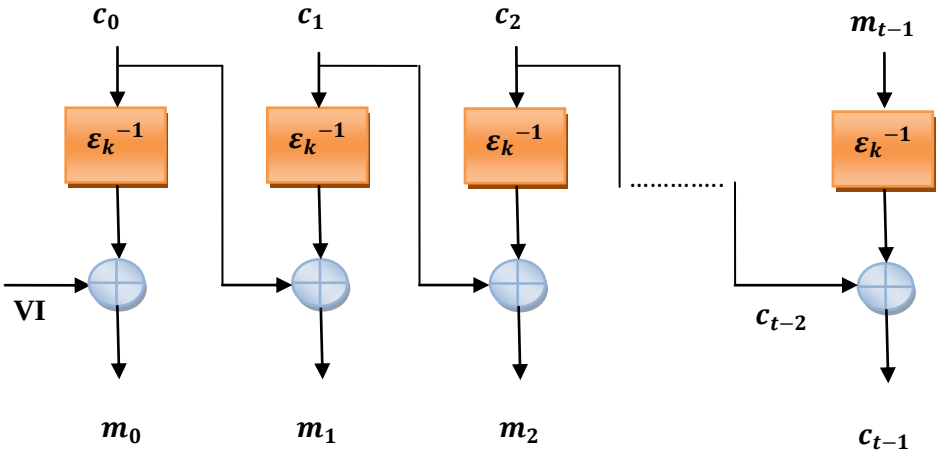
Le déchiffrement nécessite l'inverse de la fonction de chiffrement εk^{-1} :

$$m_t = C_{t-1} \oplus \varepsilon k^{-1}(C_t) \quad \text{..... [1.4]}$$

Le chainage entraine une dépendance entre un bloc chiffré et tous les blocs chiffrés précédents. Ce qui permet de rendre aléatoire les entées séquentielles de la permutation ε_k.



(a) Chiffrement en mode CBC.



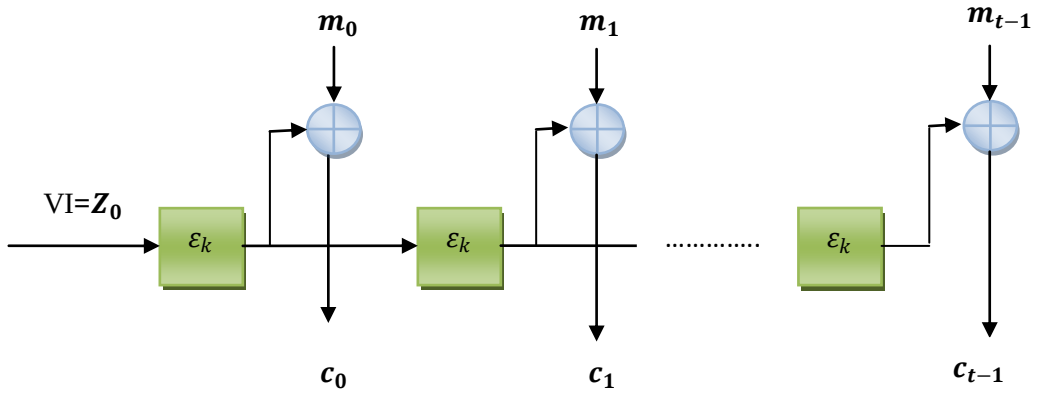
(b) Déchiffrement en mode CBC.

Figure I-7 : Le mode CBC « Cipher-Block Chaining ».

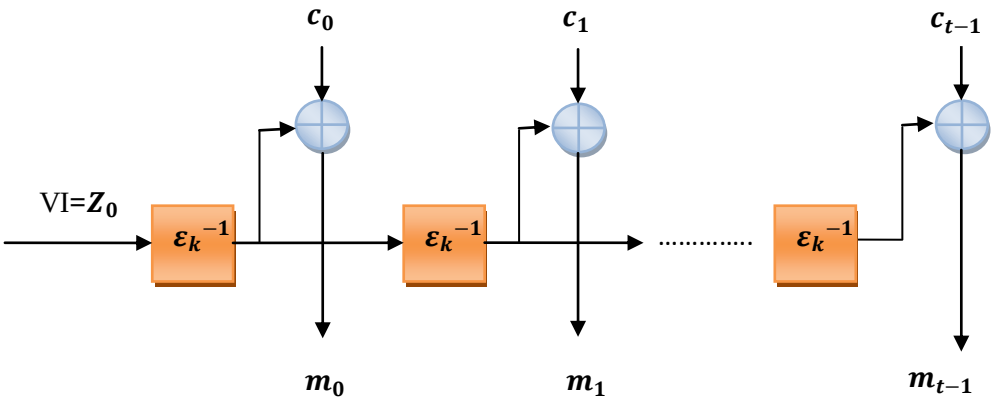
- **Les modes CFB (Cipher Feed-Back) et OFB (Output Feed-Back):**

Pour ces deux derniers modes, le chiffrement du clair se fait par une suite de XOR avec des clés qui sont issues du chiffrement symétrique.

OFB : La suite des clés est un chiffrement itéré d'une valeur initiale IV. On pose Z_0=**VI** et on calcule la suite Z_1, Z_2,….avec **Zi= ε_k(Zi-1).** Le clair est alors chiffré par$C_i = m_i \oplus Z_i$.
Le déchiffrement se fait de manière symétrique en échangeant les rôles de m_i et c_i.



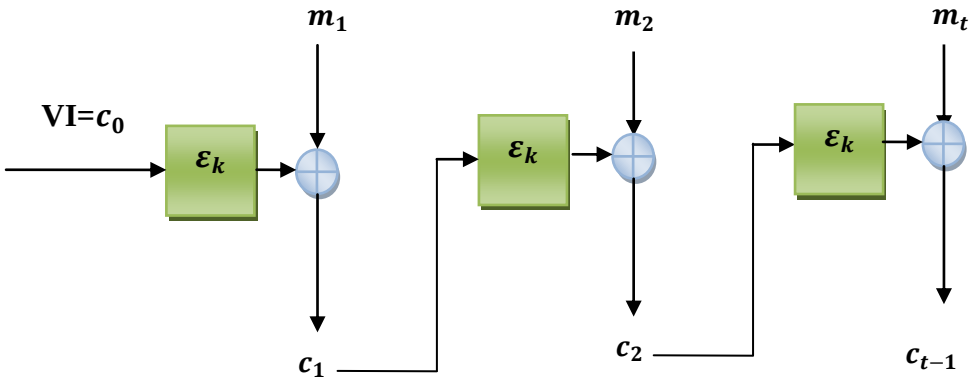
(a) Chiffrement en mode OFB.



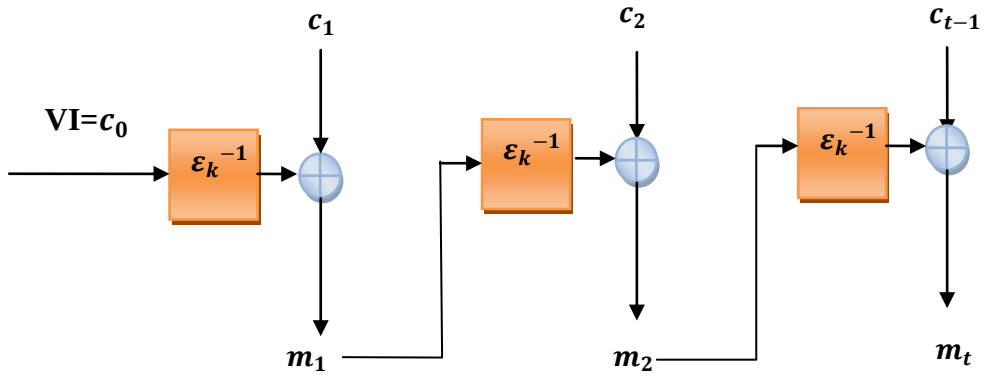
(b) Déchiffrement en mode OFB.

Figure I-8: Le mode OFB «Output Feed-Back.

CFB : On commence avec C0=IV (un bloc de 64 bits) et la clé suivante est obtenue en chiffrant le cryptogramme suivant.$Z_i = \varepsilon_k(c_{i-1})$.Sinon le reste est identique au mode OFB.



(a) Chiffrement en mode CFB.



(b) Déchiffrement en mode CFB.

Figure I-9: Le mode CFB « Cipher Feed-Back.

- **Le chiffrement par flot :**

  - **Principe de fonctionnement :**

Les algorithmes de chiffrement par flot, appelés aussi algorithmes de chiffrement en continu convertissent le plaintext (texte clair) en texte chiffré 1 bit à la fois en combinant par un XOR le flux de bits du texte clair avec un flux de bits aléatoire.

Du coté du déchiffrement, les bits du texte chiffré sont combinés un XOR avec le même flux aléatoire de bits pour retrouver les bits du texte clair.

Ce flux aléatoire peu être :

1- Une suite aléatoire entièrement secrète partagée par les deux utilisateurs.
2- Une suite pseudo-aléatoire, c'est-à-dire produite à partir d'une clé secrète.

- **Propriétés générales et avantages :**

Avec un algorithme de chiffrement par bloc, on ne peut commencer à chiffrer ou déchiffrer un message que si l'on connait la totalité du bloc. Ceci occasionne naturellement un délai dans la transmission et nécessite également le stockage successif des blocs dans une mémoire tampon. Au contraire, dans les procédés de chiffrement par flot chaque nouveau bit transmis peut immédiatement être chiffré ou déchiffré indépendamment des autres.

D'autre part, les algorithmes de chiffrement par flot contrairement aux algorithmes de chiffrement par blocs ne requirent pas de padding, c'est-à-dire l'ajout de certains bits au message clair dont l'objectif est d'atteindre une longueur multiple de la taille du bloc.

- **Contextes d'utilisation et principaux algorithmes de chiffrement :**

Les procédés de chiffrement par flot sot généralement privilégiés dans des contextes ou il est primordial de pouvoir chiffrer et déchiffrer très rapidement. Leur utilisation est par exemple systématique dans des applications qui imposent de fortes contraintes sur la taille et la consommation électrique du circuit électronique dédié au chiffrement. C'est le cas des plus part des systèmes embarqués, tels les téléphones mobiles. Nous citons ci-dessous les principaux algorithmes de chiffrement par flot :

1- ***RC4*** : C'est un algorithme de chiffrement à flot conçu en 1987 par Ron Rivest. Il est dédié aux applications logicielles, et largement déployé dans le protocole SSL/TLS et la norme WEP pour les réseaux sans fils.

2- ***A5/1 -Le chiffrement du GSM*** : A5/1 est l'algorithme à flot utilisé pour assurer la confidentialité des communications hertziennes dans la norme GSM.

3- ***E0- le chiffrement Bluetooth***: E0 est l'algorithme de chiffrement à flot utilisé pour protéger la confidentialité des communications dans le protocole de transmission sans fil de courte portée Bluetooth.

I.4.1.3 Avantages et inconvénients du chiffrement à clé secrète (symétrique) :

Ci-dessous un tableau qui récapitule les principaux avantages et inconvénients du chiffrement à clé secrète :

Avantages	Inconvénients
1. Rapidité. 2. Implémentation facile sur hardware. 3. Une clé de 64 bits ou 8 caractères qui sont facilement mémorisable.	1. Distribution des clés. 2. Nombre de clés à gérer : Pour un groupe de N personnes, il faut en tout $\frac{N(N-1)}{2}$ clés.

Tableau I-1 : Avantages et inconvénients du chiffrement symétrique.

I.4.2 Le chiffrement à clé publique (asymétrique) : [2] [3] [4] [6] [7]

Nous avons vu que les systèmes cryptographiques à clé secrète sont pratiquement sûrs et efficaces en termes de temps de calcul. Néanmoins, dès le milieu des années 70 de nouvelles interrogations furent soulevées :

-Comment convenir d'une clé lorsqu'on utilise un chiffrement à clé secrète ?

C'est en 1976, que Whitfield Diffie et Martin Hellman proposaient pour la 1ère fois une toute nouvelle façon de chiffrer, dont le but était de contourner le problème d'échange de clés en chiffrement symétrique, la cryptographie à clé publique (asymétrique) est ainsi née. Elle est fondée sur l'existence de fonctions mathématiques qui sont à la fois :

- ***A sens unique* :** C'est-à-dire qu'il est simple d'appliquer cette fonction à un message mais extrêmement difficile de le retrouver à partir du moment où on l'a transformé.
- ***A brèche secrète* :** C'est-à-dire qu'on peut l'inverser (donc retrouver le message en question), uniquement en possédant une information particulière en d'autres termes une clé.

I.4.2.1 Principe de fonctionnement

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

I. Alice crée une paire de clés asymétriques : clé privé qu'elle conserve précieusement, et une clé publique qu'elle diffuse notamment à Bob.

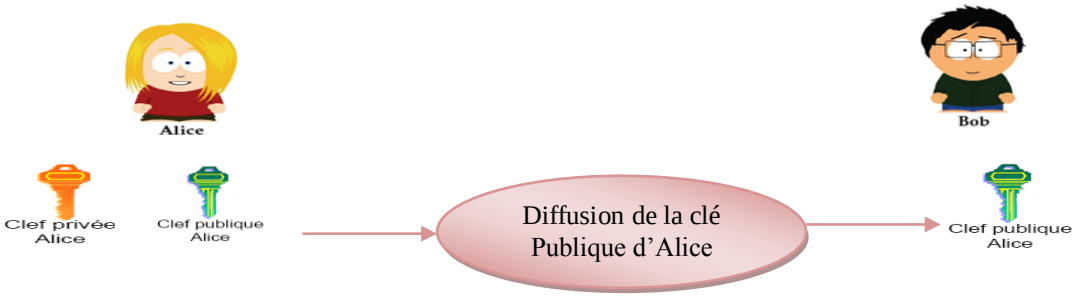


Figure I-10 : Figure illustrant l'échange de la clé publique d'Alice.

II. Bob chiffre son message avec la clé publique d'Alice.
III. Bob envoi le message chiffré à Alice.
IV. Alice reçoit le message chiffré de Bob.
V. Enfin Alice déchiffre le message avec sa propre clé privé.

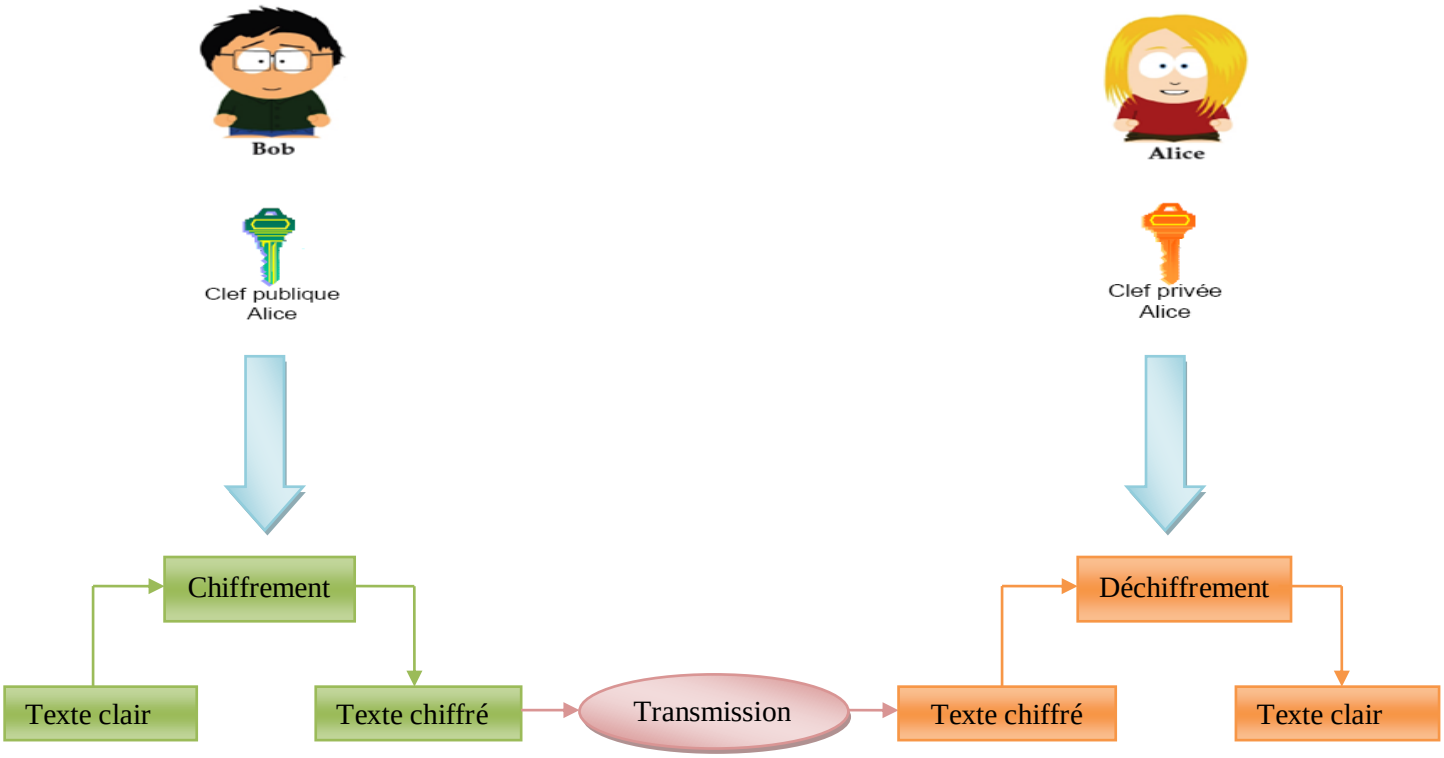


Figure I-11 : Principe d'un chiffrement asymétrique.

I.4.2.2 Applications de la cryptographie à clé publique :

En plus de la confidentialité qui est assurée par un algorithme de chiffrement, la cryptographie asymétrique offre plusieurs autres mécanismes qui sont énumérés dans les points suivants :

- ***Mécanismes d'authentification :***

  L'inconvénient qui subsiste dans l'utilisation d'un chiffrement à clé publique est le fait que la clé publique est en accès libre à toutes personnes l'a désirant. De ce fait, lorsque la personne possédant la clé privée (Alice), déchiffre les données chiffrées, elle n'a aucun moyen de vérifier avec certitude la provenance de ces données (Bob, Eve,…), On parle alors de problème d'authentification. Afin de résoudre ce problème, on utilise un mécanisme qui permet de garantir la provenance des informations chiffrées. Ce mécanisme est aussi fondé sur le chiffrement asymétrique. On l'appelle signature numérique. Le procédé est le suivant :

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

1. Bob crée une paire de clés asymétrique : Il conserve la clé privée et envoi la clé publique à Alice.
2. Alice crée une paire de clés asymétrique : Elle conserve la clé privée et diffuse la clé publique notamment à Bob.
3. Bob effectue un condensat de son message en clair puis chiffre ce message avec sa clé privée.

Le condensat des messages est obtenu grâce à une fonction de hache ou fonction de condensation, qui est une fonction qui permet d'obtenir un seul condensat ou haché d'un texte. Cette fonction doit être une fonction à sens unique pour qu'il soit impossible de retrouver le message original à partir du Haché.

Ainsi le haché représente l'empreinte digitale du message, les fonctions de hachage les plus utilisées sont :

- **MD5** (**M**essage **D**igest 5) : Développée par Rivest en 1991, il fournit une empreinte digitale de 128 bits à partir d'un texte de taille arbitraire.
- **SHA** (**S**ecure **H**ash **A**lgorithme) : Il fourni une empreinte de 160 bits, **SHA-1** une version améliorée de SHA datant de 1994 produit une empreinte de 160 bits d'un message d'une longueur maximale de 264 bits.

Le principe de fonctionnement des Hash functions cryptographiques sera détaillé plus loin dans ce chapitre.

4. Bob chiffre le message avec la clé privé d'Alice.
5. Bob envoi le message chiffré accompagné du condensat.
6. Alice reçoit la paire (message+condensat) chiffrée.
7. Alice déchiffre le message avec sa propre clé privé, à ce stade le message est lisible mais elle ne peut pas être sur que c'est bob l'expéditeur.
8. Alice déchiffre le condensat avec la clé publique de Bob.
9. Alice fait un condensat du message déchiffre et le compare avec le condensat déchiffré de Bob. Si les deux correspondent alors Bob est bien l'expéditeur, dans le cas contraire, on peut présager qu'une personne malveillante (Eve) a tentée d'envoyer un message en se faisant passer par Bob.

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

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

Si au contraire, ce dernier n'est pas l'expéditeur du message, lorsque la personne malveillante (Eve) a chiffrée le condensat, elle a utilisée sa propre clé privé, pas celle de Bob ! Ainsi le déchiffrage avec la clé publique de Bob mènera à un texte erroné et lorsque Alice le comparera à son propre condensat, elle verra qu'ils ne correspondent pas, la déduction est alors évidente « Bob n'est pas l'expéditeur du message !! ».

Le condensat des messages est obtenu grâce à une fonction de hache ou fonction de condensation, qui est une fonction qui permet d'obtenir un seul condensat ou haché d'un texte. Cette fonction doit être une fonction à sens unique pour qu'il soit impossible de retrouver le message original à partir du Haché.

Ainsi le haché représente l'empreinte digitale du message, les fonctions de hachage les plus utilisées sont :

- MD5 (Message Digest 5) : Développée par Rivest en 1991, il fournit une empreinte digitale de 128 bits à partir d'un texte de taille arbitraire.
- SHA (Secure Hash Algorithme) : Il fourni une empreinte de 160 bits, SHA-1 une version améliorée de SHA datant de 1994 produit une empreinte de 160 bits d'un message d'une longueur maximale de 264 bits.

Le principe de fonctionnement des Hash functions cryptographiques sera détaillé plus loin dans ce chapitre.

- ***Transmission sécurisée de la clé symétrique :***

La cryptographie asymétrique répond à un besoin majeur de la cryptographie à clé secrète (symétrique), qui est le partage sécurisé de la clé entre les correspondants (Alice et Bob) afin de prévenir l'interception de cette clé par une personne tierce (Eve) et donc la lecture des données chiffrée sans autorisation.

Les mécanismes de chiffrement symétriques étant moins couteux en temps de calcul ceux-ci sont privilégiés aux mécanismes de chiffrement asymétrique. Cependant toute utilisation de clé de chiffrement symétrique nécessite que les deux correspondants se partagent cette clé, Ceci peut être un problème si la communication cette clé est fait par un canal non sécurisé. Afin de palier cet inconvénient on utilise un mécanisme de chiffrement à clé publique pour la phase d'échange de la clé symétrique.

I.4.2.3 Avantages et inconvénients du chiffrement asymétrique :

Le tableau ci-dessus énumère quelques avantages et inconvénients du chiffrement à clé secrète.

Avantages	Inconvénients
1. Echanges des clés facilités. 2. Signature numérique des messages.	1. Lenteur de chiffrement et longueur des clés utilisées.

Tableau I-3 : Avantages et inconvénients du chiffrement asymétrique.

I.4.3 Fonctions de hachage cryptographiques : [2] [8]

Les algorithmes de signature classiques tels que le RSA et l'algorithme d'El Gamal, ont la fâcheuse tendance à faire croitre grandement la taille des messages signés. En effet, leur utilisation naïve suit un fonctionnement analogue au chiffrement par blocs des longues chaines de clair. On commence par découper le message à signer en blocs de même taille et on signe indépendamment chacun des blocs. Ceci pose plusieurs problèmes. Tout d'abord la taille de la signature devient très grande. Ensuite, la plupart des procédés de signatures « sûrs » sont lents. Une solution à ce problème est d'utiliser une fonction de hachage cryptographique, cette section est consacrée à la présentation du principe de fonctionnement de ces fonctions.

I.4.3.1 Définition, origines et propriétés: [4] [6] [8] [9]

Définition et origines : On définit une fonction de hachage H comme une fonction qui associe à des mots binaires de longueur arbitraire m des mots binaires de longueur n fixée appelée ***empreinte de hachage*** .

$$H : \{0,1\}^m \longrightarrow : \{0,1\}^n$$

A l'origine, les fonctions de hachage ont été créées pour faciliter la gestion de base de données. Plutôt que de manipuler des données de taille variable et potentiellement grande, on associe à ces données des empreintes de tailles fixes, sur lesquelles des comparaisons sont plus rapides à effectuer. L'utilisation de fonction de hachage permet par conséquent d'accélérer des opérations comme le tri, l'insertion ou l'accès à un élément.

Propriétés : Les fonctions de hachage ne sont pas des systèmes cryptographiques : elles ne permettent pas à elles seules d'assurer une propriété de sécurité. En particulier, elles ne constituent pas un algorithme de chiffrement. Nous verrons en section 1.2.3 quelques exemples d'utilisation de fonctions de hachage en cryptographie, mais nous pouvons d'ores et déjà souligner les propriétés suivantes des fonctions de hachage.

1- **Absence de clé.** La définition des fonctions de hachage ne fait pas intervenir de clé. Pour certaines applications, la donnée hachée est entièrement publique, et un attaquant éventuel peut effectuer lui-même les calculs de hachés.

2- **Absence d'algorithme d'inversion.** Contrairement aux algorithmes de chiffrement par blocs ou de chiffrement asymétrique, les fonctions de hachage n'ont pas besoin de pouvoir être inversées. L'impossibilité de calculer un antécédent par une fonction de hachage est même une propriété de sécurité fondamentale de ces primitives.

I.4.3.2 Sécurité des fonctions de hachage cryptographiques :

Lorsque les fonctions de hachage sont utilisées dans les bases de données, le seul écueil à éviter réside dans des collisions accidentelles : si deux entrées d'une même base s'avèrent avoir la même empreinte, elles seront considérées comme identiques. Pour éviter que cela se produise, les fonctions de hachage sont construites de manière à ce que les empreintes soient réparties uniformément sur l'ensemble d'arrivée de la fonction.
Dans le domaine de la cryptographie, les fonctions de hachage sont utilisées comme brique de base de différents mécanismes. La robustesse de ces mécanismes repose en partie sur la résistance de la fonction de hachage qu'ils utilisent à différentes attaques. Bien que d'autres chemins d'attaque soient possibles, la sécurité des fonctions de hachage est surtout évaluée dans les trois scénarios suivants.

1- **Recherche de collisions :** l'attaquant doit trouver deux messages différents M et M_0 tels que $H(M) = H(M_0)$.
2- **Recherche de secondes primages :** étant donné un message M, l'attaquant doit trouver M_0 différent de M tel que $H(M) = H(M_0)$.

3- **Recherche de préimages :** étant donné une valeur x de l'ensemble des hachés, l'attaquant doit trouver M tel que $H(M) = x$.
Les fonctions de hachages fournissent l'un des objets les plus importants pour la cryptographie moderne. Elles sont principalement utilisé dans le cadre des signatures électroniques et pour les testes d'intégrités de données dans la mesure où elles permettent d'obtenir facilement le « résumé » d'une source binaire, utilisés comme une identification « unique » de cette source.

I.4.3.3 Classification fonctionnelle des fonctions de hachage en cryptographie:

Les fonctions de hachage sont utilisées par de nombreux systèmes cryptographiques, à la fois en cryptographie symétrique et en cryptographie asymétrique. Elles en ont hérité le surnom de "couteau suisse" de la cryptographie. Nous pouvons citer les exemples suivants.

- **Codes de détection de modifications (MDC) :** qui utilisent des fonctions de hachage "sans clé " (publiques) et qui permette de vérifier l'intégrité d'une chaine binaire transmise sur un canal. l'idée est alors de joindre au message M que l'on

Souhaite transmettre l'empreinte H(M). Ainsi, lorsque le destinataire reçoit le message M, éventuellement altéré au cours du transport, il lui suffit de calculer l'empreinte H(M) du message reçu avec celle qui était jointe. Si elles diffèrent, le message a était altéré.

- **Codes d'authentification de messages(MAC) :** qui utilisent des fonctions de hachage "avec clé ", ils permettent non seulement de vérifier l'intégrité d'informations binaires transmises sur un réseau, mais également d'authentifier la source d'une donnée. Si pour calculer l'empreinte, on ajoute l'utilisation d'une valeur initiale non nulle qui joue le rôle d'une clé secrète (comme dans le fonctionnement du mode CBC décrit dans la section I-4-1-2), il est alors possible de fournir une authentification de l'origine des données en même temps qu'une garantie de l'intégrité des données transmises.

I.5 Notions de cryptanalyse. [1] [2] [4] [8] [10] [11]

La cryptanalyse ou l'attaque regroupe tous les moyens de déchiffrer un texte chiffré sans avoir connaissance de la clé. Le rôle du cryptanalyste n'a pas changé suivant les évolutions de la cryptographie, mais les moyens mis à sa disposition sont plus performants. En outre, les schémas auxquels il s'attaque sont complètement différents des simples algorithmes de substitutions utilisés dans la première ère de la cryptographie. Du point de vue de l'attaquant, on distingue cinq scénarios d'attaque différents suivant l'information que cet attaquant est capable de récupérer. Dans tous les cas, le but du cryptanalyste est de retrouver la clé utilisée pour le chiffrement des messages qu'il aura intercepté. Par attaque, on entend une tentative de cryptanalyse d'un schéma donné qui utilise de l'information dans l'un des cas suivants :

- **Chiffré seul** ***(ciphertext only).*** L'attaquant a en sa possession uniquement un ou plusieurs messages chiffrés, mais ne détient aucune information sur les messages en clair correspondants. En pratique, ce scénario est le plus courant.

- **Clair connu** ***(known Plaintext).*** Dans ce cas, le cryptanalyste a non seulement accès aux messages chiffrés, mais également aux messages en clair correspondants. On parle de paires clairs/chiffrés.

- **Clair choisi** ***(chosen Plaintext).*** En donnant plus de puissance à l'attaquant, on lui permet de chiffrer les messages qu'il souhaite avec une clef qui lui est inconnue, et son but est d'en déterminer la valeur. Bien que ce scénario avantage grandement l'attaquant, on le retrouve dans diverses implémentations pratiques telles que dans les cartes à puce. La clef secrète est protégée physiquement, mais on peut demander le chiffrement de messages par la carte et en récupérer les chiffrés.

- **Chiffré choisi** ***(chosen ciphertext).*** De la même manière, l'attaquant peut dans ce cas demander le déchiffrement de messages quelconques et obtenir leur déchiffrement par la clef secrète.

- **L'attaque par force brute ou attaque exhaustive** ***(Brute-force attack).*** L'attaquant essaye alors toutes les combinaisons de clés possibles jusque à obtention du texte clair. Cette attaque est la plus couteuse en termes de calculs et en mémoire à cause de l'attaque exhaustive. La réussite de cette attaque est contrainte au nombre de possibilités pour la clé recherchée. Plus il y aura de possibilités, plus la possibilité de trouver la bonne clé de déchiffrement est faible et l'attaque est couteuse.

La création des techniques modernes de chiffrement a fait ressortir des nouvelles méthodes de cryptanalyse. Nous pouvons alors regrouper les diverses techniques de cryptanalyse en deux grandes familles.

- **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 alors de fournir comme entrée des textes clairs avec de légères différences (un bit par exemple), on analyse en suite 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.

- **Cryptanalyse linéaire :**

Elle a été inventé par le japonais Mitsuru Matsui, elle consiste à fair une approximation linéaire de la structure interne de la méthode de chiffrement.L'idée est alors de trouver des approximations linéaires entre les bits de sortie, les bits d'entrée et les bits de la clé. Elle remonte à 1993 et s'avère étre l'attaque la plus efficace sur le DES.

I.6 Conclusion :

Ce chapitre est une introduction générale à la cryptographie, dans lequel nous avons évoqué après un bref historique des généralités sur la cryptographie qui ont présenté la terminologie propre au domaine, les différents objectifs de sécurité envisagés en cryptographie ainsi que les différentes techniques de chiffrement en les classifiant comme suit : les méthodes anciennes (Substitution, transposition et produit) et les méthodes modernes qui sont à base de la structure de Feistel.

Ensuite, nous avons définit les deux grands schémas cryptographiques classiques qui sont : en premier lieu, la cryptographie *symétrique* avec ses deux procédés de chiffrement qui sont le chiffrement par *blocs* avec ses quatre modes « *ECB*, *CBC*, *CFB* et *OFB* » et le chiffrement par *flot*, et en second lieu la cryptographie *asymétrique*. Aussi il a abordé la notion de fonction de hachage cryptographique et son contexte d'utilisation, et pour finir des notions de cryptanalyse.

Le chapitre qui suit est lui consacré à l'étude détaillée des principaux algorithmes de chaque schéma cryptographique, le DES « Data Encryption Standard » pour la cryptographie symétrique, le RSA « Rivest Shamir Adleman » pour l'asymétrique ainsi qu'une fonction de hachage SHA-1 de la famille SHA (Secure Hash Algorithme).

CHAPITRE II

ALGORITHMES DE CHIFFREMENT DE LA CRYPTOGRAPHIE CLASSIQUE.

Chapitre II

Algorithmes de chiffrement de la cryptographie classique.

II.1 Introduction

Nous mènerons dans ce chapitre une étude détaillée des principaux algorithmes de chaque schéma cryptographique classique, ainsi qu'une fonction de hachage cryptographique. On opte pour le **DES** « **D**ata **E**ncryption **S**tandard » comme algorithme de chiffrement à *clé secrète*, le RSA « **R**ivest **S**hamir **A**dleman » l'algorithme de chiffrement à *clé publique*, et la fonction de hachage **SHA**-1 de la famille **S**ecure **H**ash **A**lgorithme.

II.2 Algorithme de chiffrement symétrique. Le DES « Data Encryption Standard »

II.2.1 Historique.

Le DES est l'un des systèmes de chiffrement les plus utilisés. Il a été conçu dans les années 1970 par la firme américaine IBM pour être un système de chiffrement par blocs à clé secrète, et fut adopté comme standard de chiffrement (FIBS 43) US en 1977.

Il a été construit pour fonctionner aussi bien de façon logicielle que matérielle, il traite des messages de 64 bits avec des clés de 56 bits, et il fournit un cryptogramme de 64 bits en sortie. Le DES utilise la structure de chiffrement itératif de Feistel, dont nous avons vu le principe dans la section **I.3.3.2.**

II.2.2 Principe de fonctionnement : [02] [03] [06] [10] [12]

Le DES consiste alors à réitérer la structure Feistel 16 fois (rounds). La figure II-1 ci-dessous représente un schéma global du principe de fonctionnement du DES.

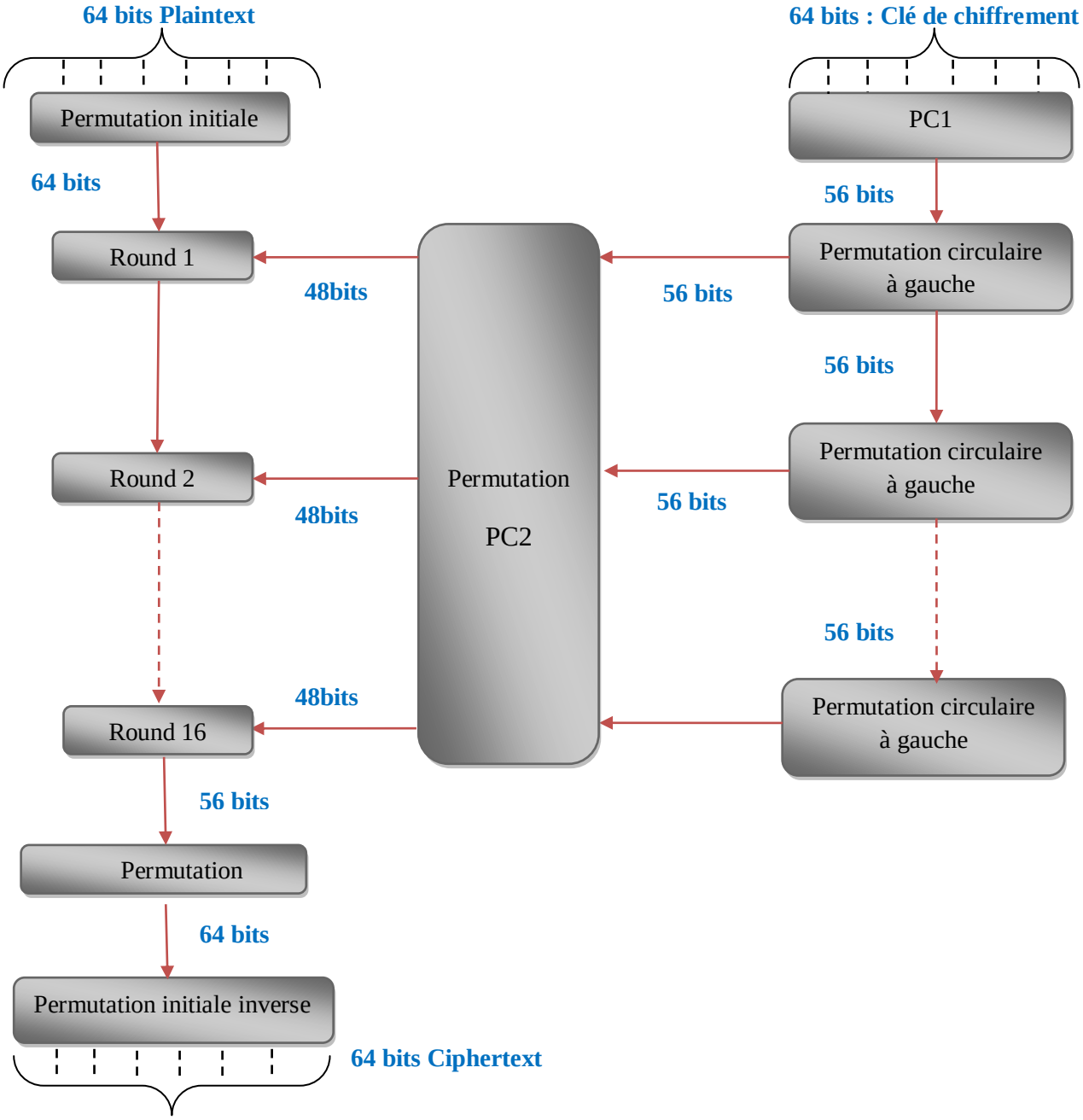


Figure II- 1: Schéma global illustrant le principe de fonctionnement du DES.

II.2.2.1 Algorithme de chiffrement.

L'algorithme de chiffrement du DES fait subir comme illustré dans la figure II.1 un traitement au plaintext et à la clé de chiffrement. Nous exposant dans ce qui suit le détaille de ces traitements.

- **Traitement sur le plaintext :**

On peut voir que le traitement du plaintext est en 3 phases. En premier il subit une permutation initiale (PI), ce qui produit les entrées permutées. Ceci est suivi de 16 itérations avec la même fonction F qui comporte les deux fonctions, de substitution et de permutation suivant la règle :

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus F(R_{i-1}, K) \end{cases}$$

A la sortie du 16^{éme} round R_{16} et L_{16} sont échangées pour produire les prés sorties. Finalement, on applique aux prés-sorties (PI^{-1}), une permutation finale.

Nous détaillons à travers le schéma de la figure I-11 la structure interne d'un seul round de l'algorithme du chiffrement du DES.

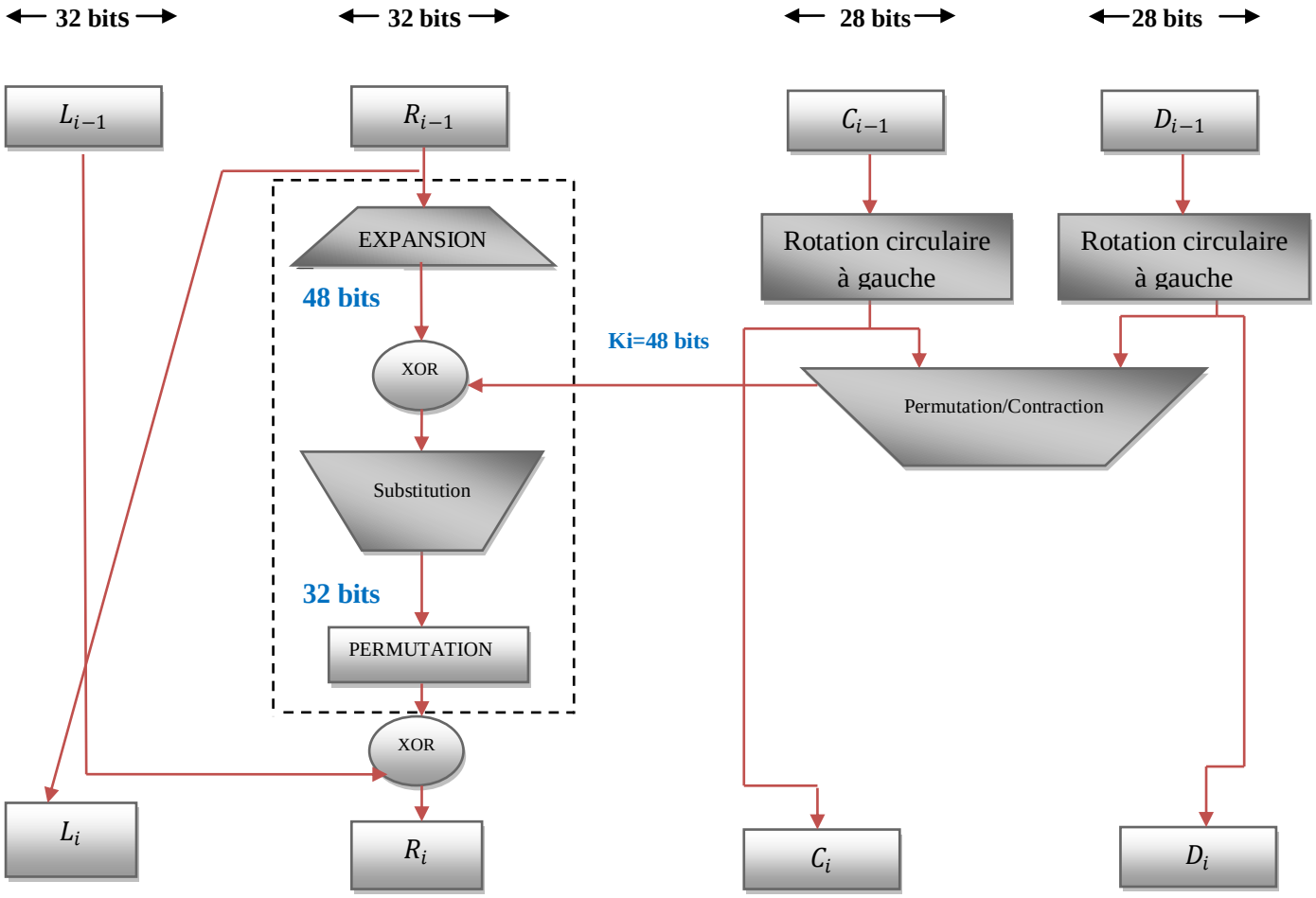


Figure II-2 : Structure interne d'un seul round du DES.

Le DES fait appel a une fonction **F** qui prend en entrée deux arguments :

1- Une chaine de 32 bits qui est la partie droite du plaintext (R_i).
2- Une chaine de 48 bits qui est la sous clé de chaque round (K_i).

Le résultat de cette fonction est une chaine de 32 bits. Le calcul se déroule en plusieurs étapes illustrées dans la figure II-3, et interprétées ci dessous.

Etape1 : R est augmenté en une chaine de 48 bits par une fonction d'expansion E.

E (R) est alors composée de tout les bits de R dans un certain ordre et 16 d'entre eux apparaissent deux fois. La fonction d'expansion est décrite dans le tableau 2-1.

Etape2 : B=E(R) $\oplus$ K est alors calculé. B est ensuite découpé en 8 sous chaine consécutives : $B=B_0, B_1,\ldots, B_8$.

Etape 3 : Cette étape utilise huit S-BOX, qui sont des tableaux de 4 ×16 entiers compris entre 0 et 15.Soit une sous chaine Bi donnée par $Bi=b_1b_2b_3b_4b_5b_6$.On calcule une chaine de 4 bits Si(Bi) de la façon suivante.

- Les bits b_1 et b_6 fournissent la représentation binaire de l'indice l de la ligne de Si à considérer **: $0 \leq l \leq 3$**.
- Les bits $b_2b_3b_4b_5$ fournissent la représentation binaire de l'indice c de la colonne S_i à utiliser **: $0 \leq c \leq 15$.**
- L'intersection de la ligne l et de la colonne c de Si fournit un entier compris entre 0 et 15, donc une sous chaine de 4 bits, qui sera le résultat $\mathbf{C_i=S_i(B_i)}$. On calcule ainsi $\mathbf{\{C_j=S_j(B_j)\}_{1\leq j\leq 8}}$.

Etape 4 : La chaine $C=C_1C_2\ldots\ldots C_8$, de longueur 32 bits, est alors réordonnée suivant une permutation fixée P détaillée dans (TAB …).

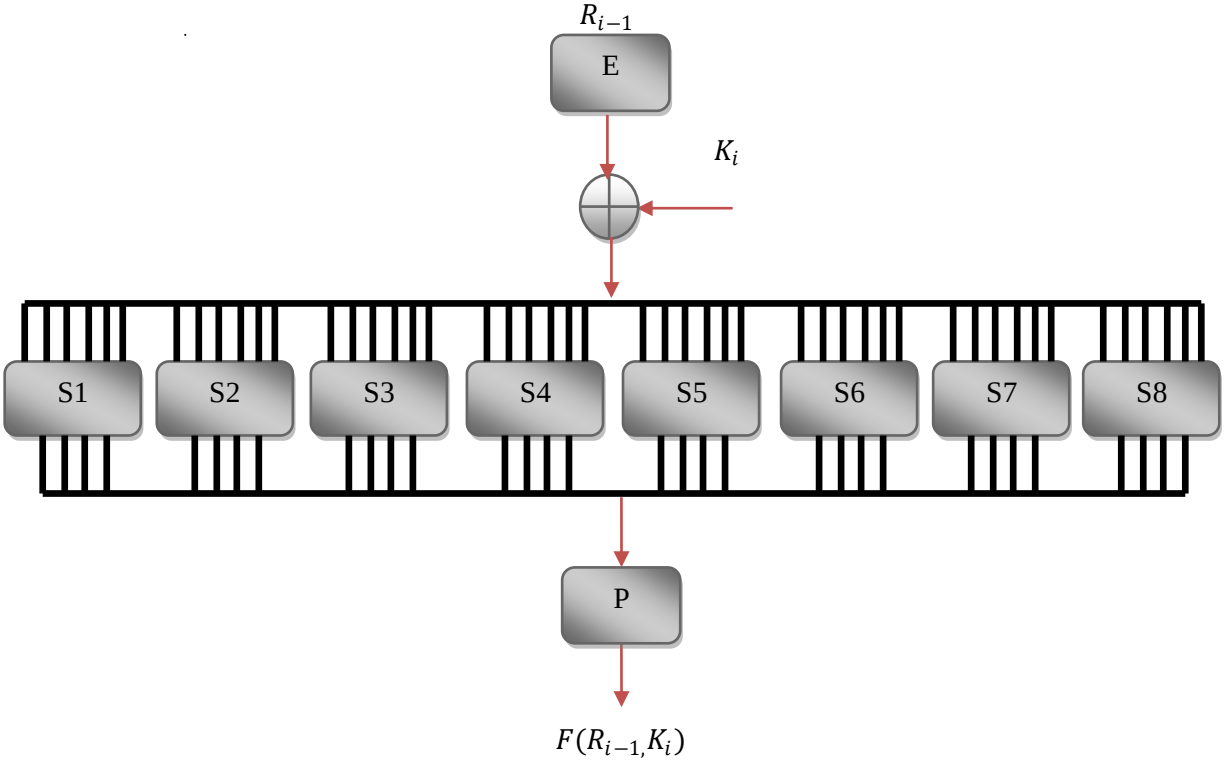


Figure II.3 : Calcul de $F(R_{i-1},K_i)$

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Permutation initiale

IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Permutation initiale inverse

E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	23	30	31	32	1

Fonction d'expansion

P

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Permutation

PC1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Permutation choix 1

PC2

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	12
4	41	52	31	37	47	55	30
40	51	45	33	48	44	49	39
56	34	53	46	42	50	36	29

Permutation choix 2

Figure II.4 : Détaille des différentes permutations du DES.

S1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	3	8	
4	1	14	8	13	6	2	11	15	12	9	7	3	5	0	
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figure II.5 : Définition des S-BOX de l'algorithme du DES.

- **Traitement sur la clé de chiffrement :**

la clé qui est initialement de 64 bits est transformée en une clé de 56 bits par une permutation PC1, les 8 bits restants sont ainsi utilisés pour tester l'intégrité de la clé entre autres, puis pour chacun des 16 rounds, une sous clé de 48 bits (Sub Key) est générée en utilisant une translation circulaire vers la gauche et permutation PC2.

On a vu que de second argument de la fonction F pour le tour i est une sous clé Ki extraite de la clé principale K. Ce paragraphe détaille l'algorithme utilisé. La clé K est une chaine de 64 bits, dont 56 seulement définissent la clé. Les huit autres bits, ceux situés en position 8,16,…. 64 sont des bits de parité qui servent à tester l'intégrité de la clé. Ces bits sont ignorés lors de la diversification de la clé. Voici l'algorithme utilisé pour chaque round i

- ✓ A partir de la clé de 64 bits, on enlève les bits de parité. On obtient ainsi une clé K' de 56 bits. On ordonne les bits de K' selon une permutation initiale PC-1.On note PC-1(K')=$C_0.D_0$, ou C_0 est composé des 28 premiers bits de PC-1(K') et D_0 des 28 restant.

- ✓ On calcul C_i, D_i et K_i pour $1 \leq i \leq 16$ de la façon suivante :

$$\begin{cases} C_i = LS_i\ (C_{i-1}) \\ D_i = LS_i\ (C_{i-1}) \\ K_i = PC\text{-}2(C_i, D_i) \end{cases}$$

Où LSi est une rotation circulaire vers la gauche d'une ou deux positions selon la valeur de i :

$$\begin{cases} \text{Si } i \in \{1, 2, 9, 16\} \text{ on décale d'une position.} \\ \text{Sinon on décale de deux positions.} \end{cases}$$

PC-2 est une autre permutation des bits de la chaine.

II.2.2.2 Algorithme de déchiffrement.

Le déchiffrement est effectué par le même algorithme en inversant l'ordre d'utilisation des clés de chaque round, autrement dit utiliser K_{16} pour la 1ère itération du déchiffrement et K_1 à la 16éme. .

Discussion :

Le DES est très efficace, et particulièrement bien adapté aux implémentations matérielles, La structure de Feistel est très rapide dans les deux sens (Chiffrement et déchiffrement).

Apres de nombreuses attaques sur le DES, l'Electronic Frontier Foundation[1] exhibe en juillet 1998 une machine qui crack le DES en 9 jours maximum, par la recherche exhaustive des 2^{56} clés possibles. Ainsi la taille des blocs (64 bits) est devenue courte pour assurer une robustesse suffisante.

[1] EFF : Association Américaine de défense et de promotion d'utilisation d'internet.

En janvier 1997, un appel à candidatures international est lancé afin de trouver un nouveau standard de chiffrement, 15 algorithmes sont alors reçus. Apres trois ans d'évaluation, soit en l'an 2000 l'AES (Advanced Encryption standard) fut adopté par le NIST[2] afin d'être le nouveau standard de chiffrement pour les organisations du gouvernement des Etats- Unis, après que le DES, standard depuis des années, devenait obsolète.

L'AES est algorithme de chiffrement symétrique, il utilise des clés de 128, 192 ou 256 bits, il travaille avec des blocs de 128 bits, il est compatible avec les deux modes de chiffrement par blocs : ECB et CBC.

II.2.3 Exemple de chiffrement DES.

Cette section introduit un exemple de chiffrement du DES sur un texte, en utilisant les deux modes de chiffrement ECB et CBC. La clé est générée aléatoirement

Avec le mode ECB :

Texte clair :

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

Texte chiffré :

l$ìã1 E þÓ¡ • Åã(òO6E×.Ql$ìã1 E > Pk´üpè¶}p□Çr§ÜÇ HÛûçô®Ø¥ÓwgÎIÜcðÈãé$j • ¨Wõ§=Å 1ÛHÔ =ÚÇj>c6ßïïðÓYYÓ`uv=ÎÂ,vò¨Hl8g¢\e>Úº • 4zêäü\Àl±&k8Â@ • cÛ • -fÒÎà¼ • ³ÍoÙÎÀ:§ê2`=öç,`âÐ6Ü/Ø XÍûÑ • éì·6?æän«Â·z0ÖÎ.³¿Sqî1)ZÎ{rQ • R>Æ • ¤éööû?¿op¾¢ÏË¿Â[?ýk^¡E1%àñì·Òµ¨Ð<÷Å]B%°é¸-Ócâ< gùÝjJ • Ä • Òå-Ð¿o¨?ÔÚðÙ%ÀòÐH;5:¼_ÄÄayvû`¹®Á'mèyE$Ý}?lª>&"ÒdbëSä • • [Òœ+¼0T'œ}7Ürüª:à à1ùè`e<"µbsá¯1§8,ÿp»D • kªZ_´ü0ì¸XbÜö?VÍ¯µ;ùÏÒ<ÏDJÑt8tÝº

Texte déchiffré :

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

[2] NIST : National Institute of Standards and Technology.

Discussion :

On peut aussi constater que notre algorithme chiffre et déchiffre correctement le texte. L'exemple met en évidence ; l'inconvénient du mode ECB soulevé lors de l'introduction des différents modes de chiffrement dans la section **I-4-1-2**, on constate alors que les blocs homogènes dans le texte sont chiffrés de la même façon, ceci constitue alors une faille que les cryptanalystes peuvent exploiter afin de récupérer de l'information.

Avec le mode CBC :

Texte clair :

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

Texte chiffré :

‚[c×öóëóY„ÖxK=QÝÆs@Yç¹!-Þ"¼K'@9÷Eµ[rdSo=rÛ÷ÕÉÚ¢ • ¬ÕíL·3IßÔfý5A]n#kctuú+'p6-
• ÈYV?b7:ìñóÍ÷Í&ïóR1ªnIç¬Ìõkâ Ã5XüÈA¿E6ËÓ±ôpÕÌ1©[]Æh¥N$û) ÒE¾ÐP±µÎ¦mÂ¹Ð®úä)oD
ù²´)«fe-$³vJYÀó¡êD,¿Ibf[7*Èpï³|äþð'}Ç=AÔ>¢b> • '[*□-
'À>6£1Æ]©ìæ&çÅh:'cb"ãÚËæyÒ<K8Â~#°ÿZUÌÞ§ÓóôÀÑ¤u,¶-ÿÃCmH • X>! • ,Ë SË)S)
<:Ã¨Ù²Î=áÜ¿ît|¿©½2áe:Êí«1ÕAb3¢]R@Mwjµ°åtQ¥ • RÀëÏ/»üÓÿ¯$□Ãmª»Pn´kP¶[E°í.3EðBúÙ¡
S{O:ÿæ0ê

Texte déchiffré :

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

Discussion :

A partir des résultats ci-dessous, on peut dire que notre algorithme chiffre et déchiffre correctement le texte. On note aussi que les mêmes blocs ne sont pas chiffrés de la même manière avec le mode CBC, ce qui nous permet alors de dire que ce mode résout le problème du mode ECB.

I.3 Algorithme de chiffrement asymétrique

« Le R-S-A : Rivest Shamir Adleman ».

I.3.1 Historique :

Le **R-S-A** : du nom de ses concepteurs, Ron **R**ivest, Adi **S**hamir et Léonard **A**dleman est le premier algorithme de chiffrement asymétrique. Il a été découvert en 1978 au MIT.

Il est devenu un système universel servant dans une multitude d'application : systèmes d'exploitation, cartes à puces bancaires, et bien sur le réseau internet pour assurer la confidentialité des courriers électroniques et authentifier les différents utilisateurs, ainsi que la protection de dossiers hautement confidentiels.

Le RSA est basé sur la théorie des nombres entiers, et sa robustesse tient du fait qu'il n'existe aucun algorithme de décomposition d'un nombre en facteurs premiers. Alors qu'il est facile de multiplier deux nombres premiers, il est très difficile de retrouver ces deux entiers si l'on connaît le produit.

I.3.2 Principe de fonctionnement du RSA : [02] [03] [04] [06] [07]

Le RSA est un algorithme de chiffrement à clé publique, ce qui signifie que l'algorithme de calcul n'est pas caché, ni la clé de chiffrement (appelé de ce fait clé publique). La connaissance permet à tous les émetteurs de chiffrer des messages qui ne pourront être déchiffrés que par le destinataire, grâce à sa clé secrète.

I.3.2.1 Génération des clés

La génération de la paire de clé (publique, secrète) citées dans le principe de fonctionnement se fait en suivant les étapes ci-dessous :

1. On choisit deux entiers assez grand p et q de l'ordre de 10^{100}.
2. On fixe $n = pq$.
3. On calcule$\varphi(n) = (p-1)(q-1)$, l'indicatrice d'Euler.
4. On choisit e tel que le PGCD(e, $\varphi(n)$) =1.
5. On calcul d tel que d.e≡1 mod $\varphi(n)$.

Alors :

La clé publique : c'est le couple [n, e].

La clé secrète : c'est le couple [n, d].

I.3.2.2 Exponentiation modulaire :

Définition : Pour implémenter le RSA, on a besoin d'un algorithme l'exponentiation modulaire rapide pour calculer $\boldsymbol{a^b mod\ n}$. On présente ci-dessous un algorithme qui permet de réaliser cette tache.

1. Diviser b en puissances de 2 en l'écrivant en binaire.
2. Commencer du chiffre le plus à droite, soit k=0 pour chaque chiffre :
   - Si le chiffre est 1, nous gardons le terme 2^k, sinon nous ne le gardons pas.
   - Incrémenter k, et on analyse le chiffre suivant.
3. On calcule le modulo n des puissances de deux ≤b.
4. Utiliser les propriétés de multiplication modulaire pour combiner les valeurs calculées du modulo n.

I.3.2.3 Chiffrement :

Le chiffrement d'un message M se fait comme suit :

1. Le message original doit être décomposé en une série d'entiers M_i de valeurs comprise entre 0 et n-1.
2. Pour chaque entier M_i il suffit alors de calculer C_i le message chiffré avec : $\boldsymbol{C = M^e\ mod\ n}$, en utilisant la méthode d'exponentiation modulaire introduite en haut.
3. Le message chiffré est alors la concaténation des entiers C_i.

I.3.2.4 Déchiffrement :

1. Conformément à la manière dont il a été chiffré, le message reçu doit être composé d'une succession d'entiers de valeurs comprises entre 0 et n-1.
2. Pour chaque entier C il faut calculer $\boldsymbol{M = C^d mod\ n}$.
3. Le message original peut alors être reconstitué à partir de la série d'entiers M.

II.3.3 Exemple de chiffrement RSA.

Nous introduisant un exemple qui illustre le chiffrement et le déchiffrement d'un message avec l'algorithme du RSA.

Bob veut envoyer le message suivant à Alice M=« BONJOUR ALICE », alors Alice doit au préalable construire ses clés publique et privée.

Soient p=3011 et q=3037, en suivant les étapes décrites en haut on aboutit aux résultats suivant :

1. n=p.q=9144407.
2. Φ(n)= (p-1) (q-1)=9138360.
3. e=13.
4. d=6326557.

La clé publique est le couple [9144407, 13].

La clé secrète est le couple [9144407,6326557].

- **Chiffrement :**

Apres avoir récupérer la clé secrète d'Alice Bob converti son message en nombre, avec l'équivalent ASCII des lettres utilisées, alors M devient M'=6679787 4798582 3265767 36769.

Le message M doit être inferieure à n-1, alors il faut décomposer d'abord le message en blocs inférieur à n-1.

Alors M' devient alors : m1=6679787, m2=4798582, m3=3265767, m4=36769.

Bob chiffre ensuite les blocs obtenus par la relation $\boldsymbol{C = M^e \bmod n.}$

On obtient alors les résultats suivants :

$\boldsymbol{C_1 = m_1^e \bmod n = 6679787^{13} \bmod 9144407 = 8215706.}$

$\boldsymbol{C_2 = m_2^e \bmod n = 4798582^{13} \bmod 9144407 = 3156858.}$

$\boldsymbol{C_3 = m_3^e \bmod n = 3265767^{13} \bmod 9144407 = 9031440.}$

$\boldsymbol{C_4 = m_4^e \bmod n = 36769^{13} \bmod 9144407 = 4275022.}$

Bob envoi alors le message suivant à Alice : C=8215706315685890314404275022.

- **Déchiffrement :**

Pour déchiffrer le message envoyé par Bob, Alice procède de la manière suivante :

Elle utilise sa clé secrète [n, d] et applique la relation de déchiffrement suivante :

$\boldsymbol{M = C^d \bmod n,}$ elle aboutit alors aux résultats suivants :

$\boldsymbol{m_1 = C_1^d \bmod n = 8215706^{6326557} \bmod 9144407 = 6679787.}$

$\boldsymbol{m_2 = C_1^d \bmod n = 3156858^{6326557} \bmod 9144407 = 4798582.}$

$\boldsymbol{m_3 = C_1^d \bmod n = 9031440^{6326557} \bmod 9144407 = 3265767.}$

$\boldsymbol{m_4 = C_1^d \bmod n = 4275022^{6326557} \bmod 9144407 = 36769.}$

Alice récupère alors le message suivant : 667978747985823265767 36769.

Message qu'elle convertit en caractères, ce qui correspond alors au message suivant : « BONJOUR ALICE ».Alice à donc pue récupérer le message de Bob.

II.4 La fonction de hachage SHA-1.

II.4.1 Historique :

Les fonctions de hachage cryptographiques les plus utilisées en pratiques appartiennent aux familles MD et SHA, dont les membres les plus connus sont les fonctions MD5 et SHA-1.
SHA Secure Hash Algorithm) a été conçu par la NSA et publié par le NIST. En mai 1993, une première version SHA-0 voit le jour, 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 section qui suit est consacrée à l'étude du principe de fonctionnement de la fonction de hachage SHA-1.

II.4.2 Principe de fonctionnement : [02] [03] [08] [09]

SHA-1 suit la construction de Merkle-Damgard[3] représentée ci-dessous

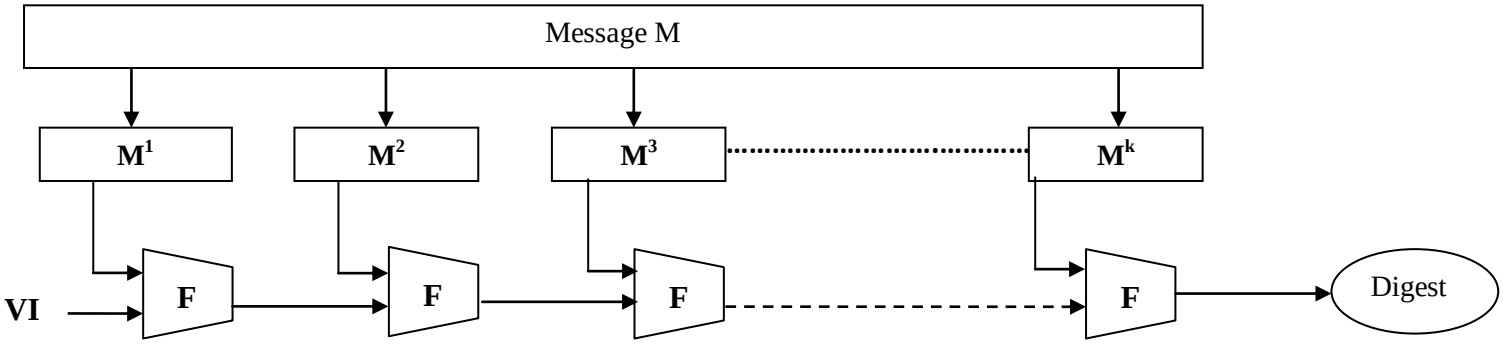


Figure II.6 : Construction de Merkle-Damgard

SHA-1 utilise les différentes représentations suivantes : la représentation de base qui est la représentation hexadécimale qui code des mots binaires de longueur 4 comme par exemple, 7_H =0111 ou A_H = 1010 ; ces entiers hexadécimaux sont en suite regroupés par mots de longueur 8 pour donner un mot de 32 bits qui représente le codage binaire d'un entier compris entre 0 et 2^{32} -1. **SHA-1** travail sur des blocs de 512 bits représentés comme 16 mots de 32 bits.

Exemple : 1010 0001 0000 0011 1111 1110 0010 0011=$A103FE23_{H.}$

La fonction SHA-1 travaille suivant les étapes suivantes :

SHA-1 commence par effectuer un complément du message initial. A l'issue de cette opération, on obtiendra un mot binaire de n blocs de 512 bits, SHA va alors travailler sur chacun des blocs.

[3] Merkle: *cryptographe américain et chercheur en nanotechnologie. L'un des pionniers de la cryptographie asymétrique.*
Damgard : *cryptologue danois et professeur à l'université d'Aarhus(Danemark).*

Soit $x \in \{0,1\}^*$ et L sa longueur en binaire avec $L<2^{64}$ sur 64 bits.

- On complète x avec un 1 en queue pour obtenir le mot u=x1.
- On ajoute m zéros pour que la longueur du mot $w=x0^mL$ soit un multiple de 512, soit $w \equiv 0 \mod 512$.

Exemple : Si le message original à hacher de longueur 40 est :

01100001 01100010 01100011 01100100 01100101.

On commence par ajouter le 1 final :

01100001 01100010 01100011 01100100 01100101 **1.**

On ajoute en suite 512-64-(40+1)=407 zéros à la fin du message pour obtenir le message suivant en hexadécimal :

61626364 65800000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000

On ajoute ensuite au message sa longueur en binaire L exprimée en hexadécimal.

Dans notre cas la longueur est 40 soit 28 en hexadécimal :

61626364 65800000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000028

A l'issue de ce complément, le message sera composé de n blocs de 512 bits (ou de manière équivalente, en n paquets de 16 mots de 32 bits) M_0 M_1 M_{n-1}.

Pour chaque bloc de 512 bits SHA-1 calcule 80 itérations appelées aussi rounds. Elles sont regroupées en quatre parties principales avec 20 transformations dans chaque une d'entre elles. D'autre part un vecteur d'initialisation sur 160 bits fourni 05 mots de 32 bits A, B, C, D en attaquant directement le premier round, Leurs valeurs initiales pour le premier hachage sont définies par la spécification :

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

- $A=67452301_H$
- $B=EFCDAB89_H$
- $C=98BADCFE_H$
- $D=10325476_H$
- $E=C3D2E1F0_H$

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

- Elles travaillent sur 3*32 bits et fournissent un résultat sur 32 bits.
- **Pour le 1er groupe de 20 rounds** : $0 \leq m \leq 19$, Avec : m étant le numéro d'un round

  $f1 = f(m, B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$; ($\neg$: complément binaire).
- **Pour le 2 ^{éme} groupe de 20 rounds** : $20 \leq m \leq 39$, $f_2 = f(m, B, C, D) = B \oplus C \oplus D$
- **Pour le 3ème groupe de 20 rounds** : $40 \leq m \leq 59$, $f_3 = f(m, B, C, D) = (B \wedge C) \vee (B \wedge D) (C \wedge D)$
- **Pour le 4ème groupe de 20 rounds** : $60 \leq m \leq 79$, $f_4 = f(m, B, C, D) = B \oplus C \oplus D$

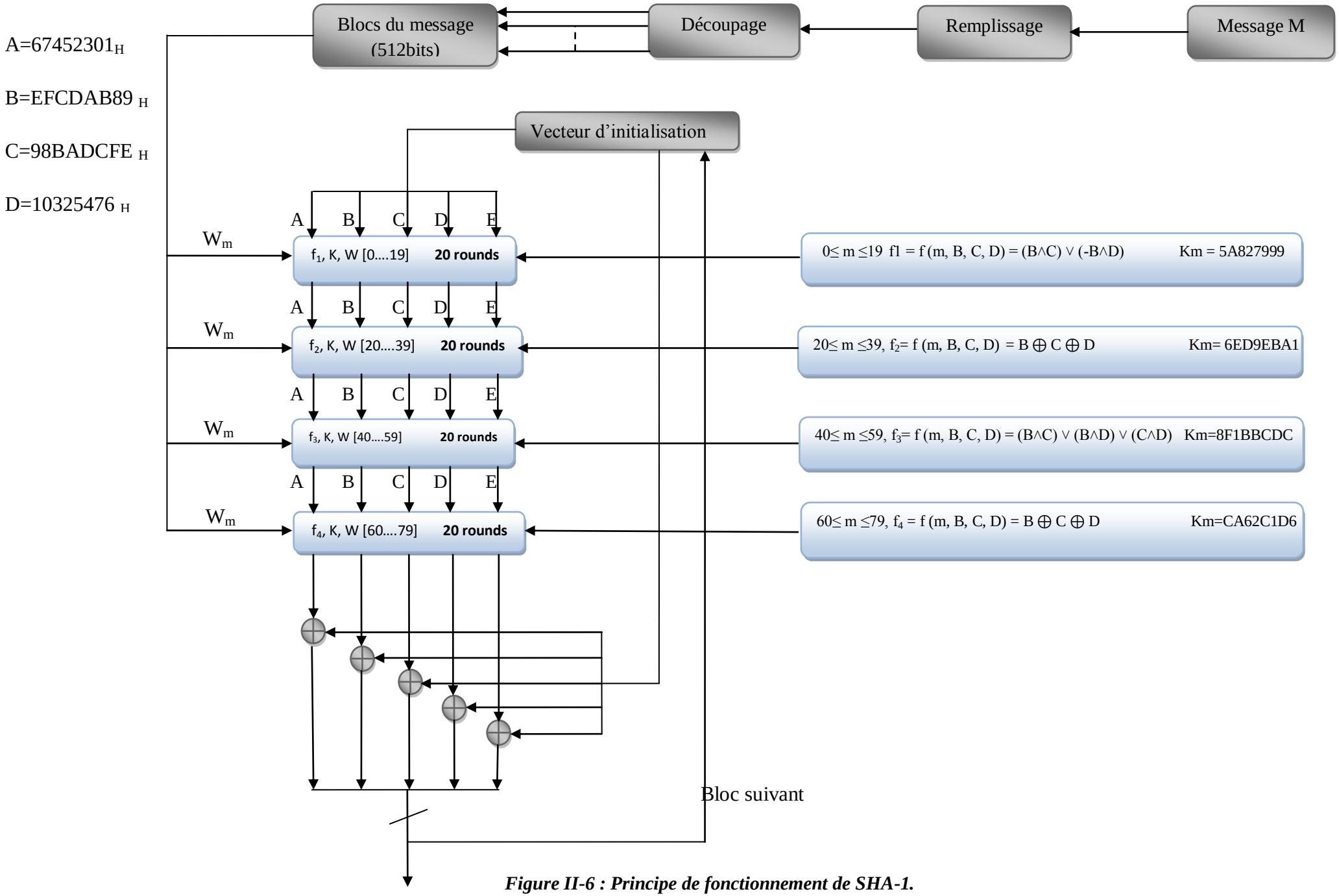


Figure II-6 : Principe de fonctionnement de SHA-1.

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

Pour créer ces mots supplémentaire, on applique un ou-exclusif entre quatre mots déjà présents dans la liste : **W[m] = W [m-3] ⊕W [m-8] ⊕ W [m-14] ⊕W [m-16]** c'est-à-dire :

- Pour $0 \leq m \leq 15$: W_m prend les 16 premières valeurs de 512 bits.
- Pour $16 \leq m \leq 19$: W_m= W[m] = W [m-3] ⊕W [m-8] ⊕ W[m-14] ⊕ W [m-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.

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

A chaque round 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″ correspond à cette ronde ainsi qu'une constante Km définie par le standard qui varie selon les tours (groupes), les valeurs de Km en hexadécimal sont définies comme suit :

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

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

(A, B, C, D, E)⟵(E +f (m, B, C, D) + (A<<5) + Wm + Km), A, (B<<30), C, D)

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

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 5 variables de 32 bits, on obtient l'empreinte de 160 bits. Si d'autres blocs doivent être hachés, cette empreinte intermédiaire est passée au bloc suivant en tant que vecteur d'initialisation.

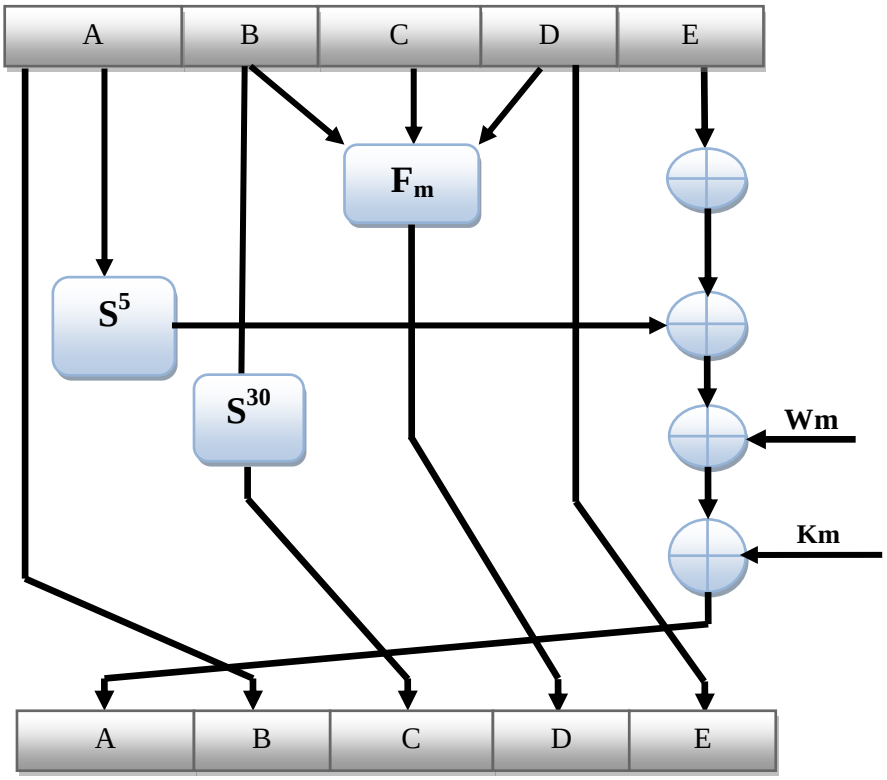


Figure II-7 : Fonction de round de SHA-1.

Récapitulatif :

1. complémentation du message initial
2. Découpage
3. Initialisation : initialiser 5 buffers de 32 bits (=160bits) : A, B, C, D, E
4. Calcul itératif : 4 rondes de 20 itérations chacune. Ces rondes ont une structure similaire mais utilisent des fonctions primitives différentes (f1, f2, f3, f4). Le SHA utilise des constantes additives Km ($0 \leq m \leq 79$). 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)⟵(E+f (m, B, C, D) + (A<<5) + Wm+Km), A, (B<<30), C, D)

où A, B , C , D, E se rapportent aux 5 registres, m est le numéro de l'étape, Wm est dérivé du bloc de message (32 bits) et Km est une valeur additive.

5. Le condensé final qui constitue le condensé attendu.

II.5 Conclusion.

Dans ce chapitre, nous avons mené une étude détaillée sur trois algorithmes : Le **DES** « **D**ata **E**ncryption **S**tandard » un des plus importants algorithmes de chiffrement à *clé secrète* dont nous avons tâché d'expliquer le principe de fonctionnement, l'étude de cet algorithme a été appuyée par un exemple de chiffrement DES sur un texte avec deux modes de chiffrement ECB et CBC. Exemple à travers duquel, nous avons mis en évidence en plus de l'efficacité notre algorithme de chiffrement et de déchiffrement, l'inconvénient majeur du mode ECB et la réussite du mode CBC à pallier à ce problème. La seconde étude a été menée sur l'algorithme à *clé publique* le plus populaire actuellement à savoir le **RSA** du nom de ses concepteurs **R**ivest **S**hamir et **A**dleman, algorithme dont nous avons expliqué les différentes étapes de fonctionnement en illustrant chaque étape par un exemple concret de chiffrement RSA. La troisième et dernière étude concernait la fonction de hachage **SHA-1** de la famille **S**ecure **H**ash **A**lgorithme.

Les trois algorithmes étudiés dans ce chapitre seront associés afin de concevoir notre cryptosystème hybride, ceci fera l'objet de notre prochain chapitre.

CHAPITRE III

CONCEPTION DU
CRYPTOSYSTEME HYBRIDE.

Application a la transmission d’images.

Chapitre III

Conception du cryptosystème hybride.
Application à la transmission d'images.

III.1 Introduction :

Les trois algorithmes abordés dans le chapitre II (DES, RSA et SHA-1) servent de support théorique pour ce troisième chapitre, qui est composé de deux parties. Dans la première, nous allons présenter une méthode de chiffrement dite ***hybride*** qui associe les trois algorithmes étudiés au chapitreII que nous allons appliquer à la transmission d'images. La seconde partie seraconsacrée à l'étude des performances, et à l'analyse de la sécurité du cryptosystème conçu.

III.2 La cryptographie hybride [4] [10] [15]:

III.2.1 Principe :

Un cryptosystème hybride consiste à utiliser les avantage des chiffrements symétrique et asymétrique tels que :

- La rapidité d'un système symétrique.
- La possibilité de transmettre la clé secrète par un cryptosystème asymétrique.
- Assurer l'intégrité des données échangées grâce à des fonctions de hachage cryptographiques qui permettent de générer des empreintes numériques des données envoyées et reçues afin de les comparer.

La plus part des cryptosystèmes hybrides procèdent de la manière suivante :
Une clé aléatoire est générée pour l'algorithme symétrique. L'algorithme de chiffrement symétrique est ensuite utilisé pour chiffrer le message. Une empreinte numérique du message à envoyer est générée grâce à une fonction de hachage, cette dernière est chiffrée avec la clé secrète de l'expéditeur. La clé aléatoire quant à elle, se voit chiffrée grâce à la clé publique du destinataire. Il suffit ensuite d'envoyer le message chiffré avec l'algorithme symétrique, accompagné de la clé et l'empreinte chiffrées correspondante. Le destinataire déchiffre la clé symétrique avec sa clé privée et via un déchiffrement symétrique retrouve le message.
Il génère une empreinte numérique du message déchiffré qu'il comparera ensuite à l'empreinte reçue une fois déchiffrée grâce a la clé publique du destinataire.

III.2.2 Choix des algorithmes

III.2.2.1 Chiffrement symétrique :

Pour la cryptographie symétrique, nous avons opté pour l'algorithme du DES « **Data Encryption Standard »**, qui comme nous avons vu au chapitre I est un chiffrement par blocs, il traite des messages de 64 bits avec des clés de sessions qui sont générées aléatoirement de taille 56 bits, et fournit un cryptogramme de 64 bits en sortie.

III.2.2.2 Chiffrement asymétrique :

Afin de permettre un transport sécurisé de la clé de session, un mécanisme de chiffrement à clé publique est privilégié, celui utilisé dans notre cryptosystème est « Le **R-S-A** : **R**ivest **S**hamir **A**dleman ».

III.2.2.3 Fonction de hachage :

Afin d'assurer les autres objectifs de sécurité qui sont l'intégrité des données reçues, l'authentification de l'expéditeur et sa non répudiation, une fonction de hachage est utilisée, notre choix s'est porté sur SHA-1, la fonction détaillée au chapitre I.

III.3 Le cryptosystème de transmission :

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

III.3.1 Bloc émetteur :

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

1- Une clé K de taille 56 bits est générée aléatoirement, celle-ci fera office de clé de session.

2- Une fois la clé générée, K est utilisée lors du chiffrement de l'image avec l'algorithme DES. Nous choisissonsun des deux modes ECB ou CBC.

3- Afin de transmettre la clé de session d'une manière sécurisée, l'émetteur chiffre cette clé en utilisant l'algorithme à clé publique RSA de la manière suivante :

- L'émetteur qui est en 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 clé publique du récepteur ce qui lui permet de générer la clé K'' qui va ensuite transmettre à travers le canal. Cette procédure de chiffrement assure l'authenticité de la clé et seul le récepteur peut alors déchiffrer l'image envoyée.

4- La 4éme et dernière étape consiste alors à générer une empreinte numérique de l'image originale 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.

III.3.2 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 du DES.

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 RSAen 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 des deux empreintes implique que l'intégrité des données reçues est vérifiée, l'expéditeur de l'image authentifié, et une non répudiation de sa partassurée.

On représente ci-dessous est représenté le cryptosystème de transmission avec les traditionnels émetteurs et récepteurs (Bob et Alice).

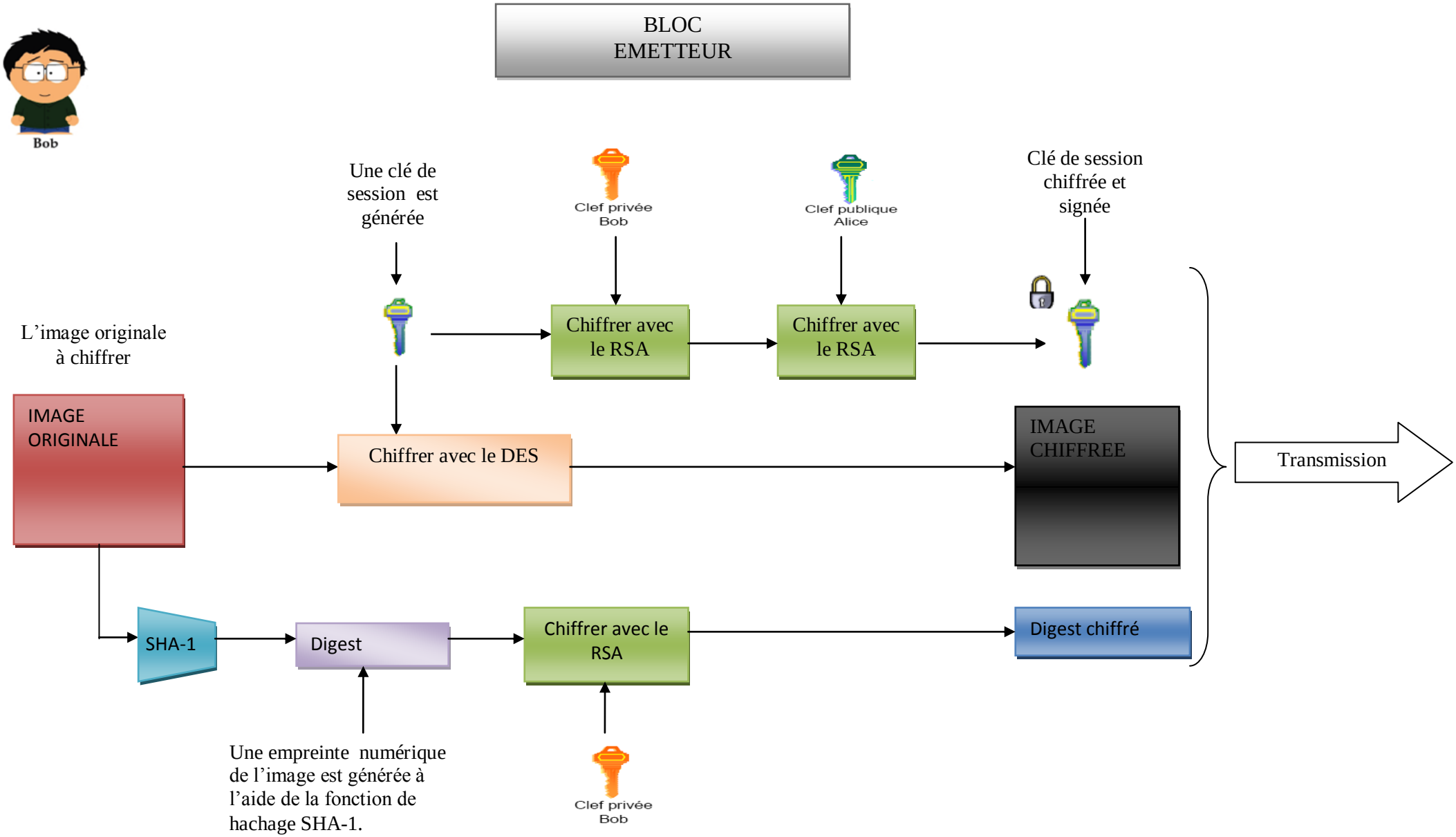


Figure III-1 : Le cryptosystème hybride de transmission d'images. « Bloc EMETTEUR ».

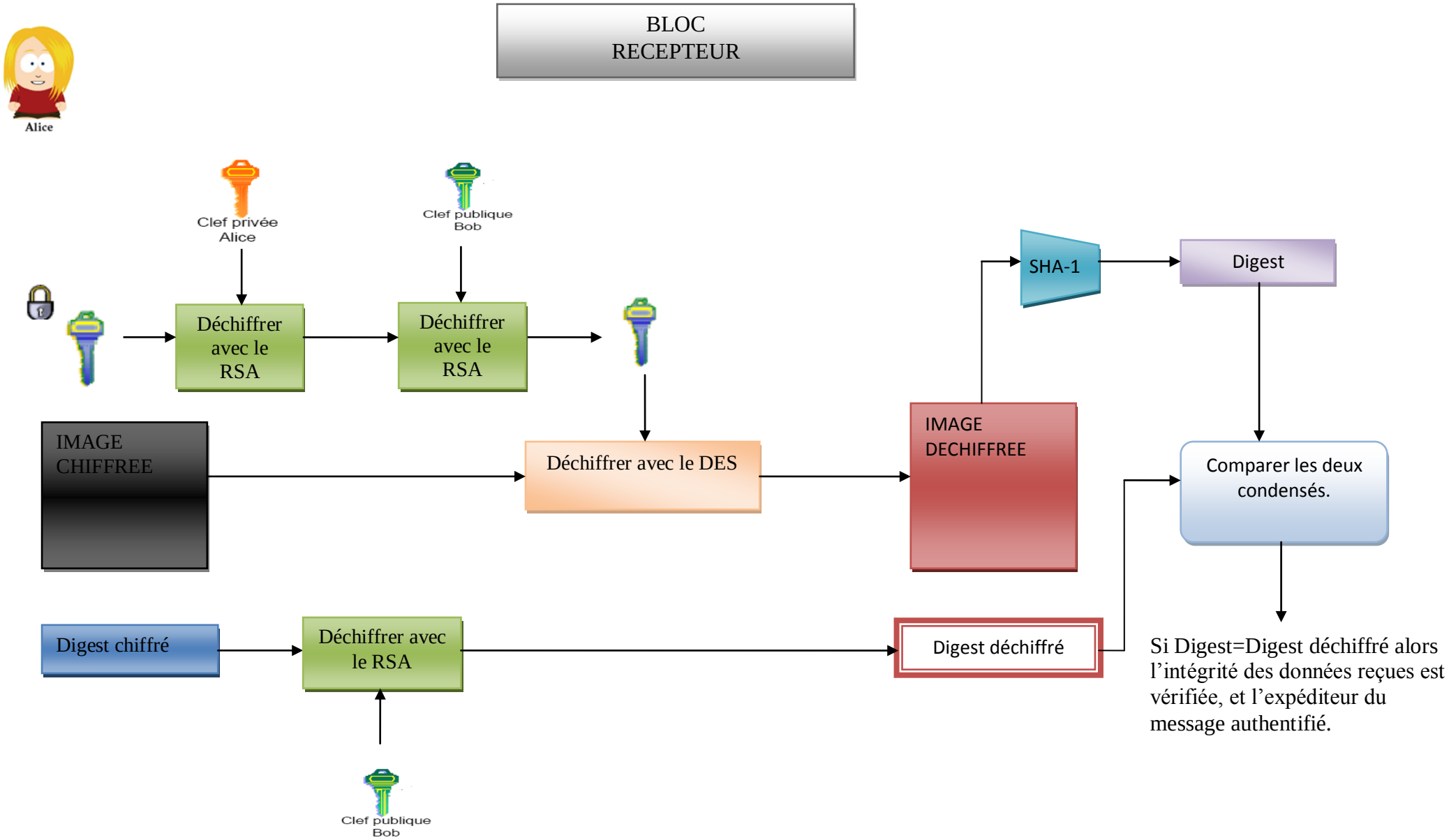


Figure III-2 : Le cryptosystème hybride de transmission d'images. « Bloc RECEPTEUR».

III.4 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.

Pour nos différents tests, on utilise l'image « **cameraman.tif** », de taille**120×120 pixels.**

Nous utiliserons aussi les deux modes de chiffrement **ECB** et **CBC**.

III.4.1. Analyse statistique :

III.4.1.1 Analyse d'histogramme :

Un histogramme d'image montre la manière de distribution des pixels dans une image en traçant le nombre de pixels correspondant à chaque intensité de couleur.

Dans ce teste nous avons chiffré notre image avec les deux modes ECB et CBC.

Le mode ECB

..

Figure III-3-a- : Image Originale

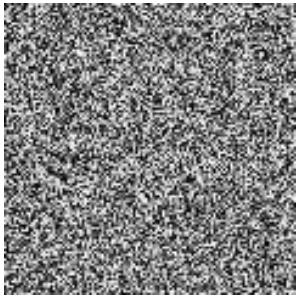


Figure III-3- b: Image Chiffrée

Figure III-3-c : Image déchiffrée.

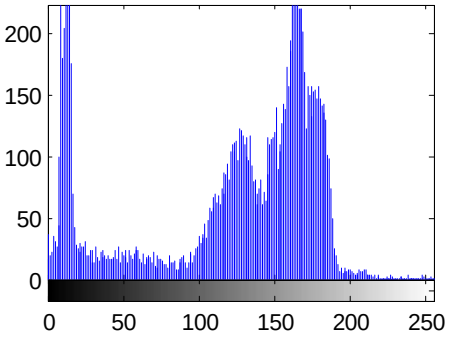


Figure III-3-d : Histogramme de l'image originale

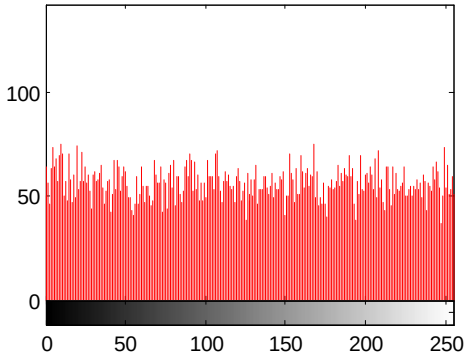


Figure III-3-e: Histogramme de l'image chiffrée

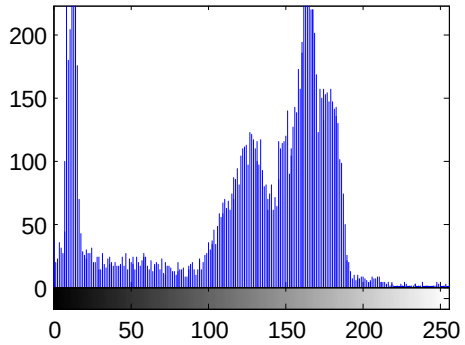


Figure III- 3-f: Histogramme de l'image Déchiffrée.

Figure III-3 :Images : originale, chiffrée et déchiffrée avec leurs histogrammes respectifs : Mode ECB.

Le mode CBC

Figure III-4-a- : Image Originale

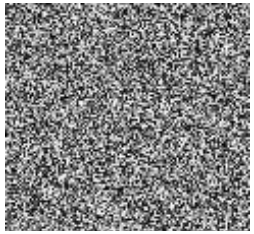


Figure III-4- b: Image Chiffrée

Figure III-4-c : Image déchiffrée.

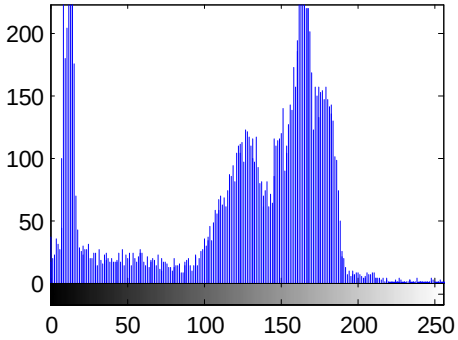


Figure III-4-d : Histogramme de l'image originale

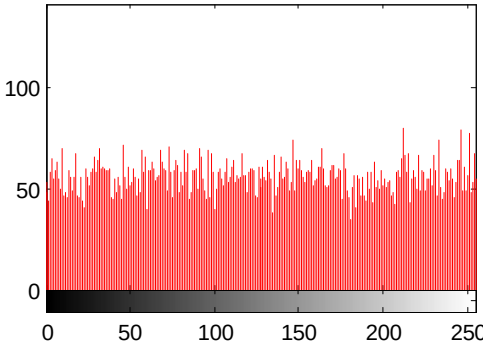


Figure III-4-e: Histogramme de l'image chiffrée

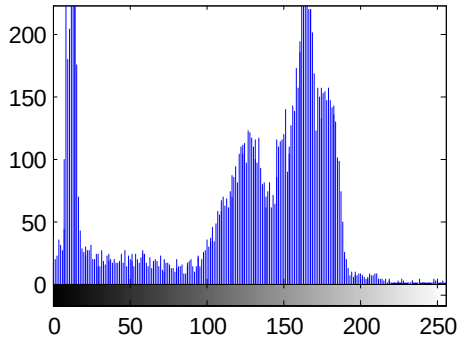


Figure III- 4-f: Histogramme de l'image Déchiffrée.

Figure II-4 : Images : originale, chiffrée et déchiffrée avec leurs histogrammes respectifs : Mode CBC.

Discussion :

A partir des figures précédentes, on constate que :

-Notre algorithme chiffre et déchiffre l'image correctement dans les deux modes de chiffrement, les histogrammes des images déchiffrées sont identique à ceux des images originales.

On peut bien voir que les histogrammes des images chiffrées dans les deux modes sont à peu prés uniformes. Ils sont très différents de ceux des images originales. L'algorithme de chiffrement utilisé fait en sorte que la dépendance des propriétés statistiques de l'image chiffrée et de l'image originale soit quasi aléatoire.

L'inconvénient du mode ECBsoulevé lors de l'exemple de chiffrement sur un texteau chapitre II, est aussi d'actualité lorsqu'on chiffre des images de fortes structures. Nous introduisons dans ce qui suit un exemple de chiffrement d'une image à forte structure de taille 120×120 pixels, avec les deux modes de chiffrement ECB et CBC.

Le mode ECB

Figure II-5-a- : Image Originale

Figure II-5- b: Image Chiffrée

Figure II-5-c : Image déchiffrée.

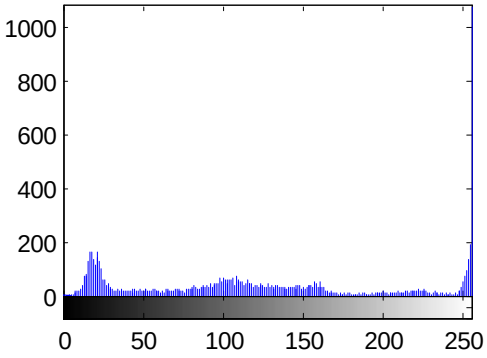


Figure II-5-d : Histogramme de l'image originale

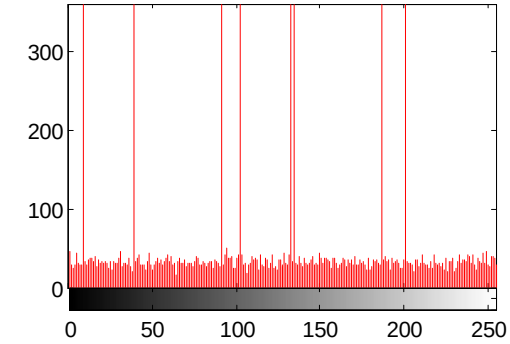


Figure II-5-e: Histogramme de l'image chiffrée

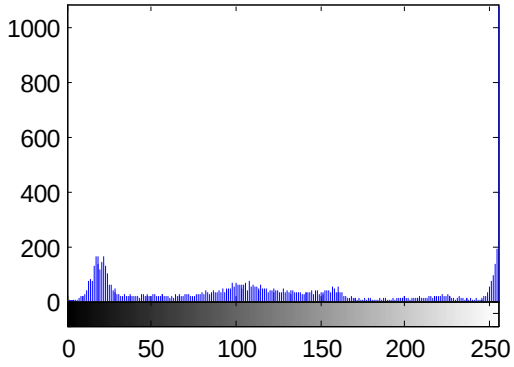


Figure II- 5-f: Histogramme de l'image Déchiffrée.

Figure II-5 : Images : originale, chiffrée et déchiffrée avec leurs histogrammes respectifs : Mode ECB.
« Cas d'une image à forte structure ».

Le mode CBC.

Figure II-6-a- : Image Originale

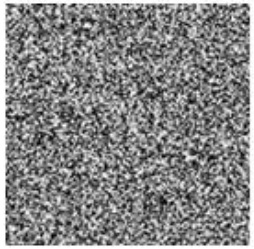


Figure II-6- b: Image Chiffrée

Figure II-6-c : Image déchiffrée.

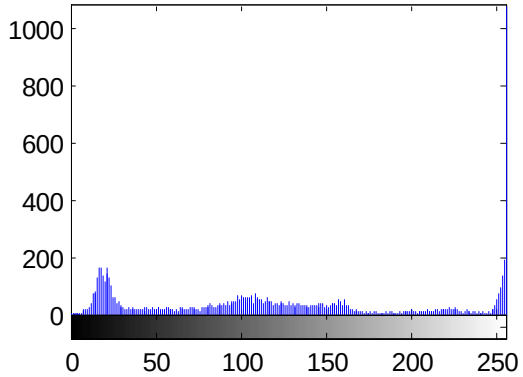


Figure II-6-d : Histogramme de l'image originale

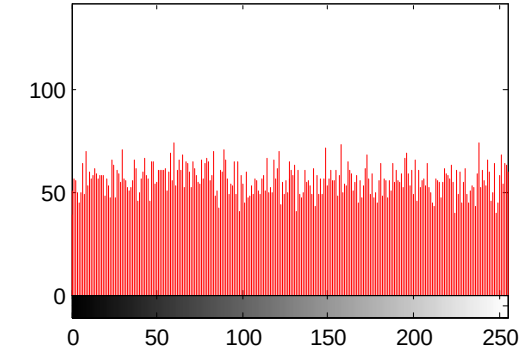


Figure II-6-e: Histogramme de l'image chiffrée

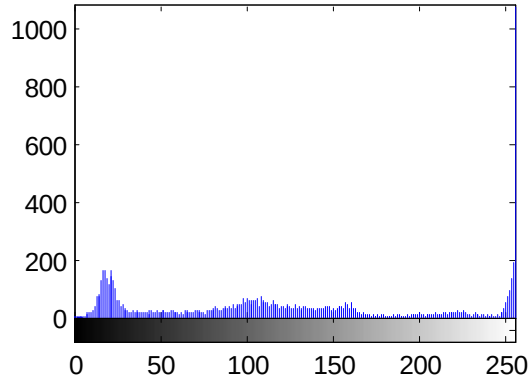


Figure II- 6-f: Histogramme de l'image déchiffrée.

Figure II-6 : Images : originale, chiffrée et déchiffrée avec leurs histogrammes respectifs : Mode CBC.
« Cas d'une image à forte structure ».

Avec l'image de la figure II-5 chiffrée en mode ECB, nous remarquons clairement que les blocs homogènes dans l'image originale sont chiffrés avec la même manière. L'histogramme de la figue II-5-e confirme cette remarque. Cet inconvénient majeur du mode ECB représente comme nous l'avons déjà vu une faille que les cryptanalystes peuvent exploiter en vu de récupérer de l'information. Tandis que l'histogramme de la même image chiffrée avec le mode CBC (figure II-6-e) présente un histogramme uniforme, ce qui permet alors de résoudre le problème du mode ECB.

III.4.1.2 Corrélation entre l'image originale et l'image chiffrée.

En plus de l'analyse d'histogrammes 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 : [11]

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

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

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 :

	Corrélation entre l'image originale et l'image chiffrée.
Mode ECB	**-0.0098**
Mode CBC	**-0.0061**

Tableau III-1 : Coefficient de corrélation entre l'image originale et l'image chiffrée avec deux modes de chiffrement : ECB et CBC.

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.

III.4.1.3 Corrélation entre les pixels adjacents :

Pour une image ordinaire, chaque pixel est fortement corrélé avec ses pixels adjacents dans la direction horizontale ou verticale. Un algorithme de chiffrement idéal devrait produite des images chiffrées dont la corrélation entre les pixels adjacents est négligeable.
Pour tester la corrélation entre les pixels adjacents horizontaux ou verticaux de l'image, on calcule les coefficients de corrélation à l'aide de la formule suivante :

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

Avec : $\bar{X}=\frac{1}{N}\sum_{i=1}^{N}X_i$ et $\bar{Y}=\frac{1}{N}\sum_{i=1}^{N}Y_i$

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

	Corrélation horizontale	**Corrélation verticale**
Image originale	**0.9480**	**0.9117**

Tableau III-2 : Coefficients de corrélation des pixels adjacents horizontaux et verticaux dans l'image originale.

Image chiffrée	**Corrélation horizontale**	**Corrélation verticale**
Mode ECB	**0.0230**	**0.0066**
Mode CBC	**0.0246**	**0.0216**

Tableau III-3 : Coefficients de corrélation des pixels adjacents horizontaux et verticaux dans l'image chiffrée, avec deux modes de chiffrement : ECB et CBC.

Il est clair qu'à partir des résultats des tableaux III.2 et III.3 que la corrélation est très petite dans l'image chiffrée. Ceci est confirmé par les figures III-7 et III-8 ci-dessous : (Résultats pour 1080 pixels).

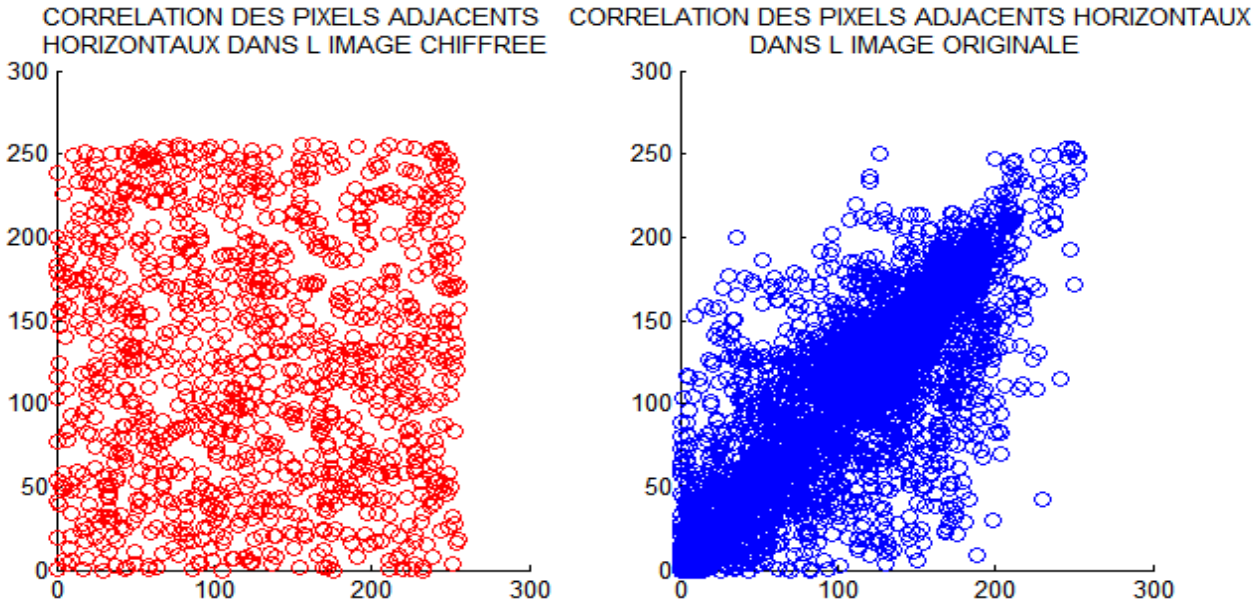


Figure III-7 : Figures illustrant la corrélation des pixels adjacents horizontaux dans l'image chiffrée et son original. « Mode ECB »

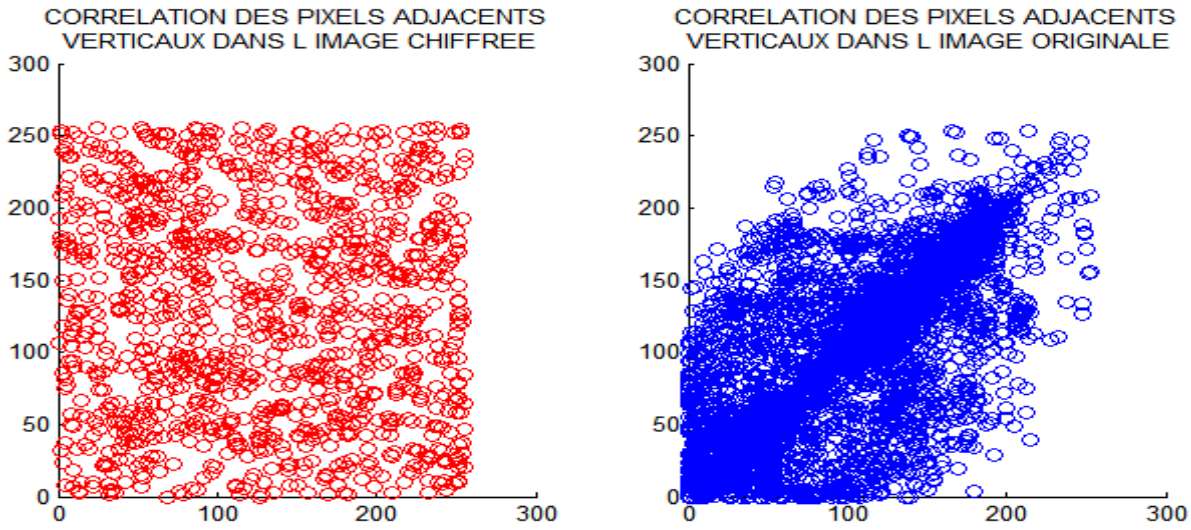


Figure III-8 : Figures illustrant la corrélation des pixels adjacents verticaux dans l'image chiffrée et son original. « Mode ECB »

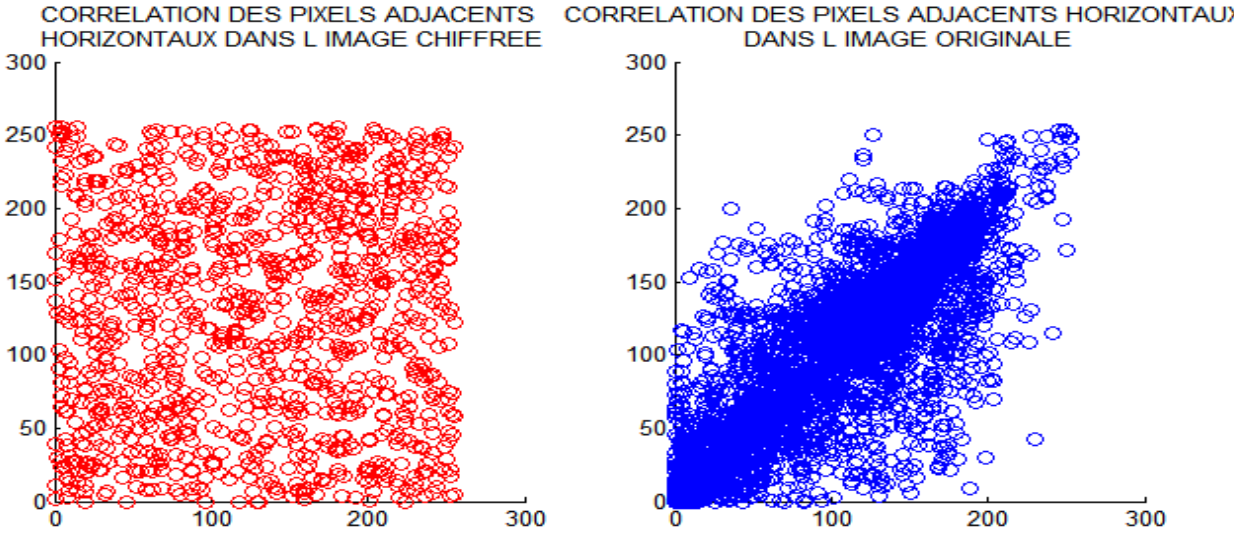


Figure III-9 : Figures illustrant la corrélation des pixels adjacents horizontaux dans l'image chiffrée et son original. « Mode CBC »

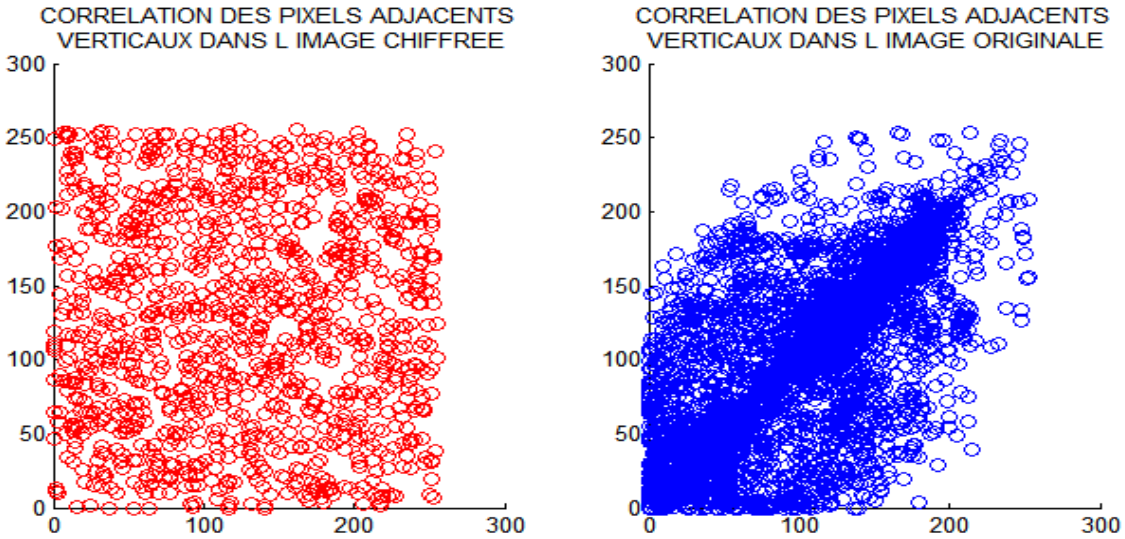


Figure III-10 : Figures illustrant la corrélation des pixels adjacents verticaux dans l'image chiffrée et son original. « Mode CBC »

III.4.1.4 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 I*** : On utilise deux clés qui différent 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 II*** : 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 I :**

Pour ce test, une image est chiffrée avec deux clés qui différent d'un seul bit, on calcule ensuite le coefficient de corrélation entre les deux images chiffrées.

	Corrélation entre les deux images chiffrées
Mode ECB	**0.0029**
Mode CBC	**-0.0019**

Tableau III-4 : Coefficients de corrélation entre les deux images chiffrées du test I avec deux modes de chiffrement : ECB et CBC.

- **Test 2 :**

Pour ce test une image est chiffrée avec une clé de chiffrement, on essai ensuite cette même image avec une clé de déchiffrement qui diffère de la clé de chiffrement d'un seul bit. Les résultats sont sur les figures III.11.

Figure III-11-a : Image originale

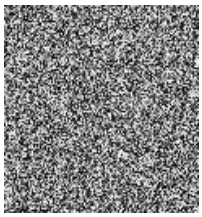


Figure III-11-b : Image chiffrée.

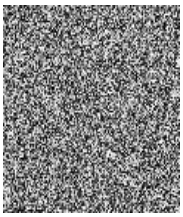


Figure III-11-c : Image déchiffrée avec la clé différente.

Figure III-11 : Figure illustrant l'image originale, son chiffré et l'image déchiffrée avec une clé différente de celle du chiffrement.

D'après la figure III-11, 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= -0.007.

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.

III.4.2 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 (I_{O1} $et I_{O2}$)qui diffèrent d'un seul pixel, les deux images chiffrées obtenues sont notées (I_{C1} $et I_{C2}$). Les trois paramètres NPCR (Number of Pixels Change Rate), l'erreur quadratique moyenne (MSE : Mean Square Error) et (MAE : Mean Absolute Error) l'erreur absolue moyenne 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 [11].

$$\mathbf{NPCR}=\frac{\sum_{i=1}^{H}\sum_{j=1}^{L}D_{ij}}{H\times L}\times 100\% \text{ .}$$

Avec :

$$\begin{cases} \text{Si } I_{C1} \neq I_{C2} \text{ ou } I_{O1} \neq I_{O2} \\ D_{ij} = 0 \text{ sinon} \end{cases}$$

On calcule aussi de le NPCR référence en utilisant l'équation suivante [14] :

$\mathbf{NPCR_{réf}}=(1-2^{-n})\times 100\%$ où **n** représente le nombre de bits utilisés pour représenter un pixel. (Dans notre cas n=8bits)

$\text{NPCR}_{\text{réf}}=(\mathbf{1-2^{-8}})\times \mathbf{100\%}$ **=99.6094%**

MAE : L'erreur absolue moyenne est donnée par la formule suivante [12] :

$$\mathbf{MAE}=\left(\frac{1}{HL}\right)\frac{\sum_{i=1}^{H}\sum_{j=1}^{L}|I_{O1,C1}-I_{O2,C2}|}{255}$$

MSE : L'erreur quadratique moyenne est donnée par la formule suivante [12] :

$$\mathbf{MSE}=\left(\frac{1}{HL}\right)\frac{\sum_{i=1}^{H}\sum_{j=1}^{L}|I_{O1,C1}-I_{O2,C2}|^{2}}{255^{2}}$$

Le tableau III.5 montre les résultats obtenus :

	NPCR	MAE	MSE
Image originale	**0.0156 %**	**0.0053 %**	**0.0018 %**
Image chiffrée	**95.5625 %**	**31.9827 %**	**16.0380 %**

Tableau III-5 : Tableau représentant le niveau de confusion entre l'image originale et l'image chiffrée.

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.

III.4.3 Contrôle d'intégrité.

Pour ce test, nous avons chiffré notre image avec le mode ECB, ensuite on a permuté deux régions dans l'image chiffrée. L'image déchiffrée à la réception est représentée sur la figure II-12-b.

Figure II-12-a: Image originale.Figure II-12-b: Image déchiffrée.

Figure III-12 : Figures représentant l'image originale et l'image déchiffrée après modification de l'image chiffrée

Afin de vérifier l'intégrité des données reçue, 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 III-6 ci-dessous.

Mode	Empreinte de l'image originale à l'émission	Empreinte de l'image déchiffrée à la réception
ECB	**E12347093A57E48807967F9A969F8CC9A22B363E**	**913EAA1E8E033AD73E6D4DFD89C9998018A13FC6**

Tableau III-6 : Tableau illustrant l'empreinte de l'image originale à l'émission et l'empreinte de l'image déchiffrée à la réception après modification de l'image chiffrée durant la transmission.

Nous constatons que toutes modifications 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 d'intégrité n'est plus posé dans notre cryptosystème.

III.5 Conclusion.

Le deuxième chapitre a introduit dans sa première partie des notions de cryptographie hybride, son principe de fonctionnement et la procédure utilisée dans de la conception de notre cryptosystème hybride.

La deuxième partie quant à elle a été consacrée à l'analyse de la sécurité et des performances du cryptosystème.

On note que les résultats obtenus sont assez satisfaisants, ce qui garantie alors à notre cryptosystème une certaine robustesse.

Dans le chapitre qui suit, nous envisageons d'implémenter notre cryptosystème sur une carte électronique de type Arduino-Uno.

CHAPITRE IV

IMPLEMENTATION DU CRYPTOSYSTEME SUR CARTES ARDUINO-UNO.

Chapitre IV

Implémentation du cryptosystème sur cartes Arduino-Uno.

IV.1 Introduction

Nous envisageons dans ce chapitre une implémentation de notre cryptosystème hybride sur deux cartes Arduino-Uno. Dans la première sera implémentée le bloc émetteur et la deuxième le bloc récepteur, ceci après avoir donnée une description détaillée de la carte utilisée. On termine par la présentation de l’ application pratique réalisée, l’interprétation et la comparaison des résultats pratiques obtenus avec les résultats théoriques des chapitres précédents.

IV.2 Description de la carte Arduino Uno

IV .2.1 Historique du projet Arduino.

Le projet Arduino est né en hiver 2005, Massimo Banzi enseigne dans une école **I**nteraction **D**esign d'**I**vrea **I**nstitute [1] (**IDII**) en Italie, et souvent ses étudiants se plaignent de ne pas avoir accès à des solutions bas prix pour accomplir leurs projets de robotique. Les coûts souvent trop élevés des autres outils de prototypage rendaient difficile le développement de nombreux projets et ceci ralentissaient la mise en œuvre concrète de leur apprentissage.

Banzi en discute avec David Cuartielles, un ingénieur Espagnol spécialisé dans les microcontrôleurs, et décident alors de créer leur propre carte en embarquant dans leur histoire un des étudiants de Banzi, David Mellis qui sera alors chargé de créer le langage de programmation allant avec la carte. En deux jours David écrira le code, trois jours de plus et la carte est crée.

La carte est alors un grand succès auprès des étudiants. Tout le mode arrive à en faire quelque chose très rapidement (Réponses à des capteurs, faire clignoter des LEDs, contrôler des moteurs ...)

Le nom Arduino trouve son origine dans le nom du bar dans lequel l'équipe avait l'habitude de se retrouver. Arduino est aussi le nom d'un roi italien, personnage historique de la ville « Arduin d'Ivrée »

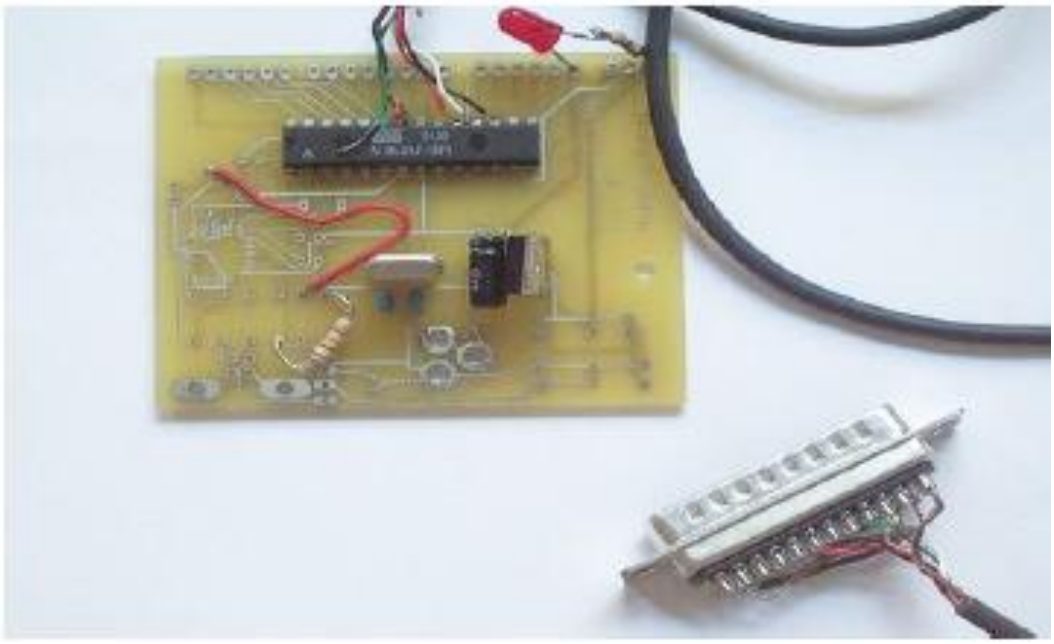


Figure IV.1 : Prototype d'Arduino.

[1] L'école Interaction Design d'Ivrea (IDII) est aujourd'hui située à Copenhague (Danemark) sous le nom de Copenhagen Institute of interaction Design.

IV.2.2 Partie matérielle. [21] [25] [22]

Arduino est un circuit imprimé en matériel libre (les plans de la carte elle- même sont publiés en licence libre, sur lequel se trouve un microcontrôleur qui peut être programmé pour analyser et produire des signaux électriques, de manière à effectuer des taches très diverses.

Un module Arduino est généralement construit autour d'un microcontrôleur Atmel AVR, et des composants complémentaires qui facilitent la programmation et l'interfaçage avec d'autres circuits.

Le modèle Uno de Arduino est une carte dont le cœur est un microcontrôleur de référence ATMega 328 cadencé à 16 Mhz. Elle possède 32 Ko de mémoire flash destinée à recevoir le programme, 2 Ko SRAM (mémoire vive) et 1 Ko d'EEPROM (mémoire morte pour les données). Elle offre 14 broches d'entrée/sortie numérique dont 6 peuvent générer des PWM. Elle permet aussi de mesurer des grandeurs analogiques grâce à ses 6 entrées analogiques.

Cette carte Arduino peut aussi s'alimenter et communiquer avec un ordinateur grâce à son port USB. Une autre alimentation est possible grâce à son connecteur power jack.

Ci-dessous est représentée une synoptique de la carte avec l'indication de ses différents composants.

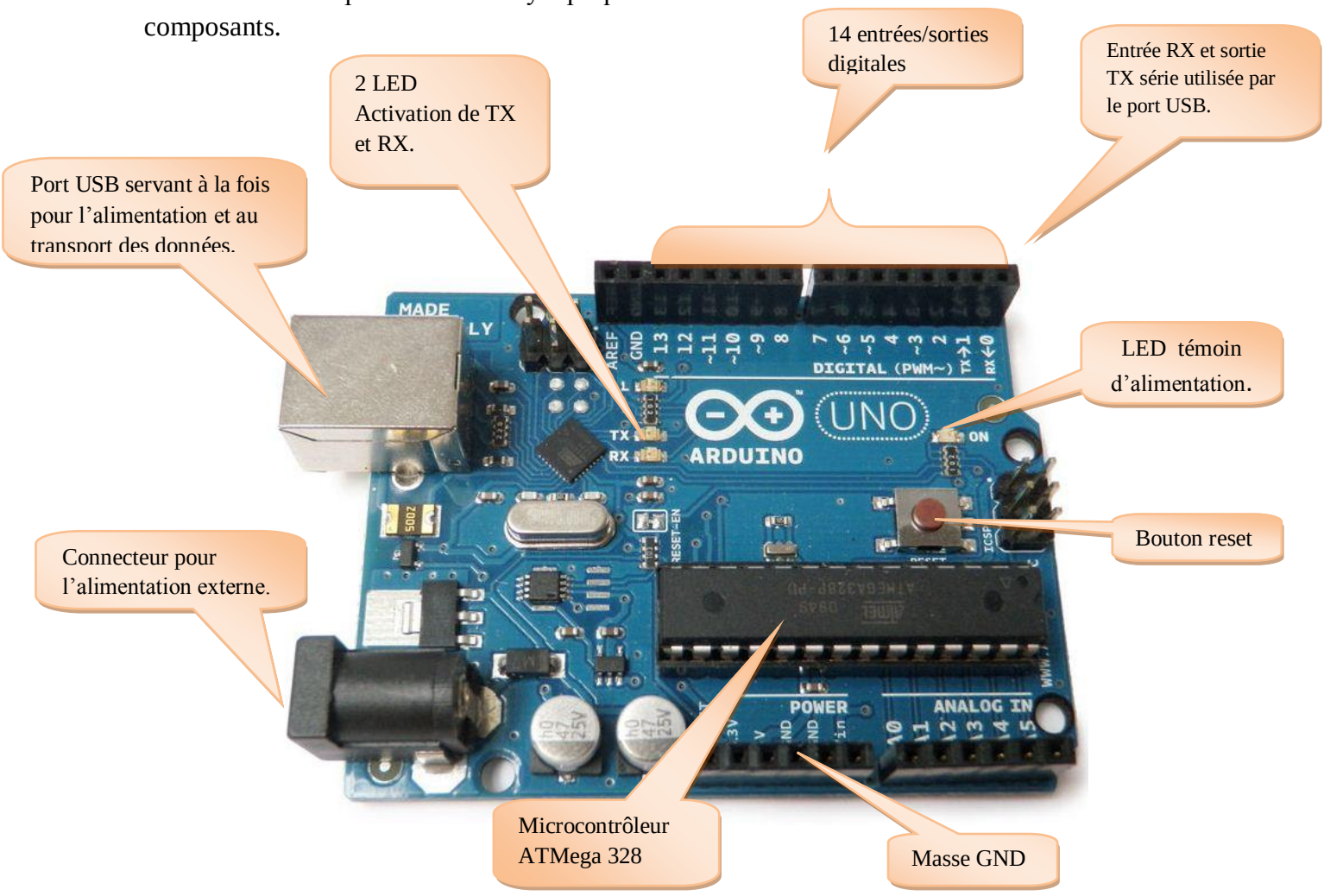


Figure IV.2 Synoptique illustrant les différents composants de la carte Arduino UNO.

On représente aussi le microcontrôleur ATMega 328, un Atmel de la famille AVR.

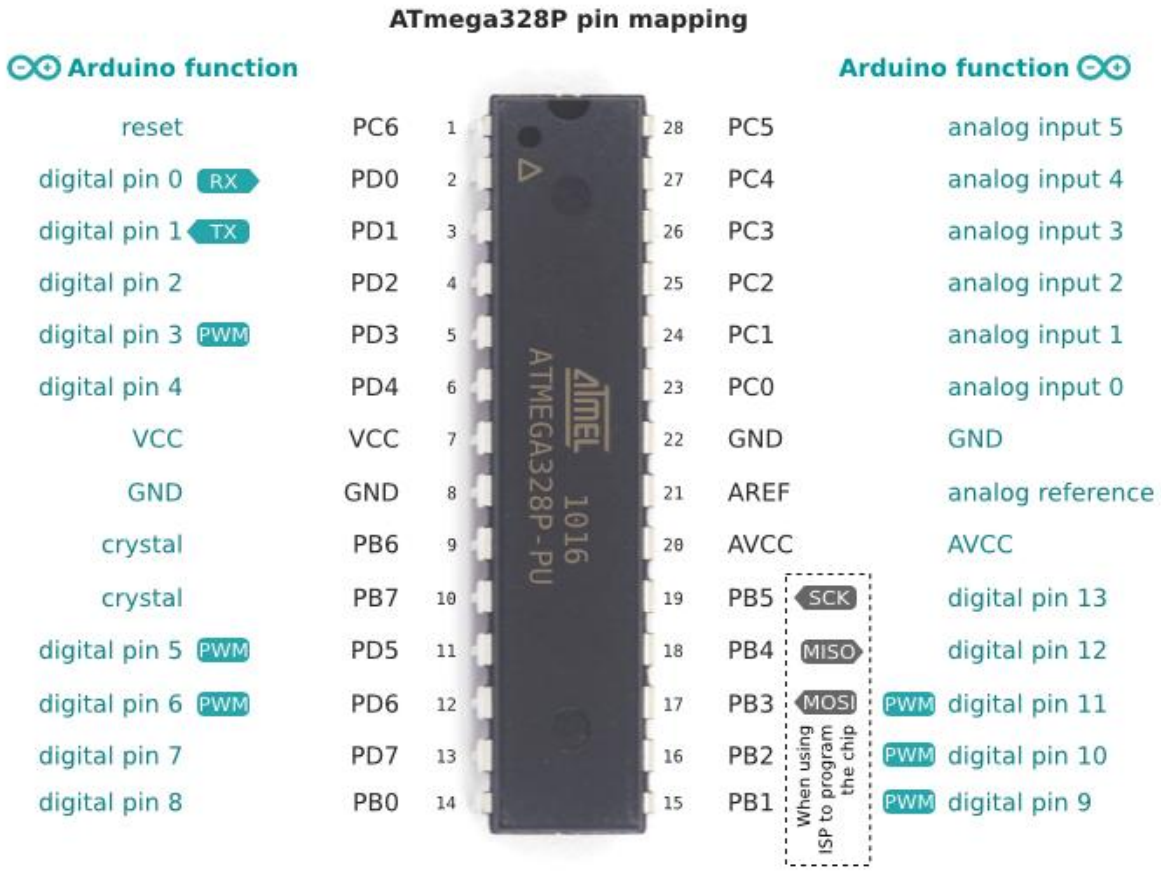


Figure IV.3 : Synoptique du microcontrôleur ATMega328 d'Arduino Uno.

Synthèse des caractéristiques.

Arduino Uno	
Microcontrôleur	**ATMega 328**
Tensions de fonctionnement	**5Volts, la plage limite est 6 -20 Volts, recommandée 7-12 Volts.**
Broches E/S numériques	**14 dont 6 disposent d'une sortie PMW.**
Broches d'entrée analogique	**6 qui sont aussi utilisables en broches E/S numériques.**
Mémoire programme flash	**32 KB dont 0.5 KB sont utilisées pour le boot loader.**
Mémoire RAM(Volatile)	**2 KB**
Mémoire EEPROM (Non volatile)	**1KB**
Vitesse d'horloge	**16MHz**

IV.2.3 Partie logicielle. [22] [23] [25]

IV.2.3.1 Programmation de l'Arduino

En parallèle au matériel, un IDE (Environnement de développement) à été développé afin de permettre la programmation des modules de l'Arduino. Cet IDE est une application Java libre servant d'éditeur de codes (programmes) et de compilateur. Les opérations de compilation et de chargement dans la mémoire du microcontrôleur sont alors ramenées à de simples clics. La communication en Arduino et le PC se fait via le port USB.

L'interface de l'IDE Arduino est plutôt simple (Figure IV.4), il offre une interface minimale et épurée pour développer les codes. Il est doté d'un **éditeur** de codes avec coloration syntaxique et d'une **barre d'outils** rapide**.** On retrouve aussi une **barre de menus** plus classique qui est utilisée pour accéder à différente fonctionnalités de l'IDE. Enfin **une console** affichant les résultats de la compilation du code source, des opérations sur la carte, … etc.

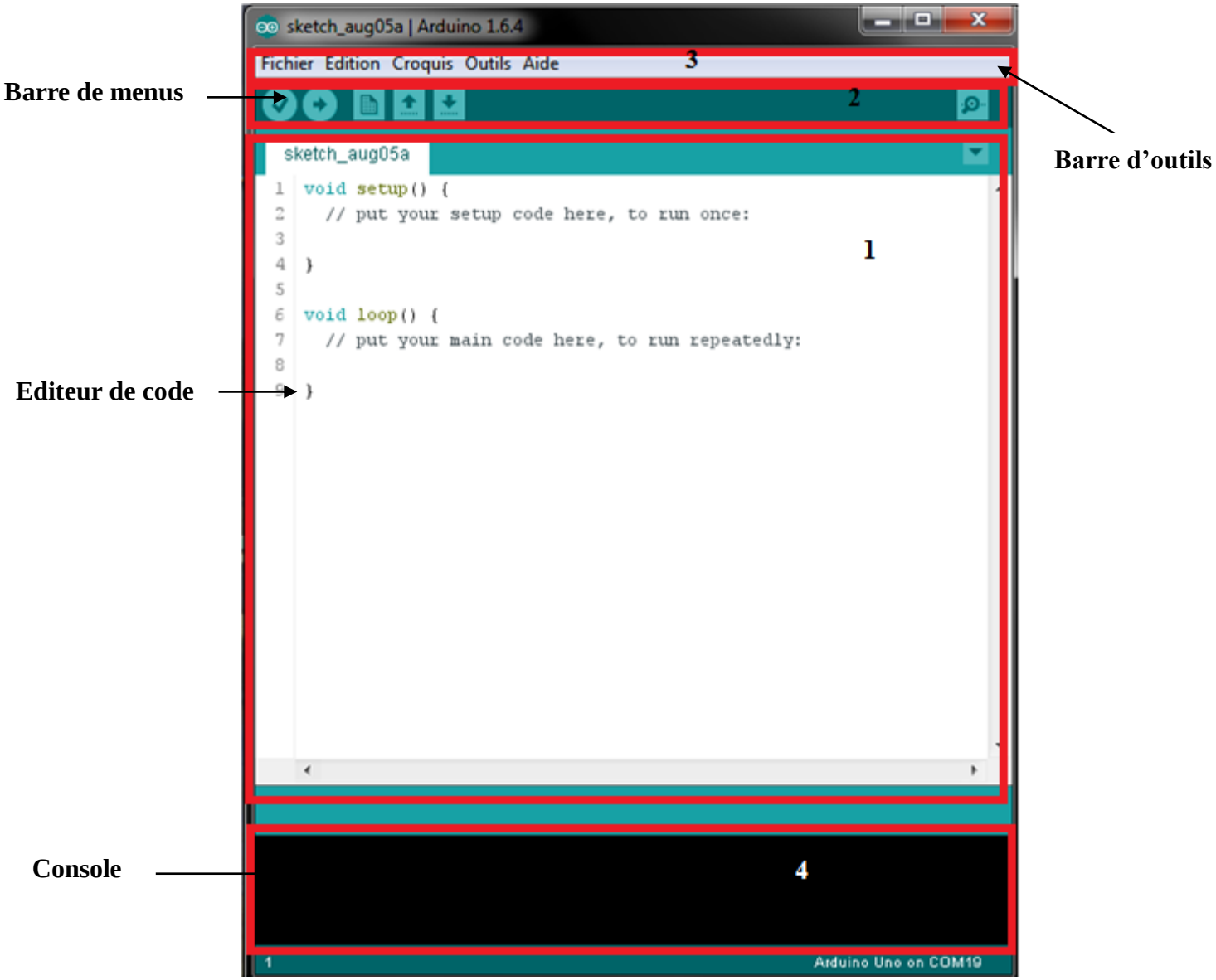


Figure IV.4 Interface de l'IDE d'Arduino.

IV.2.3.2 Structure d'un projet Arduino.

L'outil de développement d'Arduino impose au programmeur une structure de programme.

1. La fonction main () imposée et non modifiable.
2. La fonction setup () qui doit contenir les différentes initialisations du code.
3. La fonction loop () : Elle doit contenir les boucles, les fonctions répétée indéfiniment, elle est exécutée après la fonction setup.

IV.2.3.3 Communication de l'Arduino [22] [25]

La carte Arduino-Uno dispose de toute une série de facilités pour communiquer avec un ordinateur, une autre carte Arduino ou une avec d'autres microcontrôleurs. L'ATMega 328 dispose d'une UART un émetteur -récepteur asynchrone universel pour une communication série disponible au niveau des broches 0 (RX) et 1 (TX). Un circuit intégré ATMega 8U2 sur la carte assure la connexion entre cette communication série vers le port USB de l'ordinateur et apparait alors comme un COM virtuel pour les logiciels de l'ordinateur.

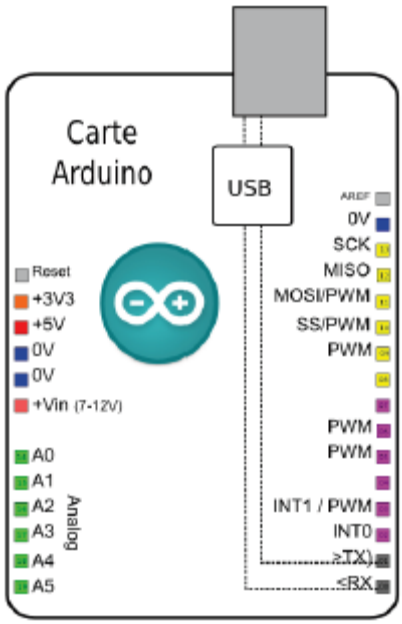


Figure IV.5 Brochages de la carte Arduino

L'IDE Arduino inclut une fenêtre terminal série (moniteur série) sur l'ordinateur qui permet d'envoyer des textes simple depuis et vers la carte Arduino (Figure(IV.6)).

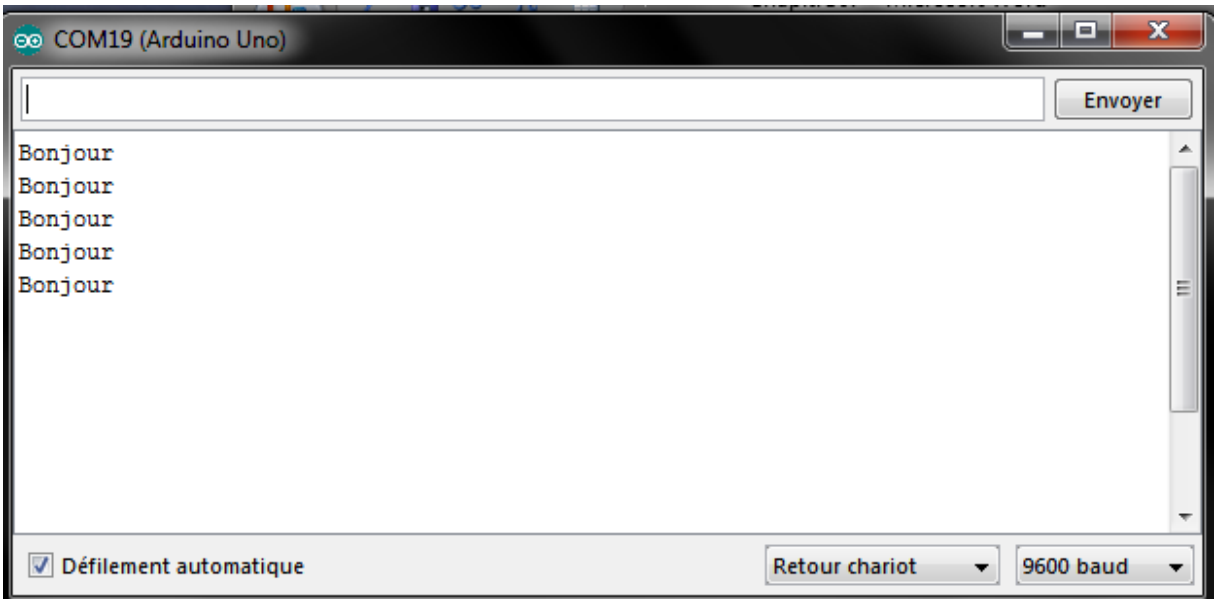


Figure IV.6 Interface du moniteur série d'Arduino.

IV.3 Présentation de l'application pratique réalisée.

Nous présentons dans ce qui suit la procédure utilisée dans l'implémentation de notre cryptosystème hybride de transmission d'images sur deux cartes Arduino-Uno, dans la première sera alors implémenté le bloc émetteur et dans la deuxième le bloc récepteur.

IV.3.1 Problématique.

Parmi les problèmes qui s'opposent à la mise en œuvre de notre cryptosystème sur une carte Arduino-Uno, nous avons le problème de la gestion de la mémoire disponible. Comme nous l'avons vu dans la première partie notre carte dispose d'une capacité mémoire RAM d'un 1KB et une mémoire Flash de 32 KB assez limitées. Ceci nous a poussé à restreindre notre application pratique à l'implémentation de l'algorithme symétrique « DES » appliqué au chiffrement d'un texte en utilisant les deux modes de chiffrement ECB et CBC.

IV.3.2 Mise en œuvre.

IV.3.2.1 Bloc émetteur.

Le bloc émetteur est construit autour d'une carte Arduino-Uno, il comprend l'algorithme de chiffrement du DES. Les résultats du chiffrement sont affichés sur le terminal série de l'IDE Arduino.

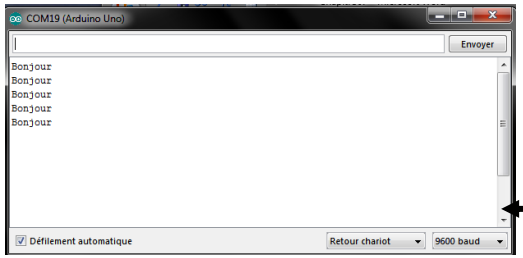
IV.3.2.2.2 Bloc récepteur.

Le bloc récepteur est aussi construit autour d'une deuxième carte Arduino, il comprend l'algorithme de déchiffrement du DES. Les résultats du déchiffrement des données reçues du bloc émetteur sont affichés sur le terminal série de l'IDE de l'Arduino.

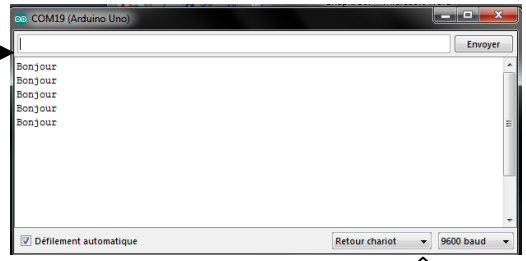
IV.3.2.3 Canal de transmission.

Pour la transmission des données chiffrées, nous utiliserons une transmission série en reliant la broche (TX) de la carte Arduino du bloc émetteur à la broche (RX) de la carte Arduino du bloc récepteur

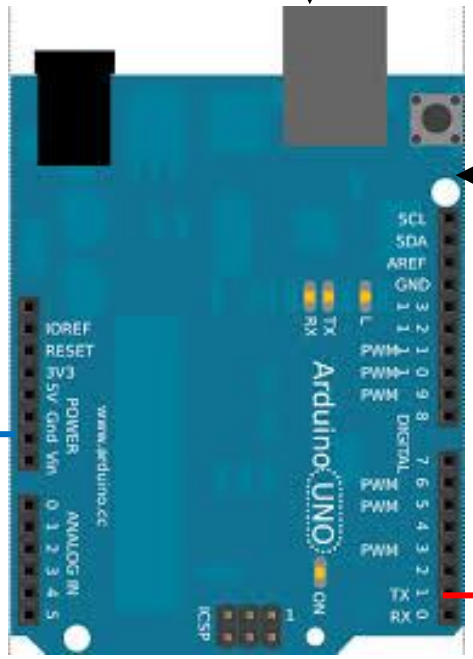
Nous représentons ci-dessous une synoptique des deux blocs et le Protocol de transmission utilisé.



Le terminal série du bloc récepteur, il permet d'afficher les données chiffrées reçues du bloc émetteur et les données déchiffrées.

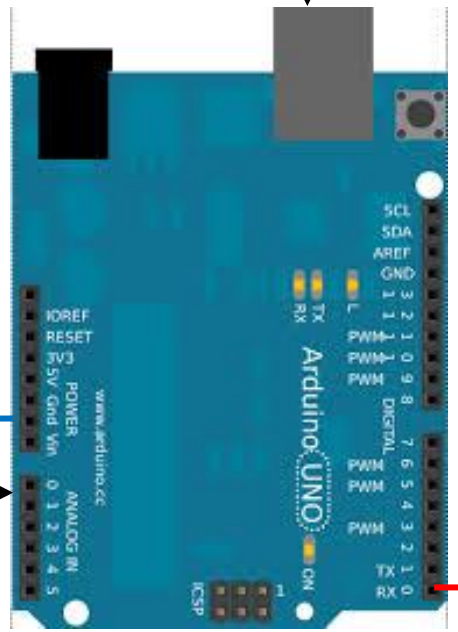


Le terminal série du bloc émetteur, il permet d'afficher le résultat du chiffrement.



Dans la première carte nous implémentons l'algorithme de chiffrement du DES avec les deux modes ECB et CBC. Une interface permet de saisir la clé de chiffrement et le vecteur initial pour le mode CBC.

Dans la seconde carte nous implémentons l'algorithme de déchiffrement du DES pour les deux modes ECB et CBC. Une interface permet de saisir la clé de déchiffrement et le vecteur initial pour le mode CBC.



Liaison série (TX RX) entre les deux cartes

Masse commune entre les deux cartes

IV.4 Tests et résultats.

Nous présentons dans cette section quelques tests et résultats pratiques, que nous allons comparer aux résultats théorique obtenus dans le second chapitre.

Test 01 :

Nous chiffrons un plaintext de taille 64 bits, avec une clé de 64 bits. Les résultats du chiffrement est dans le tableau IV. Et illustré dans la figure IV.7 qui représente le résultat du chiffrement obtenu sur le terminal série de la carte Arduino.

Plaintext	**02 46 8A CE EC A8 64 20**
Clé de chiffrement	**0F 15 71 C9 47 D9 E8 59**
Ciphertext ECB	**DA 02 CE 3A 89 EC AC 3B**
Ciphertext CBC	**C5 A4 64 33 12 14 DA 5A**

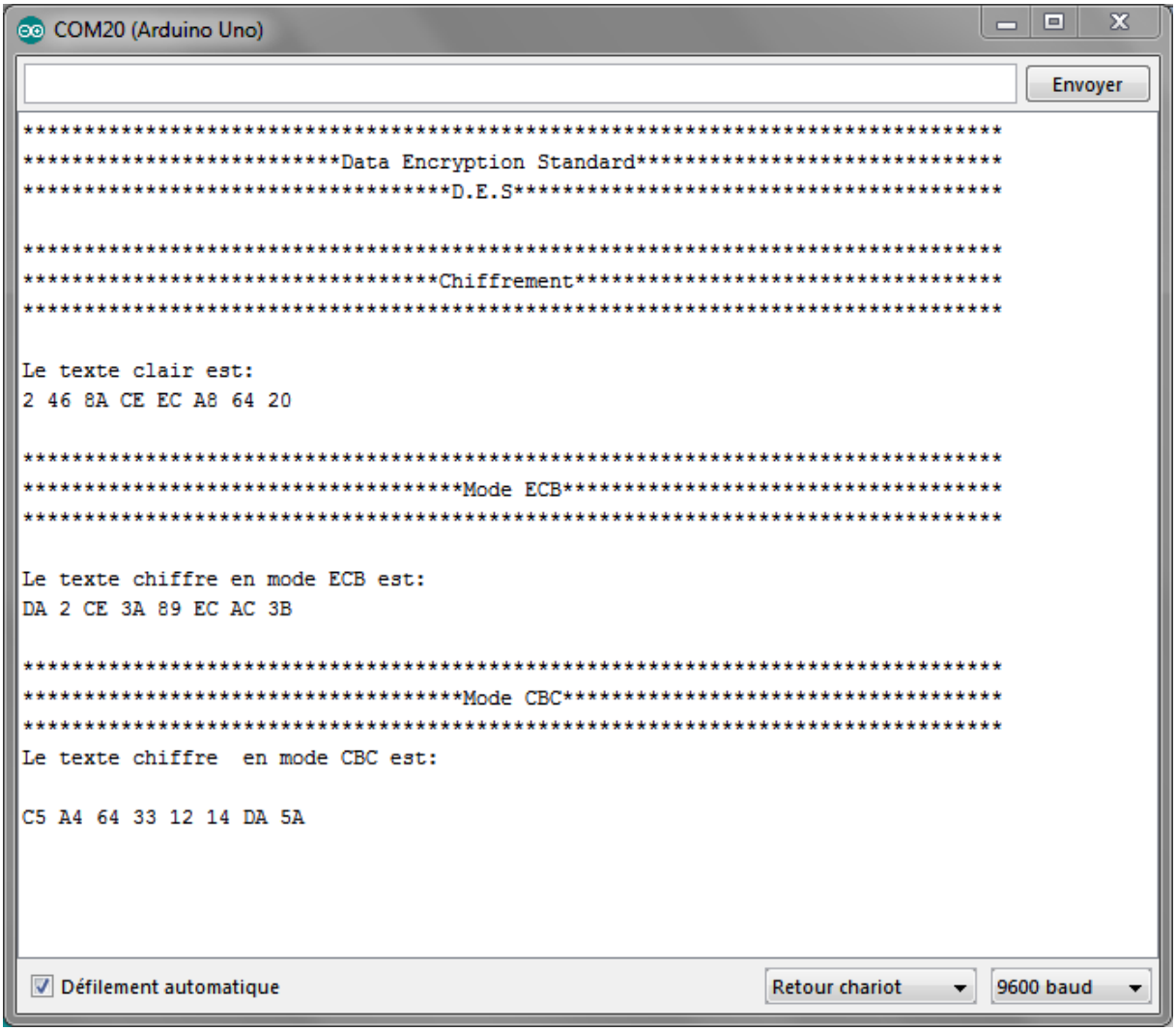


Figure IV.7 : Résultat du chiffrement dans les deux modes ECB et CBC, affiché dans le sérial monitor de l'Arduino.

Test 02 :

Pour mette en évidence l'inconvénient du mode ECB soulevé dans le chapitre II, nous chiffrons deux blocs identiques avec les deux modes ECB et CBC, le résultat est donné dans le tableau ci dessous, et illustré dans la figure IV.8.

Plaintext	**02 46 8A CE EC A8 64 20 02 46 8A CE EC A8 64 20**
Clé de chiffrement	**0F 15 71 C9 47 D9 E8 59**
Ciphertext ECB	**DA 02 CE 3A 89 EC AC 3B DA 02 CE 3A 89 EC AC 3B**
Ciphertext CBC	**C5 A4 64 33 12 14 DA 5A 8C A8 CC EE CA 4A D6 58**

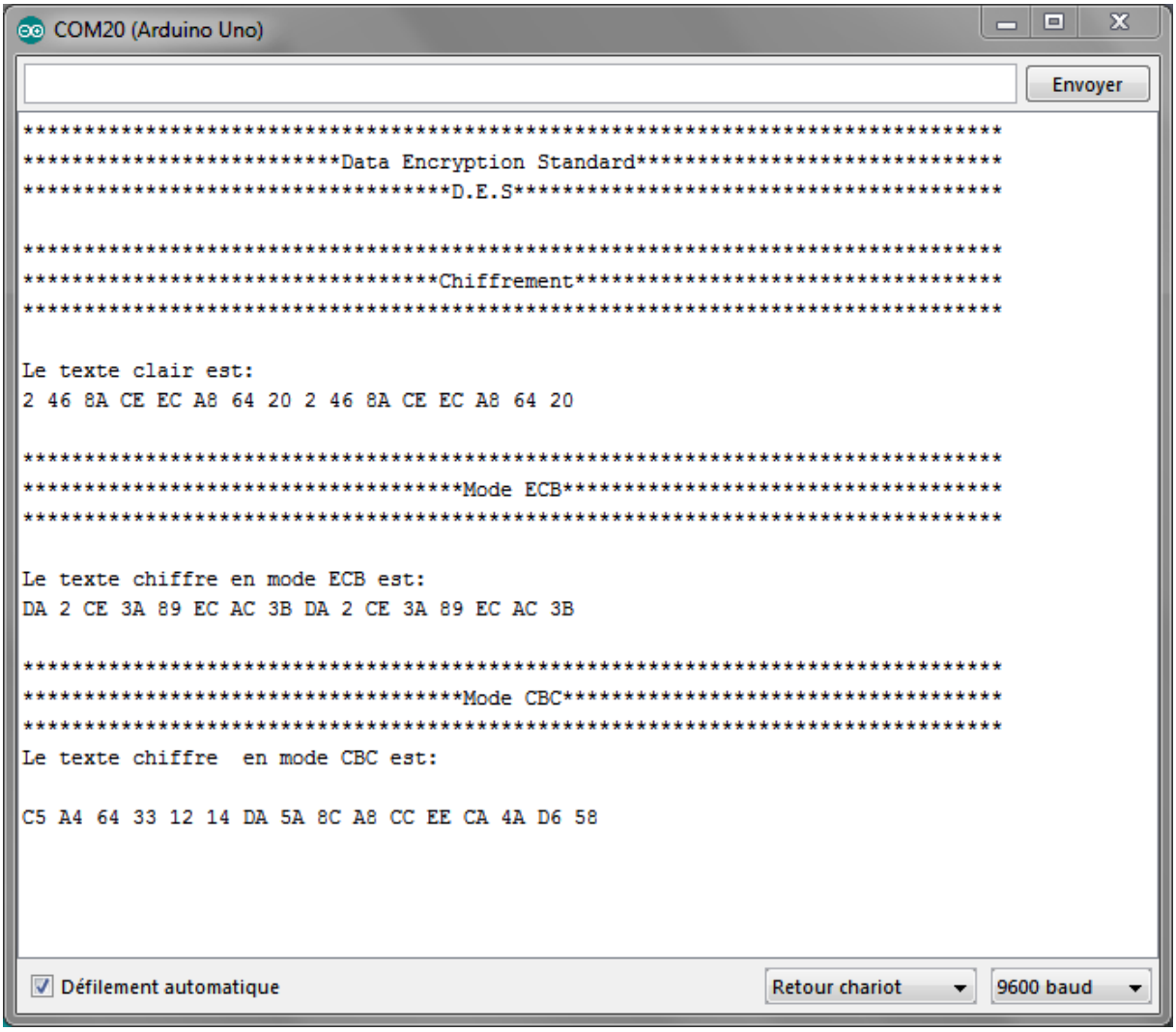


Figure IV.8 : Résultat du chiffrement de deux blocs identiques avec les modes ECB et CBC.

On constate que les blocs identiques du plaintext sont chiffrés de la même manière avec le mode ECB contrairement au mode CBC. Ceci confirme les résultats de la simulation obtenus au chapitre II.

Test 03 :

Afin de tester le l'algorithme de déchiffrement, nous utiliserons les données ci-dessous.

Clé de chiffrement	**0F 15 71 C9 47 D9 E8 59**
Ciphertext ECB	**DA 02 CE 3A 89 EC AC 3B**
Ciphertext CBC	**C5 A4 64 33 12 14 DA 5A**
Plaintext	**02 46 8A CE EC A8 64 20**

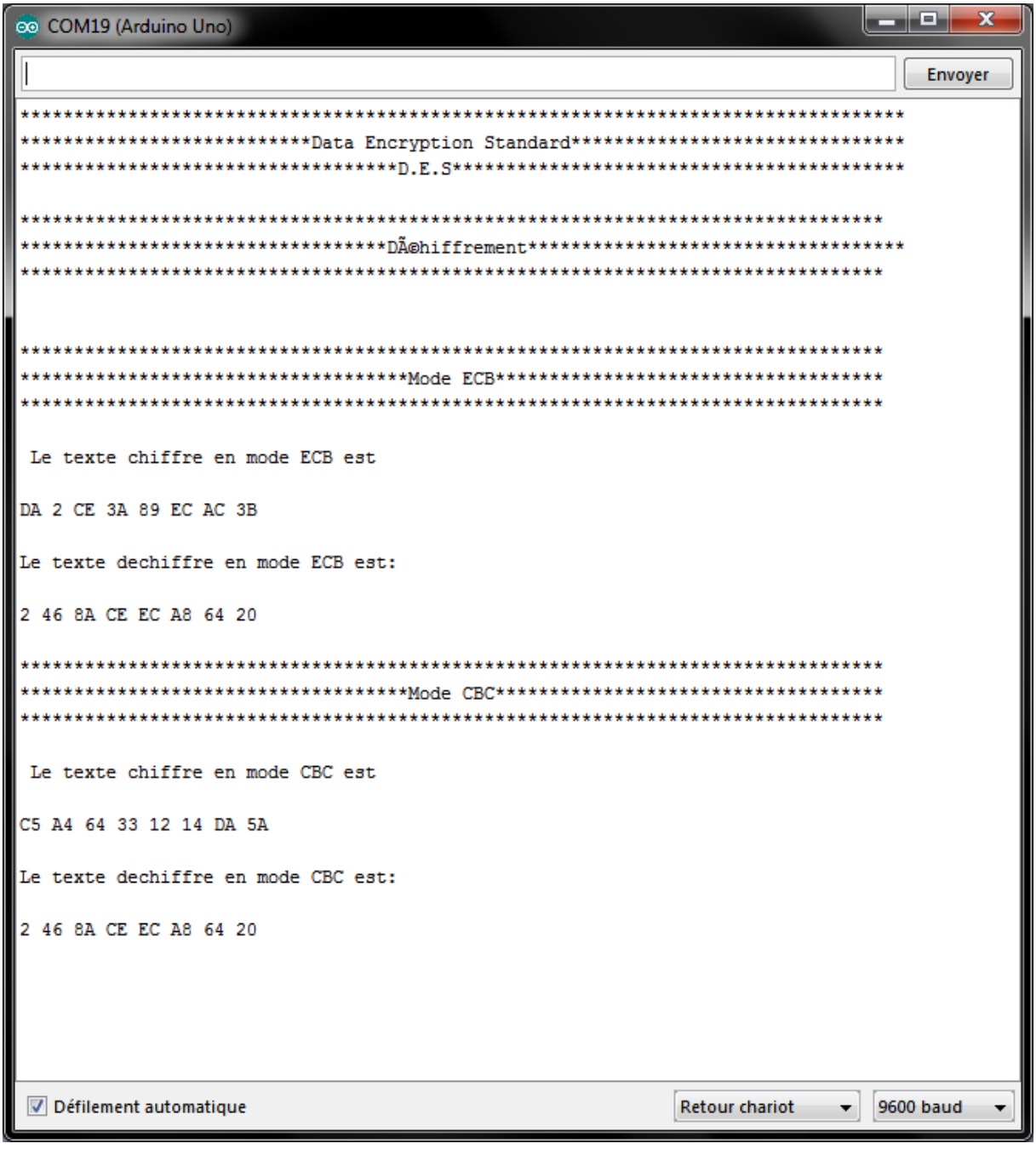


Figure IV.9 : Résultat du déchiffrement avec les deux modes ECB et CBC affiché sur le sérial monitor de l'Arduino.

Discussion

Les résultats pratiques obtenus nous permettent d'affirmer:

Que l'algorithme de chiffrement du DES « Data Encryption Standard » avec les deux modes ECB et CBC est bien implémenté dans la carte Arduino et chiffre correctement le plaintext. On note là aussi l'inconvénient du mode ECB soulevé précédemment à savoir que les mêmes blocs sont chiffrés de la même manière d, et la réussite du mode CBC à palier à ce problème.

La transmission série (TX, RX) entre la carte émettrice qui transmet les données après chiffrement vers et la carte réceptrice se fait de manière correcte,

La carte réceptrice déchiffre correctement les données reçues. Ceci nous permet de dire que l'algorithme de déchiffrement du DES est correctement implémenté sur la carte réceptrice.

Cependant, notre application est loin d'être complète, car ça serai intéressant :

1- D'ajouter des interfaces graphiques en utilisant un logiciel de programmation compatible avec l'IDE Arduino (EX : Visual C++).
2- Remplacer la transmission série (TX, RX) par une transmission sans fils (Wifi ou Bluetooth).
3- Surmonter le problème de gestion de mémoire de l'Arduino en utilisant une carte disposant de plus de ressource et plus performante (EX : Raspberry) ce qui va permettre alors d'implémenter notre cryptosystème hybride appliqué à la transmission d'images en complet.

IV.5 Conclusion

Ce chapitre clos ce travaille de mémoire est a comporté dans une première partie la présentation de la carte Arduino Uno utilisé pour l'implémentation pratique de notre cryptosystème hybride de transmission d'images.

Les ressources en termes de mémoire et de performances très limitées de l'Arduino, se sont avérées problématiques lors de l'implémentation de notre cryptosystème. Cependant, nous avons réussi à implémenter le **DES** « **D**ata **E**ncryption **S**tandard » l'algorithme de chiffrement à clé secrète avec les deux modes de chiffrement **ECB** et **CBC**, autour de deux cartes Arduino-Uno où la première a fait office d'émetteur et a comportée l'algorithme de chiffrement du **DES,** et la deuxième de récepteur et a comportée l'algorithme de déchiffrement du **DES** en utilisant une transmission série (TX, RX) entre elles.

Les résultats pratiques obtenus sur le chiffrement d'un texte démontrent l'implémentation correcte de l'algorithme de chiffrement et de déchiffrement dans les deux cartes Arduino-Uno, et confirment les résultats de simulation obtenus au chapitre II.

Conclusion générale

Notre travail rapporté dans ce mémoire a porté dans une première partie sur l'étude des techniques de cryptage basées sur les algorithmes de chiffrement classiques en particulier l'algorithme DES à clé secrète, l'algorithme RSA à clé publique et l'algorithme SHA-1 utilisant la fonction de hachage. Ces trois algorithmes sont nécessaires à la conception de notre cryptosystème.

Dans une seconde partie, nous avons proposé une méthode hybride de chiffrement et de déchiffrement appliquée au chiffrement d'images. Dans cette partie, nous avons élaboré un algorithme hybride associant les algorithmes DES, RSA et SHA-1. Nous avons donné en détail notre cryptosystème de transmission avec ses deux blocs « émetteur et récepteur », ainsi que son principe de fonctionnement. Puis, nous avons appliqué la technique hybride proposée au chiffrement d'images. Les résultats obtenus sous MATLAB ont permis d'illustrer le bon fonctionnement et l'efficacité de la méthode proposée.

Afin de tester notre cryptosystème et analyser sa sécurité, nous avons exploité plusieurs métriques d'évaluation du degré de chiffrement qui sont : analyse des différents histogrammes, le calcul des différentes corrélations entre l'image originale et son chiffré et entre les pixels adjacents, analyse de la sensibilité à la clé secrète, analyse différentielle et le test d'intégrité. Et ceci dans les deux modes de chiffrement ECB et CBC.

Les résultats obtenus nous rassurent sur la robustesse de notre cryptosystème. Cependant, quelques réserves sont émises quant à l'utilisation du mode ECB dans le chiffrement d'images de haute sécurité. Nous pouvons dire aussi que la fonction SHA-1 utilisée pour tester l'intégrité donne de très bons résultats.

Dans la troisième partie, nous avons proposé une réalisation expérimentale du cryptosystème hybride de transmission sécurisée proposé. Cette réalisation est conçue autour de deux cartes Arduino-Uno l'une étant l'émetteur et l'autre le récepteur. Les ressources limitées des cartes utilisées en termes de mémoire et de performances nous ont poussées à restreindre notre application pratique à l'implémentation de l'algorithme à clé secrète DES avec les deux modes ECB et CBC, appliqué au chiffrement d'un texte.

Les résultats expérimentaux relevés ont confirmé de la bonne implémentation des algorithmes de chiffrement et de déchiffrement sur les deux cartes Arduino Uno ainsi que la transmission série assurée des deux cartes Arduino-Uno.

Ce projet nous a permis d'aborder une thématique très attrayante qui est celle de la cryptographie et de la transmission sécurisée de données.

Les différent aspects que nous avons étudiés (cryptographie, traitement d'images, programmation sous MATLAB, réalisations pratiques matérielle et logicielle sous cartes Arduino pilotées par micro-ordinateur) 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 recherche publiés récemment.

Notre projet est loin d'être complet. En guise de perspectives, nous proposons :

- Utiliser des cartes plus élaborées comme les cartes ARDUINO Méga, Raspberry, ou autres cartes plus récentes et plus puissantes afin d'implémenter tout les algorithmes de notre cryptosystème hybride de transmission.
- Faire une étude comparative de cryptanalyse et de robustesse entre les résultats obtenus par le cryptosystème proposé et ceux obtenus par d'autres techniques de transmission sécurisée d'images.
- Introduire une étape de compression et de décompression d'image dans le cryptosystème pour réduire le volume d'information.
- Finaliser le banc expérimental afin qu'il puisse servir de support pour des travaux pratiques destinés à l'enseignement de la cryptographie.
- Envisager de développer et d'implémenter d'autres techniques de chiffrement associant en particulier celles basées sur le chaos et les courbes elliptiques.

Bibliographie

[01] J. Jean, « ***Cryptanalyse de Primitives Symétriques Basées Sur le Chiffrement AES*** » Thèse de Doctorat, université Paris Diderot (Paris 7), 2013.

[02] B. Martin, « ***Codage, Cryptologie et Applications*** », Presses Polytechniques et universitaires romandes, 2001.

[03] W. Stalling, « ***Cryptography and Network Security. Principes and Practice*** »,5th édition, Prentice Hall, 2010.

[04] R. Halit, M. Habachou « ***Conception et Réalisation d'un Cryptosystème Hybride*** ». Mémoire de fin d'études d'ingénieur en Electronique, Université MAMMERI Mouloud de Tizi Ouzou, 2008.

[05] S.B Hacini, M.T Inal « ***Implémentation d'Algorithmes de Cryptographie sous Java*** ». Mémoire de fin d'études de Licence, Université Abou Bakr Belkaid-Tlemcen, 2014.

[06] T. Ebrahimi, F Leprévost, B Warusfel. « ***Cryptographie et Sécurité des Systèmes et Réseaux*** ». Hermes science, 2006.

[07] N. Yassine, M. Ouyous, « ***Rapport sur l'Etude et l'Implémentation de Quelques Algorithmes de Chiffrement et de Signature*** ». Rapport, Université Mohammed V-AGDAL, Rabat. Maroc, 2014.

[08] T. Fuhr, « ***Conception, Preuve et Analyse de Fonctions de Hachage cryptographiques*** ». Thèse de Doctorat, TELECOM Paris Tech, 2011.

[09] Support de cours Master de cryptographie « ***Fonctions de Hachage et Applications*** », Université de Rennes. mars 2015.

[10] A .Gonsai, N. Kakkad, B. Goswami, N. Shah, ***« Study and Analusis of Symmmetric Key-Cryptograph DES, Data Encryption Standard »,*** Proceedings of UGC sponsored National Seminar on Scientific Wealth of Physics **SWP-2012**, Inde, 26 aout 2012.

[11] H. Hamiche, ***« Inversion à Gauche des Systèmes Dynamiques Hybride Chaotiques, Application à la Transmission Sécurisée de Données »,*** Thèse de Doctorat, université MAMMERI Mouloud de Tizi Ouzou, 2011.

[12] Data Encryption Standard (DES), Federal Information Processing Standard (FIPS) Publication 46-3, National Institute of Standard and Technology, U.S departement of commerce, 25 octobre 1999.

[10] J. Marconi, « ***Transfert Sécurisé d'Images par Combinaison de Techniques de Compression, Cryptage et Marquage*** ».Thèse de Doctorat, Université MONTPELLIER II, 2006.

[11] D.E Goumidi « ***Fonction Logistique et Standard Chaotique pour le Chiffrement des Images Satellitaires*** ». Mémoire de Magister, Université Mentouri de Constantine, 2010.

[12] Z. Amrani, S. Chitroub, A. Boukhari, « ***Cryptage d'Images par Chiffrement de Vigenère Basé sur le Mixage des Cartes Chaotiques*** »,4 th International Conference on Computer Integrated Manufacturing CIP'2007.Setif, Algérie. 03-04-Novembre 2007.

[13] M.B Boukhatem. « ***Application des Techniques de Cryptage pour La Transmission Sécurisée d'Images MSG*** ». Mémoire de Magister en électronique, université MAMMERI Mouloud, Tizi Ouzou, mars 2015.

[14] H. Dimassi, ***« Synchronisation des Systèmes Chaotique par Observateurs et Applications à la Transmission d'Informations »,*** Thèse de doctorat, université Paris Sud, 2012.

[15] M.Z Baba –Ahmed, F.Z Benmansour, M. Anane, ***« Conception d'Un Cryptosystème Pour les Transmissions des Données Chiffrées ».*** International Conference on Software Engineering and New Technologies, 2012.

[16] Y. Bouflih, S.Chitroub, ***« Transmission Sécurisée par le Système PGP en Vue de Protéger les Courriers Electroniques ».***Laboratoire de traitement du signal et des images, Faculté d'Electronique et d'Informatique, USTHB, Alger, Algérie.

[17] Y. Zhang, W. Liu, S. Cao, ***« Digital Image Encryption Algorithm Based on Chaos and Imroved DES »***, Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX, USA - Octobre 2009.

[18] A. Soleymani, Ali Zulkarnain, J Nordin, ***« A Survey on Principal Aspects of Secure Image Transmission »,*** *World Academy of Science, Engineering and Technology 66, 2012.*

[19] W. Tianfu, R. Babu, ***« Design of Hybrid Cryptographic Algorithm »,*** International Journal of Computer Science & Communication Networks,Vol 2(2), 277-283.

[20] J. Singh, K. Lata, J. Ashraf, ***« Image Encryption & Decryption with Symmetric Cryptography using MATLAB »***, International Journal of Current Engineering and Technology, Vol.5, No.1, 2015.

[21] J. Lechalupé. « ***Cours d'initiation à Arduino*** ». Université Paul Sabatier. Mai 2014.

[22] http://www.monclubelec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielUno Dernière consultation le 24/06/2015.

[23] M.A. Zerzri « ***Arduino et Simulink/Matlab un outil innovant à cout réduit pour le prototypage*** ».Article publié par EDP Sciences disponible sur le site : http://www.j3ea.org.

[24] M. Mc Roberts, ***« Begining Arduino »***, seconde édition, Apress, 2013.