Thèse en co-tutelle

pour obtenir le grade de

Docteur de l'Université de Montpellier II

ÉCOLE DOCTORALE INFORMATION, STRUCTURE ET SYSTEME

ET

Docteur de l'Université de Sfax

ÉCOLE DOCTORALE SCIENCES ET TECHNOLOGIES

Mention : INFORMATIQUE

par

Atef MASMOUDI

Elaboration de nouveaux algorithmes de crypto-compression basés sur le codage arithmétique

soutenue publiquement le 17 Décembre 2010

Jury :

M. Jean-Marc CHASSERY	Directeur de recherche, GIPSA, Grenoble, France	Rapporteur
M. Abdennaceur KACHOURI	Professeur, ISSIG, Gabes, Tunisie	Rapporteur
M. Mohamed Salim BOUHLEL	Professeur, ISECS, Sfax, Tunisie	Co-Directeur
M. William PUECH	Professeur, LIRMM, Montpellier, France	Co-Directeur
Mme. Caroline FONTAINE	Chargée de recherche, Télécom Bretagne, France	Examinatrice
M. Kamel HAMROUNI	Professeur, ENIT, Tunis, Tunisie	Examinateur

UNIVERSITÉ DE MONTPELLIER II (UM2)

ÉCOLE DOCTORALE I2S

INFORMATION, STRUCTURE ET SYSTEME

T H È S E

pour obtenir le grade de

Docteur en Sciences de l'Université de Montpellier II

Mention : Informatique

par

Atef Masmoudi

Elaboration de nouveaux algorithmes de crypto-compression basés sur le codage arithmétique

soutenue publiquement le 17 Décembre 2010

Jury :

M. Jean-Marc Chassery	Directeur de recherche, GIPSA, Grenoble, France	Rapporteur
M. Abdennaceur Kachouri	Professeur, ISSIG, Gabes, Tunisie	Rapporteur
M. Mohamed Salim Bouhlel	Professeur, ISECS, Sfax, Tunisie	Co-Directeur
M. William Puech	Professeur, LIRMM, Montpellier, France	Co-Directeur
Mme. Caroline Fontaine	Chargée de recherche, Télécom Bretagne, France	Examinatrice
M. Kamel Hamrouni	Professeur, ENIT, Tunis, Tunisie	Examinateur

Remerciements

Cette thèse est le fruit d'une collaboration entre l'unité de recherche SETIT (Sfax - Tunisie) et le laboratoire d'informatique de robotique et de microélectronique de Montpellier (Montpellier - France).

En premier lieu, je tiens à exprimer ma reconnaissance et ma gratitude à mes directeurs de thèse M. Mohamed Salim BOUHLEL, professeur à l'ISBS, et M. William PUECH, professeur à l'université Montpellier II et responsable du projet ICAR (Image et Interaction), pour m'avoir guidé dans mon travail et pour leur entière confiance. Je tiens également à les remercier pour leurs conseils et leurs rigueur scientifique, ce qui m'a aidé à progresser considérablement dans mes travaux. J'ai pour eux un grand respect.

Merci à M. PUECH, qui a des qualités humaines exceptionnelles, pour leur soutien quotidien et son encouragement pendant ces trois années de thèse. J'aimerais également lui dire à quel point j'ai apprécié sa grande disponibilité et son respect sans faille des délais serrés de relecture des documents que je lui ai adressés. C'est une personne très active et il est très agréable de travailler avec lui.

Merci à M. BOUHLEL pour la confiance qu'il m'a accordé pour la participation à l'organisation des journées scientifiques et la conférence internationale SETIT, ainsi que pour l'encadrement des projets de fin d'études et de Master. Je le remerci également pour son accueil chaleureux à chaque fois que j'ai sollicité son aide, ainsi que pour ses multiples encouragements, et pour l'excellente ambiance de travail.

Ensuite, je voudrais exprimer mes sincères remerciements aux membres du jury pour avoir accepté d'évaluer mon travail. Je remerci en particulier messieurs Jean-marc Chassery et Abdennaceur Kachouri d'avoir accepté la lourde tâche de relecture minutieuse de mon mémoire. Je remerci également Mme Caroline Fontaine, Chargée de recherche au CNRS, pour avoir acceptée de participer au jury de soutenance, et M. Kamel HAMROUNI, professeur à l'ENIT, pour son intérêt et pour sa présence fructueuse dans le jury en tant qu'examinateur.

Je tiens également à exprimer ma gratitude à tous ceux qui ont pu participer, tant d'un point de vue scientifique qu'humain, à ce travail. Je n'oubli bien sûr pas tous les doctorants et personnels des laboratoires SETIT et LIRMM pour les bons moments partagés aux laboratoires et à l'extérieur. Merci tout particulièrement à M. Ferid Rachdi, Directeur de recherche CNRS, pour ses conseils et pour leur entière confiance.

Je suis également reconnaissant envers mes amis de l'Institut Préparatoire aux Etudes

d'Ingénieurs de Sfax (IPEIS) pour la bonne ambiance et la convivialité qui règnent au sein de cet institut : le directeur (M. Maher Dammak), le directeur des études (M. Ali Ben Moussa), l'ensemble des enseignants (Ameur Ch'haider, Mourad Elloumi, Mohamed Ksantini, Zouhair Djemel, Lotfi Samet, Fethi Fourati, Jalel Kallel et Afif Masmoudi). Je remerci aussi les deux laboratoires SETIT et LIRMM, et l'IPEIS pour leur soutien financier.

Bien évidemment je ne pourrai terminer sans remercier chaleureusement mes amis, ma famille et plus particulièrement mes parents, Monia et Mohamed, pour leur confiance et leur soutien au cours de toutes ces années, et pour m'avoir supporté pendant ces longues études. Merci à ma sœur Fadoua et son mari Ahmed pour leurs encouragements et leurs aides précieuses lors de leurs relectures et corrections de ce mémoire. Je remerci infiniment ma femme Imen, toujours avec un grand sourire, qui m'a aidé tout au long de la préparation et de la rédaction de ce mémoire et qui a toujours été d'un soutien et d'une patience exceptionnels.

Que ceux que je n'ai pas cités ici et qui ont toujours une place dans mes pensées puissent m'excuser.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Aujourd'hui, nous vivons dans une société moderne où l'informatique est omniprésente. Ainsi, l'apparition de l'informatique, d'une part, et le développement des moyens et des protocoles de communications, d'autre part, ont transformé profondément la façon dont nous communiquons. Actuellement, nous utilisons de plus en plus le téléphone mobile pour échanger des messages textes, des images et même des vidéos. Nous communiquons également par le biais de l'Internet en échangeant des courriers électroniques, en participant à des forums de discussion et en exploitant des réseaux sociaux pour partager et échanger du contenu avec une personne située partout dans le monde d'une manière simple et rapide.

Grâce à cet avancement technologique, il devient possible d'archiver et d'échanger toute sorte d'information. Parmi ces informations, l'image occupe une place particulièrement importante. En effet, il existe actuellement plusieurs équipements qui sont capables d'acquérir, d'archiver et de transmettre des images avec des coûts très raisonnables ce qui favorise leur utilisation dans plusieurs domaines. De ce fait, le champ du traitement des images devient de plus en plus vaste et il couvre actuellement un très grand nombre d'applications comme la vision par ordinateur, l'imagerie médicale, l'imagerie satellitaire, l'Internet, la compression, la sécurité, la transmission, etc. Toutefois, l'image est caractérisée par une quantité impressionnante de données multimédia nécessitant par conséquent une importante capacité de stockage et une bande passante suffisamment grande pour assurer son transfert dans des délais raisonnables.

En effet, pour que les dernières technologies soient pratiques et à la portée de tous, il est donc indispensable de compresser les données. Cette compression à pour objectif de réduire le volume des données afin de permettre aux usagers d'archiver plus d'informations sur un même support de stockage, et de réduire le temps de transfert tout en réduisant les temps d'attente. Actuellement, il existe plusieurs standards pour la compression sans perte d'images, comme JPEG2000, JBIG et JBIG2. Ces standards ont un point commun : ils utilisent tous le codage arithmétique pour la génération des flux de données compressés,

ceci grâce à sa rapidité, la facilité de sa mise en œuvre, sa capacité de s'adapter à la statistique de la source et son efficacité en générant des codes de longueurs optimales. En plus de ces standards, il existe plusieurs autres techniques de compression qui sont basées sur le codage arithmétique. Par conséquent, le codage arithmétique sera le socle des techniques de codages utilisées dans les schémas de compression et de crypto-compression que nous avons développés dans ces travaux de recherches.

Parmi les autres problèmes que nous nous proposons de résoudre, suite à l'évolution spectaculaire de la production et de l'utilisation des images numériques, nous retrouvons celui du chiffrement. Ainsi, prenons le cas d'un utilisateur qui souhaite échanger, via Internet ou même à travers des réseaux locaux, des données confidentielles. Ces données peuvent être facilement interceptées voir même altérées ou modifiées par des attaquants qui partagent le même canal de communication. Ainsi, afin de garantir un échange efficace et sécurisé, les mesures à mettre en œuvre doivent assurer certains services comme la confidentialité qui garantit de garder secrète une communication et d'accorder un accès aux entités autorisées. Le contexte dans lequel s'insère notre travail de thèse est la combinaison des techniques de compression et de chiffrement.

Un système de compression sans perte est généralement décomposé en quatre étapes. La première étape consiste à appliquer une transformation réversible sur les données à coder afin de représenter les données originales d'une manière plus adaptée pour la compression. Ensuite, la deuxième étape consiste à convertir les données issues de la transformation en des symboles en vue de les coder d'une manière plus efficace. Puis, la troisième étape est la modélisation de la source et vise à modéliser la fonction de distribution de probabilités des symboles à coder. Enfin, la quatrième étape consiste à utiliser un codage à longueur variable pour représenter chaque symbole avec un code de longueur optimale tout en exploitant les probabilités estimées par l'étape précédente.

Selon cette décomposition, et afin de sécuriser le transfert et l'archivage des images, un cryptosystème peut être intégré à n'importe quelle étape d'un système de compression sans perte. Toutefois, il est à noter que le chiffrement des pixels de l'image, ou bien le chiffrement des symboles générés par la transformation conduit à une réduction considérable de l'efficacité du codage puisqu'il modifie les statistiques des données à coder qui auront généralement une entropie maximale après le cryptage. De plus, l'application de l'algorithme de chiffrement sur les données issues du codeur entropique engendre une augmentation considérable dans le temps de calcul et permet également de priver le codeur entropique de ses particularités comme le codage progressif.

Pour surmonter ces problèmes, il est préférable d'appliquer le chiffrement dans le codage entropique. Ainsi, un système de crypto-compression propose généralement de modifier soit des paramètres de codage soit certaines fonctions afin d'appliquer conjointement la compression et le cryptage dans une même étape.

Ce sont toutes ces considérations qui ont guidé cette thèse. Ainsi, notre objectif est de proposer de nouvelles techniques de compression, de chiffrement et de crypto-compression pour le domaine de l'imagerie numérique. Les nouvelles techniques de compression proposées sont basées sur le codage arithmétique. Aussi, allons-nous montrer que ce type de codage peut être constamment amélioré. Quant aux techniques de chiffrement, nous avons proposé d'utiliser la théorie du chaos pour développer, en premier lieu, un nouveau générateur de nombres pseudo-aléatoires et pour mettre en place, en deuxième lieu, un nouvel algorithme de chiffrement d'images basé sur la génération déterministe des flux de clés dynamiques. Ces clés sont caractérisées par des propriétés statistiques et chaotiques très appropriés en cryptographie.

Ce mémoire est organisé de la façon suivante. Tout d'abord, nous dressons dans le chapitre 2 un état de l'art des techniques et standards de compression les plus couramment utilisés tout en présentant la théorie sur laquelle s'appuie tout système de compression. Le chapitre 3 introduit les notions de bases de la cryptographie tout en détaillant les différentes techniques adoptées dans l'analyse des performances des générateurs de nombres pseudo-aléatoires et des algorithmes de chiffrement d'images. Nous aborderons par la suite nos contributions à la compression. Ainsi, le chapitre 4 présente un nouveau codage arithmétique adaptatif pour la compression sans perte d'images. Ce dernier a permis d'augmenter les performances des codeurs arithmétiques conventionnels de 5% tout en réduisant les temps de calcul relatif aux estimations et aux mises à jour des probabilités. Le chapitre 4 détaille également les modifications que nous avons apportées au codage arithmétique adaptatif d'ordre 0, en réduisant la précision de représentation des fréquences des pixels et en codant l'image bloc par bloc après les avoir triés selon des calculs basés sur la distance de Kullback. Ces modifications nous ont permis d'améliorer les performances de ce codeur, en terme de taux de compression, de 6% à 52%. Les résultats expérimentaux présentés dans le chapitre 4 sont obtenus en utilisant une base de données d'images standard qui est souvent utilisée dans l'évaluation des performances des algorithmes de compression. Le chapitre 5 se concentre sur la génération déterministe des flux de clés dynamiques utilisées dans les algorithmes de chiffrement par flot. La première partie du chapitre 5, section 5.2, expose notre nouveau générateur de nombres pseudo-aléatoires.

Ce dernier est construit à partir du système chaotique Chaotic Standard Map couplé avec la fonction de développement en fractions continues d'Engel. Les analyses expérimentales basées sur la série de tests de NIST ont montré que notre générateur est extrêmement efficace et il est capable de générer des séquences de valeurs ayant des propriétés chaotiques et statistiques très intéressantes. Ces propriétés favorisent son utilisation dans le chiffrement. Ainsi, nous avons proposé, section 5.3, un nouvel algorithme de chiffrement d'images tout en exploitant le générateur pseudo-aléatoire proposé. Les résultats expérimentaux montrent l'efficacité de cet algorithme et sa robustesse contre plusieurs types d'attaques. Finalement, nous proposons, section 5.4, un nouveau schéma de crypto-compression basé sur l'utilisation d'un codage arithmétique binaire avec le générateur pseudo-aléatoire proposé. Dans ce nouveau schéma, la compression est réalisée grâce à un codage arithmétique binaire, pour les deux modes statique et adaptatif, alors que le chiffrement est effectué en échangeant aléatoirement les deux sous-intervalles associés aux symboles binaires. Cet échange est assuré par notre générateur tout en conservant les probabilités des symboles à chaque étape du codage et du décodage. Le nouveau schéma proposé est très efficace et il conserve, d'une part, les performances du codage arithmétique binaire, et d'autre part, les particularités de ce codeur comme la progressivité et sa capacité d'être employé avec plusieurs modèles statistiques. Nous terminons ce mémoire par une conclusion générale où nous suggérons également quelques pistes à venir.

Chapitre 2

La compression

Sommaire

2.1 Introduction

Ce chapitre présente un état de l'art sur l'ensemble des techniques et des outils utilisés pour le codage de source ou la compression des données et plus particulièrement la compression sans perte d'images. La compression a pour objectif de réduire le volume des données tout en préservant l'information importante. Cette réduction conduit à l'archivage de plus d'information sur un même support de stockage, et de minimiser le temps des transferts via des réseaux de télécommunication.

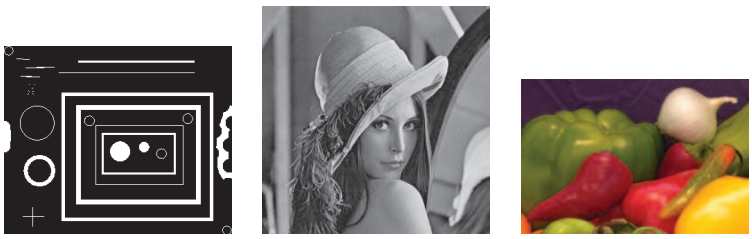
Dans ce chapitre, nous présentons des techniques de compression aussi bien pour des images en niveaux de gris et couleurs que pour des images binaires. Les outils de compression présentés dans ce chapitre forment la matière introductive à mes travaux de recherches. Dans ce chapitre, nous présentons d'abord, section 2.2, une introduction à l'image numérique de façon à justifier le besoin croissant de la compression. La section 2.3 présente les concepts de base de la compression d'images. Ces concepts couvrent les mesures de performance, l'introduction à la théorie de l'information et la présentation des deux techniques de compression sans et avec pertes. La section 2.4 présente la structure générale d'un système de compression sans perte. Dans la section 2.5 nous présentons les codeurs entropiques les plus connus, tout en détaillant en particulier le codage arithmétique (CA). La section 2.6 couvre un grand nombre de standards de compression d'images qui utilisent le CA pour la génération des flux de données compressées. Finalement, nous terminons ce chapitre par une conclusion, section 2.7.

2.2 Image numérique

2.2.1 Définition de l'image numérique

Aujourd'hui, l'imagerie numérique [Gonzalez 2002, Russ 2006, Gonzalez 2008, Bovik 2009] devient de plus en plus indispensable dans plusieurs domaines et essentiellement dans la communication entre personnes. En effet, le développement exponentiel des média de communication d'une part, et des supports de stockages numériques d'autre part, ont énormément transformé la façon dont nous communiquons. Ces nouvelles technologies sont basées essentiellement sur l'échange et le stockage efficaces des données multimédia et en particulier les images numériques. Ces dernières sont en fait des matrices de points appelés pixels de l'image. Un pixel est décrit par un ou plusieurs nombres, représentant

l'intensité de l'image en ce point. Il existe plusieurs types d'images numériques. Les images binaires (Figure 2.1 (a)) sont des matrices de pixels qui ne peuvent prendre que deux valeurs 0 ou 1 ce qui correspond respectivement aux deux couleurs noir et blanc. Une image en niveaux de gris (Figure 2.1 (b)) est une image monochrome en tons continus, offrant plusieurs niveaux d'intensité allant du noir vers le blanc. Ces images sont codées généralement sur 8 bits, et dans ce cas nous obtenons 256 intensités allant du 0 à 255, sachant que l'intensité 0 représente le noir et l'intensité 255 représente le blanc. Concernant les images couleurs (Figure 2.1 (c)), chaque pixel est représenté par un vecteur en général constitué de 3 composantes. Une représentation très répandue est la représentation dans l'espace RGB (Red Green Blue) qui considère l'image couleur comme une superposition de trois plans monochromes traités indépendamment. En général, 8 bits sont utilisés pour indiquer l'intensité du pixel pour chacune des composantes rouge, verte et bleue.



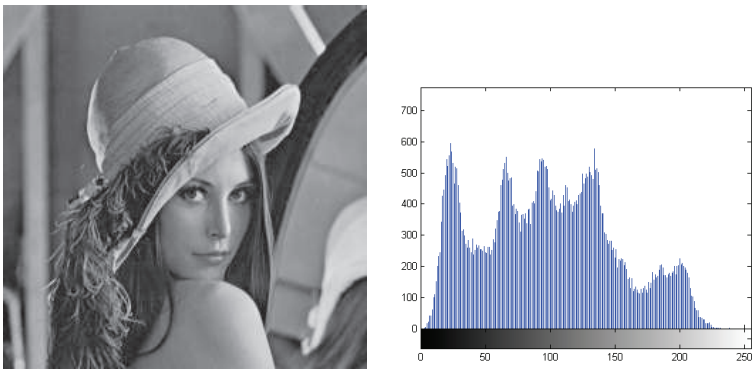
(a) (b) (c)

FIGURE 2.1 – a) Image binaire, b) Image avec 8 niveaux de gris, c) Image couleur.

2.2.2 Histogramme

Soit I une image composée de N lignes et M colonnes et ayant L niveaux de gris. L'histogramme [Burger 2009] de I est une fonction discrète définit par $h(k) = n_k$ avec k représentant le $k^{\text{ème}}$ niveau de gris de l'intervalle $[0, L-1]$, et n_k représente le nombre de pixels de l'image I ayant le niveau de gris k. En pratique, c'est généralement l'histogramme normalisé qui est utilisé. Ce dernier se calcule après la division de chacune de ses valeurs par le nombre total de pixels présents dans l'image. Ainsi, l'histogramme normalisé est donné par $p(k) = \frac{n_k}{NM}$, pour tout $k = 0, 1, ..., L-1$. Notons que la $k^{\text{ème}}$ valeur de l'histogramme normalisé présente une estimation de la probabilité du $k^{\text{ème}}$ niveau de gris et que la somme de toutes les valeurs de l'histogramme normalisé est égale à 1. Plusieurs applications du traitement numérique des images sont appliquées dans le domaine spatial tout en se basant sur l'histogramme. Ce dernier, simple à calculer, fournit

un moyen efficace pour extraire des informations statistiques sur l'image qui sont utiles pour certaines applications notamment pour la compression et le cryptage d'images. La figure 2.2 présente l'image de Lena en niveaux de gris, de taille 256×256 pixels ainsi que son histogramme.



(a) (b)

FIGURE 2.2 – a) Image de Lena en niveaux de gris, de taille de 256×256 pixels, b) Histogramme de l'image de Lena.

2.2.3 Corrélation entre pixels voisins

La corrélation entre pixels voisins ou bien la redondance spatiale exprime la dépendance d'un pixel ou d'une zone de l'image avec son environnement. En réalité, les pixels d'une image numérique sont généralement fortement corrélés avec leurs pixels voisins et en particulier avec ceux qui se trouvent sur la même ligne, la même colonne ou bien la même diagonale. La corrélation spatiale entre les pixels voisins de la même image est donnée par :

$$\gamma_{pq} = \frac{\frac{1}{R}\sum_{i=1}^{R}(p_i - E(p))(q_i - E(q))}{\sqrt{\frac{1}{R}\sum_{i=1}^{R}(p_i - E(p))^2}\sqrt{\frac{1}{R}\sum_{i=1}^{R}(q_i - E(q))^2}}, \tag{2.1}$$

avec $p = (p_1, ..., p_R)$ et $q = (q_1, ..., q_R)$ représentent respectivement le vecteur des pixels de l'image I et le vecteur des pixels voisins, R le nombre total de pairs (p_i, q_i) obtenu à partir de l'image, et $E(x) = \frac{1}{R}\sum_{i=1}^{R} x_i$ est la valeur moyenne de x. Par exemple, le Tableau 2.1 contient 3 coefficients de corrélation de l'image de Lena. La valeur absolue de chacun de ces coefficients est trés proche de 1 ce qui signifie qu'il existe une forte dépendance linéaire entre les pixels voisins de l'image. Nous pouvons, par conséquence, conclure que les pixels voisins sont fortement corrélés.

TABLE 2.1 – Les coefficients de corrélation entre les pixels voisins de l'image lena

Direction	Corrélation
horizontale	0.9706
verticale	0.9464
diagonale	0.9186

2.2.4 Nécessité de comprimer les images

Prenons le cas d'un utilisateur qui souhaite interroger à distance une base de données d'images en couleur avec 24 bits par pixel et ayant une taille de 2048×2048 pixels. Supposons que l'utilisateur est connecté au réseau avec un débit de 2Mbits/s. Sachant qu'une image nécessite 12Mo pour son stockage et plus que 48 secondes, théoriquement, pour son transfert, ce qui rend l'accès à 10000 images nécessite environ 6 jours de connexions pour leurs transferts. Le Tableau 2.2 présente le volume de données nécessaire pour le stockage d'une variété d'images sous leurs formes brutes (sans compression) et avec différentes dimensions.

Ainsi, les images numériques sont des données multimédia volumineuses nécessitant une capacité de stockage et une bande passante considérable pour la transmission. L'objectif de la compression est d'augmenter la capacité de stockage et de réduire le temps de transmission de manière à pouvoir archiver et transmettre plus de données tout en conservant les mêmes supports physiques. Les algorithmes de compression opèrent de sorte à éliminer les redondances présentées dans l'image afin de la représenter de la manière la plus compacte. De ce fait, il est intéressant de soumettre les pixels d'une image à une transformation, comme par exemple, le codage prédictif ou bien la décomposition en sous bandes afin de décorréler les données. Nous verrons, section 2.6, les transformations utilisées dans les standards actuels de compression sans perte d'images. Il est à rappeler que les nouvelles données obtenues par une transformation sont généralement modélisées avec une fonction de probabilités, et par la suite codées par un codeur entropique qui génère les données compressées.

2.3 Concepts de la compression d'images

La compression consiste à représenter une séquence de symboles de la manière la plus compacte possible, ce qui permet de réduire le nombre de bits nécessaire pour leur

TABLE 2.2 – Volume de données nécessaire pour le stockage d'images numériques en fonction de leur taille.

Dimension	Nombre de bits par pixel	type de l'image	volume de données (octets)
128×128	1	Binaire	$2Ko$
256×256	1	Binaire	$8Ko$
512×512	1	Binaire	$32Ko$
1024×1024	1	Binaire	$128Ko$
2048×2048	1	Binaire	$512Ko$
128×128	8	Niveaux de gris	$16Ko$
256×256	8	Niveaux de gris	$64Ko$
512×512	8	Niveaux de gris	$256Ko$
1024×1024	8	Niveaux de gris	$1Mo$
2048×2048	8	Niveaux de gris	$4Mo$
128×128	24	Couleur	$48Ko$
256×256	24	Couleur	$192Ko$
512×512	24	Couleur	$768Ko$
1024×1024	24	couleur	$3Mo$
2048×2048	24	couleur	$12Mo$

stockage et de diminuer les temps de transfert [Taquet 2010]. Notons que, les domaines d'application de la compression sont très nombreux, dont voici une liste non exhaustive :

- La photographie numérique
- L'imagerie médicale
- La télédétection
- La vidéosurveillance
- Le cinéma numérique
- Les archives d'images et les bases de données
- L'Internet
- ....

Ces domaines peuvent être classés en deux catégories : télécommunication et stockage. En télécommunication, la qualité et la vitesse de transmission des données sont conditionnées par les caractéristiques physiques du support de transmission. Ces caractéristiques définient la bande passante qui représente le nombre de bits par seconde que l'on peut transmettre sur ce support. Par exemple, pour les modems qui sont relativement lents

et ayant une faible bande passante, la compression de données s'impose afin d'augmenter virtuellement la bande passante du modem et d'atteindre des taux de transfert plus intéressant sans augmenter physiquement la bande passante.

Comme dans le cas d'un support de communication qui définit le nombre maximum de bits que l'on peut transmettre par seconde, un support de stockage d'information est défini par le nombre maximum de bits qu'il peut stocker et qui représente sa capacité. Certes, la capacité des supports de stockage numérique ne cessent d'augmenter, toutefois, en parallèle les nouvelles technologies, poussées par la télévision numérique et la téléphonie portable, sollicitent des données multimédia de plus en plus grandes. Ceci amène les chercheurs à trouver de nouveaux algorithmes de compression plus performants et qui doivent pouvoir s'adapter aux nouveaux usages. Enfin, et pour évaluer les performances d'une technique de compression, il faut mesurer plusieurs critères. Parmi ces critères nous citons essentiellement le taux de compression, la distorsion, la complexité de l'algorithme et la facilité de mise en œuvre.

2.3.1 Mesures de performance

Le taux de compression est une grandeur qui calcule le rapport entre le nombre de bits des données originales avec celui présent dans les données compressées. Le taux de compression est noté par $Taux$:

$$Taux = \frac{\text{La taille totale en bits des données originales}}{\text{La taille totale en bits des données compressées}}. \tag{2.2}$$

Il y a également un autre facteur pour mesurer la performance d'une technique de compression appelé le débit binaire moyen et exprimé en bits par pixel (bpp). En fait, ce dernier consiste à calculer le nombre moyen de bits nécessaire pour décrire un symbole des données originales :

$$Débit = \frac{\text{La taille totale en bits des données compressées}}{\text{Le nombre total de symboles dans les données originales}}(bpp). \tag{2.3}$$

Dans le cas des méthodes de compression avec pertes, il existe d'autres facteurs de mesure de performance. Il s'agit généralement des mesures de distorsion telles que l'Erreur Quadratique Moyenne EQM [1] et le rapport crête signal sur bruit [2].

1. (en anglais MSE pour Mean Square Error)
2. (en anglais PSNR pour Peak Signal to Noise Ratio)

Notons que ces critères permettent d'établir une mesure globale pour l'image. Le critère EQM se calcule comme la moyenne des carrés des différences entre les pixels de l'image après compression Ic et ceux de l'image originale I :

$$EQM = \frac{1}{NM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left(Ic(i,j) - I(i,j)\right)^2, \tag{2.4}$$

avec N et M représentent respectivement le nombre de lignes et le nombre de colonnes de l'image originale.
Le second critère d'évaluation, directement déduit de l'EQM, est le PSNR :

$$PSNR = 10 \log_{10} \frac{max\,(I)^2}{EQM}. \tag{2.5}$$

Les deux mesures, l'EQM et le PSNR, ne sont pas parfaitement compatibles avec le SVH [Girod 1993, Eskicioglu 1995, Eckert 1998]. Ainsi, Wang *et al.* [Wang 2004] ont proposé une nouvelle mesure de similarité entre une image originale I et une image compressée Ic, plus appropriée au SVH, notée par SSIM [3] :

$$SSIM(I, Ic) = \frac{(2\mu_I\mu_{Ic} + c_1)(2\sigma_{IIc} + c_2)}{(\mu_I^2 + \mu_{Ic}^2 + c_1)(\sigma_I^2 + \sigma_{Ic}^2 + c_2)}, \tag{2.6}$$

avec μ_I et σ_I sont respectivement la moyenne et l'écart type de I, alors que μ_{Ic} et σ_{Ic} représentent les mêmes mesures, mais pour Ic. De plus, σ_{IIc} est la covariance entre I et Ic, et c_1 et c_2 sont deux variables destinées à corriger la division quand le dénominateur est très proche de zéro.

Il est à signaler qu'il n'y a pas que ces critères pour évaluer la performance d'une méthode de compression qu'elle soit avec ou sans perte. En effet, la vitesse du codage et du décodage, la complexité de l'algorithme et la facilité de mise en œuvre, l'adaptation aux besoins de l'utilisateur, la possibilité de la transmission progressive par qualité ou par résolution, et d'autres critères contribuent également à l'étude de la performance des algorithmes de compression.

Nous présentons, section 2.3.2, les principes de bases de la théorie de l'information de Shannon ainsi que la fonction d'entropie utilisée pour mesurer la quantité d'information transportée par des sources aléatoires sans mémoires.

3. Structural SIMilarity

2.3.2 Théorie de l'information

Cette section se limite à fournir une représentation intuitive des notions de bases de la théorie de l'information. Pour plus d'informations, le lecteur pourra trouver plus de détails, avec des preuves mathématiques dans [Cover 1991].

La théorie de la compression des données a été fondée par Claude E. Shannon en 1948 [Shannon 1948a, Shannon 1948b]. Shannon a montré qu'il existe une limite théorique dans la compression sans perte de données. En effet, les techniques de compression sans pertes visent à minimiser le nombre moyen de bits nécessaire pour coder chaque symbole d'une source d'information sans introduire de pertes dans les données originales. La théorie de l'information, fondée par Shannon, définie donc une limite minimale théorique pour ce nombre moyen de bits. Ainsi, considérons une source d'information S qui prend ses valeurs dans un alphabet fini A composée de k symboles $A = \{s_1, \cdots, s_k\}$. La source S est capable de produire des suites de symboles $\{x_1, \cdots, x_n\}$ avec x_i représente le symbole émis à l'instant i et correspond à un symbole s_j de la source S. Supposons qu'à n'importe quel moment, la probabilité du symbole émis s_i est notée par $p_i = P(S = s_i)$, tel que $\sum_{i=1}^{k} p_i = 1$. Dans ce cas, une source est dite stationnaire si la loi de probabilités des symboles est indépendante du temps. De plus, une source est dite sans mémoire si les symboles générés correspondent à des variables aléatoires indépendantes et identiquement distribuées (i.i.d). C'est à dire, la génération d'une suite de symboles $\{x_1, \cdots, x_n\}$ est régie par la même loi de probabilités. Pour une source i.i.d nous avons :

$$p(x_1, \cdots, x_n) = p(x_1) \times \cdots \times p(x_n). \tag{2.7}$$

La quantité d'information associée au symbole s_i $(1 \leq i \leq k)$, notée par l'information propre est définie par :

$$\begin{aligned} I_{s_i} &= log_2(\tfrac{1}{p_i})\ (bits) \\ &= -log_2(p_i)\ (bits). \end{aligned} \tag{2.8}$$

Cette définition implique que la quantité d'information transportée par un symbole s_i $(1 \leq i \leq k)$ est inversement proportionnelle à sa probabilité. Ainsi, elle est plus importante $I_{s_i} \to \infty$ si $p_i \to 0$ (événement incertain), par contre, l'information propre d'un symbole s_i est nulle $I_{s_i} = 0$ si $p_i = 1$ (événement certain). De même, I_{s_i} est d'autant plus grande lorsque p_i est plus petite, et inversement. D'autre part, la quantité d'information transportée par une suite de symboles i.i.d est égale à la somme des informations propres de chaque symbole. La quantité d'information moyenne par symbole transportée par une

source S est appelé l'entropie de S et notée par $H(S)$. L'entropie $H(S)$ est définie par l'expression suivante :

$$H(S) = -\sum_{i=1}^{k} p_i \log_2(p_i) \ (bits/symbole). \tag{2.9}$$

Notons que, pour une source S ayant une loi de probabilités uniforme $p_i = \frac{1}{k}$ $(1 \leq i \leq k)$, l'entropie est maximale et elle est égale à $log_2(k)$ bits par symbole. Ainsi, pour pouvoir réduire le nombre moyen de bits par symbole, une technique de compression représente le symbole le plus fréquent par le mot binaire le plus petit. Supposons que nous utilisons une technique de compression sans perte notée C qui représente chaque symbole s_i $(1 \leq i \leq k)$ par un mot binaire m_i composé de l_i bits. Dans ce cas, chaque symbole s_i $(1 \leq i \leq k)$ généré par une source S, sera remplacé par le mot binaire m_i, et le flux binaire final se construit à partir de la concaténation de l'ensemble des mots binaires correspondant aux symboles émis. Afin d'assurer un décodage sans ambiguïté des mots binaires, il faut que ces derniers vérifient la condition d'unicité. Ainsi, pour garantir que le flux binaire final soit décodable de manière unique, il faut utiliser un code préfixe qui impose qu'aucun mot binaire ne soit le préfixe d'un autre. Notons l_c le nombre moyen de bits obtenu par la technique de compression C :

$$l_c = \sum_{i=1}^{k} p_i l_i \ (bits/symbole). \tag{2.10}$$

Ainsi, pour n'importe quelle technique de compression sans perte C, l'entropie $H(S)$ présente une limite minimale pour l_c :

$$l_c \geq H(S). \tag{2.11}$$

L'entropie fournit donc une limite inférieure du nombre moyen de bits nécessaire pour coder chaque symbole d'une source sans mémoire. Il est à noter qu'il est toujours possible de trouver un code préfixe décodable et capable de représenter chaque symbole par un nombre moyen de bits l_c à un bit prés de l'entropie :

$$H(S) \leq l_c \leq H(S) + 1. \tag{2.12}$$

Ce dernier résultat est garantit grâce à l'inégalité de Kraft [Rissanen 1976, Cover 1991] donnée par l'expression suivante :

$$\sum_{i=1}^{k} 2^{-l_i} \leq 1. \tag{2.13}$$

Il est également possible de réduire davantage la limite supérieure du l_c par la transformation de la source S composée par les symboles $\{s_1, \cdots, s_k\}$ en une source $S^{(M)}$ composée par les symboles $\{s_1^{(M)}, \cdots, s_r^{(M)}\}$, avec $s_i^{(M)}$ est un vecteur de la forme $(x_1, \cdots, x_M)$ et $x_i \in S$ $(1 \leq i \leq M)$. La source S est composée par k symboles, alors que la source $S^{(M)}$ est composée par k^M symboles. Ainsi, si nous utilisons une technique de codage C pour coder des blocs de M symboles, alors nous obtenons un nombre moyen de bits l_c de plus en plus proche de l'entropie lorsque M tend vers l'infini :

$$H(S) \leq l_c \leq H(S) + \frac{1}{M}. \tag{2.14}$$

Selon la théorie de l'information, il est toujours possible de compresser sans perte une suite de symboles si cette dernière présente une forte redondance. En particulier, les pixels d'une image sont généralement fortement corrélés avec leurs pixels voisins (redondance spatiale), ce qui conduit au développement de plusieurs techniques de compression sans perte d'images basées sur une étape de décorrélation pour minimiser l'information redondante et par conséquent réduire le nombre moyen de bits par pixel. Dans la section 2.4 nous détaillons la structure d'un système de compression sans perte, alors que dans la section 2.6 nous proposons une brève description des derniers standards de compression sans perte d'images.

2.3.3 La compression sans perte

L'objectif de la compression sans perte [4] [Karam 2000] est de représenter une séquence de symboles avec le nombre de bits le plus petit possible sans perte d'aucune information. Dans la compression sans perte d'images, l'image reconstruite doit être numériquement exactement identique à l'image originale. Il est important de noter que dans certains domaines de l'imagerie numérique, telles que l'imagerie médicale ou bien l'imagerie satellitaire, la réduction de la taille des images tout en conservant la qualité originale est une nécessité. Par exemple, en imagerie médicale, l'utilisation d'une méthode de compression avec pertes peut introduire des distorsions conduisant à un diagnostique erroné. Ainsi, les images médicales seront compressées sans perte, ce qui garantit une reconstruction exacte de chaque pixel. De plus, en imagerie satellitaire, les informations sont obtenues avec des coûts très élevés favorisant l'utilisation des techniques de compression sans perte.

Actuellement, il existe plusieurs techniques et outils de compression sans perte. Parmi ces techniques, il y a celles qui s'appliquent directement sur l'image sous forme brute,

4. (en anglais lossless compression)

et celles qui sont appliquées à l'image après avoir subies certaines transformations. Les techniques de compression sans perte d'images peuvent être classées en trois catégories : codage universel, codage à base de dictionnaire et codage statistique (appelé également codage entropique).

Codage universel :

Les codeurs universels, comme le codage d'Elias [Elias 1975] et celui de Golomb, sont des techniques de codage à longueur variable avec une structure régulière et indépendante des symboles à coder. Dans ce type de codage, nous transformons d'abord les symboles à coder en des entiers positifs. Ensuite, nous associons à chacun de ces entiers un mot binaire qui est construit à partir d'une table des probabilités ayant une structure particulière basée sur une distribution décroissante des entiers à coder. En d'autres termes, les entiers les plus petits sont supposés plus probables que les entiers les plus grands, et par conséquent ils peuvent être représentés par les mots binaires les plus courts.

Codage à base de dictionnaire :

Les techniques de codage à base de dictionnaire les plus connues ont été élaborées par Ziv et Lempel, souvent notées par LZ*. Ils ont proposé en 1977 l'algorithme LZ77 [Ziv 1977]. Après une année, ils ont proposé l'algorithme LZ78 [Ziv 1978] et plus tard LZW [Welch 1984], une version améliorée du LZ78 et proposée par Welch en 1984. Ces techniques sont construites à partir d'un même algorithme de codage et elles ont été largement exploitées par des outils de compression, à savoir gzip sous Linux, pkzip sous Dos, winzip sous Windows, et surtout dans les formats d'images TIFF [5] et GIF [6]. Il est à noter que ces techniques de codage ne nécessitent pas une connaissance *a priori* de la distribution de probabilités des symboles à coder. Cependant, elles construisent dynamiquement un dictionnaire composé par des symboles de longueures variables. A chaque symbole est associé un mot binaire obtenu par la représentation binaire de son index dans le dictionnaire construit.

Codage basé sur la modélisation statistique :

Nous proposons de détailler ce type de codage section 2.5.

5. Tagged Image File Format
6. Graphic Interchange Format

2.3.4 Compression avec pertes

Dans le but d'achever des taux de compression très élevés, la compression avec pertes [7] tente à éliminer les redondances de façon à fournir des images visuellement identiques, mais numériquement différentes. Dans ce cas, une perte d'information dans l'image reconstruite est autorisée si sa qualité visuelle reste identique par rapport à l'image originale. Ce jugement est effectué avec la mesure de certains critères, comme l'EQM, le PSNR et le SSIM présentés respectivement par les équations 2.4, 2.5 et 2.6, qui prennent en considération les caractéristiques du système visuel humain (SVH). Les principales méthodes de compression avec pertes reposent sur des techniques de décomposition et sur des méthodes de quantification. Les techniques de décomposition sont appliquées grâce à une technique d'analyse de fréquence comme la transformation en cosinus discrète, utilisée par la norme de compression avec perte JPEG [Wallace 1991, Pennebaker 1992], ou, plus récemment, grâce aux techniques à base d'ondelettes utilisées, par exemple, par la norme JPEG2000 [Gormish 2000, Taubman 2000, Skodras 2001, Taubman 2002].

2.3.5 Choix de l'approche de compression

De ce qui précède, il existe deux approches de compression : la première dite réversible, alors que la deuxième est irréversible. Cette dernière exploite les caractéristiques du SVH pour aboutir à des taux de compression très élevés et cela avec une dégradation acceptable. Dans cette thèse, nous avons choisi d'étudier et d'élaborer des approches de compression sans perte. Bien que les performances des méthodes de compression sans perte sont limitées, nous trouvons, actuellement, plusieurs méthodes et standards de compression sans pertes qui visent à augmenter les taux de compression. De plus, un des objectifs de ces standards est d'apporter de nouvelles fonctionnalités aux systèmes de compression, pour pouvoir s'adapter aux besoins de certaines applications, et de réduire les coûts de traitement et la complexité d'implémentation des algorithmes de codage.

Par la suite de ce chapitre, nous apportons un intérêt particulier aux techniques de codage sans perte. Ainsi, nous consacrons la section 2.4 pour introduire les concepts de base d'un système de compression sans perte, alors que la section 2.5 présente les algorithmes de codage entropique, comme le CA qui est utilisé pour obtenir des codes optimaux. Finalement, la section 2.6 expose en bref les derniers standards de compression d'images sans pertes.

7. (en anglais, lossy compression)

2.4 Structure d'un système de compression sans perte

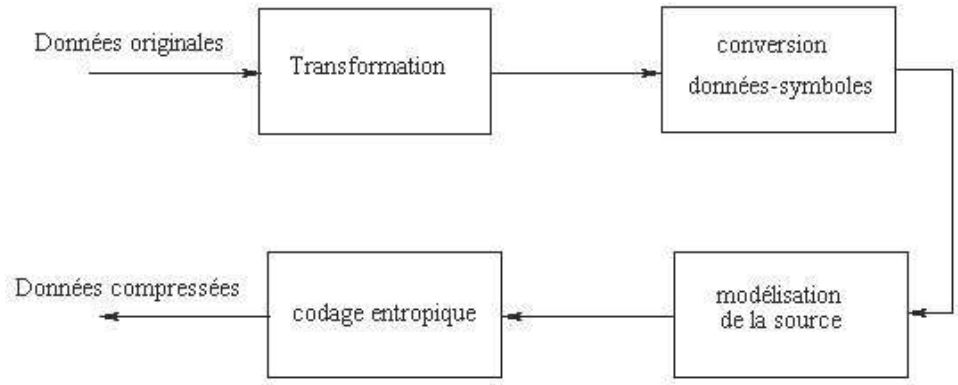


FIGURE 2.3 – Structure d'un système de compression sans perte.

La plupart des systèmes de compression sans perte de données opèrent comme présenté figure 2.3. Ainsi, le rôle du codeur est de générer, à partir d'une séquence de données, un flux compressé. Quant au décodeur, il réalise l'opération inverse et permet de reconstruire les données originales à partir du flux compressé. D'après le diagramme présenté figure 2.3, un algorithme de codage sans perte est généralement décomposé en quatre étapes.

– La première étape consiste à appliquer une transformation réversible sur les données à coder. L'objectif de cette transformation est de représenter les données originales d'une manière plus adaptée pour la compression. Généralement, une transformation vise à mettre en évidence des redondances cachées, modifier la distribution des données originales et, parfois, grouper les informations les plus importantes en un nombre réduit de symboles. Les transformations les plus fréquemment utilisées dans la compression sans perte sont basées sur les prédicteurs et la décomposition en sous bandes, comme la transformation en ondelettes. Il y a également d'autres transformations comme par exemple Burrows-Wheeler [Burrows 1994], mais elle est moins efficace et nécessite un coût de calcul très élevé. Il est à noter que certaines transformations seront plus efficaces une fois appliquées dans des applications spécifiques. Par exemple, les transformations qui sont basées sur les prédicteurs seront plus efficaces si elles sont utilisées pour la compression sans perte d'images. En effet, comme nous l'avons mentionné antérieurement, les pixels voisins d'une image sont fortement corrélés. Donc, ces pixels se ressemblent et, par conséquent, la variation entre deux pixels voisins est petite. Dans ce cas, si nous appliquons un simple pré-

dicteur qui a pour objectif, plutôt que de coder les pixels, de coder la variation entre deux pixels consécutifs, alors nous modifions la distribution des données originales. Pour des pixels fortement corrélés, les différences seront de faible magnitude et centrées sur zéro, ce qui conduit à la construction des codes efficaces. Il est à rappeler que dans certaines applications, cette première étape est ignorée et par conséquent le codage sera effectué directement sur les données originales, ce qui réduit les temps de traitement.

- La deuxième étape consiste à convertir les données issues de la transformation en des symboles en vue de les coder d'une manière plus efficace. La conversion des données en des symboles peut se faire, par exemple, avec la méthode Run Length Encoding (RLE) [Golomb 1966], ou bien à travers le partitionnement en représentant un ensemble de données voisines par un seul symbole et parfois en utilisant les méthodes basées sur la construction de dictionnaire. Le choix de la méthode de conversion est très important, et dépend essentiellement du type de données à coder. Par exemple, la méthode RLE est souvent utilisée dans la conversion des pixels d'une image binaire en des symboles représentant les longueurs des répétitions des pixels dans l'image à coder. Il est à rappeler que RLE est une technique avec mémoire, ce qui signifie qu'elle code les valeurs d'entrée en prenant en compte les valeurs précédentes. Le choix de la méthode RLE dépend de la nature des images, puisque généralement la probabilité que deux pixels voisins soient de même valeurs est très élevée. Ainsi, au lieu de coder les valeurs des pixels, nous codons les répétitions.
- La troisième étape, qui est la modélisation de la source, vise à modéliser la fonction de distribution de probabilités des symboles à coder. Comme nous le savons, une technique de compression efficace représente les symboles les plus probables avec des codes courts, et les symboles les moins probables avec des codes plus longs. Il semble intéressant de modéliser la fonction de probabilités qui s'adapte au mieux avec ce qui est observé dans la séquence des symboles à coder. Par conséquence, cette modélisation s'adapte à la statistique de la source et elle permet de générer des codes de longueurs optimales.
- Enfin, la plupart des systèmes de compression utilisent dans la dernière étape un codage à longueur variable, appelé également codage statistique ou bien codage entropique. Ce dernier permet de représenter les symboles à coder avec la forme la plus compacte possible. Les codeurs entropiques les plus utilisés et les plus efficaces sont le codage de Huffman et le CA et ils sont utilisés dans les dernières normes et

standards de compression d'images. Ce type de codeur vise à minimiser le nombre moyen de bits par symbole tout en se basant sur l'étape précédente qui fournit une estimation de la densité de probabilités des symboles à coder. Notons qu'un codeur entropique peut être soit statique soit adaptatif. Dans le premier cas, il utilise une table de probabilités (ou bien table de fréquences) fixe. Par contre, dans le deuxième cas, le codage entropique utilise une table de probabilités adaptative qui tente d'estimer au mieux la probabilité du symbole à coder. Le codage entropique sera discuté en détail section 2.5.

2.5 Codage entropique

Il existe plusieurs codeurs entropiques, les plus connus et les plus performants étant le codage de Shannon-Fano, le codage de Huffman et le CA détaillés sections 2.5.1, 2.5.2 et 2.5.3 respectivement.

2.5.1 Le codage de Shannon-Fano

Le codage de Shannon-Fano [Fano 1949], proposé par Claude Shannon et Robert Fano, est le premier algorithme développé dans le but de construire la meilleure technique de codage à longueur variable. Ainsi, soit une suite $\{x_1, \cdots, x_m\}$ de m symboles produite à partir d'une source S qui prend ses valeurs dans un alphabet A composé par k symboles s_i $(1 \leq i \leq k)$, le codage de Shannon-Fano consiste tout d'abord à calculer la distribution de probabilités $P = \{p_1, \cdots, p_k\}$ des symboles à coder, avec $p_i = P(S = s_i)$. Ensuite, cet algorithme consiste à trier les symboles s_i de la source S par ordre décroissant de probabilités, $p_1 \geq p_2 \geq \cdots \geq p_k$. L'ensemble des symboles sera alors divisé en deux parties de sorte que la somme des probabilités des symboles appartenant à la première partie soit la plus proche possible de la somme des probabilités des symboles de la deuxième partie. La valeur 0 est ajoutée au début de tous les codes des symboles de la première partie, alors que la valeur 1 est ajoutée au début de tous les codes des symboles de la seconde partie. Chaque partie est ensuite divisée récursivement en deux sous parties, en gardant les sommes des probabilités les plus proches possible, afin de trouver les bits de tous les codes. La même procédure se répète jusqu'à ce que nous atteignions des parties n'ayant qu'un seul symbole.

Cette technique fournit un code uniquement décodable et produit les meilleurs résul-

tats lorsque les probabilités sont des puissances de 2. Il est à noter également que le codage de Shannon-Fano présente une incertitude dans la procédure de découpage de l'ensemble des symboles en deux parties, sachant qu'il faut effectuer le découpage qui minimise la longueur moyenne des codes. Bien que cet algorithme soit facile à implémenter, il est toutefois moins efficace en moyenne que l'algorithme de Huffman.

2.5.2 Le codage de Huffman

Le codage de Huffman [Huffman 1952, Cormack 1984] a été proposé par D. Huffman en 1952. C'est une technique de codage à longueur variable implémentée suivant une structure d'arbre qui permet de construire un code uniquement décodable, à partir d'une distribution arbitraire, dont la longueur moyenne s'approche de l'entropie de la source. Ainsi, soit une source S qui prend ses valeurs dans un alphabet A composé par k symboles s_i $(1 \leq i \leq k)$. Le codage de Huffman est basé sur les conditions d'optimalités suivantes pour un code préfix :

1) si $p_i > p_j$, en d'autres termes, si le symbole s_i est plus probable que le symbole s_j avec $i \neq j$, alors $l_i \leq l_j$ avec l_i et l_j les longueurs des codes associés respectivement aux symboles s_i et s_j.

2) si les symboles sont triés par ordre décroissant de probabilités, alors les deux symboles ayant les plus faibles probabilités seront codés avec deux codes de même taille et qui différents uniquement au niveau du dernier bit.

Le codage de Huffman, comme pour le cas du codage de Shannon-Fano, consiste tout d'abord à calculer la distribution de probabilités $P = \{p_1, \cdots, p_k\}$ des symboles à coder, avec $p_i = P(S = s_i)$. Ces symboles sont ensuite triés par ordre décroissant de probabilités, $p_1 \geq p_2 \geq \cdots \geq p_k$, et ils correspondent aux nœuds terminaux d'un arbre binaire. Soit E l'ensemble des symboles ordonnés de la source S. Le principe de l'algorithme de Huffman consiste à regrouper les deux symboles de probabilités les plus faibles pour construire un nouveau symbole dont la probabilité est la somme des probabilités de ces deux symboles. Pour la construction de l'arbre, un nouveau nœud est ajouté pour placer le nouveau symbole avec sa probabilité. Ce nouveau nœud est le nœud père des deux autres nœuds, et par conséquent, deux nouvelles branches seront ajoutées à l'arbre. Notons qu'à cette étape, nous associons la valeur 1 pour une branche et la valeur 0 pour l'autre branche. La création d'un nouveau nœud engendre également la mise à jour de l'ensemble des symboles de E par la suppression des symboles les moins probables et de les remplacer par le nouveau

symbole créé. Ces opérations sont itérées plusieurs fois et à chaque itération le nombre de symboles diminue jusqu'à avoir un seul symbole dans E qui correspond au nœud racine de l'arbre final. Enfin, le code associé à chaque symbole s_i de S $(1 \leq i \leq k)$, peut être calculé en parcourant l'arbre du nœud racine vers le nœud terminal qui correspond à s_i et en concaténant les valeurs des branches visitées.

Pour permettre au décodeur de reconnaître les symboles, il est nécessaire de lui transmettre le dictionnaire symboles-code obtenu par le codeur. L'algorithme de Huffman produit des codes de longeurs entières ayant une longueur moyenne qui est très proche de l'entropie de la source S.

2.5.3 Le codage arithmétique

Le CA [Langdon 1984, Witten 1987, Howard 1994, Moffat 1998] est une technique de compression très puissante qui est largement utilisée dans les dernières normes de compression d'images et de vidéos tel que JBIG, JBIG2, JPEG2000 et H.264/AVC. Le principe du CA a été initialement fondé par Peter Elias en 1960 et a été décrit pour la première fois par Abramson [Abramson 1963]. Le problème rencontré était la limite de la précision des opérations arithmétiques nécessaires pour coder des messages très longs. Une première implémentation praticable a été proposée en 1976 par Pasco [Pasco] et Rissanen [Rissanen 1976]. Toutefois, l'implémentation proposée n'était pas très efficace et c'est en 1979 et en 1980 que Rubin [Rubin 1979], Guazzo [Guazzo 1979], Rissanen [Rissanen 1979] et Langdon [Langdon 1979] ont publié des versions en précision finie du CA. D'autres implémentations et d'autres études de complexité, de rapidité et d'efficacité ont guidé plusieurs recherches [Howard 1992, Moffat 1998, Witten 1987].

Actuellement, les implémentations retenues et utilisées pour le CA sont basées sur l'utilisation d'une implémentation entière à précision finie où toutes les variables utilisées sont des entiers. Ce type d'implémentation présente l'avantage d'être rapide, facile à mettre en œuvre et fournie des débits binaires très proches de l'entropie de la source à coder. Il existe également dans la littérature plusieurs autres techniques de compression, voir même de cryptage, basées sur le CA.

Le CA traite l'ensemble d'une séquence de symboles comme une seule entité, ce qui lui permet de réussir à sortir de la contrainte imposée par les techniques de codage à longueur variable qui attribuent un mot de code de longueur entière aux symboles d'une source donnée. Ainsi, le CA permet de coder une séquence avec un seul mot de code, et

par conséquent il garanti le codage de chaque symbole avec un nombre non entier de bits. En effet, pour bien comprendre l'utilité de ce principe, nous supposons avoir une séquence qui est composée par 100 bits, dont 99 bits sont égaux à 0 et 1 bit qui est égal à 1. Ainsi, la probabilité du symbole 0 est égale à $p(0) = 0.99$. En utilisant le codage de Huffman, par exemple, cette séquence est codée avec 100 bits, puisque chaque symbole est représenté au minimum avec 1 bit. Toutefois, le CA permet de coder la séquence avec uniquement 10 bits. Ce dernier résultat est calculé selon l'équation 4.12 qui sera détaillée section 4.2.4. L'exemple précédent montre l'apport du CA par rapport au codage de Huffman. Cet apport devient de plus en plus important quand la distribution des symboles à coder contient des probabilités importantes.

Le principe du CA consiste à représenter une séquence de symboles par un intervalle de nombres réels $I = [L, H)$ compris entre 0 et 1. Il est à signaler que toute valeur appartenant à ce dernier intervalle représentera d'une manière unique la séquence à coder. A mesure que la séquence s'allonge, l'intervalle requis pour représenter la séquence diminue, et le nombre de bits qui servent à décrire cet intervalle s'accroît. Les symboles successifs d'une séquence réduisent cet intervalle en concordance avec la probabilité d'apparition du symbole. Finalement, les données comprimées consistent en la partie fractionnaire du plus court nombre binaire qui se trouve dans l'intervalle final. Il est à noter qu'a la fin du processus de codage, le CA génère un fichier composé par un en-tête suivi de la représentation binaire du code choisi. De plus, l'en-tête peut contenir soit la table des probabilités, soit la table des fréquences de tous les symboles.

Soit une suite $X = \{x_1, \cdots, x_m\}$ de m symboles produite à partir d'une source S qui prend ses valeurs dans un alphabet A composé par k symboles s_i $(1 \leq i \leq k)$, l'algorithme de codage ainsi que celui de décodage opèrent respectivement selon l'algorithme 1 et l'algorithme 2.

D'après l'algorithme 1, le CA commence tout d'abord à calculer les probabilités p_i des symboles à coder s_i (ligne 2). Ensuite, il associe à chaque symbole un sous intervalle, noté par $[Ls_i, Hs_i)$, de l'intervalle $[0, 1)$ dont la longueur est proportionnelle à sa probabilité. Ainsi, nous avons $Hs_i - Ls_i = p_i$. La limite inférieure du premier symbole est égale à 0, et la limite supérieure du dernier symbole est égale à 1. Il est à préciser que pour tous les autres symboles s_i, les limites inférieures Ls_i et supérieures Hs_i sont calculées par :

$$\begin{array}{rcl} Ls_i & = & \sum_{j=1}^{i-1} p_j \\ Hs_i & = & Ls_i + p_i \end{array} . \tag{2.15}$$

Durant le processus de codage, l'intervalle courant, obtenu par le CA, est noté par $[L_i, H_i)$

Algorithme 1 Algorithme de codage.

1: **pour** les k symboles s_i de l'alphabet $\mathcal{A}$ **faire**
2: **calculer** la probabilité p_i du symbole s_i
3: Associer à chaque symbole s_i un sous intervalle $[Ls_i, Hs_i)$ proportionnel à sa probabilité, avec $Hs_i - Ls_i = p_i$
4: **si** $i = 1$ **alors**
5: $Ls_i \leftarrow 0$
6: **sinon**
7: $Ls_i \leftarrow Hs_{i-1}$
8: $Hs_i \leftarrow Ls_i + p_i$
9: $L_0 \leftarrow 0$ et $H_0 \leftarrow 1$
10: **pour** les m symboles x_i de la séquence X **faire**
11: x_i correspond à un symbole s_i de $\mathcal{A}$
12: $L_i \leftarrow L_{i-1} + (H_{i-1} - L_{i-1})Ls_i$
13: $H_i \leftarrow L_{i-1} + (H_{i-1} - L_{i-1})Hs_i$
14: représente X par n'importe quelle valeur de l'intervalle $[L_m, H_m)$ (par exemple $code(X) = \frac{L_m + H_m}{2}$)

et est calculé avec les lignes $10 - 13$ de l'algorithme 1. Finalement, le CA représente la séquence d'entrée X, composée par m symboles, avec n'importe quelle valeur de l'intervalle $[L_m, H_m)$, il est possible de choisir par exemple la valeur $\frac{L_m + H_m}{2}$.

L'algorithme 2 assure le décodage en commencent avec le code choisi à la fin du processus de codage. Les instructions des lignes $4 - 7$ se répètent m fois pour récupérer la séquence d'entrée X. A chaque itération, le CA cherche le symbole s_i qui vérifie $Ls_i \leq code(X) < Hs_i$. Après l'identification du symbole correct, la valeur du $code(X)$ est mise à jour selon le principe présenté dans la ligne 6. Les processus de codage et de décodage décrits précédemment sont impraticables, vu qu'ils nécessitent une précision infinie pour représenter les deux limites L et H. Il est à noter qu'il existe des implémentations du CA qui sont basées sur l'utilisation des nombres flottants à précision finie. Ce type d'implémentation est déconseillé en pratique puisqu'il nécessite des temps de calcul très élevés, c'est pourquoi la plupart des implémentations utilisées par les dernières normes et standards de compression sont basées sur des entiers, ce qui garanti une rapidité d'exécution et une facilité de traitement des problèmes de débordement.

La première étape du CA est l'estimation ou le calcul des probabilités des symboles à

Algorithme 2 Algorithme de décodage.

1: commencer par le code trouvé $code(X)$
2: $i \leftarrow 1$
3: **tant que** $i \leq m$ **faire**
4: chercher le symbole appartenant à l'intervalle $[Ls_i, Hs_i)$ sachant que $Ls_i \leq code(X) < Hs_i$
5: sortir le symbole s_i
6: $code(X) \leftarrow \frac{code(X)-Ls_i}{Hs_i-Ls_i}$
7: $i \leftarrow i + 1$

coder. Pour calculer les probabilités exactes des symboles il faut disposer de la totalité du fichier à compresser. Ainsi, une première passe est nécessaire pour calculer les probabilités. Il faut ensuite un deuxième passe pour réaliser l'opération de compression. C'est à dire, fournir la table des probabilités préalablement calculée par la première passe au CA qui à son tour génère le flux de données compressées. Néanmoins, si nous disposons d'un modèle statistique qui garanti une bonne approximation des probabilités, alors la première passe est omise et par conséquent il devient possible de réaliser un CA adaptatif (CAA). Le codage en une seule passe ne signifie pas qu'il est plus rapide qu'un codage en deux passes, puisque l'estimation des probabilités est une tâche très complexe qui se base dans la plupart des cas sur des calculs de contexte et des mises à jours des probabilités à chaque nouveau symbole codé/décodé.

2.6 Les standards de compression sans perte d'images

La compression sans perte d'images numériques est toujours possible grâce à la redondance qui existe dans ces images. Par exemple, dans une image, il existe souvent une forte corrélation entre des pixels voisins. Les techniques de compression exploitent cette dernière caractéristique afin de réduire le nombre de bits nécessaires pour la représentation d'une image. Actuellement, il existe plusieurs standards de compression sans perte d'images. La plupart de ces standards ont été développés par l'ISO (the International Standard Organisation), l'IEC (the International Electrotechnical Commission) et l'ITU (the International Telecommunication Union). Ces standards peuvent être classés en deux catégories : les standards de compression d'images en niveaux de gris et les standards de compression d'images binaires. Concernant les standards de compression d'images à tons

continues, nous citons essentiellement Lossless-JPEG, JPEG-LS et JPEG2000. Quand aux standards de compression d'images binaires, l'ITU et l'ISO/IEC ont proposé JBIG et JBIG2. Cette section présente un rapide aperçu de ces standards.

2.6.1 Lossless-JPEG

Lossless-JPEG est un standard de compression proposé par le comité Joint Photographic Experts Group pour le codage sans perte d'images en niveaux de gris. Ce standard utilise un schéma de codage composé par un codage prédictif suivi d'un codeur entropique. Le codeur entropique utilisé est un CAA, réalisant la compression sans perte d'une image erreur notée I_e. Cette dernière est obtenue de la manière suivante :

$$I_e(x,y) = I(x,y) - \hat{I}(x,y), \tag{2.16}$$

avec $I(x,y)$ est égale à la valeur du pixel de coordonnées (x,y) dans l'image originale I, et $\hat{I}(x,y)$ représente la valeur prédite du même pixel. Notons que pour prédire la valeur d'un pixel, Lossless-JPEG opte pour les 8 prédicteurs suivants :

0) $\hat{I}(x,y) = 0$

1) $\hat{I}(x,y) = I(x-1,y)$

2) $\hat{I}(x,y) = I(x,y-1)$

3) $\hat{I}(x,y) = I(x-1,y-1)$

4) $\hat{I}(x,y) = I(x-1,y) + I(x,y-1) - I(x-1,y-1)$

5) $\hat{I}(x,y) = I(x-1,y) + \frac{1}{2}(I(x,y-1) - I(x-1,y-1))$

6) $\hat{I}(x,y) = I(x,y-1) + \frac{1}{2}(I(x-1,y) - I(x-1,y-1))$

7) $\hat{I}(x,y) = \frac{1}{2}(I(x-1,y) + I(x,y-1))$

Sachant que les images, selon leurs différentes structures, favorisent l'utilisation de certains prédicteurs, la compression est appliquée pour chaque type de prédicteur et celui qui fourni le taux de compression le plus élevé, de manière globale, sera retenu.

2.6.2 Le codeur JPEG-LS

Le codeur JPEG-LS est le dernier standard de compression sans perte d'images en niveaux de gris du comité JPEG. Ce standard est conçu pour réaliser une compression sans perte d'images. Toutefois, il propose un deuxième mode de codage conduisant à une

compression avec perte noté **'near-lossless compression'**. JPEG-LS utilise le même schéma de codage qui a été proposé par son prédécesseur Lossless-JPEG, toutefois, il est plus efficace en terme de taux de compression. En effet, ce standard utilise dans l'étape de prédiction un algorithme ayant une faible complexité de traitement noté par LOw COmplexity LOssless COmpression for Images (LOCO-I) [Weinberger 1996, Weinberger 2000]. Cet algorithme détecte efficacement les contours horizontaux et verticaux suite à l'examen des pixels voisins d'un pixel $I(x,y)$ grâce à un prédicteur simple et rapide appelé Median Edge Detection (MED). Ce dernier opère selon l'algorithme 3. Pour chaque pixel $I(x,y)$,

Algorithme 3 l'algorithme du MED.

1: **si** $I(x-1,y-1) > max(I(x-1,y), I(x,y-1))$ **alors**
2: $\quad \hat{I}(x,y) \leftarrow min(I(x-1,y), I(x,y-1))$
3: **sinon si** $I(x-1,y-1) < min(I(x-1,y), I(x,y-1))$ **alors**
4: $\quad \hat{I}(x,y) \leftarrow max(I(x-1,y), I(x,y-1))$
5: **sinon**
6: $\quad \hat{I}(x,y) \leftarrow I(x-1,y) + I(x,y-1) - I(x-1,y-1)$

la valeur médiane de ses voisins est calculée avec l'algorithme 3, celle ci représente la valeur prédite notée par $\hat{I}(i,j)$. Cette dernière valeur n'est pas utilisée directement pour calculer le résidu à coder, c'est grâce à un contexte que la valeur prédite est corrigée afin d'être utilisée pour calculer l'image différence. Notons que dans JPEG-LS, un contexte représente la variation locale dans les valeurs des pixels. Ainsi, l'algorithme LOCO-I opte pour le calcul des résidus D_1, D_2 et D_3 afin de former un vecteur de contexte noté par $C = (C_1, C_2, C_3)$ à trois composantes de la manière suivante :

$$\begin{aligned} D_1 &= I(x-1,y+1) - I(x-1,y) \\ D_2 &= I(x-1,y) - I(x-1,y-1) \\ D_3 &= I(x-1,y-1) - I(x,y-1) \\ C_i &= f(D_i) \; pour \; i \; = \; 1 \; , \; 2 \; , \; 3. \end{aligned} \tag{2.17}$$

avec $f()$ une fonction définie par intervalles qui prend ses valeurs dans $[-T,T]$ et T est un coefficient positif défini par l'utilisateur. Notons que JPEG-LS propose d'utiliser la valeur $T = 4$. Enfin, les nouvelles erreurs, calculées à partir des vecteurs de contextes, sont modélisées par une distribution géométrique et par conséquent elles sont codées efficacement en utilisant un codage de Golomb.

2.6.3 Le codeur JPEG2000

Le codeur JPEG2000 est un standard de compression d'images développé récemment par le comité JPEG. L'objectif principal de ce standard est de répondre aux besoins des nouvelles applications à savoir la progressivité des données compressées, l'utilisation du même algorithme pour la compression avec et sans perte, le codage par région d'intérêt, la capacité d'être appliquée sur des images en niveaux de gris ou bien des images couleurs, et beaucoup d'autres nouvelles fonctionnalités. JPEG2000 est basé sur l'utilisation des ondelettes pour la décorrélation du signal image, suivi d'un codage par plan de bits. Il est à noter que JPEG2000 propose d'utiliser deux types d'ondelettes, et de les appliquer directement aux pixels des images. Ces ondelettes sont l'ondelette 9/7 de Daubechies qui réalise une transformation irréversible et qui est utilisée dans un schéma de compression avec pertes, et l'ondelette 5/3 qui est réversible et utilisée dans un schéma de compression sans perte. Les coefficients d'ondelettes sont par la suite codés plan de bits par plan de bits grâce à l'algorithme EBCOT. Ce dernier utilise un CA binaire adaptatif et avec contexte pour générer le flux de bits final.

2.6.4 Le codeur JBIG

Le codeur JBIG est un standard de compression sans perte d'images binaires proposé par Joint Bi-level Image experts Group (ITU-T Recommendation T.82, 1993) [JBI 1993, Salomon 2007]. Considérons une image binaire obtenue par la numérisation d'une page de texte. Cette image binaire contient généralement plusieurs zones uniformes, et l'intensité de chaque pixel est représentée par un seul bit et elle peut avoir deux valeurs possibles qui sont 0 pour la couleur blanche et 1 pour la couleur noire. Ainsi, pour une zone qui est presque blanche, la probabilité d'avoir 0 est proche de 1. Alors que, dans d'autres zones de l'image, la probabilité d'avoir 1 sera très élevée. Ainsi, JBIG propose d'utiliser un CA binaire (CAB) pour chaque zone (ou bien région), sachant que chacun de ces codeurs utilise sa propre table de probabilités. JBIG propose donc d'utiliser plusieurs tables de probabilités, et chaque table contient un couple de valeurs $(p_i(0), p_i(1))$ qui représentent respectivement la probabilité du symbole 0 et la probabilité du symbole 1. Pour chaque nouveau symbole, un prédicteur choisi la table de probabilités, et par conséquent le CA à utiliser pour son codage. Le prédicteur utilise les symboles voisins du symbole à coder afin de calculer un contexte qui sera utilisé par la suite dans le choix de la table de probabilités adéquate. Il est à noter que le CAB utilisé est appelé

MQ-Coder [Taubman 2002, Salomon 2007]. Ce dernier est une version modifiée du CAB adaptatif appelé Q-Coder [Pennebaker 1988, Mitchell 1988], qui est lui même une extension d'un autre CA appelé skew coder [Langdon 1983]. Bien que JBIG ait été conçu pour compresser sans perte des images binaires, il peut être utilisé pour coder des images en niveaux de gris en considérant l'image comme une superposition de plusieurs plans de bits, et par conséquent JBIG peut opérer plan de bits par plan de bits.

2.6.5 Le codeur JBIG2

Le comité JBIG a proposé plus tard un nouveau standard nommé JBIG2 (ITU-T Recommendation T.88, 2000). Ce dernier propose une nouvelle technique de compression sans perte plus performante que son prédécesseur. De plus, il rend possible la compression avec perte d'images binaires. Par comparaison avec JBIG, JBIG2 propose de partitionner une image binaire en 3 régions : région texte, région *halftone* et région générique. Grâce à ce partitionnement en région, JBIG2 propose deux modes de compression progressive. Le premier mode est la compression progressive par qualité, alors que le deuxième mode est la compression progressive par le contenu. Dans ce cas un texte peut, par exemple, être décodé avant les autres parties d'une image. Le lecteur pourra trouver plus d'informations et plus de détails sur le standard JBIG2 dans [JBI 2000].

2.7 Conclusion

Nous avons abordé dans ce chapitre un état de l'art sur les différentes méthodes de compression avec et sans pertes. Nous avons porté un intérêt particulier aux techniques de compression sans perte et nous avons présenté également une rapide description des derniers standards de compression d'images sans perte. Ces derniers standards utilisent le CA pour générer les flux de données compressées, ceci grâce à ses performances en terme de compression et à sa capacité d'être utilisé en plusieurs modes : statique, adaptatif et avec contexte. Bien qu'il a été largement exploité et qu'il a fait l'objet de plusieurs techniques de compression, nous pensons que le CA peut être constamment amélioré, soit par la développement de nouveaux modèles statistiques, soit par la proposition de nouvelles méthodes de prétraitements sur l'ensemble des symboles à coder. Ainsi, le CA sera la technique de codage utilisée dans le reste de ce mémoire.

Chapitre 3

La cryptographie

Sommaire

3.1 Introduction

Le développement croissant des nouvelles technologies et l'utilisation excessive des nouveaux moyens de communication ont donné naissance à de nouveaux problèmes. Ces problèmes sont liés essentiellement à la sécurité des données échangées. La sécurité est assurée grâce à la cryptographie qui à pour objectif de permettre à deux entités communicantes d'échanger d'une manière efficace et sécurisée des données confidentielles. La cryptographie consiste à utiliser des procédés mathématiques pour transformer un texte en clair en un texte incompréhensible en utilisant une clé. Les algorithmes de chiffrements usuels sont classés souvent selon les types de clés : symétrique, asymétrique ou hybride. Cependant, il est également possible de les classer selon le mode d'emploie : par bloc ou par flot. Il est à rappeler que parmi les différents types de données échangées, l'image occupe une place très importante. Malgré leurs efficacités, les algorithmes de chiffrements usuels ne sont pas très appropriés pour le cryptage des images. De ce fait, de nouveaux schémas de chiffrement par le chaos ont été proposés. Ces schémas sont basés sur la génération des séquences chaotiques ayant des propriétés spectrales et statistiques très importantes en cryptographie. Ces séquences chaotiques sont utilisées pour générer, d'une manière déterministe, des clés dynamiques. Ainsi, pour pouvoir être utilisé en cryptographie, ces clés doivent satisfaire certaines exigences qui sont vérifiées grâce à des séries de tests statistiques.

Dans ce chapitre, nous évoquons, section 3.2, les notions de base de la cryptographie et nous rappelons plusieurs techniques de cryptages utilisées depuis l'antiquité en temps de guerre. Dans la section 3.3, nous présentons les différents types des algorithmes de chiffrements, ensuite, section 3.4, nous détaillons plusieurs attaques employées dans la cryptanalyse. Il est à rappeler que les images se distingues par certaines propriétés, comme la forte corrélation spatiale, ainsi, nous proposons, section 3.5, de présenter les procédés servant pour analyser les performances des algorithmes de chiffrements d'images. Il est à noter qu'il n'est pas très approprié d'utiliser les algorithmes de chiffrement standard, comme le DES ou bien le RSA, pour crypter des images. Cela nous a amené à présenter les derniers travaux, liés aux chiffrement d'images, basés sur la théorie du chaos section 3.6. Ainsi, cette dernière théorie a été récemment employée en cryptographie et plus particulièrement dans le chiffrement par flot à travers la génération dynamique et déterministe des flux de clés. Ces flux de clés sont obtenues grâce à l'utilisation des générateurs de nombres pseudo-aléatoires, section 3.7. Finalement, section 3.8 introduit les nouveaux systèmes de

cryptages qui sont capable de réaliser à la fois la compression et le cryptage.

3.2 Généralités sur la cryptographie

3.2.1 Historiques

La cryptographie [Uhl 2004, Douglas 2002, Hershey 2002, Ferguson 2010] est la science qui exploite les mathématiques pour proposer des algorithmes de chiffrement (ou cryptosystèmes) permettant de protéger de l'information, alors que la cryptanalyse est l'étude du niveau de sécurité des cryptosystèmes. L'historique de la cryptographie peut être trouvée dans de nombreuses sources [Menezes 2001, Douglas 2002, Uhl 2004]. Nous présentons dans cette section quelques indications historiques afin de montrer d'une part, l'ancienneté de la cryptographie, et d'autre part, les principes de la cryptographie classique qui est fondée sur les notions de substitution, de transposition et de l'utilisation des clés de chiffrement. La cryptographie est utilisée depuis des centaines d'années. La scytale, par exemple, est un dispositif employé comme technique de chiffrement par transposition qui a été utilisée 600 ans avant J.-C par des Grecs en temps de guerre. Son principe consiste à enrouler en spirale une lanière de cuir sur un cylindre de bois appelé scytale. Sur cette lanière est écrit longitudinalement le message à transmettre. Ce dernier est finalement enroulé pour devenir incompréhensible et il est envoyé au destinataire qui possède un cylindre de même diamètre que celui utilisé par l'émetteur, sur lequel il enroule la lanière afin de décrypter et lire le message. Ainsi, sans la disposition d'un bâton ayant le même diamètre que l'émetteur, le récepteur sera incapable de décrypter le message. Ceci fut une technique simple à employer mais pas très sûre.

Il est à noter qu'il existe plusieurs autres techniques de chiffrement par transposition qui consistent à changer ou permuter l'ordre d'apparition des caractères du message original tout en conservant les mêmes caratères. 50 ans avant J.-C, Jules César utilisait une autre technique de chiffrement, appelée chiffrement par substitution qui est utilisée pendant des centaines d'années, et dont le principe consiste à remplacer tout caractère du texte clair par un autre caractère dans le texte chiffré. Par exemple, l'idée proposée par Jules César consiste à remplacer chaque lettre A du message en clair par la lettre D, les B par des E, et ainsi de suite pour tout l'alphabet. Le système cryptographique proposé est appelé chiffrement de César et il est basé sur un décalage des lettres de l'alphabet de trois lettres. Le chiffrement de César est extrêmement vulnérable puisqu'il existe seulement 26

façons différentes de chiffrer un message. En plus, l'inconvénient majeur des techniques de chiffrement par substitution étaient la connaissance *a priori* des fréquences d'apparitions des lettres.

Autre que la substitution monoalphabétique que nous venons d'introduire avec le chiffrement de César, il apparaît également un peu plus tard un autre type de substitution appelé substitution polyalphabétique. Cette dernière présente l'avantage d'être plus robuste à une cryptanalyse statistique. En 1586, le diplomate Blaise de Vigenère proposa une nouvelle technique de substitution polyalphabétique et qui consiste à appliquer 26 chiffrements de César dans un ordre bien défini. Le principe de substitution a été utilisé également lors de la seconde guerre mondiale par les allemands, qui ont construit en 1918 une machine à crypter appelée Enigma. Son principe était de remplacer chaque lettre par une autre tout en changeant la règle de substitution d'une manière automatique ou manuelle.

Depuis 1970, et avec le développement des ordinateurs et des techniques de communications, la cryptologie est devenue un thème de recherche scientifique lié à beaucoup d'autres disciplines comme les mathématiques, la complexité et la théorie de l'information.

3.2.2 Définitions

Pour sécuriser une information, la cryptographie propose d'appliquer des techniques mathématiques sur cette information en vue de la garder secrète pour tout ceux qui ne sont pas autorisés à la connaitre. La cryptographie consiste en l'application des méthodes de chiffrement [1] et de déchiffrement [2] pour cacher et rendre secret l'information à stocker ou à transmettre vers un destinataire. Un algorithme cryptographique ou de chiffrement est appelé cryptosystème. Un cryptosystème utilise une clé K afin de crypter un texte clair [3] ou un message M. Le cryptage consiste donc à transformer le message M en un texte chiffrée C [4]. Ce dernier sera envoyé au destinataire à travers un canal de communication non sécurisé. Ce canal est souvent accessible par des ennemis, appelé également cryptanalystes ou pirates, qui récupèrent et étudient l'information cryptée afin d'en découvrir le secret. La clé K et le message crypté C seront utilisés par le récepteur afin de déchiffrer le message original M. L'émetteur et le récepteur peuvent utiliser la même

1. (en anglais encryption)
2. (en anglais decryption)
3. (en anglais plaintext)
4. (en anglais ciphertext)

clé, il s'agit dans ce cas de la cryptographie symétrique. Dans le cas contraire, c'est à dire si une paire de clés est utilisée (une clé pour le cryptage et une autre clé pour le décryptage), il s'agit alors de cryptographie asymétrique. Un système classique employé en cryptographie est un quintuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ caractérisé par :

- $\mathcal{M}$ est un ensemble fini de textes clairs possibles,
- $\mathcal{C}$ est un ensemble fini de textes chiffrés possibles,
- $\mathcal{K}$ est un ensemble fini de clés possibles,
- $\mathcal{E}$ est un ensemble fini de procédés de chiffrement, tel que $\mathcal{E} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$
- $\mathcal{D}$ est un ensemble fini d'algorithmes de déchiffrement, vérifiant $\mathcal{D} : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$

Ainsi, pour chaque clé $K \in \mathcal{K}$ et pour chaque texte clair $M \in \mathcal{M}$, il y a une fonction $E() \in \mathcal{E}$ et une fonction $D() \in \mathcal{D}$ telles que $D(K, E(K, M)) = M$. Kerckhoffs [Kerckhoffs 1883], en 1883, a résumé le principe de la cryptographie moderne par la distinction entre l'algorithme de cryptage et la clé : " La sécurité d'un système de chiffrement ne doit pas dépendre du secret de l'algorithme mais seulement du secret de la clé". Ainsi, l'ennemi a accès à l'algorithme, toutefois, la clé reste secrète et connue uniquement par les entités communicantes.

3.3 Introduction à la cryptographie standard

Dans le cadre de cette thèse, nous nous intéressons à l'étude des techniques et outils qui assurent la confidentialité des données. Ainsi, pour garder le secret d'une information confidentielle, nous utilisons des techniques de chiffrement qui ont pour but de transformer une donnée afin de la rendre incompréhensible par tous ceux qui ne sont pas autorisés à y accéder. Le chiffrement est généralement effectué par substitution ou bien par transposition et voir même par la combinaison de ces deux techniques. La substitution consiste à modifier les bits du texte clair par d'autres bits par l'intermédiaire d'opérations mathématiques complexes. Ce procédé permet donc d'ajouter de la confusion. Quant à la transposition, elle permet d'ajouter de la diffusion en réarrangeant les bits du texte clair afin d'éliminer la redondance présente dans ce dernier et d'éviter quelle se retrouve dans le texte chiffré. Les algorithmes utilisés en cryptographie peuvent être employés par bloc ou bien par flot et il est possible de les classer en trois types :

- Cryptographie symétrique,
- Cryptographie asymétrique,
- Cryptographie hybride.

Nous nous intéressons dans cette section à présenter ces différents types d'algorithmes cryptographiques.

3.3.1 La cryptographie symétrique

La cryptographie symétrique est appelée également cryptographie à clé secrète. Comme tout autre système cryptographique, la cryptographie symétrique est basée sur l'utilisation d'un algorithme de chiffrement $E()$, qui prend en entrée un message M et une clé K et qui génère le message chiffré correspondant C. Ce message chiffré, appelé également cryptogramme, sera utilisé avec la même clé K par un algorithme de déchiffrement $D()$ afin de restaurer le message M. Dans ce cas, l'émetteur et le récepteur doivent échanger au préalable la clé qui doit être gardée secrète. La figure 3.1 présente le schéma général d'un algorithme de chiffrement symétrique. Ce type de cryptage présente l'avantage d'être rapide, efficace et capable de chiffrer des grandes quantités de données. Toutefois, l'inconvénient du chiffrement à clé secrète est qu'il exige que la clé doit être préalablement échangée sur un canal sécurisé.

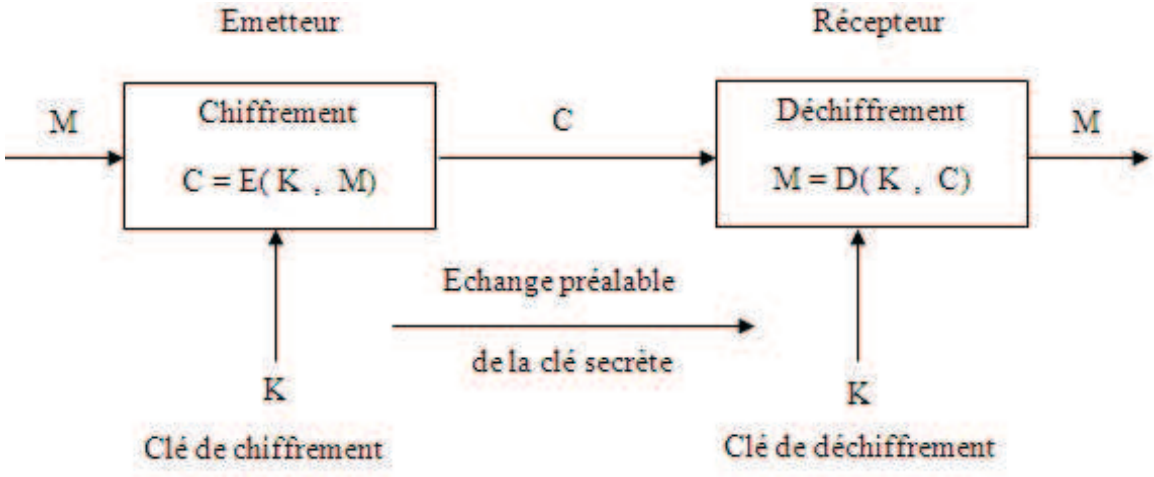


FIGURE 3.1 – Chiffrement symétrique.

Il est à noter qu'il existe deux types de chiffrement symétrique : le chiffrement à flot et chiffrement par blocs. La distinction entre ces deux types de chiffrement est parfois délicate. Ainsi, le chiffrement à flot est usuellement un algorithme de chiffrement qui opère sur des blocs de clair de taille relativement petite (typiquement un bit, un octet ou un mot) au moyen d'une transformation qui varie au cours du temps. Au contraire, un algorithme par blocs divise le texte clair en blocs de taille fixe et plus importante

(généralement 64, 128 ou 256 bits), et applique la même fonction aux différents blocs de clair [Menezes 2001]. Les algorithmes de chiffrement par blocs et à flot sont présentés respectivement section 3.3.1.1 et section 3.3.1.2.

3.3.1.1 Le chiffrement par blocs

Un algorithme de chiffrement par blocs chiffre un bloc à la fois avec la même clé. Le DES [5] et l'AES [6] sont les exemples les plus connus de chiffrement par blocs.

Le standard de chiffrement DES :

IBM a proposé en 1970 un nouveau système de chiffrement à clé secrète qui a été largement exploité dans le chiffrement des données et a été adopté, en 1976, par le gouvernement américain et le National Institute of Standards and Technology (NIST) comme un chiffrement standard officiel. Le DES [Schneier 1996, Douglas 2002, Martin 2004] est un algorithme de chiffrement par bloc qui opère sur des blocs de données de taille 64 bits en utilisant une clé de 64 bits. Le texte clair est initialement découpé en des blocs de 64 bits, ensuite le chiffrement est appliqué sur chacun de ces blocs. Le DES s'effectue avec 16 itérations enchaînant des opérations de transposition et de substitution. A chaque itération, le bloc est divisé en deux parties, une partie gauche L_i et une partie droite R_i de même longueur. Ces valeurs sont calculées par :

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \end{cases}, \tag{3.1}$$

avec $f()$ est une fonction particulière utilisé par DES et $K_{i=0}^{15}$, sont des clés calculées à partir de la clé K. La longueur de la clé proposée par DES devient de plus en plus facile à trouver en un temps raisonnable [Curtin 2005] par une recherche exhaustive. Cela est possible grâce à l'accroissement des performances des ordinateurs. Ainsi, l'année 1998, marque la naissance d'un nouveau standard de cryptage à clé secrète appelé AES (Advanced Encryption Standard), plus sécurisé et plus robuste à l'attaque par recherche exhaustive (notée également par attaque par force brute).

Le standard de chiffrement AES :

Le NIST a lancé en 1997 un appel d'offres pour remplacer le DES et fournir un algorithme de chiffrement symétrique et utilisable dans le monde entier. Le

5. (Data Encryption Standard)
6. (Advanced Encryption Standard)

AES [Douglas 2002] est un chiffrement par bloc, appelé également algorithme de Rijndael, inventé par Rijmen et Daemen. L'algorithme AES autorise un découpage en blocs de taille 125, 192 et 256 bits. Il existe également une amélioration au niveau de la taille des clés qui peuvent s'étendre sur 128, 192 ou 256 bits. Les deux nouvelles possibilités apportées par AES, possibilité de choisir la taille du bloc et possibilité de choisir la taille de la clé, ont garantit une flexibilité dans l'utilisation d'AES, et le choix de ces paramètres dépend du niveau de sécurité et de la vitesse de calcul désirée.

De plus, le chiffrement par blocs peut être employer selon plusieurs modes [Martin 2004]. Les modes les plus connus sont :

- **ECB :Electronic CodeBook**, notons par m_i le $i^{\text{ème}}$ bloc du texte clair M et c_i le bloc correspondant du texte chiffré C. Le mode ECB consiste à appliquer un algorithme de chiffrement $E()$, en prenant en entrée une clé K et un bloc m_i du texte clair, pour générer le bloc c_i du texte chiffré, $c_i = E(K, m_i)$, $i \geq 1$.
- **CBC : Cipher Bloc Chaining**, dans ce mode, le bloc c_{i-1} est utilisé pour calculer le $i^{\text{ème}}$ bloc du texte chiffré. Ainsi, avant d'utiliser l'algorithme de chiffrement $E()$, le mode **CBC** applique un "ou exclusif" entre le bloc m_i et le bloc chiffré précédent c_{i-1}, soit $c_i = E(K, c_{i-1} \oplus m_i)$. Le déchiffrement du bloc m_i est calculé par $m_i = c_{i-1} \oplus D(K, c_i)$.
- **CFB : Cipher FeedBack**, dans le mode CFB, le texte chiffré est obtenu en calculant $c_i = m_i \oplus z_i$, pour $i \geq 1$, avec $z_i = E(K, c_{i-1})$ et c_0 est un bloc initial ayant la même taille que les blocs m_i, $i \geq 1$.
- **OFB : Output FeedBack**, dans ce mode, une séquence de clé est générée pour être employer sur le texte clair par un "ou exclusif". Ainsi, le bloc m_i est chiffré en calculant $c_i = m_i \oplus z_i$, avec $z_i = E(K, z_{i-1})$, $i \geq 1$, et z_0 est un bloc initial de même taille que les blocs m_i, pour $i \geq 1$.

Pour plus de détails sur les modes de chiffrement par blocs, le lecteur pourra se référer à [Uhl 2004].

3.3.1.2 Le chiffrement par flot

Le chiffrement par flot [Menezes 2001] diffère du chiffrement par bloc, puisqu'il ne transforme pas des blocs de données en clair en des blocs de données chiffrés. Ainsi, le principe consiste à générer à partir d'une clé secrète K une suite d'octets ou de bits de

même taille que le texte clair, notée par k_i, $i \geq 1$. La suite de clé générée [7] est utilisée par un algorithme de cryptage $E()$ pour transformer le texte clair M en un texte chiffré C de la manière suivante, $c_i = E(k_i, m_i)$, avec m_i est le $i^{\text{ème}}$ élément (octet ou bit) du texte clair, c_i est le $i^{\text{ème}}$ élément du texte chiffré et k_i est la $i^{\text{ème}}$ clé générée. L'opération de décryptage ressemble à celle de cryptage puisqu'il suffit d'appliquer un algorithme de décryptage $D()$ sur le texte chiffré en utilisant la même suite de clés pour trouver le texte clair. La clé K est la clé secrète, dite statique, par contre la clé k_i dépend du temps et est nommée dynamique. Parmi les techniques de chiffrement par flot les plus utilisées, il y a l'algorithme $A5$ qui est utilisé dans les téléphones portables (GSM), ou bien pour crypter les signaux GPS [8]. Il existe également une autre technique de chiffrement par flot appelé RC4 utilisée dans les équipements des réseaux locaux sans fil et notamment dans le protocole SSL de Netscape. Le chiffrement par flot est très rapide et nécessite des ressources restreintes pour le mettre en œuvre.

Il existe différents types de chiffrement par flot et qui dépendent de la procédure adoptée dans la génération du flux de clés dynamiques. Nous citons essentiellement le chiffrement parfait, le chiffrement par flot synchrone et le chiffrement par flot asynchrone ou autosynchrone.

Le chiffrement parfait :

Le chiffrement parfait, dit également chiffrement de Verman [Vernam 1926, Kahn 1967], selon le nom de son inventeur G. Vernam en 1917, est une technique de chiffrement à clé secrète. Il s'agit d'un chiffrement à masque jetable (one-time pad) où la taille de la clé est identique à celle du message en clair. La clé est connue uniquement pour les participants à une communication, et une fois utilisée, elle devient non réutilisable et l'envoi d'un nouveau message exige l'utilisation d'une nouvelle clé. Le principe de Verman est simple et consiste à appliquer un ou exclusif $\oplus$ entre les bits du message M et ceux de la clé K pour calculer le cryptogramme C, avec $C = M \oplus K$. Pour restituer le message M, il suffit d'appliquer un "ou exclusif" entre les bits du cryptogramme C et de la clé K, $M = C \oplus K$. Le chiffrement de Verman est sûr, néanmoins il est difficile à le mettre en œuvre, ce qui a motivé les cryptographes à chercher d'autres méthodes de chiffrement.

Chiffrement par flot synchrone :

Dans le cas du chiffrement par flot synchrone, le flux de clés dynamiques est généré indépendamment du texte clair M et du texte chiffré C. Le chiffrement synchrone peut

7. (en anglais keystream)
8. Global positionning system

être décrit par l'équation :

$$\begin{cases} k_i = f_K(k_{i-1}) \\ c_i = E(k_i, m_i) \end{cases}, \tag{3.2}$$

avec $f_K()$ la fonction utilisée dans la génération du flux de clés dynamiques, K la clé secrète, m_i les symboles du texte clair, c_i les symboles du texte chiffré et $E()$ l'algorithme de cryptage. La clé K représente la valeur initiale du générateur $f_K()$. La procédure de décryptage est caractérisée par les mêmes éléments utilisés durant le cryptage et peut être décrite par l'équation :

$$\begin{cases} k_i = f_K(k_{i-1}) \\ m_i = D(k_i, c_i) \end{cases}, \tag{3.3}$$

avec $D()$ est l'algorithme de décryptage. Afin de récupérer le texte clair, l'émetteur et le récepteur doivent être synchronisés et la fonction $f_K()$ doit être initialisée avec la même clé pour les deux entités communicantes. Le schéma de chiffrement par flot synchrone est représenté figure 3.2.

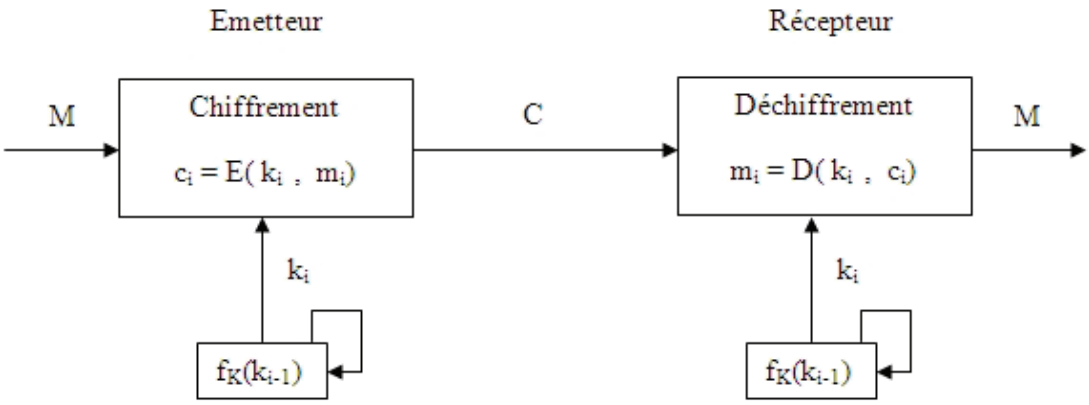


FIGURE 3.2 – Chiffrement par flot synchrone.

Chiffrement par flot asynchrone :

Le chiffrement par flot asynchrone [Menezes 2001, Masmoudi 2009] peut être décrit par l'équation :

$$\begin{cases} k_i = f_K(c_{i-1}, \cdots, c_{i-j}) \\ c_i = E(k_i, m_i) \end{cases}, \tag{3.4}$$

avec $f_K()$ est la fonction qui génère le flot de clés à partir de la clé K et d'un nombre fixe de bits issus du texte chiffré C. Il est à notons que $c_{-1}, \cdots, c_{-j}$ est un vecteur initial qui fait partie du secret de la communication. L'opération de décryptage utilise le même principe en remplaçant $E()$ par $D()$ dans l'équation 3.4. Le schéma de chiffrement par flot

asynchrone est présenté figure 3.3. Dans le cas du chiffrement par flot asynchrone, si un symbole est erroné, alors uniquement les j clés k_i suivantes seront erronées, par contre, les autres clés seront correctes.

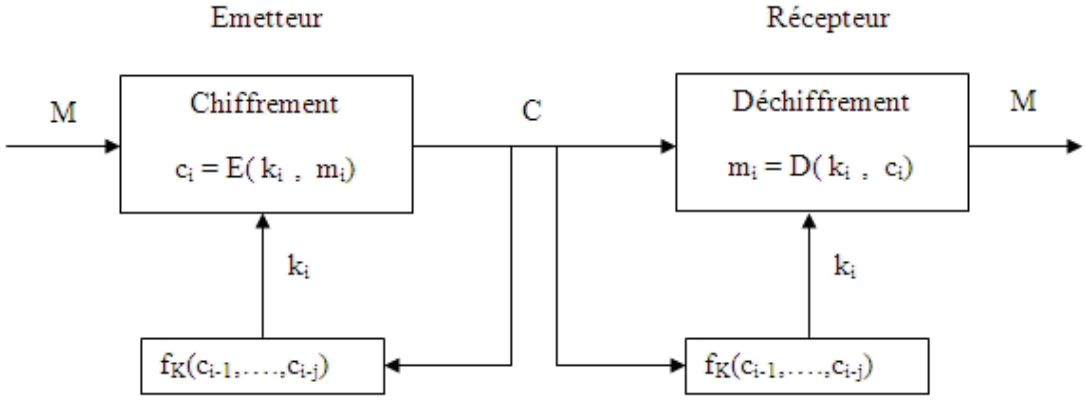


FIGURE 3.3 – Chiffrement par flot asynchrone.

3.3.2 Cryptographie asymétrique

Dans un schéma de chiffrement symétrique, la même clé K est utilisée dans la procédure de chiffrement et celle de déchiffrement. La clé doit être choisie secrètement par les entités communicantes et doit être échangée à travers un canal sûr avant la transmission du message chiffré, ce qui est difficile à réaliser en pratique. Le chiffrement asymétrique, appelé également chiffrement à clé publique, résout le problème d'échange de clé et propose d'utiliser une clé de déchiffrement différente de celle de chiffrement. Dans la paire de clés, il y a une qui est dite publique K_p et qui permet de chiffrer un message. Alors que la deuxième clé K_s, appelée clé privée, est utilisée pour le décryptage. La clé publique peut donc être échangée par l'intermédiaire d'un canal qui n'est pas forcément sécurisé. Nous dénoterons l'émetteur et le récepteur respectivement par A et B. Lorsque deux entités A et B veulent communiquer de façon sécurisée et en adoptant la technique de chiffrement à clé publique, l'émetteur A doit chiffrer le texte clair en utilisant la clé publique du récepteur B. Ce dernier utilise à son tour une clé privée pour déchiffrer le message chiffré. La figure 3.4 présente le schéma général d'un algorithme de chiffrement asymétrique. Le chiffrement à clé publique repose sur l'utilisation des fonctions à sens uniques et de fonctions à trappe. Soit une fonction $f : X \rightarrow Y$ avec X et Y des ensembles arbitraires. La fonction $f()$ est dite à sens unique s'il est facile à calculer $y = f(x), \forall x$, par contre il est difficile

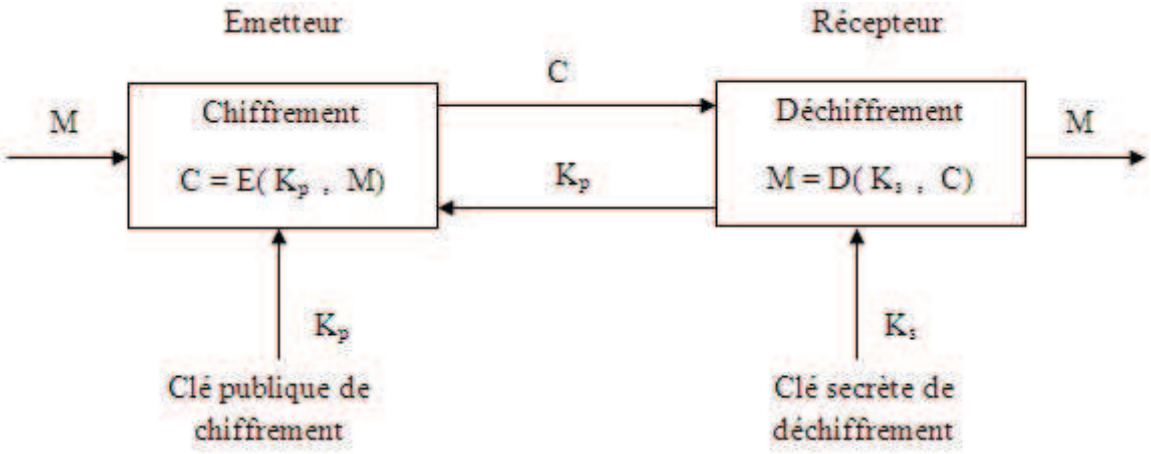


FIGURE 3.4 – Chiffrement asymétrique.

de trouver x en connaissant y, car le calcul de la fonction réciproque est trop coûteux voir même impossible en pratique. Une fonction $f()$ est dite à sens unique avec trappe, s'il y a une information supplémentaire secrète, appelée trappe, qui rend facile de calculer x à partir de y. Soient x la clé de déchiffrement, et y la clé publique de chiffrement. Si nous connaissons y, la clé publique, alors cela ne signifie pas qu'il est facile de trouver x, la clé privée, et ceci grâce à l'utilisation des fonctions à sens unique. La notion de la cryptographie à clé publique a été introduite en 1976 par Diffie et Hellman [Diffie 1976]. En 1977, apparait la première réalisation d'un cryptosystème à clé publique appelée RSA selon ses inventeurs Rive, Shamir et Adleman [Rivest 1983].

Le chiffrement RSA

Le RSA [Rivest 1983, Schneier 1996, Martin 2004] est l'algorithme le plus utilisé en chiffrement asymétrique et opére selon le principe suivant. Soient p et q deux nombres premiers très grands (de longueur au moins 1024 bits) et $n = pq$. Il faut choisir aléatoirement un entier b de sorte qu'il vérifie l'inégalité suivante : $b < n$, et qu'il soit premier avec $\phi(n) = (p-1)(q-1)$. De plus, il faut choisir d tel que $bd \equiv 1 \ mod \ \phi(n)$. Le texte chiffré est calculé par $C = M^b \ mod \ n$ qui est une fonction à sens unique, caractérisée par la difficulté de la factorisation de n. La trappe est la connaissance de p, q et d. Le déchiffrement de C se calcule par $M = C^d \ mod \ n$. Dans l'algorithme RSA, la clé publique est construite à partir du couple (b, n), alors que la clé privée est égale au couple (n, d). Le niveau de sécurité de l'algorithme RSA dépend de la taille de p et q. Ainsi, plus les nombres p et q sont grands, plus il est difficile de les retrouver à partir de leurs produits n.

3.3.3 Cryptographie hybride

Les techniques de chiffrements présentées, sections 3.3.1 et 3.3.2, sont très présentes dans les cryptosystèmes modernes. Le chiffrement symétrique est très rapide et utilise des clés de petites tailles. Par contre, le chiffrement asymétrique utilise des clés de tailles plus grandes mais nécessite un temps de calcul très élevé. La cryptographie hybride [Garfinkel 1996] se base sur l'exploitation et la combinaison des meilleurs fonctionnalités de la cryptographie symétrique et de la cryptographie asymétrique.

Le principe de la cryptographie hybride est décrit figure 3.5. Ainsi, l'émetteur choisi tout d'abord une clé K pour le cryptage du texte clair M en utilisant un chiffrement symétrique $E()$. La clé K est utilisée également par la fonction $D()$ pour assurer le décryptage. Cette clé secrète doit être échangée au préalable entre les entités communicantes. De ce fait et afin de s'assurer de la confidentialité de cette clé, elle doit être cryptée en utilisant la clé publique K_p du récepteur et le chiffrement asymétrique $E_a()$. Le récepteur qui dispose de la clé privée K_s, correspondante à sa clé publique K_p, commence à appliquer le déchiffrement $D_a()$ pour calculer la clé K. Cette clé sera utilisée finalement par la fonction de déchiffrement $D()$ pour récupèrer le texte clair. Parmi les algorithmes les plus utilisés en cryptographie hybride figure l'algorithme de chiffrement libre noté par PGP [9] et a été développé par Zimmermann [Zimmermann 1995a, Zimmermann 1995b]. Il est à noter que le chiffrement PGP se base sur l'algorithme DES, et pour l'échange de clés il utilise RSA.

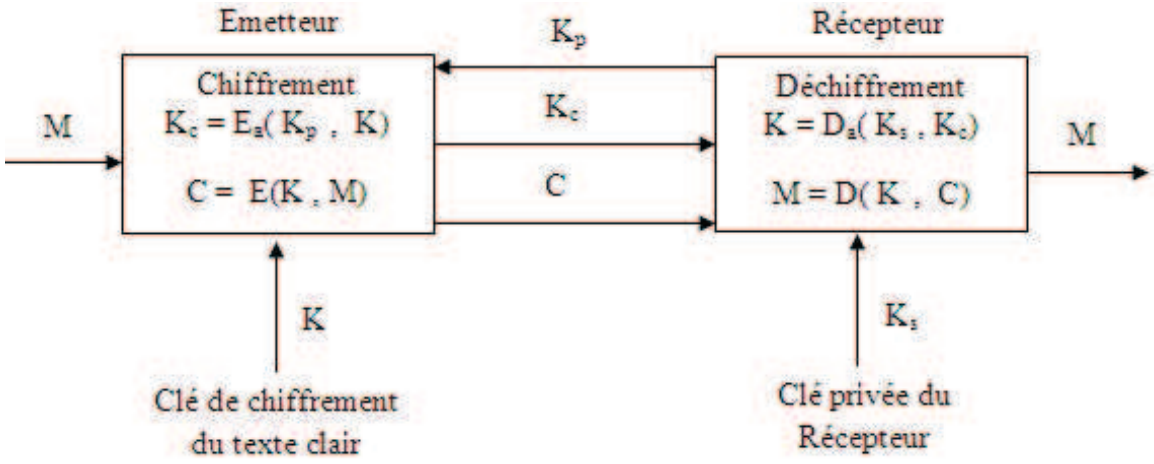


FIGURE 3.5 – Chiffrement hybride.

9. Pully Good Privacy

3.4 Cryptanalyse

Bien que la cryptographie est la discipline qui assure la sécurité des informations confidentielles, la cryptanalyse [Menezes 2001, Douglas 2002] est la discipline qui étudie et valide la robustesse des cryptosystèmes face aux attaques. Selon Kerckhoffs [Kerckhoffs 1883], la sécurité d'un cryptosystème dépend du secret de la clé et non pas du secret de l'algorithme de chiffrement. Ainsi, pour étudier la sécurité d'un cryptosystème il faut se baser sur le principe de Kerckhoff. De ce fait, le cryptanalyste doit être incapable de trouver la clé même s'il a accès à des textes clairs et les textes chiffrés correspondants. Ce dernier tente d'appliquer plusieurs attaques dont voici une liste non exhaustive :

- **Attaque par texte chiffré seul (Ciphertext only attack) :** dans ce type d'attaque, nous supposons que le cryptanalyste a uniquement l'accès au support de communication et par conséquent il ne peut connaître que les textes chiffrés sans la connaissance des textes clairs correspondant. Ce type d'attaque est le plus utilisé en pratique et n'importe quel cryptosystème vulnérable contre cette attaque, est considéré non sécurisée.
- **attaque par texte clair connu (Known plaintext attack) :** dans ce type d'attaque, nous supposons que l'ennemi dispose d'un certain nombre de textes chiffrés ainsi que les textes clairs correspondants, et il tente de trouver la clé utilisée pour crypter les messages.
- **attaque par texte clair choisi (Chosen plaintext attack) :** dans ce type d'attaque, l'ennemi peut choisir un texte clair et calculer le texte chiffré correspondant. Si un cryptosystème est vulnérable contre l'attaque par texte clair connu, alors il est nécessairement vulnérable contre l'attaque par texte clair choisi, cependant l'inverse n'est pas forcément vrai.
- **attaque par texte chiffré choisi (Chosen ciphertext attack) :** dans ce type d'attaque, nous supposons que l'ennemi a la possibilité de choisir un texte chiffré et d'accéder au texte clair correspondant. Cette attaque est très importante dans la cryptographie asymétrique puisque la clé publique de cryptage est connue par l'ennemi.

On peut trouver d'autres types d'attaques, comme l'attaque par texte clair choisi adaptative. Pour n'importe quel type d'attaque, l'objectif consiste toujours à trouver la clé ou bien à déchiffrer un texte chiffré qui correspond à un texte clair dont nous ne disposons

pas.

3.5 Analyse de performance d'un cryptosystème d'images

L'information image diffère des informations textuelles et possède des propriétés très intéressantes telles que sa grande capacité, sa forte redondance et la forte corrélation qui existe entre les pixels voisins. Selon ces propriétés, la sécurité d'un cryptosystème d'image est évaluée par l'analyse de l'espace de clés, la sensibilité à la clé, l'attaque différentielle et l'attaque statistique représentées respectivement sections 3.5.1, 3.5.2, 3.5.3 et 3.5.4.

3.5.1 Analyse de l'espace de clé

Pour un système de chiffrement d'images, l'espace de clés doit être suffisant large pour résister contre une attaque par force brute. Pour calculer la taille de l'espace de clé, il faut tout d'abord identifier l'ensemble des paramètres qui constitue la clé. Ensuite, il faut déterminer pour chaque paramètre son format (entier, réel, $\cdots$) ainsi que la plage des valeurs possibles. Finalement, le nombre de combinaisons possibles de ces paramètres est calculé en une puissance de 2 et il représente la taille en bits de l'espace de clés. Généralement, un espace de clé de taille 128 bits ou plus est suffisant pour résister à une attaque par force brute.

3.5.2 Analyse de la sensibilité à la clé

La sensibilité à la clé d'un cryptosystème d'images peut être étudiée de deux manières différentes. La première méthode consiste à chiffrer la même image avec deux clés légèrement différentes (par exemple, différente uniquement au niveau des bits de poids faible, dans le cas d'un chiffrement symétrique) et à comparer pixel par pixel les deux images chiffrées. Nous pouvons alors calculer le pourcentage de pixels différents et afficher l'image de différence entre les deux images cryptées. La deuxième méthode consiste à s'assurer qu'aucune partie de l'image originale ne peut être reconstruite en utilisant une clé de déchiffrement très proche de la clé de chiffrement.

3.5.3 Attaque par analyse différentielle

Cette attaque a été développée en 1990 par Eli Biham et Adi Shamir [Biham 1990]. Pour résister contre une attaque par analyse différentielle, il faut qu'une légère modification au niveau de l'image originale engendre une grande modification dans l'image cryptée. Ainsi, supposons qu'une image originale soit initialement cryptée en une image C_1. Ensuite, nous sélectionnons aléatoirement un pixel de l'image originale et nous le remplaçons par une valeur légèrement différente de sa valeur initiale. L'image modifiée est cryptée avec la même clé et l'image obtenue est notée C_2. Maintenant, il suffit de comparer les deux images C_1 et C_2 en utilisant les deux mesures notées par le NPCR et le UACI [Chen 2004, Huang 2009, Shatheesh 2010, Wan 2011] :

- **Number of pixels change rate (NPCR) :** cette mesure permet de calculer le pourcentage de pixels différents entre les images cryptées C_1 et C_2 en utilisant l'équation suivante :

$$NPCR = \frac{\sum_i D_i}{M \times N} \times 100\%, \tag{3.5}$$

avec $D(i,j) = 1$ si $C_1(i,j) \neq C_2(i,j)$ et $D(i,j) = 0$ sinon. Les valeurs M et N représentent la taille en pixels de l'image originale.

- **Unified average changing intensity (UACI) :** c'est une mesure de la moyenne des différences entre C_1 et C_2 :

$$UACI = \frac{1}{M \times N}\left(\sum_j \sum_i \frac{|C_1(i,j) - C_2(i,j)|}{L}\right) \times 100\%. \tag{3.6}$$

avec L le nombre de valeurs possibles pour les pixels.
Il est à noter que pour un efficace crypto-système, il faut avoir $NPCR > 99\%$ et $UACI$ proche de 33%.

3.5.4 Attaque par analyse statistique

Un cryptosystème idéal doit être robuste contre les attaques statistiques. Ainsi, pour évaluer la sécurité d'un cryptosystème d'images, les tests statistiques suivants doivent être toujours employés.

- **Histogramme :** l'histogramme d'une image chiffrée doit être uniforme.
- **Corrélation entre les pixels voisins :** l'équation 2.1 permet de calculer la corrélation entre les pixels voisins d'une image. Nous avons déjà signalé qu'une image est souvent caractérisée par une forte redondance, ce qui explique que la

corrélation entre les pixels voisins est très importante. Cependant, pour une image chiffrée la corrélation doit être très faible (proche de 0) pour éliminer toute sorte de dépendance linéaire entre les pixels voisins.

– **Entropie :** l'entropie d'un message M est définie par l'équation 2.9. En effet, une entropie est nulle si toutes les probabilités sauf une sont nulles. Autrement dit, l'entropie est nulle uniquement quand nous sommes certains des résultats. L'entropie est donc maximale quand toutes les probabilités sont égales, ce qui correspond à la situation la plus incertaine. Ainsi, plus l'entropie est petite, plus il existe un haut degré de prédictibilité et l'algorithme de chiffrement devient de plus en plus non sécurisé. Par exemple, pour une image en 256 niveaux de gris, chaque pixel est représenté sur 8 bits. Ainsi, un bon cryptosystème doit être en mesure de générer des images cryptées avec une entropie très proche de 8 bits/pixel.

3.6 Cryptographie basée sur la théorie du chaos

La sécurité d'un cryptosystème peut être étudiée de deux manières différentes : la sécurité calculatoire et la sécurité inconditionnelle. La sécurité calculatoire est liée à la quantité de calcul nécessaire pour casser un système cryptographique, et dépend donc de l'efficacité et de la rapidité des ordinateurs et des avancées technologiques. La sécurité d'un système cryptographique peut être étudiée d'une autre manière tout en supposant que nous disposons d'une puissance de calcul infinie. Dans ce cas, nous parlons de sécurité inconditionnelle ou parfaite. Selon Shannon [Shannon 1949], un système cryptographique assure une sécurité parfaite si la probabilité *a postériori* que le texte clair soit M, étant donnée le texte chiffré C est égale à la probabilité *a priori* que le texte clair soit M, c'est-à-dire $p(M|C) = p(M), \forall M \in \mathcal{M}$ et $\forall C \in \mathcal{C}$. Ce même principe peut se traduire également par le fait que le texte chiffré n'apporte aucune information sur le texte clair. Le chiffrement de Vernam, présenté section 3.3.1.2, est l'unique système cryptographique inconditionnellement sûr. Toutefois, ce chiffrement est impraticable car la clé doit être aussi longue que le message à chiffrer et n'est utilisée qu'une seule fois dans le chiffrement. De plus, le chiffrement de Vermann est vulnérable face a une attaque par texte clair connu. Il faut donc développer d'autres méthodes de chiffrement. Ainsi, pour avoir une clé de même taille que le texte clair, nous utilisons actuellement les générateurs de nombres pseudo-aléatoires GNPA [10]. En effet, un GNPA génère une séquence de nombres pseudo-aléatoires

10. (en anglais Pseudo-Random Number Generator PRNG)

à partir d'un ensemble de paramètres et d'un ensemble de valeurs initiales qui jouent le rôle de la clé. Il est à noter qu'un générateur de bonne qualité peut être utilisé dans le chiffrement à masque jetable pour assurer un niveau de sécurité très élevé. Actuellement, il existe plusieurs générateurs pseudo-aléatoires, les plus répandus étant ceux qui sont basés sur des méthodes de congruence linéaire. Pour être utilisable en cryptographie, un GNPA doit avoir de bonnes propriétés statistiques et spectrales. Ainsi, la séquence générée doit être imprédictible, uniformément distribuée, ayant une faible auto-corrélation et de longue période. Cependant, les générateurs à congruence linéaire ne satisfassent pas ces propriétés et la connaissance d'un nombre fini de bits de la suite générée peut servir pour prédire les bits qui suivent. Ainsi, un des problèmes de la cryptographie moderne est de développer des GNPA qui sont capables de générer des séquences aléatoires pourront être utilisées dans la conception des systèmes cryptographiques.

Depuis 1990, les chercheurs commencent à utiliser la théorie du chaos dans la génération des séquences de nombres pseudo-aléatoires. Les systèmes chaotiques [Kocarev 2001], qui sont des systèmes dynamiques non linéaires, génèrent à partir d'une suite de bits de taille relativement petite (jouant le rôle de la clé) une séquence de bits très longue avec des propriétés très intéressantes qui favorisent leurs utilisations à des fins de chiffrement [Yang 2004, Wu 2004, Zhang 2005, Wong 2008].

Parmi les propriétés les plus importantes d'un système chaotique est la grande sensible aux conditions initiales. Ainsi, deux conditions initiales très proches génèrent deux séquences totalement différentes. De plus, les séquences générées par un système chaotique sont aléatoirement imprédictibles, non périodiques et de faible auto-corrélation. Ces propriétés ont conduit à l'utilisation des systèmes chaotiques dans le chiffrement. Ainsi, dans un système de chiffrement chaotique, nous commençons par choisir la clé et par générer une séquence pseudo-aléatoire de bits. Cette dernière sera mélangée avec le texte clair afin de produire le texte chiffré. Pour déchiffrer le message, il suffit d'utiliser la même clé pour générer la même séquence servant à extraire le texte clair. Il est intéressant de préciser que pour qu'un système chaotique puisse être utilisé en cryptographie, il doit satisfaire également d'autres propriétés. Ainsi, l'ensemble des paramètres, qui forment la clé, doit être suffisamment grand pour résister contre les attaques par force brute. De plus, la séquence générée doit être uniformément distribuée, ce qui n'est pas toujours garanti par les systèmes chaotiques. La section suivante présente les tests statistiques utilisés dans l'évaluation des performances d'un GNPA afin d'être utilisé en cryptographie.

Il est à rappeler qu'il existe plusieurs schémas de chiffrement basés sur le chaos, nous

distinguons essentiellement l'addition chaotique, la modulation chaotique et la commutation chaotique. Les références [Kocarev 2001, Yang 2004] présentent plus de détails sur les systèmes chaotiques et leurs utilisations dans le chiffrement. Parmi les fonctions chaotiques les plus utilisées en cryptographie, il y a essentiellement la carte logistique [Kanso 2009] [11] qui est très sensible à la condition initiale et à son paramètre de contrôle. Toutefois, la sortie de cette fonction n'est pas uniformément distribuée dans l'intervalle $[0, 1]$. De plus, l'espace de clés est relativement petit et par conséquent est vulnérable à une attaque par force brute. Ainsi, l'étape la plus importante dans la conception d'un système de chiffrement par le chaos est le choix de la fonction chaotique [Alvarez 2009, Li 2001a] caractérisée par le nombre de paramètres, la vitesse de calcul et la complexité de mise en œuvre et liée au niveau de sécurité désiré.

3.7 GNPA et analyse de performance

Le développement d'un GNPA [Martin 2004, Kanso 2009, Patidar 2009b, Patidar 2009a] est une tâche très complexe. Notons qu'il est également complexe de trouver des méthodes pour le tester et le valider. Il est à rappeler qu'un bon GNPA doit être capable de générer des séquences de nombres imprédictibles, uniformément distribuées et caractérisées par une longue période. Pour vérifier ces propriétés, un ensemble de tests statistiques sont appliqués en sortie d'un GNPA. Par exemple, si nous considérons que les générateurs délivrent une séquence de bits, alors les bits doivent être uniformément distribués. Nous devons donc avoir $p(0) = p(1) = 1/2$ à n'importe quel instant, et par conséquent l'entropie de Shannon est maximale. De plus, les séquences générées doivent avoir une faible corrélation, ce qui aide à garantir l'imprédictibilité. Ainsi, vérifier l'uniformité, l'entropie, la corrélation et beaucoup d'autres propriétés statistiques des séquences générées, nous aide à la qualification d'un GNPA. Il existe plusieurs séries de tests statistiques utilisées pour la validation des GNPA, comme la série de tests DIEHARD développée par G. Marsagha de l'université de Floride, disponible sur [12], et la série de tests Crypt-XS développée pour le centre de recherche en sécurité de l'université du Queensland, disponible sur [13]. Il est à signaler que la série de tests la plus

11. (en anglais Logistic map)
12. http ://stat.fsu.edu/ geo/diehard.html
13. http ://www.isrc.qut.edu.au/cryptx/

célèbre et la plus utilisée est SP800 − 22 [Rukhin 2008] proposée par NIST [14], disponible sur [15], et contient pratiquement tous les tests présentés dans les autres séries de tests. Cette série est intitulée "a statistical test suite for random and pseudorandom number generator for cryptographic applications". Les tests statistiques sont validés selon une hypothèse à tester, appelée hypothèse nulle est notée par H_0. L'hypothèse H_0 est : "la séquence est aléatoire." et l'hypothèse H_a est : "la séquence n'est pas aléatoire.". Pour chaque test, une probabilité nommée p_value est calculée. Si $p\_value \geq \alpha$, avec α un niveau de signification le plus souvent égal à 0.01, alors l'hypothèse H_0 est acceptée, et par conséquent la séquence est supposée aléatoire avec une probabilité très importante. Actuellement, la série des tests de NIST comprend 16 tests statistiques employés pour vérifier le caractère aléatoire des séquences générées par un GNPA. Parmi ces tests, nous citons le test de fréquence (Frequency Test), le test de fréquence dans un block (Block Frequency Test) et le test sur la transformée de Fourrier discrète (Spectral Test ou bien Discrete Fourrier Transform Test).

3.8 Combinaison compression et cryptage

Pour sécuriser le transfert et l'archivage des images, un cryptosystème peut être intégré à n'importe quel étape du schéma de codage présenté figure 2.3. Ainsi, l'algorithme de chiffrement peut être appliqué sur les pixels d'une image, dans l'étape de transformation, ou bien dans l'étape du codage entropique ou même encore dans le flux compressé. Il est à noter que le chiffrement des pixels de l'image, ou bien le chiffrement des symboles générés par la transformation réduit l'efficacité du codage puisqu'il modifie les statistiques des données à coder qui auront une entropie maximale après le cryptage.

Pour surmonter ces problèmes, il est préférable d'appliquer le chiffrement dans le codage entropique. Ainsi, un codeur de crypto-compression propose de modifier des paramètres ou bien certaines fonctions afin d'appliquer conjointement la compression et le cryptage dans une même étape. Dans ce cas, le codeur entropique crypte les données en utilisant une clé secrète, et sans la connaissance de cette clé, il sera impossible de décoder et de reconstruire les données originales. Le développement d'un système

14. Il existe une implémentation en C++ de tous les tests et elle est accessible librement à partir du site de NIST. Durant la préparation de cette thèse, nous avons utilisé la version 1.6, développée en Décembre 1999.

15. http ://csrc.nist.gov/groups/ST/toolkit/rng/index.html

qui combine la compression et le cryptage en un seul processus est une tâche très délicate. Ainsi, un tel système doit conserver l'efficacité de la méthode de compression, tout en sécurisant les données. De plus, la combinaison de la compression et du cryptage en une même étape doit pouvoir s'exécuter en un temps inférieur à celui nécessaire pour appliquer une compression suivie d'un cryptage. Le CA et le codage de Huffman sont les codeurs entropiques les plus utilisés dans les dernières normes et standards de compression, et sont largement exploités pour combiner la compression avec le cryptage [Cleary 1995, Helen 1993, Wen 2006, Kim 2007, Mi 2008].

3.9 Conclusion

Dans ce chapitre, nous avons introduit les notions de base de la cryptographie, ainsi que plusieurs algorithmes et techniques de chiffrement développés au fil des siècles ou bien utilisés récemment dans l'échange sécurisé des données à travers les réseaux de communication. Ensuite, nous avons présenté les différents types des algorithmes de chiffrement symétrique, asymétrique et hybride. Nous avons abordé également plusieurs techniques de cryptanalyses, et nous avons détaillé en particulier les procédures à mettre en œuvre pour analyser les performances des algorithmes de chiffrement d'images.

Ce chapitre a introduit également la théorie du chaos qui devient de plus en plus utilisée dans les algorithmes de chiffrement moderne et dans la génération des séquences de nombres pseudo-aléatoires. Enfin, nous avons introduit les nouveaux systèmes de crypto-compression qui visent à combiner la compression et le cryptage dans un seul procédé, ce qui permet de réduire la taille des données, et en même temps, de les protéger contre tout type d'attaque.

Chapitre 4

Compression sans perte d'images basée sur un nouveau codage arithmétique adaptatif

Sommaire

4.1 Introduction

L'objectif de ce chapitre est de présenter les deux méthodes développées au cour de ma thèse, pour la compression sans perte d'images et basées sur le codage arithmétique (CA). La première méthode que nous allons présenter, section 4.2, consiste en un nouveau CA adaptatif (CAA) qui met à jour la table de probabilités après le codage de la dernière occurrence de chaque pixel. Nous montrons dans cette méthode qu'en codant l'image bloc par bloc après les avoir triés selon la moyenne, il est possible d'augmenter les taux de compression obtenus par les CA conventionnels sans augmenter leur complexité.

La deuxième méthode que nous allons détailler, section 4.3, se base sur l'utilisation du CAA d'ordre 0 (CAA-0) tout en représentant les fréquences des pixels avec une précision réduite. Cette méthode est de plus en plus efficace en codant l'image bloc par bloc après les avoir triés selon un critère basé sur la distance de Kullback. Dans les deux sections 4.2 et 4.3, nous détaillons les résultats expérimentaux obtenus par l'application de nos approches sur des images en niveaux de gris de différentes tailles. Finalement, nous terminons ce chapitre par une conclusion, section 4.4.

4.2 Un nouveau codage arithmétique adaptatif pour la compression sans perte d'images

4.2.1 Introduction

Récemment, plusieurs schémas de compression sans perte ont été proposés [Chuang 1998, Carpentieri , Sudharsanan 2000, Xiaolin 1996, Carpentieri 1997, Golchin 1998]. La plupart de ces schémas se base sur le CA, grâce à son efficacité en terme de taux de compression et sa capacité de s'affranchir aux limites des autres codeurs statistiques (comme le codage de Huffman) en associant à chaque symbole à coder un nombre non entier de bits.

Le CA utilise une estimation des probabilités des symboles à coder pour générer le flux binaire représentant les données compressées. Ainsi, pour estimer ces probabilités, le CA est précédé par une étape de modélisation statistique de la source afin de compresser les données en un nombre minimum de bits possibles. La modélisation et le codage sont deux étapes très étroitement liées, toutefois, elles sont indépendantes. D'ailleurs, la combinaison la plus utilisée dans la compression sans perte est l'utilisation de la modélisation statistique

avec le CA.

Dans l'étape de modélisation statistique, le CA peut fonctionner avec une table de probabilités exacte des symboles à coder. Cette dernière est obtenue à partir de la fréquence d'apparition de chacun de ces symboles, et il s'agit dans ce cas d'un CA statique (CAS) [Rubin 1979]. Malgré le coût de traitement relatif au calcul des fréquences, ce modèle reste rapide, efficace et offre des résultats de compression très satisfaisants pour certaines applications. En outre, le CA peut fonctionner avec un modèle adaptatif qui pour chaque nouveau symbole estime sa probabilité de façon à s'adapter le plus possible à la vraie statistique de la source. Dans ce cas, le CA devient adaptatif (CAA) [Witten 1987, Salomon 2007] offrant des résultats de compression meilleurs que le modèle statique. Néanmoins, le modèle adaptatif nécessite des coûts additionnels relatifs à la mise à jour de la table des probabilités, et par conséquent, le CAA est nettement moins rapide par rapport au CAS. De plus, le CAA permet de réduire le volume de données d'en-tête à transmettre et il tente d'affiner la loi de probabilités au fur et à mesure que les symboles sont lus et codés en fonction du passé causal des symboles encodés.

La section 4.2.2 est une introduction à la modélisation statistique et au CA. Ensuite, la section 4.2.3 montre l'optimalité du CAS et sa capacité d'obtenir un nombre moyen de bits par symbole très proche de l'entropie théorique de la source à coder. La section 4.2.4 est consacrée à détailler notre CAA, alors que dans les sections 4.2.5 et 4.2.6, nous proposons de comparer théoriquement et expérimentalement notre CAA avec les codeurs arithmétiques conventionnels. Nous finissons cette section par une conclusion, section 4.2.7.

4.2.2 Modélisation statistique et codage arithmétique

Un modèle M est une fonction définie par :

$$\begin{array}{rccl} M: & A & \rightarrow & [0,1) \\ & \alpha_i & \rightarrow & P_M(\alpha_i). \end{array} \tag{4.1}$$

Cette fonction estime ou calcule une probabilité $P_M(\alpha_i)$ pour chaque symbole α_i appartenant à un alphabet A. Le modèle est dit adaptatif d'ordre-n, s'il estime la probabilité d'un symbole en utilisant un contexte obtenu à partir des n symboles passés en causal. Le modèle adaptatif d'ordre-0 calcule la probabilité du symbole α_i en se basant uniquement sur la probabilité déjà calculée du même symbole, et il s'agit dans ce cas d'un CAA d'ordre-0 (CAA-0). Dans un CAA-0, nous commençons par un même nombre d'occurrence pour

chaque symbole, généralement égal à 1. Ensuite, ce nombre est mis à jour chaque fois qu'un nouveau symbole est codé par l'ajout de 1 à sa fréquence d'apparition. Le nombre total de symboles est également mis à jour. Les nouvelles probabilités sont utilisées par le CAA-0 pour coder le prochain symbole. Le CAA-0 est rapide et offre des résultats de compression proche du CAS, néanmoins, du fait de sa rapidité il est largement exploité en compression sans perte.

Il est à noter qu'il est difficile de comparer théoriquement les performances des CAA, toutefois il reste possible de les comparer par des analyses expérimentales en se basant sur plusieurs facteurs comme le taux de compression, le nombre moyen de bits par symboles (entropie) et le temps nécessaire pour réaliser les opérations de codage et de décodage. Le calcul entropique peut servir également à comparer les performances des CAA. Ainsi, d'après 2.9, l'entropie $H_M(S)$ de la source S dans le modèle M est donnée par :

$$H_M(S) = \sum_{i=1}^{n} P(\alpha_i) \log_2(\frac{1}{P_M(\alpha_i)}). \tag{4.2}$$

D'après cette équation, nous remarquons que l'entropie dépend du modèle M, puisque $P_M(\alpha_i)$ sont des probabilités dans le modèle M. La valeur $\log_2(\frac{1}{P_M(\alpha_i)})$ peut être interprétée comme la longueur du plus court mot binaire représentant le symbole α_i, alors que la valeur $P(\alpha_i)$ représente la probabilité exacte du symbole α_i dans la source S et est servi pour calculer le nombre de bits nécessaire pour coder le symbole α_i. Ainsi, le modèle permettant d'obtenir la plus faible entropie est le plus performant.

Trouver un modèle qui met à jour les statistiques d'une source d'une manière efficace et rapide a motivé de nombreuses recherches sur le CAA. Carpentieri [Carpentieri 1997] a présenté un nouvel algorithme de compression sans perte d'images, qui sélectionne dynamiquement pour chaque nouveau pixel, une distribution de probabilités parmi un très grand nombre. La distribution choisie est utilisée par le CA pour coder l'erreur obtenue par prédiction. Ce schéma est très efficace en terme de taux de compression, néanmoins il nécessite des temps de calcul très important surtout pour des images de grandes tailles.

Contrairement à Carpentieri et au lieu d'utiliser plusieurs densités de probabilités, Matsuda *et al.* [Matsuda 2003] ont proposé de modéliser l'image erreur obtenue par prédiction par une distribution Gaussienne. Cette approche est efficace mais elle est également très lente. Ainsi, le codeur nécessite un temps de compression de plusieurs minutes, même avec des images de tailles relativement réduites (512×512 pixels), ce qui est énorme surtout pour les applications de transmission des données avec contraintes temps réel ou bien pour les systèmes caractérisés par une faible capacité de traitement.

Plus récemment, Kuroki *et al.* [Kuroki 2004] ont présenté un nouveau CAA pour le codage sans perte d'images. Dans leur approche, une image est tout d'abord transformée en une image erreur grâce à un codage prédictif. L'image erreur est ensuite codée par un CAA qui modélise les erreurs à coder par une distribution Laplacienne de moyenne nulle. Cette technique est plus efficace que les codeurs arithmétiques conventionnels et a apporté une amélioration moyenne des performances de l'ordre de 5%. D'après les approches étudiées dans cette section, nous pouvons conclure que le CAA nécessite des temps de calcul très importants et il est basé sur des modèles statistiques très complexes.

Ainsi, l'originalité de notre approche est de mettre en œuvre un modèle statistique rapide, simple à implémenter et plus efficace en terme de compression que les CA classiques.

4.2.3 Optimalité du CAS

Soit une suite $X = \{x_1, \cdots, x_m\}$ de m symboles produite à partir d'une source S qui prend ses valeurs dans un alphabet A composé de n symboles α_k $(1 \leq k \leq n)$. La distribution de probabilités des symboles à coder est notée par $P = \{p_1, \cdots, p_n\}$, avec $p_k = p(\alpha_k) = \frac{f_k}{m}$ et f_k représente la fréquence d'apparition du symbole α_k dans la suite X.

D'après la théorie de l'information, le nombre moyen de bits nécessaire pour coder chaque symbole d'une source S ne peut pas être plus petit que l'entropie de Shannon [Shannon 1948a] $H(S)$ présentée équation 2.9. Le CA génère en sortie un intervalle $[L, H)$ et n'importe quelle valeur de cet intervalle permet d'identifier d'une manière unique le message d'entrée X. Le nombre de bits minimum B_{min} nécessaire pour représenter la valeur choisie dépend de la probabilité du message $P(X)$ qui correspond à la taille de l'intervalle $[L, H)$.

$$P(X) = p_1 \times \cdots \times p_m = \prod_{i=1}^{m} p_i. \tag{4.3}$$

Ainsi, B_{min} peut être calculé [Howard 1994] :

$$B_{min} = \lfloor log_2(\frac{1}{P(X)}) \rfloor + 2 \; bits, \tag{4.4}$$

avec $\lfloor x \rfloor$ est la partie entière de x. Pour montrer l'efficacité de notre approche, nous proposons tout d'abord d'exposer l'optimalité du CAS, ensuite de prouver théoriquement

qu'il est toujours possible, grâce à notre approche, d'avoir des résultats meilleurs que ceux obtenus par le CAS.

Nous allons montrer que le CAS est optimal et génère des flux de données compressées ayant une entropie B_S très proche de $H(S)$. Le CAS utilise la table des probabilités des symboles pour coder et décoder le message X. Cette dernière doit être envoyée au décodeur afin d'assurer la décodabilité des données originales. Nous supposons que les données additionnelles sont codées sur σ bits, alors le nombre moyen de bits par symbole nécessaire pour coder le message X est borné par :

$$B_S \leq \frac{\sigma + log_2(\frac{1}{P(X)})}{m}, \tag{4.5}$$

d'après l'équation 4.3, nous déduisons :

$$B_S \leq \frac{\sigma + log_2(\frac{1}{\prod_{i=1}^{m} p(x_i)})}{m} = \frac{\sigma + \sum_{i=1}^{m} log_2(\frac{1}{p(x_i)})}{m}, \tag{4.6}$$

Le symbole x_i correspond à un élément α_i de l'alphabet A, et par conséquent nous obtenons :

$$\sum_{i=1}^{m} log_2(\frac{1}{p(x_i)}) = \sum_{i=1}^{n} f_i log_2(\frac{1}{p(\alpha_i)}), \tag{4.7}$$

puisque chaque symbole α_i est répété f_i fois dans le message X. D'après les équations 4.6 et 4.7 :

$$B_S \leq \frac{\sigma + \sum_{i=1}^{n} f_i log_2(\frac{1}{p(\alpha_i)})}{m}, \tag{4.8}$$

ce qui correspond finalement à :

$$H(S) \leq B_S \leq H(S) + \frac{\sigma}{m}, \tag{4.9}$$

et par conséquent :

$$\lim_{m \to +\infty} B_S = H(S). \tag{4.10}$$

Ce dernier résultat signifie que le CAS est optimal, néanmoins il est toujours possible de développer des modèles qui sont plus performant.

4.2.4 Le CAA proposé

Nous proposons un nouveau CAA qui met à jour la probabilité du symbole courant tout en se basant sur tous les symboles qui le précédent dans la source S afin de réduire le nombre de bits nécessaire pour coder les suites de symboles générées par cette source.

Notre CAA nécessite deux passes. Ainsi, durant la première passe, nous devons disposer de la totalité de la séquence à compresser afin de calculer les probabilités exactes de tous les symboles. Quant à la deuxième passe, la table des probabilités est mise à jour dès que la dernière occurrence d'un symbole codé est détectée. Ainsi, si un symbole α_i est totalement codé, alors il sera plus performant de supprimer sa probabilité $P(\alpha_i)$ et de mettre à jour le nombre total de symboles de la séquence à coder. Soit M notre modèle de codage, défini par la fonction :

$$\begin{array}{rccl} M: & A & \rightarrow & [0,1) \\ & \alpha_i & \rightarrow & P_M(\alpha_i) = p(\alpha_i|\alpha_1,\ldots,\alpha_{i-1}). \end{array} \tag{4.11}$$

La mise à jour proposée joue un rôle très important et permet d'augmenter les probabilités des autres symboles $P_M(\alpha_j)$, et par conséquent, le nombre de bits nécessaire pour coder chaque symbole $\log_2(\frac{1}{P_M(\alpha_j)})$ se réduit à chaque fois qu'un symbole est totalement codé.

Soit N_{CAS} le nombre total de bits nécessaire pour représenter la séquence X en appliquant un CAS :

$$\begin{array}{rcl} N_{CAS} & = & \lfloor -log_2(\prod_{i=1}^{n} p_i^{f_i}) \rfloor + 2 \\ & = & \lfloor -log_2(\prod_{i=1}^{n} (\frac{f_i}{m})^{f_i}) \rfloor + 2. \end{array} \tag{4.12}$$

Cependant, N_{CAA} représente la taille du flux compressé obtenu par notre approche :

$$N_{CAA} = \lfloor -log_2(\prod_{i=1}^{n} p(\alpha_i|\alpha_1,\ldots,\alpha_{i-1})^{f_i}) \rfloor + 2. \tag{4.13}$$

Pour montrer que notre modèle est plus performant que le modèle statique, il faut que :

$$H_M(S) \leq B_S, \tag{4.14}$$

avec H_M est l'entropie obtenue par notre CAA, alors que B_s représente l'entropie obtenue par le CAS. Ainsi, ceci revient à montrer que :

$$N_{CAS} < N_{CAA}, \tag{4.15}$$

et par conséquent, il suffit de montrer que :

$$\prod_{i=1}^{n} p(\alpha_i|\alpha_1,\ldots,\alpha_{i-1})^{f_i} > \prod_{i=1}^{n} (\frac{f_i}{m})^{f_i}. \tag{4.16}$$

Nous allons développer, section 4.2.5, une démonstration mathématique afin d'éprouver théoriquement l'efficacité de notre approche par rapport au CAS. Plus tard, section 4.2.6, nous proposons d'observer et de calculer par l'expérimentation le gain moyen de notre CAA par rapport au CAS et au CAA-0 en terme de taux de compression.

Le modèle proposé consiste à mettre à jour les probabilités des symboles à coder après le codage de la dernière occurrence de chaque symbole. Ainsi, l'efficacité de ce modèle dépend des positions des symboles dans la séquence à coder. D'une manière générale, l'image est codée pixel par pixel en commençant du coin haut à gauche et continuant de gauche à droite pour chacune des lignes jusqu'au coin inférieur droit. Ce type de parcours n'est pas forcément efficace en terme de compression, du fait qu'il ne prend pas en considération les mises à jour proposées par notre approche et par conséquent il faut chercher le sens de parcours qui permet de bien exploiter le principe proposé et qui fournit les meilleurs résultats.

Une approche possible consiste à tester tous les parcours possibles afin de retenir celui qui donne le meilleur résultat. Ceci n'est pas possible en pratique, puisque ce type de recherche, la recherche exhaustive, nécessite des temps de calcul énorme et l'envoi dans l'en-tête du fichier compressé, en plus de la table des probabilités, la position initiale de chaque pixel afin d'assurer la reconstruction exacte de l'image originale. Pour résoudre ce problème et réduire l'information additionnelle qu'il faut transmettre dans l'en-tête, nous proposons de découper l'image en blocs de pixels. Il nous suffit alors de transmettre uniquement la position initiale de chaque bloc dans l'image originale ce qui garantit une réduction très importante des informations additionnelles.

Même si ce problème est résolu, il reste à trouver une solution pour la recherche exhaustive de tous les parcours possibles des blocs de l'image et de proposer une démarche rapide et simple à mettre en œuvre afin de trouver un parcours qui garantit des résultats satisfaisants une fois associé avec le modèle statistique proposé.

La solution que nous proposons pour ce problème consiste à trier les blocs de l'image selon un critère facile à mesurer et qui s'applique le mieux possible à notre modèle. Le critère choisi dans notre étude expérimentale est la moyenne $M_b = \frac{1}{N_b \times N_b} \sum_{i=0}^{N_b-1} \sum_{j=0}^{N_b-1} I(i,j)$ qui est calculée pour des blocs de taille $N_b \times N_b$ pixels, et qui présente l'avantage de pouvoir trier les blocs du plus clair vers le plus sombre. Ce type de tri permet de grouper au début les pixels de fortes intensités, suivi de ceux ayant une moyenne intensité, et à la fin nous trouvons les blocs de pixels sombres, c'est à dire les blocs qui sont composés par des pixels de faibles intensités. La figure 4.3 montre une image découpée en blocs de

tailles 32×32 pixels et triés selon la moyenne.

Il est à noter qu'il existe d'autres critères de tri, comme l'entropie ou bien la variance, néanmoins, suite à une étude expérimentale très approfondie nous avons remarqué que les meilleurs résultats sont obtenus en triant les blocs selon la moyenne. Une description simplifiée des processus de codage et de décodage est représentée respectivement par les figures 4.1 et 4.2.

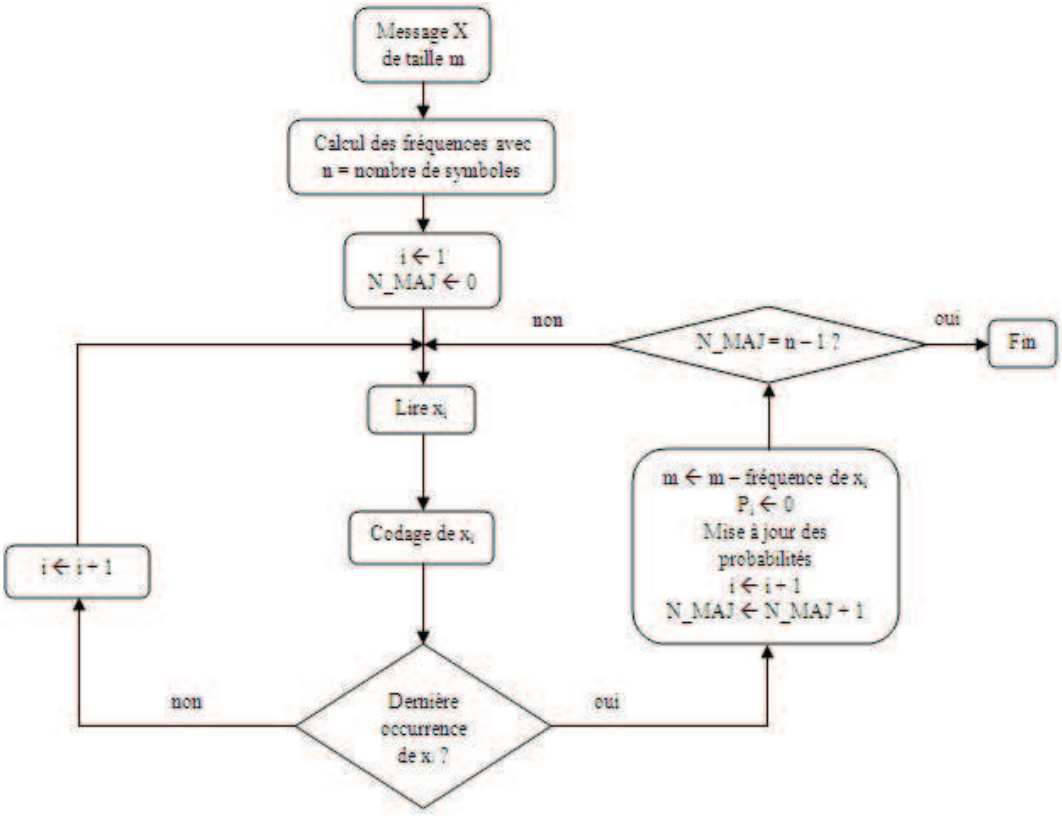


FIGURE 4.1 – Le schéma de codage proposé.

La figure 4.3 représente une image découpée en blocs de tailles 32×32 pixels et triés selon la moyenne.

4.2.5 Preuve mathématique

Dans cette section, nous présentons une démonstration mathématique afin de montrer que notre CAA est plus efficace que le CAS et offre toujours des meilleurs taux de compression. Ainsi, nous démontrons par récurrence sur n l'inégalité 4.16. Nous commençons par $n = 2$. Donc, nous supposons que le message X prend ses valeurs à partir d'un

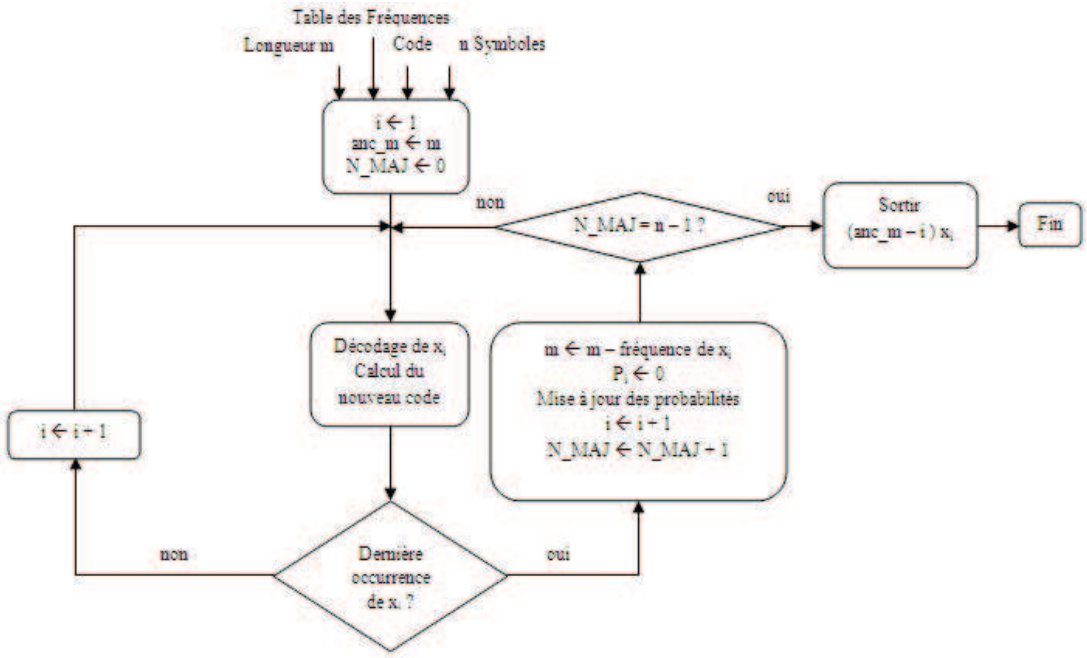
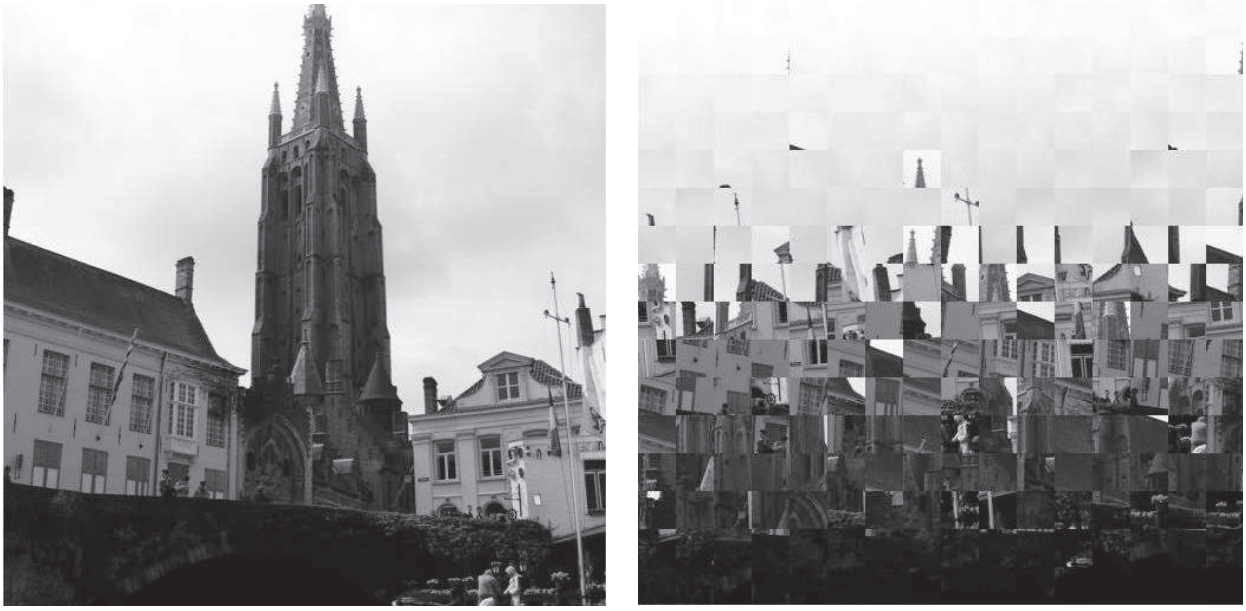


FIGURE 4.2 – Le schéma de décodage proposé.



(a) (b)

FIGURE 4.3 – a) Image originale de taille 512×512 pixels, b) Image originale découpée en blocs de tailles 32×32 pixels et triés selon la moyenne.

alphabet composé uniquement par deux symboles $\mathcal{A} = \{\alpha_1, \alpha_2\}$ avec leurs probabilités respectives $\frac{f_1}{m}$, $\frac{f_2}{m}$. Sans perte de généralité, supposons que le message X se termine par le symbole α_2.

En utilisant le CAS pour coder la séquence X, le nombre de bits nécessaire peut être calculé à partir du produit des probabilités P_{CAS} : :

$$P_{CAS} = (\frac{f_1}{m})^{f_1}(\frac{f_2}{m})^{f_2}. \tag{4.17}$$

Par contre, en utilisant notre approche, le message X est représenté par un flux binaire dont la taille est calculée à partir du produit des probabilités P_{CAA}.

$$P_{CAA} = (\frac{f_1}{m})^{f_1}(\frac{f_2}{m})^{T_1-f_1}, \tag{4.18}$$

avec $T_1 = inf\{k \geq 1 \mid \textit{codage de la derniere occurrence de } \alpha_1\}$.

De la sorte, nous devons comparer P_{CAS} et P_{CAA} présentées équations 4.17 et 4.18 respectivement. Il suffit donc de comparer $(\frac{f_2}{m})^{f_2}$ avec $(\frac{f_2}{m})^{T_1-f_1}$, et par conséquent il faut vérifier que $T_1 - f_1 < f_2$.

Nous avons $T_1 < m$, donc $T_1 - m < 0$ et $T_1 - m + f_2 < f_2$. Nous connaissons déjà que $m = f_1 + f_2$, alors $T_1 - f_1 = T_1 - m + f_2$, et nous pouvons donc conclure que :

$$T_1 - f_1 < f_2. \tag{4.19}$$

D'après l'équation 4.19, il devient facile de conclure que $N_{CAA} < N_{CAS}$, et par conséquent que la taille des données compressées obtenue par notre approche est inférieure à celle obtenue par le CAS.

Maintenant, nous choisissons $n > 2$, et nous supposons que le message X est composé de n symboles d'un alphabet $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$ avec leurs probabilités respectives $\frac{f_k}{m}$,$(1 \leq k \leq n)$. Sans perte de généralité, on suppose que le message X se termine par le symbole α_n.

La taille en bits des données compressées en utilisant le CAS est toujours liée à l'équation 4.12. Soit T_i défini par $T_i = inf\{k \geq 1 \mid \textit{codage de la derniere occurrence de } \alpha_i\}$, pour $i \in \{1, \ldots, n-1\}$, avec $T_i < T_j$ si $i < j$ pour $i, j \in \{1, \ldots, n-1\}$.

Notons par $f_k^{(i)}$ le nombre d'occurrences de α_k dans l'intervalle $]T_{i-1}, T_i]$, avec $f_k = \sum_{i=1}^{k} f_k^{(i)}$ pour $i \in \{1, \ldots, n-1\}$ et $T_0 = 1$. Pour comparer le CAS avec notre CAA, il faut montrer que :

$$\prod_{k=1}^{n-1}\prod_{i=1}^{k}(\frac{f_k}{m - f_{i-1}})^{f_k^{(i)}} > \prod_{k=1}^{n-1}(\frac{f_k}{m})^{f_k}. \tag{4.20}$$

Nous connaissons déjà que :

$$\prod_{i=1}^{k}(\frac{f_k}{m-f_{i-1}})^{f_k^{(i)}} > \prod_{i=1}^{k}(\frac{f_k}{m})^{f_k^{(i)}}. \tag{4.21}$$

avec

$$\begin{aligned}\prod_{i=1}^{k}(\tfrac{f_k}{m})^{f_k^{(i)}} &= (\tfrac{f_k}{m})^{(\sum_{i=1}^{k} f_k^{(i)})} \\ &= (\tfrac{f_k}{m})^{f_k} \quad .\end{aligned} \tag{4.22}$$

Par conséquent, l'équation 4.20 est vraie. Ainsi, nous pouvons conclure que le CAS est moins efficace que notre CAA, ce qui nous permet de réduire davantage la taille des fichiers compressés. Dans la section 4.2.6, nous présentons une étude expérimentale afin de calculer le gain moyen apporté par notre CAA par rapport au CAS et CAA-0.

4.2.6 Résultats expérimentaux

Les résultats présentés dans cette section sont obtenus en utilisant une implémentation entière du CA basée sur le principe décrit par [Witten 1987]. Il est à noter que la précision utilisée est de 64 bits et le nombre total des fréquences est représenté sur 30 bits. Il est très intéressant de bien choisir ces paramètres, d'une part pour assurer la décodabilité des données compressées, et d'autre part pour bien comprendre les valeurs trouvées et présentées dans notre étude expérimentale. Il est à rappeler qu'il est possible de trouver des résultats peu semblables suite à une simple modification dans ces paramètres. Nous présentons, les sections 4.2.6.1 et 4.2.6.2, une étude comparative entre notre approche et les codeurs arithmétiques classiques (CAS et CAA-0) en terme de taux de compression et de temps de calcul.

4.2.6.1 Comparaison des taux de compression

Dans la section 4.2.4, nous avons détaillé le principe de notre approche, et dans cette section nous présentons une étude comparative des taux de compression obtenus sur des images réelles. Notre approche a été testée sur 100 images choisies aléatoirement de la base d'images (BOWS) [1]. Les images ont une taille de 512×512 pixels et sont en niveaux de gris avec 8 bits/pixel. Nous avons calculé pour chaque image le taux de compression obtenu

1. http ://bows2.gipsa-lab.inpg.fr/

afin de valider et d'évaluer les performances de notre approche. Le taux de compression, donné par l'équation 2.2, permet de calculer le rapport entre le nombre de bits de l'image originale et celui de l'image compressée.

Les valeurs moyenne, minimale et maximale des taux de compression de toutes les images de tests sont listées dans le tableau 4.1. Selon ces résultats, nous pouvons conclure que notre approche s'avère toujours plus efficace en terme de taux de compression avec un gain moyen situé entre 5.201% et 5.881% par rapport au CAA-0 et au CAS. Ainsi, les résultats obtenus ont montré une augmentation moyenne du taux de compression d'un facteur de 5.5% par rapport aux codeurs arithmétiques classiques. Il est à noter que les résultats présentés dans le tableau 4.1 sont obtenus après avoir ajouté dans l'en-tête du fichier compressé la table des probabilités et les positions initiales des blocs de pixels. Des tests sur d'autres images très fréquemment utilisées dans l'évaluation des algorithmes

TABLE 4.1 – Résultats de comparaison entre notre CAA, le CAS et le CAA-0.

	CAS	CAA-0	**CAA** proposé		
	taux de compression	taux de compression	taux de compression	Gain(CAS) (%)	Gain(CAA-0) (%)
moyenne	1.140	1.144	**1.207**	5.811	5.201
min	1.046	1.047	**1.094**	0.602	0.497
max	1.998	2.001	**2.102**	11.310	7.087

de compression [2], ont montré des résultats semblables. Les résultats présentés dans le tableau 4.2 ont mis en évidence une augmentation moyenne du taux de compression d'un facteur de 5.07% par rapport au CAS et de 4.46% par rapport au CAA-0. Il est à noter que les résultats présentés dans les tableaux 4.1 et 4.2 sont très satisfaisants du fait que nous appliquons une compression sans perte d'images et pour certaines images notre approche fonctionne beaucoup mieux que le CAS et le CAA-0 offrant un gain du taux de compression pouvant atteindre 13%.

2. http ://links.uwaterloo.ca/repository.html

TABLE 4.2 – Résultats de Comparaison entre notre CAA, le CAS et le CAA-0 appliqués sur des images standards.

	CAS	CAA-0	**CAA proposé**		
	taux de compression	taux de compression	taux de compression	Gain(CAS) (%)	Gain(CAA-0) (%)
Barbara	1.068	1.070	**1.096**	2.55	2.37
Boat	1.103	1.122	**1.133**	2.64	0.97
France	1.229	1.234	**1.423**	13.63	13.28
Goldhill	1.066	1.069	**1.102**	3.26	2.99
Lena	1.071	1.073	**1.117**	4.11	3.93
Peppers	1.053	1.056	**1.079**	2.40	2.13
Zelda	1.085	1.100	**1.135**	4.40	3.08
moyenne	**1.097**	**1.104**	**1.155**	**5.07**	**4.46**

4.2.6.2 Comparaison des temps de calcul

Il existe un autre critère pour comparer l'efficacité des techniques de compression, à savoir le temps de calcul. Ainsi, supposons que nous avons une image composée de m pixels qui prend ses valeurs à partir de n niveaux de gris. Le CAA-0 [Salomon 2007] met à jour la table des probabilités $\frac{n}{2}$ fois pour chaque pixel codé, toutefois, notre approche permet de réduire énormément le nombre de mise à jour. Ce dernier est égal à $n-1$ et ceci quelque soit la taille de l'image à coder. Les résultats présentés dans le tableau 4.3 sont obtenus pour des images de différentes tailles et en utilisant un ordinateur caractérisé par la configuration suivante : Intel core 2 Duo 2.93 GHZ CPU avec 2 GB de RAM. D'après le tableau 4.3 nous remarquons que notre approche est nettement plus rapide que le CAA-0. Ainsi, le CAA-0 nécessite $1.06s$ pour coder une image de taille 1024×1024 pixels, alors que notre CAA nécessite uniquement $0.36s$.

TABLE 4.3 – Comparaison des temps de calcul entre notre CAA et le CAA-0.

Taille de l'image size en pixels	Temps de calcul avec le CAA-0 (s)	Temps de calcul avec notre CAA (s)
256×256	0.06	0.03
512×512	0.25	0.09
1024×1024	1.06	0.36

4.2.7 Conclusion

Dans la section 4.2, nous avons présenté et analysé les performances d'un nouveau schéma de compression sans perte d'images basé sur le CA [Masmoudi 2010a]. Le schéma proposé consiste à appliquer un CAA qui met à jour la table de probabilités des pixels à coder uniquement après la détection de la dernière occurrence de chaque pixel. Le schéma présenté est très efficace en terme de taux de compression et permet d'obtenir un gain moyen de 5.5% par rapport aux résultats obtenus par les codeurs arithmétiques conventionnels. De plus, le schéma proposé est nettement plus rapide qu'un CAA-0. L'efficacité de notre approche a été prouvée théoriquement et par l'expérimentation après l'avoir

appliquée sur plusieurs images de tests.

4.3 Amélioration du CAA-0

4.3.1 Introduction

Le CA a été décrit pour la première fois par [Abramson 1963] en 1963 et il a fallu attendre presque 20 ans pour avoir une première implémentation praticable pour ce type de codage. Actuellement, la plupart des implémentations utilisées en pratique sont à l'origine de celle proposée par [Witten 1987]. Cette dernière est basée sur une implémentation entière avec une précision finie, ce qui présente l'avantage d'être rapide et de s'affranchir aux limites des implémentations basées sur les nombres flottants en autorisant le codage d'une séquence composée par un très grand nombre de symboles. Nous montrons que l'utilisation d'une précision réduite pour représenter les fréquences des pixels, dans le cas de la compression d'images, peut améliorer les performances du CAA-0 en autorisant ce dernier à s'adapter à la variation dynamique dans les statistiques des images. De plus, nous montrons que le codage d'image bloc par bloc après les avoir organisés selon un critère basé sur la distance de Kullback, peut également améliorer le CAA-0 en terme de taux de compression surtout pour des images présentant des propriétés statistiques inter-blocs très variées.
Dans la section 4.3.2, les notions de bases de l'implémentation entière du CA sont rappelées tout en détaillant les exigences qu'il faut satisfaire pour assurer la décodabilité des données compressées. Puis, section 4.3.3, l'étude de l'effet de la précision de représentation des fréquences sur les performances du CAA-0 est présentée. Le CAA-0 bloc par bloc des images fait l'objet de la section 4.3.4. Dans la section 4.3.5, nous détaillons la méthode de codage basée sur l'utilisation d'une précision réduite pour représenter les fréquences des pixels à coder après avoir découper l'image en des blocs et de les trier selon un critère calculé grâce à la distance de kullback. Finalement, nous terminons cette partie par une conclusion, section 4.3.6.

4.3.2 Implémentation du CA

La plupart des implémentations du CA sont basées sur une implémentation entière avec une précision finie, généralement égale à 16 ou 32 bits. Plus la précision augmente plus les résultats de compression sont améliorés. L'utilisation d'une implémentation entière est beaucoup plus rapide qu'une implémentation avec des nombres flottants. Toutefois, elle réduit la précision de calcul durant le codage et la précision de représentation des

probabilités. De ce fait, l'utilisation des implémentations entières avec 64 bits (ou plus) permet de résoudre les limitations liées à l'utilisation d'une précision réduite.

Il est à noter que pour n'importe quelle implémentation du CA, il faut définir des contraintes sur la somme maximale des fréquences et sur la valeur du code. Ainsi, soit p le nombre de bits définissant la précision de l'implémentation utilisée, f le nombre de bits utilisé pour représenter les fréquences et c le nombre de bits utilisé pour représenter la valeur du code. Pour que l'implémentation fonctionne correctement, il faut tout d'abord que les limites L et H obtenues par le CA soient dans l'intervalle $[0, 2^p)$. Il faut également garantir les inégalités suivantes :

$$\begin{matrix} f \leq c-2 \\ f+c \leq p \end{matrix}. \tag{4.23}$$

Ainsi, pour une implémentation de 64 bits ($p = 64$), il est possible de représenter la valeur du code sur 32 bits ($c = 32$) et les fréquences sur 30 bits ($f = 30$). Dans le modèle statique, si la somme des fréquences de tous les symboles dépasse la valeur $Max\_Frequence = 2^f$, qui définit la fréquence maximale, alors il suffit de diviser les fréquences par un facteur de redimensionnement et d'appliquer le codage avec les nouvelles valeurs. En revanche, dans un modèle adaptatif, la table des fréquences est redimensionnée à chaque fois que la somme des fréquences dépasse $Max\_Frequence$. Généralement, nous divisons les fréquences par 2 en appliquant l'algorithme 4 tout en s'assurant que les fréquences soient non nulles. Ceci est très utile du fait qu'il donne la possibilité au modèle de s'adapter à la variation statistique de la source.

Algorithme 4

1: **si** $total\_Freq \geq Max\_Frequence$ **alors**
2: $total\_Freq \leftarrow 0$
3: **pour** $i \leftarrow 0$ to $nbr\_symbole$ **faire**
4: $Freq[i] \leftarrow (Freq[i] + 1)/2$
5: $total\_Freq \leftarrow total\_Freq + Freq[i]$

Les fréquences estimées des $nbr\_symboles$ symboles par le modèle statistique sont représentées par un tableau de type entier nommé $Freq$. Ainsi, pour chaque symbole s_i ayant une fréquence $Freq[i]$, sa probabilité est calculée par $Freq[i]/total\_Freq$ avec $total\_Freq$ étant une variable entière qui correspond au nombre total des fréquences.

Dans le cas d'une implémentation avec une précision de 64 bits, la valeur de f est

30 bits. Cette dernière valeur est très grande pour la plupart des situations en pratique, et par conséquent les fréquences ne sont pas divisées par 2 lors du codage/décodage des symboles, ce qui diminue les performances des modèles adaptatifs. Ainsi, pour qu'un modèle adaptatif d'ordre 0 puisse se rapprocher au maximum de la variation dynamique dans la statistique de la source, il faut choisir la valeur adéquate pour f. C'est dans ce cadre que s'inscrit notre travail, qui consiste en premier lieu à étudier l'effet de la valeur choisie pour f sur la performance du CAA-0 et de proposer en deuxième lieu une nouvelle approche de compression sans perte d'images qui est basée sur l'utilisation du CAA-0 avec la valeur adéquate de f pour coder les blocs d'une image après les avoir ordonner selon la distance de Kullback [Kullback 1951, Kullback 1968, Johnson 2000].

4.3.3 Etude de l'effet de la précision de représentation des fréquences sur les performances du CAA-0

Le CAA-0 est un algorithme de codage très efficace qui est capable de coder tout type de source sans connaissance *à priori* de ses statistiques ce qui nous permet de le considérer comme un algorithme de codage universel. Le principe du CAA-0 est très simple et il consiste à commencer tout d'abord avec une estimation plus ou moins grossière de la source à coder, ensuite, au fur et à mesure du codage il s'adapte aux variations des statistiques par la mise à jour des fréquences des symboles à coder.

En effet, le CAA-0 commence avec une table de fréquences uniforme, c'est à dire nous supposons que tous les symboles ont initialement le même nombre d'occurrence, généralement égal à 1, et au fur et à mesure du codage, la fréquence $Freq(i)$ de chaque symbole s_i est mise à jour en lui ajoutant la valeur 1. Il est à noter que le codeur et le décodeur utilisent les mêmes fréquences initiales et qu'ils utilisent également les mêmes mises à jour du modèle statistique afin d'assurer la décodabilité des données compressées. Si la somme des fréquences de tous les symboles dépasse $Max\_Frequence$, alors les fréquences seront divisées par 2, autorisant par conséquent au CAA-0 de s'adapter à la variation des statistiques de la source.

Dans une implémentation entière avec une précision de 64 bits, la valeur $Max\_Frequence$ est égale à 2^f et doit satisfaire les contraintes présentées par l'équation 4.23. Généralement, la valeur choisie pour f est égale à 30. Pour une image en niveau de gris et de taille 512×512 pixels, le nombre total de fréquences est égal à 2^{18} et par conséquent, les fréquences des pixels ne seront pas divisées par 2 pour $f >= 18$ ce qui diminue

les performances du CAA-0. Le tableau 4.4 présente les résultats de codage en utilisant le CAA-0 pour compresser des images en niveaux de gris pour des valeurs différentes de f.

Il est à signaler que la plus petite valeur étudiée pour f est 9, puisque pour une image en niveau de gris chaque pixel est représenté sur 8 bits. Il importe de s'assurer que toutes les fréquences soient non nulles. Le tableau 4.4 présente les taux de compression des images compressées pour différentes valeurs de f, de plus, pour vérifier l'importance du choix d'une valeur réduite pour f, nous avons présenté les meilleurs taux de compression ainsi que les gains maximaux.

Selon les résultats trouvés, nous pouvons conclure qu'une réduction de la valeur de f permet d'améliorer les résultats de compression et nous pouvons atteindre un gain moyen de 3.611% par rapport au CAA-0 opérant avec $f = 18$. Le tableau 4.4 montre également que les meilleurs taux sont obtenus pour $f = 13$ et $f = 12$, et le taux moyen est nettement supérieur à celui obtenu par la plupart des implémentations. En effet, nous pouvons conclure que le choix de la valeur f et très important et qu'il agit sur la performance du CAA-0. De plus, les résultats favorisent l'utilisation d'une petite valeur pour f afin de donner la possibilité au CAA-0 de s'adapter à la variation de statistiques des images.

4.3.4 Découpage de l'image en blocs

L'image est généralement caractérisée par une forte redondance, et par conséquent les pixels voisins sont fortement corrélés. De plus, la loi de probabilités d'une région d'une image est généralement constante, toutefois, en passant d'une région à une autre cette loi peut changer énormément. La figure 4.5 présente deux séries de deux blocs consécutifs de l'image Lena ainsi que leurs distributions respectives.

Cette dernière caractéristique est très importante et nous pouvons l'exploiter pour améliorer les performances du CAA-0. Ainsi, au lieu de coder l'image ligne par ligne, nous proposons de la coder bloc par bloc selon le principe décrit par la figure 4.4 et de redimensionner la table des fréquences après le codage des pixels de chaque bloc. Nous présentons dans le tableau 4.5 les résultats de codage obtenus par le CAA-0 en opérant bloc par bloc et en faisant varier la valeur de f.

Nous remarquons que les meilleurs résultats sont obtenus pour des blocs de taille 32×32 pixels et avec $f = 9$. Les résultats trouvés ont montré une amélioration moyenne du CAA-0 d'un facteur de 14.636%. D'après ces résultats nous pouvons conclure que cette

TABLE 4.4 – L'effet de la précision f sur les performances du CAA-0 en codant les images ligne par ligne.

Image	Valeur f en bits											Taux maximal	Gain maximal
	18	17	16	15	14	13	12	11	10	9			
washsat	2.780	2.791	2.806	2.816	**2.817**	2.801	2.753	2.647	2.437	2.048		2.817	1.316
zelda	1.100	1.102	1.107	1.115	1.126	1.135	**1.140**	1.138	1.123	1.088		1.140	3.550
lena	1.074	1.083	1.097	1.114	1.132	1.149	1.161	**1.163**	1.152	1.123		1.163	7.700
goldhill	1.069	1.078	1.087	1.095	1.102	1.106	**1.107**	1.102	1.089	1.061		1.107	3.426
mandrill	1.086	1.089	1.093	1.100	1.106	**1.110**	**1.110**	1.104	1.089	1.060		1.110	2.146
mountain	1.285	1.295	1.306	1.316	1.322	**1.324**	1.319	1.304	1.271	1.212		1.324	2.900
peppers	1.056	1.059	1.070	1.085	1.100	1.112	**1.119**	1.118	1.109	1.090		1.119	5.614
barbara	1.047	1.049	1.055	1.063	1.072	1.081	**1.086**	1.085	1.076	1.056		1.086	3.585
boat	1.122	1.130	1.139	1.148	1.155	**1.159**	**1.159**	1.153	1.139	1.115		1.159	3.172
France	1.275	1.278	1.283	1.288	1.291	**1.293**	1.292	1.287	1.266	1.268		1.293	1.366
library	1.366	1.368	1.402	1.431	1.457	1.478	1.491	**1.492**	1.464	1.389		1.492	8.451
frog	1.611	1.636	1.664	1.689	1.709	**1.717**	1.709	1.678	1.607	1.475		1.717	6.189
moyenne	1.323	1.330	1.343	1.355	1.366	**1.372**	1.370	1.356	1.318	1.249		1.372	3.611

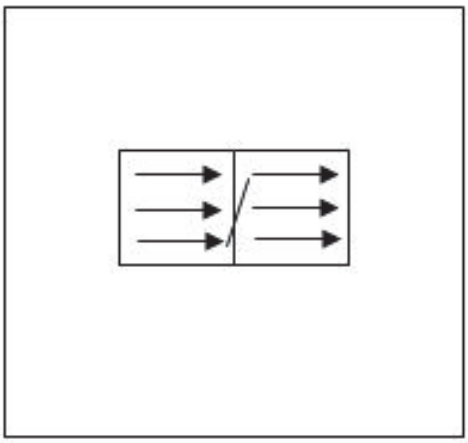


FIGURE 4.4 – Codage par bloc d'images.

approche devient de plus en plus performante s'il existe une grande différence entre les lois de probabilités des régions des images à compresser. Néanmoins, pour des images ayant une loi de probabilités constante pour toutes leurs régions, comme c'est le cas pour les images washsat et mandrill, l'apport est relativement faible et il est respectivement égal à 0.321% et 4.996%.

Il est à noter que le codage du bloc b_i de l'image est réalisé en utilisant la table des fréquences calculée par les blocs précédents. Dans ce cas, le CAA-0 suppose que cette dernière table est une approximation des fréquences des pixels du bloc b_i, et par conséquent elle présente les fréquences initiales des pixels à coder. En réalité, ceci n'est pas toujours vrai, puisque les blocs consécutifs d'une image n'ont pas forcément des lois de probabilités similaires. De ce fait, nous proposons dans la section suivante une nouvelle méthode de compression, qui consiste à découper l'image en blocs et de les ordonner de sorte que les lois de probabilités des blocs consécutifs soient le plus proche possible.

4.3.5 Tri des blocs avec la distance de Kullback

L'idéal, pour un schéma de codage basé sur le CA, est de coder chaque bloc de l'image avec sa vraie table de probabilités. Cependant, nous avons besoin dans ce cas d'insérer toutes les tables de probabilités dans l'en-tête du fichier compressé, ce qui engendre l'ajout d'une importante quantité d'information à ce dernier.

Le tableau 4.6 présente les résultats de compression de l'image Lena, découpée en blocs de taille 32×32 pixels et 16×16 pixels, avec et sans l'ajout des tables de probabilités

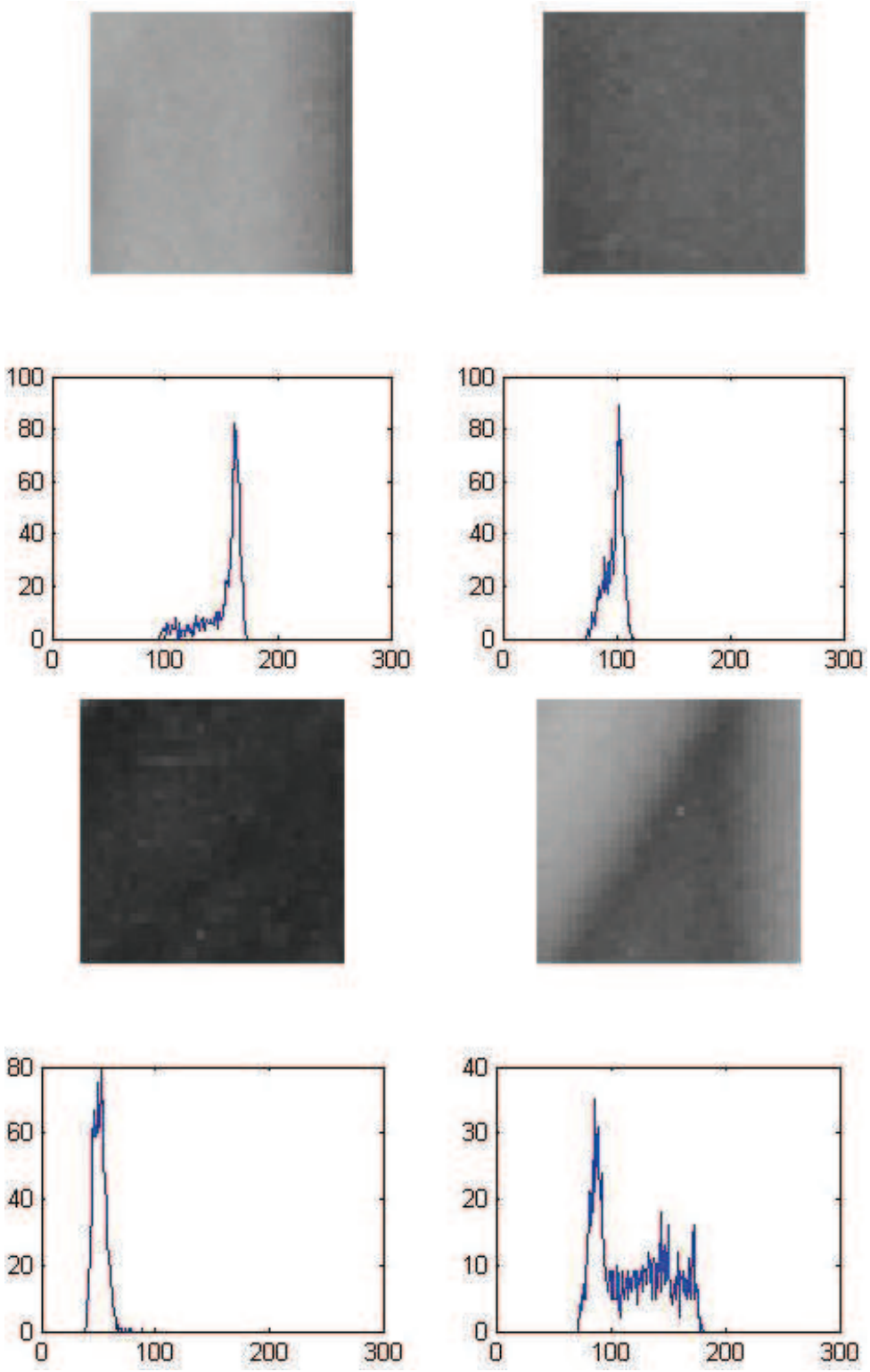


FIGURE 4.5 – Exemple de deux séries de deux blocs consécutifs de l'image Lena et leurs histogrammes.

TABLE 4.5 – L'effet de la précision f sur les performances du CAA-0 en codant les images bloc par bloc.

image	Valeur f en bits											Taux maximal	Gain maximal
	18	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**			
washsat	2.780	2.782	2.786	2.788	2.786	**2.789**	2.779	2.764	2.723	2.624		2.789	0.321
zelda	1.100	1.102	1.105	1.112	1.123	1.135	1.150	1.171	1.200	**1.224**		1.224	10.115
lena	1.074	1.077	1.081	1.088	1.094	1.105	1.121	1.153	1.197	**1.245**		1.245	13.804
goldhill	1.069	1.074	1.089	1.113	1.138	1.163	1.186	1.211	1.233	**1.244**		1.244	14.089
mandrill	1.086	1.087	1.091	1.095	1.099	1.103	1.110	1.120	1.134	**1.143**		1.143	4.996
mountain	1.287	1.313	1.348	1.389	1.426	1.453	1.480	1.513	1.541	**1.543**		1.543	16.632
peppers	1.056	1.059	1.063	1.070	1.079	1.090	1.109	1.142	1.189	**1.234**		1.234	14.424
barbara	1.047	1.051	1.054	1.062	1.069	1.077	1.087	1.109	1.139	**1.167**		1.167	10.229
boat	1.122	1.140	1.156	1.185	1.212	1.240	1.264	1.293	1.323	**1.339**		1.339	16.178
France	1.292	1.346	1.432	1.559	1.731	1.911	2.062	2.159	**2.186**	2.132		2.186	40.891
library	1.366	1.372	1.385	1.402	1.432	1.471	1.515	1.574	1.637	**1.680**		1.680	18.714
frog	1.608	1.625	1.650	1.688	1.732	1.774	1.826	1.906	1.995	**2.035**		2.035	20.953
moyenne	1.324	1.336	1.353	1.379	1.410	1.442	1.474	1.510	1.541	**1.551**		1.551	14.636

TABLE 4.6 – Les taux de compression obtenus en découpant l'image Lena en blocs de pixels, et en codant chaque bloc avec sa vraie table de probabilités.

	Taux de compression de l'image originale Lena	
	Sans prise en compte de la taille de l'en-tête	Avec prise en compte de la taille de l'en-tête
Découpage en blocs de taille 32×32	**1.420**	1.210
Découpage en blocs de taille 16×16	**1.595**	1.139

dans le fichier compressé. D'après ce tableau, nous remarquons que le taux de compression, obtenu en codant chaque bloc de l'image Lena avec sa vraie table de probabilités, est très élevé de l'ordre de 1.595 pour des blocs de taille 16×16 pixels. En partant de ce principe, nous avons pensé d'organiser les blocs de l'image de sorte que les distributions de deux blocs consécutifs, b_{i-1} et b_i, soient très proches. Par conséquent, la distribution du bloc b_{i-1} sera une bonne approximation de celle du bloc b_i et il devient très approprié d'utiliser cette distribution pour le codage de ce dernier bloc. Grâce à ce principe, nous pouvons coder chaque bloc avec une bonne approximation de sa vraie distribution sans avoir besoin d'ajouter des données additionnelles au fichier compressé.

Dans cette section, nous définissons un critère de similarité entre deux blocs b_1 et b_2 d'une même image afin de détecter des similarités entre les lois de probabilités des deux blocs. Nous proposons une mesure issue de la théorie de l'information qui est la distance de Kullback [Kullback 1951, Kullback 1959, Kullback 1968, Johnson 2000] entre les distributions de deux blocs. Rappelons qu'en 1945, Shannon a introduit la notion d'entropie qui est définie comme la moyenne de la quantité d'information apportée par les symboles générés à partir d'une source S. Cette notion a eu un très grand usage dans le codage et la compression des données.

En 1959, Kullback [Kullback 1959] a introduit une mesure de l'information relative à une loi de probabilité par rapport à une autre. Cette nouvelle mesure a également été considérée comme une mesure de distance entre ces deux lois. La distance de Kullback

entre deux lois de probabilités p_1 et p_2 est calculée selon l'équation suivante :

$$D(p_1, p_2) = -\sum_{i=0}^{k-1} p_1(i) \log_2(\frac{p_1(i)}{p_2(i)}). \tag{4.24}$$

$D(p_1, p_2)$ est une mesure de ressemblance entre les distributions p_1 et p_2. Cette mesure a été exploitée dans le domaine de l'imagerie et plus particulièrement dans le domaine de la segmentation d'images. La distance de Kullback n'est pas symétrique, et par conséquent nous définissons une nouvelle mesure symétrique qui mesure la différence entre p_1 et p_2 comme suit :

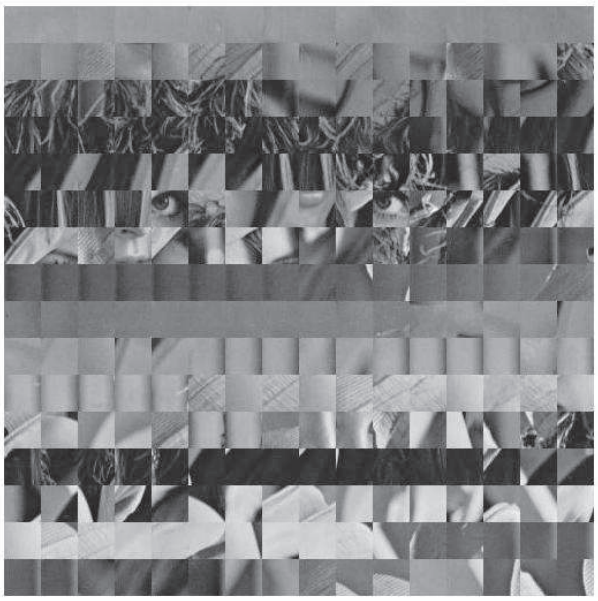
$$J(p_1, p_2) = \frac{D(p_1, p_2) + D(p_2, p_1)}{2}. \tag{4.25}$$

Nous proposons dans cette section un nouvel algorithme de CA qui améliore le CAA-0 pour la compression sans perte d'images. L'amélioration apportée consiste à modifier la précision f utilisée pour représenter le modèle adaptatif adopté par le CAA-0. Ainsi, nous proposons de réduire la valeur de f, afin de permettre au CAA-0 de s'adapter rapidement aux variations des statistiques des régions d'une image. De plus, l'image numérique est généralement caractérisée par des régions ayant une forte similarité spatiale et avec des distributions très similaires. De ce fait, nous proposons de couper l'image en des blocs (qui représentent les régions de l'image), $I = (b_1, \cdots, b_n)$ et de les ordonner en $(\hat{b_1}, \cdots, \hat{b_n})$ de sorte que la distance entre les distributions des blocs consécutifs soit la plus petite possible. $J(\hat{b_i}, \hat{b_{i+1}}) = min(J(\hat{b_i}, \hat{b_j}) \ \forall j > i$. Les figures 4.6 et 4.7 présentent les images Lena et peppers découpées en blocs de tailles 32×32 pixels et 16×16 pixels respectivement, et ordonnées selon le calcul basé sur la distance de Kullback. Il est à signaler qu'il faut transmettre au décodeur les positions initiales des blocs pour que ce dernier puisse reconstruire l'image originale après avoir reçu tous les blocs.

Ainsi, grâce à ce principe et en utilisant une précision f réduite pour représenter les probabilités estimées pour le modèle statistique, le CAA-0 devient plus performant et il fonctionne beaucoup mieux en comparaison à l'utilisation d'une précision f assez grande et au codage ligne par ligne de l'image. De plus, nous proposons d'utiliser l'algorithme 5 pour le redimensionnement de la table des fréquences au lieu de l'algorithme 4 puisque les blocs consécutifs, ordonnés selon la distance de Kullback, ont des lois très similaires. Par conséquent, nous proposons de coder chaque bloc b_i avec la vraie table des fréquences du bloc précédent b_{i-1}. Ce principe est très bénéfique et nous a permis de trouver des résultats très satisfaisants.

Ce principe, simple à mettre en œuvre, permet d'améliorer le CAA-0, selon les résultats

(a)

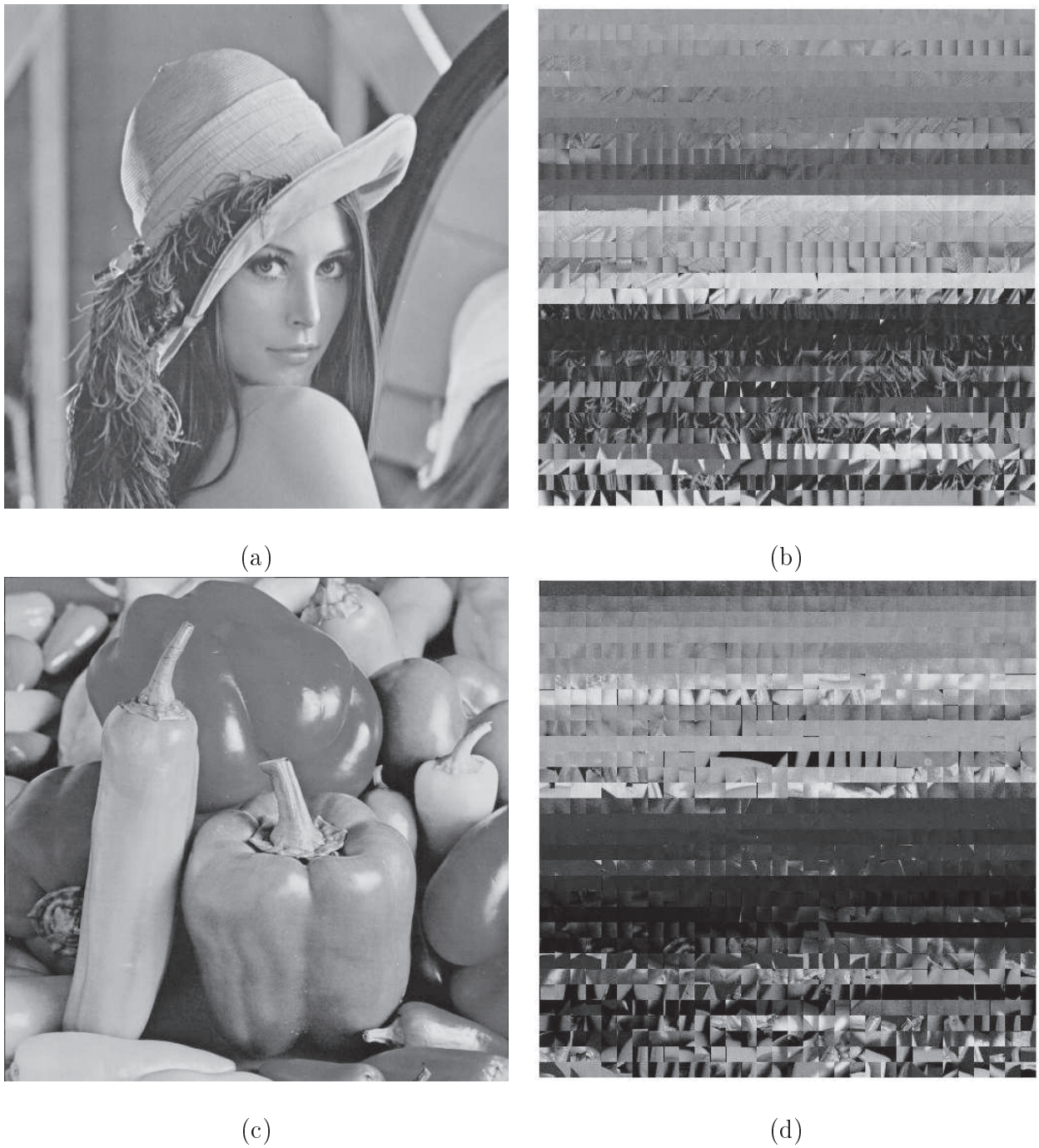


(b)

(c)

(d)

FIGURE 4.6 – a) Image originale Lena, b) Tri des blocs de taille 32×32 pixels de l'image Lena selon le calcul basé sur la distance de Kullback, c) Image originale Peppers, d) Tri des blocs de taille 32×32 pixels de l'image Peppers selon le calcul basé sur la distance de Kullback.



(a) (b)

(c) (d)

FIGURE 4.7 – a) Image originale Lena, b) Tri des blocs de taille 16×16 pixels de l'image Lena selon le calcul basé sur la distance de Kullback, c) Image originale Peppers, d) Tri des blocs de taille 16×16 pixels de l'image Peppers selon le calcul basé sur la distance de Kullback.

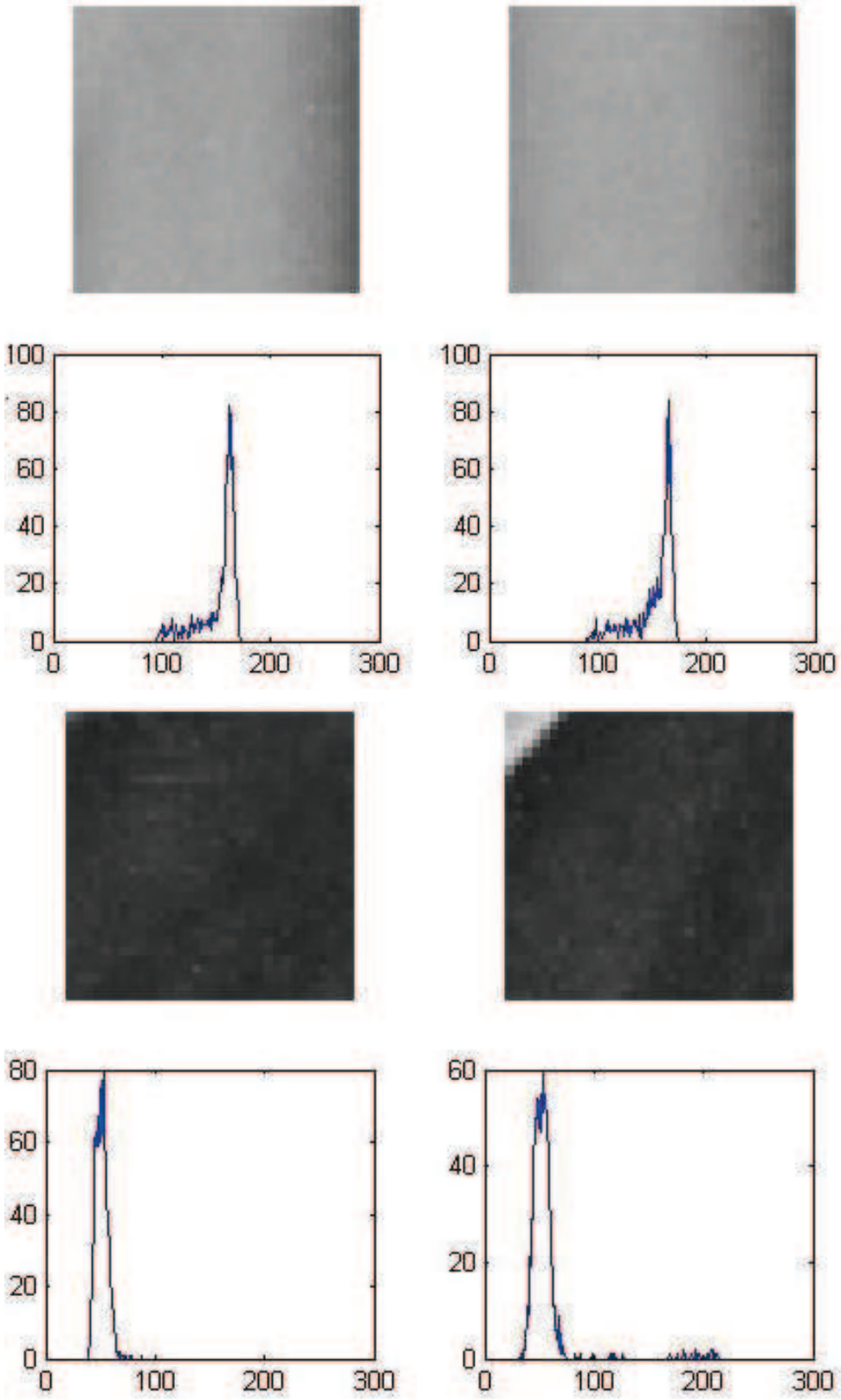


FIGURE 4.8 – Exemple de deux séries de deux blocs consécutifs de l'image Lena et leurs histogrammes. Les blocs sont obtenus après avoir découpé l'image Lena en des blocs de taille 32×32 pixels et de les triés selon la distance de Kullback.

Algorithme 5

```
1: si total_freq ≥ Max_frequence alors
2:    total_freq ← 0
3:    pour i ← 0 to nbr_symbole faire
4:       Freq[i] ← Freq[i] − Freq0[i] + 1
5:       Freq0[i] ← Freq[i]
6:       total_Freq ← total_Freq + Freq[i]
```

présentés tableau 4.7, d'un facteur moyen qui est de l'ordre de 18% pour des blocs de taille 32×32 pixels et qui dépasse 22% pour des blocs de taille 16×16. En effet, grâce à cette nouvelle approche, le CAA-0 devient plus performant et permet d'avoir des taux de compression nettement supérieurs à la plupart des implémentations du CAA-0. Notons également que d'après le tableau 4.7, les meilleurs résultats sont obtenus pour les images frog, library et france avec des gains très importants et compris entre 24% et 52%.

4.3.6 Conclusion

Dans la section 4.3, nous avons présenté un nouveau CAA-0 pour la compression sans perte d'images. Le CAA-0 proposé utilise une précision réduite pour représenter les fréquences des pixels à coder et il code l'image bloc par bloc après les avoir triés selon la distance de Kullback. Ce nouveau codeur permet d'améliorer les taux de compression obtenus par le CAA-0 d'un facteur moyen de l'ordre de 22% sans augmenter la complexité de ce dernier. Ce codeur est basé sur le fait que la distribution de probabilités à l'intérieur d'un bloc d'une image est souvent constante, toutefois, elle est généralement très différente en passant d'un bloc à un autre. Cette méthode est particulièrement appropriée pour les images caractérisées par des régions ayant des distributions très éloignées.

4.4 Conclusion

Dans ce chapitre, nous avons présenté deux nouvelles approches de compression sans perte d'images. La première approche [Masmoudi 2010a] est appliquée dans le domaine spatial et est basée sur un CAA qui met à jour les probabilités après le codage de la dernière occurrence de chaque pixel. L'approche proposée est plus efficace que les codeurs arithmétiques conventionnels en terme de taux de compression et est plus rapide que le

TABLE 4.7 – L'effet de la précision f sur les performances du CAA-0 en codant les images bloc par bloc après les avoir triés selon le calcul basé sur la distance de Kullback.

image	Découpage en blocs de tailles 32×32 pixels					Découpage en blocs de tailles 16×16 pixels				
	11	**10**	**9**	**Taux maximal**	**Gain maximal**	**11**	**10**	**9**	**Taux maximal**	**Gain maximal**
washsat	2.631	2.826	**2.903**	2.903	4.259	**2.963**	2.920	2.780	2.963	6.197
zelda	1.245	**1.276**	1.265	1.276	13.807	1.346	**1.365**	1.326	1.365	19.405
lena	1.281	**1.311**	1.291	1.311	18.113	1.383	1.378	**1.383**	1.383	22.350
goldhill	1.250	**1.269**	1.260	1.269	15.783	1.284	**1.305**	1.302	1.305	18.065
mandrill	1.153	1.173	**1.175**	1.175	7.583	1.187	**1.192**	1.183	1.192	8.877
mountain	1.545	**1.575**	1.564	1.575	18.399	1.575	**1.603**	1.597	1.603	19.818
peppers	1.259	**1.282**	1.260	1.282	17.625	1.325	1.361	**1.367**	1.367	22.748
barbara	1.185	**1.199**	1.183	1.199	12.647	1.221	1.250	**1.258**	1.258	16.751
boat	1.343	**1.356**	1.336	1.356	17.242	1.375	1.412	**1.415**	1.415	20.694
France	2.137	2.230	**2.234**	2.234	42.923	2.600	**2.662**	2.581	2.662	52.116
library	1.691	**1.720**	1.676	1.720	20.630	1.733	1.794	**1.801**	1.801	24.187
frog	2.080	**2.143**	2.113	2.143	24.842	2.081	**2.153**	2.136	2.153	25.174
moyenne	1.567	**1.613**	1.605	1.613	18.033	1.673	**1.700**	1.677	1.700	22.187

CAA-0.

La deuxième approche a étudié, en premier lieu, l'effet de la précision de représentation des fréquences des pixels sur les performances du CAA-0, et a proposé, en deuxième lieu, de coder l'image bloc par bloc après les avoir triés selon la distance de Kullback. Ces deux approches s'orientent vers le développement des nouveaux modèles statistiques extrêmement appropriés pour le codage d'images, d'une part, et l'ajout d'étapes de prétraitement sur l'ensemble des pixels afin d'exploiter au maximum l'efficacité des modèles proposés, d'autre part.

Chapitre 5

Contribution à la crypto-compression d'images

Sommaire

5.1 Introduction

Dans ce chapitre, nous présentons nos contributions au chiffrement et à la crypto-compression d'images. En premier lieu, nous proposons, section 5.2, un nouveau générateur de nombres pseudo-aléatoires (GNPA). Nous comparons notre GNPA aux générateurs déjà présentés par d'autres auteurs, et nous détaillons les analyses effectuées pour prouver l'efficacité de notre GNPA en nous basant sur la série de tests de NIST. Ensuite, nous présentons un nouvel algorithme de chiffrement par flot, section 5.3, tout en exploitant notre GNPA. L'analyse de la sécurité et de la robustesse contre plusieurs types d'attaques est ensuite détaillée. Puis, nous présentons, section 5.4, un nouveau schéma de crypto-compression basé sur la combinaison d'un codage arithmétique binaire avec notre GNPA, et nous détaillons les résultats obtenus sur des images pour le modèle statique et le modèle adaptatif. Enfin, nous terminons ce chapitre par une conclusion, section 5.2.7.

5.2 Elaboration d'un nouveau Générateur pseudo-aléatoire

5.2.1 Introduction

C'est à partir de 1990 [Pecora 1990], que les systèmes chaotiques (SC) ont commencé à intéresser des chercheurs en proposant de les utiliser dans la conception des systèmes cryptographiques modernes. En effet, les SC, sont capables de générer des séquences chaotiques appelées pseudo-aléatoires, qui sont caractérisées par des propriétés cryptographiques et statistiques très importantes. De ce fait, le chaos a été utilisé en cryptographie et essentiellement dans le chiffrement par flot. Avec ce type de chiffrement, il faut avoir un mécanisme capable de générer d'une manière dynamique et déterministe la liste des clés dynamiques $\{k_j\}_{j=1}^{N}$ utilisée pour chiffrer un texte clair M composé de N symboles. En cryptographie, la séquence des clés K_j doit nécessairement satisfaire certaines propriétés comme l'aléatoire et l'imprédictibilité comme présenté chapitre 3. Ainsi, pour répondre à ces besoins, nous utilisons des générateurs de nombres pseudo-aléatoires GNPA. Ce type de générateur est capable, à partir d'une séquence initiale de bits (valeurs initiales), de produire de manière déterministe une séquence de bits de taille très importante qui est pseudo-aléatoire et imprédictible quand les valeurs initiales ne sont pas connues. Le déterminisme, noté également par la reproductibilité, s'explique par le fait que le GNPA est

capable de reproduire la même séquence pour les mêmes valeurs initiales. Cette dernière propriété est très importante dans la génération des séquences pseudo-aléatoires et dans la conception des algorithmes de chiffrement par flot puisqu'elle assure la synchronisation des séquences de clés entre l'émetteur et le récepteur.

Le principe d'un système de chiffrement par flot basé sur le chaos consiste à mélanger un texte clair avec la séquence pseudo-aléatoire générée par un SC. Les données chiffrées seront par conséquent transmises au récepteur, qui sera capable d'extraire l'information initiale à partir du texte chiffré et des valeurs initiales choisies par l'émetteur lui permettant de générer la même séquence chaotique utilisée durant le processus de chiffrement. Dans la littérature, il existe plusieurs techniques basées sur le chaos qui visent à générer des séquences pseudo-aléatoires pour être utilisées dans le chiffrement par flot [Li 2001a, Yang 2004, Wong 2008, Wu 2004, Zhang 2005].

La plupart des SC proposés dans la littérature sont caractérisés par certaines propriétés qui favorisent leurs utilisations dans la conception des systèmes cryptographiques modernes. Ces propriétés sont essentiellement l'ergodicité, la sensibilité aux valeurs initiales et la sensibilité aux paramètres de contrôles [Kocarev 2001]. Cependant, ll est à noter qu'il n'est pas très approprié d'utiliser certains SC en cryptographie à cause de leurs densités, qui ne sont pas forcément uniformes, et de leurs espaces de clés, qui sont relativement petits, sachant que la clé de cryptage/décryptage d'un cryptosystème basé sur le chaos est constitué à partir des valeurs initiales et des paramètres de contrôles. Ainsi pour résoudre ces problèmes, certaines techniques sont basées sur la combinaison de plusieurs SC [Patidar 2009b, Li 2001b, Wanga 2009], d'autres sont basées sur l'utilisation des SC de haute dimensions [García 2002, Li 2009, Chen 2004] ou bien sur l'itération du même SC un certain nombre de fois [Zhou 1997]. Il existe également plusieurs autres techniques qui visent à améliorer les cryptosystèmes basés sur le chaos [Li 2007, Fallahi 2008]. Ainsi, la conception d'un cryptosystème basé uniquement sur un SC n'est pas forcément efficace et robuste contre certaines attaques, comme l'attaque par analyse statistique [Short 1997, Li 2007, Yang 1998]. C'est dans ce cadre que s'inscrit notre travail qui consiste à concevoir un nouveau GNPA basé sur la combinaison d'un SC bidimensionnel, ayant un espace de clé largement suffisant pour être robuste à l'attaque par force brute, et d'une fonction qui convertit la séquence des valeurs générées par le SC en une nouvelle séquence caractérisée par une densité de probabilité uniforme et ayant d'autres propriétés statistiques très importantes en cryptographie. Le SC utilisé est le Chaotic Standard Map (CSM). Ce dernier est couplé avec la fonction de développement en fraction conti-

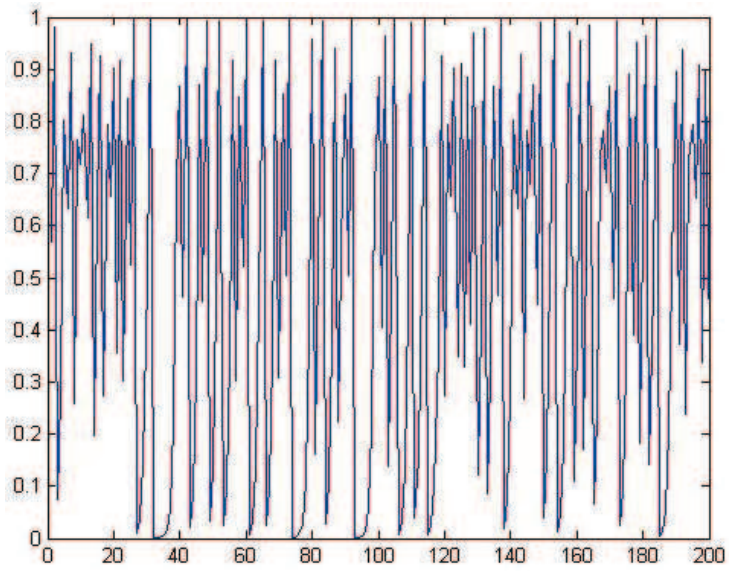


FIGURE 5.1 – La trajectoire d'une séquence de 200 valeurs générées par la LM avec $x_0 = 0.56923148$ et $\lambda = 3.996$.

nue d'Engel (ECF-map) pour générer des flux de clés dynamiques avec des propriétés chaotiques et statistiques très importantes et très appropriées en cryptographie.

Nous consacrons la section 5.2.2 pour présenter brièvement la construction des GNPA basés sur les systèmes chaotiques. Les sections 5.2.3 et 5.2.4 présentent respectivement une introdution aux fractions continues et au système chaotique CSM. Nous détaillons ensuite, section 5.2.5, la procédure proposée pour la génération des séquences aléatoires. Nous examinons, section 5.2.6, la qualité de notre GNPA en appliquant des analyses statistiques sur les séquences générées. Enfin, nous finissons par une conclusion, section 5.2.7.

5.2.2 Description des GNPA basés sur les systèmes chaotiques

Dans la cryptographie moderne, les SC ont été largement exploités dans le développement des cryptosystèmes et des GNPA. Parmi les SC les plus connus et les plus utilisés en cryptographie, on cite la Logistic Map (LM). La LM est un SC unidimensionnel et est défini :

$$x_{n+1} = \lambda x_n(1 - x_n), \tag{5.1}$$

avec x_0 est la valeur initiale, $x_n \in (0, 1)$ $\forall n$ et λ est le paramètre de contrôle et $\in (0, 4]$. La figure 5.1 présente la trajectoire de la suite x_n pour $x_0 = 0.56923148$ et $\lambda = 3.996$.

Il est à noter que la LM est extrêmement sensible à la valeur initiale x_0 et les valeurs

obtenues par itérations successives sont imprédictibles. Grâce à ces propriétés, la LM a été utilisé pour la conception de plusieurs crypto-systèmes. En effet, Mi et *al.* [Mi 2008] ont proposé un GNPA qui est basé sur la LM, pour la cryptocompression des données. De plus, Kanso et Smaoui [Kanso 2009] ont proposé également deux générateurs de bits pseudo-aléatoire en utilisant la LM. Il est à noter qu'il existe d'autres crypto-systèmes qui sont basés sur la LM. Toutefois, les dernières recherches dans le domaine de la cryptanalyse ont montré que la LM n'est pas très approprié pour être utilisé en cryptographie. La clé de cryptage pour un cryptosystème basé sur la LM est constituée à partir de la valeur initiale x_0 et du paramètre de contrôle λ, et toutes les deux sont des nombres réels à virgule flottante. Pour une représentation en double précision, les nombres flottants sont codés sur 52 bits pour assurer une représentation de 15 chiffres après la virgule. Dans ce cas, l'espace des clés qui est égal à 2^{104} n'est pas suffisamment robuste pour une attaque à force brute. De plus, plusieurs études ont montré que pour avoir un comportement chaotique du LM, il faut que λ soit supérieur à 3.57, ce qui réduit l'espace de clé. Le choix d'une valeur supérieure à 3.57 est une condition nécessaire mais elle n'est pas suffisante pour générer des séquences chaotiques. Pour avoir un comportement parfaitement chaotique du LM, il faut que la valeur λ soit égale à 4. Cependant, l'espace de clés dans ce cas sera très petit et égal à 2^{52}, ce qui n'est pas robuste à l'attaque par force brute.

Il existe un autre problème, lié à l'utilisation du LM en cryptographie, c'est celui de la génération des séquences périodiques pour certaines valeurs de λ. En ce qui concerne la distribution de probabilités des valeurs générées par la LM, nous pouvons remarquer qu'elle n'est pas uniforme dans $[0, 1]$ [Alvarez 2003, Alvarez 2009]. La figure 5.2 montre les distributions de probabilités pour des séquences de $100,000$ valeurs générées par la LM avec $x_0 = 0.56923148$ et pour différentes valeurs de λ. Ces inconvénients défavorisent l'utilisation du LM en cryptographie.

La figure 5.2 montre que la distribution des probabilités des valeurs générées par la LM se rapproche d'une distribution uniforme lorsque λ se rapproche de 4. Nous proposons de convertir les séquences de valeurs générées par la LM en des séquences binaires en utilisant la fonction $F : [0, 1] \rightarrow [0, 1]$ donnée par :

$$F(x) = \begin{cases} 0 & si \quad x < 0.5 \\ 1 & sinon \end{cases} . \tag{5.2}$$

Ainsi, nous avons utilisé cette fonction pour présenter, figure 5.3, les pourcentages de

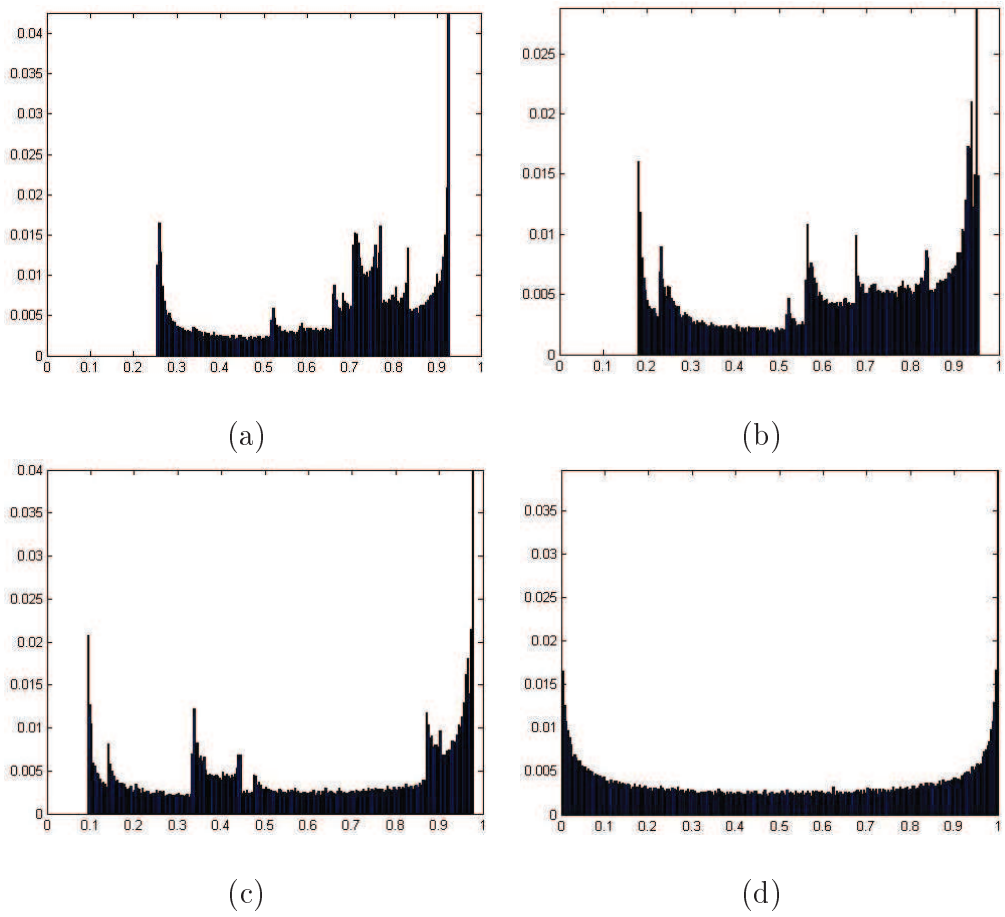


FIGURE 5.2 – Distributions de probabilités pour des séquences de 100, 000 valeurs générées par la LM avec $x_0 = 0.56923148$ et a) $\lambda = 3.7$, b) $\lambda = 3.8$, c) $\lambda = 3.9$, d) $\lambda = 4$.

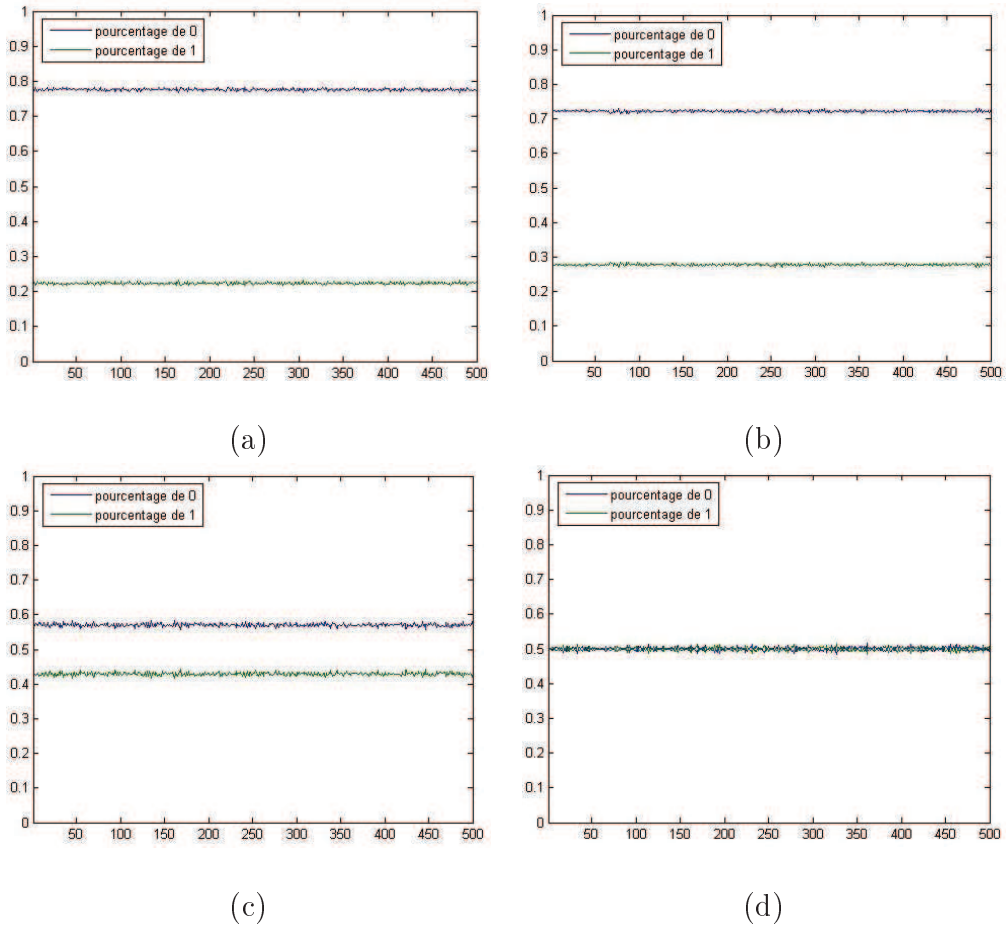


FIGURE 5.3 – Les pourcentages de 0 et de 1 pour 500 séquences générées avec la LM, chaque séquence contient 100000 valeurs avec $x_0 = 0.56923148$ et a) et $\lambda = 3.7$, b) $\lambda = 3.8$, c) $\lambda = 3.9$, d) $\lambda = 4$.

0 et de 1 pour 500 séquences générées par la LM dont chacune contient 100000 valeurs. La figure 5.3 justifie le choix de la valeur 4 pour le paramètre λ puisque les pourcentages de 0 et de 1 sont presque égaux à 0.5 et par conséquent les séquences binaires sont uniformément distribuées dans $[0, 1]$.

Dans la section 5.2.6, nous présentons également dans les tableaux 5.1 et 5.2 une analyse statistique basée sur la série de tests de NIST, présentée chapitre 3, appliquée sur des séquences générées par la LM avec différentes clés. D'après les résultats présentés dans ces tableaux, nous avons conclu que ces séquences ne sont pas pseudo-aléatoires. En plus du LM, il existe plusieurs autres SC qui sont utilisés pour la conception des GNPA. Par exemple, Patidar et Sud [Patidar 2009b] ont proposé un GNPA pour le cryptage par flot des données qui est basé sur l'utilisation du système chaotique CSM. Le schéma présenté

a été évalué et vérifié en utilisant les tests statistiques proposés par NIST [Rukhin 2008] et DIEHARD [Marsaglia 1997]. Cependant, d'après l'analyse statistique présentée dans les tableaux 5.3 et 5.4, section 5.2.6, basée sur la série de tests de NIST, nous avons remarqué que le générateur présenté par [Patidar 2009b] ne réussit pas tous les tests et par conséquent nous pouvons en déduire qu'il n'est pas totalement pseudo-aléatoire.

5.2.3 Les fractions continues

Une fraction continue [Lorentzen 1992, Seidensticker 1983, Wall 1973, Vuillemin 1987] est une expression qui s'écrit sous la forme :

$$x = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \cfrac{a_4}{\cdots}}}}, \tag{5.3}$$

avec $a_i (i > 0)$ les numérateurs partiels, b_i les dénominateurs partiels et b_0 est la partie entière du nombre réel x [Khintchin 1963]. Notons que pour $x \in [0,1]$, b_0 est égal à 0. Hartono et *al.* [Hartono 2002] ont introduit une nouvelle fraction, notée par fraction continue d'Engel (FCE), cette dernière est calculé à partir de la fonction $T_E : [0,1] \rightarrow [0,1]$ donnée par :

$$T_E(x) = \begin{cases} \frac{1}{\lfloor \frac{1}{x} \rfloor}\left(\frac{1}{x} - \lfloor \frac{1}{x} \rfloor\right) & si \quad x \neq 0 \\ 0 \qquad\qquad sinon & . \end{cases} \tag{5.4}$$

En effet, soit $x \in [0,1]$, la fonction T_E génère une unique et finie fraction continue sous la forme :

$$x = \cfrac{1}{b_1 + \cfrac{b_1}{b_2 + \cfrac{b_2}{b_3 + \ddots + \cfrac{b_{n-1}}{b_n + \ddots}}}}, \quad b_n \in \mathbb{N}, \quad b_n \leq b_{n+1} \tag{5.5}$$

Nous définissons :

$$\begin{aligned} b_1 &= b_1(x) = \lfloor \tfrac{1}{x} \rfloor \\ b_n &= b_n(x) = b_1(T_E^{n-1}(x)), \quad n \geq 2, \quad T_E^{n-1}(x) \neq 0, \end{aligned} \tag{5.6}$$

avec $T_E^0(x) = x$ et $T_E^n(x) = T_E(T_E^{n-1}(x))$ pour $n \geq 1$. D'après la définition de la fonction T_E, nous pourrons déduire que tout nombre $x \in [0,1]$ peut s'écrire sous la forme :

$$\begin{aligned} x &= \frac{1}{b_1 + b_1 T_E(x)} \\ &= \cfrac{1}{b_1 + \cfrac{1}{b_2 + \cfrac{b_2}{b_3 + \ddots + \cfrac{b_{n-1}}{b_n + b_n T_E^n(x)}}}}. \end{aligned} \tag{5.7}$$

L'algorithme 6, décrit la méthode de génération de la représentation en FCE d'un nombre réel x.

Algorithme 6 Développement en FCE

Initialiser $x_0 \leftarrow x, i \leftarrow 0$

tant que $x_i \neq 0$ **faire**

$i \leftarrow i + 1$

$b_i \leftarrow \lfloor \frac{1}{x_{i-1}} \rfloor$

$x_i \leftarrow \frac{1}{\lfloor \frac{1}{x_{i-1}} \rfloor} (\frac{1}{x_{i-1}} - \lfloor \frac{1}{x_{i-1}} \rfloor)$

D'après le théorème présenté par [Hartono 2002], un nombre réel $x \in [0,1]$ possède une unique représentation finie en FCE (c'est-à-dire qu'il existe $n_0 \in N^*$, tel que $T_E^n(x) = 0\ , \forall n \geq n_0\ )$ si et seulement si $x \in Q$. D'après ce théorème, nous pouvons conclure que tout nombre réel représenté avec une précision finie, possède une unique représentation finie en FCE. Dans notre travail, nous portons un intérêt particulier à la séquence $Z_n(x)$ donnée par :

$$Z_n(x) = b_n(x) T_E^n(x),\ n \geq 1. \tag{5.8}$$

Ainsi, $\forall x \in [0,1]$ la séquence $\{Z_i(x)\}_{i=1}^n$ est finie et prend ses valeurs dans $[0,1]$. De plus, la séquence $Z_n(x)$ est caractérisée par des propriétés statistiques [Hartono 2002] très importantes qui nous a motivé pour l'utiliser dans le développement de notre nouveau GNPA.

5.2.4 Le système chaotique CSM

Le CSM est un système dynamique non-linéaire, initialement utilisé en physique et il est défini par :

$$\begin{cases} x_j = x_{j-1} + t \times sin(y_{j-1}) \\ y_j = y_{j-1} + x_j \end{cases}, \tag{5.9}$$

avec $x_n, y_n \in [0, 2\pi)$, et t est le paramètre de contrôle. La figure 5.4 montre l'évolution du CSM, avec $x_0 = 3.245$, $y_0 = 0.851$ et pour différentes valeurs du paramètre t. Nous constatons que le CSM possède une trajectoire qui devient de plus en plus dense dans sa région d'évolution en augmentant la valeur de t. De plus, nous remarquons que les deux suites x_n et y_n sont linéairement indépendantes pour des grandes valeurs du paramètre t.

En outre, la figure 5.5.a présente la trajectoire de la suite x_n, alors que la figure 5.5.b illustre l'évolution de la trajectoire de la suite y_n pour des valeurs initiales très proches. Ainsi, nous remarquons que pour deux conditions initiales très proches, le CSM conduit à générer des trajectoires très éloignées. C'est grâce à ces propriétés que le CSM est un excellent SC.

Il est à noter que dans [Chirikov 1971], il existe plus de détails sur l'origine du CSM sont présentés, de son utilisation dans d'autres disciplines comme la physique, et de son caractère chaotique.

5.2.5 Un nouveau générateur de nombres pseudo-aléatoire

Dans cette section, nous décrivons en détail notre nouveau GNPA. Dans notre approche, nous proposons de coupler le système chaotique CSM avec la fonction de développement en fractions continues d'Engel afin de générer des séquences ayant des propriétés chaotiques et statiques très intéressantes pour les utiliser en cryptographie. En effet, le système chaotique CSM est caractérisé par la sensibilité aux valeurs initiales, la sensibilité au paramètre de contrôle, l'ergodicité, la non-périodicité, l'aléatoire et l'imprédictibilité. De plus, la fonction de représentation en FCE est capable de générer à partir d'un réel x, une séquence de valeurs générées à partir de l'équation 5.8 possédant d'intéressantes propriétés statistiques.

La clé secrète K de notre GNPA est composée de trois nombres réels et d'un entier $K = (x_0, y_0, t, N_0)$, avec le couple $(x_0, y_0) \in [0, 2\pi)$ représentant les valeurs initiales du CSM, t est le paramètre de contrôle du CSM et N_0 est le nombre d'itération initiale du CSM. L'espace de clé est très suffisant pour que notre GNPA soit robuste contre une attaque par force brute dans la mesure où il dépasse 169 bits pour une précision de 10^{-15}. Dans la suite de cette section, nous proposons de détailler la procédure de génération

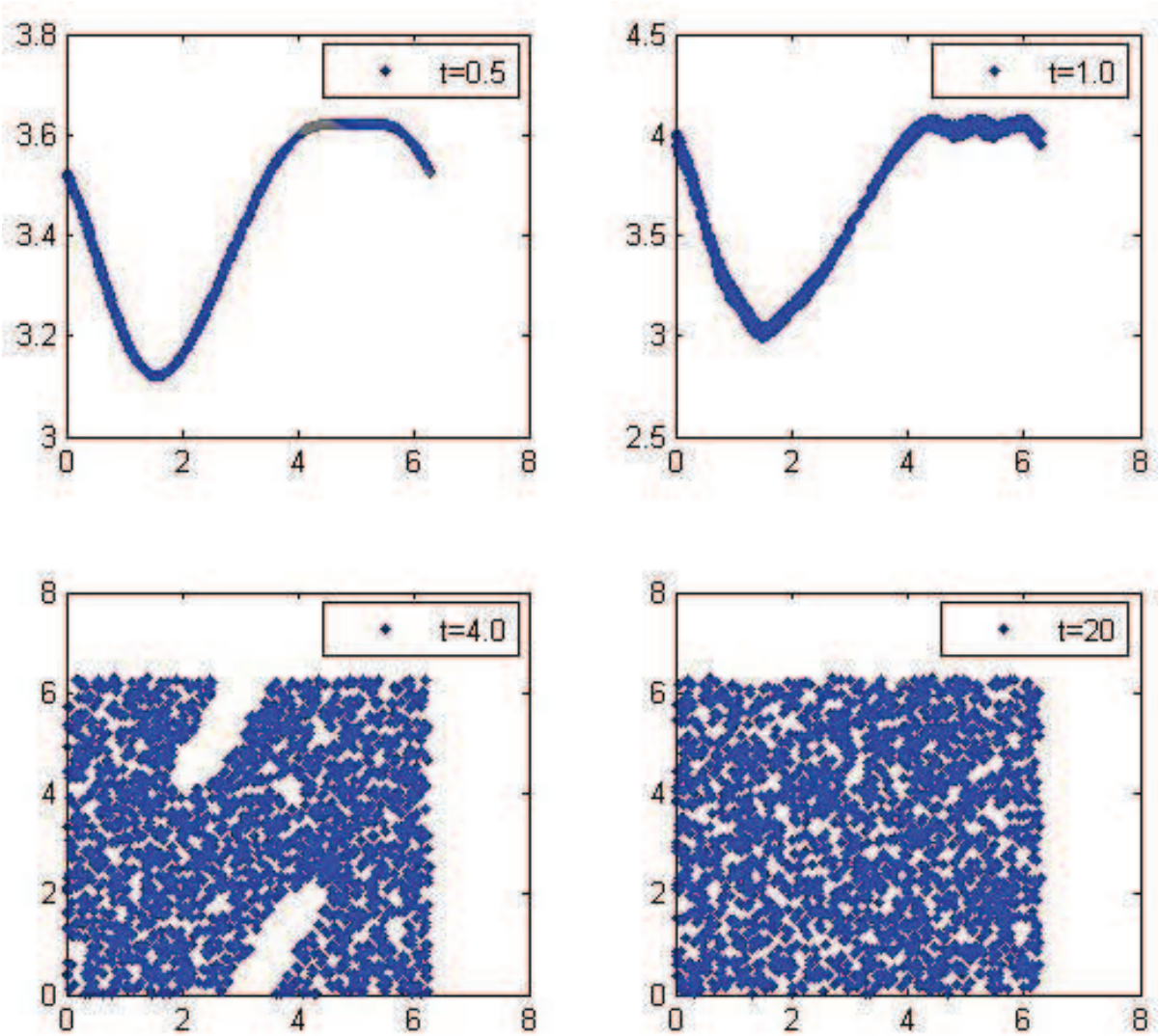
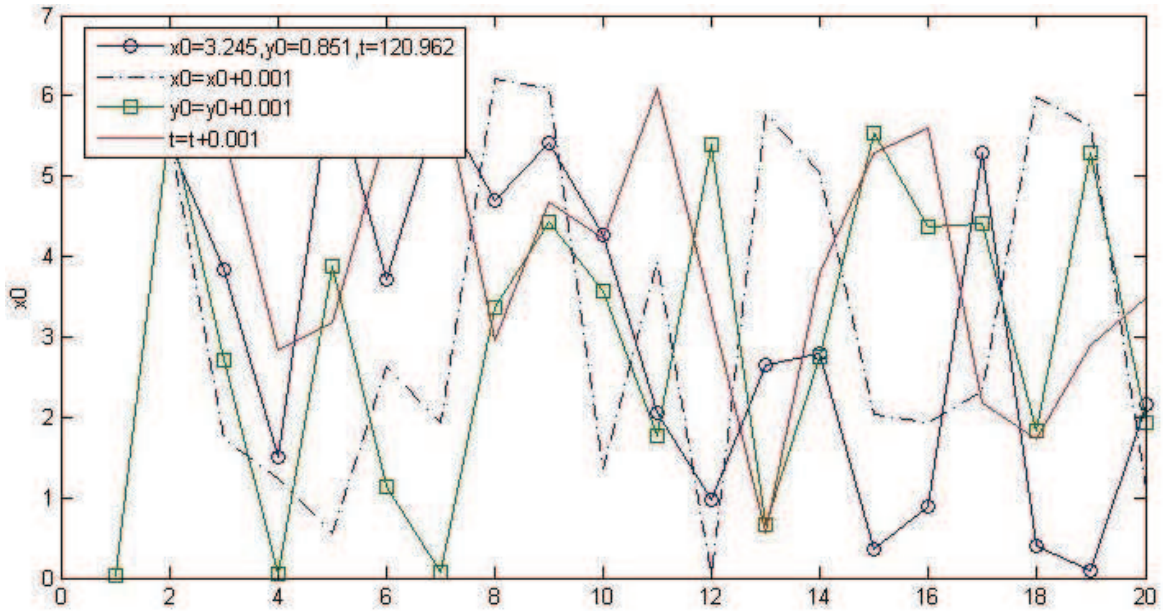
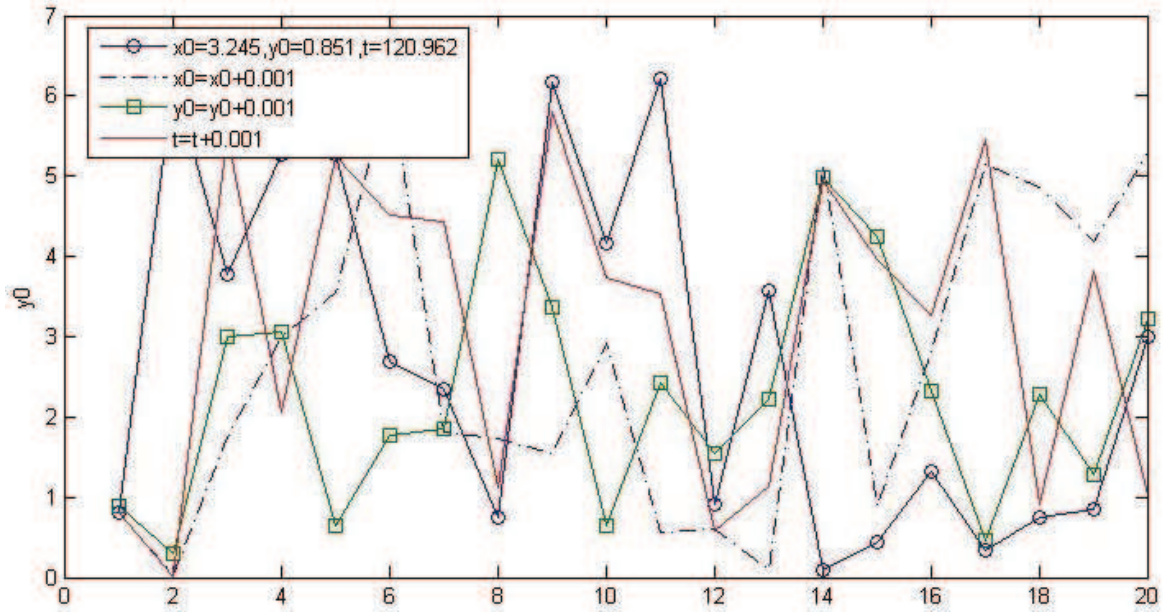


FIGURE 5.4 – L'évolution du CSM pour x0 = 3.245, y0 = 0.851 et avec différentes valeurs du paramètre t.



(a)



(b)

FIGURE 5.5 – Sensibilité du CSM aux valeurs initiales et au paramètre de contrôle, a) la trajectoire de la suite x_n pour des valeurs initiales très proches, b) la trajectoire de la suite y_n pour des valeurs initiales très proches.

des séquences de nombres pseudo-aléatoire. Nous définissons tout d'abord la fonction $G : [0, 1] \rightarrow [0, 1]$ telle que :

$$G(x) = \sum_j Z_j(x) - \lfloor \sum_j Z_j(x) \rfloor. \tag{5.10}$$

Initialement, nous proposons d'itérer le CSM N_0 fois et d'appliquer par la suite les opérations suivantes pour générer la séquence de bits finale (flux de clés dynamiques) :

- **1** Le CSM est itéré d'une manière continue, pour la $j^{\text{ème}}$ itération, la sortie du CSM est un nouveau couple $(x_j, y_j) \in [0, 2\pi)$.
- **2** Nous proposons de calculer la séquence $\{a_j\}_{j=1}^{N}$ en utilisant l'équation suivante :

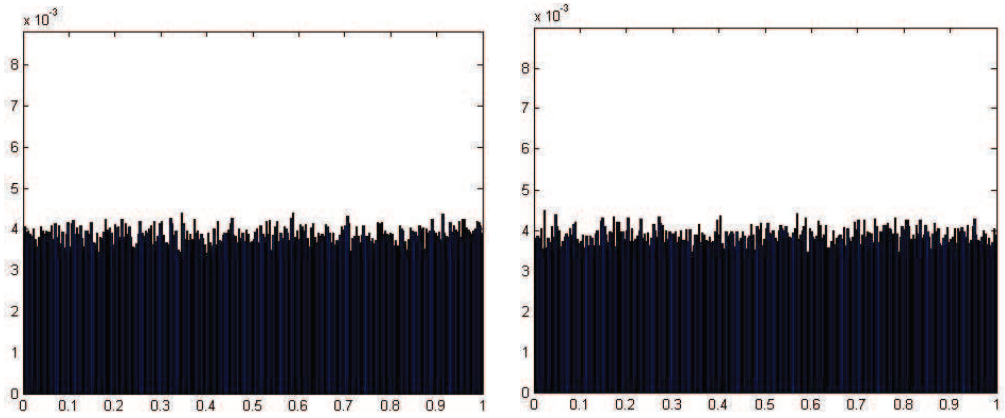
$$a_j = y_j - \lfloor y_j \rfloor. \tag{5.11}$$

- **3** Finalement, la séquence $\{k_j\}_{j=1}^{N}$ représentant le flux de clés dynamiques est donnée par :

$$k_j = F(G(a_j)). \tag{5.12}$$

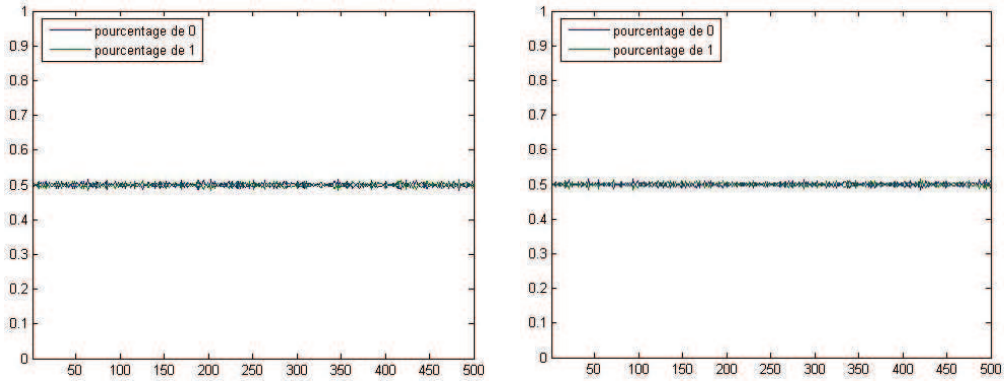
Le CSM et la fonction de représentation en FCE sont itérés plusieurs fois jusqu'à générer d'une manière dynamique et déterministe les flux des clés dynamiques à utiliser en cryptage/décryptage. La figure 5.6 présente les distributions pour 100000 valeurs générées en utilisant uniquement le CSM, figure 5.6.a, et combinant le CSM avec la fonction ECF-map, figure 5.6.b. Nous pouvons remarqués que les distributions calculées sont presque uniformes. De plus, la figure 5.7 présente les pourcentages de 0 et de 1 pour 500 séquences générées sans et avec l'utilisation de la fonction ECF-map. D'après cette figure, nous remarquons que les pourcentages sont presque égaux à 50%. En outre, la figure 5.8 montre la sensibilité de notre GNPA aux valeurs initiales et au paramètre de contrôle.

En effet, d'après les résultats présentés figures 5.6- 5.8, nous pouvons remarquer que le CSM possède une distribution uniforme et il est parfaitement sensible aux valeurs initiales et au paramètre de contrôle. Toutefois, ces études ne sont pas suffisantes pour montrer l'efficacité d'un GNPA. Ainsi, nous proposons, section 5.2.6, d'analyser les performances de notre GNPA tout en nous basant sur la série de tests statistiques proposée par NIST.



(a) (b)

FIGURE 5.6 – Les distributions de probabilités pour 100000 valeurs générées avec $x_0 = 3.59587469543$, $y_0 = 0.8512974635$ et $t = 120.9625487136$ en utilisant a) uniquement le CSM, b) le CSM et le ECF-map. Chaque séquence contient 100000 valeurs.



(a) (b)

FIGURE 5.7 – Les pourcentages de 0 et de 1 pour 500 séquences générées avec $x_0 = 3.59587469543$, $y_0 = 0.8512974635$ et $t = 120.9625487136$ en utilisant a) uniquement le CSM, b) le CSM et le ECF-map. Chaque séquence contient 100000 valeurs.

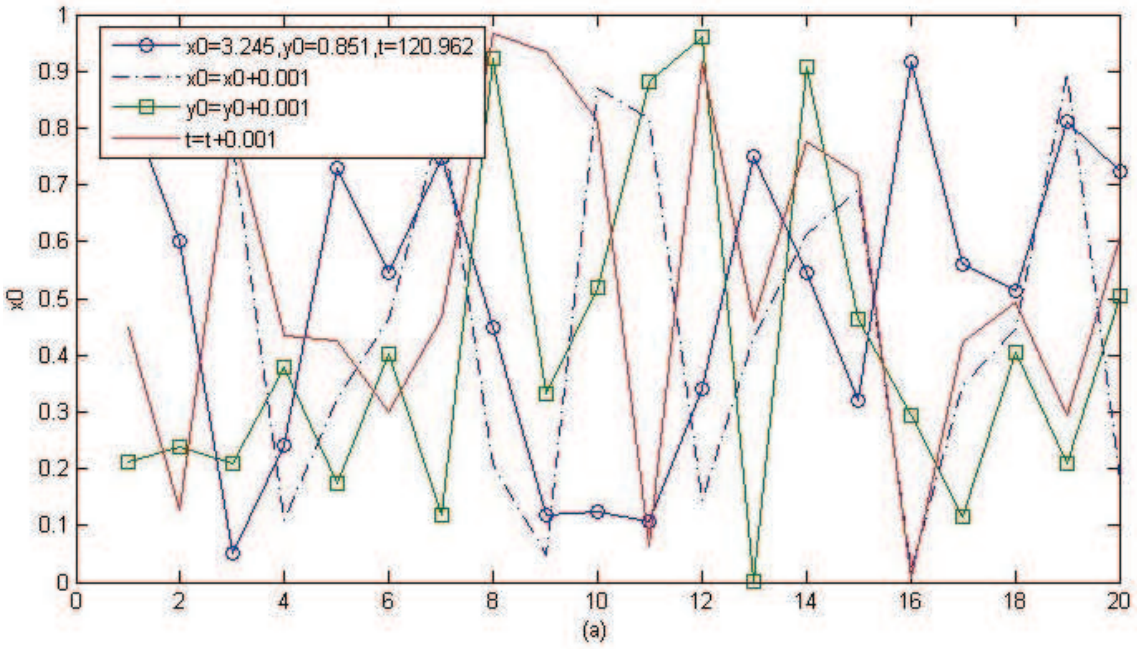


FIGURE 5.8 – Sensibilité de notre GNPA aux valeurs initiales et au paramètre de contrôle.

5.2.6 Analyse des performances du GNPA proposé

5.2.6.1 L'espace de clés

La clé de notre GNPA est composée de 3 nombres réels et un entier $K = (x_0, y_0, t, N_0)$, avec x_0, $y_0 \in [0, 2\pi)$, t peut être n'importe quel flottant supérieur à 18 et N_0 est un entier supérieur à 100. Nous proposons d'utiliser une précision de 15 chiffres après la virgule. Dans ce cas, le nombre de valeurs possibles pour x_0 est presque égal à 6.28×10^{15}. De même, le nombre de valeurs possibles pour y_0 est $\cong 6.28 \times 10^{15}$. Le paramètre t fait partie de la clé et il peut prendre n'importe quelle valeur supérieur à 18, validée expérimentalement [Patidar 2009a], et par conséquent le nombre de valeurs possibles de t est supérieur à 18×10^{15}. De plus, la valeur de N_0 peut être n'importe quel entier supérieur à 100. Néanmoins, le choix d'une valeur très grande pour N_0 diminue la vitesse de notre GNPA. Ainsi, nous proposons que N_0 puisse prendre n'importe quelle valeur de l'intervalle $[100, 1100]$ et par conséquent le nombre total de N_0 est égal à 10^3. Finalement, l'espace de clé de notre GNPA peut être supérieur à $6.28^2 \times 18 \times 10^{48} \cong 2^{169}$ ce qui est largement suffisant pour résister à n'importe quel type d'attaque par force brute.

5.2.6.2 Analyse statistique

La qualité d'un GNPA dépend de sa capacité de résister contre tout type d'attaque. Par exemple, un bon GNPA doit être robuste contre les attaques par prédiction du bit suivant. Ce type d'attaque vise à prédire la valeur d'un bit à partir de la connaissance des n bits qui le précèdent. Un bon GNPA doit être également robuste aux attaques par recouvrement des valeurs initiales, dont le but est de retrouver à partir d'une séquence aléatoire les valeurs initiales utilisées par le générateur. Il est à mentionner qu'il existe beaucoup d'autres type d'attaques et pour bien tester et vérifier la qualité d'un GNPA, il faut l'appliquer à plusieurs tests statistiques.

La série de tests du NIST est une librairie fournissant une implémentation d'une série de 16 tests statistiques et qui est détaillée dans la publication spéciale du NIST 802-22 [Rukhin 2008]. Les 16 tests peuvent être classés en deux catégories : les tests paramétriques et les tests non-paramétriques.

Les tests non paramétriques :

- Test de Fréquence (Frequency Test : FT),
- Test de suites homogènes (Runs Test : RT),
- Test sur la plus longue suite de un dans un bloc (Test for the Longest Run of Ones in a Block : LROT),
- Test de compression Lempel-Ziv (Lempel-Ziv Compression Test : LZT),
- Test sur le rang de la matrice binaire aléatoire (Binary Matrix Rank Test : MRT),
- Test de la somme cumulée (Cumulative Sums Test : CST),
- Test sur la transformée de Fourrier discrète (Spectral Test : SPT),
- Test d'excursions aléatoires (Random Excursion Test : RET),
- Variante de test d'excursions aléatoires (Random Excursion Variant Test : REUT)

Les tests paramétriques :

- Test de Frequence dans un bloc (Block Frequency Test : BFT),
- Test d'entropie approchée (Approximate Entropy Test : AET),
- Test de la compléxité linéaire (Linear Complexity Test : LCT),
- Test statistique universel de Maurer (Maurer's Universal Statistical Test : MUST),
- Test série (Serial Test : ST),
- Recherche d'un motif périodique (Overlapping (Periodic) Template Matching Test : OTMT)
- Recherche d'un motif apériodique (Non-Overlapping (Aperiodic) Template Mat-

ching Test : NOTMT).

Dans notre étude expérimentale, nous avons choisi de valider notre GNPA par l'intermédiaire de la série de tests de NIST. En effet, il faut tout d'abord énoncer une hypothèse, notée par H_0, qui représente le caractère à tester et qui consiste dans notre cas à tester si une séquence est aléatoire. Le contraire de l'hypothèse H_0 est également défini et il est noté par H_a. Ainsi, à la fin de chaque test, le résultat trouvé permet de choisir entre l'acceptation ou le rejet de l'hypothèse H_0. La décision dépend de la probabilité p_value obtenue pour chaque test et du niveau de signification noté par α qui est fixé dans nos expérimentations à 0.01. Ainsi, pour un test donné, si la p_value calculée est inférieure à α, alors la séquence de bits générée par un GNPA est considérée non-aléatoire. Par contre, si tous les p_values sont supérieures à α, alors nous acceptons que la séquence binaire générée par un GNPA est pseudo-aléatoire.

L'aspect aléatoire d'un GNPA est une propriété probabiliste. Il existe plusieurs tests statistiques et chacun tend à vérifier la présence ou l'absence d'un certain modèle, qui une fois détecté nous permette de conclure que la séquence n'est pas aléatoire.

Pour évaluer la qualité de notre GNPA et justifier l'utilisation de la fonction ECF-map, nous présentons dans les tableaux 5.5- 5.10 les résultats des tests statistiques appliqués sur les séquences k_j et b_j. Il est à rappeler que la séquence k_j est générée selon l'équation 5.12, alors que la séquence b_j est obtenue en utilisant uniquement le système chaotique CSM et en appliquant la transformation suivante :

$$b_j = F(a_j), \tag{5.13}$$

avec a_j est la séquence calculée par l'équation 5.11.

Les tableaux 5.5- 5.10 présentent, entre parenthèse et devant chaque test paramétrique, les paramètres choisis dans notre étude expérimentale. De plus, les valeurs présentées dans ces tableaux sont les p_value_T et les facteurs de proportions obtenus pour $m = 100$ séquences de tailles 1000000 bits chacune et avec différentes clés. Selon le NIST, un bon générateur doit avoir des p_values_T supérieurs à 0.0001 et des facteurs de proportions compris entre 0.960150 et 1.019049. D'après les résultats obtenus, nous pouvons conclure que les séquences b_j ne satisfassent pas les critères des séquences aléatoires. Les séquences b_j ne réussissent pas certains tests, comme le FT, la p_value_T est inférieure à 0.0001 et le facteur de proportion est en dehors des limites permises.

Les tableaux 5.5- 5.10 présentent également les résultats obtenus en appliquant les tests du NIST sur les séquences k_j générées par différentes clés. D'après ces résultats,

nous pouvons conclure que notre GNPA réussit tous les tests et par conséquent nous acceptons qu'il est aléatoire.

Nous détaillons, section 5.2.6.3, la manière adoptée par le NIST pour l'examen et l'interprétation des résultats obtenus par les tests statistiques.

5.2.6.3 Interprétation des résultats expérimentaux

- **Examen des facteurs de proportions :**
  Pour vérifier la qualité des résultats obtenus par les tests statistiques, il faut calculer le facteur de proportion. Ce dernier est obtenu en divisant le nombre de p_value supérieures au niveau de signification α par le nombre total des séquences testées.

$$proportion = \frac{nombre\ de\ p\_value > \alpha}{m}, \tag{5.14}$$

  avec m le nombre de séquences utilisées durant le test. Pour chaque test, nous acceptons le facteur de proportion qui se trouve dans le rang défini par NIST et qui est égal à $\hat{p} \pm 3\sqrt{\hat{p}(1-\hat{p})/m}$ avec $\hat{p} = 1-\alpha$. Dans le cas où le facteur de proportion se trouve en dehors de cet intervalle, nous pouvons conclure que les séquences générées ne sont pas aléatoires. Pour $m = 100$ et $\alpha = 0.01$, le rang des proportions acceptables est égal à $[0.960150, 1.019049]$.
  Les valeurs des facteurs de proportions sont présentées figure 5.9 avec les bornes de la région des proportions acceptables. Nous remarquons que toutes les proportions se trouvent à l'intérieur de cette région et par conséquent nous acceptons que les séquences générées sont pseudo-aléatoires.
- **Examen de l'uniformité des p_values :**
  Une autre indication très importante qui vérifie la qualité des tests réalisés est le facteur d'uniformité. Ce dernier consiste à vérifier pour chaque test l'uniformité de la distribution des p_value obtenues. Il est à rappeler que les p_values appartiennent à l'intervalle $[0, 1]$. Ainsi, pour tester l'uniformité des p_values trouvées, le NIST a proposé de diviser l'intervalle $[0, 1]$ en 10 sous-intervalles et de calculer pour chaque test les fréquences F_i pour $i = 1, \cdots, 10$ avec F_i représente le nombre de p_values qui se trouvent dans le $i^{\text{ème}}$ intervalle $[\frac{(i-1)}{10}, \frac{i}{10})$. Ensuite, il faut calculer la valeur $\mathcal{X}^2$ qui présente le test de Pearson. Cette valeur est nécessaire pour vérifier l'uniformité de la distribution des p_values :

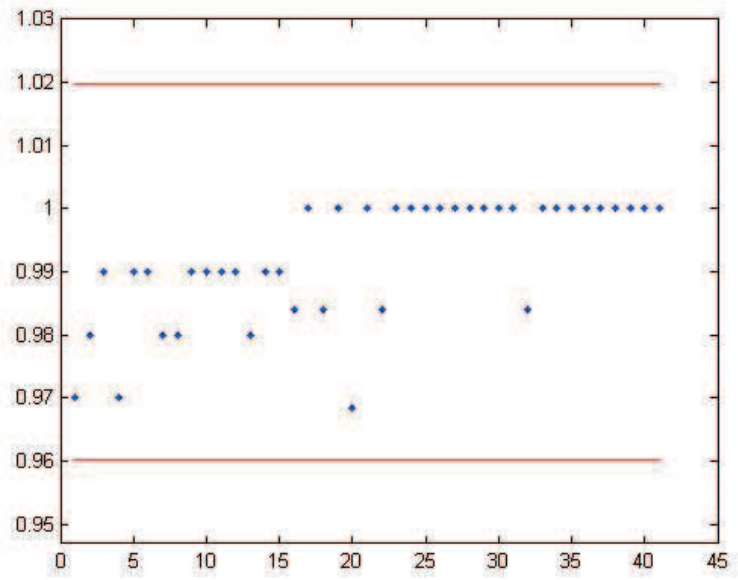


FIGURE 5.9 – Les proportions obtenues pour 100 séquences de taille 1000000 bits chacune. La région entre les deux lignes horizontales est la région des proportions acceptées.

$$\mathcal{X}^2 = \sum_{i=1}^{10} \frac{(F_i - \frac{m}{10})^2}{\frac{m}{10}}. \tag{5.15}$$

Finalement, il faut calculer la p_value_T de toutes les p_values :

$$p\_value_T = igamc(\frac{X^2}{2}, \frac{9}{2}), \tag{5.16}$$

avec $igamc$ est la fonction gamma incomplète [1]. Pour plus de détails, le lecteur pourra se référer à [Rukhin 2008, Abramowitz 1968, Press 1992]. D'après NIST, si $p\_value_T > 0.0001$, alors nous considérons que les p_values sont uniformément distribuées dans $[0, 1]$.

La figure 5.10 présente les p_value_T, pour tous les tests et en utilisant différentes clés, et la limite inférieure qui est égale à 0.0001. De plus, la figure 5.11 présente les histogrammes des p_values de quelques tests. D'après ces figures, nous pouvons conclure que les p_values sont uniformément distribuées.

1. Incomplete gamma function

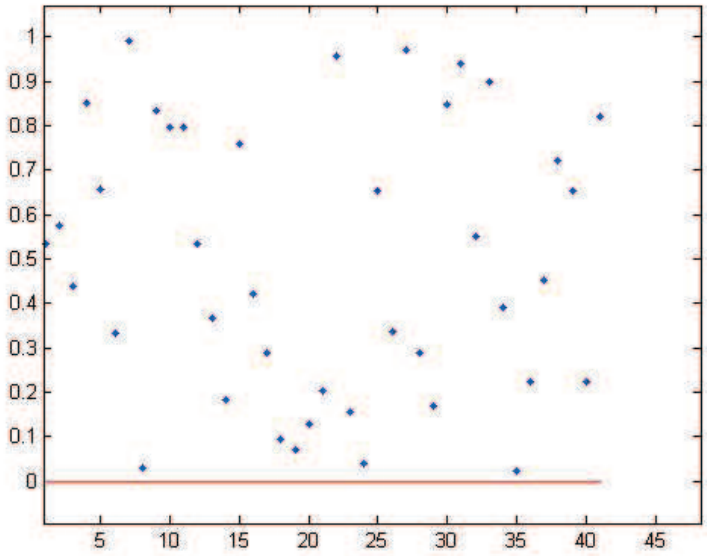


FIGURE 5.10 – Les p_value_T pour tous les tests de NIST. La ligne horizontale présente la limite inférieure qui est égale à 0.0001.

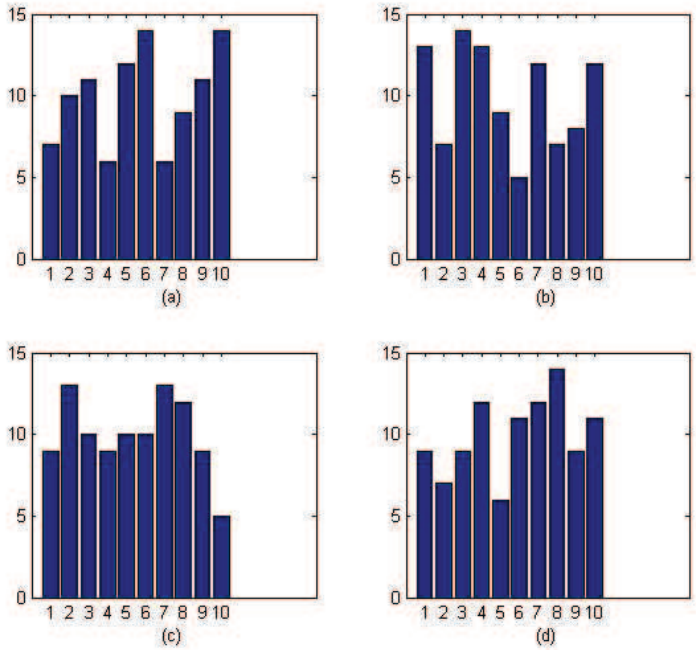


FIGURE 5.11 – Histogramme des p_values pour les tests a) FT, b) RT, c) MUST et d) LZT.

TABLE 5.1 – Etude des facteurs de proportions et de l'uniformité des valeurs des p_values obtenues en appliquant les 14 premiers tests de NIST sur 100 séquences générées par la LM et de taille 1000000 bits chacune.

Test	$x_0 = 0.236415897$ $\lambda = 3.9965248$		$x_0 = 0.94136278$ $\lambda = 3.99996524$	
	$\{b_j\}$		$\{b_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
FT	**0.000000**	**0.0000**	**0.000000**	**0.3300**
BFT (m = 128)	**0.000000**	**0.0000**	**0.000000**	1.0000
RT	**0.000000**	**0.0000**	**0.000000**	**0.3400**
LROT	**0.000000**	**0.0000**	0.236810	0.9900
MRT	0.075719	0.9900	0.437274	0.9700
SPT	**0.000000**	**0.8600**	0.062821	0.9900
NOTMT (m = 9) Template = 000000001	**0.000000**	**0.0000**	**0.000000**	**0.0000**
OTMT (Template = 111111111)	**0.000000**	**0.7800**	0.437274	1.0000
MUST (L = 7,Q = 1280)	**0.000000**	**0.1800**	0.002043	**0.9400**
LZT	**0.000000**	**0.0000**	**0.000000**	**0.0000**
LCT (M = 500)	0.153763	0.9900	0.779188	0.9800
ST (m = 16)	**0.000000**	**0.0000**	**0.000000**	**0.0000**
AET (m = 10)	**0.000000**	**0.0000**	**0.000000**	**0.0000**
CST Forward	**0.000000**	**0.0000**	**0.000000**	**0.3600**
Reverse	**0.000000**	**0.0000**	**0.000000**	**0.3600**

TABLE 5.2 – Suite du tableau 5.1 pour les 15ème et 16ème tests de NIST.

Test	$\{b_j\}$		$\{b_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
RET				
x = -4	**0.000000**	**0.0000**	0.242986	1.0000
x = -3	**0.000000**	**0.0000**	0.186566	1.0000
x = -2	**0.000000**	**0.0000**	0.057146	1.0000
x = -1	**0.000000**	**0.0000**	0.010606	1.0000
x = 1	**0.000000**	**0.0000**	0.484646	1.0000
x = 2	**0.000000**	**0.0000**	0.311542	1.0000
x = 3	**0.000000**	**0.0000**	0.105618	1.0000
x = 4	**0.000000**	**0.0000**	0.041438	0.9630
REVT				
x = -9	**0.000000**	**0.0000**	0.689019	1.0000
x = -8	**0.000000**	**0.0000**	0.186566	1.0000
x = -7	**0.000000**	**0.0000**	0.078086	**0.9259**
x = -6	**0.000000**	**0.0000**	0.186566	**0.9259**
x = -5	**0.000000**	**0.0000**	0.141256	**0.9259**
x = -4	**0.000000**	**0.0000**	0.010606	0.9630
x = -3	**0.000000**	**0.0000**	0.186566	1.0000
x = -2	**0.000000**	**0.0000**	0.242986	1.0000
x = -1	**0.000000**	**0.0000**	0.041438	1.0000
x = 1	**0.000000**	**0.0000**	0.186566	1.0000
x = 2	**0.000000**	**0.0000**	0.186566	1.0000
x = 3	**0.000000**	**0.0000**	0.010606	1.0000
x = 4	**0.000000**	**0.0000**	0.105618	1.0000
x = 5	**0.000000**	**0.0000**	0.141256	1.0000
x = 6	**0.000000**	**0.0000**	0.585209	1.0000
x = 7	**0.000000**	**0.0000**	0.311542	1.0000
x = 8	**0.000000**	**0.0000**	0.105618	1.0000
x = 9	**0.000000**	**0.0000**	0.788728	1.0000

TABLE 5.3 – Etude des facteurs de proportions et de l'uniformité des valeurs des p_values obtenues en appliquant les 14 premiers tests de NIST sur 100 séquences générées avec le GNPA proposé par [Patidar 2009b], de taille 1000000 bits chacune.

Test	$x2_0 = 2.2548745491$ $y2_0 = 3.9654128766$ $t2 = 100.6$ $x1_0 = 5.6984257$ $y1_0 = 1.96532548$ $t1 = 50.3975246$		$x2_0 = 3.59587469543$ $y2_0 = 0.8512974635$ $t2 = 120.9625487136$ $x1_0 = 5.285647931$ $y1_0 = 0.456329817$ $t1 = 20.$	
	$\{b_j\}$		$\{b_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
FT	0.071177	0.9800	0.455937	0.9800
BFT (m = 128)	**0.000000**	**0.0500**	**0.000000**	**0.0000**
RT	**0.000000**	**0.0000**	**0.000000**	**0.0000**
LROT	**0.000000**	**0.5800**	**0.000000**	**0.3000**
MRT	0.514124	0.9700	0.834308	1.0000
SPT	0.437274	1.0000	0.016717	1.0000
NOTMT (m = 9) Template = 000000001	**0.000000**	**0.0000**	**0.000000**	**0.0000**
OTMT (Template = 111111111)	**0.000000**	**0.0000**	**0.000000**	**0.0000**
MUST (L = 7,Q = 1280)	**0.000014**	**0.9300**	**0.000000**	**0.8100**
LZT	**0.000000**	**0.2200**	**0.000000**	**0.0000**
LCT (M = 500)	0.289667	0.9900	0.366918	0.9900
ST (m = 16)	**0.000000**	**0.3600**	**0.000000**	**0.0500**
AET (m = 10)	**0.000000**	**0.0000**	**0.000000**	**0.0000**
CST Forward	0.554420	1.0000	0.366918	0.9900
Reverse	0.051942	0.9900	0.574903	0.9800

TABLE 5.4 – Suite du tableau 5.3 pour les 15ème et 16ème tests de NIST.

Test	$\{b_j\}$		$\{b_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
RET				
x = -4	0.108791	1.0000	0.008366	**0.9420**
x = -3	0.003201	0.9828	0.026648	0.9710
x = -2	0.350485	0.9828	0.018969	**0.9565**
x = -1	0.955835	1.0000	0.095617	**0.9420**
x = 1	0.350485	0.9655	0.008366	**0.9275**
x = 2	0.213309	0.9655	0.010606	0.9710
x = 3	0.262249	1.0000	**0.000000**	1.0000
x = 4	0.236810	0.9655	0.155209	0.9710
REVT				
x = -9	0.455937	1.0000	0.551026	1.0000
x = -8	0.153763	1.0000	0.788728	0.9855
x = -7	0.289667	1.0000	0.364146	**0.9565**
x = -6	0.657933	1.0000	0.551026	0.9855
x = -5	0.494392	1.0000	0.116519	1.0000
x = -4	0.108791	1.0000	0.063482	1.0000
x = -3	0.574903	0.9828	0.819544	1.0000
x = -2	0.455937	0.9828	0.517442	1.0000
x = -1	0.171867	1.0000	0.392456	1.0000
x = 1	0.058984	1.0000	0.023812	1.0000
x = 2	0.171867	0.9828	0.264458	0.9855
x = 3	0.657933	1.0000	0.452799	0.9855
x = 4	0.213309	1.0000	0.001156	1.0000
x = 5	0.851383	1.0000	0.242986	1.0000
x = 6	0.779188	1.0000	0.392456	1.0000
x = 7	0.289667	1.0000	0.141256	1.0000
x = 8	0.023545	1.0000	0.116519	1.0000
x = 9	0.066882	1.0000	0.005166	1.0000

TABLE 5.5 – Comparaison des résultats obtenus en appliquant les 14 premiers tests de NIST sur les séquences $\{b_j\}$, générées uniquement avec le CSM, et sur les séquences, $\{k_j\}$, obtenues avec notre GNPA. La clé choisie pour la générations des séquences $\{b_j\}$ et $\{k_j\}$ est ($x_0 = 3.59587469543$, $y_0 = 0.8512974635$, $t = 120.9625487136$, $N_0 = 250$).

Test	$x_0 = 3.59587469543$, $y_0 = 0.8512974635$ $t = 120.9625487136$, $N_0 = 250$			
	$\{b_j\}$		$\{k_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
FT	0.401199	0.9700	0.534146	0.9700
BFT (m = 128)	**0.000000**	**0.0100**	0.574903	0.9800
RT	0.236810	0.9800	0.437274	0.9900
LROT	**0.000000**	**0.3900**	0.851383	0.9700
MRT	0.275709	1.0000	0.657933	0.9900
SPT	0.023545	1.0000	0.334538	0.9900
NOTMT (m = 9) Template = 000000001	**0.000000**	**0.0000**	0.991468	0.9800
OTMT (Template = 111111111)	**0.000000**	**0.0000**	0.028817	0.9800
MUST (L = 7,Q = 1280)	**0.000000**	**0.7500**	0.834308	0.9900
LZT	**0.000000**	**0.0000**	0.798139	0.9900
LCT (M = 500)	0.055361	1.0000	0.798139	0.9900
ST (m = 16)	**0.000000**	**0.0000**	0.534146	0.9900
AET (m = 10)	**0.000000**	**0.0000**	0.366918	0.9800
CST Forward	0.030806	0.9700	0.181557	0.9900
Reverse	0.051942	0.9900	0.759756	0.9900

TABLE 5.6 – Suite du tableau 5.5 pour la comparaison des résultats obtenus avec les 15ème et 16ème tests de NIST.

Test	$\{b_j\}$		$\{k_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
RET				
x = -4	0.108791	**0.9423**	0.422034	0.9841
x = -3	0.096578	0.9808	0.287306	1.0000
x = -2	0.350485	0.9615	0.095617	0.9841
x = -1	0.108791	1.0000	0.070445	1.0000
x = 1	0.534146	1.0000	0.128379	0.9683
x = 2	0.883171	1.0000	0.204076	1.0000
x = 3	0.262249	1.0000	0.957319	0.9841
x = 4	0.319084	1.0000	0.155209	1.0000
REVT				
x = -9	0.030806	1.0000	0.041438	1.0000
x = -8	0.383827	1.0000	0.654467	1.0000
x = -7	0.350485	1.0000	0.337162	1.0000
x = -6	0.153763	1.0000	0.970538	1.0000
x = -5	0.171867	1.0000	0.287306	1.0000
x = -4	0.035174	0.9808	0.170294	1.0000
x = -3	0.262249	0.9808	0.848588	1.0000
x = -2	0.455937	0.9808	0.941144	1.0000
x = -1	0.996335	0.9808	0.551026	0.9841
x = 1	0.534146	1.0000	0.900104	1.0000
x = 2	0.455937	1.0000	0.392456	1.0000
x = 3	0.739918	1.0000	0.023812	1.0000
x = 4	0.699313	1.0000	0.222869	1.0000
x = 5	0.191687	1.0000	0.452799	1.0000
x = 6	0.001296	1.0000	0.723129	1.0000
x = 7	0.494392	1.0000	0.654467	1.0000
x = 8	0.191687	1.0000	0.222869	1.0000
x = 9	0.494392	1.0000	0.819544	1.0000

TABLE 5.7 – Comparaison des résultats obtenus en appliquant les 14 premiers tests de NIST sur des séquences $\{b_j\}$, générées uniquement avec le CSM, et sur des séquences $\{k_j\}$, obtenues avec notre GNPA. La clé choisie pour la générations des séquences $\{b_j\}$ et $\{k_j\}$ est ($x_0 = 2.2548745491$, $y_0 = 3.9654128766$, $t = 20.6$, $N_0 = 250$).

Test	$x_0 = 2.2548745491$, $y_0 = 3.9654128766$ $t = 20.6$, $N_0 = 250$			
	$\{b_j\}$		$\{k_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
FT	0.494392	1.0000	0.004301	1.0000
BFT (m = 128)	**0.000000**	**0.0000**	0.055361	1.0000
RT	0.004301	0.9800	0.719747	0.9900
LROT	**0.000000**	**0.0000**	0.779188	0.9900
MRT	0.678686	1.0000	0.883171	1.0000
SPT	**0.000000**	**0.9000**	0.699313	0.9900
NOTMT (m = 9) Template = 000000001	**0.000000**	**0.0000**	0.816537	0.9800
OTMT (Template = 111111111)	**0.000000**	**0.0000**	0.616305	0.9900
MUST (L = 7,Q = 1280)	**0.000000**	**0.0000**	0.883171	0.9900
LZT	**0.000000**	**0.0000**	0.191687	1.0000
LCT (M = 500)	0.678686	1.0000	0.834308	1.0000
ST (m = 16)	**0.000000**	**0.0000**	0.153763	1.0000
AET (m = 10)	**0.000000**	**0.0000**	0.016717	0.9800
CST				
Forward	0.014550	0.9800	0.019188	1.0000
Reverse	0.000145	0.9900	0.574903	1.0000

TABLE 5.8 – Suite du tableau 5.7 pour la comparaison des résultats obtenus avec les 15ème et 16ème tests de NIST.

Test	$\{b_j\}$		$\{k_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
RET				
x = -4	**0.000005**	0.9672	0.141256	0.9665
x = -3	0.003161	**0.9508**	0.311542	0.9855
x = -2	0.029796	**0.9508**	0.046169	1.0000
x = -1	0.422034	1.0000	0.033288	1.0000
x = 1	0.204076	0.9836	0.654467	1.0000
x = 2	0.116519	**0.9344**	0.170294	1.0000
x = 3	0.128379	0.9836	0.242986	0.9855
x = 4	0.116519	**0.9180**	0.723129	1.0000
REVT				
x = -9	0.170294	1.0000	0.105618	1.0000
x = -8	0.689019	1.0000	0.063482	1.0000
x = -7	0.585209	1.0000	0.186566	1.0000
x = -6	0.551026	0.9672	0.392456	0.9855
x = -5	0.551026	1.0000	0.046169	0.9855
x = -4	0.337162	1.0000	0.095617	0.9855
x = -3	0.116519	1.0000	0.070445	0.9855
x = -2	0.788728	1.0000	0.204076	1.0000
x = -1	0.970538	1.0000	0.186566	1.0000
x = 1	0.422034	0.9836	0.011931	0.9855
x = 2	0.128379	0.9836	0.364146	0.9855
x = 3	0.311542	1.0000	0.452799	1.0000
x = 4	0.204076	1.0000	0.551026	0.9855
x = 5	0.070445	1.0000	0.242986	0.9855
x = 6	0.086458	1.0000	0.654467	1.0000
x = 7	0.002465	1.0000	0.170294	1.0000
x = 8	0.141256	1.0000	0.004573	1.0000
x = 9	0.155209	1.0000	0.021262	1.0000

TABLE 5.9 – Comparaison des résultats obtenus en appliquant les 14 premiers tests de NIST sur des séquences $\{b_j\}$, générées uniquement avec le CSM, et sur des séquences $\{k_j\}$, obtenues avec notre GNPA. La clé choisie pour la générations des séquences $\{b_j\}$ et $\{k_j\}$ est ($x_0 = 6.0125485265$, $y_0 = 0.2015036089$, $t = 50.951236874$, $N_0 = 250$).

Test	$x_0 = 6.0125485265$, $y_0 = 0.2015036089$ $t = 50.951236874$, $N_0 = 250$			
	$\{b_j\}$		$\{k_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
FT	0.851383	0.9800	0.798139	0.9900
BFT (m = 128)	**0.000000**	**0.0000**	0.171867	0.9800
RT	0.066882	1.0000	0.699313	0.9900
LROT	**0.000000**	**0.0000**	0.275709	0.9800
MRT	0.851383	0.9800	0.350485	1.0000
SPT	**0.000000**	**0.2300**	0.474986	1.0000
NOTMT (m = 9) Template = 000000001	**0.000000**	**0.0000**	0.153763	1.0000
OTMT (Template = 111111111)	**0.000000**	**0.0000**	0.383827	0.9900
MUST (L = 7,Q = 1280)	**0.000000**	**0.0000**	0.554420	0.9900
LZT	**0.000000**	**0.0000**	0.455937	1.0000
LCT (M = 500)	0.419021	1.0000	0.171867	0.9900
ST (m = 16)	**0.000000**	**0.0000**	0.739918	1.0000
AET (m = 10)	**0.000000**	**0.0000**	0.534146	1.0000
CST				
Forward	0.122325	0.9900	0.574903	0.9900
Reverse	0.002043	0.9900	0.595549	0.9800

TABLE 5.10 – Suite du tableau 5.9 pour la comparaison des résultats obtenus avec les $15^{\text{ème}}$ et $16^{\text{ème}}$ tests de NIST.

Test	$\{b_j\}$		$\{k_j\}$	
	P_value_T	Proportion	P_value_T	Proportion
RET				
x = -4	**0.000008**	**0.9216**	0.401199	1.0000
x = -3	**0.000002**	**0.8235**	0.224821	1.0000
x = -2	**0.000000**	**0.9216**	0.514124	0.9818
x = -1	0.275709	0.9804	0.275709	0.9818
x = 1	0.964295	1.0000	0.037566	0.9818
x = 2	0.000184	0.9608	0.719747	0.9818
x = 3	0.055361	0.9608	0.202268	0.9818
x = 4	0.012650	**0.9020**	0.637119	0.9818
REVT				
x = -9	0.834308	0.9804	0.366918	1.0000
x = -8	0.678686	0.9804	0.249284	0.9818
x = -7	0.759756	0.9804	0.798139	0.9818
x = -6	0.202268	0.9804	0.437274	0.9818
x = -5	0.048716	0.9804	0.514124	0.9818
x = -4	0.678686	0.9804	0.102526	0.9818
x = -3	0.834308	0.9804	0.514124	1.0000
x = -2	0.897763	0.9804	0.275709	0.9636
x = -1	0.249284	0.9804	0.759756	0.9818
x = 1	0.554420	1.0000	0.678686	0.9818
x = 2	0.637119	1.0000	0.437274	1.0000
x = 3	0.080519	1.0000	0.798139	1.0000
x = 4	0.055361	1.0000	0.055361	1.0000
x = 5	0.129620	1.0000	0.474986	0.9818
x = 6	0.514124	1.0000	0.867692	0.9636
x = 7	0.437274	1.0000	0.304126	0.9818
x = 8	0.249284	1.0000	0.867692	0.9818
x = 9	0.554420	1.0000	0.637119	0.9818

5.2.7 Conclusion

Dans la section 5.2, nous avons présenté un nouveau GNPA [Masmoudi 2010b] pour la génération d'une manière déterministe des flux de clés dynamiques à des fins de chiffrement. Nous avons montré, section 5.2.6, que notre GNPA est très efficace et a réussit tous les tests de la série de tests de NIST. L'efficacité de notre GNPA vient de l'utilisation du système chaotique CSM, caractérisé par la sensibilité aux valeurs initiales et au paramètre de contrôle, et de la fonction ECF-map, possédant des propriétés statistiques très intéressantes. Notre générateur sera utilisé pour le développement d'un nouvel algorithme de chiffrement par flot, section 5.3, et le développement d'une nouvelle méthode de crypto-compression, section 5.4.

5.3 Un nouvel algorithme de chiffrementd'images

5.3.1 Introduction

Récemment, plusieurs crypto-systèmes basés sur le chaos ont été proposés [Li 2001a, Yang 2004, Wong 2008, Wu 2004, Zhang 2005]. Le caractère chaotique a trouvé sa place en cryptographie puisqu'il garantit la sensibilité, des séquences de valeurs générées, aux valeurs initiales et aux paramètres de contrôles [Kocarev 2001]. De plus, certains systèmes chaotiques sont capables de générer des séquences possédant des propriétés statistiques très intéressantes, comme l'aléatoire, l'imprédictibilité, la non-corrélation entre les valeurs successives, et la non périodicité. Ces propriétés ont fait du chaos un très bon moyen pour le développement des algorithmes de chiffrement par flot. Cependant, le choix du système chaotique dépend du niveau de sécurité exigé et de la capacité du calcul de l'équipement qui va l'exécuter.

De plus, l'espace de clés de certains systèmes chaotiques n'est pas suffisamment robuste contre tout type d'attaque par force brute. D'autre part, les séquences générées par un système chaotique n'ont pas forcément une densité de probabilité uniforme pour empêcher tout type d'attaque par analyse statistique [Li 2009, Wei 2007].

En effet, le choix du système chaotique est la tâche la plus importante dans le développement d'un algorithme de chiffrement. La plupart des crypto-systèmes qui sont basés uniquement sur l'utilisation d'un seul système chaotique ne sont pas très efficaces puisqu'il est toujours possible d'extraire les informations caractérisant un système chaotique à partir de sa trajectoire [Short 1997, Li 2007, Yang 1998]. Ainsi, nous proposons de combiner le système chaotique CSM avec la fonction ECF-map pour augmenter la complexité d'un crypto-système basé uniquement sur le CSM. En effet, la fonction ECF-map conserve les propriétés du CSM, comme la sensibilité aux valeurs initiales et au paramètre de contrôle, et ajoute de nouvelles propriétés statistiques et spectrales très appropriées en cryptographie. En effet, le système CSM et la fonction ECF-map ont été combinés pour développer un nouveau GNPA présenté section 5.2.5.

D'après les expérimentations déjà présentées section 5.2.6, nous avons montré que le GNPA proposé réussit tous les tests de la série de tests statistiques proposés par NIST, et par conséquent nous proposons de l'utiliser dans le développement d'un nouvel algorithme de chiffrement symétrique par flot. La section 5.3.2 présente l'algorithme de chiffrement proposé. Ensuite, la section 5.3.3 se consacre à l'étude de la sécurité de notre nouvelle

méthode de chiffrement tout en montrant sa robustesse contre plusieurs types d'attaques. Nous terminons cette section par une conclusion, section 5.3.4.

5.3.2 Chiffrement symétrique par flot proposé

Dans cette section, nous proposons de détailler notre algorithme de chiffrement symétrique par flot appliqué aux images. Le chiffrement est réalisé grâce à notre GNPA, présenté section 5.2.5, pour chiffrer les pixels d'une image. Notre générateur a été légèrement modifié afin de pouvoir intégrer les pixels de l'image originale dans la procédure de génération des flux de clés aléatoires en vue d'améliorer la sécurité du chiffrement proposé et d'assurer la robustesse contre l'attaque par texte clair connu.

Il est à signaler que la clé de cryptage/décryptage de notre chiffrement est la même utilisée par notre GNPA, et est composée par trois nombres réels et un entier $K = (x_0, y_0, t, N_0)$.

Nous proposons tout d'abord de transformer la matrice des pixels de l'image originale de taille $N \times M$ en un vecteur noté par S, avec $S = (S_1, ..., S_R)$ et $R = N \times M$ représente le nombre total de pixels. Quant à l'image chiffrée, elle est transformée en un vecteur noté par $C = (C_1, ..., C_R)$. Chaque élément de ces deux vecteurs est représenté sur m bits, et correspond à l'intensité du pixel qui varie entre 0 et $L-1$, avec $L = 2^m$. Il est à noter que pour une image en niveaux de gris, m est généralement égal à 8 et par conséquent nous obtenons $L = 256$ intensités différentes. Pour crypter une image et modifier les valeurs de ses pixels, nous proposons d'appliquer un ou exclusif entre les valeurs obtenues par notre GNPA et les valeurs des pixels de l'image originale. Nous procédons maintenant à détailler la procédure de chiffrement proposée :

- **1** Initialement, le système CSM génère un couple de valeurs (x_j, y_j) à chaque nouveau pixel.
- **2** Ensuite, nous proposons d'utiliser la valeur y_j pour calculer la valeur a_j :

$$a_j = (y_j + \frac{S_j}{L}) - \lfloor (y_j + \frac{S_j}{L}) \rfloor. \tag{5.17}$$

  avec S_j est le $j^{\text{ème}}$ pixel du vecteur S et $a_j \in (0, 1)$.
- **3** Nous proposons par la suite de convertir la séquence $\{a_j\}$ en une nouvelle séquence $\{b_j\}$ en utilisant la fonction G définie équation 5.10 :$b_j = G(a_j)$, avec $b_j \in (0, 1)$. Cette dernière conversion garantit le pseudo-aléatoire des valeurs générées qui seront utilisées dans le chiffrement par flot des images. Toutefois, pour pouvoir appliquer

un ou exclusif entre les valeurs des pixels de l'image originale et les valeurs générées, il faut convertir ces dernières valeurs en des entiers compris entre 0 et $L-1$. Pour ce faire, nous proposons de définir la fonction $f_L : [0,1) \rightarrow \{0,...,L-1\}$ donnée par $f_L(x) = i$, si $x \in I_i$, sachant que l'intervalle $[0,1)$ est découpé en L sous-intervalles de même largeur et qui sont caractérisés par $\bigcup_{i=0}^{L-1} I_i = [0,1]$ et $\bigcap_{i=0}^{L-1} I_i = \emptyset$. De ce fait, le flux de clés se calcule par $k_j = f_L(b_j)$ et le vecteur des pixels cryptés C sera obtenu par :

$$C_j = S_j \oplus k_j, \tag{5.18}$$

avec $\oplus$ désigne l'opérateur ou exclusif.

Ces opérations sont itérées plusieurs fois jusqu'à crypter la totalité de l'image. En effet, grâce à notre approche, les valeurs des pixels de l'image originale interviennent dans la génération des flux de clés, et par conséquent, chaque image sera cryptée avec un flux de clés différent, même si nous utilisons la même clé de cryptage.

Notre algorithme s'applique aussi pour des images binaires et couleurs. Ainsi, pour une image binaire, chaque pixel est représenté par un seul bit et par conséquent il faut diviser l'intervalle $[0,1)$ en deux sous-intervalles ce qui nous permet de chiffrer l'image bit par bit. En outre, pour chiffrer une image couleur, il suffit d'appliquer notre algorithme sur chaque composante de l'image.

Le nouveau chiffrement symétrique par flot proposé est très efficace. La section 5.3.3 est consacrée à l'évaluation des performances de notre méthode, tout en analysant sa robustesse contre plusieurs types d'attaques.

5.3.3 Analyse de la sécurité de l'algorithme de chiffrement proposé

Dans cette section, nous allons analyser la sécurité de l'algorithme de chiffrement proposé section 5.3.2 contre plusieurs types d'attaques. Ainsi, nous nous intéressons tout d'abord, section 5.3.3.1, à analyser les statistiques des images cryptées (histogramme, corrélation et entropie). Ensuite, par une analyse des mesures NPCR et UACI, la section 5.3.3.2 montre la robustesse de notre approche contre l'attaque différentielle. Finalement, la section 5.3.3.3 se consacre à l'exposition de la sensibilité de notre algorithme à tout petit changement dans la clé secrète.

TABLE 5.11 – Les clés utilisées dans les expérimentations.

	k_0	k_1	k_2	k_3	k_4
x_0	5.87574682393162	5.87574682393161	5.87574682393162	5.87574682393162	5.87574682393162
y_0	0.20543974869398	0.20543974869398	0.20543974869399	0.20543974869398	0.20543974869398
t	90.41936758463719	90.41936758463719	90.41936758463719	90.41936758463720	90.41936758463719
N_0	250	250	250	250	251

5.3.3.1 Analyses statistiques

a) ***Histogrammes des images cryptées :***

Un crypto-système d'image doit nécessairement être robuste contre tout type d'attaque par analyse statistique. Ainsi, nous avons proposé dans un premier temps d'analyser les histogrammes des images cryptées. Pour ce faire, nous avons analysé les histogrammes de 100 images originales, choisies d'une manière aléatoire, ainsi que les histogrammes des images cryptées correspondantes. Nous avons présenté dans cette section uniquement l'histogramme de l'image Lena, avec une taille de 512×512 pixels et 256 niveaux de gris, ainsi que l'histogramme de l'image cryptée correspondante obtenue en utilisant la clé k_0 indiquée tableau 5.11. Il est à noter que nous avons trouvé les mêmes résultats pour les autres images. L'image Lena et l'image cryptée correspondante sont représentées respectivement par figure 5.12.a et figure 5.12.b. De plus, l'histogramme de l'image Lena est illustré figure 5.12.c, alors que l'histogramme de l'image cryptée correspondante est représenté figure 5.12.d. Nous remarquons que les deux histogrammes sont très différents, et par conséquent, aucune information, sur l'image originale, ne peut être déduite à partir de l'image cryptée et de son histogramme. De plus, nous pouvons remarquer aussi, figure 5.12.d, que l'histogramme de l'image cryptée est uniforme, ce qui rend impossible tout type d'attaque basé sur l'analyse de l'histogramme des images cryptées.

b) ***Corrélation des pixels adjacents :***

Comme le décrit chapitre 2, dans une image, chaque pixel est généralement fortement corrélé avec ses pixels voisins, et en particulier avec ceux qui se situent sur la même ligne, la même colonne ou bien sur la même diagonale. Grâce à cette forte corrélation, il devient possible de prédire la valeur d'un pixel à partir de la connaissance

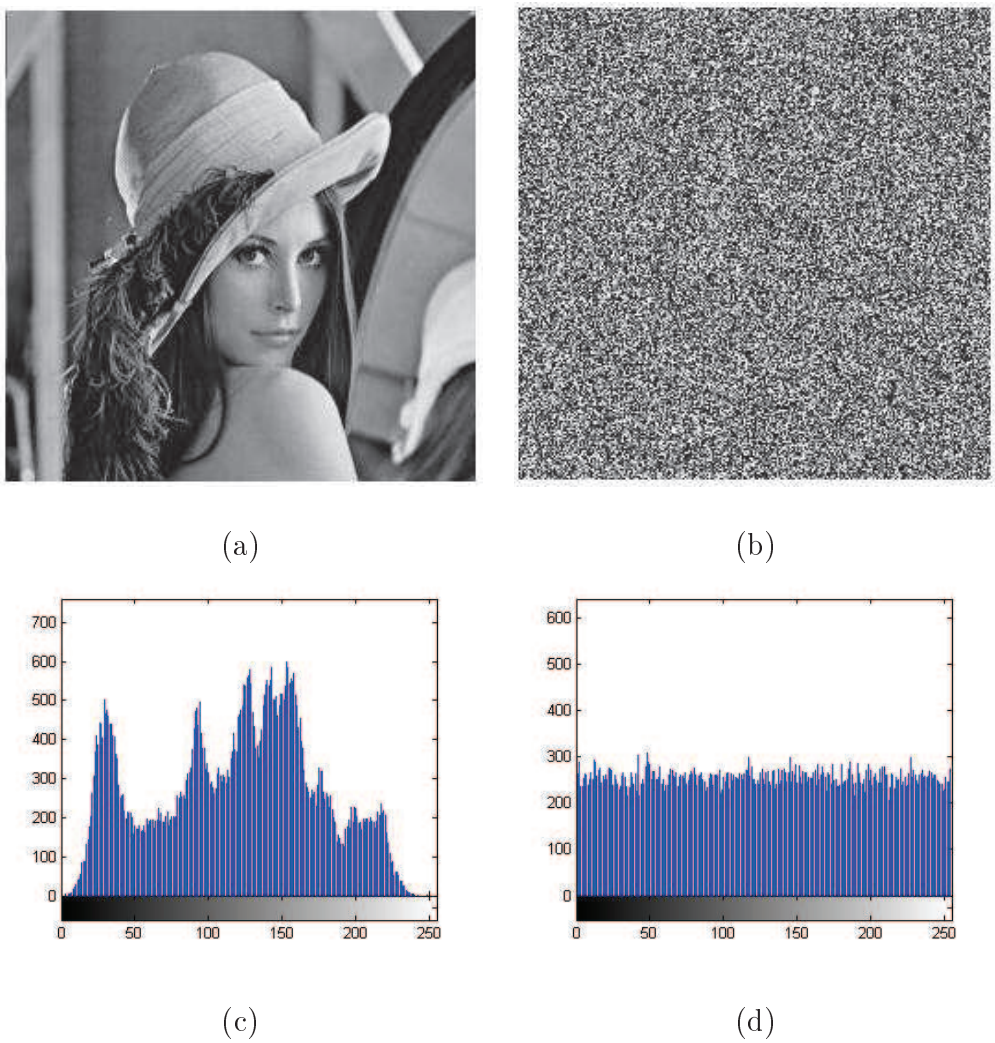


FIGURE 5.12 – a) Image originale Lena, b) Image cryptée en utilisant la clé k_0, c) Histogramme de l'image Lena, d) Histogramme de l'image cryptée obtenue avec la clé k_0.

des valeurs de ses voisins. Dans ce cas, un algorithme de chiffrement doit impérativement éliminer toute sorte de corrélation ou bien de relation entre un pixel et ses voisins, ce qui augmente énormément la complexité de la cryptanalyse [Wu 2004]. Le tableau 5.12 montre les corrélations obtenues pour trois images originales ainsi que leurs images cryptées. De plus, nous avons présenté dans ce même tableau la moyenne des corrélations obtenues pour 100 images originales et les images cryptées correspondantes. En outre, la figure 5.13 montre la distribution de la corrélation entre des pixels adjacents situés sur la même ligne, la même colonne et la même diagonale de l'image claire et de l'image cryptée correspondante obtenue avec la clé k_0. D'après ces distributions, nous remarquons que l'algorithme de chiffrement

proposé réussit à obtenir une corrélation nulle entre les pixels voisins dans toutes les directions. Par conséquent, il n'y a pas de dépendance linéaire entre les pixels voisins, et la connaissance d'un pixel de l'image cryptée n'apporte aucune information sur les valeurs de ses voisins.

TABLE 5.12 – Les coefficients de corrélation des pixels adjacents, des images originales et des images cryptées correspondantes, dans les trois directions : Horizontale, verticale et diagonale.

	images originales	images cryptées
Lena		
horizontale	0.9411	-0.0003
verticale	0.9702	0.0014
diagonale	0.9153	0.0001
Boat		
horizontale	0.9368	0.0012
verticale	0.9709	0.0026
diagonale	0.9293	-0.0002
House		
horizontale	0.9736	-0.0005
verticale	0.9504	0.0004
diagonale	0.9246	0.0022
Moyenne sur 100 images		
horizontale	0.9714	-0.0009
verticale	0.9727	0.0002
diagonale	0.9531	-0.0004

c) ***Entropie des images cryptées***

L'entropie a été initialement définie par Shannon [Shannon 1948a, Shannon 1948b]

et est donnée par l'équation 2.9. D'après la figure 5.12.d, l'image cryptée possède un histogramme uniforme, ce qui veut dire que les niveaux de gris ont le même nombre d'occurrence et par conséquent l'entropie est maximale. Ainsi, pour une image en niveaux de gris, où chaque pixel est représenté par 8 bits, il faut avoir une entropie, pour l'image cryptée, la plus proche possible de 8 bits/pixel [Behina 2008]. L'entropie de l'image Lena cryptée en utilisant la clé k_0 est égale à 7.9995 $\approx$ 8 bits/pixel. De plus, la valeur moyenne obtenue pour 100 images cryptées est égale à 7.9993 bits/pixel. Les résultats trouvés sont très proches de la valeur théorique et par conséquent nous déduisons que notre algorithme de chiffrement est robuste contre l'attaque par analyse de l'entropie.

5.3.3.2 Analyse des mesures : NPCR et UACI

L'objectif de cette analyse est de montrer que les images cryptées sont totalement différentes des images originales correspondantes. Ainsi, nous avons calculé les mesures NPCR et UACI [Chen 2004, Chiaraluce 2002], présentées chapitre 3, entre des images originales et les images cryptées correspondantes. Il est à rappeler que pour un idéal cryptosystème, la valeur théorique du NPCR doit être la plus proche possible de 100%, alors que le UACI doit être très proche de la valeur 33%. Ainsi, nous avons présenté, tableau 5.13, les valeurs moyennes des NPCR et UACI obtenues pour 100 images, et nous avons montré que les valeurs expérimentales sont très proches aux valeurs théoriques. Par conséquent, nous pouvons déduire que les images cryptées avec notre algorithme sont très différentes des images originales correspondantes.

TABLE 5.13 – La moyenne des NPCR et des UACI calculée à partir de 100 images originales et les images cryptées correspondantes (obtenues avec la clé k_0).

	Résultats
NPCR	99.61 %
UACI	33.87 %

5.3.3.3 Sensibilité à la clé

Tout système de chiffrement sécurisé doit être impérativement hyper sensible à toute petite modification dans la clé secrète [Maniccam 2001]. Ainsi, nous proposons dans cette section d'analyser la sensibilité de notre algorithme de chiffrement à la clé secrète que ce soit en émission ou en réception.

a) La clé secrète de notre algorithme de chiffrement est composée par 4 valeurs. Ainsi, pour analyser la sensibilité de notre approche à tout petit changement dans la clé secrète, nous avons proposé tout d'abord de crypter l'image Lena avec la clé k_0, ensuite de crypter la même image mais avec plusieurs clés qui sont légèrements différentes à la clé k_0. Toutes les clés utilisées dans cette analyse expérimentales sont indiquées dans le tableau 5.11. Les figures 5.14.a, 5.14.c, 5.14.e et 5.14.g montrent les images cryptées en utilisant respectivement les clés k_1, k_2, k_3 et k_4. Pour prouver la sensibilité de notre approche à la clé secrète en émission, nous avons montré également dans les figures 5.14.b, 5.14.d, 5.14.f et 5.14.h la différence entre figure 5.12.b et figure 5.14.a, figure 5.12.b et figure 5.14.c, figure. 5.12.b et 5.14.e et figure. 5.12.b et figure 5.14.g respectivement. Ainsi, d'après ces figures, nous pouvons conclure que même avec deux clés qui sont légèrement différentes, de l'ordre de 10^{-14} pour les réels x_0, y_0 et t et de l'ordre de 1 bit pour l'entier N_0, nous obtenons deux images cryptées qui sont totalement différentes. De plus, le tableau 5.14 indique les valeurs du NPCR et UACI obtenues pour des images cryptées avec des clés très proches. Nous remarquons d'après ces valeurs, que toutes les NPCR sont supérieures à 99% et que toutes les UACI sont supérieures à 33%. Ce qui nous permet de déduire que notre algorithme de chiffrement est sensible à la clé secrète en émission.

TABLE 5.14 – Différence entre les images cryptées avec des clés très proches.

	k_1	k_2	k_3	k_4
NPCR (%)	99.60	99.56	99.57	99.60
UACI (%)	33.54	33.58	33.44	33.53

b) Nous proposons maintenant d'analyser la sensibilité à la clé secrète en réception. Pour ce faire, l'image claire est cryptée avec une clé k_0, par contre le décryptage est réalisé avec une autre clé qui est légèrement différente de la clé de cryptage. Par exemple, les figures 5.15.a et 5.15.b présentent respectivement l'image déchiffrée

avec la clé k_1 ainsi que son histogramme. Nous pouvons clairement voir que l'image déchiffrée est absolument différente de l'image claire Lena. Autrement dit, même avec une bonne approximation de la clé, nous n'arrivons pas à trouver l'image claire. De plus, l'histogramme de l'image décryptée avec une clé différente de celle utilisée dans le cryptage est uniforme et absolument différent de l'histogramme de l'image claire Lena.

5.3.4 Conclusion

Dans la section 5.3, nous avons présenté et analysé un nouvel algorithme de chiffrement d'images [Masmoudi 2010c]. L'algorithme proposé permet d'ajouter de la confusion aux pixels des images originales. Cette confusion consiste à mélanger les valeurs des pixels avec des clés dynamiques générées avec notre GNPA et possédant des propriétés statistiques et chaotiques très intéressantes. Les analyses effectuées sur l'algorithme proposé montrent sa robustesse aux différents types d'attaques, ce qui le rend très approprié pour le cryptage d'images.

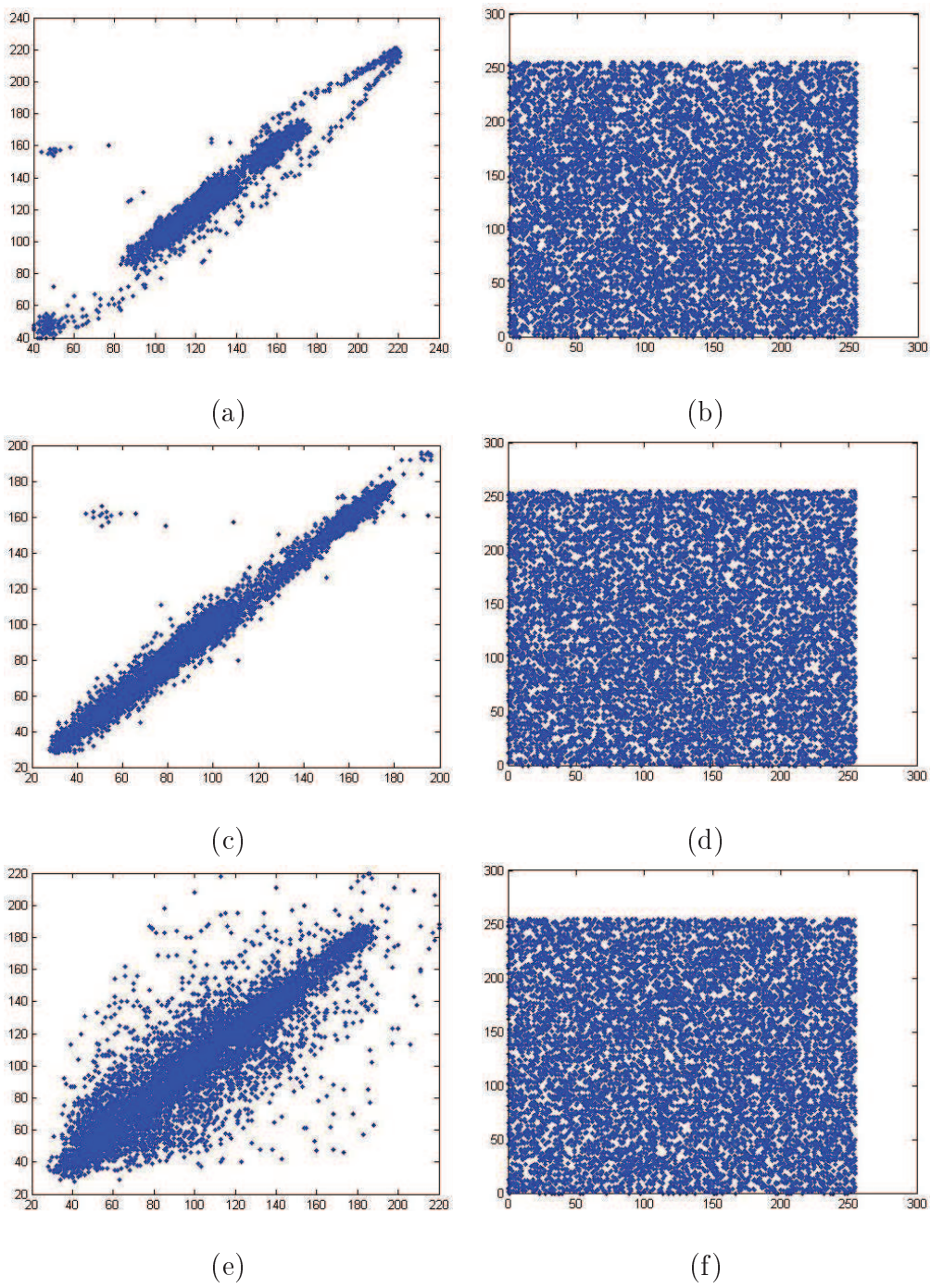


FIGURE 5.13 – Distribution de la corrélation de pixels adjacents a) horizontalement de l'image originale Lena, b) horizontalement de l'image Lena cryptée en utilisant la clé k_0, c) verticalement de l'image originale Lena, d) verticalement de l'image Lena cryptée en utilisant la clé k_0, e) diagonalement de l'image originale Lena, f) diagonalement de l'image Lena cryptée en utilisant la clé k_0.

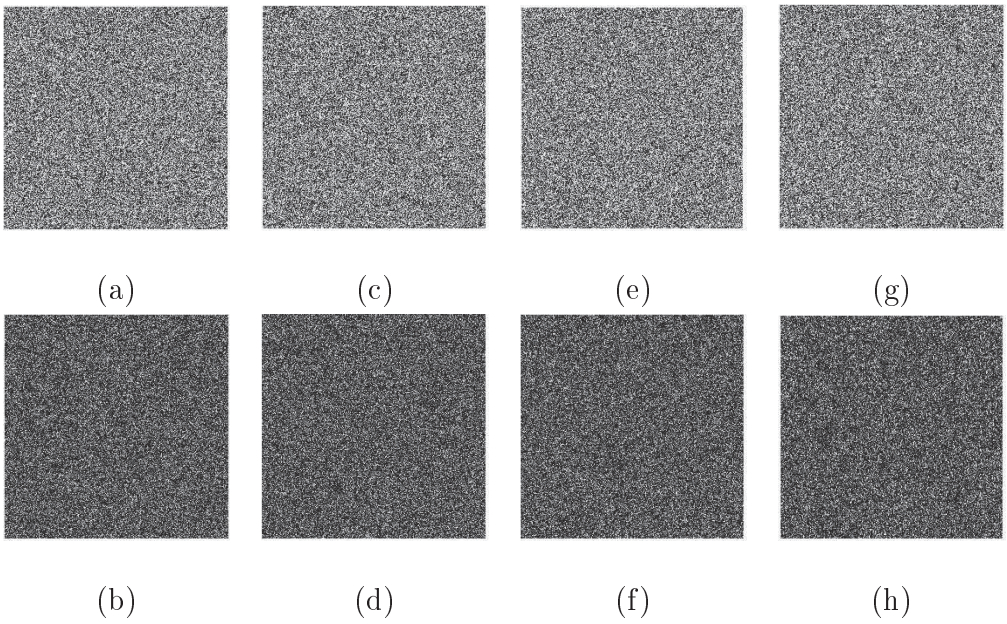


FIGURE 5.14 – Analyse de la sensibilité à la clé : a) Image originale Lena cryptée avec la clé k_1, b) Différence entre deux images cryptées avec les clés k_0 et k_1, c) Image originale Lena cryptée avec la clé k_2, d) Différence entre deux images cryptées avec les clés k_0 et k_2, e) Image originale Lena cryptée avec la clé k_3, f) Différence entre deux images cryptées avec les clés k_0 et k_3, g)Image originale Lena cryptée avec la clé k_4, h) Différence entre deux images cryptées avec les clés k_0 et k_4.

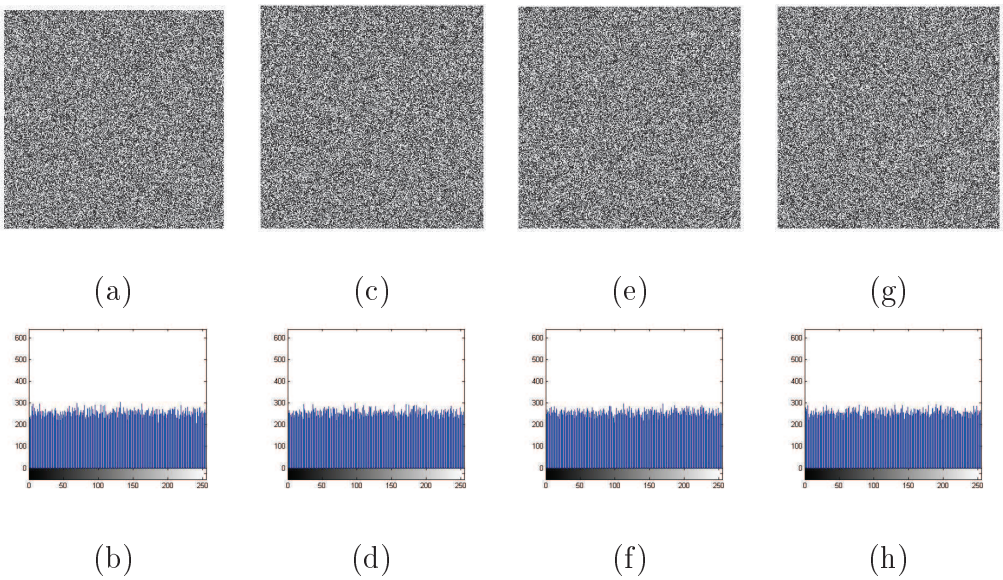


FIGURE 5.15 – Analyse de la sensibilité à la clé : a) Image décryptée avec la clé k_1, b) Son histogramme, c) Image décryptée avec la clé k_2, d) Son histogramme, e) Image décryptée avec la clé k_3, f) Son histogramme, g) Image décryptée avec la clé k_4, h) Son histogramme.

5.4 Un nouveau système de crypto-compression d'images

5.4.1 Introduction

Actuellement, la transmission de données devient de plus en plus importante dans la communication interpersonnelle. Ces données peuvent être de différentes natures, du texte, des images ou bien des vidéo, et qui nécessitent très souvent un espace de stockage et une bande passante très importante pour les sauvegarder et les transmettre à travers des réseaux de communication. De plus, certaines données sont considérées comme confidentielles et il ne faut accorder l'accès qu'aux entités autorisées. De ce fait, la compression et le cryptage des données deviennent de plus en plus indispensables dans les systèmes de communications modernes.

Il n'est pas sans intérêt de rappeler que dans la section 3.8 nous avons introduit la notion de crypto-compression et nous avons énoncé qu'il est préférable d'appliquer le chiffrement dans le codage entropique, ce qui conduit à conserver l'efficacité du codage et à ne pas introduire des temps de calculs très importants.

Il est à noter que le développement d'un système combinant la compression et le cryptage reste une tâche très délicate. D'ailleurs, nous trouvons aujourd'hui plusieurs systèmes de compression [Chuang 1998, Carpentieri , Sudharsanan 2000, Xiaolin 1996] et qui ne sont pas capables de réaliser à la fois la compression et le cryptage. Toutefois, nous pouvons trouver dans la littérature quelques propositions de systèmes de crypto-compression mais ils ne sont pas toujours efficaces car ils sont vulnérables à certaines attaques. Notons que la plupart des crypto-systèmes proposés dans la littérature sont basés sur le codage arithmétique (CA), vu qu'il est efficace et qu'il est présent dans les derniers standards et normes de compression.

En effet, Cleary *et al.* [Cleary 1995] ont considéré le CA comme un algorithme de chiffrement et ils ont montré qu'il est vulnérable contre l'attaque par texte clair choisi et l'attaque par texte clair connu. De plus, Bergen et *al.* [Helen 1993] ont étudié la sécurité des données fournies par un CA adaptatif (CAA) et ils ont proposé de réinitialiser et d'ajuster les paramètres du modèle adaptatif d'une manière régulière ce qui conduit à améliorer la sécurité de ce codeur. Néanmoins, il reste vulnérable contre l'attaque par texte clair choisi. Plus tard, Wen *et al.* [Wen 2006] ont conçu un CA binaire (CAB) avec fractionnement pseudo-aléatoire du vecteur de probabilité cumulative utilisé pendant le

codage/décodage. Le fractionnement se fait grâce à un flux de clés généré d'une manière pseudo-aléatoire. Ainsi, grâce à ce mécanisme, il devient possible de réaliser à la fois la compression et le cryptage. Toutefois, Kim *et al.* [Kim 2007] ont montré que la technique proposée par [Wen 2006] n'est pas robuste contre l'attaque par texte clair choisi et l'attaque par texte clair connu. Kim *et al.* ont proposé une amélioration du mécanisme de fractionnement pseudo-aléatoire d'intervalle en permutant les symboles dans la séquence à coder.

Il est à noter que grâce à cette permutation, la complexité de l'algorithme de chiffrement augmente et en contre partie la performance du CA diminue, surtout s'il est combiné avec un modèle adaptatif ou avec contexte qui cherche à exploiter la redondance entre les symboles voisins. Zhou *et al.* [Zhou 2008] ont proposé d'améliorer la solution proposée par [Kim 2007] en supprimant l'étape de permutation et en utilisant l'opérateur ou exclusif dans le schéma de cryptage. Zhou *et al.* ont proposé également un nouveau schéma de crypto-compression qui est très rapide et qui s'applique d'une manière sélective. Récemment, Mi *et al.* [Mi 2008] ont conçu également un nouveau système de cryptage basé sur la permutation pseudo-aléatoire des intervalles d'un CA en utilisant le système chaotique Logistic-map pour la génération des flux de clés. Cependant, le LM présente des défauts de sécurité puisqu'il est caractérisé par une densité non uniforme et un espace de clés très réduit [Alvarez 2003, Alvarez 2009]. D'après les méthodes étudiées, nous remarquons que la plupart des systèmes de crypto-compression présentent des défauts de sécurité et des dégradations dans les performances des méthodes de compression.

Ainsi, nous proposons un nouveau schéma de crypto-compression qui est basé sur la combinaison du CA avec un GNPA. Nous proposons d'utiliser notre GNPA, qui est basé sur le CSM et ECF-map, pour la génération des flux de clés dynamiques qui seront utilisées pour sécuriser le CA. De plus, nous proposons d'utiliser un CA binaire (CAB) grâce à sa capacité de coder tout type de données et de réduire les temps de calculs relatifs à la mise à jour des probabilités dans un schéma de codage adaptatif.

Nous introduisons, section 5.4.2, les notions de base du CAB qui a la capacité d'être utilisé pour coder tout type de données et qui réduisent la complexité des mises à jour des probabilités pour un modèle adaptatif. La section 5.4.3 présente notre nouvelle méthode de crypto-compression. Puis, la section 5.4.4 détaille les résultats expérimentales qui montrent que notre méthode conserve les performances du CAB en terme de taux de compression, et assure un transfert dans un temps raisonnable. Enfin, nous terminons cette section par une conclusion, section 5.4.5.

5.4.2 Le codage arithmétique binaire

le CA [Langdon 1984, Witten 1987, Howard 1994, Moffat 1998] est un codeur entropique qui est largement utilisé dans les derniers standard de compression comme JPEG2000, JBIG, JBIG2 et H.264/AVC. Son principe consiste à présenter une séquence de symboles par un nombre réel de l'intervalle $[0, 1)$, comme décrit chapitre 2. Durant le processus de codage/décodage, le CA utilise un modèle statistique qui estime les probabilités des symboles à coder. A partir de ces probabilités, nous définissons un vecteur de probabilité cumulative VPC afin d'accorder à chaque symbole S_i un sous intervalle de l'intervalle $[0, 1]$ dont la longueur est égale à sa probabilité p_i. Ainsi, pour chaque nouveau symbole S_i, le CA propose de continuer avec le sous intervalle accordé à ce symbole et qui se réduit au fur et à mesure du codage de la séquence d'entrée.

Le CA peut être soit statique, soit adaptatif autorisant au codeur la capacité de s'adapter à la variation statistique de la séquence à coder. Le modèle adaptatif est plus performant, toutefois il est plus lent à cause des mises à jour des probabilités durant le processus de codage/décodage. Il existe plusieurs types de CA, le plus important étant le CAB. En effet, le CAB est capable, d'une part, de réduire la complexité des mises à jour des probabilités dans un schéma de codage adaptatif, et d'autre part, de coder tout type de symboles. Pour un CAB, le VPC s'écrit sous la forme $[0, p_0, 1]$ avec p_0 est égale à la probabilité du symbole 0. Ainsi, l'intervalle $[0, 1]$ est divisé en deux parties pour représenter les deux symboles binaires 0 et 1.

De plus, le sous intervalle associé au symbole 0 est $[0, p_0)$ et celui associé au symbole 1 est $[p_0, 1)$. Les algorithmes 7 et 8 présentent respectivement les processus de codage et le décodage pour le CAB. Les variables *base* et *taille* représentent respectivement la limite inférieure et la taille du sous intervalle calculé à chaque étape du codage/décodage. Les fonctions *mise_echelle_codage*() et *mise_echelle_decodage*() sont utilisées pour assurer la mise à l'échelle de l'intervalle de codage/décodage, alors que la fonction *retenu*() est utilisée pour la gestion des retenus.

5.4.3 La méthode de crypto-compression proposée

Nous supposons que la séquence binaire à coder est représentée par $B = b_1 \cdots b_n$. Soient p_0 et p_1, les probabilités du symbole 0 et 1 respectivement. Nous proposons d'utiliser un flux des clés noté par $\{k_j\}_{j=1}^{n}$ pour échanger aléatoirement les deux sous intervalles du VPC utilisé par le CAB durant les processus de codage et de décodage. Ainsi, avant de

Algorithme 7 Algorithme de codage d'un CAB

```
1: Initialiser base ← 0, taille ← 2^N − 1
2: pour i ← 1 to n faire
3:    x ← taille × p(0)
4:    si b_i = 0 alors
5:       taille ← x
6:    sinon
7:       base0 ← base
8:       base ← base + x
9:       taille ← taille − x
10:      si base0 > base alors
11:         retenu()
12:   si taille < taille_min alors
13:      mise_echelle_codage()
```

Algorithme 8 Algorithme de décodage d'un CAB

```
1: Initialiser base ← 0, taille ← 2^N − 1, code = 4 premiers octets du fichier compressé
2: tant que non (fin du fichier compressé) faire
3:    x ← taille × p(0)
4:    si code ≥ x alors
5:       b_i ← 1
6:    sinon
7:       b_i ← 0
8:    si b_i = 0 alors
9:       taille ← x
10:   sinon
11:      code ← code − x
12:      taille ← taille − x
13:   si taille < taille_min alors
14:      mise_echelle_decodage()
15:   output b_i
```

coder le bit b_j de la séquence B, nous proposons de générer à partir d'un GNPA, la $j^{\text{ème}}$ clé k_j. De plus, nous proposons d'utiliser dans les algorithmes de cryptage et de décryptage, présentés respectivement par les algorithmes 9 et 10, deux nouvelles variables notées par *lower* et *upper* et qui sont initialement égaux à 0 et 1 respectivement. Dans le cas où la valeur de la $j^{\text{ème}}$ clés k_j est égale à 1, alors nous proposons de permuter ces deux variables et de modifier le VPC à $[0, p_1, 1]$. Ainsi, la permutation de l'emplacement des deux sous intervalles associés aux symboles 0 et 1 avec leurs probabilités respectives p_0 et p_1, dépend de la clé k_j. Ceci est très important du fait que la séquence k_j est pseudo-aléatoire et par conséquent les permutations s'effectuent d'une manière pseudo-aléatoire. Il est à noter que la séquence k_j est générée en utilisant le même principe décrit section 5.2.5 tout en lui introduisant une petite modification. Cette dernière vise à améliorer la sécurité de notre algorithme de cryptage et elle consiste à lier le GNPA avec le texte clair. En effet, la procédure adoptée pour la génération des flux de clés k_j est représentée par les étapes suivantes :

- **1** Initialement, le CSM génère, avant le codage/décodage du $j^{\text{ème}}$ bit b_j de la séquence B, un nouveau couple (x_j, y_j).
- **2** Ensuite, nous proposons de calculer pour chaque bit b_j, le nombre décimale S_j dont la représentation binaire est égale à $b_{j-8}...b_{j-1}$. Il est à signaler que pour les 8 premiers bits de la séquence B, S_j est égale à S_0, avec S_0 est un nombre décimale compris entre 0 et 255 et il fait partie de la clé de cryptage. Ensuite nous proposons de calculer la séquence $\{a_j\}_{j=1}^{n}$ en utilisant le principe suivant :

$$a_j = (y_j + \frac{S_j}{256}) - \lfloor (y_j + \frac{S_j}{256}) \rfloor. \tag{5.19}$$

- **3** Finalement, nous obtenons la séquence de clés $\{k_j\}_{j=1}^{n}$ en appliquant l'équation 5.12.

5.4.4 Analyse des performances de l'algorithme proposé

Après avoir présenté notre méthode dans le détail, nous présentons les résultats obtenus sur des données multimédia. Ainsi, nous proposons de coder les plans de bits des images en niveaux de gris (8 bits par pixel) et avec différentes tailles. Le tableau 5.15 présente les résultats de compression des plans de bits de l'image Lena obtenus en utilisant le CAB et notre approche dans le cas d'un modèle statique et d'un modèle adaptatif d'ordre 0. Nous remarquons d'après ces modèles que notre approche conserve les performances du CAB.

Algorithme 9 Algorithme de cryptage

Initialiser $base \leftarrow 0$, $taille \leftarrow 2^N - 1$, $lower \leftarrow 0$, $upper \leftarrow 1$,
pour $i \leftarrow 1$ to n **faire**
  générer k_i en utilisant notre GNPA
  si $k_i = 1$ **alors**
    $permute(lower, upper)$
  $x \leftarrow taille \times p(lower)$
  si $b_i = lower$ **alors**
    $taille \leftarrow x$
  sinon
    $base0 \leftarrow base$
    $base \leftarrow base + x$
    $taille \leftarrow taille - x$
    si base0 $>$ base **alors**
      retenu()
  si taille $<$ taille_min **alors**
    mise_echelle_codage()

D'ailleurs, à chaque étape de codage/décodage, nous utilisons les mêmes probabilités avec ou sans cryptage. Il y a un autre facteur très important pour évaluer la qualité d'un système de crypto-compression qui est le temps d'exécution. De ce fait, nous proposons d'analyser la rapidité de notre algorithme en utilisant une machine Intel core 2 Duo 2.93 GHZ CPU avec 2 Go de RAM fonctionnant sur Windows XP. Le tableau 5.16 présente les temps d'exécution (en seconde) de notre approche appliquée sur les plans de bits de plusieurs images en niveaux de gris et de différentes tailles. De plus, nous présentons, figure 5.16, la trajectoire du texte chiffré pour les modèles statiques et adaptatifs. D'après cette figure, nous remarquons que les symboles du texte chiffré sont uniformément répartis et, par conséquent, ils ne fourniront aucune information qui peut être exploitée dans une attaque par analyse statistique.

5.4.5 Conclusion

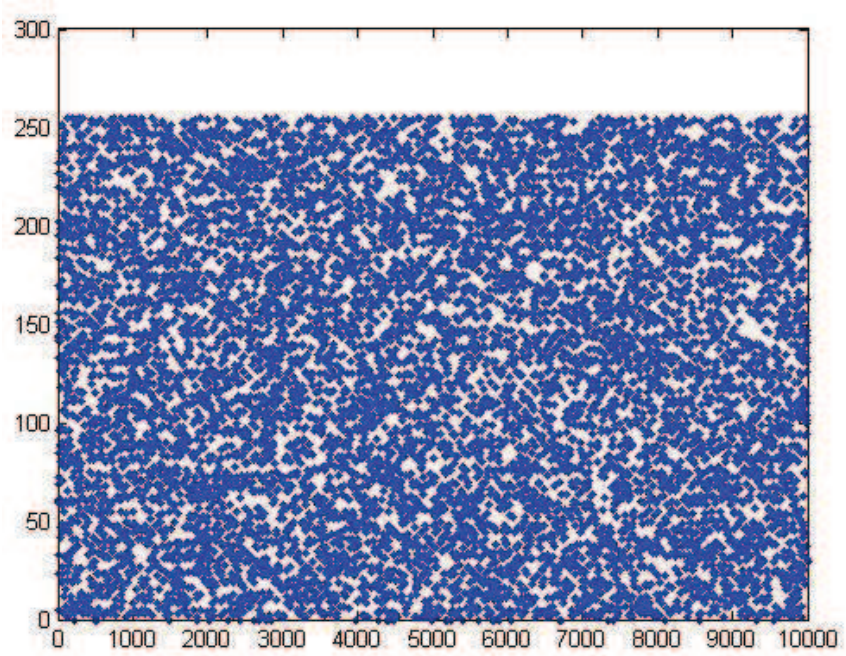
La section 5.4 nous a permis d'exposer notre méthode de crypto-compression. L'originalité de cette méthode est qu'elle réalise à la fois la compression et le cryp-

Algorithme 10 Algorithme de décryptage

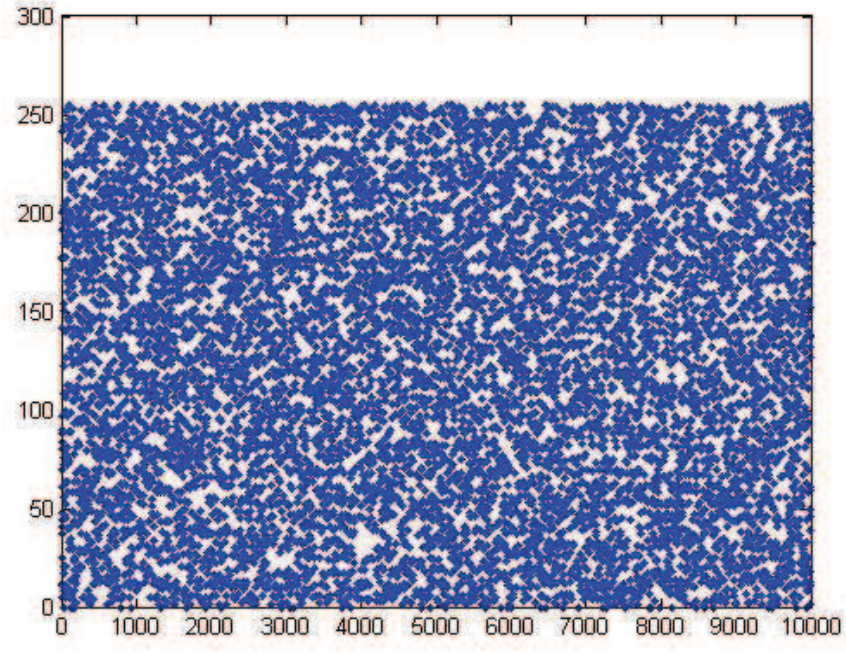
Initialiser $base \leftarrow 0$, $taille \leftarrow 2^N - 1$, code = 4 premiers octets du fichier compressé
tant que non (fin du fichier compressé) **faire**
  générer k_i using the PRBG
  si $k_i = 1$ en utilisant notre GNPA **alors**
    $permute(lower, upper)$
  $x \leftarrow taille \times p(lower)$
  si code $\geq x$ **alors**
    $b_i \leftarrow upper$
    $code \leftarrow code - x$
    $taille \leftarrow taille - x$
  sinon
    $b_i \leftarrow lower$
    $taille \leftarrow x$
  si taille $<$ taille_min **alors**
    mise_echelle_decodage()
  output b_i

TABLE 5.15 – Le résultat de la compression (taille en octet) des plans de bits possédant des entropies différentes.

Les plans de bits de l'image	Modèle statique		Modèle adaptatif	
Lena 512×512 pixels	CAB	Notre méthode	CAB	Notre méthode
plan de bit n° 8	32780	32780	27622	27622
plan de bit n° 7	32182	32182	30085	30085
plan de bit n° 6	32786	32786	31151	31151
plan de bit n° 5	32790	32790	32295	32295



(a) Modèle statique



(b) Modèle adaptatif

FIGURE 5.16 – Etude de l'uniformité des 10000 premiers bits de l'image cryptée de Lena pour a) Modèle statique et b) Modèle adaptatif.

TABLE 5.16 – La vitesse de traitement de notre méthode appliquée pour les modèles statiques et adaptatifs.

Taille de l'image en pixels	Temps total (en s)	
	modèle statique	modèle adaptatif
256×256	2.30	3.00
512×512	9.23	12.00
1024×1024	34.30	45.50

tage [Masmoudi 2010d]. L'algorithme de compression utilisé est le CAB qui présente l'avantage d'être appliqué avec plusieurs modèles statistiques : statique, adaptatif et avec contexte, et d'être capable de comprimer sans perte n'importe quelles types de données. Quant au cryptage, il est effectué grâce à notre nouveau GNPA. L'efficacité de ce GNPA et sa capacité de générer des séquences de nombres pseudo-aléatoires, nous a aidés à échanger aléatoirement les sous intervalles du VPC utilisés par le CAB durant les processus de codage et de décodage. Cet échange est réalisé tout en conservant les probabilités des symboles à coder, ce qui nous a permis de maintenir les performances du CAB.

5.5 Conclusion

Dans ce chapitre, nous avons décrit dans le détail un nouveau GNPA. Le générateur proposé est basé sur la combinaison du système chaotique CSM avec la fonction ECF-map. Pour analyser les performances de notre GNPA, nous avons utilisé la série de tests statistiques proposée par NIST. D'après les résultats obtenus, nous avons montré que le GNPA proposé est très efficace et peut être utilisé dans plusieurs domaines scientifiques ou industriels. En effet, nous avons proposé d'appliquer notre GNPA en cryptographie par le développement d'un nouvel algorithme de chiffrement d'images. Nous avons montré que notre cryptosystème est robuste à plusieurs attaques et il est très approprié pour le chiffrement d'images. Finalement, nous avons proposé dans ce chapitre un nouveau système de crypto-compression. Le schéma proposé est basé sur la combinaison d'un codeur arithmétique binaire avec notre GNPA, et il présente l'avantage de conserver les performances du codeur arithmétique employé, d'une part, et il nécessite des temps de calcul très raisonnables, d'autre part. Les travaux développés dans ce chapitre ont fait

l'objet de plusieurs publications [Masmoudi 2010b, Masmoudi 2010d, Masmoudi 2010c].

Chapitre 6

Conclusion et perspectives

La nécessité de compresser et de crypter les images apparaît suite à l'évolution spectaculaire de la production et de l'utilisation de ces dernières dans plusieurs domaines. Ainsi, le travail développé s'est articulé autour de trois axes principaux à savoir la compression, le cryptage et la crypto-compression. La compression vise à réduire la taille des images archivées et transmises afin d'augmenter la capacité de stockage et de réduire le temps de transmission tout en conservant les mêmes supports physiques. Ainsi, nous avons rappelé les algorithmes et les standards de compression les plus couramment utilisés et nous avons porté un intérêt particulier pour la compression sans perte d'images. La plupart des schémas de compression étudiés sont basés sur le codage arithmétique, grâce à son efficacité en terme de taux de compression, sa facilité de le mettre en œuvre et sa rapidité. Ce dernier type de codage a été le socle des techniques de codages développées dans nos travaux de recherches.

L'autre problème étudié est celui du chiffrement. Ainsi, il est communément admis que les techniques de chiffrement standard ne conviennent pas au cas particuliers des images. Les systèmes de chiffrement asymétrique, par exemple, sont très coûteux en temps de calcul. Les systèmes de chiffrement par bloc présentent également l'inconvénient que tous les blocs identiques de l'image originale sont également identiques après chiffrement, et par conséquent l'image cryptée contient des zones texturées et son entropie n'est pas maximale. Ainsi, pour certains domaines d'applications où le facteur temps est très important, comme en télévision numérique ou en télémédecine, et dans un environnement où les erreurs sont fréquentes, les algorithmes de chiffrement par flot apportent la solution et ils garantissent une sécurité maximale. Le chiffrement par flot est une approximation de la sécurité parfaite et il est basé sur la génération déterministe des flux de clés dynamiques. Les flux de clés générés doivent satisfaire certaines exigences pour être employés en cryptographie, comme l'aléa, l'imprédictibilité et la non périodicité. Ainsi, les systèmes chaotiques s'imposent pour avoir des flux de clés parfaitement pseudo-aléatoires afin d'assurer une sécurité maximale. Nous avons effectué quelques rappels sur les systèmes de chiffrement

standard, la synthèse et la cryptanalyse des systèmes de chiffrement d'images et l'évaluation des générateurs des séquences pseudo-aléatoires. Après avoir étudié les techniques de compression et de chiffrement, l'objectif souhaité de notre travail de recherche est de fournir un schéma de crypto-compression basé sur les systèmes chaotiques et le codage arithmétique afin d'assurer le stockage et le transfert sécurisé des donnés images.

Dans ce travail, nous avons commencé à étudier et analyser les nouveaux soucis, liés à la compression et au cryptage d'images, et nous avons exposé par la suite nos méthodes de compression, de cryptage et de crypto-compression appliquées dans le domaine de l'imagerie numérique. Nous avons validé théoriquement certaines méthodes, par contre d'autres méthodes ont été validées grâce à nombreuses expérimentations.

Notre première contribution, qui s'inscrit dans le cadre de la compression sans perte d'images, consiste en un nouveau codage arithmétique adaptatif, qui est plus efficace et plus rapide que les codeurs arithmétiques conventionnels. La méthode proposée est basée sur l'ajout d'une étape de prétraitement sur l'ensemble des blocs de l'image afin de les trier selon la moyenne : du plus clair vers le plus sombre. Ensuite, nous avons proposé d'utiliser un nouveau modèle statistique qui commence à coder les pixels avec leur vraie probabilité et de mettre à jour ces probabilités après le codage/décodage de la dernière occurrence de chaque pixel.

Notre deuxième contribution, repose sur l'amélioration du codage arithmétique adaptatif d'ordre 0 (CAA-0). Ce dernier présente l'avantage de coder l'image en une seule passe, et il peut être considéré comme un codeur universel en codant toute sorte de données sans connaissance *à priori* de ses statistiques. Ainsi, nous avons montré, expérimentalement, que le CAA-0 devient plus efficace si nous utilisons une précision réduite pour représenter les fréquences des pixels à coder. Ceci s'explique par le fait que le modèle statistique met à jour les probabilités après le codage/décodage d'un nombre réduit de pixels. Ce choix donne la possibilité au CAA-0 de s'adapter rapidement à la variation statistique inter-blocs. En outre, nous avons proposé de coder l'image bloc par bloc après les avoir triés selon la distance de Kullback. Ce principe est très efficace, et les blocs successifs sont caractérisés, dans ce cas, par des lois de probabilités similaires conduisant à coder chaque bloc avec une excellente approximation de sa vraie table de probabilités.

Notre troisième contribution se situe dans le cadre de la génération des nombres pseudo-aléatoires (GNPA). Ce type de générateur est essentiel pour de nombreuses utilisations scientifiques et industrielles. Le générateur proposé est basé sur la combinaison d'un système chaotique avec les fractions continues, et il a été validé en se basant sur la

série de tests de NIST qui est la plus populaire et la plus couramment utilisée dans la qualification d'un bon générateur.

Notre quatrième contribution est fondée sur le développement d'un nouvel algorithme de chiffrement d'images. Bien que les algorithmes de chiffrement standard soient sécurisés, ils ne sont pas très appropriés pour le cryptage d'images puisqu'ils opèrent sur des blocs de pixels et nécessitent des temps de calcul très élevés. Le nouvel algorithme proposé est fondé sur le principe de mélanger le texte clair avec des flux de clés générés par le GNPA proposé. L'algorithme de chiffrement proposé a été analysée dans le détail, et les résultats expérimentaux obtenus sur des images réelles montrent sa robustesse contre plusieurs types d'attaques.

Enfin, nous avons proposé une nouvelle méthode de crypto-compression basée sur l'utilisation du codage arithmétique binaire (CAB) avec le GNPA proposé. Ainsi, nous avons proposé de garder les mêmes probabilités calculées par l'étape de modélisation statistique, durant les processus de codage et de décodage, tout en échangeant aléatoirement les limites inférieures et supérieures associées à chaque symbole. Cela nous a permis de conserver les performances du CAB et de rendre le décodage impossible sans la possession de la clé secrète. Nous avons illustré les apports de la méthode proposée pour les modèles statiques et adaptatifs.

Il est toujours possible d'améliorer les travaux élaborés dans cette thèse, et de s'ouvrir vers des contributions futures. Loin d'être exhaustif, nous proposons de lister quelques idées qui peuvent contribuer au développement de nouvelles solutions.

Pour la première méthode de compression proposée, une première amélioration serait d'utiliser plusieurs critères de tris, autre que la moyenne, ou bien même d'effectuer une recherche exhaustive sur tous les parcours et de maintenir, pour chaque image, le parcours le plus efficace. Cependant, ce type de traitement peut engendrer des temps de calcul très élevés, ce qui peut diminuer l'efficacité de notre méthode en terme de temps. En outre, au lieu de découper l'image en blocs carrés, nous envisageons de la découper en blocs de différentes tailles ce qui peut conduire à mieux ordonner les blocs des pixels à coder. De plus, dans notre étude expérimentale, nous avons utilisé une seule table de probabilités, par contre, nous pouvons imaginer de grouper les blocs triés en plusieurs classes, et de coder/décoder chaque classe avec sa propre table de probabilités tout en exploitant le principe proposé pour la mise à jour des probabilités. Il est également possible d'appliquer la technique proposée pour le codage par bloc d'images après les avoir triés selon la moyenne, et la mise à jour des probabilités après le codage de la dernière occurrence de

chaque symbole avec le CAA-0. Avec cette solution, nous pouvons non seulement retirer les probabilités des symboles totalement codés du vecteur de probabilités cumulatives, mais aussi de s'adapter à la variation dynamiques des statistiques de l'image à coder.

Concernant notre deuxième méthode de compression, elle se déroule en deux passes, une première passe pour le tri, alors que la deuxième passe se charge du codage. Ainsi, nous pouvons envisager de créer des classes pour les distributions de probabilités des blocs de pixels, afin de pouvoir coder les images en une seule passe. L'idée consiste à calculer dynamiquement un ensemble de tables de probabilités tout en indiquant pour chaque bloc la table de probabilités utilisée pour son codage et qui est la proche possible, en terme de la distance de Kullback, avec sa vraie table de probabilités. Grâce à ce principe, nous pouvons réduire de façon significative les temps de calculs relatifs au tri des blocs.

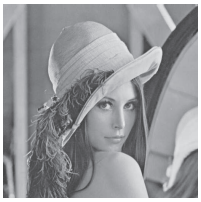
Le développement d'autres techniques de crypto-compression en utilisant différents types de codeurs arithmétiques est une autre piste d'investigation. Nous pouvons envisager d'appliquer la technique proposée pour échanger aléatoirement les deux sous intervalles associés aux symboles binaires à coder sur d'autres types de codeur arithmétique, comme le codage arithmétique binaire adaptatif et avec contexte employé dans la norme de compression JPEG2000. Nous pouvons également envisager d'appliquer un cryptage sélectif par région d'intérêt pour des images au format JPEG2000. Une autre suggestion pour la crypto-compression des données multimédia est d'utiliser un codage arithmétique qui code directement les pixels de l'image sans avoir besoin de les transformer en des données binaires. Dans cette perspective, il faut pouvoir minimiser les échanges au niveau des sous intervalles associés aux symboles à coder pour avoir une solution qui soit à la fois sécurisée et rapide.

Annexe A

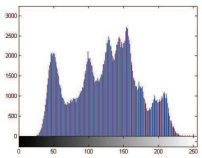
Liste des images

Table A.1 – Les images de tests utilisées dans les expérimentations. La première colonne présente les noms des images. La deuxième colonne présente la taille de chaque image, et la troisième colonne présente l'entropie.

Image	Taille en pixels	Entropie en bits/pixel
Barbara	512×512	7.47
Boat	512×512	7.12
France	672×496	6.28
Frog	621×498	4.97
Goldhill	512×512	7.48
Lena	512×512	7.45
Library	464×352	5.85
Mandrill	512×512	7.36
Moutains	640×480	6.22
Peppers	512×512	7.57
Washsat	512×512	2.87
Zelda	512×512	7.27

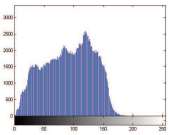


Lena



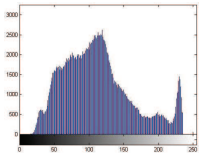
Histogramme de Lena

Zelda



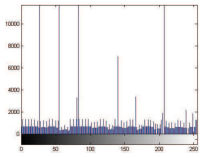
Histogramme de Zelda

Goldhill



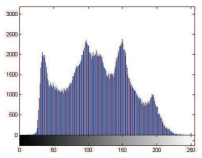
Histogramme de Goldhill

France



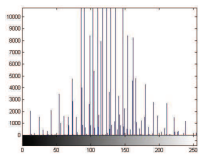
Histogramme de France

Barbara



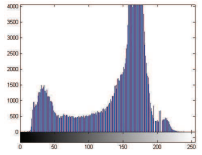
Histogramme de Barbara

Frog

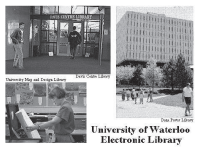


Histogramme de Frog

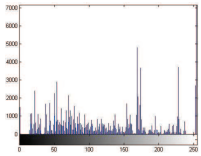
Boat



Histogramme de Boat

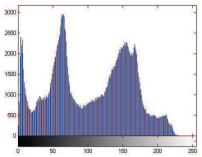


Library

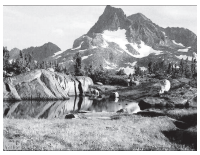


Histogramme de Library

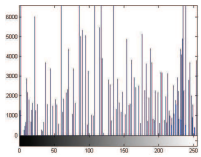
Peppers



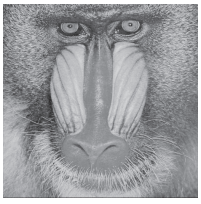
Histogramme de Peppers



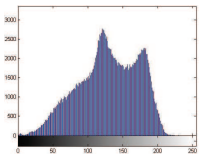
Mountain



Histogramme de Mountain

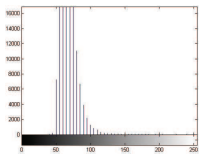


Mandrill



Histogramme de Mandrill

Washsat



Histogramme de Washsat

FIGURE A.1 – Les images de tests utilisées dans les expérimentations ainsi que leurs histogrammes.

Annexe B

Liste des publications

B.1 Revues

1. A. Masmoudi, W. Puech and M.S. Bouhlel, An Efficient PRBG Based on Chaotic Map and Engel Continued Fractions, Journal of Software Engineering and Applications, 2010.
2. A. Masmoudi, W. Puech and M.S. Bouhlel, A New Joint Lossless Compression And Encryption Scheme Combining A Binary Arithmetic Coding With A Pseudo Random Bit Generator, LJS Publisher and IJCSIS Press, vol.8, n.1, 2010.
3. A. Masmoudi, W. Puech and M.S. Bouhlel, Efficient adaptive arithmetic coding based on updated probability distribution for lossless image compression, J. Electronic Imaging, vol.19, n.2, 2010.
4. A. Masmoudi, W. Puech and M.S. Bouhlel, A new cryptosystem based on chaotic map and continued fractions, soumis à The Journal of Systems and Software, Avril 2010.

B.2 Conférences internationales

1. A. Masmoudi, W. Puech and M.S. Bouhlel, A New Image Cryptosystem Based on Chaotic Map and Continued Fractions, 18th European Signal Processing Conference EUSIPCO, Aalborg, Denmark, 2010.
2. A. Masmoudi, W. Puech and M.S. Bouhlel, A Generalized Continued Fraction-Based Asynchronous Stream Cipher For Image Protection, 17th European Signal Processing Conference EUSIPCO, Glasgow, Scotland, 2009.
3. A. Masmoudi, M.S. Bouhlel and W. Puech, A new model for arithmetic coding based on probability distribution and scan order of the encoded data : application

to medical imagery, 2ème Conférence Internationale : E-MEDISYS'08, Sfax, Tunisie, 2008.

4. A. Masmoudi et M.S. Bouhlel, Un nouveau modèle de codage arithmétique basé sur l'exploitation des probabilités et de la répartition spatiale des symboles à coder, 4ème Conférence Internationale : Sciences Electronique, Technologie de l'Information et des Télécommunications SETIT'07, Hammamet, Tunisie, 2007.

Bibliographie

[Abramowitz 1968] M. Abramowitz et I. Stegun. *Handbook of Mathematical Functions*. Applied Mathematics Series,Dover Publications, New York, vol. 55, pages 2286–2293, 1968. 103

[Abramson 1963] N. Abramson. Information Theory and Coding. McGraw-Hill Book Company, Inc., New York, NY, USA, 1963. 22, 69

[Alvarez 2003] G. Alvarez, F. Montoya, M. Romera et G. Pastor. *Cryptanalysis of a Discrete Chaotic Cryptosystem Using External Key*. Physics Letters, vol. 9, pages 319–334, 2003. 89, 129

[Alvarez 2009] G. A. Alvarez et L. B. Shujun. *Cryptanalyzing a Nonlinear Chaotic Algorithm (NCA) for Image Encryption*. Communications in Nonlinear Science and Numerical Simulation, vol. 14, no. 11, pages 3743–3749, 2009. 49, 89, 129

[Behina 2008] S. Behina, A. Akhshani, H. Mahmodi et A. Akhavan. *A Novel Algorithm for Image Encryption on Mixture of Chaotic Maps*. Chaos, Solitions and Fractals, vol. 35, pages 408–419, 2008. 122

[Biham 1990] E. Biham et A. Shamir. *Differential Cryptanalysis of DES-like Cryptosystems*. In CRYPTO, pages 2–21, 1990. 46

[Bovik 2009] A. Bovik. Academic Press is an imprint of Elsevier, California, USA, 2009. 6

[Burger 2009] W. Burger et M. J. Burge. Springer ; 1st Edition, 2009. 7

[Burrows 1994] M. Burrows et D. J. Wheeler. *A Block-Sorting Lossless Data Compression Algorithm*. Rapport technique 124, 1994. 18

[Carpentieri] B. Carpentieri, M. J. Weinberger et G. Seroussi. *Lossless Compression of Continuous-Tone Images*. Proceedings of the IEEE. 54, 128

[Carpentieri 1997] B. Carpentieri. *A New Lossless Image Compression Algorithm Based on Arithmetic Coding*. In Proceedings of the 9th International Conference on Image Analysis and Processing-Volume II, pages 54–61, 1997. 54, 56

[Chen 2004] G. Chen, Y. Mao et C. K. Chui. *A Symmetric Image Encryption Scheme Based on 3D Chaotic Cat Maps*. Chaos, Solitons and Fractals, vol. 21, pages 749–761, 2004. 46, 87, 122

[Chiaraluce 2002] F. Chiaraluce et L. Ciccarelli. *A New Chaotic Algorithm for Video Encryption.* IEEE Transactions on Consumer Electronics, vol. 48, no. 4, pages 838–844, 2002. 122

[Chirikov 1971] B. V. Chirikov. Research concerning the theory of non-linear resonance and stochasticity. CERN, Geneva, 1971. Translated at CERN from the Russian (IYAF-267-TRANS-E). 94

[Chuang 1998] T. J. Chuang et J. C. Lin. *A New Algorithm for Lossless Still Image Compression.* Pattern Recognition, vol. 31, no. 9, pages 1343–1352, September 1998. 54, 128

[Cleary 1995] J. Cleary, S. Irvine et I. Rinsma-Melchert. *On the Insecurity of Arithmetic Coding.* Computers and Security, vol. 14, pages 167–180, 1995. 51, 128

[Cormack 1984] G. V. Cormack et R. N. Horspool. *Algorithms for Adaptive Huffman Codes.* Inf. Process. Lett., vol. 18, no. 3, pages 159–165, 1984. 21

[Cover 1991] T. Cover et J. Thomas. Elements of information theory. Wiley and Sons, 1991. 13, 14

[Curtin 2005] M. Curtin. Springer ; 1 edition, 2005. 37

[Diffie 1976] W. Diffie et M. E. Hellman. *New Directions in Cryptography.* IEEE Transactions on Information Theory, vol. 22, no. 6, pages 644–654, 1976. 42

[Douglas 2002] S. Douglas. Cryptography : Theory and practice, second edition. CRC/C&H, 2002. 33, 37, 38, 44

[Eckert 1998] M. P. Eckert et A. P. Bradley. *Perceptual Quality Metrics Applied to Still Image Compression.* Signal Processing, vol. 70, no. 3, pages 177–200, 1998. 12

[Elias 1975] P. Elias. *Universal codeword sets and representations of the integers.* IEEE Transactions on Information Theory, vol. 21, no. 2, pages 194–203, 1975. 16

[Eskicioglu 1995] A. M. Eskicioglu et P. S. Fisher. *Image Quality Measures and Their Performance.* IEEE Transactions on Communications, vol. 43, no. 12, pages 2959–2965, 1995. 12

[Fallahi 2008] K. Fallahi, R. Raoufi et H. Khoshbin. *An Application of Chen System for Secure Chaotic Communication Based on Extended Kalman Filter and Multi-Shift Cipher Algorithm.* Communications in Nonlinear Science and Numerical Simulation, vol. 13, no. 4, pages 763–781, 2008. 87

[Fano 1949] R. M. Fano. *The Transmission of Information*. Rapport technique 65, Research Laboratory of Electronics, M.I.T., 1949. 20

[Ferguson 2010] N. Ferguson, B. Schneier et T. Kohno. Wiley, 2010. 33

[García 2002] P. García, A. Parravano, M. G. Cosenza, J. Jiménez et A. Marcano. *Coupled Map Networks as Communication Schemes*. vol. 65, no. 4, page 045201, 2002. 87

[Garfinkel 1996] S. Garfinkel et G. Spafford. Practical unix and internet security (2nd ed.). O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1996. 43

[Girod 1993] B. Girod. *What's wrong with mean-squared error ?* pages 207–220, 1993. 12

[Golchin 1998] F. Golchin et K.K. Paliwal. *A Lossless Image Coder with Context Classification, Adaptive Prediction and Adaptive Entropy Coding*. In Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing, pages 2545–2548, 1998. 54

[Golomb 1966] S. W. Golomb. *Run-Length Encodings*. IEEE Transactions on Information Theory, vol. 12, pages 399–401, September 1966. 19

[Gonzalez 2002] R. C. Gonzalez et R. E. Woods. Prentice Hall, New Jersey, 2002. 6

[Gonzalez 2008] R. C. Gonzalez et R. E. Woods. Prentice Hall, New Jersey, 2008. 6

[Gormish 2000] M. J. Gormish, D. Lee et M. W. Marcellin. *Jpeg 2000 : Overview, Architecture and Applications*. In Proceedings of ICIP'2000, pages 29–32, 2000. 17

[Guazzo 1979] M. Guazzo. *A General Minimum-Redundancy Source-Coding Algorithm*. IEEE Transactions on Information Theory, vol. IT-26, pages 15–25, 1979. 22

[Hartono 2002] Y. Hartono, C. Kraaikamp et F. Schweiger. *Algebraic and Ergodic Properties of a New Continued Fraction Algorithm with Non-Decreasing Partial Quotients*. Journal de théorie des nombres de Bordeaux, vol. 14, no. 2, pages 497–516, 2002. 92, 93

[Helen 1993] A. B. Helen et M. H. James. *A Chosen Plaintext Attack On An Adaptive Arithmetic Coding Compression Algorithm*. Computers and Security, vol. 12, pages 157–167, 1993. 51, 128

[Hershey 2002] J. Hershey. McGraw-Hill Professional ; 1 edition, 2002. 33

[Howard 1992] P. G. Howard et J. R. S. Vitter. *Practical Implementations of Arithmetic Coding 1*, 1992. 22

[Howard 1994] P. G. Howard et J. S. Vitter. *Arithmetic Coding for Data Compression*. Proceedings of the IEEE, vol. 82, no. 6, pages 857–865, Jun. 1994. 22, 57, 130

[Huang 2009] C. K. Huang et H. H. Nien. *Multi Chaotic Systems Based Pixel Shuffle for Image Encryption*. Optics Communications, vol. 282, no. 11, pages 2123–2127, 2009. 46

[Huffman 1952] D. A. Huffman. *A Method for the Construction of Minimum-Redundancy Codes*. Proceedings of the Institute of Radio Engineers, vol. 40, no. 9, pages 1098–1101, September 1952. 21

[JBI 1993] *Information TechnologymProgressive Bi-Level Image Compression*. Iso/iec 11544 and itu-t recommendation t.82, 1993. FCD. 28

[JBI 2000] *Bi-Level Image Compression Standard*. Iso/iec 14492 and itu-t recommendation t.88, 2000. FCD. 29

[Johnson 2000] D. H. Johnson et S. Sinanovic. *Symmetrizing the Kullback-Leibler Distance*. Rapport technique, IEEE Transactions on Information Theory, 2000. 71, 77

[Kahn 1967] D. Kahn. The codebreakers : The story of secret writing. New York : Macmillan Publishing Co., 1967. 39

[Kanso 2009] A. Kanso et N. Smaoui. *Logistic Chaotic Maps for Binary Numbers Generations*. Chaos, Solitons and Fractals, vol. 40, pages 2557–2568, 2009. 49, 89

[Karam 2000] L. J. Karam. Lossless coding, chapitre 5.1, pages 461–474. 2000. 15

[Kerckhoffs 1883] A. Kerckhoffs. *La cryptographie militaire*. Journal des sciences militaires, vol. IX, pages 5–83, January 1883. 35, 44

[Khintchin 1963] A. Y. Khintchin. *Continued Fractions*. Noordhoff, Groningen, 1963. 92

[Kim 2007] H. Kim, J. Wen et J. Villasenor. *Secure arithmetic coding*. IEEE Transactions Signal Processing, vol. 55, no. 5, pages 2263–2272, 2007. 51, 129

[Kocarev 2001] L. Kocarev. *Chaos-Based Cryptography : A Brief Overview*. IEEE Circuits and Systems, vol. 1, no. 3, pages 6–21, 2001. 48, 49, 87, 116

[Kullback 1951] S. Kullback et R. A. Leibler. *On Information and Sufficiency*. The Annals of Mathematical Statistics, vol. 22, no. 1, pages 79–86, 1951. 71, 77

[Kullback 1959] S. Kullback. Information theory and statistics. Wiley, New York, 1959. 77

[Kullback 1968] S. Kullback. *Information Theory and Statistics*, 1968. 71, 77

[Kuroki 2004] N. Kuroki, T. Manabe et M. Numa. *Adaptive Arithmetic Coding for Image Prediction Errors*. In Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 04), volume III, pages 961–964, May 2004. 57

[Langdon 1979] G. G. Langdon. *Arithmetic coding*. IBM Journal of Research and Development, vol. 23, pages 149–162, 1979. 22

[Langdon 1983] G. G. Langdon et J. J. Rissanen. *A simple general binary source code*. IEEE Transactions on Information Theory, vol. 29, no. 5, pages 778–, 1983. 29

[Langdon 1984] G. G. Langdon. *An Introduction to Arithmetic Coding*. IBM Journal of Research and Development, vol. 28, no. 2, Mar. 1984. 22, 130

[Li 2001a] S. Li et X. Mou. *Improving Security of a Chaotic Encryption Approach*. Physics Letters A, vol. 290, no. 3-4, pages 127–133, 2001. 49, 87, 116

[Li 2001b] S. J. Li, X. Q. Mou et Y. L. Cai. *Pseudo-Random Bit Generator Based on Couple Chaotic Systems and its Application in Stream-Ciphers Cryptography, 2001, Lecture Notes in Computer Science 2247 (2001), pp. 316-329.* Progress in cryptology-INDOCRYPT, Lecture Notes in Computer Science 2247, pages 316–329, 2001. 87

[Li 2007] P. Li, Z. Li, W. A. Halang et G. Chen. *A Stream Cipher Based on a Spatiotemporal Chaotic System*. Chaos, Solitons and Fractals, vol. 32, no. 7, pages 1867–1876, 2007. 87, 116

[Li 2009] H. Li et J. Zhang. *A Secure and Efficient Entropy Coding Based on Arithmetic Coding*. Communications in Nonlinear Science and Numerical Simulation, vol. 14, no. 12, pages 4304–4318, 2009. 87, 116

[Lorentzen 1992] L. Lorentzen et H. Waadeland. *Continued Fractions with Applications*. North Holland, 1992. 92

[Maniccam 2001] S. S. Maniccam et N. G. Bourbakis. *Lossless Image Compression and Encryption using SCAN*. Pattern Recognition, vol. 34, pages 1229–1245, 2001. 123

[Marsaglia 1997] G. Marsaglia. *DIEHARD : A Battery of Tests of Randomness*. http ://stat.fsu.edu/geo/diehard.html, 1997. 92

[Martin 2004] B. Martin. Codage, cryptologie et applications. Presses Polytechniques et Universitaires Romandes, 2004. 37, 38, 42, 49

[Masmoudi 2009] A. Masmoudi, W. Puech et M. S. Bouhlel. *A Generalized Continued Fraction-Based Asynchronous Stream Cipher For Image Protection*. 2009. 40

[Masmoudi 2010a] A. Masmoudi, W. Puech et M. S. Boublell. *Efficient Adaptive Arithmetic Coding Based on Updated Probability Distribution for Lossless Image Compression.* J. Electronic Imaging, vol. 19, no. 2, page 023014, 2010. 67, 82

[Masmoudi 2010b] A. Masmoudi, W. Puech et M. S. Bouhlel. *An Efficient PRBG Based on Chaotic Map and Engel Continued Fractions.* Journal of Software Engineering and Applications, 2010. 115, 137

[Masmoudi 2010c] A. Masmoudi, W. Puech et M. S. Bouhlel. *A New Image Cryptosystem Based on Chaotic Map and Continued Fractions.* 2010. 124, 137

[Masmoudi 2010d] A. Masmoudi, W. Puech et M. S. Bouhlel. *A New Joint Lossless Compression And Encryption Scheme Combining A Binary Arithmetic Coding With A Pseudo Random Bit Generator.* vol. 8, no. 1, 2010. 136, 137

[Matsuda 2003] I. Matsuda, N. Shirai et S. Itoh. *Lossless Coding Using Predictors and Arithmetic Code Optimized for Each Image.* In Proceedings of Intlernational Workshop Visual Content Processing and Representation, volume 2849, pages 199–207, Sep. 2003. 56

[Menezes 2001] A. J. Menezes, P/ C. van Oorschot et S. A. Vanstone. Handbook of applied cryptography. CRC Press, 2001. 33, 37, 38, 40, 44

[Mi 2008] B. Mi, X. Liao et Y. Chen. *A Novel Chaotic Encryption Scheme Based on Arithmetic Coding.* Chaos, Solitons and Fractals, vol. 38, pages 1523–1531, 2008. 51, 89, 129

[Mitchell 1988] J. L. Mitchell et W. B. Pennebaker. *Software Implementations of the Q-Coder.* IBM Journal of Research and Development, vol. 32, no. 6, pages 753–774, 1988. 29

[Moffat 1998] A. Moffat, R. M. Neal et I. H. Witten. *Arithmetic Coding Revisited.* ACM Transactions on Information Systems, vol. 16, no. 3, pages 256–294, Jul. 1998. 22, 130

[Pasco] R. Pasco. *Source Coding Algorithms for Fast Data Compression.* Rapport technique. 22

[Patidar 2009a] V. Patidar, N. K. Parekk et K. K. Sud. *A New Substitution-Diffusion Based Image Cipher Using Chaotic Standard and Logistic Maps.* Communications in Nonlinear Science and Numerical Simulation, vol. 14, pages 3056–3075, 2009. 49, 99

[Patidar 2009b] V. Patidar et K. K. Sud. *A Novel Pseudo Random Bit Generator Based on Chaotic Standard Map and its Testing*. Electronic Journal of Theoretical Physics, vol. 6, no. 20, pages 327–344, 2009. xi, 49, 87, 91, 92, 107

[Pecora 1990] L. M. Pecora et T. L. Carroll. *Synchronization in chaotic systems*. Physical Review Letters, vol. 64, no. 8, pages 821–824, 1990. 86

[Pennebaker 1988] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon et R. Arps. *An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder*. IBM Journal of Research and Development, vol. 32, no. 6, pages 717–726, 1988. 29

[Pennebaker 1992] W. B. Pennebaker et J. L. Mitchell. Jpeg still image data compression standard. Kluwer Academic Publishers, Norwell, MA, USA, 1992. 17

[Press 1992] W. H. Press. *Numerical Recipes in C : the Art of Scientific Computing*. Cambridge, Cambridge University Press, pages 169–173, 1992. 103

[Rissanen 1976] J. J. Rissanen. *Generalized Kraft Inequality and Arithmetic Coding*. IBM Journal of Research and Development, vol. 20, no. 3, pages 198–203, May 1976. 14, 22

[Rissanen 1979] J. J. Rissanen et G. G. Langdon. *Arithmetic Coding*. IBM Journal of Research and Development, vol. 23, no. 2, pages 149–162, Mars 1979. 22

[Rivest 1983] R. L. Rivest, A. Shamir et L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems*. Commun. ACM, vol. 26, no. 1, pages 96–99, 1983. 42

[Rubin 1979] F. Rubin. *Arithmetic Stream Coding Using Fixed Precision Registers*. IEEE Transactions on Information Theory, vol. IT-25, pages 672–675, 1979. 22, 55

[Rukhin 2008] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray et S. Vo. *Statistical Test Suite for Random and Pseudo Random Number Generators for Cryptographic Applications*. NIST special publication 800-22 Revision 1, 2008. 50, 92, 100, 103

[Russ 2006] J. C. Russ. Information theory and statistics. CRC Press, 2006. 6

[Salomon 2007] D. Salomon. Data compression : The complete reference. pub-SV, 2007. 28, 29, 55, 67

[Schneier 1996] B. Schneier. Wiley ; 2nd edition, 1996. 37, 42

[Seidensticker 1983] R. B. Seidensticker. *Continued Fractions for High-Speed and High-Accuracy Computer Arithmetic*. Proc. 6th IEEE Symp. Comput. Arithmetic, 1983. 92

[Shannon 1948a] C. E. Shannon. *A Mathematical Theory of Communication I*. Bell System Technical Journal), vol. 27, no. 7, pages 379–423, Jul. 1948. 13, 57, 121

[Shannon 1948b] C. E. Shannon. *A Mathematical Theory of Communication II*. Bell System Technical Journal), vol. 27, no. 10, pages 623–656, Jul. 1948. 13, 121

[Shannon 1949] C. E. Shannon. *Communication Theory of Secrecy Systems*. Bell System Technical Journal, vol. 28, pages 656–715, 1949. 47

[Shatheesh 2010] S. I. Shatheesh, P. Devaraj et R. S. Bhuvaneswaran. *Enhanced Substitution-Diffusion Based Image Cipher Using Improved Chaotic Map*. In Vinu V Das et R. Vijaykumar, editeurs, Information and Communication Technologies, volume 101 of *Communications in Computer and Information Science*, pages 116–123. Springer Berlin Heidelberg, 2010. 46

[Short 1997] K. M. Short. *Signal Extraction from Chaotic Communications*. International Journal of Bifurcation and Chaos in Applied Sciences and Engineering7, vol. 7, no. 7, pages 1579–1597, 1997. 87, 116

[Skodras 2001] A. Skodras, C. Christopoulos et T. Ebrahimi. *The JPEG 2000 still image compression standard*. IEEE Signal Processing Magazine, vol. 18, pages 36–58, 2001. 17

[Sudharsanan 2000] S. Sudharsanan et P. Sriram. *Block-based Adaptive Lossless Image Coder*. In International Conference on Image Processing, volume 1, 2000. 54, 128

[Taquet 2010] J. Taquet et C. Labit. *Une introduction a la compression d'images medicales volumiques*. Research Report RR-7324, INRIA, 06 2010. 10

[Taubman 2000] D. Taubman. *High performance scalable image compression with EBCOT*. IEEE Transactions on Image Processing, vol. 9, pages 1158–1170, 2000. 17

[Taubman 2002] D. S. Taubman et M. W. M. Jpeg2000 : image compression fundamentals, standards, and practice. Kluwer Academic Publishers, Boston, 2002. 17, 29

[Uhl 2004] A. Uhl et A. Pommer. Image and video encryption : From digital rights management to secured personal communication (advances in information security). Springer-Verlag TELOS, Santa Clara, CA, USA, 2004. 33, 38

[Vernam 1926] G. Vernam. *Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications*. Journal of the American Institute for Electrical Engineers, vol. 55, pages 109–115, 1926. 39

[Vuillemin 1987] J. Vuillemin. *Exact Real Computer Arithmetic with Continued Fractions*. INRIA Report 760. Le Chesnay, France : INRIA, NOV. 1987. 92

[Wall 1973] H. S. Wall. *Analytic Theory of Continued Fractions*. Chelsea, 1973. 92

[Wallace 1991] G. K. Wallace. *The JPEG still picture compression standard*. Commun. ACM, vol. 34, no. 4, pages 30–44, 1991. 17

[Wan 2011] *A New Chaos-Based Fast Image Encryption Algorithm*. Applied Soft Computing, vol. 11, no. 1, pages 514–522, 2011. 46

[Wang 2004] Z. Wang, A. C. Bovik, H. R. Sheikh et E. P. Simoncelli. *Image Quality Assessment : From Error Visibility to Structural Similarity*. IEEE Transactions on Image Processing, vol. 13, no. 4, pages 600–612, 2004. 12

[Wanga 2009] X. Y. Wanga et Q. Yu. *A Block Encryption Algorithm Based on Dynamic Sequences of Multiple Chaotic Systems*. Communications in Nonlinear Science and Numerical Simulation, vol. 14, no. 2, pages 574–581, 2009. 87

[Wei 2007] J. Wei, X. F. Liao, K. W. Wong et T. Zhout. *Cryptanalysis of Cryptosystem Using Multiple one-Dimensional Chaotic Maps*. Communications in Nonlinear Science and Numerical Simulation, vol. 12, pages 814–22, 2007. 116

[Weinberger 1996] M. J. Weinberger, G. Seroussi et G. Sapiro. *LOCO-I : A Low Complexity, Context-Based, Lossless Image Compression Algorithm*, 1996. 27

[Weinberger 2000] M. J. Weinberger, G. Seroussi et G. Sapiro. *The LOCO-I Lossless Image Compression Algorithm : Principles and Standardization into JPEG-LS*. IEEE Transactions on Image Processing, vol. 9, no. 8, pages 1309–1324, 2000. 27

[Welch 1984] T. A. Welch. *A Technique for High-Performance Data Compression*. IEEE Computer, vol. 17, no. 6, pages 8–19, 1984. 16

[Wen 2006] J.G. Wen, H. Kim et J.D. Vilasenor. *Binary Arithmetic Coding Using Key-Based Interval Splitting*. IEEE Signal Process Lett, vol. 13, no. 2, pages 69–72, 2006. 51, 128, 129

[Witten 1987] I. H. Witten, R. M. Neal et J. G. Cleary. *Arithmetic Coding for Data Compression*. Communications of the ACM, vol. 30, no. 6, pages 520–540, Jun. 1987. 22, 55, 64, 69, 130

[Wong 2008] K. W. Wong, B. S. H. Kwoka et C. H. Yuena. *An Afficient Diffusion Approach for Chaos-Based Image Encryption*. Chaos, Solitons and Fractals, vol. 41, no. 5, pages 2652–2663, 2008. 48, 87, 116

[Wu 2004] X. G. Wu, H. P. Hu B. L. et Zhang. *Analyzing and Improving a Chaotic Encryption Method*. Chaos, Solitons and Fractals, vol. 22, no. 2, pages 367–373, 2004. 48, 87, 116, 120

[Xiaolin 1996] W. Xiaolin. *An Algorithmic Study on Lossless Image Compression*. In Data Compression Conference, pages 150–159. IEEE Computer Society Press, 1996. 54, 128

[Yang 1998] T. Yang, L. B. Yang et C. M. Yang. *Cryptanalyzing Chaotic Secure Communications Using Return Maps*. hysics Letters, Section A : General, Atomic and Solid State Physics, vol. 245, no. 6, pages 495–510, 1998. 87, 116

[Yang 2004] T. Yang. *A Survey of Chaotic Secure Communication Systems*. Journal of Computational Cognition, vol. 2, no. 2, pages 81–130, 2004. 48, 49, 87, 116

[Zhang 2005] L. Zhang, X. Liao et X. Wang. *An Image Encryption Approach Based on Chaotic Maps*. Chaos, Solitons and Fractals, vol. 24, no. 3, pages 759–765, 2005. 48, 87, 116

[Zhou 1997] H. Zhou et X. T. Ling. *Problems With the Chaotic Inverse System Encryption Approach*. IEEE Circuits and Systems, vol. 44, no. 3, pages 268–271, 1997. 87

[Zhou 2008] J. Zhou, O. C. Au, X. Fan et P. H. W. Wong. *Joint security and performance enhancement for secure arithmetic coding.* ICIP, pages 3120–3123, 2008. 129

[Zimmermann 1995a] P. R. Zimmermann. *The Official PGP User's Guide*. Rapport technique, Boston : MIT Press, 1995. 43

[Zimmermann 1995b] P. R. Zimmermann. *PGP Source Code and Internals*. Rapport technique, Boston : MIT Press, 1995. 43

[Ziv 1977] J. Ziv et A. Lempel. *A Universal Algorithm for Sequential Data Compression*. IEEE Transactions on Information Theory, vol. 23, no. 3, pages 337–343, 1977. 16

[Ziv 1978] J. Ziv et A. Lempel. *Compression of Individual Sequences via Variable-Rate Coding*. IEEE Transactions on Information Theory, vol. 24, no. 5, pages 530–536, 1978. 16

Résumé : Actuellement, nous vivons dans une société numérique. L'avènement de l'Internet et l'arrivée du multimédia et des supports de stockage numériques, ont transformé profondément la façon dont nous communiquons. L'image en particulier occupe une place très importante dans la communication interpersonnelle moderne. Toutefois, elle présente l'inconvénient d'être représentée par une quantité d'information très importante. De ce fait, la transmission et le stockage des images soulèvent certains problèmes qui sont liés essentiellement à la sécurité et à la compression d'images. Ce sont ces considérations qui ont guidé cette thèse. En effet, la problématique que nous posons dans cette thèse est de proposer une solution conduisant à la crypto-compression d'images afin d'assurer un archivage et un transfert sécurisés tout en conservant les performances de la méthode de compression utilisée. En effet, nos travaux de recherche ont porté essentiellement sur la compression et le cryptage des images numériques. Concernant la compression, nous avons porté un intérêt particulier au codage arithmétique vu sont efficacité en terme de taux de compression et son utilisation par les nouvelles normes et standards de compression tel que JPEG2000, JBIG, JBIG2 et H.264/AVC. Quant au cryptage, nous avons opté pour l'utilisation du chaos combiné avec les fractions continues afin de générer des flux de clés ayant à la fois de bonnes propriétés cryptographiques et statistiques. Ainsi, nous avons proposé deux nouvelles méthodes de compression sans perte basées sur le codage arithmétique tout en introduisant de nouveaux paramètres de codage afin de réduire davantage la taille en bits des images compressées. Deux autres méthodes s'appuient sur l'utilisation du chaos et des fractions continues pour le développement d'un générateur de nombres pseudo-aléatoires et le cryptage par flot d'images. Enfin, nous proposons une nouvelle méthode qui emploie conjointement le cryptage avec la compression. Cette dernière méthode se base sur l'échange des sous-intervalles associés aux symboles d'un codeur arithmétique binaire de façon aléatoire tout en exploitant notre générateur de nombres pseudo-aléatoire. Elle est efficace, sécurisée et conserve le taux de compression obtenu par le codage arithmétique et ceci quelque soit le modèle statistique employé : statique ou adaptatif.

Mots clés : compression, codage arithmétique, cryptage, chaos, fraction continue, crypto-compression

Abstract : Actually, we live in a digital society. The proliferation of the Internet and the rapid progress in information technology on multimedia, have profoundly transformed the way we communicate. An enormous amount of media can be easily exchanged through the Internet and other communication networks. Digital image in particular occupies an important place in modern interpersonal communication. However, image data have special features such as bulk capacity. Thus, image security and compression issues have became exceptionally acute. It is these considerations that have guided this thesis. Thus, we propose throw this thesis to incorporating security requirements in the data compression system to ensure reasonable security without downgrading the compression performance. For lossless image compression, we have paid most attention to the arithmetic coding (AC) which has been widely used as an efficient compression algorithm in the new standards including JBIG, JBIG2, JPEG2000 and H.264/AVC. For image encryption, we are based on the combination of a chaotic system and the Engel continued fraction map to generate key-stream with both good chaotic and statistical properties. First, we have proposed two new schemes for lossless image compression based on adding new pretreatment steps and on proposing new modeling methods to estimate probabilities for AC. Experimental results demonstrate that the proposed schemes give mean compression ratios that are significantly higher than those by the conventional AC. In addition, we have proposed a new pseudo-random bit generator (PRBG). The detailed analysis done by NIST statistical test Suite demonstrates that the proposed PRGB is suitable for cryptography. The proposed PRBG is used to develop a new symmetric stream cipher for image encryption. Theoretic and numerical simulation analyses indicate that our image encryption algorithm is efficient and satisfies high security. Finally, we have proposed a new scheme which performs both lossless compression and encryption of image. The lossless compression is based on the binary AC (BAC) and the encryption is based on the proposed PRBG. The numerical simulation analysis indicates that the proposed compression and encryption scheme satisfies highly security with no loss of the BAC compression efficiency.

Keywords : compression, arithmetic coding, encryption, chaos, continued fraction, crypto-compression