جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITE SIDI MOHAMED BEN ABDALLAH DE FES
كلية العلوم ظهر المهراز فاس
FACULTE DES SCIENCES DHAR EL MEHRAZ FES

N° d'Ordre : **29 /23**

Formation Doctorale : Sciences et Technologies de l'Information et de Communication

Discipline : Informatique

Spécialité : Informatique

Laboratoire : Laboratoire d'Informatique, Signaux, Automatique et Cognitivisme

THESE de DOCTORAT

Présentée par

Mohammed ES-SABRY

Nouvelles Approches De Sécurité De Données A Base Des Algorithmes Cryptographiques

Soutenue le 11 mai 2023 devant le jury composé de :

Nom et Prénom	*Grade*	*Etablissement*	*Qualité*
Pr. MAHDOU Najib	*PES*	*Faculté des Sciences et Techniques - Fès*	*Président*
Pr. HALLI Akram	*PH*	*Faculté des Sciences Juridiques, Economiques et Sociales - Meknès*	*Rapporteur & examinateur*
Pr. NFAOUI El Habib	*PES*	*Faculté des Sciences Dhar El Mahraz - Fès*	*Rapporteur & examinateur*
Pr. LAZAAR Mohammed	*PH*	*Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes - Rabat*	*Rapporteur & examinateur*
Pr. BOULAALAM Abdelhak	*PH*	*Ecole Nationale des Sciences Appliquées - Fès*	*Examinateur*
Pr. EL ABDERRAHMANI Abdellatif	*PH*	*Faculté des Sciences Dhar El Mahraz - Fès*	*Examinateur*
Pr. SATORI Hassan	*PES*	*Faculté des Sciences Dhar El Mahraz - Fès*	*Examinateur*
Pr. EL AKKAD Nabil	*PH*	*Ecole Nationale des Sciences Appliquées - Fès*	*Co-Directeur de thèse*
Pr. SATORI Khalid	*PES*	*Faculté des Sciences Dhar El Mahraz - Fès*	*Directeur de thèse*

Année universitaire 2022/2023

Remerciements

Tous les travaux présentés dans ce manuscrit de mémoire ont été réalisés au sein de Laboratoire d'Informatique, Signalisation, Automatisme et Cognitivisme, Faculté des Sciences, Dhar El Mahraz, Université Sidi Mohamed Ben Abdellah, Fès.

Je tiens à remercier le Professeur **Khalid SATORI** en tant que directeur de thèse pour m'avoir accueilli dans son équipe, pour ses consignes scientifiques, pédagogiques et même personnelles, auxquelles je lui dois une grande volonté. Je lui suis également très reconnaissant pour son soutien humain et moral et son aide précieuse dans la préparation et la rédaction de cette thèse.

Je tiens à remercier tout particulièrement le Professeur **Nabil EL AKKAD** en tant que co-directeur de thèse, pour son aide, son soutien continu, sa bienveillance, son dévouement et sa compassion dont il a fait preuve durant toutes ses années. Merci de m'avoir écouté, compris, soutenu, et encouragé tout au long de l'encadrement.

Mes sincères remerciements à Monsieur **HALLI Akram**, professeur à la Faculté des Sciences Juridiques, Economiques et Sociales de Meknès, Monsieur **NFAOUI El Habib**, professeur à la Faculté des Sciences Dhar El Mahraz de Fès et Monsieur **LAZAAR Mohammed**, professeur à l'Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes, d'avoir pris le temps d'évaluer ce travail, de m'avoir fait l'honneur d'être rapporteurs de ma thèse et pour leurs remarques et suggestions pertinentes.

Je remercie chaleureusement Monsieur **MAHDOU Najib**, professeur à la Faculté des Sciences et Technologie de Fès, pour l'intérêt qu'il a porté à ce travail et pour avoir accepté de me faire l'honneur de présider le jury de ma soutenance. Toute ma profonde gratitude va également à Monsieur **BOULAALAM Abdelhak,** professeur à l'Ecole Nationale des Sciences Appliquées de Fès**,** Monsieur **EL ABDERRAHMANI Abdellatif,** professeur à la Faculté des Sciences Dhar El Mahraz de Fès et à Monsieur **SATORI Hassan,** professeur à la Faculté des Sciences Dhar El Mahraz de Fès, pour avoir accepté d'examiner ce travail et de faire partie de ce jury.

Mes remerciements vont également à tous ceux qui m'ont soutenu ou qui ont contribué à l'élaboration de ce travail. Je remercie spécialement mes amis et mes collègues au laboratoire LISAC.

Enfin, je tiens à remercier mes parents, mes frères et sœurs, pour m'avoir soutenu, aidé et encouragé pour réaliser ce travail.

Résumé

Ces dernières années, le développement rapide du réseau numérique est devenu une tendance inévitable de la communication moderne. Les données multimédias (images, vidéos, animations, audios, etc.) transmises sur ces réseaux sont des données particulières car elles contiennent beaucoup d'informations. La transmission d'images pose donc un certain nombre de problèmes qui ne sont pas encore entièrement résolus. Il existe plusieurs types de données qui sont très sensible telles que les images médicales, les images militaires, les vidéos conférences et même aussi les images personnelles privées. Le défi c'est de faire la transmission de ces données entre les différents types de périphériques, tout en gardant la confidentialité, l'authenticité et l'intégrité de ces données.

L'objectif visé par cette thèse c'est de développer des algorithmes sécurisés et rapides pour la cryptographie des images couleurs, en se basant sur des fonctions chaotiques de type 1D (Logistic Map, Sine Map, Chebyshev Map, et Tent Map) avec des formes géométriques basiques telles que les carrés et le cube. Pour la partie de la confusion, nous avons utilisé une autre transformation de type 2D nommée (Arnold Cat Map), cette transformation nous permet de changer les positions des pixels cryptés.

Pour montrer l'efficacité et la robustesse de nos méthodes, nous avons utilisé plusieurs attaques et tests pour la cryptographie des images incluant, entre autres, les attaques différentielles, les attaques statistiques, les attaques entropiques. De plus, nous avons évalués nos différentes méthodes par une série d'expériences sur des données de référence et nous les avons comparées avec de nombreuses approches très connues dans la littérature en se basant sur un ensemble de métriques, à savoir : PSNR, MSE, entropie, précision, F-score, coefficient de corrélation. Les différents résultats obtenus par nos algorithmes montrent leurs performances et fiabilités en termes de sécurité, de précision, de confidentialité et de robustesse.

Mots-clés : Cryptage d'images ; Décryptage d'images ; Sécurité ; Fonctions chaotiques ; Cryptographie.

Abstract

In recent years, the rapid development of the digital network has become an inevitable trend in modern communication. The multimedia data (images, videos, animations, audios, etc.) transmitted over these networks are particular data because they contain a lot of information. Image transmission, therefore, poses a number of problems that have not yet been fully resolved. Several types of data are very sensitive, such as medical images, military images, video conferences, and private personal images. The challenge is to transmit this data between different types of devices while maintaining the confidentiality, authenticity, and integrity of this data.

This thesis aims to develop secure and fast algorithms for the cryptography of color images, based on chaotic functions of the 1D type (Logistic Map, Sine Map, Chebyshev Map, and Tent Map) with geometric shapes basics such as squares and the cube. For the confusing part, we used another 2D type transformation named (Arnold Cat Map), this transformation allows us to change the positions of the encrypted pixels.

To show the effectiveness and the robustness of our methods, we used several attacks and tests for the cryptography of the images including, among others, the differential attacks, the statistical attacks, the entropic attacks. In addition, we evaluated our different methods by a series of experiments on reference data and we compared them with many well-known approaches in the literature based on a set of metrics, namely: PSNR, MSE, entropy, precision, F-score, correlation coefficient. The different results obtained by our algorithms show their performance and reliability in terms of security, precision, confidentiality and robustness.

Keywords : Image encryption ; Image decryption ; Security ; Chaotic Maps ; Cryptography.

Table des figures

Figure 1-1: Opération de cryptage et décryptage. 5

Figure 1-2: Chiffrement de César avec la clé égale à 3. 6

Figure 1-3 : Diagramme du schéma de Feistel 13

Figure 1-4 : Diagramme de Triple DES 15

Figure 1-5 : Diagramme de DES 17

Figure 1-6 : Diagramme de AES 20

Figure 2-1 : Classes de complexité. 49

Figure 3-1 : Organigramme de la méthode proposée 75

Figure 3-2 : Matrice de dimension 16 × 16 contenant des valeurs entre 0 et 255. 76

Figure 3-3 : Décomposition de l'image couleur en 3 canaux et extraction des matrices décimales. 77

Figure 3-4 : Opération de cryptage du digramme **202, 58** situant dans des colonnes différentes. 79

Figure 3-5 : Opération de cryptage du digramme **75, 184** situant dans la même colonne. 80

Figure 3-6 : Opération de cryptage d'une matrice 4×4. 81

Figure 3-7 : Rotation circulaire à droite d'une matrice 4×4. 83

Figure 3-8 : Trois faces du cube de notre système de cryptage. 85

Figure 3-9 : Bifurcation Diagram of: (a1) logistic Map. (a2) Sine Map. (a3) Chebyshev Map. Lyapunov exponent Diagram of: (a4) logistic Map. (a5) Sine Map. (a6) Chebyshev Map. 87

Figure 3-10 : Rotation circulaire pour chaque pixel trouvé dans le cube. 90

Figure 3-11 : Opération de cryptage au niveau du canal rouge 90

Figure 3-12 : Opération de cryptage au niveau du canal verte. 91

Figure 3-13 : Opération de cryptage au niveau du canal bleu. 91

Figure 3-14 : Organigramme de la méthode proposée 92

Figure 3-15 : Système de cryptage rempli avec des entiers uniques générés par Quatre cartes chaotiques. 94

Figure 3-16 : Diagrammes de bifurcation et d'exposant de Lyapunov : (b1) Carte logistique (b2) Carte sinus. 95

Figure 3-17: les diagrammes de bifurcation et de l'exposant de Liapounov (c1) Carte de Tchebychev (c2) Carte de la tente. 97

Figure 3-18: Opération de chiffrement des pixels rouges et alphas. 99

Figure 3-19: Opération de chiffrement des pixels verts et bleus. 100

Figure 3-20: Opération de chiffrement pour tous les canaux. 101

Figure 4-1 : (a0) Lena image, **(a1)** Histogramme du canal rouge de Lena image, **(a2)** Histogramme du canal vert de Lena image, **(a3)** Histogramme du canal bleu de Lena image, **(a4)** Lena image cryptée, **(a5)** Histogramme du canal rouge de Lena image cryptée, **(a6)** Histogramme du canal vert de Lena image cryptée, **(a7)** Histogramme du canal bleu de Lena image cryptée. 107

Figure 4-2 : (b0) Baboon image, **(b1)** Histogramme du canal rouge de Baboon image, **(b2)** Histogramme du canal vert de Baboon image, **(b3)** Histogramme du canal bleu de Baboon image, **(b4)** Baboon image cryptée, **(b5)** Histogramme du canal rouge de Baboon image cryptée, **(b6)** Histogramme du canal vert de Baboon image cryptée, **(b7)** Histogramme du canal bleu de Baboon image cryptée. 108

Figure 4-3 : (c0) Fruits image, **(c1)** Histogramme du canal rouge de Fruits image, **(c2)** Histogramme du canal vert de Fruits image, **(c3)** Histogramme du canal bleu de Fruits image, **(c4)** Fruits image cryptée, **(c5)** Histogramme du canal rouge de Fruits image cryptée, **(c6)** Histogramme du canal vert de Fruits image cryptée, **(c7)** Histogramme du canal bleu de Fruits image cryptée. 108

Figure 4-4 : (d0) Peppers image, **(d1)** Histogramme du canal rouge de Peppers image, **(d2)** Histogramme du canal vert de Peppers image, **(d3)** Histogramme du canal bleu de Peppers image, **(d4)** Peppers image cryptée, **(d5)** Histogramme du canal rouge de Peppers image cryptée, **(d6)** Histogramme du canal vert de Peppers image cryptée, **(d7)** Histogramme du canal bleu de Peppers image cryptée. 109

Figure 4-5 : (e0) Bird image, **(e1)** Histogramme du canal rouge de Bird image, **(e2)** Histogramme du canal vert de Bird image, **(e3)** Histogramme du canal bleu de Bird image, **(e4)** Bird image cryptée, **(e5)** Histogramme du canal rouge de Bird image cryptée, **(e6)** Histogramme du canal vert de Bird image cryptée, **(e7)** Histogramme du canal bleu de Bird image cryptée. 109

Figure 4-6 : (f0) Airplane image, **(f1)** Histogramme du canal rouge de Airplane image, **(f2)** Histogramme du canal vert de Airplane image, **(f3)** Histogramme du canal bleu de Airplane image, **(f4)** Airplane image cryptée, **(f5)** Histogramme du canal rouge de Airplane image cryptée, **(f6)** Histogramme du canal vert de Airplane image cryptée, **(f7)** Histogramme du canal bleu de Airplane image cryptée. 110

Figure 4-7 : (g0) Flowers image, **(g1)** Histogramme du canal rouge de Flowers image, **(g2)** Histogramme du canal vert de Flowers image, **(g3)** Histogramme du canal bleu de Flowers image, **(g4)** Flowers image cryptée, **(g5)** Histogramme du canal rouge de Flowers image cryptée, **(g6)** Histogramme du canal vert de Flowers image cryptée, **(g7)** Histogramme du canal bleu de Flowers image cryptée. 110

Figure 4-8 : (h0) Lake image, **(h1)** Histogramme du canal rouge de Lake image, **(h2)** Histogramme du canal vert de Lake image, **(h3)** Histogramme du canal bleu de Lake image, **(h4)** Lake image cryptée, **(h5)** Histogramme du canal rouge de Lake image cryptée, **(h6)** Histogramme du canal vert de Lake image cryptée, **(h7)** Histogramme du canal bleu de Lake image cryptée. 111

Figure 4-9 : Corrélation de deux pixels adjacents suivant la direction horizontale : (i0) canal rouge de l'image originale, (i1) canal vert de l'image originale, (i2) canal bleu de l'image originale, (i3) canal rouge de l'image cryptée, (i4) canal vert de l'image cryptée, (i5) canal bleu de l'image cryptée. 113

Figure 4-10 : Corrélation de deux pixels adjacents suivant la direction verticale : (j0) canal rouge de l'image originale, (j1) canal vert de l'image originale, (j2) canal bleu de l'image originale, (j3) canal rouge de l'image cryptée, (j4) canal vert de l'image cryptée, (j5) canal bleu de l'image cryptée. 113

Figure 4-11 : Corrélation de deux pixels adjacents suivant la direction diagonale : (k0) canal rouge de l'image originale, (k1) canal vert de l'image originale, (k2) canal bleu de l'image

originale, (k3) canal rouge de l'image cryptée, (k4) canal vert de l'image cryptée, (k5) canal bleu de l'image cryptée. 114

Figure 4-12 : (l0) Lena image, **(l1)** Cryptage de Lena Image avec la Clé 1 (Matrice1, Matrice2), **(l2)** Décryptage de Lena Image avec la Clé 1 (Matrice1, Matrice2), **(l3)** Décryptage de Lena Image avec la Clé 2 (Matrice3, Matrice2)..... 117

Figure 4-13 : (a1) Lena image **(a2)** Histogramme rouge de Lena image **(a3)** Histogramme vert de Lena image **(a4)** Histogramme bleu de Lena image **(a5)** Lena image cryptée **(a6)** Histogramme rouge de Lena image cryptée **(a7)** Histogramme vert de Lena image cryptée **(a8)** Histogramme bleu de Lena image cryptée. 119

Figure 4-14 : (b1) Airplane image **(b2)** Histogramme rouge de Airplane image **(b3)** Histogramme vert de Airplane image **(b4)** Histogramme bleu de Airplane image **(b5)** Airplane image cryptée **(b6)** Histogramme rouge de Airplane image cryptée **(b7)** Histogramme vert de Airplane image cryptée **(b8)** Histogramme bleu de Airplane image cryptée. 120

Figure 4-15 : (c1) Bird image **(c2)** Histogramme rouge de Bird image **(c3)** Histogramme vert de Bird image **(c4)** Histogramme bleu de Bird image **(c5)** Bird image cryptée **(c6)** Histogramme rouge de Bird image cryptée **(c7)** Histogramme vert de Bird image cryptée **(c8)** Histogramme bleu de Bird image cryptée..... 120

Figure 4-16 : Distribution des pixels suivant la direction horizontale de Lena image : (d1) Canal rouge originale (d2) Canal vert originale (d3) Canal bleu originale (d4) Canal rouge cryptée (d5) Canal vert cryptée (d6) Canal bleu cryptée. 122

Figure 4-17 : Distribution des pixels suivant la direction verticale de Lena image : (e1) Canal rouge originale (e2) Canal vert originale (e3) Canal bleu originale (e4) Canal rouge cryptée (e5) Canal vert cryptée (e6) Canal bleu cryptée. 122

Figure 4-18: Distribution des pixels suivant la direction diagonale de Lena image : (f1) Canal rouge originale (f2) Canal vert originale (f3) Canal bleu originale (f4) Canal rouge cryptée (f5) Canal vert cryptée (f6) Canal bleu cryptée..... 123

Figure 4-19 : *Sensibilité de la clé :* ***(g1)*** *Image Originale* ***(g2)*** *Image cryptée* ***(g3)*** *Image décryptée avec une clé correcte* ***(g4)*** *Image décryptée avec une clé fausse.* 127

Figure 4-20 : (d1) Image Lena **(d2)** Histogramme de l'image Lena **(d3)** Image Lena cryptée (**d4**) Histogramme de l'image Lena cryptée **(e1)** Image de babouin **(e2)** Histogramme de

l’image de babouin **(e3)** Image de babouin cryptée **(e4)** Histogramme d’image de babouin cryptée. 130

Figure 4-21 : (f1) Image des poivrons **(f2)** Histogramme de l’image des poivrons **(f3)** Image des poivrons cryptés **(f4)** Histogramme de l’image des poivrons cryptés (**g1)** Image du lac **(g2)** Histogramme de l’image du lac **(g3)** Image cryptée du lac **(g4)** Histogramme de l’image du lac cryptée. 131

Figure 4-22 : Distribution de la corrélation pour Lena Image (direction horizontale). 133

Figure 4-23 : Distribution de la corrélation pour Lena Image (direction verticale)..... 133

Figure 4-24 : Distribution de la corrélation pour Lena Image (direction diagonale)..... 134

Figure 4-25 : Sensibilité de la clé : **(k1)** Image originale **(k2)** Image cryptée **(k3)** Image décryptée avec la clé correcte **(k4)** Image décryptée avec la mauvaise clé. 138

Liste des Tableaux

Tableau 1-1: Cryptage du message « Bonjour tout le monde » avec le mot-clé « cacher ». 8

Tableau 1-2 : Règles de l'opération XOR. 11

Tableau 1-3 : Chiffrement de Vernam du Message 100110001. 11

Tableau 1-4 : Factorisation des entiers 26

Tableau 2-1 : Distances d'unicité de texte ASCII chiffré avec différentes longueurs de clé . 44

Tableau 2-2 : Temps de calcul des différentes classes d'algorithmes. 48

Tableau 3-1 : Résultats obtenus pour chaque type d'attaque. 81

Tableau 4-1 : Configuration de la machine..... 102

Tableau 4-2 : Coefficient de corrélation des images originales et des images cryptées. 112

Tableau 4-3 : Comparaison du CC de deux pixels adjacents de notre méthode avec d'autre approches. 115

Tableau 4-4 : Comparaison de CC de notre méthode avec celle de ZHU. 115

Tableau 4-5 : Sensibilité de l'algorithme pour un changement d'un seul pixel de l'image originale. 116

Tableau 4-6 : Paramètres de la qualité de chiffrement et de déchiffrement. 118

Tableau 4-7 : Coefficient de corrélation entre les pixels adjacents de l'image originale et l'image cryptée. 121

Tableau 4-8 : Comparaison du coefficient de corrélation des pixels adjacents avec d'autres méthodes..... 124

Tableau 4-9 : Comparaison du coefficient de corrélation entre l'image originale et l'image cryptée avec d'autres méthodes..... 125

Tableau 4-10 : Comparaison de NPCR et UACI avec d'autres méthodes. 126

Tableau 4-11 : Taux de changement en %. 127

Tableau 4-12 : Comparaison de l'entropie avec d'autres méthodes. 128

Tableau 4-13 : Analyse de MSE et PSNR. 129

Tableau 4-14 : Coefficient de corrélation entre pixels adjacents du jeu de données expérimenté. 132

Tableau 4-15 : Résultats quantitatifs du coefficient de corrélation entre les pixels adjacents par rapport aux autres méthodes (l'image de Lena). 135

Tableau 4-16 : Résultats quantitatifs du coefficient de corrélation entre l'image originale et cryptée par rapport à d'autres méthodes. 135

Tableau 4-17 : Valeurs NPCR et UACI de plusieurs images pour chaque canal. 136

Tableau 4-18 : Comparaison des valeurs UACI et NPCR pour l'image de Lena avec d'autres approches. 136

Tableau 4-19 : Clé correcte. 137

Tableau 4-20 : Clé incorrecte. 137

Tableau 4-21 : Taux de variation en pourcentage. 137

Tableau 4-22 : Entropie de l'information de l'image originale et cryptée. 139

Tableau 4-23 : Comparaison de la valeur d'entropie avec d'autres méthodes. 139

Tableau 4-24 : Valeurs MSE et PSNR de diverses images. 140

Liste des algorithmes

Algorithme 1-1 : Etapes de la cryptographie en utilisant la méthode DES. 16

Algorithme 1-2 : Etapes de la cryptographie en utilisant la méthode AES. 19

Algorithme 1-3 : Procédure d'exécution de l'algorithme Diffie-Hellman..... 24

Algorithme 1-4 : Etapes de la cryptographie en utilisant la méthode RSA. 25

Algorithme 1-5 : Etapes de la cryptographie en utilisant la méthode ElGamal..... 27

Algorithme 2-1 : Procédure d'exécution de l'algorithme de Solovay-Strassen..... 67

Algorithme 2-2 : Procédure d'exécution de l'algorithme de Lehmann 68

Algorithme 2-3 : Procédure d'exécution de l'algorithme de Rabin-Miller..... 68

Algorithme 2-4 : Génération pratique des nombre premiers. 69

Algorithme 5-1 : Description du fonctionnement de MD5..... 146

Algorithme 5-2 : Description du fonctionnement de SHA. 147

Table des matières

Remerciements ..... **I**

Résumé ..... **II**

Abstract ..... **III**

Table des figures ..... **IV**

Liste des Tableaux ..... **V**

Liste des algorithmes ..... **VI**

Table des matières ..... **VII**

Introduction générale **1**

Contexte de la thèse 1

Problématique 1

Contributions 2

Plan de manuscrit 3

Chapitre 1 : Principes Fondamentaux de la Cryptographie **4**

1.1 Introduction 4

1.2 Fondements de la Cryptographie 4

1.3 Cryptographie Classique 5

1.3.1 Chiffrement de César 6

1.3.2 Chiffrement de Vigenère 7

1.4 Cryptographie Symétrique 9

1.4.1 Chiffrement de Vernam (One-Time Pad) 10

1.4.2 Chiffrement de Feistel 12

1.4.3 Chiffrement de DES (Data Encryption Standard) 13

1.4.4 Chiffrement de AES (Advanced Encryption System) 18
1.5 Cryptographie Asymétrique 21
1.5.1 Chiffrement de Diffie et Hellman 23
1.5.2 Chiffrement de RSA 24
1.5.3 Chiffrement de ElGamal 26
1.6 Signature électronique 27
1.6.1 Fonction de hachage 28
1.6.2 Vérification de l'intégrité d'un message 28
1.6.3 Scellement des données 29
1.7 Certificats 29
1.7.1 Structure d'un certificat 30
1.7.2 Niveau de signature 31
1.7.3 Types d'usages 31
1.8 Cryptographie des images couleurs 32
1.9 Sécurité fournie par l'infrastructure 32
1.10 Chiffrement sélectif et chiffrement complet 33
1.11 Propriétés du chiffrement d'images 34
1.11.1 Méthodes naïves 35
1.11.2 Méthodes basées sur le chaos 36
1.12 Méthodes basées sur les opérations des séquences DNA 37
1.12.1 Chiffrement-puis-Compression 37
1.13 Conclusion 38
Chapitre 2 : Outils Mathématiques pour la Cryptographie 40
2.1 Introduction 40
2.2 Théorie de l'information 40

2.2.1 Entropie et incertitude 40
2.2.2 Taux de langage 41
2.2.3 Cryptosystème sécurisé 42
2.2.4 Distance d'unicité 43
2.2.5 Confusion et diffusion 45
2.3 Théorie de la complexité 46
2.3.1 Complexité des algorithmes 46
2.3.2 Complexité des problèmes 48
2.3.3 Quelques problèmes NP-Complets 51
2.4 Théorie des nombres 52
2.4.1 Congruences 52
2.4.2 Nombres premiers 54
2.4.3 Plus grand commun diviseur 54
2.4.4 Inverse modulo n 55
2.4.5 Petit théorème de Fermat 56
2.4.6 Fonction d'Euler 56
2.4.7 Théorème du reste chinois 57
2.4.8 Résidus quadratiques 58
2.4.9 Symbole de Legendre 59
2.4.10 Symbole de Jacobi 60
2.4.11 Entiers de Blum 61
2.4.12 Générateurs 61
2.4.13 Groupe de Galois 63
2.5 Factorisation 64
2.5.1 Racines carrées modulo n 66

2.6 Génération de nombres premiers 67
2.6.1 Solovay-Strassen 67
2.6.2 Lehmann 67
2.6.3 Rabin-Miller 68
2.6.4 Considérations pratiques 69
2.6.5 Nombres premiers forts 70
2.7 Logarithmes discrets dans un corps fini 70
2.7.1 Calcul de logarithmes discrets dans un groupe fini 71
2.8 Conclusion 72
Chapitre 3 : Méthodes et algorithmes proposés pour la cryptographie des images couleurs 73
3.1 Introduction 73
3.2 Algorithmes et méthodes de la cryptographie des images couleurs 73
3.2.1 Algorithme de chiffrement d'images couleurs rapide basé sur la génération de nombres aléatoires de deux matrices et les opérateurs de décalage de bits 74
3.2.1.1 Synthèse 84
3.2.2 Nouvel algorithme de cryptage d'images couleurs en utilisant plusieurs cartes chaotiques avec la méthode d'intersection des plans 84
3.2.2.1 Logistic Map 85
3.2.2.2 Sine Map 86
3.2.2.3 Chebychev Map 86
3.2.2.4 Synthèse 93
3.2.3 Nouvelle méthode de cryptage d'images couleurs 32 bits basée sur plusieurs cartes chaotiques de type 1D et la méthode de chiffrement à quatre carrés 93
3.2.3.1 Cartes chaotiques 94
3.2.3.2 Interprétation mathématique du système de cryptage 97

3.3 Conclusion 101

Chapitre 4 : Expérimentations et discussions 102

4.1 Introduction 102

4.2 Critères d'évaluation utilisés dans les approches proposées 102

4.2.1 Histogramme 102

4.2.2 Analyse de corrélation 103

4.2.3 Analyse des attaques différentielles 104

4.2.4 Analyse de l'entropie 105

4.2.5 Analyse de MSE et PSNR 106

4.3 Résultats d'expérimentation et discussions 107

4.3.1 Algorithme de chiffrement d'images couleurs rapide basé sur la génération de nombres aléatoires de deux matrices et les opérateurs de décalage de bits 107

4.3.1.1 Histogramme 107

4.3.1.2 Analyse de corrélation 111

4.3.1.3 Analyse des attaques différentielles 116

4.3.1.4 Analyse de l'entropie 117

4.3.1.5 Analyse de MSE et PSNR 117

4.3.2 Nouvelle méthode de cryptage d'image couleur 32 bits basée sur plusieurs cartes chaotiques de type 1D et la méthode de chiffrement à quatre carrés 119

4.3.2.1 Histogramme 119

4.3.2.2 Analyse de la corrélation 121

4.3.2.3 Analyse des différentielles attaques 125

4.3.2.4 Analyse de l'entropie 128

4.3.2.5 Analyse de MSE et PSNR 128

4.3.3 Nouvelle méthode de cryptage d'images couleurs 32 bits basée sur plusieurs cartes chaotiques de type 1D et la méthode de chiffrement à quatre carrés 129

4.3.3.1 Histogramme 129
4.3.3.2 Analyse de corrélation 132
4.3.3.3 Analyse des attaques différentielles 135
4.3.3.4 Analyse d'entropie 138
4.3.3.5 Analyse de MSE et PSNR 139
4.4 Discussions 140
4.5 Conclusion 141
Conclusion générale et perspectives 142
Conclusion générale 142
Perspectives 143
Annexe A : Algorithmes des signatures électroniques 145
Algorithme de MD5 : 145
Algorithme de SHA : 146
Publications 148
Articles publiés dans des journaux internationaux 148
Article accepté avec modification 148
Communications internationales 148
Bibliographie 150

Introduction générale

Contexte de la thèse

Aujourd'hui, l'utilisation des données multimédias (images, vidéos, animations, audio, etc.) a fortement augmenté en raison du fort développement d'internet et du développement rapide des technologies de stockage. Chaque minute, une énorme quantité de données multimédia est transférée et enregistrée sur des supports de stockage ou à l'aide de la technologie cloud. En effet, plusieurs données sont très intéressantes et plus sensibles incluant, entre autres, les images militaires, les vidéoconférences, les images médicales et les images personnelles privées.

Par conséquent, la résolution du problème de sécurité de la transmission ou du stockage est devenue un gros souci. Dans l'énorme société de l'information numérique sur Internet d'aujourd'hui, la sécurité de l'information est devenue une technologie clé pour protéger les informations numériques contre la violation et la discrimination contre l'identité juridique des utilisateurs. En fait, les interactions réseau entre les expéditeurs et les destinataires doivent être des communications confidentielles. Les techniques de la cryptographie des images sont utilisées pour transformer les images de leur état de données normal en un état non reconnaissable. Contrairement aux informations textuelles, les images numériques ont une redondance élevée et l'œil humain a une capacité limitée à reconnaître les niveaux de gris, de sorte que les images numériques peuvent avoir un certain degré de distorsion. Cependant, compte tenu de certaines caractéristiques des images numériques, l'étude du système de cryptage d'image peut faire en sorte que la technologie de transmission de données réponde aux attentes et aux besoins des personnes en matière de sécurité des communications, ce qui a une signification sociale et économique très importante.

Problématique

Contrairement à la cryptographie traditionnelle, la sécurité n'est pas nécessairement l'objectif le plus important des systèmes de cryptage d'images et de vidéos. D’autres propriétés, telles que le temps de réponse, peuvent être tout aussi importantes. Dans ce contexte, les termes « cryptage souple » ou « cryptage sélectif » sont parfois utilisés à la place des schémas de cryptage « durs » classiques tels que le cryptage AES complet. Le but de ces schémas n'est pas

une sécurité maximale, mais plutôt un compromis entre la sécurité et la complexité de calcul. Ils sont conçus pour protéger le contenu multimédia et répondre aux exigences de sécurité d'applications multimédia spécifiques. Par exemple, le cryptage en temps réel d'un flux vidéo entier à l'aide de chiffrements traditionnels nécessite plus de calculs en raison de la grande quantité de données impliquées.

D'autre part, de nombreuses applications multimédias nécessitent un niveau de sécurité plus faible (par exemple, la diffusion télévisée) ou elles ont besoin de protéger leurs données que pendant une courte période (la diffusion d'informations). Les applications de sécurité multimédia doivent donc rechercher des programmes de chiffrement rapides spécialement adaptés à l'environnement cible. Les domaines d'intérêt pour le déploiement de telles solutions vont être des systèmes de gestion des droits numériques (DRM) à la protection des communications personnelles.

Contributions

Dans ce travail, nous introduisons les trois algorithmes de la cryptographie des images couleurs que nous avons développés durant la préparation de cette thèse :

Le premier est intitulé : « Un nouvel algorithme de cryptage d'images utilisant la génération de nombres aléatoires de deux matrices et des opérateurs de décalage de bits ». Cet algorithme est basé sur un système de cryptage composé de deux carrés (matrices) de taille 16×16, les cases de ces derniers sont remplies avec des entiers uniques entre 0 et 255, et qui sont généré aléatoirement en utilisant l'une des méthodes de la génération aléatoire des nombres telle que True random number generators (TRNGs), Pseudo-random number generators (PRNGs), ou bien Chaotic random number generators (CRNGs).

Le deuxième est intitulé : « Un nouvel algorithme de cryptage d'images couleurs utilisant plusieurs cartes chaotiques avec la méthode d'intersection des plans ». L'idée de cet algorithme est basée sur l'utilisation des cartes chaotiques pour générer une séquence d'entiers unique afin de remplir notre système de cryptage qui est composé de trois faces adjacentes d'un cube de taille $16 \times 16 \times 16$. Ensuite, on a utilisé la méthode d'intersection des plans à l'intérieur du cube pour crypter les valeurs des pixels des images couleurs.

Le troisième est intitulé : « Une nouvelle méthode de cryptage d'images couleurs de 32 bits basés sur plusieurs cartes chaotiques de type 1D et la méthode de chiffrement à quatre carrés ». Cette méthode est une amélioration de notre première contribution, l'idée principale est basée sur l'utilisation des cartes chaotiques pour générer une séquence d'entiers unique afin de remplir 4 matrices de taille 16×16. Et pour crypter les valeurs des pixels des images utilisées, on a utilisé la méthode de chiffrement à quatre carrés.

Plan de manuscrit

Le reste du manuscrit est composé de quatre chapitres et organisé de la façon suivante :

Le premier chapitre présente des principes fondamentaux de la cryptographie de données. Dans ce chapitre, on va traiter les différentes techniques de cryptage classique et moderne. De plus, on va montrer pourquoi ces algorithmes ne sont pas utilisés dans le domaine de la cryptographie des images couleurs, même si leur robustesse et leur niveau de sécurité très élevé, ainsi que les différentes techniques utilisées dans la littérature dans le domaine de la cryptographie des images.

Le deuxième chapitre est consacré aux outils mathématiques qui sont nécessaires pour aborder le domaine de la cryptographie des données.

Le troisième chapitre présente les différents algorithmes que nous avons développés dans le domaine de la cryptographie des images.

Le quatrième chapitre présente une description des métriques utilisées pour évaluer les résultats, avec une interprétation des résultats obtenus et des comparaisons des performances de nos contributions avec d'autres méthodes de cryptographie des images couleurs.

La conclusion générale et les perspectives présentent un bilan qui synthétise les travaux réalisés et ouvre des perspectives qui peuvent être étudiées pour améliorer les techniques de la cryptographie des données.

Les annexes présentent quelques algorithmes utilisés pour la génération de la clé, ainsi que des algorithmes exploités dans la cryptographie classique, moderne et la cryptographie des images couleurs.

Chapitre 1 : Principes Fondamentaux de la Cryptographie

1.1 Introduction

Les humains ont toujours ressenti le besoin de cacher des informations, avant même les premiers ordinateurs et calculatrices. Depuis sa création, le réseau Internet s'est développé si rapidement qu'il est devenu un outil de communication incontournable. Cependant cette communication met de plus en plus en jeu des flux financiers pour les entreprises présentes sur le Web. Les transactions effectuées sur le réseau peuvent être interceptées, d'autant plus que les lois sont difficiles à faire respecter sur Internet, il est donc nécessaire de conserver ces informations en toute sécurité : c'est la cryptographie qui les gère.

La cryptographie est la science de la sécurité des informations et des communications. Elle est entrée sur le marché des produits de masse assez récemment et chaque citoyen des pays développés l'utilise quotidiennement. Elle est utilisée pour l'authentification et le cryptage (cartes bancaires, téléphonie sans fil, le commerce électronique, la télévision à péage), le contrôle d'accès (systèmes de verrouillage des voitures, remontées mécaniques), le paiement (cartes téléphoniques prépayées, e-cash), et pourrait devenir l'instrument fondamental de la démocratie avec l'avènement des systèmes de vote électronique. La maîtrise des outils cryptographiques devient une exigence pour la plupart des ingénieurs. Le présent ouvrage vise à présenter les principes fondamentaux de la cryptographie moderne.

Dans ce chapitre, Nous introduisons d'abord les bases de la cryptographie. Nous décrivons ensuite deux grandes classes d'algorithmes de chiffrement, à savoir : le chiffrement symétrique et le chiffrement asymétrique, ainsi que quelques exemples de crypto système moderne.

1.2 Fondements de la Cryptographie

. Le mot cryptographie est un terme générique qui désigne l'ensemble des techniques permettant de chiffrer les messages, c'est-à-dire qu'ils deviennent incompréhensibles sans opérations spécifiques. La cryptographie est intrinsèquement basée sur l'arithmétique. Dans le

cas du texte, il s'agit de convertir les lettres qui composent le message en une suite de nombres (dans le cas du calcul, sous forme de bits, puisque les ordinateurs fonctionnent en binaire), puis un calcul sur ces nombres pour d'une part, les modifier de telle façon à les rendre incompréhensibles. Le résultat de cette modification (le message chiffré) est appelé cryptogramme (CipherText) par opposition au message initial, appelé message en clair (PlainText), d'autre part, faire en sorte que le destinataire saura les déchiffrer.

Encoder un message d'une manière qui le garde secret s'appelle le cryptage (le chiffrement). La méthode inverse consistant à retrouver le message d'origine est appelée le décryptage (le déchiffrement). Cette opération vous permet de stocker des informations confidentielles ou de les transmettre sur un réseau non sécurisé (comme Internet) afin que personne d'autre que le destinataire ne puisse les lire.

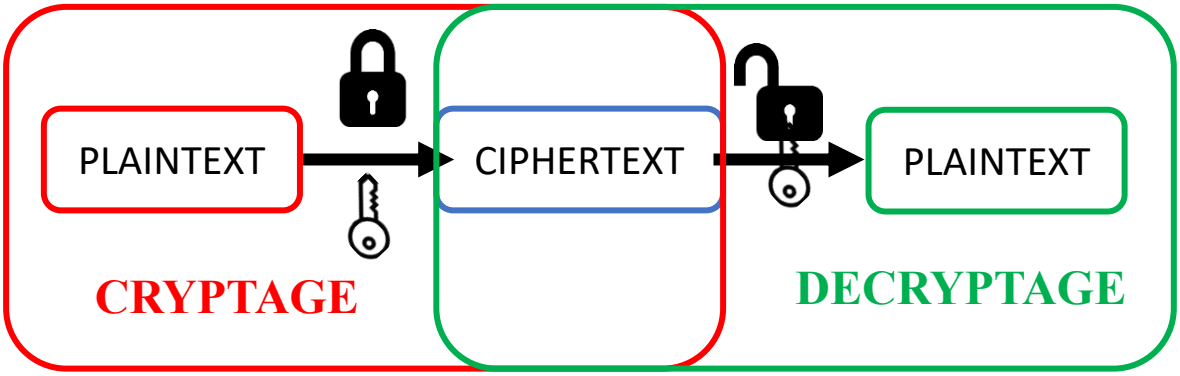


Figure 1-1: *Opération de cryptage et décryptage.*

D'une façon formelle, les crypto systèmes sont caractérisés par les éléments (P, C, K, E, D), où P représente l'ensemble des messages en clairs (PlainText) possibles, C représente l'ensemble des messages cryptés (CipherText) possibles, et K représente l'ensemble des clés possibles. Pour chaque clé $k \in K$, il y a une règle de cryptage $e_k \in E$, et une règle de décryptage correspondante $d_k \in D$, ces fonctions de cryptage et décryptage sont définies respectivement comme suite : pour chaque $e_k : P \to C$ et $d_k : C \to P$ telles que $\forall x \in P \quad d_k(e_k(x)) = x$, avec x représente un message en clair (PlainText).

1.3 Cryptographie Classique

Nous discuterons dans cette partie des systèmes cryptographiques conçus avant la création des ordinateurs. Cette discussion fournira des concepts et des fondements pour l'évolution de

plusieurs algorithmes symétriques encore utilisés aujourd'hui. Les cryptosystèmes classiques sont divisés en chiffrement à une lettre et à plusieurs lettres. Le cryptage à une lettre est assez basique, c'est un remplacement simple. Remplacez chaque lettre par une autre lettre ou un autre symbole basé sur un algorithme.

1.3.1 Chiffrement de César

Ce code crypté est l'un des plus anciens codes utilisés par Jules César. L'encodage fonctionne en attribuant une valeur constante à tous les caractères d'un message, ou plus précisément à leur code ASCII (l'encodage utilisé par les ordinateurs). Ce qui signifie tout simplement que l'ensemble de valeurs des caractères du message doit être décaler d'un certain nombre de positions, autrement dit, remplacer chaque lettre par une autre. Cependant, déplacer le message "**BONJOUR TOUT LE MONDE**" de trois positions donne "**ERQMRXU WRXW OH PRQGH**". Il suffit de commencer par **A** lorsque la valeur ajoutée donne plus de lettres que la lettre **Z**, ce qui équivaut à une opération **modulo 26**.

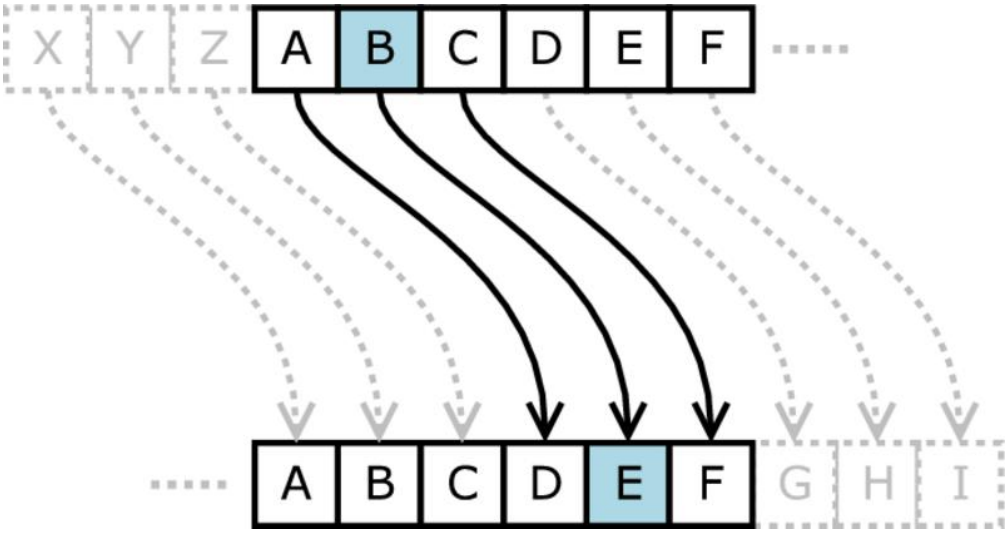


***Figure 1-2**: Chiffrement de César avec la clé égale à 3.*

Généralement, la fonction de ce type de cryptage $e_k(x) = x + k \ mod\ 26$, la fonction de décryptage $d_k(y) = y - k \ mod\ 26$, ces fonctions sont définies dans le groupe cyclique $\mathbb{z}/26\mathbb{z}$, avec la clé k égale à 3.

Ce système de chiffrement est bien sûr simple à mettre en œuvre, mais il présente l'inconvénient d'être totalement symétrique, ce qui signifie que le message initial peut être connu par simple soustraction. L'approche principale pourrait consister simplement à soustraire

les nombres à partir de 1 jusqu'à 26 pour vérifier si on peut trouver un entier dans cet intervalle [1,26] qui donne un texte intelligible.

Il existe une approche plus avancée qui repose à compter la fréquence des lettres dans un message crypté (c'est plus facile à faire puisque le message est long). En effet, selon les langues, certaines lettres apparaissent plus fréquemment que d'autres (par exemple, en français, la lettre la plus courante est la lettre E), ainsi la lettre la plus courante dans un texte chiffré par César peut correspondre à la lettre E, une simple soustraction donne alors la clé de chiffrement.

Ce système de chiffrement est certes simple à mettre en œuvre, mais son inconvénient est qu'il est complètement symétrique, ce qui signifie que le message d'origine peut être connu par simple soustraction. L'approche principale pourrait être de simplement soustraire des nombres de 1 à 26 et de voir si l'un de ces nombres donne un message compréhensible.

Une approche plus avancée consisterait à compter la fréquence des lettres dans le message codé (c'est plus facile car le message est long). En effet, selon la langue, certaines lettres apparaissent plus fréquemment que d'autres (par exemple, en français la lettre la plus courante est la lettre E), ainsi la lettre la plus courante dans le texte crypté par César pourrait correspondre à la lettre E, un Une simple soustraction donne alors la clé de chiffrement

1.3.2 Chiffrement de Vigenère

Le prochain chiffrement classique de grand intérêt s'appelait le chiffrement de Vigenère. Bien qu'il s'agisse d'un concept relativement simple, il a influencé de nombreux chiffrements utilisés à travers l'histoire, y compris certains qui sont encore utilisés aujourd'hui. L'idée est de modifier le concept du chiffrement de César discuté dans la section précédente. Au lieu d'une clé de cryptage unique, nous utilisons une clé entière, qui est une chaîne de lettres sélectionnée par l'utilisateur. Par exemple, disons que nous décidons d'utiliser le mot "CRYPTER" comme mot-clé. Le processus de cryptage est effectué comme suit. Le mot-clé est répété autant de fois que nécessaire pour couvrir la longueur du message. Chaque lettre correspond à un chiffre compris entre 0 et 25 ($A = 0, B = 1, C = 2, \dots, Z = 25$), Ensuite, pour le chiffrement, on additionne les numéros correspondants de la clé et du message, et on soustrait 26 si la somme est supérieure à 25. Ensuite, nous revenons aux lettres pour obtenir le texte chiffré.

Par exemple, si la lettre du message est $Z = 25$, et le lettre du mot-clé est $C = 2$, alors la lettre du message crypté correspond à $25 + 2 = 27$, cependant, 27 est plus grand que 26, donc on soustrait 26 pour obtenir 1, qui est la lettre **B.** et par suite, la lettre **B** est le cryptage de la lettre **Z** en utilisant le mot-clé **C.**

Tableau 1-1: *Cryptage du message « Bonjour tout le monde » avec le mot-clé « cacher ».*

Texte	B	O	N	J	O	U	R	T	O	U	T	L	E	M	O	N	D	E
Mot-Clé	C	R	Y	P	T	E	R	C	R	Y	P	T	E	R	C	R	Y	P
Texte Crypté	D	F	L	Y	H	Y	I	V	F	S	I	E	I	D	Q	E	B	T

Notez que chaque lettre du texte chiffré est décalée de quantités variables, contrairement au chiffrement de César. En interprétant cela comme de nombreux chiffrements monoalphabétiques agissant sur des caractères individuels, il est facile de comprendre pourquoi le chiffrement de Vigenère est appelé chiffrement polyalphabétique.

Pour décrypter, soustrayez simplement la valeur des lettres clés respectives de chaque lettre du texte chiffré. Par exemple, pour déchiffrer le texte chiffré « DFLYHYIVFSIEIDQEBT » ci-dessus, nous utilisons le mot-clé « CRYPTER » comme suit :

$$D - C = 3 - 2 = 1 = B$$

$$F - R = 5 - 17 = -12 \equiv -12 + 26 = 14 = O$$

$$L - Y = 11 - 24 = -13 \equiv -13 + 26 = 13 = N$$

Répétez ce processus jusqu'à ce que tous les caractères du message en clair soient identifiés. Ce processus de cryptage est facile si vous connaissez la clé. Ce processus de chiffrement est facile si l'on connaît la clé. Cependant, il peut être difficile de casser le chiffrement sans ces informations. Le chiffrement de Vigenère est appelé chiffrement par bloc où la longueur du bloc est égale à la longueur de la clé. Dans un processus de chiffrement par bloc, le texte brut ou les messages sont chiffrés bloc par bloc plutôt que caractère par caractère.

Il y a quelques points à noter concernant l'utilisation du chiffrement de Vigenère. Premièrement, la cryptanalyse de tels chiffrements est beaucoup plus complexe qu'un simple chiffrement de substitution à une seule lettre, puisque les mêmes caractères en clair sont chiffrés en caractères différents en fonction de leur position. A noter également que le chiffrement de

Vigenère a été inventé pour masquer la fréquence des lettres en anglais. Par exemple, le chiffre de César ne le fait pas. De plus, si le mot-clé « CRYPTERLESMESSAGESX » était utilisé à la place de "CRYPTER" dans l'exemple précédent, le message chiffré aurait un secret parfait. Un secret parfait peut être atteint si le texte chiffré ne fournit aucune information sur le texte en clair, ce qui se produit à peu près lorsque le mot-clé est aussi long que le message lui-même. Un tel système de sécurité peut être obtenu à l'aide de l'algorithme One-Time Pad.

1.4 Cryptographie Symétrique

Le chiffrement symétrique (également appelé chiffrement à clé privée ou chiffrement à clé secret) implique l'utilisation de la même clé pour le chiffrement et le déchiffrement. Par conséquent, la sécurité d'un cryptosystème repose sur l'échange sécurisé de clés ; seuls l'expéditeur et le destinataire du message ont besoin de connaître les clés. De plus, cette clé doit être suffisamment grande pour empêcher les attaques par force brute. Il existe deux types de cryptosystèmes symétriques, à savoir : les algorithmes de chiffrement par flux et les algorithmes de chiffrement par bloc. Les algorithmes de chiffrement de flux traitent un message en clair comme un flux de données (bits ou octets). Lors du cryptage, chaque caractère est crypté indépendamment, l'opération est très simple et la vitesse d'exécution de l'algorithme est très rapide. En fait, le chiffrement de flux peut s'exécuter en temps réel sans attendre que les données à chiffrer soient entièrement reçues.

Les principes des chiffrements de flux ont été définis par Vernam en 1926. Dans cette approche, la clé est utilisée comme graine d'initialisation cryptographiquement sécurisée, et donc comme générateur de nombres pseudo-aléatoires déterministes, pour générer une séquence pseudo-binaire-aléatoire, appelée flux chiffré, de même taille que le message à chiffrer. Le chiffrement du message consiste alors à effectuer un XOR entre les données à chiffrer et la séquence binaire pseudo-aléatoire générée. Les algorithmes de chiffrement par blocs impliquent de diviser les données ordinaires en blocs de taille fixe (généralement 64, 128 ou 256 bits). Ces blocs sont ensuite chiffrés un par un. Notez qu'il existe plusieurs modes de chiffrement par blocs. Les algorithmes de chiffrement par blocs les plus connus sont l'algorithme Data Encryption Standard (DES), apparu en 1976 et utilisant une clé de 56 bits, le Triple DES (3DES), dérivé du DES utilisant deux ou trois clés de même taille, et l'algorithme Advanced Encryption Standard (AES), désormais considéré comme la norme pour les chiffrements par blocs, qui peut utiliser des clés jusqu'à 256 bits.

Dans les années 1940, Claude Shannon a démontré que, pour être complètement sécurisé, un système à clé privée doit utiliser une clé de longueur au moins égale à la longueur du message à chiffrer. De plus, le cryptage symétrique nécessite un canal sécurisé pour l'échange de clés, ce qui réduit considérablement les avantages de tels systèmes de cryptage. Le principal inconvénient des cryptosystèmes à clés privées (également appelés cryptosystèmes symétriques) provient de l'échange de clés. En effet, le chiffrement symétrique repose sur l'échange de secrets (clés) et pose le problème de la distribution des clés. En revanche, les utilisateurs souhaitant communiquer avec plusieurs personnes tout en assurant différents niveaux de confidentialité doivent utiliser autant de clés privées que d'interlocuteurs.

1.4.1 Chiffrement de Vernam (One-Time Pad)

Le chiffrement de Vernam est un algorithmes cryptographique inventé par Gilbert Vernam en 1917 et perfectionnés par Joseph Mauborgne, qui y a ajouté le concept de clés aléatoires. Cependant, le banquier américain Frank Miller en a jeté les bases dès 1882. Bien que l'encodage et le décodage soient simples, faciles et rapides, ce chiffrement est théoriquement impossible à déchiffrer, mais le fait que le masque soit à usage unique impose une transmission précoce par "d'autres moyens", ce qui rend difficile l'application de la sécurité des échanges Internet. Il peut être simplement décrit comme un chiffre de Vigenère, mais la clé répond aux trois exigences suivantes :

- Aussi longtemps que le texte à chiffrer.
- C’est complètement aléatoire.
- Il n'est utilisé que pour chiffrer un seul message, puis immédiatement détruit.

Claude Shannon, également chercheur chez AT&T Labs, a prouvé en 1949 que le nombre était totalement sûr. Si nous interceptons un message crypté, la seule information dont nous disposons est la longueur du message en clair. De plus, tout mot de passe parfaitement sécurisé est nécessairement une déclinaison du mot de passe Vernam.

Les chiffrements modernes de Vernam sont implémentés comme suit. Le message est d'abord converti par l'ordinateur en une séquence de bits, 0 et 1, et nous prenons une clé (complètement aléatoire) qui consiste également en une séquence de 0 et de 1, tant que le

message est chiffré. Ensuite, on prend chaque bit du message en clair et de la clé, et on en fait le ou exclusif (Xor) en respectant les règles du tableau suivant.

Tableau 1-2 : *Règles de l'opération XOR.*

A	**B**	$\boldsymbol{A \oplus B}$
0	0	0
0	1	1
1	0	1
1	1	0

Par exemple, le texte en clair est 100110001, si la clé est 010100110, alors le texte chiffré est 110010111, comme indiqué dans le tableau suivant :

Tableau 1-3 : *Chiffrement de Vernam du Message 100110001.*

Message en Clair	1	0	0	1	1	0	0	0	1
Clé	0	1	0	1	0	0	1	1	0
Message Crypté	1	1	0	0	1	0	1	1	1

Lorsque le message est reçu, le destinataire du message fait la même chose avec le message chiffré : il prend donc chaque bit du message chiffré et fait l'opération de XOR avec le bit correspondant de la clé. Il trouve le message initial grâce aux deux propriétés précédentes.

Malheureusement, le chiffrement de Vernam n'est pas une bonne solution. Tout d'abord, il nécessite que la clé ne soit utilisée qu'une seule fois. Si vous utilisez deux fois la même clé, vous pouvez extraire de nombreuses informations des messages chiffrés.

Deuxièmement, le chiffrement de Vernam nécessite des clés extrêmement longues et une synchronisation parfaite des clés. Par conséquent, l'échange de clés qui doit être sécurisé est difficile à réaliser. Enfin, les clés utilisées doivent être complètement aléatoires, ce qui n'est pas facile à garantir.

C'est pourquoi ce chiffrement n'est implémenté que dans des cas très particuliers. A ce titre, il servait à protéger le téléphone rouge, la ligne directe entre la Maison Blanche et le Kremlin pendant la guerre froide. Les clés circulaient dans des valises diplomatiques, transportées dans

des avions remplis d'agents. Pour générer des clés aléatoires, les Soviétiques auraient utilisé « lancer les dés » : leur travail consistait à lancer les dés toute la journée et à enregistrer les résultats.

Les systèmes de chiffrement symétrique utilisés dans la pratique tentent d'imiter le chiffrement de Vernam, donnant l'impression (illusion ?) que les informations chiffrées sont aléatoires. Bien sûr, il est impossible de créer des choses complètement aléatoires dans les logiciels, et les performances de ces systèmes sont inférieures aux chiffres de Vernam. Cependant, ils sont plus pratiques à mettre en œuvre et offrent généralement une sécurité suffisante pour les besoins.

1.4.2 Chiffrement de Feistel

Bien sûr, les algorithmes à clé secret réduite visent la perfection. La perfection, en cryptographie, c'est l'aléatoire : le message encodé doit être le plus aléatoire possible, en minimisant le risque d'attaque par l'analyse du texte chiffré, la redondance, etc.

Le problème est que si nous savions très tôt comment construire des fonctions qui semblaient aléatoires, nous ne savions pas comment construire des bijections aléatoires avant les travaux de Feistel. La solution proposée par l'ingénieur IBM Horst Feistel, qui a dirigé l'équipe qui a conçu l'algorithme de chiffrement DES, est assez élégante : par exemple, disons que nous avons une fonction « presque aléatoire » $\boldsymbol{f}$ qui prend un message de $\boldsymbol{n}$ bits comme argument, et renvoie un message de $\boldsymbol{n}$ bits (donne l'impression d'être choisi au hasard). $\boldsymbol{f}$ ne doit pas être bijectif, c'est-à-dire que deux message différents peuvent avoir la même image par $\boldsymbol{f}$ (ce qui signifie encore une fois que si nous avons un message de n bits chiffré par $\boldsymbol{f}$, nous ne pouvons pas trouver le message initial).

Un algorithme utilisant le schéma de Feistel procéderait en chiffrant des blocs de 2n bits divisés en 2, la partie gauche G, et la partie droite D. L'image du bloc (G, D) du schéma de Feistel est le bloc (L, R), où $L = D\ et\ G = Rf(L)$ où $\oplus$ représente l'opération de XOR (exclusif ou). Cette fois la transformation est bijective, car si on a un tel couple (L, R), on peut trouver (G, D) par $D = L\ et\ G = R \oplus f(L)$. Bien sûr, la partie droite du texte en clair est inchangée par rapport à la partie gauche du texte chiffré. Pour éviter cette faiblesse, on répète le schéma de Feistel un certain nombre de fois (on parle de tours - DES en a 16).

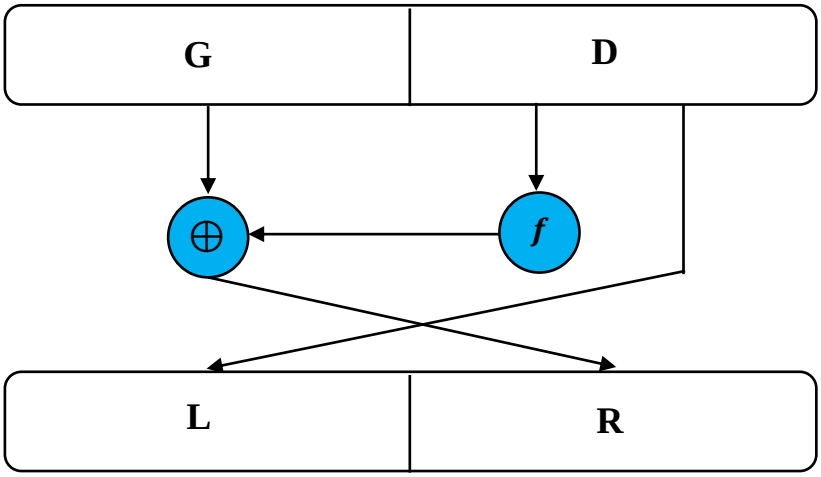


Figure 1-3 : *Diagramme du schéma de Feistel*

La plupart des algorithmes à clé secrète de la fin du 20e siècle, était des schémas de Feistel. L'émergence d'AES, au début des années 2000, n'en était plus une, marquant la fin de la domination de tels algorithmes.

1.4.3 Chiffrement de DES (Data Encryption Standard)

Dans les années 1970, seuls les militaires disposaient d'un algorithme de clé secret fiable. Face à l'émergence de la demande civile, le NBS (National Bureau of Standards) lance un appel d'offres le 15 mai 1973, dans le Federal Register pour la création d'un système cryptographique. Les spécifications sont les suivantes :

- L'algorithme repose sur une clé relativement petite pour le chiffrement et le déchiffrement.
- L'algorithme doit être facile à mettre en œuvre, tant au niveau logiciel que matériel, et doit être très rapide.
- Le chiffrement doit être hautement sécurisé, associé uniquement à la clé, et non à la confidentialité de l'algorithme.

IBM a fourni Lucifer à la fin de 1974, et un effort conjoint avec la NSA (National Security Agency) a conduit au développement de DES (Data Encryption Standard), l'algorithme de chiffrement le plus utilisé au monde, au dernier trimestre 2019. 21e siècle. Le DES a été publié en tant que norme par NBS le 15 janvier 1977.

Une clé DES est une chaîne de 64 bits (0 et 1 consécutifs), mais seuls 56 bits sont réellement utilisés pour définir la clé. Les bits 8, 16, 24, 32, 40, 48, 56, 64 sont des bits de parité (= bits de détection d'erreur). Le 8e bit sert à faire en sorte que les 8 premiers bits aient un nombre impair de 1, par exemple, si les 7 premiers bits sont 1010001, alors le 8e bit est 0, ce qui évite les erreurs de transmission. Ainsi, DES a 2^{56} clés possibles, soit environ 720 milliards de possibilités, ce qui est un nombre assez important, mais qui est en fait assez insuffisant pour les capacités des ordinateurs modernes.

Le DES est controversé depuis sa création. Bien que sa description soit assez compliquée (voir ci-dessous), la plupart des opérations qu'il effectue sont linéaires. Cependant, nous savons depuis longtemps comment surmonter les graphiques linéaires. Par conséquent, toute la sécurité de DES repose sur les opérations non linéaires qu'il effectue. Cela se fait à l'aide de 8 S-box "mystérieuses", qui sont des tableaux de 4 lignes et 16 colonnes, chaque ligne contenant 16 nombres de 0 à 15. Ces S-box sont choisies par la NSA, ce qui a semé une certaine confusion : comment s'assurer que la NSA ne choisit pas ces S-box en plaçant un "piège" qui lui permettrait de tout déchiffrer tout en confirmant que l'algorithme est sécurisé ?

Honnêtement, rien dans l'histoire du DES ne le confirme objectivement. En particulier, il a résisté aux travaux des cryptanalystes basés sur la force non brute. D'autre part, c'est l'extraordinaire avancement de la puissance informatique qui a signé l'arrêt de mort DES. Le 17 juin 1997, DES a été piraté en 3 semaines par un consortium de petites machines sur Internet. Et très formellement estimé (dans un rapport au Sénat américain) combien de temps il faudrait à un pays pour percer les secrets des messages chiffrés avec DES en quelques secondes à cette date. Ainsi, DES a été progressivement abandonné à la fin des années 1990 et a été initialement remplacé par Triple DES, qui consiste en trois applications de DES avec 2 clés différentes (la clé est donc de 112 bits).

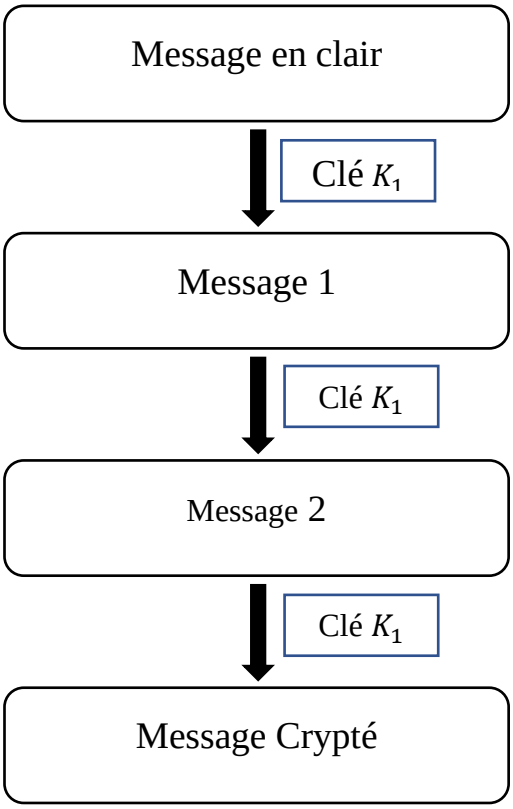


Figure 1-4 : *Diagramme de Triple DES*

Alors que la sécurité est largement garantie, Triple DES est malheureusement trois fois plus lent que DES. C'est pourquoi en janvier 1997, le NIST (National Institute of Standards and Technology) lance un nouvel appel pour la création d'un successeur au DES. Une nouvelle saga d'AES (Advanced Encryption Standard) commence.

DES est un algorithme de chiffrement de blocs de 64 bits, ce qui signifie que le texte est d'abord coupé en blocs de 64 bits, puis l'algorithme est appliqué à chaque bloc. Comme mentionné ci-dessus, la clé de cryptage est également constituée de 64 bits, et même si seuls 56 bits sont utiles, les 8 bits restants sont des bits de contrôle utilisés pour éviter les erreurs de transmission. Les grandes lignes de l'algorithme sont :

Début

Étape 1 : Préparation - Diversification des clés.

La clé K est diversifiée, c'est-à-dire qu'elle est constituée de 16 sous-clés de 48 bits $K_1, K_2, \dots, K_{16}$. Chaque K_i se compose de 48 bits de K avec $i \in \{1,2, \dots,16\}$, disposés dans un certain ordre.

Étape 2 : Permutation initiale.

Pour chaque bloc de texte X composant de 64 bits, calculez la permutation $Y = P(X)$. Y est représenté par $Y = G_0 D_0$, G_0 est les 32 bits de gauche de Y et D_0 est les 32 bits de droite.

Étape 3 : Itération - Schéma de Feistel

Le même schéma de Feistel a été appliqué pendant 16 tours. A partir de $G_{i-1}D_{i-1}$ avec $i \in \{1,2, \dots,16\}$, on calcule $G_i D_i$ en posant : $G_i = D_{i-1}$ et $D_i = G_{i-1} \oplus f(D_{i-1}, K_i)$ où $\oplus$ est le "ou exclusif" au niveau du bit et f est la fonction confusion, une série de substitutions et de permutations.

Étape 4 : Permutation final.

L'inverse de la disposition initiale est appliqué à $G_{16}D_{16}$. $Z = P^{-1}(G_{16}D_{16})$ est un bloc de 64 bits chiffré à partir de X.

Fin

Algorithme 1-1 : *Etapes de la cryptographie en utilisant la méthode DES.*

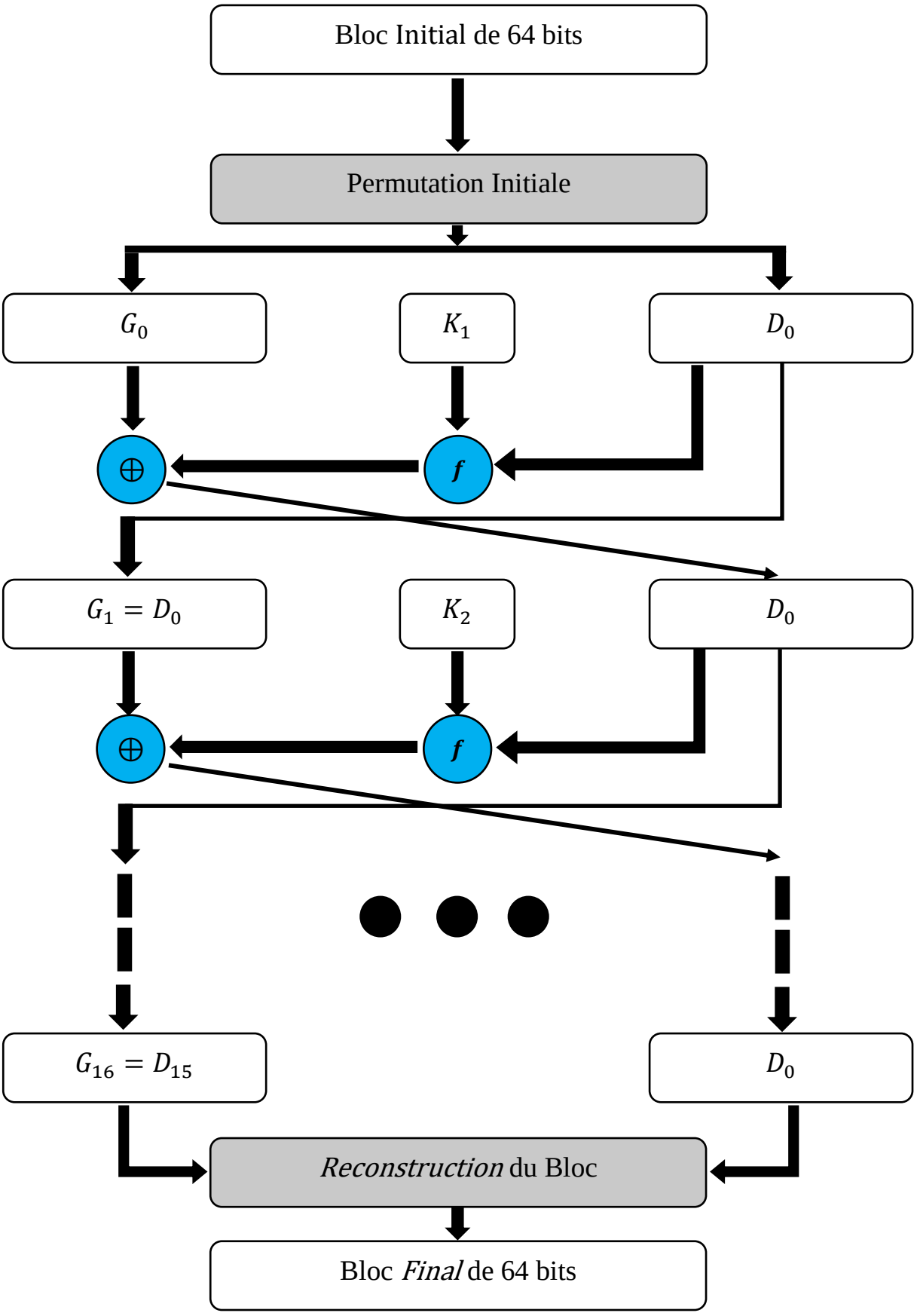


Figure 1-5 : *Diagramme de DES*

1.4.4 Chiffrement de AES (Advanced Encryption System)

Au fil du temps et des progrès de l'informatique, 2^{56} clés DES possibles ne sont plus une barrière insurmontable. Désormais, même avec des moyens modestes, il est possible de déchiffrer un message crypté DES en un temps raisonnable. En janvier 1997, le NIST américain (National Institute of Standards and Technology) lance un appel d'offres pour développer AES (Advanced Encryption System). Le cahier des charges comprend les points suivants :

- De toute évidence, très sécurisé.
- Portabilité étendue : Remplaçant l'algorithme de DES, il peut être utilisé à la fois dans les cartes à puce dotées de processeurs 8 bits à faible consommation et dans les processeurs spécialisés pour chiffrer des milliers de télécommunications à la volée.
- La rapidité.
- L'algorithme est facile à lire car il est destiné à être public.
- Techniquement, le chiffrement doit se faire en blocs de 128 bits, avec des clés composées de 128, 192 ou 256 bits.

A la date limite de candidature du 15 juin 1998, 21 projets avaient été soumis. Certains sont l'œuvre d'entreprises (IBM, etc.), d'autres sont l'œuvre d'universitaires en pool (CNRS, etc.), ces derniers étant rédigés par peu de personnes. Pendant deux ans, les algorithmes ont été évalués par des experts, un forum de discussion a été mis en place sur Internet et des conférences ont été organisées. Le 2 octobre 2000, le NIST donne la réponse : Rijndael, un algorithme développé par 2 Belges, Joan Daemen et Vincent Rijmen, est choisi. Depuis, Rijndael, devenu AES, s'est largement déployé et a progressivement remplacé DES.

Rijndael fonctionne en blocs de 128 bits et possède également des clés de 128 bits. Chaque bloc subit une séquence de 5 transformations, répétée 10 fois.

Début

1. Ajoutez la clé (via l'opération XOR).
2. Transformation non linéaire des octets : 128 bits sont divisés en 16 blocs de 8 bits, eux-mêmes ordonnancés dans une table 4x4. Chaque octet est transformé par une fonction non linéaire S. S peut simplement être considéré comme un remplacement pour un entier compris entre 1 et 256. En particulier, il peut être implémenté sur un ordinateur avec un simple tableau.
3. Décalage de ligne : faites pivoter les 3 dernières lignes vers la gauche : la ligne 2 déplace 1 colonne, la ligne 3 déplace 2 colonnes et la ligne 4 déplace 3 colonnes.
4. Brouillage de colonne : Chaque colonne est transformée par une combinaison linéaire des différents éléments de la colonne (équivalent à multiplier une matrice 4x4 par une autre matrice 4x4). Les calculs d'octets de 8 bits sont effectués dans un corps de 28 éléments.
5. Ajout de clés de tour : A chaque tour, la clé de tour est générée à partir de la clé par un sous-algorithme (appelé timing). Cette clé en main est obtenue par exclusivité ou adjonction au dernier bloc.

Fin

Algorithme 1-2 : *Etapes de la cryptographie en utilisant la méthode AES.*

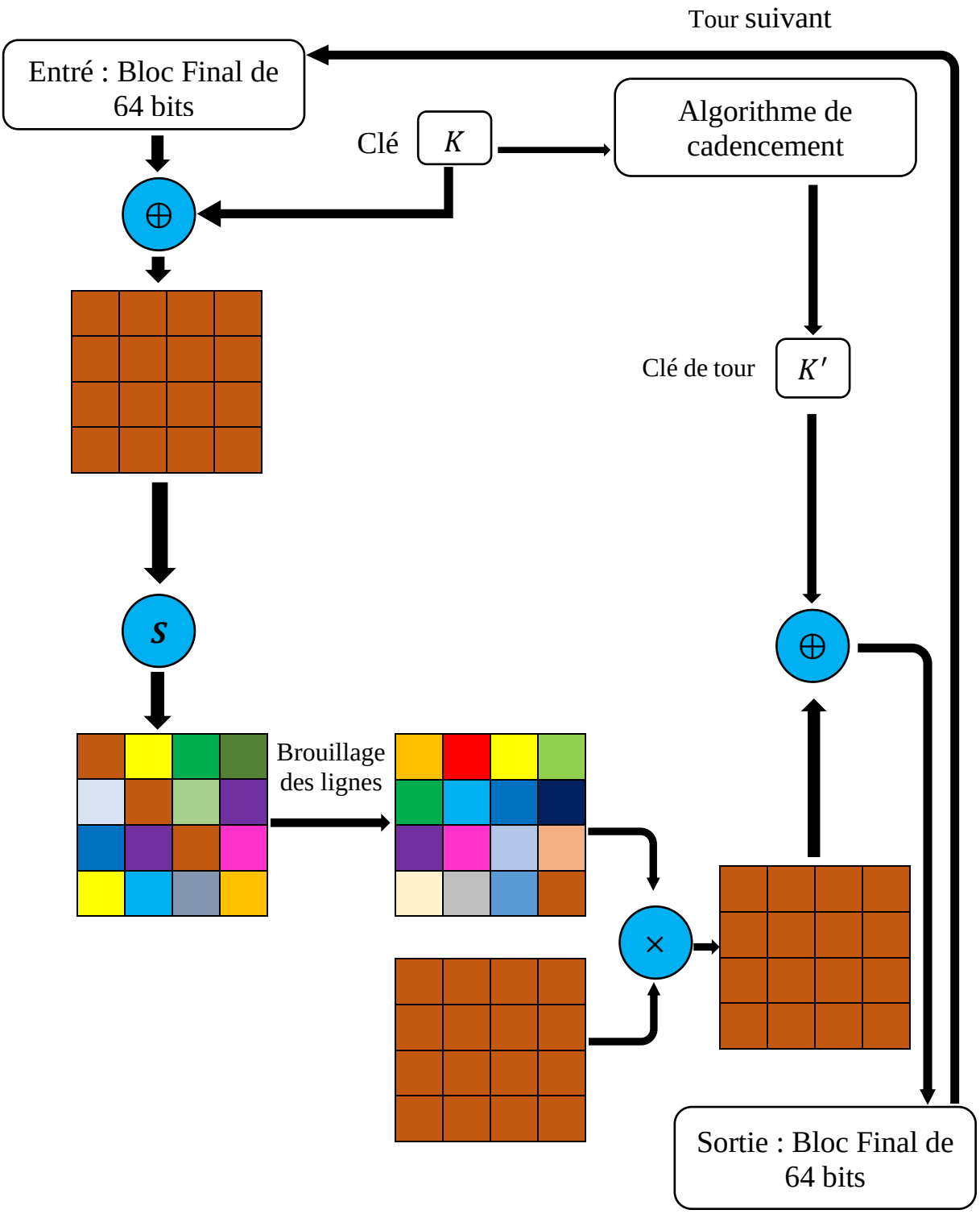


Figure 1-6 : *Diagramme de AES*

Une description exacte des opérations effectuées dans AES nécessite quelques connaissances en algèbre, en particulier l'arithmétique des polynômes dans des corps de 2^8 éléments.

1.5 Cryptographie Asymétrique

Dans les années 1970, la cryptographie n'était plus l'apanage des militaires. Les banques, en particulier, sont devenues de grands consommateurs d'informations cryptées pour la sécurité des transactions. Les chiffres disponibles à l'époque, comme le fameux DES, étaient fiables, du moins par rapport aux méthodes disponibles à l'époque pour les surmonter. Ils sont sécurisés, à condition que le destinataire et l'expéditeur parviennent à échanger des clés de chiffrement. Les États utilisent la valise diplomatique pour ces échanges, mais les civils ne le peuvent pas.

En 1976, Whitfield Diffie et Martin Hellman ont proposé une nouvelle approche pour contourner ce piège. Commençons par expliquer cela visuellement. Un ami doit vous envoyer un message très important par la poste, mais vous ne faites pas confiance au facteur que vous soupçonnez d'ouvrir votre lettre. Comment puis-je m'assurer que je reçois ce message sans être lu ? Vous commencez par envoyer à votre ami un cadenas sans clé mais en position ouverte. Ensuite, il fourre les informations dans une boîte qui est verrouillée avec un cadenas, et il vous envoie la boîte. Le facteur ne peut pas ouvrir la boîte car elle ne peut être ouverte que si vous avez la clé.

La cryptographie à clé publique (Asymétrique) est basée sur ce principe. Nous avons une fonction P qui nous permet de chiffrer un message. Ce processus est réversible, ce qui signifie que nous avons une fonction de déchiffrement S. On peut générer un couple (P, S) en même temps, mais ne connaissant que P, il est impossible (ou du moins très difficile) de trouver S.

- P est la clé publique (cadenas) et vous pouvez la révéler à n'importe qui. Si Louis veut vous envoyer un message X, il vous enverra $P(X)$.
- S est la clé (clé du cadenas) et c'est toujours la seule chose que vous ayez. Vous décodez le message X en calculant $S(P(X)) = X$.
- La connaissance de P par un tiers ne compromettrait pas la sécurité de l'envoi du message codé, car il ne serait pas en mesure de trouver S. Il est possible de donner librement P, sa véritable clé publique.

Bien sûr, il reste une difficulté : comment trouver de telles fonctions P et S ? Diffie et Hellman eux-mêmes n'ont pas trouvé de fonction satisfaisante, mais dès 1977 D. Rivest, A. Shamir et L. Adleman ont trouvé une solution possible, la meilleure et la plus largement utilisée à ce jour la cryptographie RSA. Le RSA est basé sur la dichotomie suivante :

- Faire de grands nombres premiers p et q est facile (idée fixe, 500 chiffres).
- Étant donné un produit de 2 grands nombres premiers avec un entier $n = p \times q$, il est difficile de trouver les facteurs p et q.

La donnée de n est la clé publique : il suffit de chiffrer. Pour déchiffrer, les p et q qui composent la clé privée doivent être connus. Le problème de la factorisation des grands entiers est très difficile, sachant que la clé publique n ne permet pas de trouver les entiers p et q qui composent la clé.

Cela dit, les algorithmes à clé publique (également connus sous le nom de chiffrement asymétrique) sont rarement utilisés pour chiffrer entièrement les longs messages. Ils sont en effet très lents, beaucoup plus lents que leurs homologues symétriques. Ils ne sont pas disponibles en pratique pour les applications qui doivent échanger de grandes quantités de données. Utilisez ensuite un cryptosystème hybride. Choisissez un chiffrement symétrique (AES, etc.) pour échanger des messages. Les clés de ce chiffrement symétrique sont ensuite échangées à l'aide de la cryptographie à clé publique. De nombreux protocoles utilisés en pratique (SSL pour le e-commerce, logiciel PGP pour la messagerie) fonctionnent sur ce principe.

Soulignons également la différence essentielle entre la sécurité des chiffrements asymétriques (RSA, etc.) et la sécurité des chiffrements symétriques (AES, etc.). La sécurité de ce dernier repose uniquement sur des heuristiques. Nous avons compliqué le message, nous l'avons "randomisé" suffisamment pour le rendre illisible à quiconque ne connaît pas la clé. Pour les nombres asymétriques, la sécurité repose sur des problèmes mathématiques très spécifiques considérés comme insolubles (décomposition d'entiers, recherche de logarithmes dans des groupes discrets, etc.). Au final, il existe très peu d'algorithmes à clé publique différents, car nous connaissons très peu ce genre de problèmes. Par exemple, une grande percée théorique (telle que la découverte d'un algorithme de factorisation rapide) pourrait compromettre la sécurité de toute la cryptographie à clé publique basée sur RSA.

1.5.1 Chiffrement de Diffie et Hellman

Suite à leur découverte des principes de la cryptographie à clé publique, Diffie et Hellman ont proposé en 1976 un protocole d'échange de clé entièrement sécurisé. Voici le problème. Alice et Bob veulent échanger des messages chiffrés à l'aide d'un algorithme qui nécessite la clé K. Ils veulent échanger cette clé K, mais ils n'ont pas de canal sécurisé. Le protocole d'échange de clés de Diffie et Hellman résout ce problème lorsque K est un entier. Il est basé sur l'arithmétique modulo et repose sur les hypothèses suivantes :

Soient des entiers p, a, x, où p est premier et $1 \leq a \leq p-1$:

- Il est facile de calculer l'entier $y = a^x \ mod \ p$
- Si nous connaissons $y = a^x \ mod \ p$, a et p, alors tant que p est suffisamment grand, il sera difficile de trouver x.

Connaître $a^x \ mod \ p$, a et p pour trouver x s'appelle résoudre le problème du logarithme discret. Quant à la factorisation des entiers, c'est un problème pour lequel nous n'avons pas d'algorithme efficace.

Expliquons maintenant comment Alice et Bob échangent des clés en utilisant le protocole Diffie-Hellman. Ils effectuent des opérations en parallèle, comme décrit dans le tableau suivant :

	Alice	**Bob**
Phase 1	Alice et Bob choisissent ensemble un nombre premier p très grand et un entier $1 \leq a \leq p-1$. Cet échange n'a pas besoin d'être sécurisé.	
Phase 2	Alice choisit d'une façon secrète x_1	Bob choisit d'une façon secrète x_2
Phase 3	Alice fait le calcul de $y_1 = a^{x_1} \ mod \ p$	Bob fait le calcul de $y_2 = a^{x_2} \ mod \ p$
Phase 4	Alice et Bob s'échangent les valeurs de y_1 et y_2. Cet échange n'a pas besoin d'être sécurisé.	
Phase 5	Alice fait le calcul de	Bob fait le calcul de

	$y_2^{x_1} = (a^{x_2})^{x_1} = a^{x_1 x_2} \ mod \ p$ et appelle ce nombre K, la clé secrète à partager avec Bob.	$y_1^{x_2} = (a^{x_1})^{x_2} = a^{x_1 x_2} \ mod \ p$ et appelle ce nombre K, la clé secrète à partager avec Alice.

Algorithme 1-3 : *Procédure d'exécution de l'algorithme Diffie-Hellman.*

Donc, à la fin du protocole, Alice et Bob ont la même clé K, et ils n'échangent pas directement cette clé. Si quelqu'un écoute sa conversation, il sait p, a, y_1 et y_2. Il ne peut pas trouver K comme Alice ou Bob car il lui manque encore l'une des informations nécessaires, qui est x_1 ou x_2. Connaître $y = a^{x_1} \ mod \ p$, a et p ne permet pas non plus de trouver x_1 car la résolution de logarithmes discrets est un problème difficile.

Cette découverte de Diffie et Hellman a été une véritable révolution dans l'histoire de la cryptographie. Le problème d'échange de clé est en effet résolu. Cependant, ce protocole a un défaut : il nécessite des actions simultanées d'Alice et de Bob. Si Alice voulait envoyer un e-mail à Bob pendant qu'il dormait ou qu'il était hors ligne, elle ne pourrait pas le faire tout de suite. C'est pourquoi le protocole a en fait été rapidement remplacé par des méthodes de chiffrement à clé publique telles que RSA, qui étaient accessibles à tous. Cependant, il est utilisé pour deux problèmes d'appairage dans la technologie Bluetooth.

1.5.2 Chiffrement de RSA

Ron Rivest, Adi Shamir et Len Adleman ont inventé la méthode de chiffrement RSA en 1977, après que Diffie et Hellman ont découvert le chiffrement à clé publique. RSA reste le cryptosystème à clé publique le plus largement utilisé aujourd'hui. Fait intéressant, son invention était accidentelle : au départ, Rivest, Shamir et Adleman voulaient prouver que tout système à clé publique était défectueux. Comment ça marche : Si Bob veut recevoir un message en utilisant RSA, il fait ce qui suit :

Début

Créer une clé : Bob crée 4 nombres $p, q, e \ et \ d$:

- $p \ et \ q$ sont deux grands nombres premiers différents. Leur génération se fait aléatoirement, en utilisant un algorithme probabiliste de test premier.
- e est un nombre premier avec le produit $(p-1) \times (q-1)$.

- d est définit tel que $e \times d = 1\ mod\ (p-1) \times (q-1)$. En d'autres termes, $e \times d - 1$ est un multiple de $(p-1) \times (q-1)$. Nous pouvons utiliser l'algorithme euclidien pour fabriquer d de $e, p\ et\ q$.

Distribution des clés : Le couple (n, e) constitue la clé publique de Bob. Il la rend disponible par exemple en la mettant dans un annuaire. Le couple (n, d) constitue sa clé privée. Il la garde secrète.

Envoi du message codé : Alice veut envoyer un message codé à Bob. Elle le représente sous la forme d'un ou plusieurs entiers M compris entre 0 et $n-1$. Alice possède la clé publique (n, e) de Bob. Elle calcule $C = M^e\ mod\ n$. C'est ce dernier nombre qu'elle envoie à Bob.

Réception du message codé : Bob reçoit C, et il calcule grâce à sa clé privée $D = C^d\ mod\ n$. D'après un théorème du mathématicien Euler, $D = M^{d \times e} = M\ mod\ n$. Il a donc reconstitué le message initial.

Fin

Algorithme 1-4 : *Etapes de la cryptographie en utilisant la méthode RSA.*

La sécurité de RSA repose sur le fait que connaissant $n, e\ et\ M^e\ mod\ n$, si on ne connaît pas d, il est difficile de trouver M. On dit que le problème du calcul des logarithmes discrets est difficile modulo n. En fait, le seul moyen connu est de trouver l'exposant secret d en décomposant n en produit $p \times q$. C'est pourquoi on dit que la sécurité du RSA repose sur la difficulté de factoriser de grands entiers dans le produit de deux nombres premiers.

Par conséquent, la plupart des attaques sur la partie théorique de RSA sont basées sur des tentatives de factorisation d'entiers. Actuellement, c'est encore un problème très difficile et nous ne connaissons pas d'algorithme efficace. Le tableau suivant résume certains enregistrements de factorisation d'entiers.

Tableau 1-4 : *Factorisation des entiers*

Nombre de chiffres de n	**Tailles en bits**	**Date de factorisation**
232	768	2009
200	664	2005
174	576	2003
155	512	1999

Ce tableau montre des progrès lents, mais vrais. Et nous savions en 2009 factoriser une clé de 768 bits, qui était la longueur de clé encore largement utilisée au début des années 2000. En 2012, les clés de 1024 bits étaient obsolètes, nous recommandions des clés de 2048 bits pour les clés privées et des clés de 4096 bits minimum pour une utilisation sensible. En 2020, le plus grand numéro RSA factorisé connu du public avait 829 bits (250 chiffres décimaux, RSA-250).

Si nous ne savons pas actuellement comment factoriser de grands nombres entiers, les choses pourraient changer plus ou moins dans un avenir proche si nous parvenons à construire des ordinateurs quantiques. Ces opérateurs peuvent utiliser des méthodes de décomposition efficaces. Bien sûr, il reste encore des difficultés techniques à fabriquer un tel ordinateur, et nous ne savons pas s'il sera surmonté un jour, donc pour l'instant nous savons comment penser au prototype actuel.

1.5.3 Chiffrement de ElGamal

Le chiffrement El-Gamal est une méthode de chiffrement à clé publique inventée par Taher ElGamal en 1985. Sa sécurité, comme les protocoles Diffie et Hellman, repose sur la difficulté de calculer des logarithmes discrets. Supposons que Bob veuille envoyer un message à Alice :

Début

- Alice choisit deux paramètres non-secrets : un nombre premier p et une racine primitive de p (un nombre g dont les puissances modulo p engendrent $\mathbb{z}_p - \{0\}$).
- Elle choisit aléatoirement un nombre a dans l'intervalle $[1,2, \dots, p-2]$ et calcule $\alpha = g^a \ mod \ p$.

- Elle publie sa clé (p, g, α) et garde secrète sa clé a.
- Bob veut envoyer le message m à Alice, en le représentant comme un nombre compris entre 0 et $p-1$ (même si cela signifie le décomposer en sous-blocs de longueur correcte).
- Il choisit aléatoirement un nombre b dans l'intervalle $[1,2,\dots,p-2]$ et calcule $\beta = g^b \ mod \ p$.
- Il crypte alors son message m sous la forme $m' = \alpha^b \times m \ mod \ p$.
- Il transmet finalement le couple (β, m') à Alice.
- Pour décrypter le message de Bob, Alice détermine le nombre $x = p - 1 - a$.
- Elle calcule $\beta^x \times m' \ mod \ p$ et retrouve le message initial m.

Fin

Algorithme 1-5 : *Etapes de la cryptographie en utilisant la méthode ElGamal.*

Ce cryptosystème repose sur la difficulté de résoudre des logarithmes discrets. Il a l'avantage d'utiliser deux processus aléatoires (le choix de a et b), mais l'inconvénient de générer des messages chiffrés particulièrement longs.

Le logiciel gratuit GNU Privacy Guard, les versions récentes de PGP et d'autres systèmes de cryptage utilisent cet algorithme. Il peut être utilisé pour le chiffrement, mais aussi pour les signatures électroniques, comme l'algorithme DSA du NIST.

Dans la plupart des cas, casser l'algorithme d'ElGamal est au moins aussi difficile que de calculer des logarithmes discrets. Cependant, il est possible de casser l'algorithme sans résoudre le problème du logarithme discret.

1.6 Signature électronique

Le paradigme de la signature électronique (également appelée signature numérique) est un processus qui garantit l'authenticité de l'expéditeur (une fonction d'authentification) et vérifie l'intégrité du message reçu.

Une signature électronique assure également la non-répudiation, c'est-à-dire qu'elle garantit que l'expéditeur a effectivement envoyé le message (en d'autres termes, elle empêche l'expéditeur de nier que le message a été envoyé).

1.6.1 Fonction de hachage

Une fonction de hachage (parfois appelée fonction de condensation) est une fonction qui prend un résumé de texte (également appelé condensé ou hachage, résumé de message), une séquence assez courte de caractères représentant le texte qu'elle condense.

Une fonction de hachage doit associer un et un seul hachage au texte en clair (ce qui signifie que toute modification du document entraînera la modification de son hachage). D'autre part, il doit s'agir d'une fonction à sens unique, de sorte qu'il est impossible de retrouver le message d'origine à partir du résumé. Une fonction de hachage est dite "secrètement cassée" s'il existe un moyen de trouver le message en clair à partir du hachage.

Ainsi, le hachage représente en quelque sorte l'empreinte digitale du document.

Les algorithmes de hachage les plus utilisés actuellement sont :

- **MD5 (MD signifie Message Digest) :** MD5 a été développé par Rivest en 1991 pour créer des empreintes digitales de 128 bits en traitant du texte de n'importe quelle taille en blocs de 512 bits. Les documents à télécharger sont souvent vus sur Internet avec un fichier MD5, qui est une version compressée du document, ce dernier peut être vérifié pour l'intégrité).
- **SHA** (Secure Hash Algorithm, qui peut être traduit par algorithme de hachage sécurisé) crée une empreinte digitale d'une longueur de 160 bits.
- **SHA-1** est une version améliorée de SHA, datant de 1994, qui produit une empreinte digitale de 160 bits en traitant des messages jusqu'à 264 bits de longueur dans des blocs de 512 bits.

1.6.2 Vérification de l’intégrité d’un message

En envoyant un message avec une valeur de hachage, l'intégrité du message est garantie, c'est-à-dire que le destinataire peut vérifier que le message n'a pas été altéré (intentionnellement ou non) pendant la communication.

Lorsque le destinataire reçoit le message, il calcule simplement le hachage du message reçu et le compare avec le hachage joint au document. Si le message (ou le hachage) est falsifié pendant la communication, les deux hachages ne correspondront pas.

1.6.3 Scellement des données

En utilisant une fonction de hachage, il est possible de vérifier que le hachage correspond au message reçu, mais il n'y a aucune preuve que le message a bien été envoyé par la personne supposée être l'expéditeur.

Par conséquent, afin de garantir l'authenticité du message, l'expéditeur n'a qu'à chiffrer (nous disons généralement la signature) le condensé avec sa propre clé privée (le hachage signé est appelé le sceau), puis envoyer le sceau au destinataire.

Après avoir reçu le message, le destinataire déchiffre simplement le sceau à l'aide de la clé publique de l'expéditeur et compare le hachage obtenu à l'aide de la fonction de hachage avec le hachage reçu en pièce jointe. Ce mécanisme de création de sceau est appelé un scellement.

1.7 Certificats

Les algorithmes de chiffrement asymétriques sont basés sur le partage de clés publiques entre différents utilisateurs. Typiquement, le partage de cette clé se fait via un annuaire électronique (généralement au format LDAP) ou un site internet.

Cependant, ce modèle de partage présente un inconvénient majeur : rien ne garantit que la clé soit bien la clé de l'utilisateur auquel elle est associée.

En fait, un pirate peut compromettre la clé publique dans l'annuaire en remplaçant la clé publique dans l'annuaire par sa clé publique.

Ainsi, les pirates pourront déchiffrer tous les messages qui ont été chiffrés avec les clés présentes dans l'annuaire.

Ainsi, un certificat peut associer une clé publique à une entité (humaine, machine, etc.) pour assurer sa validité. Un certificat est en quelque sorte une carte d'identité pour une clé publique, délivrée par une organisation appelée autorité de certification (généralement désignée par CA, autorité de certification).

L'organisme de certification est chargé de délivrer le certificat, de lui attribuer une date de péremption (équivalente à la date de péremption de l'aliment), et éventuellement de révoquer le certificat avant cette date si la clé est compromise (ou le propriétaire).

1.7.1 Structure d’un certificat

Un certificat est un petit fichier divisé en deux parties :

- La partie contenant les informations.
- La partie contenant la signature de l’autorité de certification.

La structure du certificat est normalisée par la norme X.509 de l’UIT (plus précisément X.509v3), qui définit les informations contenues dans le certificat :

- La version X.509 correspondant au certificat.
- Le numéro de série du certificat.
- L'algorithme de chiffrement utilisé pour signer le certificat.
- Le nom (DN, Distinguished Name) de l'autorité de certification émettrice.
- La date de début de la période de validité du certificat.
- La date d'expiration du certificat.
- Le but de l'utilisation de la clé publique.
- La clé publique du propriétaire du certificat.
- La signature (empreinte digitale) de l'émetteur du certificat.

Toutes ces informations (message + clé publique du demandeur) sont signées par l'autorité de certification, ce qui signifie qu'une fonction de hachage crée une empreinte digitale de l'information puis crypte ce résumé à l'aide de l'autorité de certification privée de la clé ; clé publique auparavant largement diffusée pour permettre aux utilisateurs pour vérifier les signatures avec la clé publique de l'autorité de certification.

Lorsqu'un utilisateur souhaite communiquer avec une autre personne, il lui suffit d'obtenir le certificat du destinataire.

Le certificat contient le nom du destinataire, ainsi que sa clé publique, et est signé par une autorité de certification.

Ainsi, la validité du message peut être vérifiée en appliquant d'abord une fonction de hachage aux informations contenues dans le certificat, puis en déchiffrant la signature de l'autorité de certification à l'aide de la clé publique de cette dernière et en comparant les deux résultats.

1.7.2 Niveau de signature

Selon le niveau de signature, il existe différents types de certificats :

- Les **Certificats Auto-Signés** sont des certificats à usage interne. Ce type de certificat est signé par le serveur local et garantit la confidentialité des échanges au sein d'une organisation, par exemple pour des besoins intranet. Par conséquent, l'authentification de l'utilisateur peut être effectuée à l'aide d'un certificat auto-signé.
- Les **Certificats Signés Par Les Autorités De Certification** sont nécessaires pour assurer la sécurité des communications avec les utilisateurs anonymes, comme dans le cas des sites Web sécurisés accessibles au public. Un vérificateur tiers garantit aux utilisateurs que le certificat appartient à l'organisation à laquelle ils prétendent appartenir.

1.7.3 Types d'usages

Les certificats sont principalement utilisés dans trois types de contextes :

- Le **Certificat Client**, stocké sur le poste de travail de l'utilisateur ou embarqué dans un conteneur tel qu'une carte à puce, identifie l'utilisateur et lui associe des autorisations. Dans la plupart des cas, il est transféré au serveur lors de la connexion et le serveur attribue des autorisations en fonction des informations d'identification de l'utilisateur. Il s'agit d'une véritable identification numérique qui utilise une paire de clés asymétrique d'une longueur de 512 à 1024 bits.
- Le **Certificat Serveur** installé sur le serveur web assure le lien entre le service et le propriétaire du service. Dans le cas d'un site Web, il est garanti que l'adresse (URL), en particulier le domaine de la page Web, appartient à une société spécifique. De plus, grâce au protocole SSL, il protège les transactions avec les utilisateurs.
- Le **certificat VPN** est un certificat installé dans un périphérique réseau qui crypte le flux de communication de bout en bout entre deux points, tels que deux sites d'une entreprise. Dans ce scénario, l'utilisateur dispose d'un certificat client, le serveur implémente le certificat serveur et le dispositif de communication utilise un certificat spécifique (généralement un certificat IPSec).

1.8 Cryptographie des images couleurs

Par rapport au chiffrement traditionnel, la sécurité n'est peut-être pas l'objectif le plus important d'un système de chiffrement d'images. Selon le type d'application, d'autres propriétés telles que la vitesse peuvent être tout aussi importantes. Par exemple, les termes « cryptage logiciel » ou « cryptage sélectif » sont parfois utilisés à la place des schémas de cryptage « durs » classiques tels que le cryptage AES complet dans ce cas. Le but de ces schémas n'est pas d'offrir une sécurité maximale, mais plutôt un compromis entre la sécurité et la complexité de calcul. Ils sont conçus pour protéger le contenu multimédia et répondre aux exigences de sécurité d'applications multimédia spécifiques. Par exemple, le cryptage en temps réel d'un flux vidéo entier à l'aide d'un cryptage classique nécessite beaucoup de calculs en raison de la grande quantité de données impliquées. D'autre part, de nombreuses applications multimédia nécessitent un niveau de sécurité beaucoup plus faible (par exemple, les émissions de télévision) ou n'ont besoin de protéger leurs données que pendant une courte période (par exemple, les émissions d'actualités). Par conséquent, les applications de sécurité multimédia doivent rechercher des programmes de cryptage rapides et particulièrement adaptés à l'environnement cible. Les domaines d'intérêt pour le déploiement de telles solutions vont des systèmes de gestion des droits numériques (DRM) aux communications personnelles sécurisées.

1.9 Sécurité fournie par l'infrastructure

En raison de la grande quantité de données impliquées, les images et les vidéos (également appelées données visuelles) nécessitent une capacité de stockage ou une bande passante de transmission massive. Seul le chiffrement symétrique (plutôt que le chiffrement à clé publique) peut fournir des performances d'exécution raisonnables lors du chiffrement d'aussi grandes quantités de données. Comme cela se fait dans la plupart des applications actuelles qui nécessitent la confidentialité, les techniques à clé publique ne sont utilisées que pour l'échange de clé ou la génération de signature (de tels schémas sont communément appelés "hybrides").

Il existe deux méthodes pour assurer la confidentialité d'une application de stockage ou de transmission. Premièrement, l'infrastructure informatique sous-jacente fournit des mécanismes de confidentialité. L'avantage est une transparence totale, ce qui signifie que l'utilisateur ou une application spécifique n'a pas à se soucier de la confidentialité. L'inconvénient évident est que

la confidentialité est assurée pour toutes les applications, requises ou non, et que les propriétés spécifiques de certaines applications ne peuvent pas être exploitées. Prenons l'exemple de l'infrastructure de base de données médicale distribuée mentionnée dans l'introduction.

Si les connexions entre les composants sont basées sur des connexions Internet TCP/IP (qui ne sont pas confidentielles en elles-mêmes), la confidentialité peut être assurée en établissant un réseau privé virtuel (VPN) utilisant IPSec (qui étend le protocole IP en ajoutant des fonctionnalités de confidentialité et d'intégrité). Dans ce cas, l'ensemble des données visuelles est crypté pour chaque transmission, ce qui met à rude épreuve le système de cryptage. La couche application est la deuxième option pour assurer la confidentialité. Seuls les applications et services nécessitant la confidentialité sont ici protégés.

L'inconvénient est que chaque application doit assurer sa propre confidentialité ; l'avantage est que des propriétés spécifiques de certaines applications peuvent être utilisées pour créer des schémas de chiffrement plus efficaces, ou le chiffrement peut être omis s'il n'est pas requis. Le chiffrement sélectif des données visuelles tire parti de la redondance des données visuelles au niveau de l'application et tombe donc dans la deuxième catégorie.

1.10 Chiffrement sélectif et chiffrement complet

Divers systèmes de cryptage pour les types de données visuelles ont été présentés ces dernières années, car les approches visant à garantir le secret doivent être correctement conçues pour sécuriser le contenu multimédia et répondre aux exigences de sécurité d'une application multimédia spécifique.

La technique la plus simple et la plus sécurisée consiste à prendre le flux binaire multimédia et à le crypter avec un chiffrement fort sur le plan cryptographique tel que AES. En raison de la complexité de la méthode de chiffrement requise, un tel système ajoute automatiquement une latence importante, qui entre souvent en conflit avec les restrictions en temps réel. Des approches plus efficaces ont été proposées car la vitesse d'exécution est souvent cruciale dans l'encodage et le décodage vidéo. Ces méthodes, souvent appelées systèmes de chiffrement "sélective" ou "soft", compensent généralement les performances d'exécution pour la sécurité, et sont donc légèrement moins sécurisées que la méthode naïve. Alors que les approches de chiffrement sélectif utilisent des structures de données spécifiques à l'application pour créer des systèmes de chiffrement plus efficaces à l'aide de chiffrements "classiques" sécurisés mais lents,

les systèmes de chiffrement doux utilisent des systèmes de chiffrement plus faibles (tels que des permutations) pour accélérer le traitement.

Le chiffrement sélectif semble être un concept intelligent dans tous les cas, car la réduction de la charge de calcul associée aux applications de traitement du signal est généralement souhaitable. Cependant, par rapport au cryptage complet, la sécurité de ces techniques est toujours pire. La seule raison d'accepter cet inconvénient est d'économiser des quantités considérables de temps de traitement ou d'électricité. Par conséquent, le contexte dans lequel le chiffrement sélectif doit être utilisé doit être soigneusement étudié avant de décider si son utilisation est appropriée.

1.11 Propriétés du chiffrement d'images

De plus, les méthodes de chiffrement d'image doivent vérifier deux propriétés :

- Conformité au format : l'image cryptée doit être au même format que l'image d'origine,
- Préservation de la taille : la taille de l'image cryptée doit être identique (ou très proche) à la taille de l'image d'origine.

La première propriété indique que les données avant et après chiffrement doivent être structurées de la même manière. Par conséquent, les méthodes de cryptage traditionnelles ne peuvent pas être directement appliquées aux images sans tenir compte de la particularité du format de l'image. Notez également que la conformité à l'attribut format signifie que les données cryptées peuvent être visualisées avec le même éditeur d'image que les données d'origine.

Concernant la deuxième propriété, on observe qu'une augmentation limitée de la taille des données après chiffrement peut être tolérée dans certaines applications. De plus, selon Shannon, les méthodes de chiffrement doivent introduire la confusion et la diffusion. Le principe d'obfuscation appliqué aux images correspond à la volonté de complexifier au maximum la relation entre clés de chiffrement et images chiffrées. Les propriétés de diffusion, d'autre part, indiquent que la redondance statistique entre les pixels de l'image claire doit se dissiper dans les statistiques de l'image cryptée. En d'autres termes, il ne doit pas y avoir de corrélation entre les pixels de l'image claire se trouvant dans l'image chiffrée. Pour introduire un crénelage et augmenter la dispersion, des opérations de permutation et de remplacement de pixels sont effectuées.

Si les algorithmes de chiffrement classiques décrits dans le chapitre précédent peuvent être adaptés pour les images, des algorithmes spécifiques ont également été développés pour ce type de données. Notez que ces algorithmes sont généralement basés sur des méthodes de chiffrement symétriques. Le chiffrement asymétrique est moins utilisé en raison de la grande quantité de données à traiter. En effet, il est beaucoup plus coûteux en temps de calcul et augmente la taille des données d'origine. Dans un premier temps, nous détaillons une méthode originale de chiffrement d'images basée sur le principe d'utilisation et de substitution ou de permutation de GNPA. Ensuite, nous expliquons comment la théorie du chaos s'applique au chiffrement d'images. Enfin, nous décrivons la méthode chiffrer-puis-compresser, qui permet de chiffrer une image puis de la compresser.

1.11.1 Méthodes naïves

Les méthodes de chiffrement d'images basées sur des opérations de substitution et/ou de permutation [1]–[3] impliquent l'utilisation de GNPA cryptographiquement sécurisé. Les données fournies à ces générateurs sont la clé K, le type d'éléments et la longueur de la séquence à générer.

De plus, certains auteurs suggèrent de randomiser les positions non pas de tous les pixels de l'image, mais des lignes et des colonnes ou même des blocs de pixels de l'image.

Des méthodes de cryptage alternatives ont également été développées. Ils sont basés sur l'opération XOR [4]–[22] entre une séquence pseudo-aléatoire générée et le contenu clair de l'image. De plus, cela peut se faire de deux manières différentes. Les images peuvent être chiffrées pixel par pixel, bit par bit (par exemple, du bit le moins significatif au bit le plus significatif) ou plan binaire (par exemple, du bit le moins significatif au bit le plus significatif).

Bien que cette méthode de chiffrement soit facile à mettre en œuvre, elle présente l'inconvénient de conserver certaines propriétés statistiques de l'image d'origine. En revanche, la répartition des pixels substitués dans l'image cryptée est proche d'une répartition uniforme. En effet, si des pixels voisins ont des valeurs similaires et sont fortement corrélés dans le domaine clair, ce n'est pas le cas dans le domaine chiffré en utilisant cette approche. Ainsi, contrairement aux résultats obtenus avec les chiffrements mélangés, l'analyse des images chiffrées par substitution montre que le contenu original des images en clair est visuellement confidentiel.

1.11.2 Méthodes basées sur le chaos

Beaucoup de chercheurs sont dirigés vers ce domaine de la cryptographie des images [23]–[28] afin de développer des algorithmes qui conviennent la particularité des données multimédias. La plupart de ces algorithmes sont basés sur des fonctions chaotiques qui ont des bonnes caractéristiques en termes de la sensibilité des conditions initiales, les paramètres de système, la non-périodicité et le pseudo aléatoire des valeurs générées.

Parmi les travaux réalisés à propos de ce domaine, on trouve Huang et al [29] qui ont proposé un système de cryptage basé sur l'algorithme de Gerchberg–Saxton Modifié pour crypter l'image source, puis ils ont utilisé « Josephus traversing » pour changer les positions des pixels de l'image cryptée d'une manière aléatoire, et par la suite, un système hyper-chaotique a été utilisé pour la phase de confusion. Kandar et al [30] ont proposé un nouveau système de cryptage basé sur les techniques de permutation, ce système bénéfice des propriétés des groupes cycliques. Broumandnia [31] a développé un algorithme de cryptage d'image basé sur des cartes chaotiques modulaires réversibles, ces cartes sont réversibles et de type 2D et 3D avec des mathématiques modulaire afin d'augmenter l'espace de clé. Un algorithme de cryptage d'image multi-tours est proposé et implémenté dans le travail de Li et al [32], cet algorithme est Basé sur un générateur de nombres aléatoires amélioré qui est principalement conçu par la fonction d'erreur de Gauss et la technologie d'élimination numérique.

Hu et al [33] ont développé Un nouveau système chaotique nommé UT-CCS, ce système a comme but de générer des nouvelles cartes chaotiques avec d'excellentes performances. Ils ont développé aussi un nouveau générateur de nombres pseudo-aléatoires en utilisant le phénomène du chaos, ce générateur nommé (CBPRNG) a des bonnes caractéristiques aléatoires qui lui permet d'être utilisé dans un algorithme de cryptage des images couleurs.

Zhou et al [34] ont proposé une nouvelle méthode basée sur un mécanisme de diffusion en boucle fermée entre blocs qui dépend à la fois de ce qu'on appelle « Plaintext » et « Ciphertext ». Ce mécanisme de diffusion en boucle fermée est complètement différé des autres mécanismes de diffusion en boucle ouverte existant, puisqu'il peut d'une part, améliorer significativement la fiabilité du système de cryptage, d'autre part, il implémente le processus de diffusion complet qui consiste à crypter le premier bloc de pixels du « PlainText » pour crypter le dernier bloc de pixels, et par la suite, modifier le premier bloc du « Ciphertext ».

1.12 Méthodes basées sur les opérations des séquences DNA

Il existe d'autres travaux qui s'intéressent à la cryptographie des images couleurs en utilisant des opérations des séquences DNA [35]–[38], [39, p. 2], [40]–[72]. Parmi ces travaux, on trouve Chai et al [73], qui ont proposé un système de cryptage basé sur des séquences d'acide désoxyribonucléique (DNA) et des fonctions chaotiques avec des paramètres renouvelés par « Haming Distance » de l'image originale. Dans Xiangjun et al [74], un nouvel algorithme de cryptage d'image couleur basé sur des opérations de séquence DNA avec des clés à usage unique et des fonctions chaotiques spatio-temporel. Les auteurs dans Chai et al [75] ont développé une nouvelle méthode de cryptage d'image en utilisant des fonctions chaotiques memristive avec des automates cellulaires 2D et des opérations de séquence DNA.

Jithin et al [76] ont proposé un système de cryptage d'images couleurs en se basant sur des fonctions chaotiques et un codage d'acide désoxyribonucléique (DNA). Ils ont choisi la transformation 2D « Arnold Cat Map » comme la meilleure transformation à utiliser avec le système de cryptage. Le choix de cette transformation est accompli tout en concevant une stratégie de sélection simple et appropriée. L'opération de cryptage est appliquée séparément sur chacun des trois canaux (rouge, vert et bleu) de l'image couleur afin d'améliorer la sécurité.

Ul Haq et al [77] ont suggéré une nouvelle technique de cryptage d'images couleurs en utilisant la notion de « Algebra-Chaos Amalgamated 256-length-12-bit random sequence » avec une transformation d'acide désoxyribonucléique (DNA). La séquence pseudo-aléatoire proposée est un S-box qui peut occuper juste un petit espace mémoire de l'ordinateur par rapport aux autres S-box existantes. De plus, pour plus de simplicité et de nouveauté, ils ont construit une nouvelle fonction mixte de type 1D à partir de deux fonctions chaotiques de même type, qui sont la fonction Sin et la fonction logistique.

1.12.1 Chiffrement-puis-Compression

La méthode Chiffrement-puis-Compression chiffre d'abord une image, puis la comprime. Le défi avec de telles méthodes [78]–[92] est que la compression doit être effectuée dans le domaine crypté, sans accès à la clé, ni à l'image originale en clair. À première vue, il apparaît que les données chiffrées ne peuvent pas être compressées car la redondance entre les pixels est supprimée par l'opération de chiffrement. Cependant, en 2004, Johnson et al ont affirmé que les données cryptées en flux sont compressibles sans changer leur sécurité visuelle [93]. En plus

des résultats théoriques, un algorithme pratique pour la compression sans perte d'images binaires cryptées est présenté. Par la suite, Kumar et Makur [94] ont appliqué cette approche aux erreurs de prédiction calculées dans le domaine crypté, ce qui a permis d'améliorer les taux de compression.

Lazzeretti et Barni [95] ont proposé plusieurs méthodes pour la compression sans perte d'images cryptées en niveaux de gris et en couleur. Les auteurs décomposent les images en plans binaires et exploitent les corrélations intra et inter-plans. De plus, d'autres méthodes de Chiffrement-puis-Compression sont basées sur le mélange de blocs d'images [96]–[119]. L'image originale est divisée en blocs de 16 × 16 pixels, qui sont ensuite disposés de manière pseudo-aléatoire à l'aide de la première clé. La deuxième touche sert à effectuer une rotation et/ou une inversion sur le bloc. La transformation s'applique également aux pixels de chaque bloc : avec la troisième clé, les bits de poids fort de certains pixels sont modifiés. Enfin, si l'image est en couleur, les trois composantes sont agencées selon la valeur de la quatrième clé. En utilisant ce procédé de cryptage, l'image cryptée obtenue est robuste à la compression JPEG conventionnelle.

Malgré des efforts considérables ces dernières années, les systèmes existants (chiffrement puis compression) sont encore très insuffisants en termes de performances de compression par rapport aux encodeurs d'images et vidéo avec ou sans perte de données. De plus, beaucoup d'entre eux ne sont pas sécurisés. En particulier, beaucoup sont vulnérables aux attaques de puzzle [120]. Le format d'image compressée le plus couramment utilisé est le JPEG. Par conséquent, de nombreuses approches suggèrent d'effectuer conjointement le chiffrement et la compression JPEG.

1.13 Conclusion

Dans ce chapitre, nous avons introduit plusieurs techniques cryptographiques et quelques théories qui nous permettront de comprendre cet axe de recherche très important de la sécurité de l'information. Nous avons d'abord discuté du concept formel de sécurité et de ses implications. Ensuite, nous avons exploré différentes classifications d'algorithmes de diffraction et leurs environnements d'application. Nous observons que les clés jouent un rôle important, et définir correctement leurs longueurs est essentiel pour assurer la sécurité des cryptosystèmes. Ce chapitre a présenté également les principaux algorithmes de chiffrement symétrique, asymétrique, par flux et par bloc, et décrit les différentes formes d'attaque et leurs

classifications. nous avons vu pourquoi Les méthodes de cryptage traditionnelles ne peuvent pas être directement appliquées aux images en raison de la forte corrélation entre les pixels dans les image ainsi que la capacité de données qui est énorme, c'est la raison pour laquelle des algorithmes comme AES et DES ne sont plus utilisées dans ce domaine, puisque l'espace clé n'est pas assez grand ce qui affecte sérieusement la confidentialité du réseau d'information, ainsi qu'elles prennent énormément du temps au niveau du temps de calcul. Par conséquent, nous avons présenté les différentes méthodes de cryptage des images non compressées. Les métriques et les critères d'évaluations utilisées pour évaluer le niveau de sécurité seront décrit dans les prochains chapitres.

Chapitre 2 : Outils Mathématiques pour la Cryptographie

2.1 Introduction

En 1949, Claude Shannon a publié un article intitulé **"The Communication Theory of Secrecy Systems"** dans le **Bell System Technical Journal** [121], qui a eu un impact considérable sur l'étude de la cryptographie. Il permet de donner un sens à la notion d'information contenue dans un message. A partir de cette théorie, il a défini en 1949, la notion de secret.

Dans ce chapitre, de nombreuses idées de Shannon seront détaillées [122], ainsi que quelques notions fondamentales pour mieux apprendre le domaine de la cryptographie telle que la théorie de l'information, la théorie de la complexité, et la théorie des nombres.

2.2 Théorie de l'information

2.2.1 Entropie et incertitude

La théorie de l'information définit la quantité d'informations dans un message comme le nombre minimum de bits requis pour coder toutes les significations possibles de ce message. Par exemple, le champ "jour de la semaine" d'une base de données ne contient pas plus de 3 bits d'information, car une information peut être encodée sur 3 bits :

000 = dimanche

001 = lundi

010 = mardi

011 = mercredi

100 = jeudi

101 = vendredi

110 = samedi

Si ces informations sont représentées par la chaîne ASCII correspondante, elles occuperont plus d'espace mémoire, mais elles ne contiendront pas plus d'informations. De même, le champ "sexe" dans la base de données ne contient qu'un seul bit d'information, bien qu'il semble qu'il doive être stocké comme l'une des deux chaînes ASCII à 5 caractères "homme" ou "femme".

Formellement, la quantité d'information dans un message M est mesurée par L'entropie d'un message, notée $H(M)$. L'entropie d'un message indiquant le sexe est de 1 bit ; l'entropie d'un message indiquant le jour de la semaine n'est que légèrement inférieure à 3 bits. En général, l'entropie d'un message est $log_2(n)$, où n est le nombre de significations possibles. Cela suppose que toutes les significations sont équiprobables.

L'entropie d'un message mesure également son incertitude. C'est le nombre de bits de texte en clair qui doivent être extraits pour extraire l'intégralité du texte en clair du texte chiffré. Par exemple, si le texte en clair du bloc de texte chiffré "QHP*5" est "masculin" ou "féminin", l'incertitude du message est de 1. Le cryptanalyste doit trouver un bit soigneusement choisi pour récupérer le message.

2.2.2 Taux de langage

Pour un langage donné, le Taux de langage est

$$r = H(M)/N \qquad (2\text{-}1)$$

Où N est la longueur du message. Pour les plus grandes valeurs de N, le taux de la langue anglaise courante prennent plusieurs valeurs entre 1 et 1,5 bit par lettre. Dans son livre, Shannon déclare que l'entropie dépend de la longueur du message. Plus précisément, il indique que les fragments de 8 lettres ont un débit de 2,3 bits par lettre, mais que les fragments de 16 lettres tombent entre 1,3 et 1,5 bits par lettre. Thomas Cover utilise des techniques de théorie des jeux pour découvrir 1,3 bits d'entropie par lettre. Le taux absolu d'une langue est le nombre maximum de bits qui peuvent être encodés par caractère, en supposant que toutes les séquences de caractères sont également probables. S'il y a L caractères dans la langue, le taux absolu R est :

$$R = log_2(L) \quad (2\text{-}2)$$

C'est l'entropie maximale des caractères isolés.

Pour l'anglais à 26 lettres, le rapport absolu est $log_2(26) = 4{,}7$ bits par lettre. Le calcule est le même pour n'importe quelle langue. Il n'est pas surprenant que le taux effectif d'anglais soit nettement inférieur au taux absolu : l'anglais est une langue très redondante.

La redondance d'un langage, notée D, est définie par :

$$D = R - r \quad (2\text{-}3)$$

L'anglais ayant un rapport de 1,3, la redondance est de 3,5 bits par lettre. La redondance de la langue française est un petit peu plus élevé. Cela signifie que chaque caractère fournit 3,5 bits d'informations redondantes. Un message ASCII en anglais clair, avec seulement 1,2 bits d'informations par message de 8 bits. Cela signifie qu'il y a 6,8 bits de redondance, c'est-à-dire

Cela donne une redondance globale de 0,85 bit d'information par bit de texte ASCII et une entropie de 0,15 bit d'information par bit de texte ASCII. Le même message codé par le système BAUDOT, 5 bits par caractère, avec 0,76 bits de redondance et 0,34 bits d'entropie par bit de texte BAUDOT. La prise en compte des espaces, de la ponctuation, des chiffres et de la mise en page modifie ces résultats.

2.2.3 Cryptosystème sécurisé

Shannon a défini un modèle mathématique précis qu'un cryptosystème sécurisé peut comprendre. Le but du cryptanalyste est de déterminer la clé K, le texte en clair M, ou les deux. Mais si M est une voix numérisée, un texte allemand, des données de feuille de calcul ou autre chose, ils peuvent se contenter d'informations probabilistes sur M. Pour la plupart des cryptanalyses du monde réel, le cryptanalyste dispose d'informations probabilistes sur M avant de commencer. Ils peuvent connaître le langage du texte en clair. Ce langage a une certaine redondance associée. Si c'était un message à Bob, ça pourrait commencer par "Cher Bob". Bien sûr, "Cher Bob" est plus probable que "e8T&g[,m".

Le but de la cryptanalyse est de modifier analytiquement les probabilités associées à tous les textes en clair possibles. Finalement, un texte en clair considéré comme certain (ou du moins très probable) émergera du pool de textes en clair possibles.

Il existe quelque chose que l'on peut appeler cryptosystème à confidentialité parfaite. Il s'agit d'un cryptosystème où le texte chiffré ne fournit aucune information sur le texte en clair (sauf peut-être sa longueur). Shannon suppose que cela n'est possible que si le nombre de clés possibles est au moins aussi grand que le nombre de messages possibles. En d'autres termes, la clé doit être au moins aussi longue que le message lui-même, et aucune clé ne peut être réutilisée. En d'autres termes, le masque jetable (comme nous l'avons vu dans le chapitre précédent) est le seul cryptosystème garantissant une parfaite confidentialité.

En plus de la confidentialité parfaite, un texte chiffré fournit des informations sur le texte en clair correspondant. C'est inévitable. Un bon algorithme cryptographique minimise ces informations ; un bon cryptanalyste utilise ces informations pour trouver le texte en clair.

Les cryptanalystes utilisent la redondance naturelle du langage pour réduire le nombre de textes clairs possibles. Plus le langage est redondant, plus la cryptanalyse est facile. C'est pourquoi de nombreuses implémentations de chiffrement spécifiques sont utilisées avant les compresseurs de chiffrement pour réduire la taille du texte. La compression réduit la redondance du message et le travail nécessaire pour chiffrer et déchiffrer.

L'entropie d'un cryptosystème est une mesure de la taille de l'ensemble de clés. Il est approximé par le logarithme en base 2 du nombre de clés :

$$H(K) = log_2(nombre\ de\ clé) \quad (2\text{-}4)$$

Un cryptosystème avec une clé de 64 bits a une entropie de 64 ; un cryptosystème avec une clé de 56 bits a une entropie de 56. En général, plus l'entropie est élevée, plus il est difficile de casser un système cryptographique.

2.2.4 Distance d'unicité

Pour un message de longueur n, le nombre de clés distinctes qui traduisent le message chiffré en texte clair compréhensible de la même langue est donné par cette équation :

$$2^{H(K)-nD} - 1 \quad (2\text{-}5)$$

Shannon définit la distance d'unicité U, également connue sous le nom de point d'unicité, comme une approximation de la quantité de texte chiffré, par exemple, la somme des informations réelles (entropie) dans le texte en clair correspondant et l'entropie de la clé de

chiffrement est égale à l'utilisation des chiffres du texte chiffré. Il a ensuite montré que les textes chiffrés plus longs que cette valeur peut être raisonnablement déterminés comme n'ayant qu'un seul déchiffrement raisonnable. Les textes chiffrés qui sont beaucoup plus courts que cela peut avoir plusieurs déchiffrements également valides, augmentant ainsi la sécurité car il est difficile pour un attaquant de choisir le bon déchiffrement. Pour la plupart des cryptosystèmes clés, la distance d'unicité est définie comme l'entropie du cryptosystème divisée par la redondance du langage :

$$U = H(K)/D \qquad (2\text{-}6)$$

Comme pour toute mesure d'informations statistiques ou théoriques, les distances uniques ne font pas de prédictions déterministes, mais donnent des résultats probabilistes. La distance d'unicité représente le plus petit texte chiffré pour lequel il ne peut y avoir qu'un seul texte clair raisonnablement correspondant lors de l'exécution d'une attaque exhaustive. En général, plus la distance d'unicité est grande, meilleur est le cryptosystème. Pour DES, avec une clé de 56 bits et un message anglais codé en ASCII, la distance d'unicité est d'environ 8,2 caractères ASCII, soit 66 bits. Le tableau suivant donne la distance d'unicité pour différentes longueurs de clé.

Tableau 2-1 : *Distances d'unicité de texte ASCII chiffré avec différentes longueurs de clé*

Longueur de la clef en bits	**Distance d'unicité en nombre de lettres**
40	5.9
56	8.2
64	9.4
80	11.8
128	18.8
256	37.6

La distance d'unicité n'est pas une mesure de la quantité de texte chiffré nécessaire pour la cryptanalyse, mais de la quantité de texte chiffré nécessaire pour une seule solution cryptanalytique raisonnable. Même s'il est théoriquement possible de casser un système cryptographique avec une petite quantité de texte chiffré, il est possible de casser un système

cryptographique par calcul. La distance d'unicité est inversement proportionnelle à la redondance. Lorsque la redondance approche de zéro, même un chiffrement trivial peut être inviolable par une attaque à texte chiffré seulement.

Shannon définit un cryptosystème avec une distance d'unicité infinie comme un système de confidentialité idéal. Notez qu'un cryptosystème idéal n'est pas nécessairement un cryptosystème parfait, bien qu'un cryptosystème parfait doive être un cryptosystème idéal. Si un cryptosystème offre une confidentialité idéale, même une cryptanalyse réussie laissera une incertitude quant à savoir si le texte décodé est le texte correct.

Bien que ces concepts aient une grande valeur théorique, la cryptanalyse pratique s'appuie rarement sur eux. Trop petit, et la distance d'unicité est toujours aussi peu sûre qu'elle est grande. Peu d'algorithmes pratiques sont complètement à l'abri de l'analyse ; diverses fonctionnalités peuvent être utilisées comme levier pour casser certains messages cryptés. Cependant, des considérations théoriques de l'information sont parfois utiles, telles que la détermination de l'intervalle de changement de clé pour un algorithme donné. Les cryptanalystes utilisent divers tests statistiques basés sur la théorie de l'information pour orienter leur analyse dans la direction la plus efficace. Malheureusement, la plupart des publications sur la théorie de l'information appliquée à la cryptanalyse restent classifiées, comme les actes d'atelier d'Alan Turing en 1940.

2.2.5 Confusion et diffusion

Selon Shannon [121], les deux techniques de base pour supprimer les redondances dans le texte en clair sont la confusion et la diffusion.

La confusion brouille la relation entre le texte en clair et le texte chiffré. Il évite l'analyse du texte chiffré en recherchant la redondance et les modèles statistiques. Le moyen le plus simple d'y parvenir est de remplacer. Un chiffrement de substitution simple, tel que le système de décalage de Jules César, est un système dans lequel chaque lettre identique dans le texte en clair est remplacée par la même lettre dans le texte chiffré. Les chiffrements de substitution modernes sont plus complexes : un long bloc de texte en clair est remplacé par un autre bloc de texte chiffré, et le mécanisme de remplacement change avec chaque bit du texte en clair. Ce type de remplacement n'est pas nécessairement suffisant ; le système allemand ENIGMA était un algorithme de substitution complexe qui a été cassé pendant la Seconde Guerre mondiale.

La diffusion distribue la redondance du texte en clair en la répartissant sur l'ensemble du texte chiffré. Les cryptanalystes à la recherche de ces redondances auront plus de mal à les

trouver. La façon la plus simple d'induire la diffusion est de transposer (également appelée permutation). Un chiffrement de transposition simple, tel qu'une transposition en colonne, réorganise simplement les lettres du texte en clair. Les chiffrements modernes implémentent ce type de permutation, mais ils emploient d'autres formes de diffusion qui diffusent des parties d'un message à travers le message.

Les numéros consécutifs sont basés sur la confusion uniquement. Les algorithmes de bloc utilisent la confusion et la diffusion. En règle générale, la diffusion seule est facile à casser (bien que la double transposition soit meilleure que de nombreuses autres méthodes de bricolage avec stylo et papier).

2.3 Théorie de la complexité

La théorie de la complexité fournit un moyen d'analyser la complexité de calcul de différents algorithmes et techniques cryptographiques. Il compare les algorithmes et les techniques de chiffrement pour déterminer leur niveau de sécurité. La théorie de l'information nous dit que tous les algorithmes cryptographiques (à l'exception des masques à usage unique) peuvent être piratés. La théorie de la complexité nous dit s'ils peuvent être brisés avant la fin du monde.

2.3.1 Complexité des algorithmes

La complexité d'un algorithme dépend de la puissance de calcul nécessaire à son exécution. La complexité de calcul d'un algorithme est mesurée par deux paramètres : T (complexité temporelle) et S (complexité spatiale, ou mémoire requise). En règle générale, T et S sont désignés par n, où n est la taille de l'entrée.

Habituellement, la complexité de calcul d'un algorithme est exprimée en notation dite "grand O" : un ordre de grandeur de la complexité de calcul, c'est-à-dire le terme de fonction de complexité à croissance la plus rapide avec n ; tous les termes constants et d'ordre inférieur sont ignorés. Par exemple, si la complexité temporelle d'un algorithme donné est $4n^2 + 7n + 12$, alors la complexité de calcul sera de l'ordre de n^2, qui peut également être exprimée par $O(n^2)$.

De cette façon, la mesure de la complexité temporelle est indépendante du système. Vous n'avez pas besoin de connaître la synchronisation exacte des différentes instructions ou le

nombre de bits utilisés pour représenter différentes variables ou même la vitesse du processeur. Un ordinateur peut être une fois et demie plus rapide qu'un autre ordinateur, ou un autre ordinateur peut avoir un chemin de données deux fois plus large, et l'ordre de grandeur de la complexité algorithmique reste le même. Ce n'est pas de la triche ; lorsque vous avez affaire à des algorithmes aussi complexes que ceux décrits dans cette thèse, les autres informations sont généralement négligeables par rapport à l'ordre de grandeur de la complexité.

Cette notation vous permet de voir comment les exigences de temps et d'espace varient avec la taille de l'entrée. Par exemple, si $T = O(n)$, doubler la taille de l'entrée double le temps de calcul de l'algorithme. Si $T = O(n^2)$, ajouter 1 bit à la taille de l'entrée double le temps de calcul de l'algorithme (dans un facteur constant).

Généralement, les algorithmes sont classés selon leur complexité temporelle ou spatiale. Un algorithme est constant si sa complexité est indépendante de n: $O(1)$. Un algorithme est linéaire $O(n)$ si sa complexité augmente linéairement avec n. Les algorithmes peuvent également être quadratiques, cubiques, etc. Tous ces algorithmes sont polynomiaux : leur complexité est $O(n^t)$, où t est une constante. Une classe d'algorithmes à complexité temporelle polynomiale est appelée algorithmes temporels polynomiaux.

Les algorithmes de complexité $O(t^{f(n)})$ où t est une constante et $f(n)$ est une fonction polynomiale de n est appelé exponentiel. La complexité est $O(t^{f(n)})$, où t est une constante, et ceux dont $f(n)$ est plus que constant mais moins que linéaire sont appelés superpolynomiaux.

Idéalement, les cryptographes veulent pouvoir dire que tout algorithme qui déchiffre leur mot de passe devrait être exponentiel. En pratique, compte tenu de l'état de l'art en théorie de la complexité, l'affirmation la plus forte qui puisse être faite est que "tout algorithme de cassage connu pour ce nombre a une complexité temporelle superpolynomiale". Autrement dit, pour les nombres réellement utilisés, les algorithmes de déchiffrement que nous connaissons ont une complexité superpolynomiale dans le temps, mais il n'y a pas encore de preuve que nous ne pourrons jamais découvrir d'algorithmes de déchiffrement polynomial. Les progrès de la théorie de la complexité pourraient un jour concevoir des chiffrements qui excluraient mathématiquement de manière déterministe l'existence d'algorithmes de rupture de polynômes temporels.

Lorsque n croît, la complexité temporelle de l'algorithme peut avoir un impact énorme sur l'applicabilité de l'algorithme. Le tableau ci-dessus montre les temps d'exécution des différentes

classes d'algorithmes pour lesquels n vaut un million. Le tableau masque les constantes et montre pourquoi il est raisonnable de les ignorer.

Tableau 2-2 : *Temps de calcul des différentes classes d'algorithmes.*

Classe	**Complexité**	**Nombre d'opérations pour $n = 10^6$**	**Temps pour 10^6 opérations par seconde**
Constants	$O(1)$	1	$1\ \mu s$
Linéaires	$O(n)$	10^6	$1\ s$
Quadratiques	$O(n^2)$	10^{12}	$11.6\ j$
Cubiques	$O(n^3)$	10^{18}	$32000\ années$
Exponentiels	$O(2^n)$	10^{301030}	10^{301006} fois l'âge de l'univers

En supposant que l'unité de temps de votre ordinateur est la microseconde (μs), l'ordinateur peut effectuer une arithmétique constante en une microseconde, une arithmétique linéaire en une seconde et une arithmétique quadratique en 11,6 jours. Il faudrait 32 000 ans pour exécuter l'algorithme trois fois ; ce n'est pas très pratique, mais dans ce cas, l'univers existe toujours et un ordinateur pourrait éventuellement fournir une solution. Peu importe ce que vous extrapolez sur la puissance de calcul, le traitement parallèle et l'aide d'extraterrestres super intelligents, exécuter des algorithmes exponentiels est futile.

Prenons, par exemple, une attaque exhaustive contre un cryptosystème. La complexité temporelle de cette attaque est proportionnelle au nombre de clés possibles, qui est une fonction exponentielle de la longueur de la clé. Si n est la longueur de la clé, la complexité de l'attaque exhaustive est $O(2^n)$. La complexité d'une attaque exhaustive sur un cryptosystème DES avec une clé de 56 bits est de 2^{56} ; pour une clé de 112 bits, la complexité est de 2^{112}. Le premier résultat est à la limite de ce qui est réalisable, le second est complètement hors de portée.

2.3.2 Complexité des problèmes

La théorie de la complexité classe également la complexité inhérente d'un problème, pas seulement la complexité de l'algorithme particulier utilisé pour le résoudre. La théorie détermine la quantité minimale de temps et d'espace nécessaire pour résoudre les cas les plus difficiles d'un problème sur un ordinateur théorique appelé machine de Turing. C'est une

machine à états finis avec une mémoire de lecture et d'écriture sous forme de bandes infinies. Il s'avère que les machines de Turing sont le véritable modèle de calcul.

Les problèmes qui peuvent être résolus avec un algorithme en temps polynomial sont dits solubles car ils peuvent généralement être résolus en un temps raisonnable avec des entrées de taille raisonnable. Les problèmes qui ne peuvent pas être résolus en temps polynomial sont dits insolubles car le calcul de leurs solutions devient rapidement impossible. Les problèmes insolubles sont parfois appelés difficultés simples. Ils le sont. Les problèmes qui peuvent être résolus par des algorithmes exponentiels ou superpolynomiaux sont informatiquement insolubles, même pour des valeurs relativement petites de n.

Et pire. Alan Turing a prouvé que certains problèmes sont indécidables. Il est impossible de concevoir un algorithme pour les résoudre, certainement pas polynomial.

Les problèmes peuvent être divisés en catégories de complexité en fonction de la complexité de la solution. La figure ci-dessus montre les catégories de complexité les plus importantes et leurs connexions supposées (malheureusement, il n'y a pas beaucoup de preuves mathématiques).

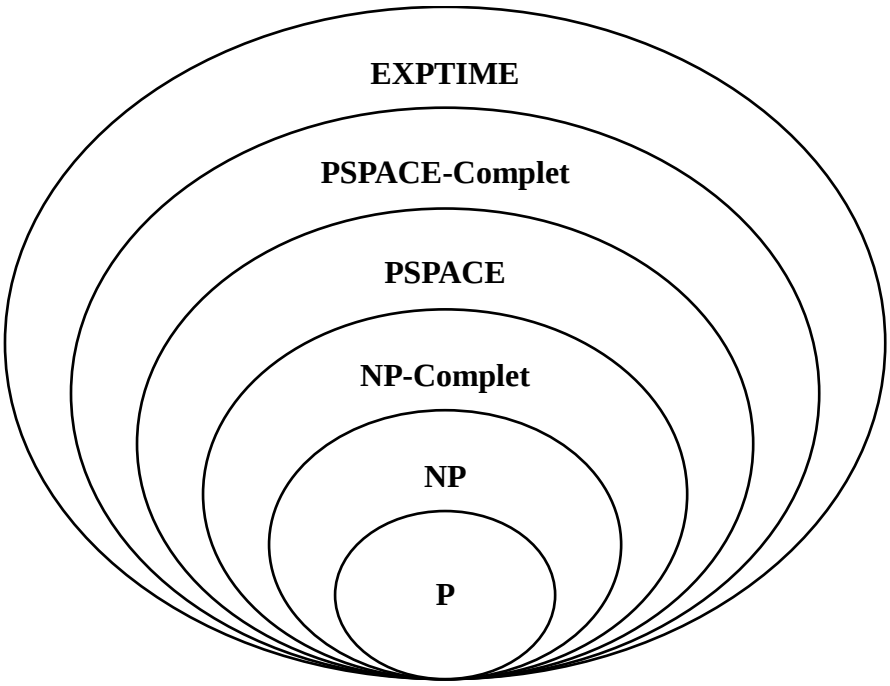


Figure 2-1 : *Classes de complexité.*

Tout en bas, la classe P contient tous les problèmes qui peuvent être résolus en temps polynomial. La classe NP contient tous les problèmes qui peuvent être résolus en temps polynomial sur une machine de Turing non déterministe. C'est une variante de la machine de Turing ordinaire qui devine la solution. La machine devine une solution à un problème - soit par hasard en faisant une bonne supposition, soit en essayant toutes les possibilités en parallèle - et vérifie sa supposition en temps polynomial.

L'intérêt cryptographique de la classe NP est que de nombreuses classes de chiffrements traditionnels peuvent être déchiffrées en temps polynomial non déterministe. Étant donné un texte chiffré C, le cryptanalyste n'a qu'à deviner le texte clair X et la clé k, et exécuter l'algorithme de chiffrement sur les entrées X et k en temps polynomial, en vérifiant finalement si le résultat est égal à C Ceci est théoriquement important car il donne Une limite supérieure sur la complexité cryptanalytique de ces classes de chiffrements. Bien sûr, en pratique, ce que recherchent les cryptanalystes, ce sont des algorithmes déterministes en temps polynomial. De plus, cet argument ne s'applique pas à tous les types de chiffrements, en particulier les masques jetables (One Time Pad) - il existe de nombreuses paires X pour tout C, k donné C lors de l'utilisation d'un algorithme de chiffrement, mais la plupart de ces X n'ont pas de sens.

La classe NP inclut la classe P car tout problème qui peut être résolu en temps polynomial sur une machine de Turing déterministe peut également être résolu en temps polynomial sur une machine de Turing non déterministe car l'étape de devinette peut être simplement éliminée.

Si tous les problèmes NP sont solubles en temps polynomial sur une machine déterministe, alors $P = NP$. Bien qu'il semble évident que certains problèmes NP se montrent nettement plus difficiles que d'autres (une attaque exhaustive contre un chiffrement vs le chiffrement d'un bloc quelconque de texte en clair). Il n'a jamais été prouvé que $P = NP\ ou\ P \neq NP$. Pourtant, tous ceux qui travaillent sur la théorie de la complexité soupçonnent qu'ils sont inégaux.

Encore plus étrange, un problème particulier dans NP s'avère aussi difficile que n'importe quel autre problème de la même classe. COOK a prouvé que le problème SAT (« Étant donné une formule propositionnelle booléenne, y a-t-il un moyen d'attribuer des valeurs de vérité à des variables telles que la formule soit vraie ? ») est $NP - Complet$. Cela signifie que si SAT peut être résolu en temps polynomial, alors$P = NP$. Inversement, s'il peut être démontré qu'il n'y a pas d'algorithmes déterministes en temps polynomial pour aucun problème dans NP, cela

prouverait que SAT ne reconnaît pas non plus les algorithmes déterministes en temps polynomial. Il n'y a donc pas de problème plus difficile dans NP que SAT.

Plus haut dans la hiérarchie de complexité se trouve$PSPACE$. Les problèmes dans $PSPACE$ peuvent être résolus dans l'espace polynomial, mais pas nécessairement en temps polynomial. $PSPACE$ comprend NP, mais il y a des problèmes dans $PSPACE$ qui sont considérés comme plus difficiles que NP. Bien sûr, cela n'a pas été confirmé non plus. Il existe une classe de problèmes, appelés problèmes $PSPACE - Complets$, avec les propriétés suivantes : si l'un d'entre eux est dans NP, alors $PSPACE = NP$, et si l'un d'entre eux est dans P, alors $PSPACE = P$.

Enfin, il existe une classe de problèmes appelés $EXPTIME$. Ces problèmes peuvent être résolus en temps exponentiel. Les problèmes $EXPTIME - Complets$ sont ceux qui ne peuvent pas être résolus en temps déterministe polynomial. Il a été montré que P n'est pas égal à $EXPTIME$.

2.3.3 Quelques problèmes NP-Complets

Problème de l'agent de voyage. Un voyageur de commerce doit visiter n villes différentes, il n'a qu'un seul plein d'essence (il ne peut donc parcourir que la distance maximale). Existe-t-il un itinéraire qui lui permettrait de se rendre dans chaque ville une fois et une seule avec le plein d'essence ? (C'est une généralisation du problème du Circuit Hamiltonien).

Le problème du groupement par 3. Considérez n hommes et n femmes dans une pièce, et n clergé (prêtres, rabbins, etc.) ; ou une liste de groupes acceptables comprenant des hommes, des femmes et des membres du clergé désireux d'officier. Compte tenu de cette liste de triplés possibles, est-il possible d'organiser n groupements de manière à ce que chacun soit marié à quelqu'un, soit officié ?

Le Problème $3 - SAT$. Soit une liste de n propositions logiques, chacune comportant trois littéraux :

- $(x \wedge y) \Rightarrow z$
- $(x \wedge w) \vee (\neg z)$
- $((\neg u \wedge \neg x) \vee (z \wedge (u \wedge \neg x))) \Rightarrow ((\neg z \wedge u) \vee x)$
- Etc.

Existe-t-il une assignation de valeurs de vérité pour tous les littéraux qui satisfasse toutes les propositions ? (C'est un cas particulier du problème SAT).

2.4 Théorie des nombres

2.4.1 Congruences

La notation $a \equiv b \ (mod\ n)$ signifie qu'il existe un entier k, tel que $a = b + kn$. Si a et b sont positifs et b est inférieur à n, alors b peut être considéré comme le reste de la division de a par n. Nous avons également la propriété que a et b obtiennent le même reste lorsqu'ils sont divisés par n. Parfois b est appelé le reste modulo n. On dit aussi que a et b sont congruents modulo n (la notation $\equiv$ signifie congruence). Ce sont juste des façons différentes de dire la même chose.

L'ensemble des entiers de 0 à $n - 1$ forme ce qu'on appelle l'ensemble de tous les restes modulo n. Cela signifie que pour tout entier a, son reste modulo n est un nombre compris entre 0 et $n - 1$, y compris.

L'opération $a\ mod\ n$ représente le résidu de a. Le résidu est un entier compris entre 0 à $n - 1$. Cette opération est parfois appelée réduction modulo n. Par exemple, $5\ mod\ 2 = 3$.

Notez que la définition de « mod » peut différer de la définition utilisée dans certains langages de programmation. Par exemple, l'opérateur modulo du langage Pascal donne parfois des résultats négatifs. Dans ce cas, le résultat est compris entre $-n - 1$ et $n - 1$. En C, l'opérateur % fournit le reste de la division du premier opérande par le second ; ce résultat peut être négatif si l'un des opérandes est négatif.

Comme l'arithmétique classique, l'arithmétique modulo est commutative, associative et distributive. De plus, lors du calcul, vous obtenez la même valeur finale que vous appliquiez la réduction modulo n à chaque étape intermédiaire ou uniquement au résultat final. Les formules suivantes expriment formellement certaines de ces propriétés :

$$(a + b)\ mod\ n = \big((a\ mod\ n) + (b\ mod\ n)\big)\ mod\ n \qquad (2\text{-}7)$$

$$(a - b)\ mod\ n = \big((a\ mod\ n) - (b\ mod\ n)\big)\ mod\ n \qquad (2\text{-}8)$$

$$(a \times b)\ mod\ n = \big((a\ mod\ n) \times (b\ mod\ n)\big)\ mod\ n \quad (2\text{-}9)$$

$$(a \times (a + b))\ mod\ n = \big(((a \times b)\ mod\ n) + ((a \times c)\ mod\ n)\big)\ mod\ n \quad (2\text{-}10)$$

En cryptographie, nous utilisons abondamment l'arithmétique modulo n. Cela vous permet de limiter la taille de tous les résultats intermédiaires et des valeurs finales. Ceci est essentiel, par exemple, pour le calcul de logarithmes discrets ou de racines carrées, qui sont des opérations très coûteuses en termes de ressources de calcul (temps et espace mémoire). Pour un modulo n qui peut être représenté en k bits, le résultat de toute addition, soustraction ou multiplication peut être représenté en jusqu'à $2k$ bits. Comme on peut appliquer la réduction modulo n à toutes les étapes intermédiaires du calcul, il ne faut jamais opérer sur des nombres représentés par plus de $2k$ bits.

Ainsi, par exemple, les exposants peuvent être calculés en arithmétique modulo n sans produire de résultats intermédiaires disproportionnés. Élever un nombre à une puissance entière modulo n, $a^x\ mod\ n$, par une série de multiplications et de divisions, mais il existe certaines techniques Faites-le efficacement. L'une de ces techniques consiste à minimiser le nombre de multiplications modulo, l'autre consiste à optimiser le calcul de multiplication modulo n. Puisque l'arithmétique modulo n est distribuée, il est plus efficace d'entrelacer la multiplication avec la réduction modulo n à chaque étape intermédiaire. Pour l'exemple ci-dessous, cela ne semble pas être d'une grande utilité, mais pour les nombres à 200 chiffres, c'est essentiel.

Par exemple, si vous voulez calculer $a^8\ mod\ n$, n'utilisez pas la méthode simple consistant à faire 7 multiplications suivies d'une réduction modulo n finale coûteuse :

$$(a \times a \times a \times a \times a \times a \times a \times a)\ mod\ n$$

Il est plus avantageux d'effectuer 3 petites multiplications et 3 petites réductions modulo :

$$((a^2\ mod\ n)^2\ mod\ n)^2\ mod\ n$$

Le même principe est appliqué sur une puissance 16^{eme}, ça donne :

$$a^{16}\ mod\ n = (((a^2\ mod\ n)^2\ mod\ n)^2\ mod\ n)^2\ mod\ n \quad (2\text{-}11)$$

Calculer $a^x\ mod\ n$ lorsque x n'est pas une puissance de 2 n'est que légèrement plus difficile. Illustrons cela avec $x = 25$. La première étape consiste à représenter x en notation

binaire (c'est-à-dire comme une somme de puissances de 2). La représentation binaire de 25 est 11001, donc $25 = 2^4 + 2^3 + 2^0$. Ensuite, certaines transformations de base donnent :

$$\begin{aligned} a^{25}\ mod\ n &= (a \times a^8 \times a^{16})\ mod\ n \\ &= (a \times (((a^2)^2)^2) \times ((((a^2)^2)^2)^2))\ mod\ n \\ &= ((((a^2 \times a)^2)^2)^2 \times a)\ mod\ n \end{aligned} \quad (2\text{-}12)$$

Par un arrangement judicieux de la mémorisation des résultats intermédiaires, seules 6 multiplications peuvent être effectuées :

$$(((((((a^2\ mod\ n) \times a)\ mod\ n)^2\ mod\ n)^2\ mod\ n)^2\ mod\ n) \times a)\ mod\ n$$

Cette technique de calcul est appelée sommation en chaîne. Il utilise une séquence d'addition simple et évidente basée sur la représentation binaire.

Si k est le nombre de bits dans l'exposant x, la technique précédente nécessite en moyenne $1.5 \times k$ opérations. Trouver la séquence d'opérations la plus courte possible est un problème difficile (il a été démontré que de telles séquences contiennent au moins $k - 1$ opérations), mais réduire le nombre d'opérations à $1.1 \times k$ ou même mieux k est important.

2.4.2 Nombres premiers

Un nombre premier est un entier strictement supérieur à 1 dont les seuls diviseurs sont 1 et lui-même : aucun autre nombre ne le divise exactement. Le nombre 2 est premier, de même pour : 97, 3511, 2159638561. Les nombres non premiers sont appelés nombres composés. Il existe une infinité de nombres premiers. En cryptographie on utilise souvent de très grands nombres premiers (512 bits ou plus).

2.4.3 Plus grand commun diviseur

Deux nombres sont premiers entre eux lorsqu'ils n'ont pas d'autres diviseurs en commun que 1. En d'autres termes, si le plus grand diviseur commun ($PGCD$en abrégé) de a et n est 1, alors a et n sont dits premiers entre eux :

$$pgcd(a, n) = 1 \quad (2\text{-}13)$$

Les nombres 15 et 28 et 13 et 500 sont premiers entre eux, tandis que 15 et 27 ne le sont pas (3 est un facteur commun supérieur à 1). Un nombre premier est le nombre premier de tous les autres nombres à l'exception des multiples de lui-même.

La méthode la plus simple pour calculer le plus grand commun diviseur de deux nombres est donnée par l'algorithme d'Euclide. Cet algorithme est décrit dans les « Eléments d'Euclide » écrits 300 ans avant J-C Euclide ne l'a pas inventé. Les historiens pensent que l'algorithme en question pourrait être plus vieux de 200 ans. C'est un des plus vieux algorithmes non triviaux connus et nous nous en servons encore de nos jours.

2.4.4 Inverse modulo n

Rappelons les bases de l'inverse d'un nombre, l'inverse de 5 est 1/5 parce que $5 \times \frac{1}{5} = 1$.

Dans le domaine de l'arithmétique modulo n, c'est plus compliqué.

$$4 \times x \equiv 1 \, mod \, 7 \tag{2-14}$$

Cette équation revient à chercher le couple (x, k) tels que

$$4x = 7k + 1 \tag{2-15}$$

Avec $(x, k) \in \mathbb{Z}^2$

Le problème général est de trouver un x tel que

$$(a \times x) \, mod \, n = 1 \tag{2-16}$$

Ceci est également écrit

$$a^{-1} \equiv x \, (mod \, n) \tag{2-17}$$

Le calcul de l'inverse modulo n est un problème difficile à résoudre. Parfois il y a une solution, parfois il n'y en a pas. Par exemple, l'inverse de $5 \, mod \, 14$ est 3 car $5 \times 3 = 15 = 1 \, (mod \, 14)$. Par contre, 2 n'a pas d'inverse modulo 14.

En général, $a^{-1} \equiv x \, (mod \, n)$ admet une unique solution x si a et n sont premiers entre eux. Si a et n ne sont pas premiers entre eux, $a^{-1} \equiv x \, (mod \, n)$ n'a pas de solution. Si n est un nombre premier, alors chaque nombre entre 1 et $n - 1$ est un nombre premier par rapport à n,

et chacun a exactement un inverse modulo a dans cet intervalle. Mais comment calcule-t-on l'inverse de a modulo n? Il existe plusieurs façons. L'algorithme d'Euclide peut également calculer l'inverse d'un nombre modulo n.

2.4.5 Petit théorème de Fermat

Si m est premier et a n'est pas un multiple de m, le petit théorème de Fermat énonce que

$$a^{m-1} \equiv 1 \ (mod\ m) \tag{2-18}$$

2.4.6 Fonction d'Euler

Il existe une autre façon de calculer l'inverse du module, mais elle n'est pas toujours applicable. L'ensemble restreint des résidus modulo n est l'ensemble des résidus premiers par rapport à n. Par exemple, l'ensemble restreint des résidus modulo 12 est $\{1,5,7,11\}$. Si n est premier, alors l'ensemble restreint des résidus est l'ensemble de tous les nombres entre 1 et $n-1$. Le nombre 0 ne fait jamais partie de l'ensemble restreint des résidus.

La fonction d'Euler de n (parfois appelée indicateur d'Euler de n) est le nombre d'éléments (« le cardinal ») de l'ensemble restreint des résidus modulo n. Cette fonction s'exprime par : $\phi(n)$. En d'autres termes, $\phi(n)$ est le nombre d'entiers positifs inférieurs à n et premiers par rapport à n.

Si n est premier, alors $\phi(n) = n - 1$. Si $n = p \times q$ et p et q sont premiers, alors $\phi(n) = (p-1)(q-1)$. Ces chiffres reviennent fréquemment lors de discussions sur les algorithmes à clé publique.

D'après le petit théorème de Fermât généralisé par Euler, si $pgcd(a,n) = 1$, alors :

$$a^{\phi(n)}\ mod\ n = 1 \tag{2-19}$$

Maintenant, il est facile de calculer $a^{-1}\ mod\ n$:

$$x = a^{\phi(n)-1}\ mod\ n \tag{2-20}$$

Par exemple, quel est l'inverse de $5\ mod\ 7$? Puisque 7 est premier, $\phi(7) = 7 - 1 = 6$. Donc, l'inverse de $5\ mod\ 7$ est $5^{6-1}\ mod\ 7 = 5^5\ mod\ 7 = 3$.

Ces deux méthodes d'inversion peuvent être étendues pour calculer x dans l'équation (si $pgcd(a,n) = 1$) :

$$(a \times x)\ mod\ n = b \quad (2\text{-}21)$$

Par la généralisation d'Euler, on trouve :

$$x = \left(b \times a^{\phi(n)-1}\right) mod\ n \quad (2\text{-}22)$$

En utilisant l'algorithme d'Euclide, on trouve :

$$x = (b \times (a^{-1}\ mod\ n))\ mod\ n \quad (2\text{-}23)$$

En général, l'algorithme d'Euclide est un moyen plus rapide pour calculer les inverses modulo n, plus particulièrement pour des nombres d'environ 500 bits. Si $pgcd(a,n) \neq 1$, rien n'est perdu. Dans le cas général, $(a \times x)\ mod\ n = b$ peut avoir plusieurs solutions ou aucune solution du tout.

2.4.7 Théorème du reste chinois

Si vous connaissez la décomposition en facteurs premiers de n, alors vous pouvez utiliser le théorème du reste chinois pour résoudre un système d'équations particulier. La version de base du théorème a été découverte au premier siècle de notre ère par le mathématicien chinois Sun Tse.

En général, si la décomposition en facteurs premiers de n est $n = p_1 \times p_2 \times \ldots \times p_t,$, alors le système d'équations :

$$(x\ mod\ p_i) = a_i, pour\ i \in \{1, 2, \ldots, t\} \quad (2\text{-}24)$$

Admet une solution unique x, telle que x soit inférieur à n. Certains nombres premiers peuvent apparaître plusieurs fois dans le produit. Par exemple, p_1 peut être égal à p_2. Ainsi, pour a et b quelconques tels que $a < p$ et $b < q$ (p et q premiers), il existe un seul x tel que x soit inférieur à $p \times q$ et tel que :

$$x \equiv a\ (mod\ p)\ \ et\ \ x \equiv b\ (mod\ q) \quad (2\text{-}25)$$

Pour calculer x, on utilise l'algorithme d'Euclide pour calculer u tel que :

$$u \times q \equiv 1 \ (mod \ p) \tag{2-26}$$

Et après on calcule

$$x = \Big(\big((a-b) \times u\big) \ mod \ p\Big) \times q + b \tag{2-27}$$

Un corollaire du théorème du restes chinois peut être utilisé pour résoudre un problème similaire : si p et q sont premiers et p est inférieur à q, alors il existe un unique x inférieur à $p \times q$ tel que :

$$a \equiv x \ (mod \ p), et \ b \equiv x \ (mod \ q) \tag{2-28}$$

Si $a \geq b \ mod \ p$, alors :

$$x = \Big(\big((a - (b \ mod \ p)\big) \times u\Big) \ mod \ p\Big) \times q + b \tag{2-29}$$

Sinon :

$$x = \Big(\big((a + p - (b \ mod \ p)\big) \times u\Big) \ mod \ p\Big) \times q + b$$

2.4.8 Résidus quadratiques

Si p est premier et a est inférieur à p, alors a est un résidu quadratique modulo p si :

$$x^2 \equiv a \ (mod \ p), pour \ un \ certain \ x \tag{2-31}$$

Une valeur quelconque de a ne satisfait pas nécessairement cette propriété, Par exemple, si $p = 7$, les résidus quadratiques sont 1, 2 et 4 :

$$1^2 = 1 \equiv 1 \ (mod \ 7)$$

$$2^2 = 4 \equiv 4 \ (mod \ 7)$$

$$3^2 = 9 \equiv 2 \ (mod \ 7)$$

$$4^2 = 16 \equiv 2 \ (mod \ 7)$$

$$5^2 = 25 \equiv 4 \ (mod \ 7)$$

$$6^2 = 36 \equiv 1 \ (mod \ 7)$$

Notez que chaque résidu quadratique apparaît deux fois dans la liste.

Il n'y a pas de valeurs x qui satisfont à l'une des équations suivantes :

$$x^2 \equiv 3 \ (mod\ 7)$$

$$x^2 \equiv 5 \ (mod\ 7)$$

$$x^2 \equiv 6 \ (mod\ 7)$$

Donc les nombres 3, 5 et 6 ne sont pas des résidus quadratiques modulo 7.

Il est facile de montrer qu'il existe exactement $(p-1)/2$ résidus quadratiques modulo p et $(p-1)/2$ nombres qui ne sont pas des résidus quadratiques modulo p. De même, si a est le résidu quadratique modulo p, alors a a exactement deux racines carrées : l'une entre 0 et $(p-1)/2$, et l'autre entre $(p-1)/2$ et $(p-1)$. L'une de ces racines carrées est aussi le résidu quadratique modulo p ; c'est ce qu'on appelle la racine carrée principale.

Si n est le produit de deux nombres premiers, p et q, il y a exactement $(p-l)(q-l)/4$ résidus quadratiques modulo n. Un résidu quadratique modulo n est un carré parfait modulo n. Par exemple, il y a 6 résidus quadratiques modulo 35 : 1, 4, 9, 11, 16, 29. Chacun d'entre eux a exactement 4 racines carrées.

2.4.9 Symbole de Legendre

Le symbole de Legendre, noté $L(a,p)$, est défini quand a est un entier quelconque et p est un nombre premier plus grand que 2. Il vaut 0, 1 ou -1 :

$$L(a,p) = 0, si\ a\ est\ divisible\ par\ p.$$

$$L(a,p) = 1, si\ a\ est\ un\ résidu\ quadratique\ modulo\ p. \qquad (2\text{-}32)$$

$$L(a,p) = -1, si\ \ a\ n'est\ pas\ un\ résidu\ quadratique\ modulo\ p.$$

Une manière simple de calculer $L(a,p)$ est donnée par :

$$L(a,p) = a^{\frac{p-1}{2}} \ (mod\ p) \qquad (2\text{-}33)$$

Une autre façon de le calculer est donnée par les formules récursive suivantes :

Si $a = 1$, alors :

$$L(a,p) = 1 \quad (2\text{-}34)$$

Si a est pair, alors :

$$L(a,p) = L\left(\frac{a}{2},p\right) \times (-1)^{(p^2-1)/8} \quad (2\text{-}35)$$

Si a est impair, alors :

$$L(a,p) = L(p\ mod\ a,p) \times (-1)^{(a-1)\times(p-1)/4} \quad (2\text{-}36)$$

Notez que c'est aussi un moyen valable de déterminer si a est le résidu quadratique modulo p (uniquement si p est premier, bien sûr).

2.4.10 Symbole de Jacobi

Le symbole de Jacobi, noté $J(a,n)$ est une généralisation du symbole de Legendre il est défini pour toute paire d'entiers a et n. Cette fonction est utilisée dans les testes de primalité. Le symbole de JACOBI est une fonction de l'ensemble restreint des résidus des facteurs de n. Il peut être calculé à l'aide de différentes formules. Celle-ci est basée sur les définitions suivantes :

- $J(a,n)$ est défini que pour n est impair.
- $J(0,n) = 0$.
- Si n est premier, alors $J(a,n) = 0$, si n divise a.
- Si n est premier, alors $J(a,n) = 1$, si a est un résidu quadratique modulo n.
- Si n est premier, alors $J(a,n) = -1$, si a n'est pas un résidu quadratique modulo n.
- Si n est un nombre composé, alors $J(a,n) = J(a,p_1) \times J(a,p_2) \times \ldots \times J(a,p_m)$, où p_1, p_2, …, $p_m$est la décomposition en facteurs premiers de n.

Les règles suivantes permettent de calculer le symbole de Jacobi récursivement :

- $J(1,n) = 1$.
- $J(a \times b,n) = J(a,n) \times J(b,n)$.
- $J(2,n) = 1$, si $(n^2-1)/8$ est pair, -1 sinon.
- $J(a,n) = J((a\ mod\ n),n)$.

- $J(a, p_1 \times p_2) = J(a, p_1) \times J(a, p_2)$.
- Si $pgcd(a, b) = 1$, et a et b sont tous deux impairs (La loi de réciprocité ne fonctionne pas pour les nombres pairs) :
  - $J(a, b) = +J(b, a)$, si $(a-1)(b-1)/4$ est pair.
  - $J(a, b) = -J(b, a)$, si $(a-1)(b-1)/4$ est impair.

Si nous savons que n est premier, nous n'avons qu'à calculer $a^{(n-1)/2}\ mod\ n$; dans ce cas, $J(a, n)$ équivaut au symbole de Legendre. Le symbole de Jacobi ne permet pas de déterminer si a est un résidu Si $J(a, n) = 1$ et n est un nombre composé, alors a n'est pas nécessairement un résidu quadratique modulo n. Par exemple :

$$J(7{,}143) = J(7{,}13) \times J(7{,}11) = (-1) \times (-1) = 1$$

Cependant, il n'y a pas d'entier où $x^2 \equiv 7\ (mod\ 143)$.

2.4.11 Entiers de Blum

Si p et q sont deux nombres premiers et qu'ils sont tous les deux congrus à 3 modulo 4, alors $n = p \times q$ est appelé entier de Blum. Si n est un entier de Blum, chaque résidu quadratique modulo n a exactement 4 racines carrées. L'une de ces racines carrées est aussi un carré ; c'est la racine carrée principale. Par exemple, la racine carrée principale de 139 modulo 437 est 24. Les trois autres racines carrées sont 185, 252 et 413.

2.4.12 Générateurs

Si p est premier, et g est inférieur à p, alors g est un générateur modulo p si :

$$\forall b \in \{1{,}2, \dots, p-1\}, \exists a\ tel\ que\ g^a \equiv b\ (mod\ p) \qquad (2\text{-}37)$$

On dit également de façon équivalente, que g est primitif par rapport à n.

Par exemple, si on prend $p = 11$, 2 est générateur modulo 11 :

$$2^{10} = 1024 \equiv 1\ (mod\ 11)$$

$$2^1 = 2 \equiv 2\ (mod\ 11)$$

$$2^8 = 256 \equiv 3\ (mod\ 11)$$

$$2^2 = 4 \equiv 4\ (mod\ 11)$$

$$2^4 = 16 \equiv 5\ (mod\ 11)$$

$$2^9 = 512 \equiv 6\ (mod\ 11)$$

$$2^7 = 128 \equiv 7\ (mod\ 11)$$

$$2^3 = 8 \equiv 8\ (mod\ 11)$$

$$2^6 = 64 \equiv 9\ (mod\ 11)$$

$$2^5 = 32 \equiv 10\ (mod\ 11)$$

Tout nombre de 1 à 10 peut être représenté par $2^a\ (mod\ p)$.

Pour $p = 11$, les générateurs sont $2, 6, 7\ et\ 8$. Les autres nombres ne sont pas des générateurs. Par exemple, 3 n'est pas un générateur car l'équation suivante n'a pas de solution :

$$3^a \equiv 2\ (mod\ 11) \qquad (2\text{-}38)$$

En général, trouver un générateur n'est pas un problème facile, à moins que l'on connaisse la décomposition en facteurs premiers de $p - 1$. Soit $q_1, q_2, \dots, q_n$ les facteurs premiers de $p - 1$. Pour déterminer si un nombre g est générateur modulo p, on calcule :

$$g^{(p-1)/q}\ (mod\ p) \qquad (2\text{-}39)$$

Pour toutes les valeurs de $q = q_1, q_2, \dots, q_n$.

Si cette expression égale à 1 pour un certain q_i, alors g n'est pas un générateur. Si aucun q_i ne rend cette expression égale à 1, alors g est un générateur modulo p. Par exemple, pour $p = 11$, les facteurs premiers de $p - 1 = 10$ sont 2 et 5. Pour tester si 2 est un générateur, on calcule :

$$2^{(11-1)/5}\ mod\ 11 = 4$$

$$2^{(11-1)/2}\ mod\ 11 = 10$$

Aucun des résultats n'est égal à 1, donc 2 est un générateur.

Pour tester si 3 est un générateur on calcule :

$$3^{(11-1)/5}\ mod\ 11 = 9$$

$$3^{(11-1)/5}\ mod\ 11 = 1$$

Comme le dernier résultat est égal à 1, 3 n'est pas un générateur.

Si vous avez besoin d'un générateur modulo p, choisissez simplement un nombre aléatoire entre 1 et $p-1$ et testez s'il s'agit bien d'un générateur. Il y en a beaucoup, donc vous pourriez en trouver un assez rapidement.

2.4.13 Groupe de Galois

En fait, depuis quelques paragraphes, nous avons effectué des calculs dans le groupe de Galois. Dans le cas de l'arithmétique modulo n, si n est un nombre premier ou une puissance entière d'un nombre premier, alors nous avons ce que les mathématiciens appellent un groupe fini. Dans le cas particulier de n premier, nous utiliserons la lettre p à la place. Ce type de groupe fini est si important que les mathématiciens lui donnent le nom de son promoteur : on va parler de groupe de Galois, et on l'écrira Z/p. Le mathématicien français Evariste Galois, qui a vécu au début du XIXe siècle, a beaucoup contribué au domaine de la théorie des nombres, puis est mort en duel, à l'âge de 20 ans.

Dans les groupes de Galois, l'addition, la soustraction, la multiplication et la division sont bien définies. L'addition a un élément neutre : 0 et la multiplication a un élément neutre : 1. Chaque nombre non nul a un inverse unique (cela ne tiendra pas si p n'est pas premier). Les lois commutatives, associatives et distributives s'appliquent.

En cryptographie, on utilise beaucoup l'arithmétique dans les groupes de Galois. Toute la théorie des nombres s'applique, les nombres restent de taille finie et la division n'a pas d'erreurs d'arrondi. De nombreux cryptosystèmes sont basés sur Z/p, où p est un très grand nombre premier.

Pour compliquer un peu les choses, les cryptographes utilisent aussi l'arithmétique modulo des polynômes irréductibles de degré u et à coefficients entiers modulo q (avec q est un nombre premier). Ces groupes sont notés Z/q^n. Tous les calculs sont effectués modulo un polynôme irréductible $P(X)$ de degré n.

On ne va pas aborder la théorie de ces groupes dans cette thèse, Néanmoins, on décrira quelques cryptosystèmes qui les utilisent.

Quand on parle de polynômes, le terme « premier » est souvent remplacé par le terme « irréductible ». Un polynôme est dit irréductible s'il ne peut être représenté comme le produit de deux autres polynômes (différents de 1 et de lui-même). Le polynôme x^2+1 est irréductible

sur des nombres entiers. Par contre, le polynôme $x^3 + 2x^2 + x$ est réductible : il est équivalent au produit $x(x + 1)(x + 1)$.

On appelle primitif un polynôme qui est générateur dans un corps donné ; tous ses coefficients sont premiers entre eux.

Les calculs dans $Z/2^n$ peuvent être rapidement réalisés matériellement grâce à des registres à décalage linéaire à rétroaction. Pour cette raison, les calculs dans $Z/2^n$ sont généralement plus rapides que ceux dans Z/p. Tout comme l'exponentiation peut être réalisée de manière très efficace dans $Z/2^n$, il en est de même pour les logarithmes discrets.

Pour les corps de Galois $Z/2^n$, les cryptographes aiment utiliser le trinôme $P(X) = x^n + x + 1$ comme module, car la longue série de coefficients nuls entre x^n et x rend la multiplication très efficace. Le trinôme doit être primitif ou il ne fonctionne pas. Pour n inférieur à 1000, le trinôme $x^n + x + 1$ est primitif si n est égal à :

$$1, 3, 4, 6, 9, 15, 22, 28, 30, 46, 60, 63, 127, 153, 172, 303, 471, 532, 865, 900.$$

2.5 Factorisation

Factoriser un nombre, c'est trouver ses facteurs premiers, par exemple :

$$10 = 2 \times 5$$

$$60 = 2 \times 2 \times 3 \times 5$$

$$252601 = 61 \times 41 \times 101$$

$$2^{113} - 1 = 3391 \times 23279 \times 65993 \times 1868569 \times 1066818132868207$$

Le problème de factorisation est l'un des problèmes les plus anciens de la théorie des nombres. Il est facile de factoriser un nombre mais c'est coûteux en temps. Cependant, des progrès ont été réalisés dans l'état de l'art.

Actuellement, le meilleur algorithme de factorisation est Le crible sur corps numérique (NFS pour « Number Field Sieve »), qui est la méthode connue la plus rapide pour factoriser des nombres autour de 110 chiffres. Cela ne fonctionnait pas lorsqu'il a été proposé pour la première fois, mais cela a changé grâce à une série d'améliorations au cours des dernières années. Le crible sur corps numérique est encore trop nouveau pour établir un record de

factorisation, mais il arrivera bientôt. Le NFS a été utilisé pour factoriser le neuvième nombre de FERMAT : $2^{512} + 1$

Il existe d 'autres algorithmes de factorisation, mais ils ont été remplacés par le crible sur corps numérique :

- **Le crible quadratique :** Il s'agit de l'algorithme connu le plus rapide pour les nombres de moins de 110 chiffres et il a été largement utilisé. Une version plus rapide de cet algorithme est appelée crible quadratique multipolynomial. La version la plus rapide de cet algorithme est appelée variante à deux grands facteurs premiers du crible quadratique multipolynomial.
- **Méthode des courbes elliptiques** (ECM pour « Elliptic Curve Method »). Cette méthode a été utilisée pour trouver des facteurs de 43 chiffres décimaux mais rien de plus grand.
- **Tentative de division :** C'est le plus ancien algorithme de factorisation : il consiste à tester tous les nombres premiers jusqu'à la racine carrée du nombre à factoriser.

Si n est le nombre à factoriser, la variante la plus rapide du crible quadratique nécessite un nombre d'opérations équivalent à : $e^{(1+O(1))(\ln(n))^{\frac{1}{2}}(\ln(\ln\,(n)))^{\frac{1}{2}}}$.

Le crible sur corps numérique est plus rapide, le temps heuristique asymptotique estimé est : $e^{(1.923+O(1))(\ln(n))^{\frac{1}{3}}(\ln(\ln\,(n)))^{\frac{2}{3}}}$.

En 1970, la factorisation d'un nombre « difficile » de 41 chiffres était à la une des journaux. Dix ans plus tard, factoriser des nombres difficiles de taille double prenait quelques heures de calcul sur un ordinateur CRAY.

En 1988, Carl POMERENE a conçu une machine à factoriser modulaire en utilisant des puces VLSI personnalisées. La taille du nombre que vous pouvez factoriser dépend de la taille de la machine que vous pouvez vous permettre de construire. Il ne l'a jamais construite.

En 1993, un nombre difficile à 120 chiffres a été factorisé à l'aide de la méthode du crible quadratique ; le calcul a nécessité l'équivalent de 825 millions d'instructions par seconde pendant un an et a été exécuté en temps réel en trois mois.

Aujourd'hui, la factorisation tend à utiliser les réseaux d'ordinateurs. En factorisant un nombre de 116 chiffres, Arjen LENSTRA et Mark MANASSE utilisèrent 400 mips-an, le temps d'inaction d 'une batterie d'ordinateurs tout autour du monde pendant quelques mois.

En mars 1994, un groupe de mathématiciens dirigé par LENSTRA a factorisé 129 nombres (428 bits) en utilisant une variante du crible quadratique multipolynomial. Des volontaires sur Internet ont fait le calcul : 600 personnes et 1600 machines pendant huit mois, peut-être le plus grand multiprocesseur ad hoc jamais assemblé. Les exigences de calcul sont équivalentes à 4000 à 6000 mips-an. Les machines communiquent par e-mail, envoyant leurs résultats respectifs à un bureau central, qui est responsable des dernières étapes d'analyse. Le calcul est basé sur des cribles quadratiques et des théories vielles de 5 ans ; le crible sur le corps numérique ne nécessite qu'un dixième du temps de calcul.

Pour suivre le rythme de la technologie de la factorisation, RSA Data Security a lancé un concours de factorisation nommé "RSA Factoring Challenge" en mars 1991. Il s'agit d'une liste de nombres difficiles, dont chacun est le produit de deux nombres premiers de taille comparable. Tout nombre premier est égal à 2 modulo 3. Il y a 42 numéros dans le jeu ; leur taille passe de 100 à 500 chiffres et de 10 en 10 chiffres (plus un nombre de 129 chiffres). Jusqu'à maintenant, RSA-100, RSA-110, RSA-120 jusqu'à RSA-129 (829 bits) ont été factorisé, tout cela grâce à des cribles quadratiques. Le RSA-130, RSA-140 jusqu'à RSA-250 ont été factorisé grâce au crible sur corp numérique, Le RSA-250 a été factorisé en 28 février 2020, mais jusqu'à maintenant, on n'a pas pu factoriser le RSA-260, peut-être dans les mois prochains.

C'est un domaine qui se développe rapidement. Il est difficile d'extrapoler les progrès des techniques de factorisation car personne ne peut prédire les progrès des mathématiques. Avant la découverte des cribles sur corps numérique, beaucoup pensaient que les cribles quadratiques étaient aussi rapides que toute autre méthode de factorisation. Ils avaient tort.

2.5.1 Racines carrées modulo n

Si n est le produit de deux nombres premiers, la possibilité de calculer la racine modulo n nécessite autant de puissance de calcul équivalente à celle nécessaire pour factoriser n. En d'autres termes, quelqu'un qui connaît les facteurs premiers de n peut facilement calculer la racine carrée modulo n, mais pour d'autres, le calcul s'avère tout aussi difficile que de calculer les facteurs premiers de n.

2.6 Génération de nombres premiers

Générer des nombres aléatoires puis essayer de les factoriser est une mauvaise façon de générer des nombres premiers. La bonne méthode consiste à générer des nombres aléatoires et à tester s'ils sont premiers. Il existe plusieurs tests probabilistes de primalité qui déterminent si un nombre est premier avec un certain degré de confiance. En supposant que cette "confiance" est suffisante, alors ce type de test est assez bon.

2.6.1 Solovay-Strassen

Solovay et Strassen ont développé un algorithme probabiliste de test de primalité. Cet algorithme utilise le symbole de JACOBI quand on a expliqué précédemment pour tester si p est premier :

Début

- Choisissez un nombre aléatoire a plus petit que p.
- Si le $pgcd(a,p) \neq 1$, alors p échoue au test et il est composé.
- Calculez $j = a^{(p-1)/2}\ mod\ p$.
- Calculez le symbole de JACOBI $J(a,p)$.
- Si $j \neq J(a,p)$, alors p n'est certainement pas premier.
- Si $j = J(a,p)$, alors la probabilité que p ne soit pas premier est supérieur à 50%.

Fin

Algorithme 2-1 : *Procédure d'exécution de l'algorithme de Solovay-Strassen.*

Un nombre a pour lequel le nombre p n'est définitivement pas premier est appelé un témoin. Si p est composé, il y a au moins 50% de chances qu'un nombre aléatoire a soit un témoin. Répétez ce test t fois pour un nombre a avec t valeurs aléatoires différentes. La probabilité qu'un nombre composé réussisse les t tests est de $1/2^t$.

2.6.2 Lehmann

Un autre test plus simple a été développé indépendamment par Lehmann. Voici l'algorithme pour tester si t est premier :

Début

- Choisissez un nombre aléatoire a plus petit que p.
- Calculez $a^{(p-1)/2}\ mod\ p$.
- Si $a^{(p-1)/2} \neq 1\ ou - 1\ (mod\ p)$, pour tous les i, alors p n'est pas premier.
- Si $a^{(p-1)/2} = 1\ ou - 1\ (mod\ p)$, alors la probabilité que p soit composé est de 50% au plus.

Fin

Algorithme 2-2 : *Procédure d'exécution de l'algorithme de Lehmann*

À nouveau, un nombre aléatoire a a au moins 50% de chances d'être un témoin. Répétez le test t fois. Si le calcul vaut 1 ou -1, mais pas toujours 1, alors p est probablement premier avec un taux d'erreur de $1/2^t$.

2.6.3 Rabin-Miller

L'algorithme que tout le monde utilise (il est facile) a été développé par Rabin, en partie basé sur des idées de Miller. En fait, il s'agit d'une version simplifiée de l'algorithme recommandé par la proposition pour le DSS du NIST. Choisissez d'abord un nombre aléatoire p à tester. Calculez b, où b est le nombre de fois que 2 divise $p-1$ (c'est-à-dire que 2^b est la plus grande puissance de 2 qui divise $p-1$). Puis calculez m tel que $p = 1 + 2^b m$.

Début

- Choisissez un nombre aléatoire a plus petit que p.
- Posez $j = 0$ et $z = a^m\ mod\ p$.
- Si $z = 1$ ou $z = p - 1$, alors p passe le test et peut être premier.
- Si $j > 0$ et $z = 1$, alors p n'est pas premier.
- Posez $j = j + 1$. Si $j < b$ et $z \neq p - 1$, posez $z = z^2\ mod\ p$ et retournez à la phase 4. Si $z = p - 1$, alors p passe le test et peut être premier.
- Si $j = b$ et $z \neq p - 1$, alors p n'est pas premier.

Fin

Algorithme 2-3 : *Procédure d'exécution de l'algorithme de Rabin-Miller.*

La chance qu'un nombre composé réussisse ce test diminue plus rapidement que l'algorithme précédent. Les trois quarts des valeurs possibles de a sont bien des témoins. Cela signifie qu'un nombre composé réussit le test moins d'une fois en 4^t, où t est le nombre d'itérations. En réalité, ces estimations sont très pessimistes. Pour la plupart des nombres aléatoires, environ 99,9% des valeurs possibles de a sont des témoins.

Il existe de meilleures estimations. Pour des candidats n bits (où n est supérieur à 100), la probabilité que le test soit erroné est de $1/4n2^{\sqrt{k/2}}$. Pour un candidat de 256 bits, le risque d'erreur après six tests était inférieur à $1/2^{51}$.

2.6.4 Considérations pratiques

Dans l'implémentation mondiale actuelle, la génération de nombres premiers est rapide.

Début

1. Générez un nombre aléatoire p de n bits.
2. Mettez le bit de poids fort et le bit de poids faible à 1 (le bit de poids fort garantit que p a la longueur requise et le bit de poids faible garantit que p est impair).
3. Attention à vérifier que p n'est pas divisible par de petits nombres premiers : 3, 5, 7, 11, etc. De nombreuses implémentations testent que p est divisible par tous les nombres premiers inférieurs à 256. Le plus efficace est de tester la divisibilité de tous les nombres premiers inférieurs à 2000.
4. Exécutez le test de Rabin-Miller sur un nombre aléatoire a. Si p réussit le test, générez un autre nombre aléatoire a et testez à nouveau. Choisir une petite valeur de a peut accélérer le calcul. Faites cinq tests. Si p échoue à l'un des tests, générer un autre p et réessayer.

Fin

Algorithme 2-4 : *Génération pratique des nombre premiers.*

Une autre option consiste à ne pas générer p au hasard, mais à essayer tous les nombres commençant à un point aléatoire l'un après l'autre jusqu'à ce qu'un nombre premier soit trouvé.

L'étape 3 est facultative, mais c'est une bonne idée. Tester si un p impair aléatoire est divisible par 3, 5 et 7 élimine 54% des chances avant d'atteindre l'étape 4. Tester tous les

nombres premiers inférieurs à 100 élimine 76% des nombres impairs ; tester avec tous les nombres premiers inférieurs à 256 élimine 80% des nombres impairs. En général, le rapport des nombres impairs qui ne sont pas des multiples de tout nombre premier inférieur à n est de $1.12/\ln(n)$. Plus vous prenez n grand, plus vous avez besoin de précalcule avant d'atteindre le test de Rabin-Miller.

L'implémentation de cette méthode sur Sparc II est capable de trouver des nombres premiers de 256 bits en 2,8 secondes en moyenne, des nombres premiers de 512 bits en 24,0 secondes en moyenne, des nombres premiers de 768 bits en 2 minutes en moyenne, des nombres premiers de 1024 bits, avec un temps moyen de 5,1 minutes.

2.6.5 Nombres premiers forts

Si n est le produit de deux nombres premiers p et q, il peut être désirable d'utiliser des nombres premiers forts pour p et q. Ce sont des nombres qui ont certaines propriétés qui rendent leur produit n difficile à factoriser par des méthodes spécifiques. Voici quelques propriétés parmi celles qui ont été suggérées :

- Le plus grand commun diviseur de $p-1$ et $q-1$ doit être petit.
- $p-1$ et $q-1$ doivent tous deux avoir des grands facteurs premiers, respectivement p' et q'.
- $p'-1$ et $q'-1$ doivent tous deux avoir des grands facteurs premiers.
- $(p-1)/2$ et $(q-1)/2$ doivent tous deux être premiers. (Notez que si cette condition est vérifiée, alors les deux premières le sont aussi).

2.7 Logarithmes discrets dans un corps fini

L'exponentiation modulaire est une autre fonction unidirectionnelle couramment utilisée en cryptographie. L'évaluation de l'expressions suivante est facile : $a^x \ mod \ n$.

Le problème inverse de l'exponentiation modulaire consiste à trouver le logarithme discret d'un nombre. C'est un problème difficile. Trouver un x tel que $a^x \equiv b \ (mod \ n)$.

Par exemple : si $3^x \ mod \ 17 = 15$, alors $x = 6$.

Les logarithmes discrets n'ont pas tous des solutions (rappelez-vous que les seules solutions valides sont entières). Il est facile de voir qu'il n'y a pas de solution x pour l'équation :

$$3^x \ mod\ 13 = 7$$

Il est encore plus difficile de résoudre ces problèmes pour des nombres de 1024 bits.

2.7.1 Calcul de logarithmes discrets dans un groupe fini

Il y a trois groupes principaux dont les logarithmes discrets sont intéressants pour les cryptographes :

- Le groupe multiplicatif des corp Z/p .
- Le groupe multiplicatif des corps finis de caractéristique $Z/2^n$.
- Le groupe des courbes elliptiques sur des corps fini $F\ EC(F)$.

La sécurité de nombreux algorithmes à clé publique repose sur le problème de la détermination du logarithme discret, ce problème a donc été étudié de manière intensive.

Si p, le module, est premier, alors la complexité du calcul des logarithmes discrets dans Z/p est essentiellement la même que celle de la factorisation d'un nombre n d'à peu près la même taille où n est le produit de deux nombres premiers de tailles comparables. C'est-à-dire

$$e^{(1+O(1))(\ln(n))^{\frac{1}{2}}(\ln(\ln\ (n)))^{\frac{1}{2}}}$$

Le crible sur corps numérique est plus rapide, le temps heuristique asymptotique estimé est : $e^{(1.923+O(1))(\ln(n))^{\frac{1}{3}}(\ln(\ln\ (n)))^{\frac{2}{3}}}$.

Stephen POHLIG et Martin HELLMAN ont trouvé un moyen rapide de calculer des logarithmes discrets dans Z/p si $p-1$ n'a que des petits facteurs premiers. Pour cette raison, seuls les corps tels que $p-1$ a au moins un grand facteur premier sont utilisés en cryptographie. Un autre algorithme calcule les logarithmes discrets à une vitesse comparable à la factorisation ; il a été ensuite étendu aux corps de la formes Z/p^n. Cet algorithme a été critiqué pour avoir quelques défauts théoriques.

Le calcul des logarithmes discrets est étroitement lié à la factorisation. Si vous savez calculer des logarithmes discrets, alors vous savez factoriser (l'inverse n'a jamais été trouvé). À l'heure actuelle, il existe trois méthodes pour calculer le logarithme discret dans le corps premier :

- Le crible linéaire.
- Le schéma d'entiers de Gauss.

- Le crible sur corps numérique.

La plupart des calculs préliminaires ne doivent être effectués qu'une seule fois pour chaque corps. Après cela, le logarithme donné peut être rapidement calculé. Il s'agit d'une faille de sécurité dans les systèmes basés sur ces corps. L'important est que différents utilisateurs utilisent différents corps premiers. Cependant, les utilisateurs d'une même application peuvent utiliser le même corps premier.

Récemment, le précalcule pour $Z/2^{227}$, $Z/2^{313}$ et $Z/2^{401}$ a été achevé, et $Z/2^{503}$ a fait des progrès significatifs. Ces calculs ont été effectués sur l'ordinateur massivement parallèle NCUBE-2 avec 1024 processeurs, le calcul des logarithmes discrets en $Z/2^{593}$ est toujours hors de portée.

Comme pour le logarithme discret dans un corps premier, le précalcule requis pour calculer le logarithme discret dans le domaine polynomial ne doit être effectué qu'une seule fois. ElGamal (l'algorithme qu'on a expliqué dans le chapitre précédent) donne un algorithme pour calculer le logarithme discret dans le corps Z/p^n.

2.8 Conclusion

Dans ce chapitre, nous avons présenté des nombreux outils différents des mathématiques qui sont pertinentes pour la cryptographie, y compris la théorie des nombres, l'algèbre, les probabilités et les statistiques. Sans une solide compréhension de ces outils, il est impossible de concevoir ou de comprendre des algorithmes cryptographiques. Il est également difficile d'évaluer la sécurité d'un système cryptographique, ou de trouver des faiblesses exploitables. Une partie de ces outils ont été utiliser pour développer nos propre méthodes et algorithmes, c'est ce qu'on va détailler dans le prochain chapitre.

Chapitre 3 : Méthodes et algorithmes proposés pour la cryptographie des images couleurs

3.1 Introduction

Le but de la cryptographie des images est de garantir la sécurité visuelle du contenu en clair d'une image. Le cryptage est dit « total » lorsque l'ensemble du contenu original de l'image est sécurisé après l'opération du cryptage. Dans ces conditions, aucune information relative à l'image originale ne peut être extraite de l'image cryptée. D'un autre point de vue, lorsqu'une partie des pixels est seulement sélectionnée pour être cryptée, ce type de cryptage s'appelle « sélectif ». Finalement, on dit qu'un cryptage est « partiel », lorsque seulement une zone spécifique de l'image est cryptée et que les pixels en dehors de cette zone gardent le même contenu.

3.2 Algorithmes et méthodes de la cryptographie des images couleurs

Dans le but d'améliorer et de développer les algorithmes et les méthodes de ce domaine, nous avons proposé trois contributions qui seront présentées dans ce chapitre. La première est intitulée : Un algorithme de chiffrement d'image couleur rapide basé sur la génération de nombres aléatoires de deux matrices et les opérateurs de décalage de bits. Cet algorithme est basé sur l'utilisation de deux matrices, de taille 16×16 dont ces valeurs ce sont des nombres entiers compris entre 0 et 255 et générés aléatoirement, suivis d'une rotation circulaire des bits. Le point faible de cette méthode c'est que pour crypter une image couleur, on doit forcément appliquer cette méthode de cryptage sur chaque canal de l'image (rouge, vert, et bleu), puis on fusionne les résultats des trois canaux pour obtenir une seule image couleur cryptée, ce qui rend l'algorithme un petit peu long. C'est la raison pour laquelle nous avons développé une deuxième méthode intitulée : Un nouvel algorithme de cryptage d'image couleur utilisant plusieurs fonctions chaotiques avec la méthode des plans sécants. Cette méthode est basée sur l'utilisation

des trois faces adjacentes d'un cube rempli avec des entiers uniques, qui sont générés avec des fonctions chaotiques de type 1D. Elle permet de faire l'opération de cryptage d'une image couleur en parcourant la matrice une seule fois. La troisième contribution est intitulée : Une nouvelle méthode de cryptage d'image couleur 32 bits basée sur plusieurs fonctions chaotiques 1D et le chiffrement à quatre carrés. Cette méthode est une amélioration de la première méthode en utilisant quatre matrices (carrés) au lieu de deux, chaque matrice représente un canal d'une image couleur de 32 bits (rouge, vert, bleu, et alpha).

3.2.1 Algorithme de chiffrement d'images couleurs rapide basé sur la génération de nombres aléatoires de deux matrices et les opérateurs de décalage de bits

Cette méthode se base sur l'utilisation de deux matrices de dimensions 16×16 qui vont nous servir pour faire l'opération de cryptage de l'image. Chaque matrice est remplie avec des entiers uniques compris entre 0 et 255 et qui sont générées aléatoirement à l'exception de la première case de la première matrice qui est calculée à partir des 3 canaux de l'image originale (**Figure 3-2)**.

L'organigramme ci-dessous (**Figure 3-1)** montre les différentes étapes utilisées pour l'opération de cryptage d'une image couleur.

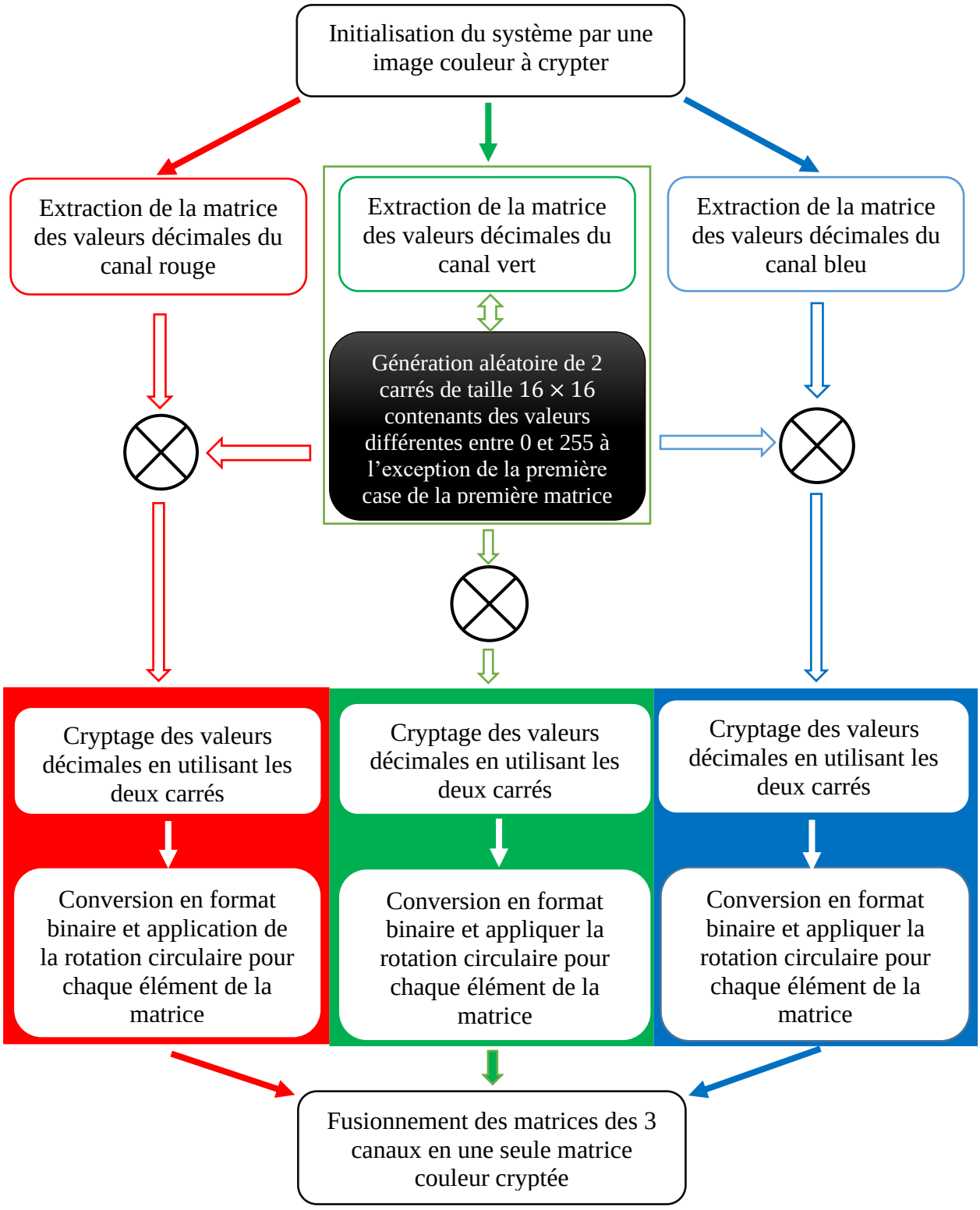


Figure 3-1 : *Organigramme de la méthode proposée*

164	250	132	167	189	48	155	98	67	187	45	166	21	102	137	242
204	168	34	160	74	27	141	226	251	205	52	134	181	10	39	38
13	90	0	246	207	153	19	196	95	245	97	59	9	175	8	147
161	143	7	216	228	213	22	229	41	188	223	218	25	99	128	240
88	198	55	28	113	194	210	192	254	30	80	138	174	248	16	235
215	117	125	191	154	197	146	4	60	91	114	159	201	96	3	220
185	150	241	209	1	170	173	208	75	24	139	255	156	32	202	57
186	214	15	123	212	162	227	6	244	222	206	93	112	109	176	61
238	44	165	121	58	172	211	183	221	82	20	234	76	108	89	130
237	145	104	69	133	46	105	43	182	180	37	177	136	110	33	135
12	119	47	120	65	118	230	149	152	71	35	5	190	171	92	62
184	232	54	106	63	68	224	14	203	115	87	11	225	42	17	122
40	101	79	126	157	86	36	247	73	100	151	179	142	124	163	26
148	77	64	85	140	169	193	231	158	253	23	199	195	29	107	66
127	51	131	200	78	103	18	84	56	83	50	219	252	243	53	178
239	111	94	144	70	217	249	72	49	129	233	31	116	236	2	81

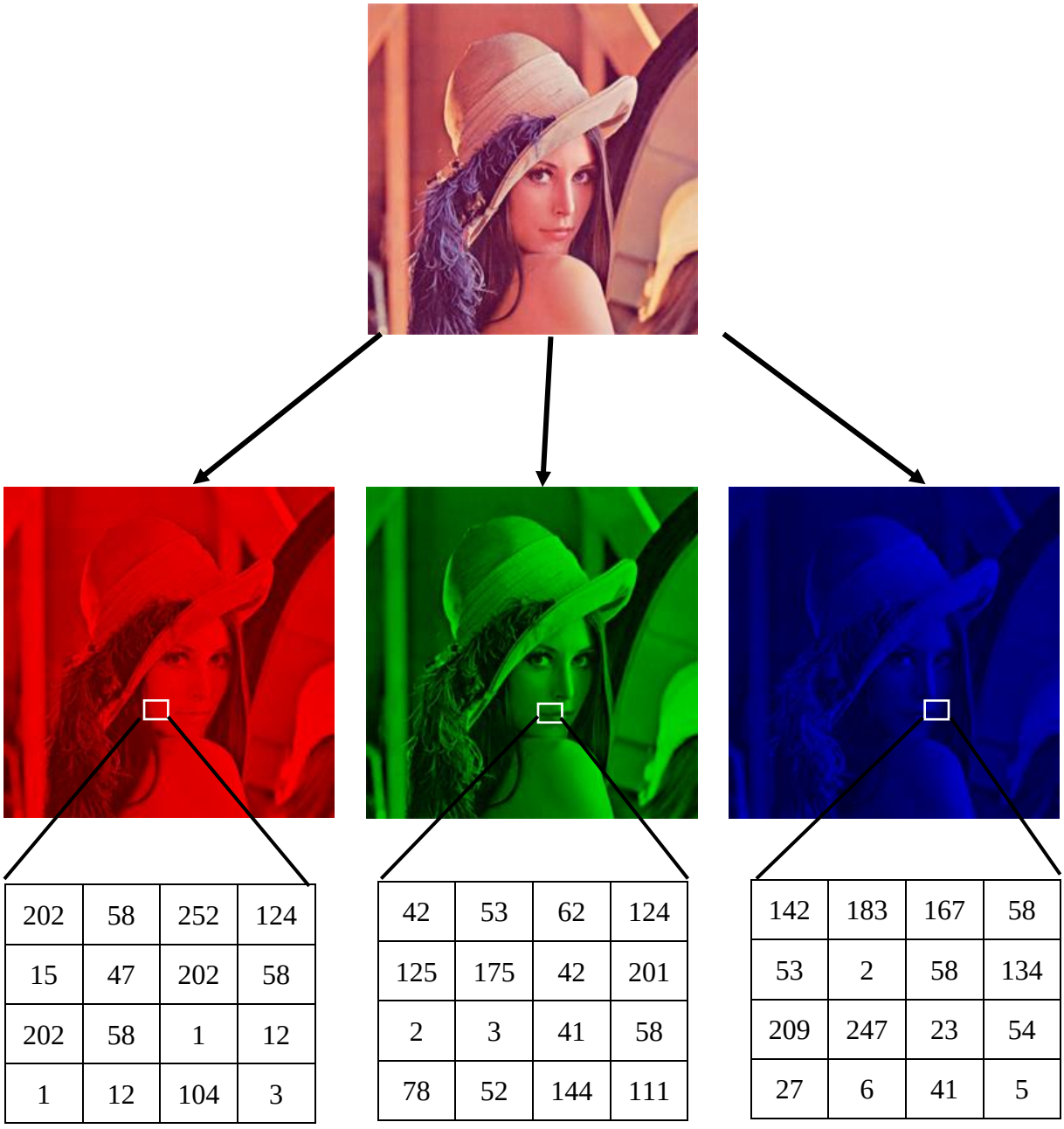
Figure 3-2 : *Matrice de dimension* 16×16 *contenant des valeurs entre 0 et 255.*

L'opération du cryptage de l'image couleur revient à crypter chaque image extraite du trois canaux puis les fusionner ce qui va nous donner à la sortie une image couleur cryptée.

Etant donné une image couleur à crypter, la première étape consiste à décomposer l'image couleur en 3 canaux, canal rouge, canal vert et canal bleu, ensuite on va extraire la matrice des valeurs décimales pour chaque canal (**Figure 3-3**), ces valeurs sont comprises entre 0 et 255, pour chaque canal, on va calculer la somme de tous les pixels modulo 255, puis on va prendre la partie entière de la moyenne des valeurs obtenues, le résultat va être un entier entre 0 et 255, cette valeur on va la mettre dans la première case de la première matrice, le reste des cases de la matrice va être rempli par des valeurs distinguées et générées aléatoirement entre 0 et 255. La même procédure va être appliquée pour toutes les cases de la 2eme matrice.

Le calcul de la première case de la première matrice en fonction des pixels de l'image originale va être justifié dans la partie expérimentation.

La figure ci-dessous (**Figure 3-3**) représente un exemple de 16 pixels d'une matrice extraite à partir d'une image couleur de taille après avoir la décomposer en 3 canaux.



202	58	252	124
15	47	202	58
202	58	1	12
1	12	104	3

42	53	62	124
125	175	42	201
2	3	41	58
78	52	144	111

142	183	167	58
53	2	58	134
209	247	23	54
27	6	41	5

Figure 3-3 : *Décomposition de l'image couleur en 3 canaux et extraction des matrices décimales.*

Après avoir effectué l'opération de la génération des deux matrices, décomposer l'image couleur en 3 canaux et extraire la matrice des valeurs décimales pour chaque canal, on passe maintenant à l'opération de cryptage.

On crypte les valeurs de chaque matrice par digramme (séquence de deux chiffre). À titre d'exemple, cryptons le digramme 202, 58. On repère le chiffre 202 dans la matrice 1, le chiffre 58 dans la matrice 2, puis on cherche dans ces matrices les chiffres qui complètent le rectangle, dans notre exemple (**Figure 3-4**), le chiffre 209 dans la matrice 1 et le chiffre 199 dans la matrice 2. Le digramme 202, 58 est crypté en digramme 199, 209, car par convention l'un des deux chiffres cryptés est sur la même colonne que le chiffre de la matrice originale.

Si les deux chiffres se situent dans la même colonne, leur inversion forme sera le digramme crypté. Par exemple, 75, 184 devient 184, 75 (**Figure 3-5**).

164	250	132	167	189	48	155	98	67	187	45	166	21	102	137	242
204	168	34	160	74	27	141	226	251	205	52	134	181	10	39	38
13	90	0	246	207	153	19	196	95	245	97	59	9	175	8	147
161	143	7	216	228	213	22	229	41	188	223	218	25	99	128	240
88	198	55	28	113	194	210	192	254	30	80	138	174	248	16	235
215	117	125	191	154	197	146	4	60	91	114	159	201	96	3	220
185	150	241	209	1	170	173	208	75	24	139	255	156	32	202	57
186	214	15	123	212	162	227	6	244	222	206	93	112	109	176	61
238	44	165	121	58	172	211	183	221	82	20	234	76	108	89	130
237	146	104	69	133	46	105	43	182	180	37	177	136	110	33	135
12	119	47	120	65	118	230	149	152	71	35	5	190	171	92	62
184	232	54	106	63	68	224	14	203	115	87	11	225	42	17	122
40	101	79	126	157	86	36	247	73	100	151	179	142	124	163	26
148	77	64	85	140	169	193	231	158	253	23	199	195	29	107	66
127	51	131	200	78	103	18	84	56	83	50	219	252	243	53	178
239	111	94	144	70	217	249	72	49	129	233	31	116	236	2	81

254	230	108	116	152	119	32	88	10	81	90	148	174	169	136	227
237	50	144	34	103	216	11	222	125	41	170	61	183	196	53	224
197	177	76	139	70	229	4	17	37	46	110	51	44	111	247	188
167	241	255	153	78	12	213	49	29	43	62	245	163	19	120	104
82	157	220	155	240	166	14	65	48	38	168	212	250	128	146	193
97	26	246	198	52	6	109	147	113	175	15	181	21	200	95	60
159	130	64	156	252	115	71	232	100	45	157	18	173	2	105	121
150	75	23	3	236	13	96	27	1	118	39	69	202	74	22	239
114	162	251	112	165	248	243	102	242	192	98	219	75	164	194	160
208	83	35	178	0	149	84	142	79	214	132	211	191	180	85	134
117	143	77	58	195	182	54	215	184	137	244	231	87	158	199	221
47	89	24	30	209	226	122	124	42	133	123	56	138	28	203	80
99	210	91	151	7	234	5	223	8	67	127	20	126	228	185	186
238	253	106	176	135	68	207	59	161	205	171	217	36	63	86	154
9	225	140	189	31	93	187	249	94	40	204	201	172	235	72	145
16	92	131	66	218	55	190	141	33	206	233	25	129	101	179	107

Figure 3-4 : *Opération de cryptage du digramme **202, 58** situant dans des colonnes différentes.*

164	250	132	167	189	48	155	98	67	187	45	166	21	102	137	242
204	168	34	160	74	27	141	226	251	205	52	134	181	10	39	38
13	90	0	246	207	153	19	196	95	245	97	59	9	175	8	147
161	143	7	216	228	213	22	229	41	188	223	218	25	99	128	240
88	198	55	28	113	194	210	192	254	30	80	138	174	248	16	235
215	117	125	191	154	197	146	4	60	91	114	159	201	96	3	220
185	150	241	209	1	170	173	208	75	24	139	255	156	32	202	57
186	214	15	123	212	162	227	6	244	222	206	93	112	109	176	61
238	44	165	121	58	172	21	183	221	82	20	234	76	108	89	130
237	145	104	69	133	46	105	43	182	180	37	177	136	110	33	135
12	119	47	120	65	118	230	149	152	71	35	5	190	171	92	62
184	232	54	106	63	68	224	14	203	115	87	11	225	42	17	122
40	101	79	126	157	86	36	247	73	100	151	179	142	124	163	26
148	77	64	85	140	169	193	231	158	253	23	199	195	29	107	66
127	51	131	200	78	103	18	84	56	83	50	219	252	243	53	178
239	111	94	144	70	217	249	72	49	129	233	31	116	236	2	81

254	230	108	116	152	119	32	88	10	81	90	148	174	169	136	227
237	50	144	34	103	216	11	222	125	41	170	61	183	196	53	224
197	177	76	139	70	229	4	17	37	46	110	51	44	111	247	188
167	241	255	153	78	12	213	49	29	43	62	245	163	19	120	104
82	57	220	155	240	166	14	65	48	38	168	212	250	128	146	193
97	26	246	198	52	6	109	147	113	175	15	181	21	200	95	60
159	130	64	156	252	115	71	232	100	45	197	18	173	2	105	121
150	73	23	3	236	13	96	27	1	118	39	69	202	74	22	239
114	162	251	112	165	248	243	102	242	192	98	219	75	164	194	160
208	83	35	178	0	149	84	142	79	24	132	211	191	180	85	134
117	143	77	58	195	182	54	215	184	137	244	231	87	158	199	221
47	89	24	30	209	226	122	124	42	133	123	56	138	28	203	80
99	210	91	151	7	234	5	223	8	67	127	20	126	228	185	186
238	253	106	176	135	68	207	59	161	205	171	217	36	63	86	154
9	225	140	189	31	93	187	249	94	40	204	201	172	235	72	145
16	92	131	66	218	55	190	141	33	206	233	25	129	101	179	107

Figure 3-5 : *Opération de cryptage du digramme **75, 184** situant dans la même colonne.*

Le problème qui se pose c'est que cette méthode est vulnérable aux attaques cryptanalytiques, chaque digramme a une seule forme de cryptage (**Figure 3-6**), ce qui va démunir grandement le nombre d'essai d'une attaque de brute force cryptanalytique basée sur l'analyse fréquentielle des digrammes.

202	**58**	252	124
15	47	**202**	**58**
202	**58**	**1**	**12**
1	**12**	104	3



199	**209**	138	84
24	186	**199**	**209**
199	**209**	**78**	**170**
78	**170**	23	69

Figure 3-6 : *Opération de cryptage d'une matrice 4×4.*

Dans l'exemple ci-dessus, on a appliqué notre méthode sur les 16 pixels extraits du canal rouge de l'exemple (**Figure 3-7)**, si on analyse les résultats obtenus, on remarquera que le digramme 201, 58 se répète 3 fois et le digramme 1, 12 se répète 2 fois, avec une attaque brute Force cryptanalytique basée sur l'analyse fréquentielle des digrammes, on va obtenir les résultats suivants (**Tableau 3-1**).

Tableau 3-1 : *Résultats obtenus pour chaque type d'attaque.*

Brute Force Normal	$\mathbf{256^{16} \approx 3.2 \times 10^{38}}$
Brute Force Cryptanalytique	$256^{10} \approx 1.6 \times 10^{24}$

Les résultats obtenus dans le tableau ci-dessus sont calculés de la façon suivante :

Nous avons une matrice de taille 4×4 qui contient des entiers entre 0 et 255, donc la matrice contient 16 case, chaque case peut prendre 256 valeurs, et par suite une attaque de brute force normal va tester 25516 possibilités au pire des cas pour trouver la matrice originale, en revanche, si on prend en considération les digrammes répétées, on va les éliminer puisqu'ils vont prendre la même valeur que celle de la première occurrence, donc une attaque de brute force cryptanalytique va tester 25610 possibilités, ces résultats montrent qu'il existe une large différence au niveau du nombre de possibilités entre une attaque du brute force normal et une attaque de brute force cryptanalytique basée sur l'analyse fréquentielle des digrammes. Donc pour rendre notre méthode est invulnérable pour ce type d'attaque, on va ajouter une deuxième phase de cryptage basée sur l'utilisation des rotations circulaires à droite de la représentation

binaire de la matrice des valeurs décimales obtenue à partir de la première phase de cryptage, ces valeurs sont codées sur 8 bits. Le nombre de bits nécessaire pour faire la rotation doit respecter la formule suivante.

$$N(X) = (K_1 + K_2 + i(X) + j(X))\ mod\ 8 \quad (3\text{-}1)$$

Avec N est une fonction qui calcule le nombre de bits nécessaire pour faire la rotation, X représente une valeur binaire, i et j sont deux fonctions qui retournent respectivement l'indice ligne et colonne de la valeur X, K_1 et K_2 sont des valeurs calculées à partir des matrices 16×16 de cryptage, et elles sont extrêmement importantes puisque tout changement fait au niveau des clés (les 2 matrices) va changer complètement l'image cryptée. Ces valeurs sont calculées à partir des formules suivantes.

$$K_1 = \left(\sum_{i=0}^{15} \sum_{j=0}^{15} M_{ij}^1 \times (i + j) \right)\ mod\ 8 \quad (3\text{-}2)$$

$$K_2 = \left(\sum_{i=0}^{15} \sum_{j=0}^{15} M_{ij}^2 \times (i + j) \right) mod\ 8 \quad (3\text{-}3)$$

Avec M_{ij}^1 la valeur de la i^{eme} ligne et la j^{eme} colonne de la première matrice et M_{ij}^2 la valeur de la i^{eme} ligne et la j^{eme} colonne de la deuxième matrice.

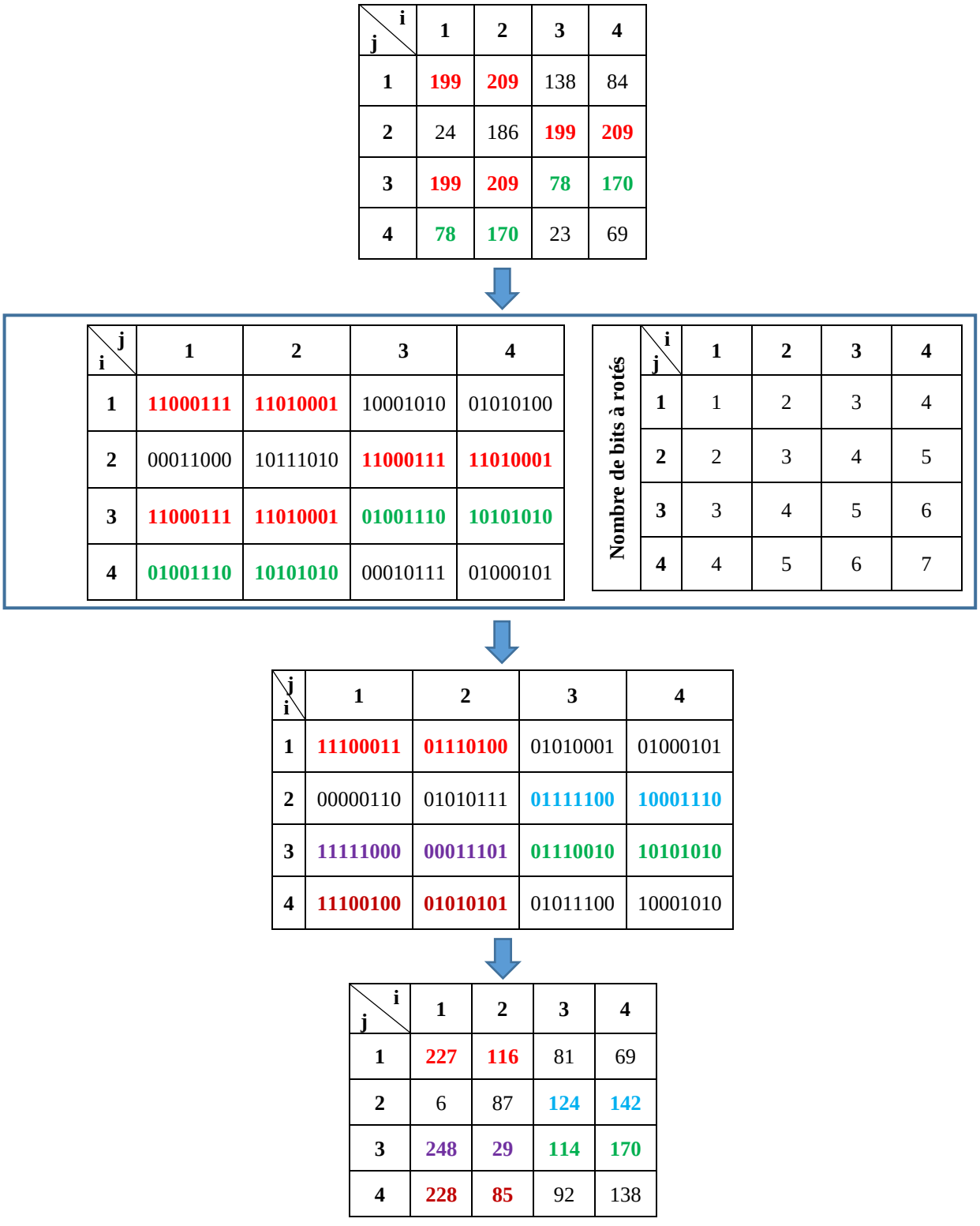


Figure 3-7 : *Rotation circulaire à droite d'une matrice 4×4.*

La **figure 3-7** ci-dessus applique la deuxième phase de cryptage basée sur la rotation circulaire à droite des bits avec $K_1 = 1$ et $K_2 = 6$, ces valeurs sont obtenues en appliquant les équations (2) et (3) sur les matrices de la **figure 3-4**. On remarque que le digramme (199, 209) qui a été répéter 3 fois avant cette phase n'a plus la même forme cryptée à la sortie et la même chose pour le digramme (78, 170), et par suite toute attaque cryptanalytique basée sur l'analyse fréquentielle des digrammes n'aura plus d'influence sur le décryptage de l'image.

3.2.1.1 Synthèse

Dans le travail présenté précédemment, on a proposé une nouvelle approche pour crypter les images couleurs basée sur l'utilisation de deux matrices, de taille 16×16 dont les valeurs entières sont comprises entre 0 et 255 et générées aléatoirement avec la rotation circulaire des bits. Ces matrices sont utilisées pour effectuer la première phase de cryptage. La première valeur de la première matrice est calculée à partir des pixels de chaque canal de l'image originale, le reste des valeurs est généré aléatoirement de telle sorte qu'elles soient distinguées ainsi que les valeurs de la deuxième matrice sont différentes et générées aléatoirement. Le cryptage des pixels de l'image originale est fait par digramme (séquence de deux chiffres). On repère le premier chiffre dans la première matrice, le deuxième chiffre dans la deuxième matrice, puis on cherche dans ces matrices les chiffres qui complètent le rectangle. La deuxième phase de cryptage consiste à faire une rotation circulaire à droite des bits suivant une fonction qui prend en compte les valeurs des deux matrices ainsi que leurs positions (ligne et colonne). En effet tout changement au niveau des deux clés (les 2 matrices) va changer complètement l'image cryptée.

3.2.2 Nouvel algorithme de cryptage d'images couleurs en utilisant plusieurs cartes chaotiques avec la méthode d'intersection des plans

Ce nouvel algorithme se compose de deux phases. Dans la première phase, on a utilisé les trois faces adjacentes d'un cube qui a comme dimension $16 \times 16 \times 16$, chaque face représente un des trois canaux d'une image couleur (rouge, vert et bleu) **(Figure 3-20)** et il représente une matrice de taille 16×16 dont les valeurs sont uniques et comprises entre 0 et 255. Ces valeurs sont générées pseudo-aléatoirement en utilisant trois fonctions chaotiques (Logistic map, Sine map et Chebychev map) qu'on va les discuter brièvement par la suite.

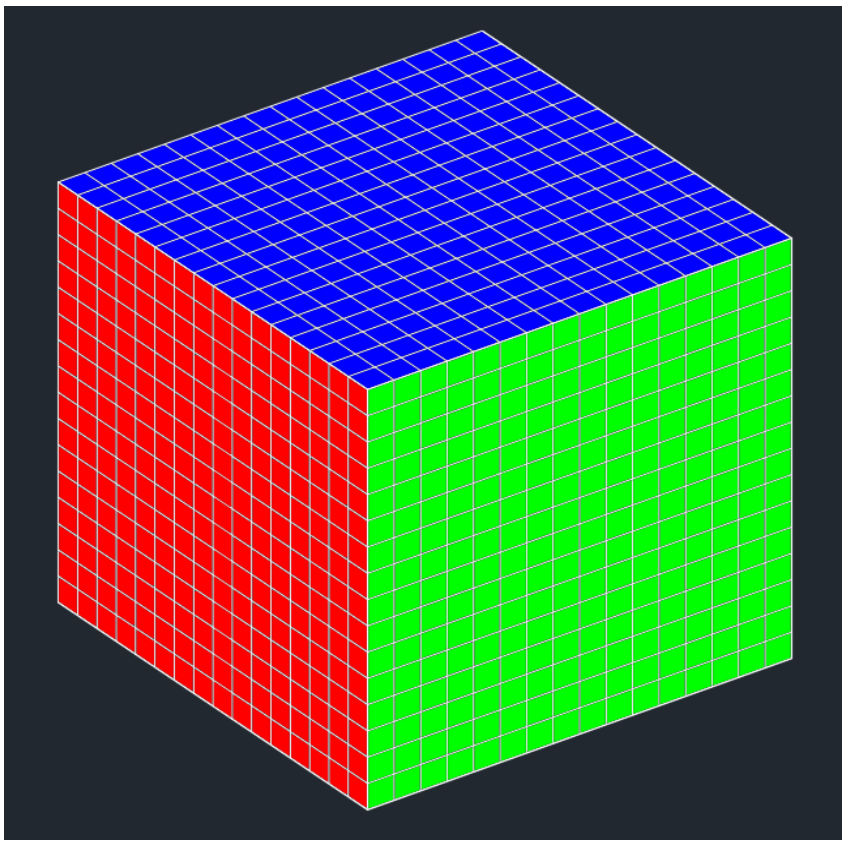


Figure 3-8 : Trois faces du cube de notre système de cryptage.

3.2.2.1 Logistic Map

C'est une fonction non linéaire qui a un comportement chaotique lorsque le paramètre μ appartient à l'intervalle $[3.57, 4]$, elle est définie comme suivant :

$$x_{n+1} = L(\mu, x_n) = \mu \times x_n \times (1 - x_n) \tag{3-19}$$

Avec $\mu \in [0\ ;4]$ et $x_0 \in [0\ ;1]$.

Le comportement chaotique de cette fonction est présenté dans la **figure 3-20**. D'après le diagramme de bifurcation, On peut voir que l'intervalle chaotique est $[3.57\ ;4]$, ce qui est approuver par le diagramme de Lyapunov Exponent, ce dernier prends des valeurs inferieures à zéro lorsque $\mu < 3.57$, et d'autres supérieures à zéro lorsque $\mu \geq 3.57$, ainsi que les valeurs appartient à l'intervalle $[0\ ;1]$.

Pour remplir la face rouge de notre cube, on a utilisé la formule suivante

$$L'(\mu, x_n) = E(255 \times L(\mu, x_n)) \tag{3-20}$$

3.2.2.2 Sine Map

C'est une fonction non linéaire qui a un comportement chaotique similaire à celle de la fonction Logistic Map, elle est définie comme suivant :

$$y_{n+1} = S(r, y_n) = r \times sin(\pi \times y_n) \quad (3\text{-}21)$$

Avec $r \in [0\ ;1]$ et $y_0 \in [0\ ;1]$.

D'après le diagramme de bifurcation et le diagramme de Lyapunov Exponent (**Figure 3-21**), on peut constater que le Sine Map a un comportement similaire avec Logistic Map.

Pour remplir la face verte de notre cube, on a utilisé la formule suivante

$$S'(r, y_n) = E(255 \times S(r, y_n)) \quad (3\text{-}22)$$

3.2.2.3 Chebychev Map

C'est une fonction non linéaire qui a un comportement chaotique lorsque $a > 1$, elle est définie comme suivant :

$$z_{n+1} = C(a, z_n) = cos(a \times \arccos(z_n)) \quad (3\text{-}23)$$

Avec $a \in \mathbb{N}$.

Les diagrammes de bufrication et de Lyapunov Exponent présentés dans la **figure 3-21** montrent que Chebyshev Map prends un comportement chaotique lorsque le paramètre a est supérieur à 1, ainsi que les valeurs sont comprises entre -1 et 1.

$$C'(a, z_n) = E(255 \times |C(a, z_n)|) \quad (3\text{-}24)$$

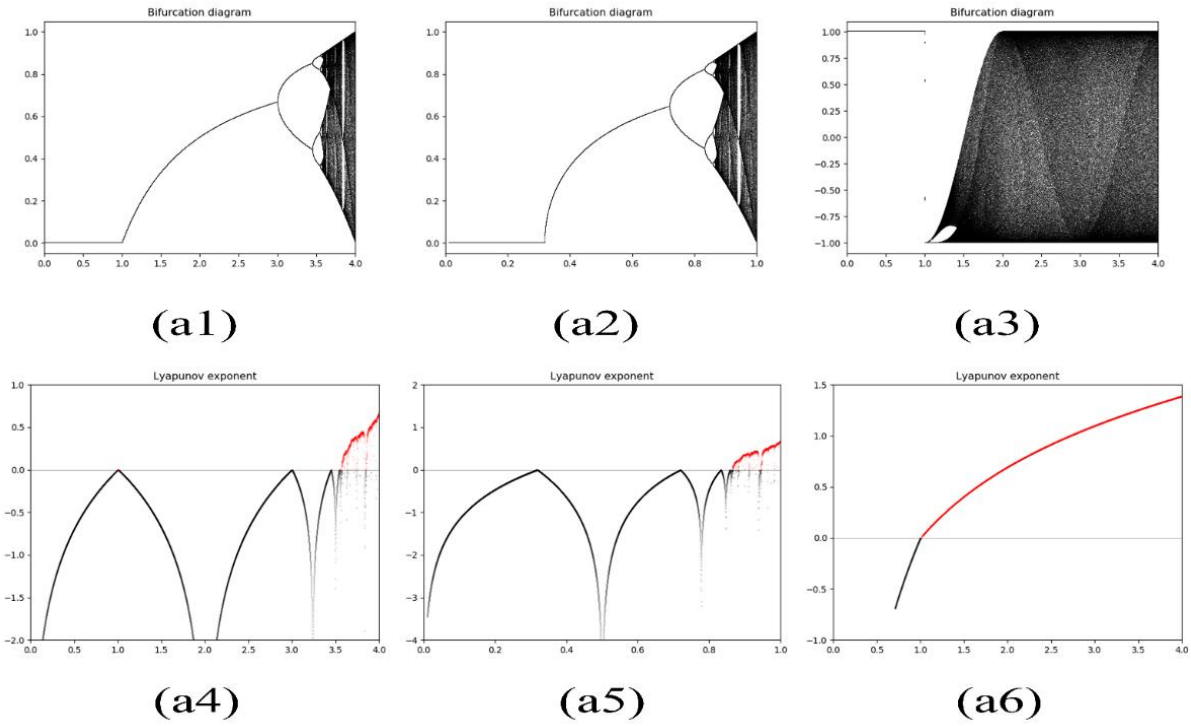


Figure 3-9 : *Bifurcation Diagram of: (a1) logistic Map. (a2) Sine Map. (a3) Chebyshev Map. Lyapunov exponent Diagram of: (a4) logistic Map. (a5) Sine Map. (a6) Chebyshev Map.*

Interprétation mathématique de la méthode :

Après avoir remplir les trois faces du cube avec les fonctions chaotiques L', S' et C' définit précédemment, on va maintenant commencer l'opération de cryptage

Soient $N, M \in \mathbb{N}^*$

Soient $P = (P_{ij})_{\substack{0 \le i \le N-1 \\ 0 \le j \le M-1}} \in \mathcal{M}_{N \times M}(\{0,1,\dots,255\}^3)$

Où : $\mathcal{M}_{N \times M}(\{0,1,\dots,255\}^3)$ est l'ensemble des matrices carrées dont les composantes appartient à $\{0,1,\dots,255\}^3$

Et $R, G, B \in \mathcal{M}_{16 \times 16}(\{0,1,\dots,255\})$

Avec : $R = (r_{i_1,j_1})_{0 \le i_1,j_1 \le 15}$ $G = (g_{i_2,j_2})_{0 \le i_2,j_2 \le 15}$ $B = (b_{i_3,j_3})_{0 \le i_3,j_3 \le 15}$

Où :

$$\forall\, r_{i_1,j_1}, r_{i_1{}',j_1{}'} \in R,\ r_{i_1,j_1} \neq r_{i_1{}',j_1{}'}\ si\ (i_1,j_1) \neq (i_1{}',j_1{}')$$

$$\forall\, g_{i_2,j_2}, g_{i_2{}',j_2{}'} \in G,\quad g_{i_2,j_2} \neq g_{i_2{}',j_2{}'}\ si\ (i_2,j_2) \neq (i_2{}',j_2{}')$$

$$\forall\, b_{i_3,j_3}, b_{i_3{}',j_3{}'} \in B,\quad b_{i_3,j_3} \neq b_{i_3{}',j_3{}'}\ si\ (i_3,j_3) \neq (i_3{}',j_3{}')$$

Soit $P_{i,j} \in P$, donc $P_{i,j} = \left({x_R}^{(i,j)}, {y_G}^{(i,j)}, {z_B}^{(i,j)}\right)$

On sait qu'il existe $i_1, j_1 \in \{0,1,\dots,15\}$ / ${x_R}^{(i,j)} = \ r_{i_1,j_1}$

De même :

$$\exists!\ i_2, j_2 \in \{0,1,\dots,15\}\ / {y_G}^{(i,j)} = g_{i_2,j_2}$$

$$\exists!\ i_3, j_3 \in \{0,1,\dots,15\}\ / {z_B}^{(i,j)} = \ b_{i_3,j_3}$$

Nous avons fait l'extraction des pixels de l'image couleur et cherché la valeur correspondante dans les trois faces du cube (rouge, vert et bleu). Avant de faire l'opération de cryptage, une rotation circulaire **(Figure 3-22)** est faite en fonction du positionnement (ligne, colonne) du pixel, cette rotation est très importante pour notre algorithme, car il permet d'avoir différentes valeurs cryptées pour les pixels qui ont les mêmes valeurs au niveau de chaque canal (rouge, vert et bleu).

Pour $k \in \{1,2,3\}$.

Considérons l'application σ définit par :

$$\begin{aligned} \sigma \quad : \quad & \{0,1,\dots,15\}^2 \to \{0,1,\dots,15\}^2 \\ & (i_k, j_k) \to \sigma(i_k, j_k) = (\sigma(i_k), \sigma(j_k)) \end{aligned}$$

Où :

$$\begin{cases} i + i_k \equiv \sigma(i_k) \quad mod \quad 16 \\ j + j_k \equiv \sigma(j_k) \quad mod \quad 16 \end{cases}$$

Dès que la rotation circulaire expliquée précédemment est terminé. On commence directement l'opération de cryptage en utilisant la méthode d'intersection des plans avec les faces du cube **(Figure 3-23, 3-24, et 3-25)** qui est définit par la fonction suivante.

Considérons l'application φ définit par :

$$\begin{aligned} \varphi \quad : \quad & \{0,1,\dots,255\}^3 \to \{0,1,\dots,255\}^3 \\ & \left({x_R}^{(i,j)}, {y_G}^{(i,j)}, {z_B}^{(i,j)}\right) \to \varphi\left({x_R}^{(i,j)}, {y_G}^{(i,j)}, {z_B}^{(i,j)}\right) \end{aligned}$$

Avec :

$$\varphi\left({x_R}^{(i,j)}, {y_G}^{(i,j)}, {z_B}^{(i,j)}\right) = (\ r_{i_2,j_3},\ g_{i_3,j_1},\ b_{i_1,j_2})$$

Pour la partie de la confusion, on a utilisé Arnold Cat Map qui est une transformation 2D basée sur une matrice de taille 2×2 avec un déterminant qui égale à 1, cette matrice va nous permettre de mélanger les pixels et changer leurs positionnements.

Cette matrice est définie comme suivant :

$$\begin{pmatrix} 1 & A \\ B & AB+1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix} \ mod\ n \tag{3-25}$$

Avec :

(x, y) représente la position des pixels obtenus après l'exécution de la phase précédente qui vont être mappés vers leurs nouvelles positions (x', y').

A est un entier calculé à partir des pixels de l'image original en utilisant l'opérateur XOR.

$$A = XOR\left(\left(P_{ij}\right)_{\substack{0 \le i \le N-1 \\ 0 \le j \le M-1}}\right) \tag{3-26}$$

B est la partie entière de la moyenne des variances pour chaque canal (rouge, vert et bleu).

$$V_x = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{3-27}$$

$$B = E\left(\frac{V_R + V_G + V_B}{3}\right) \tag{3-28}$$

Dans la phase de décryptage, nous utilisons la transformation suivante afin de transformer l'image mélangée en image d'entrée.

$$\begin{pmatrix} AB+1 & -A \\ -B & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \ mod\ n \tag{3-29}$$

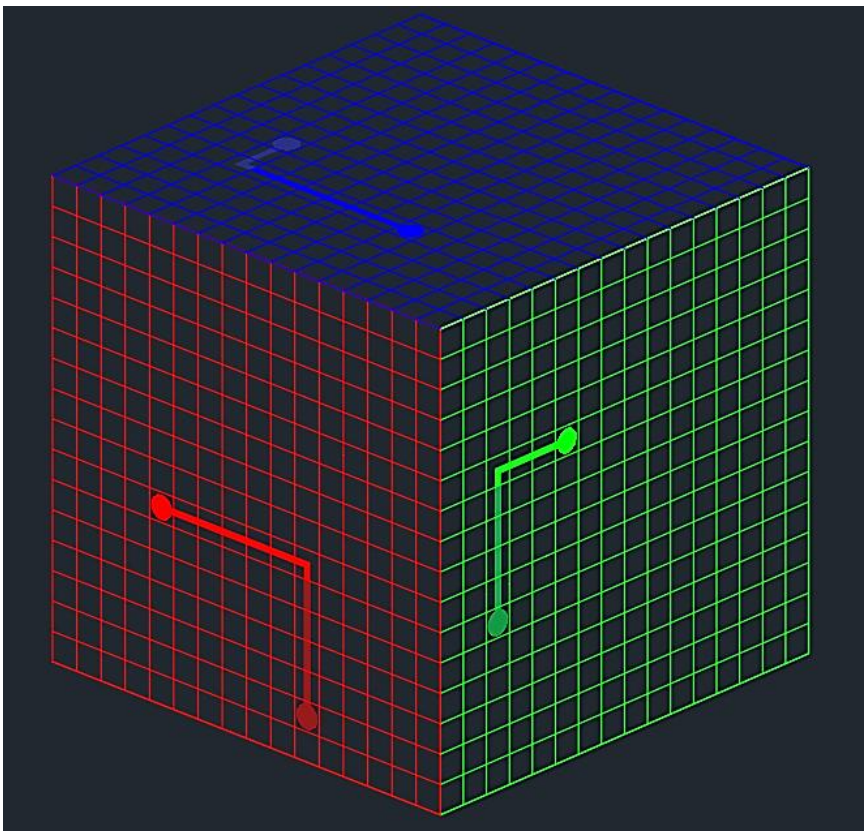


Figure 3-10 : *Rotation circulaire pour chaque pixel trouvé dans le cube.*

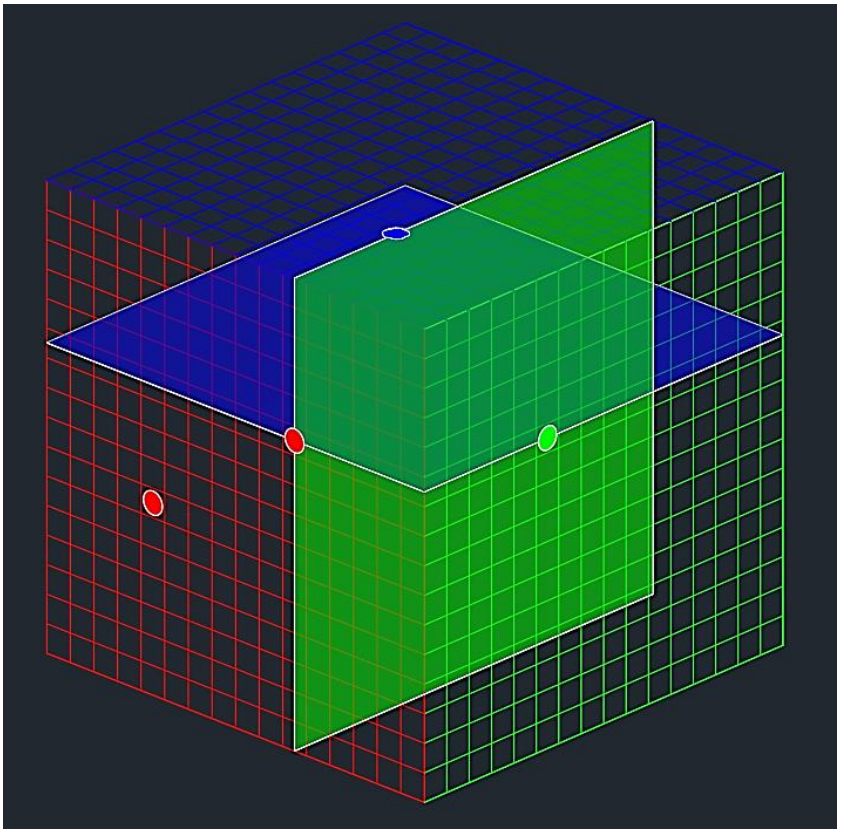


Figure 3-11 : *Opération de cryptage au niveau du canal rouge*

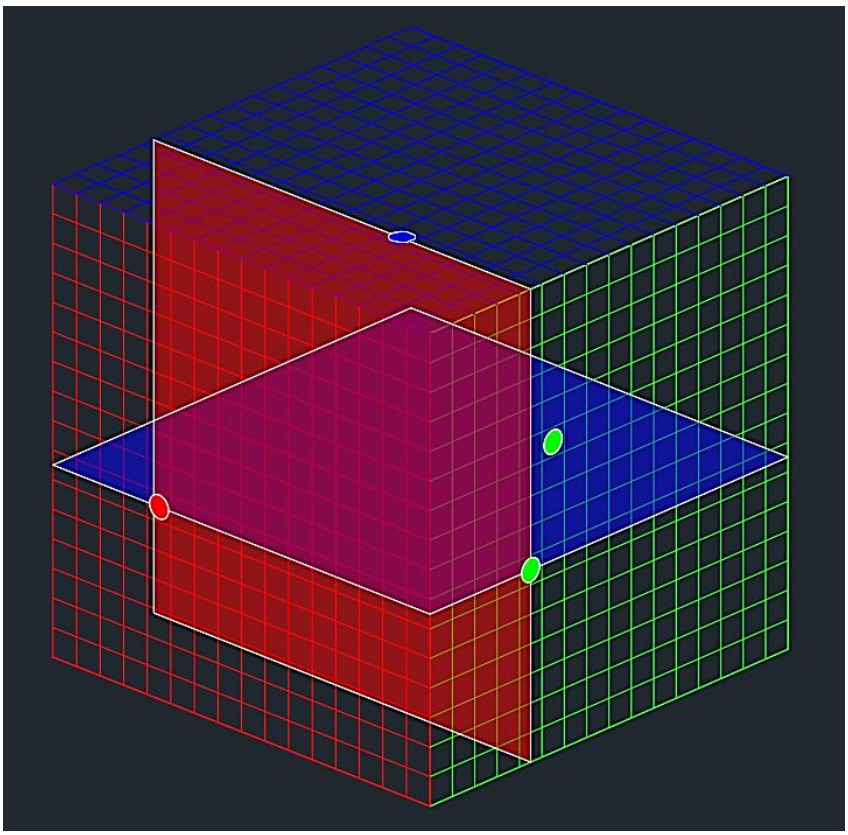


Figure 3-12 : *Opération de cryptage au niveau du canal verte.*

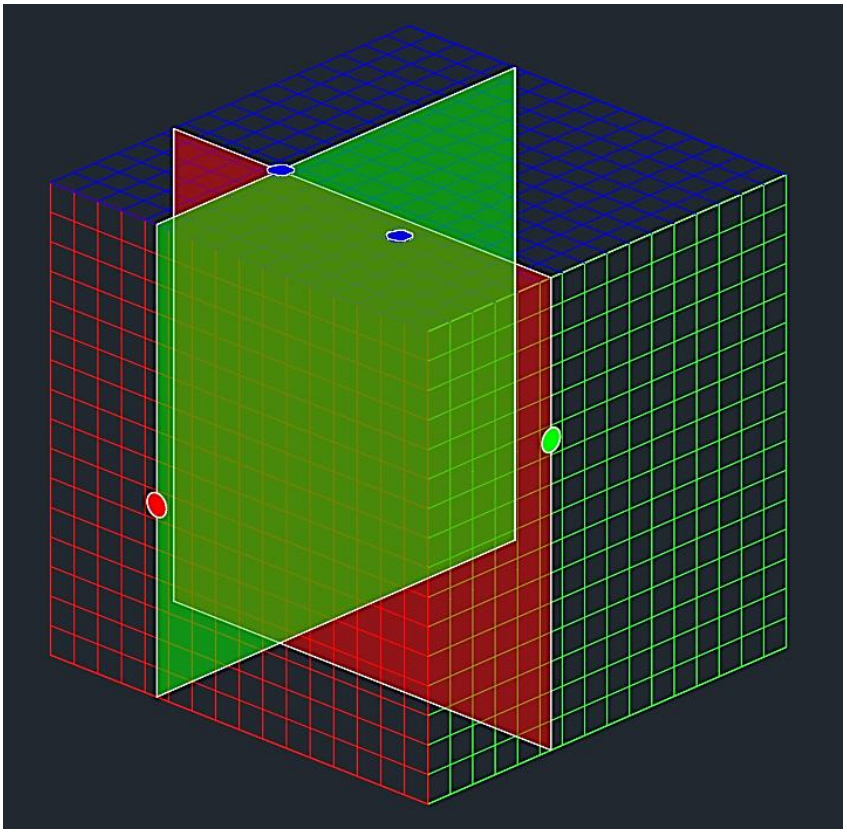


Figure 3-13 : *Opération de cryptage au niveau du canal bleu.*

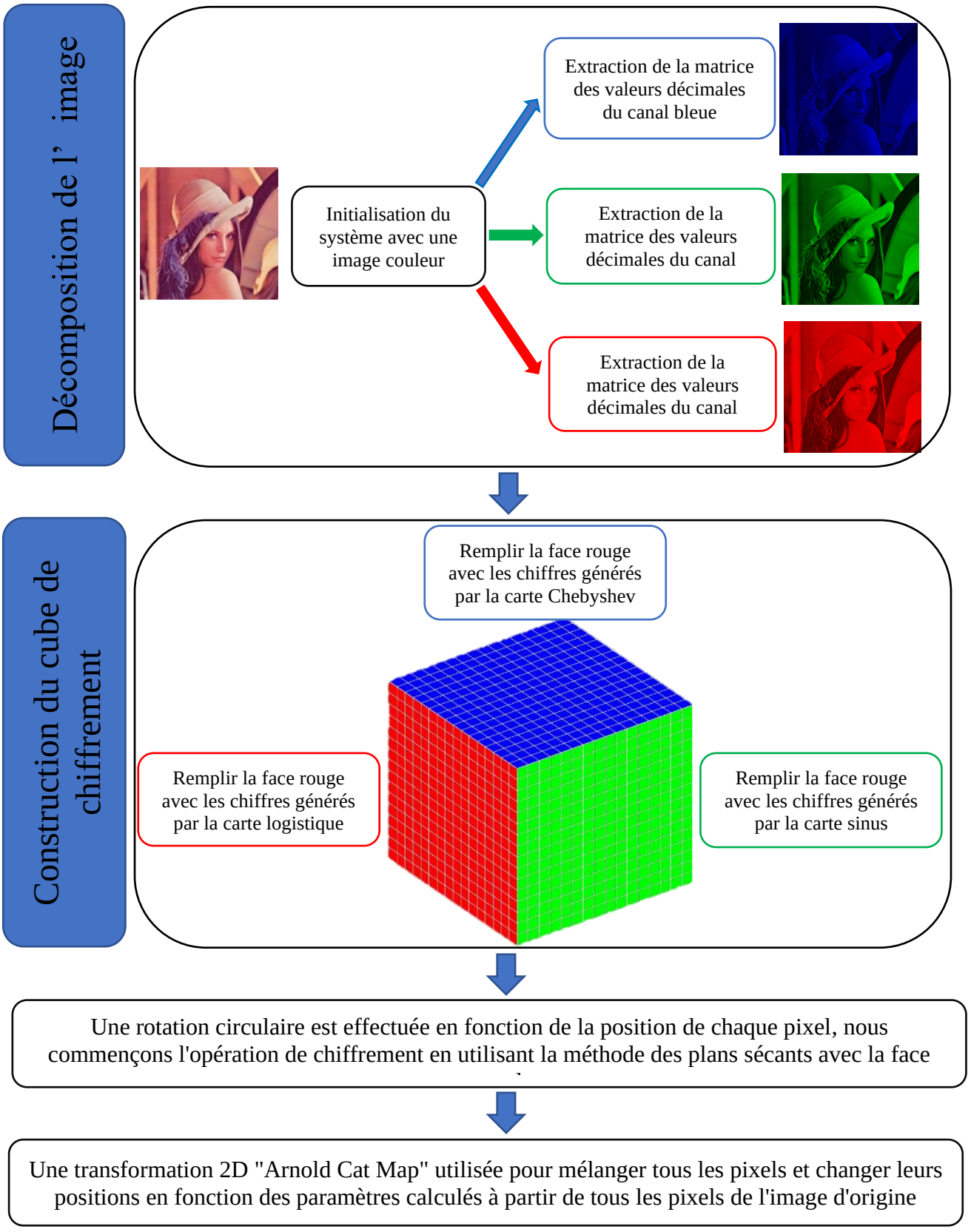


Figure 3-14 : *Organigramme de la méthode proposée*

3.2.2.4 Synthèse

Dans la méthode présentée précédemment, on a développé une nouvelle méthode pour la cryptographie des images couleurs basée sur l'utilisation de plusieurs fonctions chaotiques (logistic map, sine map et chebychev map). Ces fonctions nous ont servi pour remplir les trois faces adjacentes de notre cube de cryptage, ces faces représentent les trois canaux d'une image couleur (rouge, vert et bleu). La première phase de notre algorithme consiste à faire une extraction des pixels de l'image originale et chercher les valeurs correspondantes de chaque pixel dans le cube. Une rotation circulaire est faite en se basant sur la position de chaque pixel (ligne, colonne), cette rotation permet d'éviter le cryptage de deux pixels identiques avec la même valeur. La méthode d'intersection des plans avec la face correspondante est utilisée pour crypter les pixels. Pour la partie de la confusion, on a choisi la transformation 2D « Arnold Cat Map » pour mélanger les pixels et changer leurs positionnements avec des paramètres dépendant de tous les pixels de l'image originale.

3.2.3 Nouvelle méthode de cryptage d'images couleurs 32 bits basée sur plusieurs cartes chaotiques de type 1D et la méthode de chiffrement à quatre carrés

Cette méthode est une amélioration de notre première contribution. Elle est principalement composée de deux parties. Dans la première partie, nous avons utilisé quatre cartes chaotiques de type 1D (carte logistique, carte de tente, carte de Chebyshev et carte du sinus) pour remplir les quatre matrices de notre système de cryptage avec des entiers uniques allant de 0 à 255 (**Figure 3-34),** chaque matrice ayant une taille de 16×16. Ces matrices représentent les quatre canaux d'une image couleur 32 bits (rouge, vert, bleu et alpha).

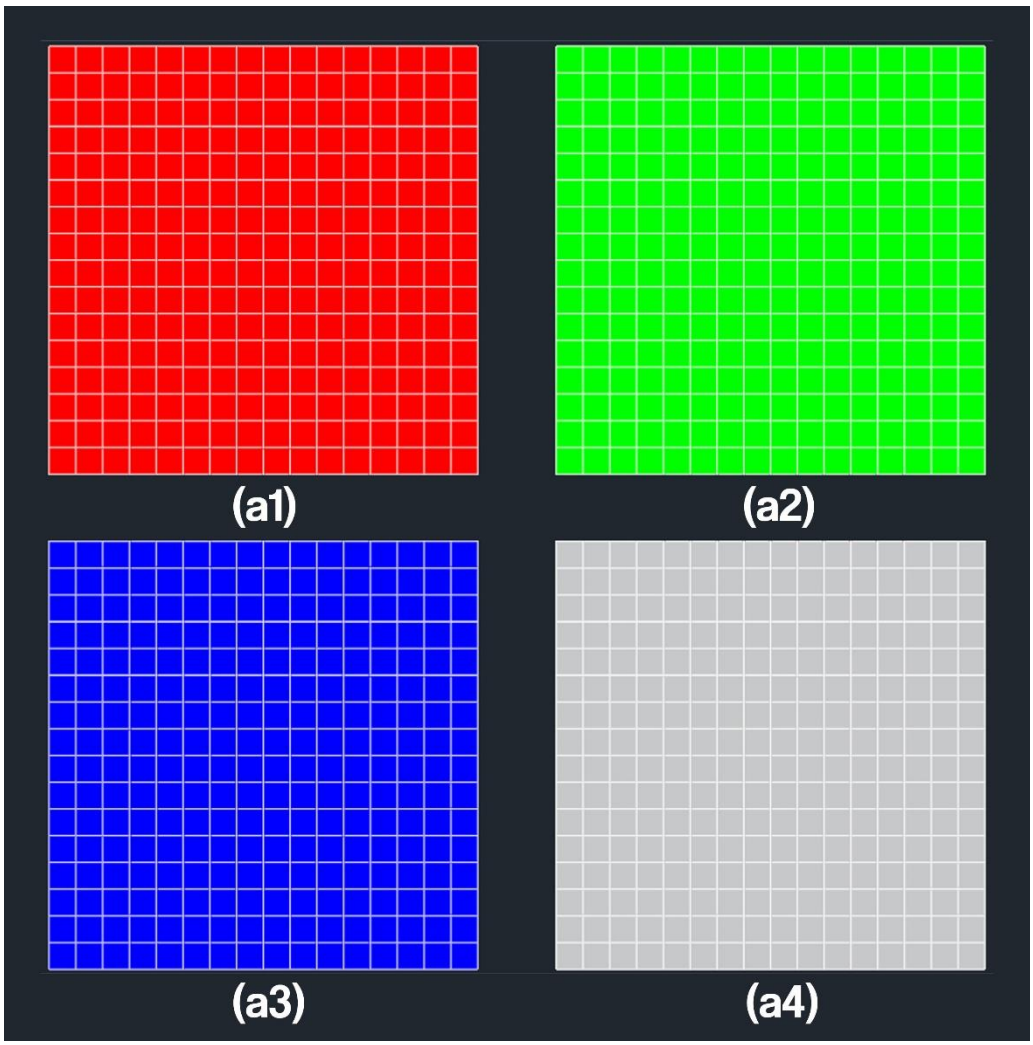


Figure 3-15 : *Système de cryptage rempli avec des entiers uniques générés par Quatre cartes chaotiques.*

3.2.3.1 Cartes chaotiques

La première carte chaotique que nous avons utilisée pour remplir toutes les cases du carré rouge **(Figure 3-34 (a1))** est la carte logistique, qui est une fonction non linéaire avec un comportement chaotique lorsque, $3.57 \leq \alpha \leq 4$ et nous la définissons comme suit.

$$x_{n+1} = L(\alpha, x_n) = \alpha \times x_n \times (1 - x_n) \qquad (3\text{-}54)$$

Avec $\alpha \in [0, 4]$ et $x_0 \in [0, 1]$

Le comportement chaotique de cette carte est illustré dans la **figure 3-35**. L'intervalle de chaos est compris entre 3,57 et 4, comme le montre le diagramme de bifurcation, et cela est confirmé par le diagramme de l'exposant de Liapounov. Si le paramètre α est inférieur à 3,57, nous pouvons voir que ces valeurs appartiennent à l'intervalle $[0, 1]$.

Nous remplissons le carré rouge de notre système de cryptage (les quatre carrés) en utilisant l'équation suivante.

$$L'(\alpha, x_n) = E(255 \times L(\alpha, x_n)) \tag{3-55}$$

La deuxième carte chaotique que nous avons utilisée pour remplir toutes les cases du carré vert **(Figure 3-34 (a2))** est la carte Sinus, qui est similaire à la carte logistique, ils ont le même comportement et la même non-linéarité, elle est définie comme suit.

$$y_{n+1} = S(\beta, y_n) = \beta \times sin(\pi \times y_n) \tag{3-56}$$

Avec $\beta \in [0, 1]$ et $y_0 \in [0, 1]$

La bifurcation et les diagrammes de l'exposant de Lyapunov **(Figure 3-35)** montrent que la carte sinus se comporte de la même manière que la carte logistique. Il a un comportement chaotique lorsque $\beta \in [0.87, 1]$.

Nous avons utilisé l'équation suivante pour remplir la matrice alpha de notre système de cryptage.

$$S'(\beta, y_n) = E(255 \times S(\beta, y_n)) \tag{3-57}$$

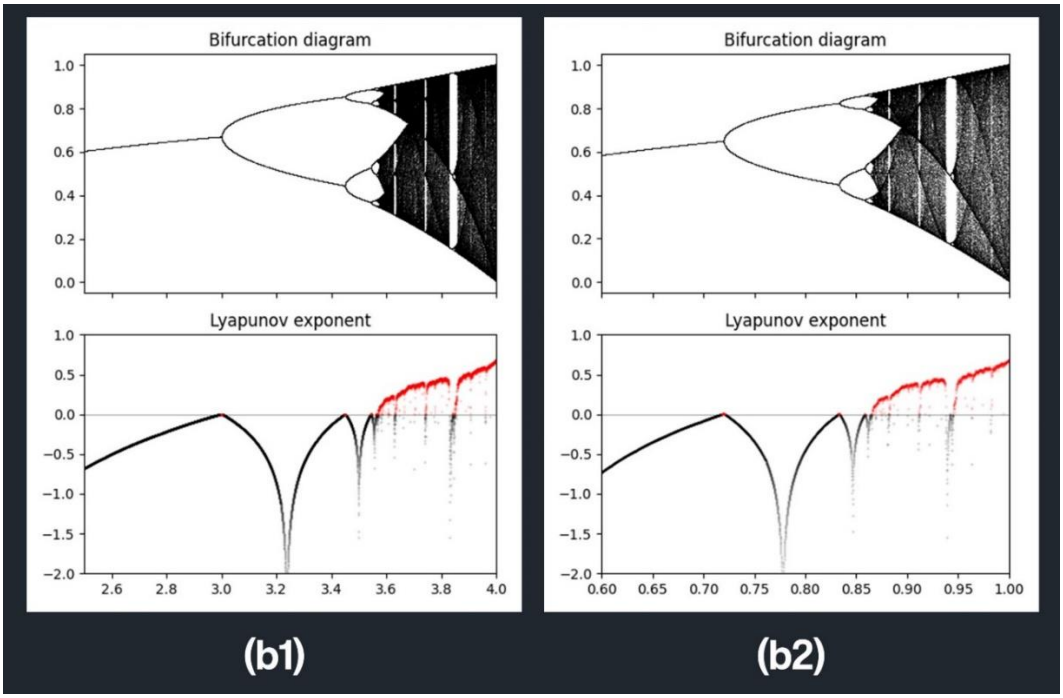


Figure 3-16 : *Diagrammes de bifurcation et d'exposant de Lyapunov : (b1) Carte logistique (b2) Carte sinus.*

La troisième carte chaotique que nous avons utilisée pour remplir toutes les cases du carré bleu **(Figure 3-34 (a3))** est la carte de Chebyshev, le comportement chaotique de cette carte commence lorsque, $\gamma > 1$ nous la définissons comme suit.

$$z_{n+1} = C(\gamma, z_n) = cos(\gamma \times \arccos(z_n)) \tag{3-58}$$

Avec.$\gamma \in \mathbb{N}$

Le comportement chaotique de cette carte est illustré dans la **figure 3-36**. Le diagramme de bifurcation montre que la carte de Chebyshev se comporte de manière chaotique lorsque le paramètre γ est supérieur à 1, et que les valeurs sont dans l'intervalle $[-1, 1]$.

Nous avons utilisé l'équation suivante pour remplir la matrice bleue de notre système de cryptage.

$$C'(\gamma, z_n) = E(255 \times |C(\gamma, z_n)|) \tag{3-59}$$

La carte de la tente qui est la quatrième carte chaotique que nous avons utilisée pour remplir toutes les cases du carré gris **(Figure 3-34 (a4))** est une fonction par morceaux qui met à l'échelle ou plie les valeurs d'entrée en fonction de leur plage. Sa forme généralisée peut être définie mathématiquement comme suit.

$$t_{n+1} = T(\varepsilon, t_n) = \begin{cases} 2 \times \varepsilon \times t_n, & t_n < 0.5 \\ 2 \times \varepsilon \times (1 - t_n), & t_n \geq 0.5 \end{cases} \tag{3-60}$$

Avec le paramètre $\varepsilon \in [0, 1]$.

La **figure 3-36** représente le comportement chaotique de la carte de la tente. L'intervalle chaotique est compris entre 0,5 et 1, comme le montre le diagramme de bifurcation, et cela est confirmé par le diagramme de l'exposant de Liapounov. Lorsque ε est inférieur à 0,5, ce diagramme accepte les valeurs inférieures à zéro et d'autres supérieures à zéro lorsque ε est supérieur à 0,5. Et nous pouvons voir que les valeurs se situent dans l'intervalle $[0, 1]$.

Nous avons utilisé l'équation suivante pour remplir la matrice grise de notre système de cryptage.

$$T'(\varepsilon, t_n) = E(255 \times |T(\varepsilon, t_n)|) \tag{3-61}$$

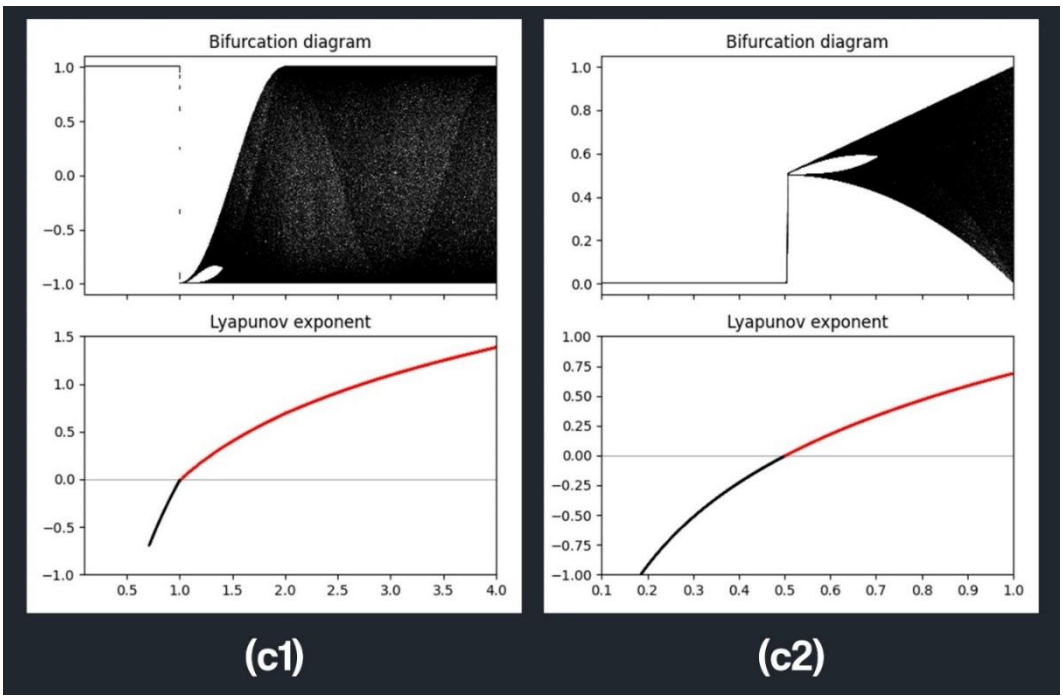


Figure 3-17: *les diagrammes de bifurcation et de l'exposant de Liapounov (c1) Carte de Tchebychev (c2) Carte de la tente.*

3.2.3.2 Interprétation mathématique du système de cryptage

Après avoir rempli les quatre carrés (matrices) de notre système de cryptage avec les quatre cartes chaotiques de type 1D, $L', S', C', et\ T'$ et que nous avons définies précédemment, l'opération de cryptage est définie comme suit.

Soient $N, M \in \mathbb{N}^*$

Soient $P = (P_{ij})_{\substack{0 \le i \le N-1 \\ 0 \le j \le M-1}} \in \mathcal{M}_{N \times M}(\{0,1,\dots,255\}^4)$

Où : $\mathcal{M}_{N \times M}(\{0,1,\dots,255\}^4)$ est l'ensemble des matrices carrées dont les composantes appartiennent à $\{0,1,\dots,255\}^4$.

Et $R, G, B, A \in \mathcal{M}_{16 \times 16}(\{0,1,\dots,255\})$

Avec :

$$R = (r_{i_1,j_1})_{0 \le i_1,j_1 \le 15} \quad G = (g_{i_2,j_2})_{0 \le i_2,j_2 \le 15}$$
$$B = (b_{i_3,j_3})_{0 \le i_3,j_3 \le 15} \quad A = (a_{i_4,j_4})_{0 \le i_4,j_4 \le 15}$$

Où :

$$\forall\, r_{i_1,j_1}, r_{i_1{}',j_1{}'} \in R, \qquad r_{i_1,j_1} \neq r_{i_1{}',j_1{}'}\ if\ (i_1,j_1) \neq (i_1{}',j_1{}')$$
$$\forall\, g_{i_2,j_2}, g_{i_2{}',j_2{}'} \in G, \qquad g_{i_2,j_2} \neq g_{i_2{}',j_2{}'}\ if\ (i_2,j_2) \neq (i_2{}',j_2{}')$$

$$\forall\, b_{i_3,j_3}, b_{i_3{}',j_3{}'} \in B, \qquad b_{i_3,j_3} \neq b_{i_3{}',j_3{}'} \;\; if\;\; (i_3,j_3) \neq (i_3{}',j_3{}')$$

$$\forall\, a_{i_4,j_4}, a_{i_4{}',j_4{}'} \in A, \qquad a_{i_4,j_4} \neq a_{i_4{}',j_4{}'} \;\; if\;\; (i_4,j_4) \neq (i_4{}',j_4{}')$$

Soit $P_{ij} \in P$, donc $P_{ij} = \left({x_R}^{(i,j)}, {y_G}^{(i,j)}, {z_B}^{(i,j)}, {t_A}^{(i,j)}\right)$

Nous savons qu'il existe $i_1, j_1 \in \{0,1,\dots,15\}\;\; tel\; que\;\; {x_R}^{(i,j)} = \;\; r_{i_1,j_1}$

Aussi :

$$\exists!\;\; i_2, j_2 \in \{0,1,\dots,15\}\; / {y_G}^{(i,j)} = g_{i_2,j_2}$$

$$\exists!\;\; i_3, j_3 \in \{0,1,\dots,15\}\; / {z_B}^{(i,j)} = \;\; b_{i_3,j_3}$$

$$\exists!\;\; i_4, j_4 \in \{0,1,\dots,15\}\; / {t_A}^{(i,j)} = \;\; a_{i_4,j_4}$$

Nous extrayons les pixels d'une image couleur 32 bits et vérifions les valeurs correspondantes dans les quatre carrés (matrices) (rouge, vert, bleu et alpha) du système de cryptage. La rotation circulaire est effectuée en fonction de la position du pixel (ligne, colonne) avant le début de l'opération de cryptage. Cette rotation est cruciale pour notre approche car elle nous permet d'avoir différentes valeurs codées pour les pixels de chaque canal (rouge, vert, bleu et alpha) qui ont la même valeur.

Pour chaque $k \in \{1,2,3,4\}$.

Considérons l'application σ définie par :

$$\begin{aligned} \sigma \quad : \quad & \{0,1,\dots,15\}^2 \to \{0,1,\dots,15\}^2 \\ & (i_k, j_k) \quad \to \sigma(i_k, j_k) = (\sigma(i_k), \sigma(j_k)) \end{aligned}$$

Où :

$$\begin{cases} \sigma(i_k) \equiv i + i_k \mod 16 \\ \sigma(j_k) \equiv j + j_k \mod 16 \end{cases}$$

Lorsque la rotation circulaire spécifiée précédemment est terminée. Nous commençons le processus de cryptage tout de suite en utilisant la technique de chiffrement à quatre carrés. Les canaux rouge et alpha sont chiffrés à l'aide des matrices supérieures gauche et inférieure droite, respectivement. Les canaux vert et bleu sont chiffrés à l'aide des matrices supérieures droite et inférieure gauche, respectivement (**figures 3-37, 3-38 et 3-39**). Ce qui suit est une définition mathématique de sa version généralisée.

Considérons φ l'application définie par :

$$\varphi \quad : \quad \{0,1,\dots,255\}^4 \to \{0,1,\dots,255\}^4$$
$$\left({x_R}^{(i,j)}, {y_G}^{(i,j)}, {z_B}^{(i,j)}, {t_A}^{(i,j)}\right) \to \varphi\left({x_R}^{(i,j)}, {y_G}^{(i,j)}, {z_B}^{(i,j)}, {t_A}^{(i,j)}\right)$$

Où :

$$\varphi\left({x_R}^{(i,j)}, {y_G}^{(i,j)}, {z_B}^{(i,j)}, {t_A}^{(i,j)}\right) = (g_{i_1,j_4},\ r_{i_2,j_3}, a_{i_3,j_2}, b_{i_4,j_1})$$

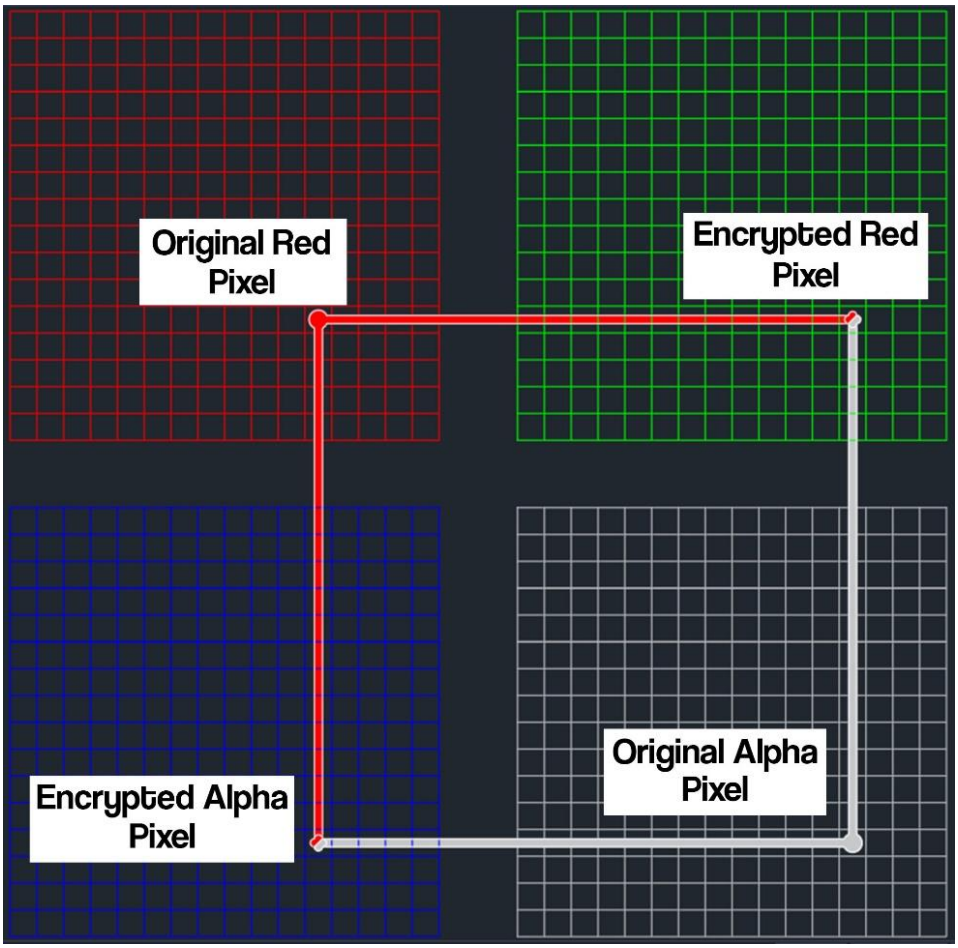


Figure 3-18: *Opération de chiffrement des pixels rouges et alphas.*

Nous avons utilisé la transformation Arnold Cat Map dans la section confusion, qui est une transformation 2D basée sur une matrice 2×2 avec un déterminant de 1. Vladimir Arnold l'a présentée, et il a utilisé un croquis d'un chat pour démontrer les qualités chaotiques de cette carte. Cette matrice nous permettra de mélanger et de déplacer les pixels.

Cette matrice est définie comme suit :

$$\begin{pmatrix} 1 & X \\ Y & XY+1 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} i' \\ j' \end{pmatrix} \ mod\ n \qquad (3\text{-}61)$$

Où :

(i, j) représente les coordonnées des pixels après l'étape précédente, mappées à leurs nouvelles coordonnées (i', j') .

A est un entier calculé à l'aide de l'opérateur XOR à partir des pixels de l'image d'origine. .

$$A = XOR\left(\left(P_{ij}\right)_{\substack{0\le i\le N-1\\ 0\le j\le M-1}}\right) \tag{3-62}$$

B est la partie entière de la moyenne de toutes les variances calculées pour chaque canal (rouge, vert, bleu et alpha).

$$V_x = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{3-63}$$

$$B = E\left(\frac{V_{Red} + V_{Green} + V_{Blue} + V_{Alpha}}{4}\right) \tag{3-64}$$

Pendant la phase de déchiffrement, l'image chiffrée a été transformée en image 32 bits d'origine à l'aide de la transformation suivante.

$$\begin{pmatrix} XY+1 & -X \\ -Y & 1 \end{pmatrix}\begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{pmatrix} i \\ j \end{pmatrix} \; mod\; n \tag{3-65}$$

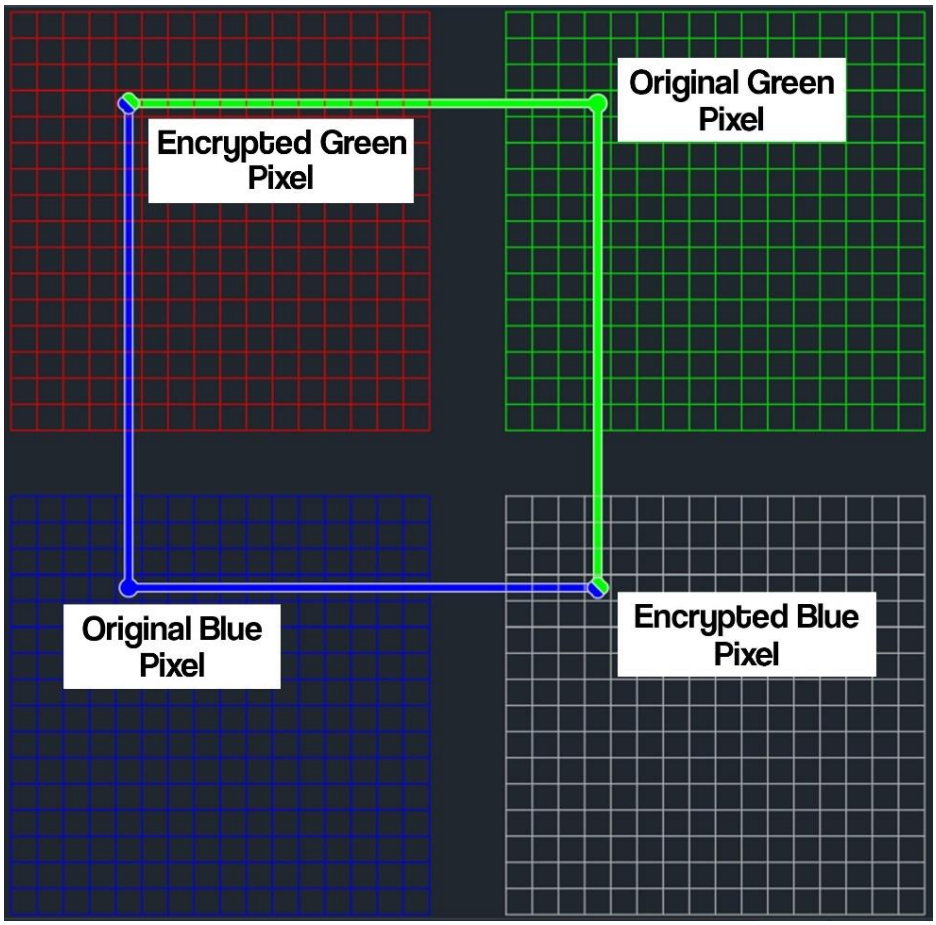


Figure 3-19: *Opération de chiffrement des pixels verts et bleus.*

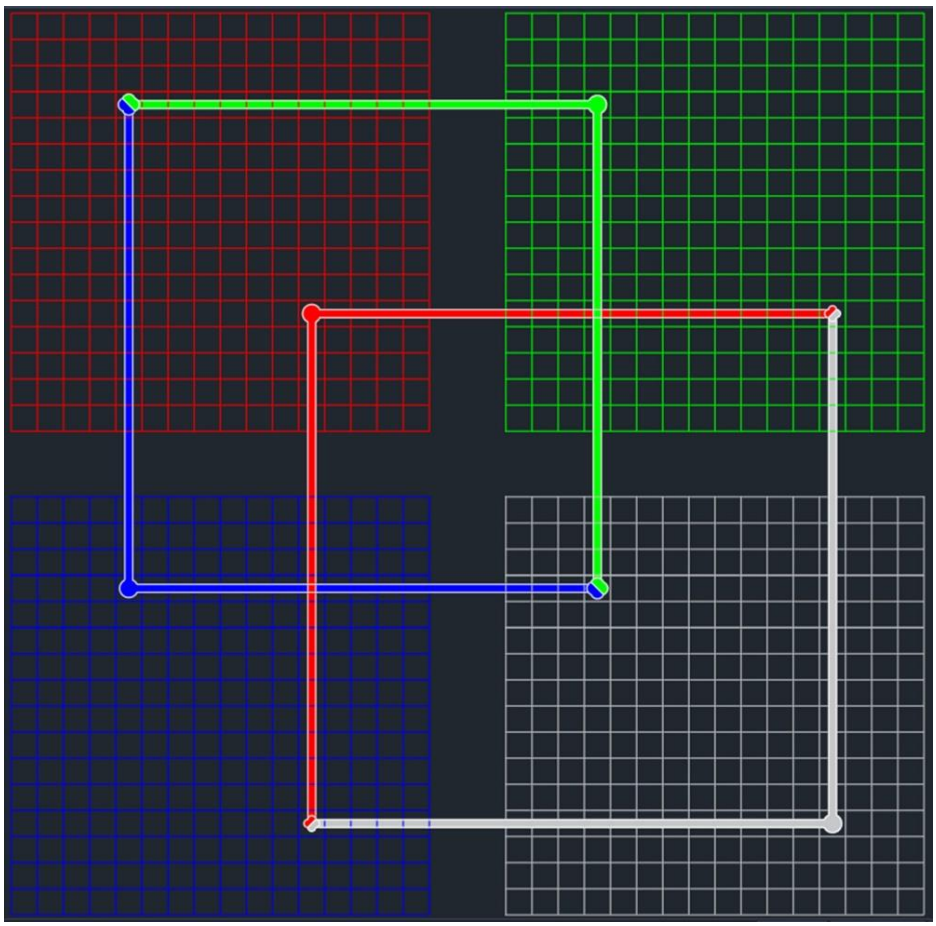


Figure 3-20: *Opération de chiffrement pour tous les canaux.*

3.3 Conclusion

Dans ce chapitre, nous avons présenté trois contributions dans la cryptographie des images couleurs (24 bits, 32 bits). Pour construire nos crypto systèmes, nous avons utilisé plusieurs cartes chaotiques pour la génération de la clé, telles que la carte logistique, la carte sinus, la carte Chebyshev, et la carte de tente. En utilisant des systèmes chaotiques, un algorithme cryptographique est capable de randomiser les données afin qu'elles soient beaucoup plus difficiles à déchiffrer. Cela en fait une solution intéressante pour sécuriser les transmissions de données et s'assurer que les communications restent confidentielles. De plus, les systèmes chaotiques peuvent être mis en œuvre de manière à pouvoir s'adapter aux conditions changeantes, ce qui leur permettent de rester efficaces tout au long de la vie du système cryptographique. La performance de nos contributions sera discutée dans le prochain chapitre.

Chapitre 4 : Expérimentations et discussions

4.1 Introduction

Pour valider la performance et la robustesse de nos algorithmes, nous avons effectué plusieurs tests et attaques qui existent dans le domaine de la cryptographie des images, tels que, les attaques statistiques, les attaques différentielles, l'entropie et d'autres.

Donc, on a pris plusieurs images couleurs de différentes tailles (256×256, 512×512 et 1024×1024), puis on a appliqué nos algorithmes de cryptage. Les résultats et la performance ont été comparés avec celles d'autres méthodes qu'on va les expliquer dans les prochaines sections.

La configuration de notre machine ainsi que le langage de programmation utilisée sont montrées dans le tableau ci-dessous.

Tableau 4-1 : *Configuration de la machine*

Système d'exploitation	Windows 10 Famille 64 bits
Processeur	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHZ (8 CPUS), ~2.8GHZ
RAM	16 Go
Carte Graphique	NVIDIA GeForce GTX 1050
Langage de programmation	JAVA

4.2 Critères d'évaluation utilisés dans les approches proposées

4.2.1 Histogramme

Le premier critère consiste à analyser les histogrammes des images couleurs originales et ceux des images cryptées avec nos propres algorithmes et faire une comparaison entre eux au niveau de la distribution des pixels pour chaque canal (rouge, vert et bleu). On sait que l'histogramme contient plusieurs informations et caractéristiques de l'image. Un attaqueur peut

utiliser ces informations pour décrypter une image. C'est la raison pour laquelle un bon algorithme doit casher ou bien détruire tels informations afin de rendre l'image cryptée plus sécurisée.

4.2.2 Analyse de corrélation

Le coefficient de corrélation représente une information interne très importante de l'image, la majorité des attaques statistiques utilisent cette information pour décrypter les images. C'est pourquoi, un bon algorithme doit se débarrasser de cette information pour qu'il ne soit plus exploiter par les attaqueurs, et rendre les images plus vulnérables par la suite.

Pour ce faire, il faut minimiser la valeur du coefficient de corrélation et le rendre plus proche de 0.

Pour ce faire, on a pris 10000 couples des pixels adjacents aléatoirement choisi à partir de l'image originale et l'image cryptée suivant trois directions (horizontale, vertical et diagonale) pour chacun des trois canaux (rouge, vert, bleu).

Dans ce test, le calcul est appliqué en utilisant les équations suivantes :

$$r_{xy} = \frac{Cov(x,y)}{\sqrt{D_x}\sqrt{D_y}} \tag{3-4}$$

$$Cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E_x)(y_i - E_y) \tag{3-5}$$

$$D_x = \frac{1}{N}\sum_{i=1}^{N}(x_i - E_x)^2 \tag{3-6}$$

$$E_x = \frac{1}{N}\sum_{i=1}^{N}x_i \tag{3-7}$$

Avec N représente le nombre de pixels de l'image originale.

Après le calcul du coefficient de corrélation de deux pixels adjacents, on calcule le coefficient de corrélation (CC) pour chaque canal de l'image originale avec celui de l'image cryptée en appliquant la formule suivante :

$$CC = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}(A_{ij} - \bar{A})(B_{ij} - \bar{B})}{\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(A_{ij} - \bar{A})^2}\sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}(B_{ij} - \bar{B})^2}} \quad (3\text{-}8)$$

$$\bar{A} = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N} A_{ij} \quad (3\text{-}9)$$

$$\bar{A} = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N} A_{ij} \quad (3\text{-}10)$$

Avec A représente l'image originale, B représente l'image cryptée, $\bar{A}$ et $\bar{B}$ représentent respectivement la moyenne de toutes les valeurs des matrice A et B, M et N représentent respectivement la longueur et la largeur des matrices A et B.

4.2.3 Analyse des attaques différentielles

Parfois les attaqueurs font juste un petit changement au niveau d'un pixel de l'image originale pour voir l'effet de ce changement sur l'image cryptée, un bon système de cryptage est celui qui prend en compte ces changement-là, l'image cryptée après le changement doit être largement différente que celle avant le changement. Pour analyser la résistance de notre propre système contre ce type d'attaque, on a mesuré le nombre de pixels changés en calculant le NPCR (Taux de changement du nombre de pixel) et le UACI (Intensité moyenne changeante unifiée) en appliquant les formules suivantes :

$$NPCR = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N} D(i,j) \times 100\% \quad (3\text{-}11)$$

$$UACI = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N} \frac{|C_1(i,j) - C_2(i,j)|}{255} \times 100\% \quad (3\text{-}12)$$

$$D(i,j) = \begin{cases} 0, & if\ C_1(i,j) = C_2(i,j) \\ 1, & if\ C_1(i,j) \neq C_2(i,j) \end{cases} \quad (3\text{-}13)$$

Avec M et N représentent respectivement la longueur et la largeur de l'image cryptée, C_1 et C_2 sont les images cryptées avec une petite différence au niveau d'un pixel pour chaque canal (Rouge, vert et bleu) de l'image originale. Pour dire que notre propre système est résistant contre les attaques différentielles, il faut que les mesures de NPCR et UACI soient assez grandes.

Pour dire qu'une méthode est résistante contre les attaques différentielles, les valeurs de NPCR et UACI doivent être respectivement supérieurs à 99.6% et 33.4%, ces valeurs sont justifiées [121] par les équations (3-14) et (3-15).

$$NPCR_E = (1 - 2^{-n}) \times 100\% \quad (3\text{-}14)$$

$$UACI_E = \frac{1}{2^{2n}} \frac{\sum_{i=1}^{2^n-1} i(i+1)}{2^n - 1} \times 100\% \quad (3\text{-}15)$$

Avec n représente le nombre de bits nécessaire pour coder un pixel. Ce dernier est codé sur 8 bits, puisque le calcul est fait pour chaque canal (rouge, vert et bleu). Et par la suite, $NPCR_E = 99.6094\%$ et $UACI_E = 33.4635\%$.

La même chose pour la clé, si un petit changement au niveau de la clé va changer complètement l'image cryptée, on dit que le système de cryptage est sensible au changement de la clé.

4.2.4 Analyse de l'entropie

L'entropie est l'une des caractéristiques les plus remarquables pour mesurer le caractère aléatoire de l'algorithme de cryptage d'image. Pour calculer l'information de l'entropie, on utilise la formule suivante.

$$H(s) = \sum_{i=0}^{2^n-1} P(s_i) \log_2[P(s_i)] \quad (3\text{-}16)$$

Avec $P(s_i)$ est la probabilité d'occurrence pour chaque s_i, 2^n est le nombre total d'états de la source d'information. Pour une source d'information aléatoire parfaite ayant 2^n états, l'entropie devrait être égale à n.

Pour une vraie source aléatoire, $H(x)$ doit être égale à 8.

Un bon algorithme essaye d'approcher de cette valeur afin de simuler une vraie source aléatoire et éviter toute sorte de prédiction des nombres générés par cet algorithme.

4.2.5 Analyse de MSE et PSNR

Les deux derniers paramètres pour évaluer la fiabilité de notre algorithme sont le MSE (Mean Square Error) et le PSNR (Peak Signal to Noise Ratio). Le premier critère (MSE) donne la marge d'erreur entre l'image originale et l'image cryptée, et il est calculé en utilisant la formule suivante.

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (A_{ij} - B_{ij})^2 \quad (3\text{-}17)$$

Avec A représente l'image originale, B représente l'image cryptée, M et N représentent respectivement la longueur et la largeur des images A et B. La sécurité de l'image cryptée augmente avec l'augmentation de la valeur du MSE.

La qualité de l'image cryptée est évaluée en utilisant le deuxième critère (PSNR), ce dernier est calculé en utilisant la formule suivante.

$$PSNR = 10 \times \log_{10}\left(\frac{I_{Max}^2}{MSE}\right) \quad (3\text{-}18)$$

Avec I_{Max} représente la valeur maximale d'un pixel de l'image. La valeur du PSNR doit être petite, ce qui signifie une grande différence entre l'image originale et l'image cryptée.

L'objective de ces deux mesures c'est d'obtenir une grande valeur de MSE et une petite valeur de PSNR, ce qui signifie une large différence entre l'image originale et l'image cryptée.

4.3 Résultats d'expérimentation et discussions

4.3.1 Algorithme de chiffrement d'images couleurs rapide basé sur la génération de nombres aléatoires de deux matrices et les opérateurs de décalage de bits

4.3.1.1 Histogramme

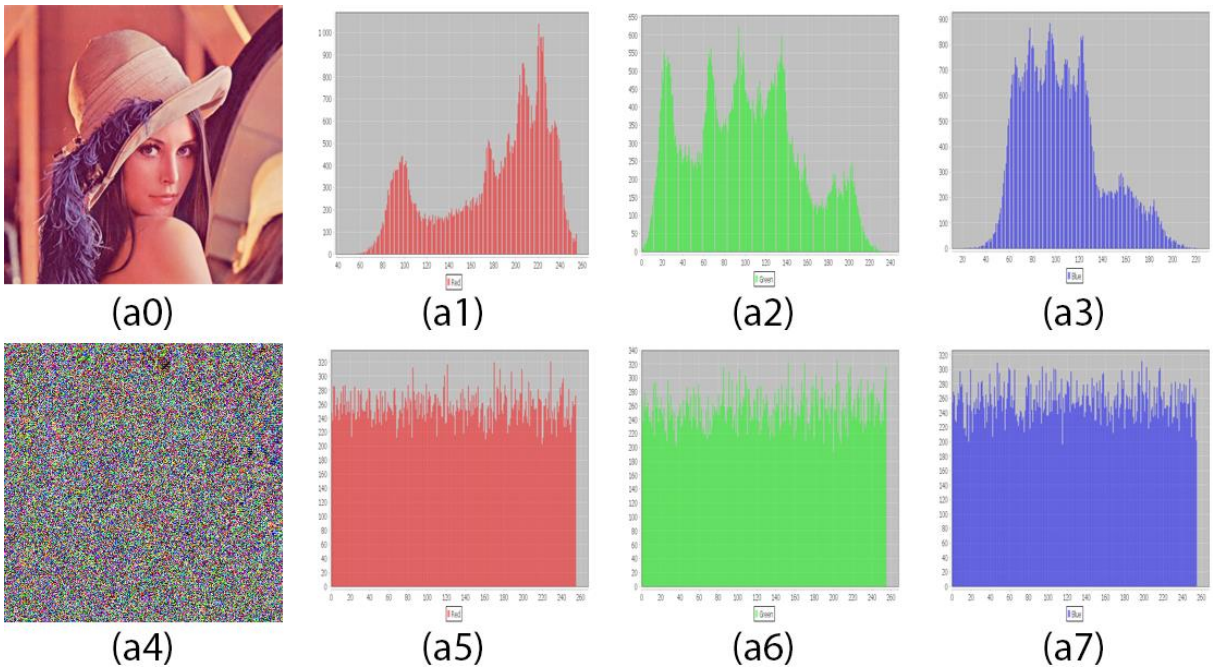


Figure 4-1 : (a0) *Lena image,* ***(a1)*** *Histogramme du canal rouge de Lena image,* ***(a2)*** *Histogramme du canal vert de Lena image,* ***(a3)*** *Histogramme du canal bleu de Lena image,* ***(a4)*** *Lena image cryptée,* ***(a5)*** *Histogramme du canal rouge de Lena image cryptée,* ***(a6)*** *Histogramme du canal vert de Lena image cryptée,* ***(a7)*** *Histogramme du canal bleu de Lena image cryptée.*

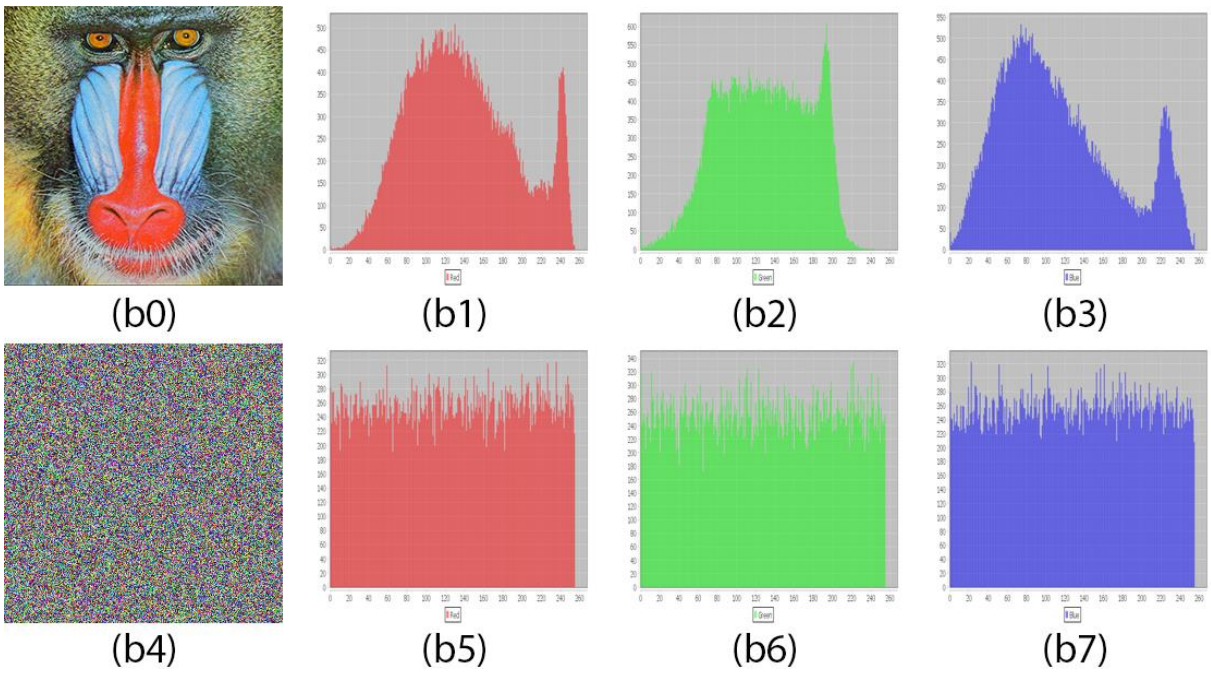


Figure 4-2 : (b0) *Baboon image,* ***(b1)*** *Histogramme du canal rouge de Baboon image,* ***(b2)*** *Histogramme du canal vert de Baboon image,* ***(b3)*** *Histogramme du canal bleu de Baboon image,* ***(b4)*** *Baboon image cryptée,* ***(b5)*** *Histogramme du canal rouge de Baboon image cryptée,* ***(b6)*** *Histogramme du canal vert de Baboon image cryptée,* ***(b7)*** *Histogramme du canal bleu de Baboon image cryptée.*

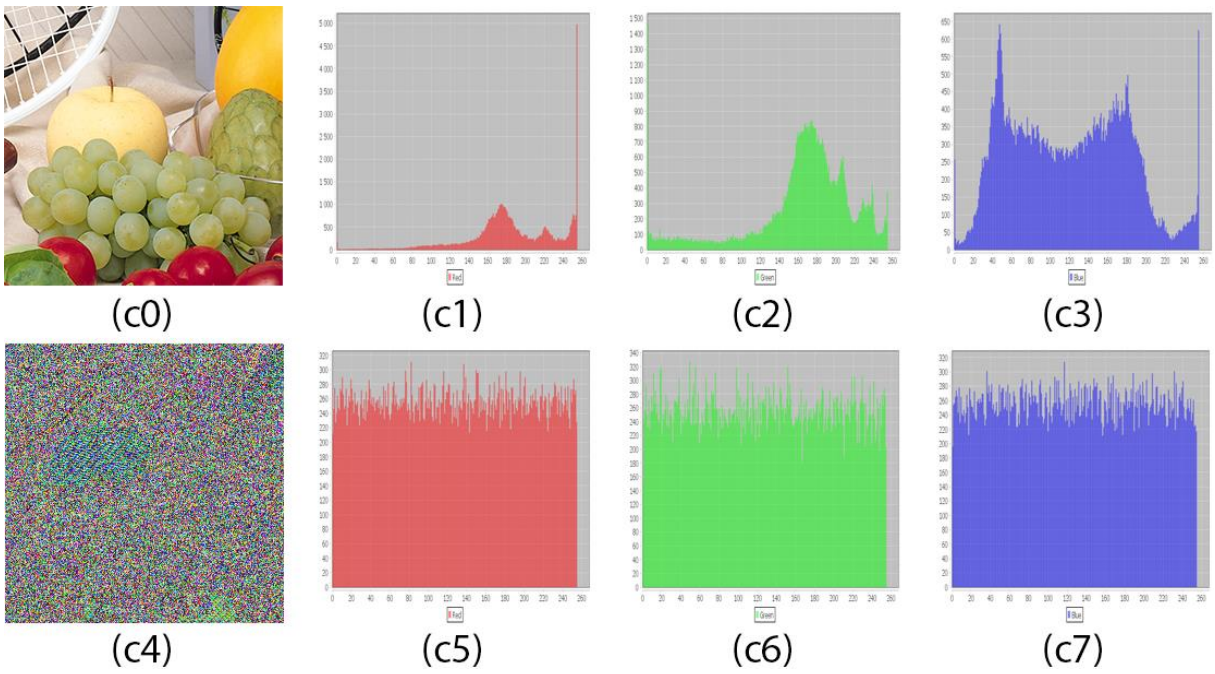


Figure 4-3 : (c0) *Fruits image,* ***(c1)*** *Histogramme du canal rouge de Fruits image,* ***(c2)*** *Histogramme du canal vert de Fruits image,* ***(c3)*** *Histogramme du canal bleu de Fruits image,* ***(c4)*** *Fruits image cryptée,* ***(c5)*** *Histogramme du canal rouge de Fruits image cryptée,* ***(c6)*** *Histogramme du canal vert de Fruits image cryptée,* ***(c7)*** *Histogramme du canal bleu de Fruits image cryptée.*

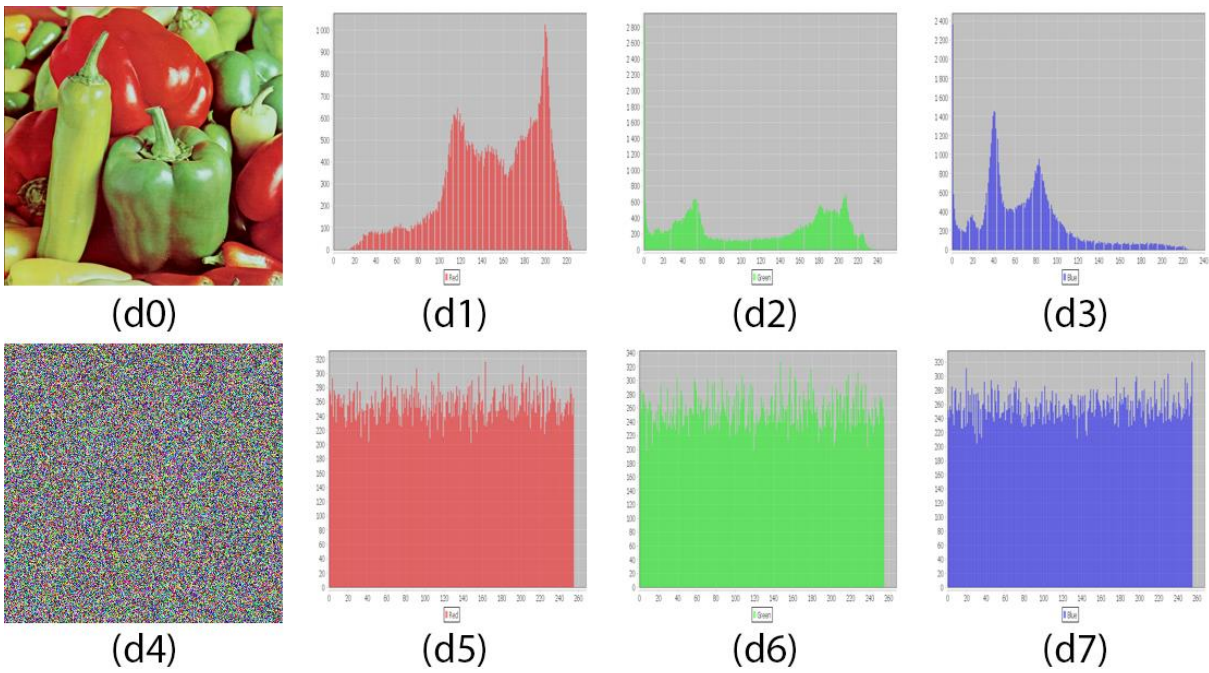


Figure 4-4 : (d0) *Peppers image,* ***(d1)*** *Histogramme du canal rouge de Peppers image,* ***(d2)*** *Histogramme du canal vert de Peppers image,* ***(d3)*** *Histogramme du canal bleu de Peppers image,* ***(d4)*** *Peppers image cryptée,* ***(d5)*** *Histogramme du canal rouge de Peppers image cryptée,* ***(d6)*** *Histogramme du canal vert de Peppers image cryptée,* ***(d7)*** *Histogramme du canal bleu de Peppers image cryptée.*

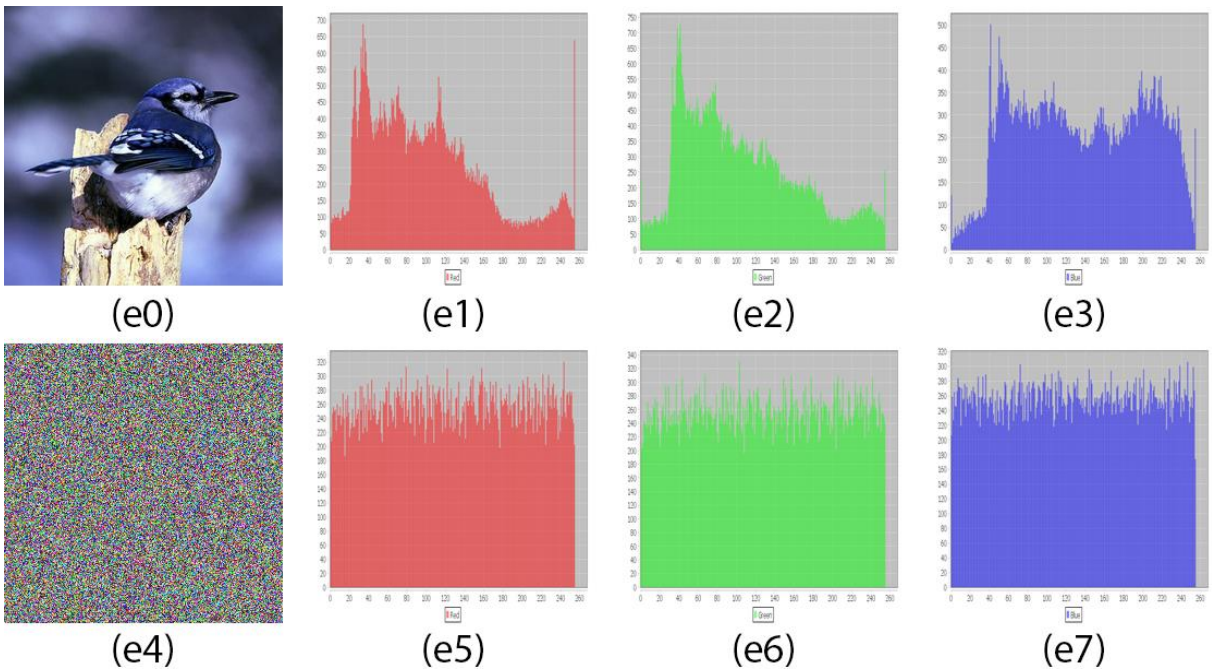


Figure 4-5 : (e0) *Bird image,* ***(e1)*** *Histogramme du canal rouge de Bird image,* ***(e2)*** *Histogramme du canal vert de Bird image,* ***(e3)*** *Histogramme du canal bleu de Bird image,* ***(e4)*** *Bird image cryptée,* ***(e5)*** *Histogramme du canal rouge de Bird image cryptée,* ***(e6)*** *Histogramme du canal vert de Bird image cryptée,* ***(e7)*** *Histogramme du canal bleu de Bird image cryptée.*

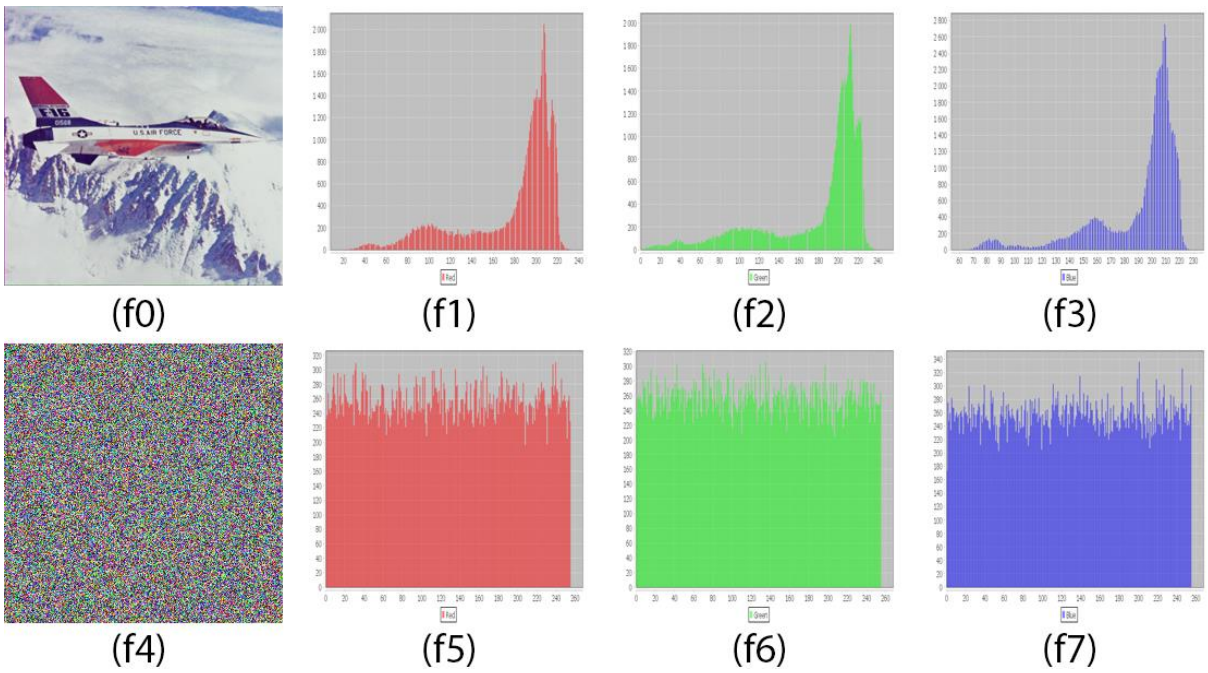


Figure 4-6 : (f0) *Airplane image,* ***(f1)*** *Histogramme du canal rouge de Airplane image,* ***(f2)*** *Histogramme du canal vert de Airplane image,* ***(f3)*** *Histogramme du canal bleu de Airplane image,* ***(f4)*** *Airplane image cryptée,* ***(f5)*** *Histogramme du canal rouge de Airplane image cryptée,* ***(f6)*** *Histogramme du canal vert de Airplane image cryptée,* ***(f7)*** *Histogramme du canal bleu de Airplane image cryptée.*

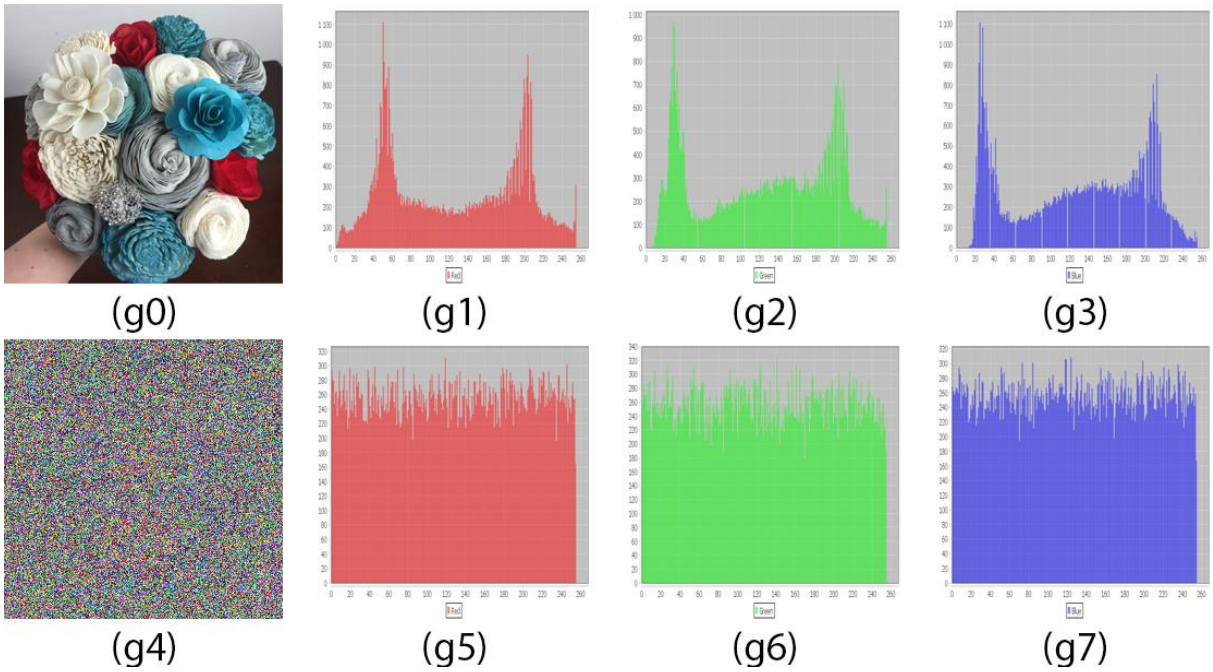


Figure 4-7 : (g0) *Flowers image,* ***(g1)*** *Histogramme du canal rouge de Flowers image,* ***(g2)*** *Histogramme du canal vert de Flowers image,* ***(g3)*** *Histogramme du canal bleu de Flowers image,* ***(g4)*** *Flowers image cryptée,* ***(g5)*** *Histogramme du canal rouge de Flowers image cryptée,* ***(g6)*** *Histogramme du canal vert de Flowers image cryptée,* ***(g7)*** *Histogramme du canal bleu de Flowers image cryptée.*

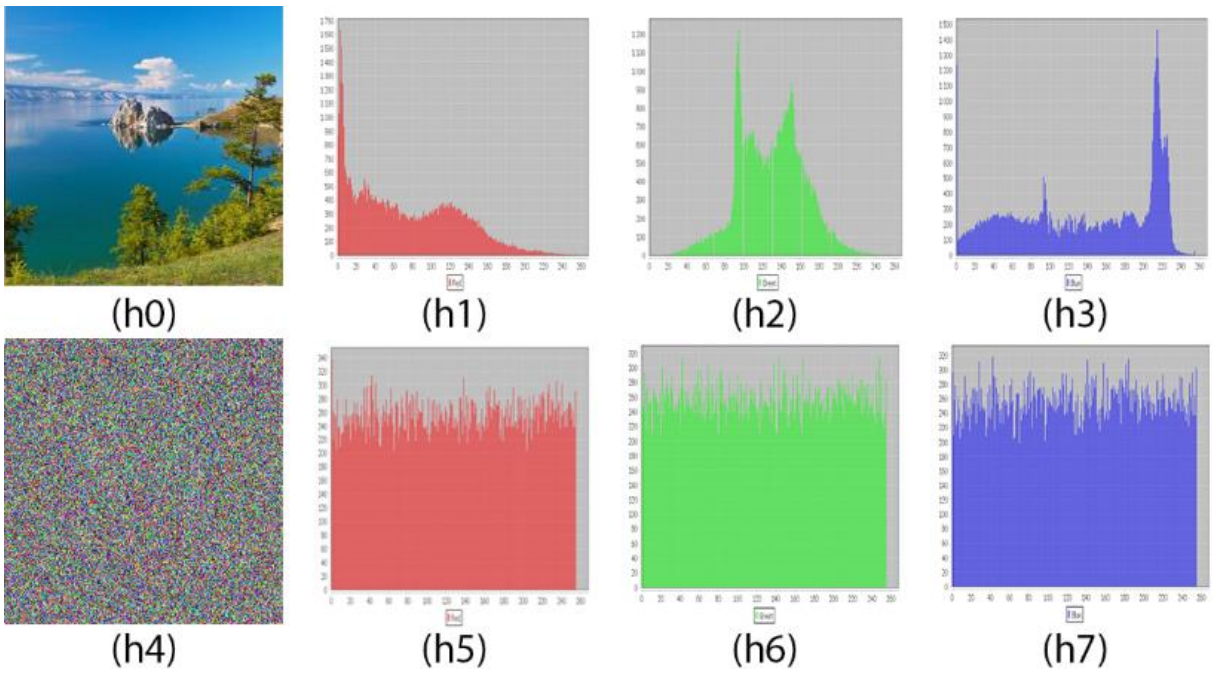


Figure 4-8 : (h0) *Lake image,* ***(h1)*** *Histogramme du canal rouge de Lake image,* ***(h2)*** *Histogramme du canal vert de Lake image,* ***(h3)*** *Histogramme du canal bleu de Lake image,* ***(h4)*** *Lake image cryptée,* ***(h5)*** *Histogramme du canal rouge de Lake image cryptée,* ***(h6)*** *Histogramme du canal vert de Lake image cryptée,* ***(h7)*** *Histogramme du canal bleu de Lake image cryptée.*

D'après les **figures 4-1, 4-2, 4-3, 4-4, 4-5, 4-6, 4-7 et 4-8**, on remarque que non seulement les histogrammes des images cryptées sont complètement différents de ceux des images originales, mais aussi la distribution des pixels est presque uniforme, ce qui nous ramène à dire que notre méthode peut résister contre les attaques statistiques et enfin permet d'augmenter plus la protection des informations incluses dans les images.

4.3.1.2 Analyse de corrélation

Le **tableau 4-2** ci-dessous montre les résultats obtenus après avoir calculer le coefficient de corrélation, les **figures 4-9, 4-10 et 4-11** montrent la distribution de la corrélation des couples des pixels adjacents dans les trois canaux (rouge, vert et bleu) et suivant les trois directions (horizontale, vertical et diagonale).

Tableau 4-2 : *Coefficient de corrélation des images originales et des images cryptées.*

Image	Canal	Image Originale			Image Cryptée		
		Horizontale	**Vertical**	**Diagonale**	**Horizontale**	**Vertical**	**Diagonale**
Lena	**Rouge**	0,975772	0,952779	0,918953	-0,02093	0,048895	0,017922
	Vert	0,970801	0,941692	0,935441	0,00813	0,010824	-0,012458
	Bleu	0,940875	0,913843	0,882293	-0,008816	0,017843	-0,034915
	Moyenne	0,962483	0,936105	0,912229	-0,007205	0,025854	-0,009817
Baboon	**Rouge**	0,867365	0,918216	0,853941	-0,024236	0,019584	0,017811
	Vert	0,778304	0,805673	0,704966	0,002682	-0,015062	-0,010772
	Bleu	0,860251	0,872245	0,805728	0,011979	-0,008435	0,021916
	Moyenne	0,835307	0,865378	0,788212	-0,003192	-0,001304	0,009651
Fruits	**Rouge**	0.934084	0.918770	0.877011	0.063041	0.038538	0.019305
	Vert	0.935958	0.955326	0.909279	0.009570	0.007922	-0.016128
	Bleu	0.944211	0.953427	0.912598	0.017280	0.016386	0.005465
	Moyenne	0.938084	0.942507	0.899629	0.029963	0.020948	0.002880
Peppers	**Rouge**	0.961869	0.962090	0.924277	0.008809	-0.002264	0.010336
	Vert	0.974738	0.968608	0.955928	-0.027486	0.037774	-0.007964
	Bleu	0.968433	0.952052	0.924415	0.018322	0.032341	-0.008736
	Moyenne	0.968346	0.960916	0.934873	-0.000118	0.022617	-0.002121

On remarque que le coefficient de corrélation (CC) dans les trois canaux (rouge, vert et bleu) et suivant les trois directions (horizontale, vertical et diagonale) prend des valeurs proches de 1 dans le cas des images originales. En revanche il prend des valeurs proches de 0 dans le cas des images cryptées, ce qui signifie que le degré de dépendance entre les deux pixels adjacents est très fort avant l'application de notre méthode, or cette dépendance est totalement détruite après avoir appliqué notre méthode de cryptage.

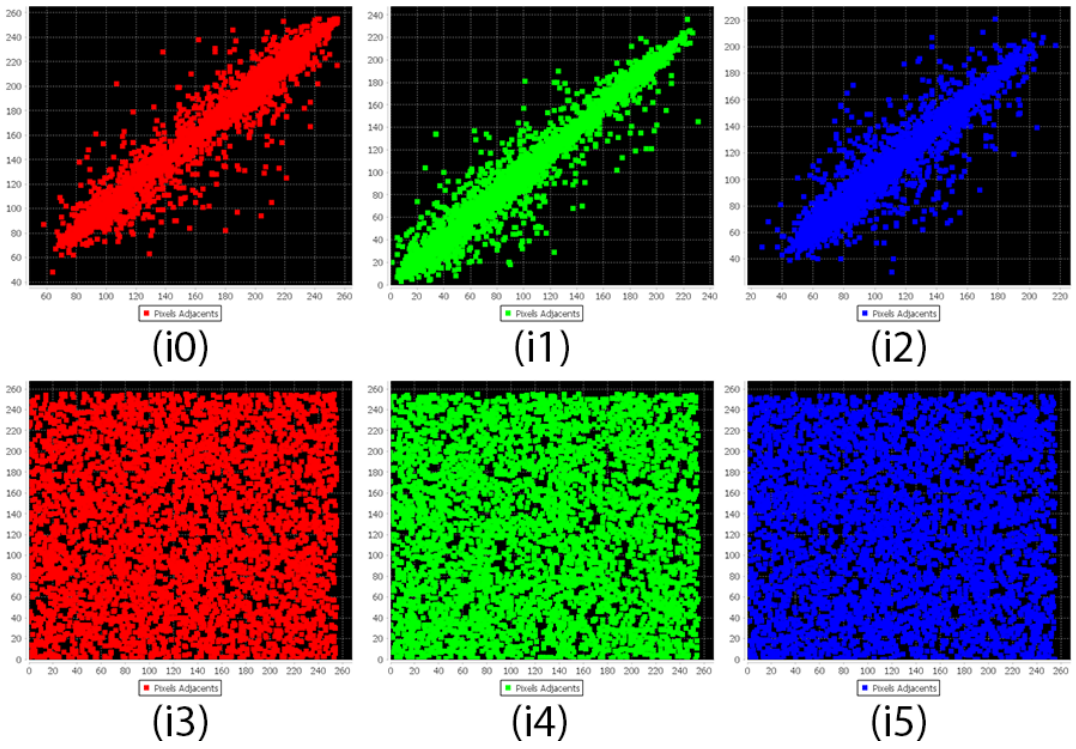


Figure 4-9 : *Corrélation de deux pixels adjacents suivant la direction horizontale : (i0) canal rouge de l'image originale, (i1) canal vert de l'image originale, (i2) canal bleu de l'image originale, (i3) canal rouge de l'image cryptée, (i4) canal vert de l'image cryptée, (i5) canal bleu de l'image cryptée.*

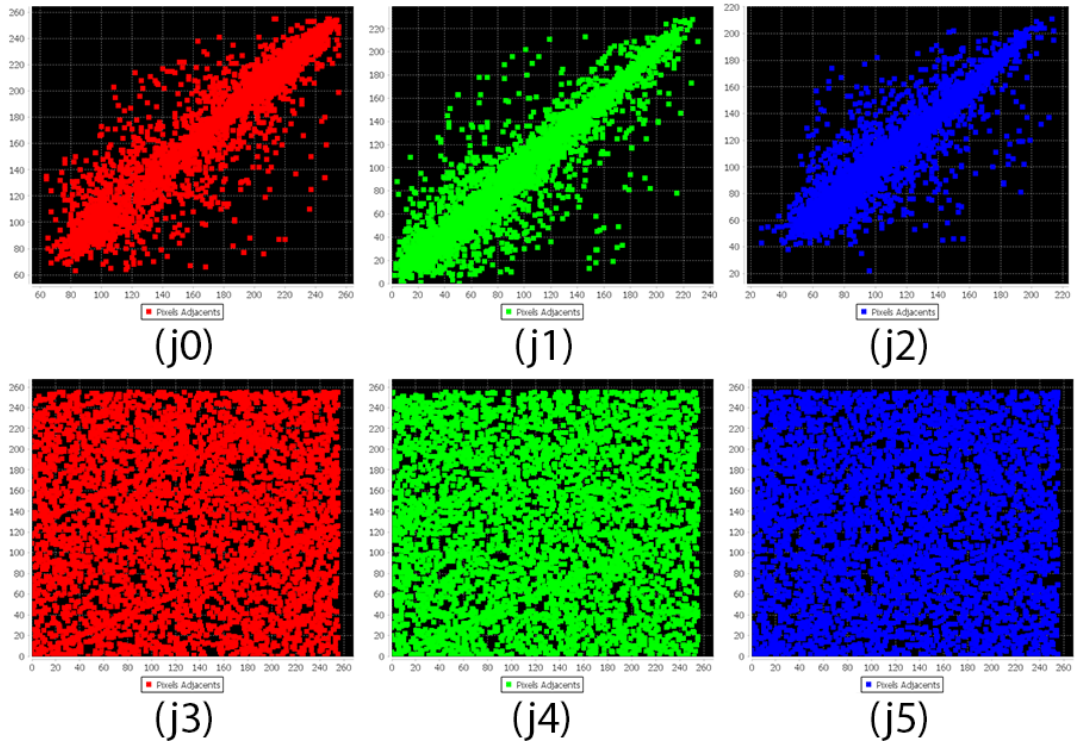


Figure 4-10 : *Corrélation de deux pixels adjacents suivant la direction verticale : (j0) canal rouge de l'image originale, (j1) canal vert de l'image originale, (j2) canal bleu de l'image originale, (j3) canal rouge de l'image cryptée, (j4) canal vert de l'image cryptée, (j5) canal bleu de l'image cryptée.*

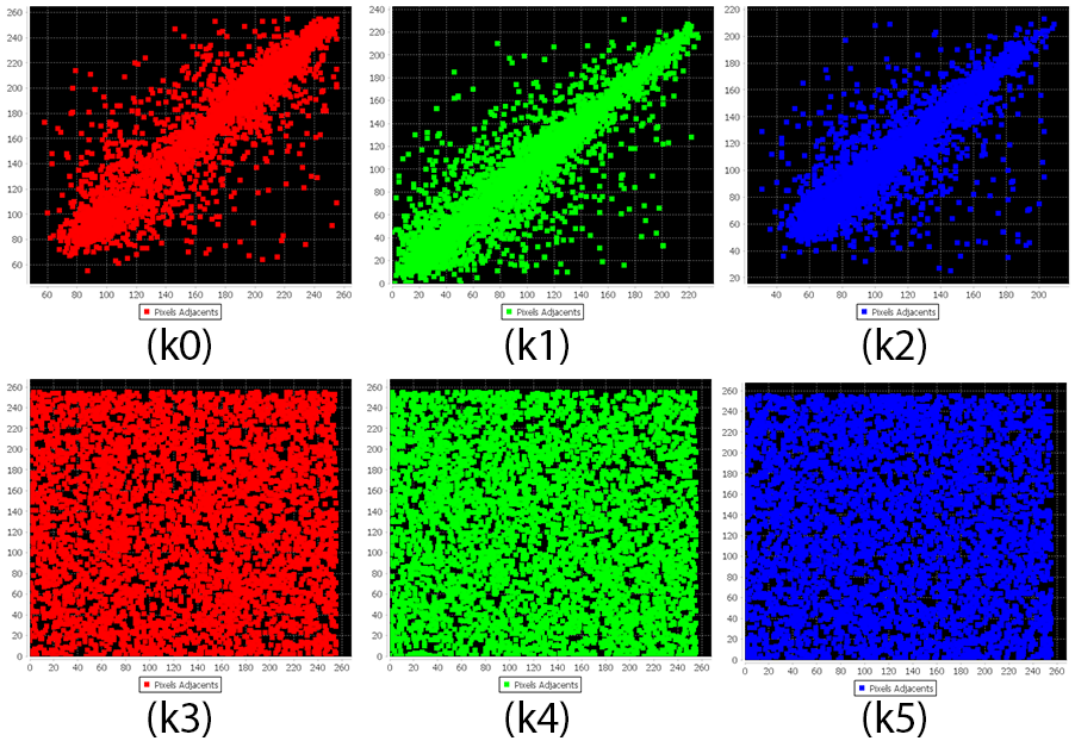


Figure 4-11 : *Corrélation de deux pixels adjacents suivant la direction diagonale : (k0) canal rouge de l'image originale, (k1) canal vert de l'image originale, (k2) canal bleu de l'image originale, (k3) canal rouge de l'image cryptée, (k4) canal vert de l'image cryptée, (k5) canal bleu de l'image cryptée.*

Les **figures 4-9, 4-10 et 4-11** montrent la distribution des pixels dans les trois canaux (rouge, vert et bleu) et suivant les trois directions (horizontale, vertical et diagonale) de l'image de Lena. On peut constater que la distribution avant l'application de notre méthode suit plus au moins une droite, ce qui montre que la corrélation est forte entre les pixels. En revanche, cette corrélation devient faible après avoir appliqué notre approche, ce qui signifie la méthode proposée donne des bons résultats. Les figures montrent que les pixels sont éparpillés par tout.

Tableau 4-3 : *Comparaison du CC de deux pixels adjacents de notre méthode avec d'autre approches.*

Image cryptée	Directions			Moyenne
	Horizontale	Vertical	Diagonale	
Tong [123]	0.017188	0.009852	0.033045	0.020028
Borujeni [124]	0.004100	0.030800	0.005300	0.013400
Wang [125]	0.006300	0.006200	0.006900	0.006466
Hua [126]	0.002383	0.008576	0.040242	0.017067
Tong [127]	0.003800	0.005800	0.013300	0.007633
Méthode Proposée	-0,007205	0,025854	-0,009817	0.002944

Le **tableau 4-3**, ci-dessus, représente les coefficients de corrélation (CC) obtenus par notre approche ainsi que par la méthode de Tong [123], la méthode de Borujeni [124], la méthode de Wang [125], la méthode de Hua [126] et celle de Tong [127].

Puisqu'on a travaillé sur trois canaux (rouge, vert et bleu), on a pris la moyenne pour chaque canal suivant chaque direction (horizontale, vertical et diagonale), ces valeurs sont calculées en appliquant notre méthode sur l'image Lena (**Figure 4-1**).

Les résultats montrent que la corrélation existante entre les pixels adjacents a été détruite par notre algorithme, ce qui montre la performance de notre approche par rapports aux autres méthodes.

Tableau 4-4 : *Comparaison de CC de notre méthode avec celle de ZHU.*

Image cryptée	Canal			Moyenne	ZHU [128]
	Rouge	Vert	Bleu		
Lena	0.007621	0.005257	-0.007645	0.001744	0.002851
Baboon	0.002012	0.007464	0.008313	0.005929	-0.006632
Fruits	0.001015	-0.006242	-0.010838	-0.005355	–
Peppers	0.014940	-0.020151	0.002337	-0.001332	-0.001650
Bird	0.004091	0.000611	0.012817	0.005839	–
Airplane	0.004597	0.007441	0.006604	0.006214	–
Flowers	-0.010705	0.012209	0.001583	0.001029	–
Lake	-0.009461	0.014982	-0.007487	-0.000655	–

Le tableau ci-dessus (**tableau 4-4**) montre les différentes valeurs obtenues après avoir calculé le coefficient de corrélation (CC) pour chaque canal (rouge, vert et bleu) en utilisant les images des **figures 4-1, 4-2, 4-3, 4-4, 4-5, 4-6, 4-7 et 4-8**, la 5ème colonne contient la valeur moyenne des 3 canaux pour chaque image, cette valeur est comparée par la suite avec la valeur obtenue par la méthode de Zhu [128].

Les résultats montrent la performance de notre méthode au niveau de la corrélation entre l'image originale et l'image cryptée puisque les valeurs sont très proches de 0, ces valeurs sont meilleures que celles obtenues par la méthode de Zhu [128].

4.3.1.3 Analyse des attaques différentielles

Le tableau ci-dessous montre les résultats obtenus après avoir calculer le NPCR et UACI tout en comparant ces résultats avec d'autres approches

Tableau 4-5 : *Sensibilité de l'algorithme pour un changement d'un seul pixel de l'image originale.*

Image	Canal	Méthode Proposée		Norouzi [129]		Tong [127]	
		NPCR	UACI	NPCR	UACI	NPCR	UACI
Lena	**Rouge**	99.662780	33.440426	–	–	99.571230	33.341510
	Vert	99.686486	33.523906	–	–	99.618720	33.352270
	Bleu	99.678039	33.830632	–	–	99.692230	33.358140
	Moyenne	99.675768	33.598321	99.668900	33.556100	99.627393	33.350640
Baboon	**Rouge**	99.653625	33.555644	–	–	–	–
	Vert	99.630737	33.736075	–	–	–	–
	Bleu	99.636159	33.577522	–	–	–	–
	Moyenne	99.640173	33.623080	99.638400	33.630500	–	–

On remarque que les résultats obtenus par notre méthode sont satisfaisants (**Tableau 4-5**) et meilleurs que ceux obtenus par Tong [127] et Norouzi [129], ce qui signifie que notre méthode est plus performante que celle de Tong et Norouzi.

On a pris l'image Lena de taille 256×256, et on l'a crypté avec les deux clés (Matrice1, Matrice2) de la **figure 4-1**. Ensuite, on a pris la première matrice et on a permuté les 2 derniers valeurs situées en bas à droite de notre matrice (2, 81), puis on a crypté l'image de Lena avec les nouvelles clés (Matrice3 qui est Matrice1 modifiée et Matrice2).

La figure ci-dessous montre le résultat de décryptage de l'image de Lena avec la clé initiale (Matrice1, Matrice2) et avec la clé (Matrice3, Matrice2). Après avoir effectué un petit changement au niveau d'un seul pixel de la matrice1, on remarque que l'opération de décryptage a échoué complètement.



Figure 4-12 : (l0) *Lena image,* ***(l1)*** *Cryptage de Lena Image avec la Clé 1 (Matrice1, Matrice2),* ***(l2)*** *Décryptage de Lena Image avec la Clé 1 (Matrice1, Matrice2),* ***(l3)*** *Décryptage de Lena Image avec la Clé 2 (Matrice3, Matrice2).*

4.3.1.4 Analyse de l'entropie

Le **tableau 4-6** ci-dessous montre les différentes valeurs d'entropie obtenues pour chacune des **figures 4-1, 4-2, 4-3 et 4-4**. Ces valeurs sont très proches de la valeur idéale 8, ce qui signifie que notre méthode est résistante contre les attaques d'entropie.

4.3.1.5 Analyse de MSE et PSNR

Les calculs sont faits pour les 4 images des **figures 4-1, 4-2, 4-3 et 4-4**. Le **tableau 4-6** représente les valeurs obtenues pour chaque canal (rouge, vert et bleu). D'après les résultats présentés dans ce tableau, on remarque que les valeurs du MSE sont très grandes. En revanche les valeurs du PSNR sont très petites. Donc on peut dire que notre approche donne des bons résultats, ce qui montre l'image cryptée est bien sécurisé.

Tableau 4-6 : *Paramètres de la qualité de chiffrement et de déchiffrement.*

Image	Canal	Cryptage			Décryptage		
		MSE	PSNR	Entropie	MSE	PSNR	CC
Lena	Rouge	10442.20	7.942880	7.997080	0	inf	1
	Vert	8756.56	8.034901	7.997886	0	inf	1
	Bleu	6878.10	8.513158	7.997364	0	inf	1
	Moyenne	8692.29	8.163647	7.997443	0	inf	1
Baboon	Rouge	8283.85	8.948480	7.997216	0	inf	1
	Vert	7556.56	9.347557	7.997747	0	inf	1
	Bleu	9134.69	8.523861	7.997340	0	inf	1
	Moyenne	8325.03	8.939966	7.997434	0	inf	1
Fruits	Rouge	10701.97	7.836163	7.999871	0	inf	1
	Vert	9559.38	8.326501	7.996846	0	inf	1
	Bleu	9029.87	8.573987	7.992466	0	inf	1
	Moyenne	9763.74	8.245550	7.996394	0	inf	1
Peppers	Rouge	7809.55	8.194256	7.999087	0	inf	1
	Vert	10642.07	7.583677	7.990610	0	inf	1
	Bleu	10490.70	6.988663	7.991868	0	inf	1
	Moyenne	9647.44	7.588865	7.993855	0	inf	1

Après avoir effectué l'opération de décryptage, on a refait les calculs pour évaluer la qualité du décryptage en calculant aussi le CC (Coefficient de Corrélation) pour chaque image et pour chaque canal (rouge, vert et bleu). Le MSE vaut 0, le CC vaut 1 et le PSNR tend vers l'infini (**tableau 4-6**), ce qui montre que l'image décryptée et l'image originale sont identiques. En effet, on peut conclure que notre méthode donne une grande précision.

4.3.2 Nouvelle méthode de cryptage d'image couleur 32 bits basée sur plusieurs cartes chaotiques de type 1D et la méthode de chiffrement à quatre carrés

4.3.2.1 Histogramme

Les **figures 4-13, 4-14 et 4-15** représente une comparaison de l'histogramme de l'image originale avec celui de l'image cryptée pour chacun des trois canaux (rouge, vert et bleu).

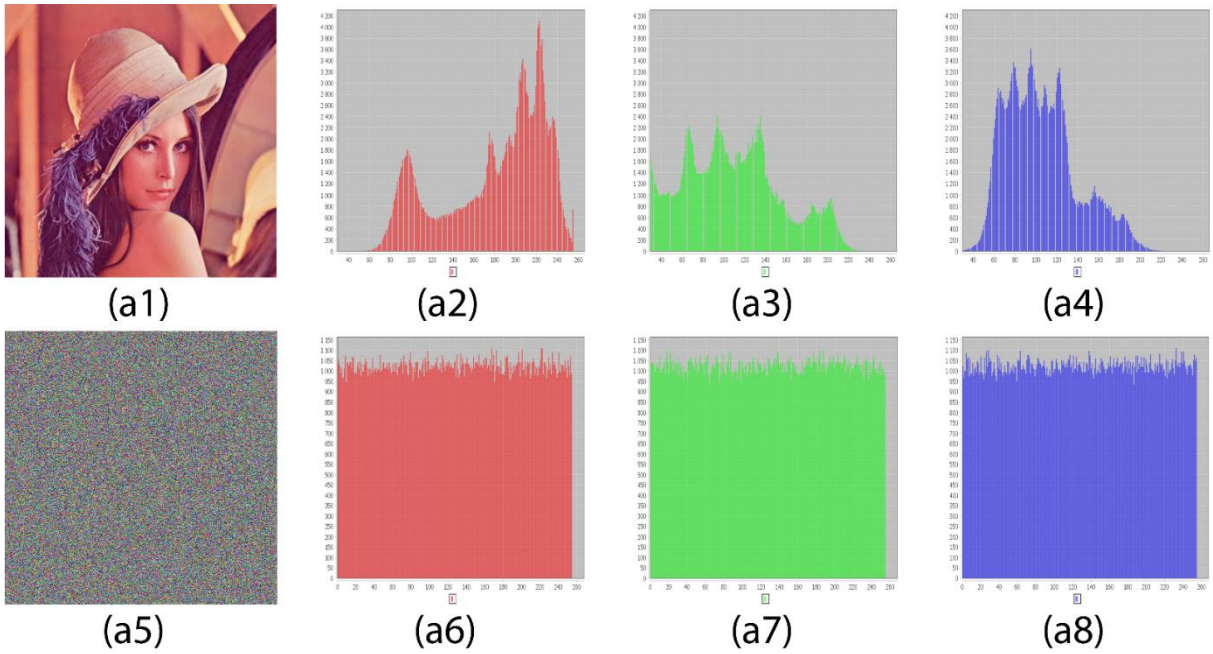


Figure 4-13 : (a1) *Lena image* ***(a2)*** *Histogramme rouge de Lena image* ***(a3)*** *Histogramme vert de Lena image* ***(a4)*** *Histogramme bleu de Lena image* ***(a5)*** *Lena image cryptée* ***(a6)*** *Histogramme rouge de Lena image cryptée* ***(a7)*** *Histogramme vert de Lena image cryptée* ***(a8)*** *Histogramme bleu de Lena image cryptée.*

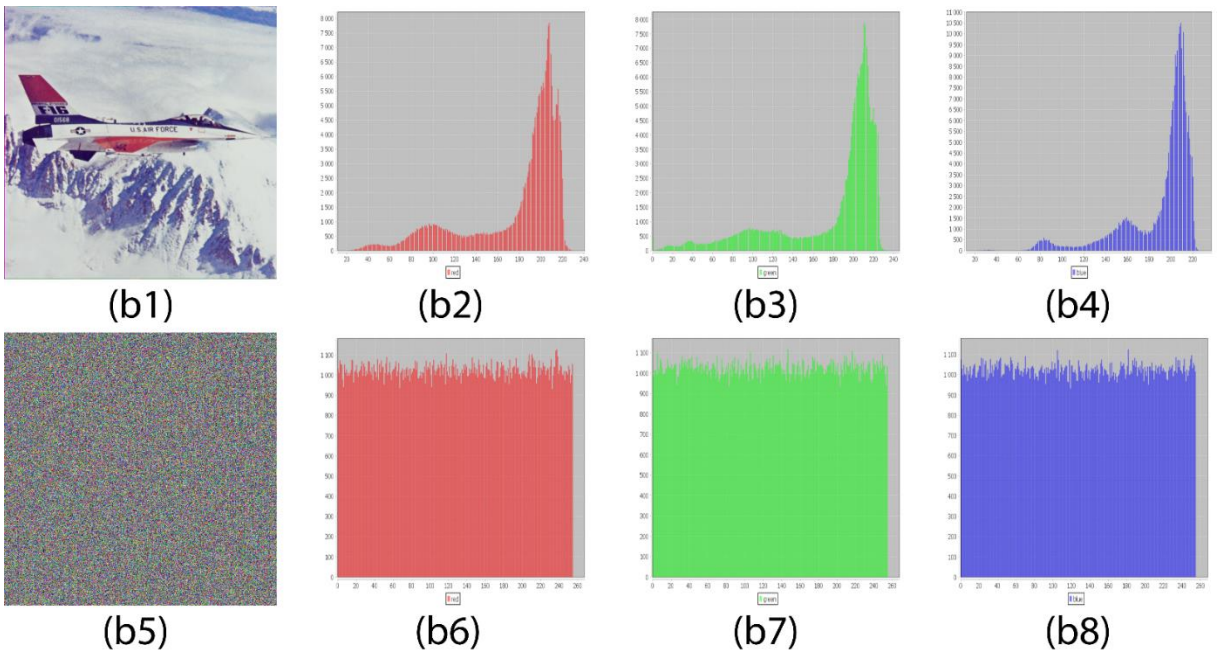


Figure 4-14 : (b1) *Airplane image* ***(b2)*** *Histogramme rouge de Airplane image* ***(b3)*** *Histogramme vert de Airplane image* ***(b4)*** *Histogramme bleu de Airplane image* ***(b5)*** *Airplane image cryptée* ***(b6)*** *Histogramme rouge de Airplane image cryptée* ***(b7)*** *Histogramme vert de Airplane image cryptée* ***(b8)*** *Histogramme bleu de Airplane image cryptée.*

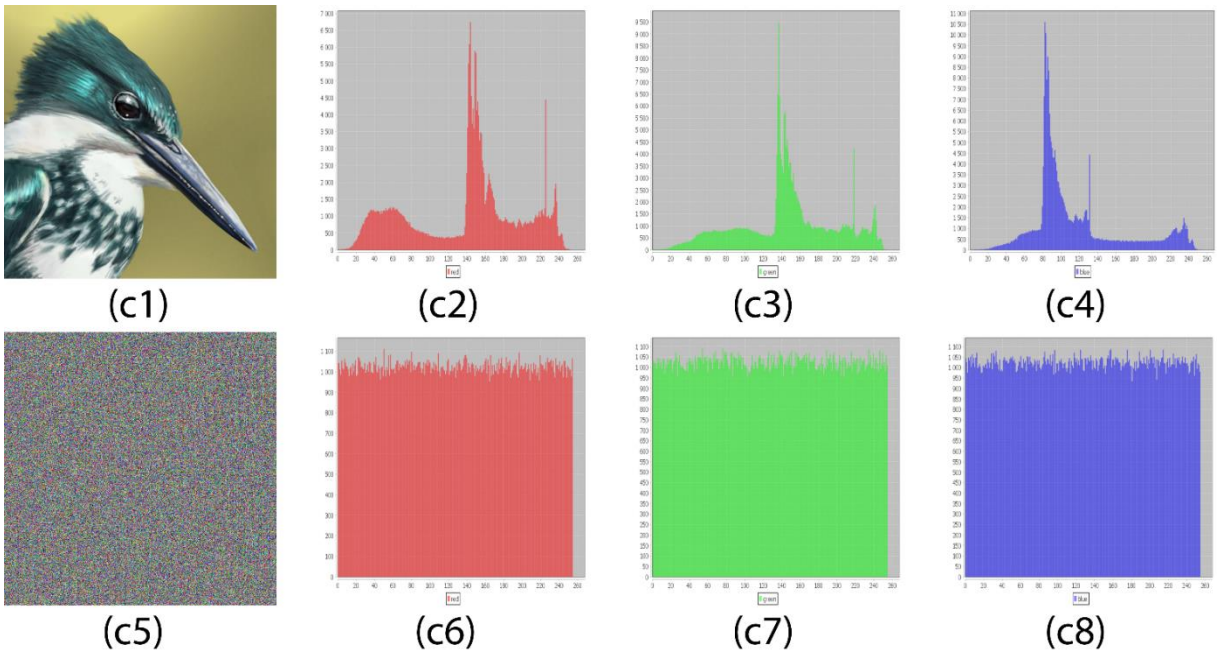


Figure 4-15 : (c1) *Bird image* ***(c2)*** *Histogramme rouge de Bird image* ***(c3)*** *Histogramme vert de Bird image* ***(c4)*** *Histogramme bleu de Bird image* ***(c5)*** *Bird image cryptée* ***(c6)*** *Histogramme rouge de Bird image cryptée* ***(c7)*** *Histogramme vert de Bird image cryptée* ***(c8)*** *Histogramme bleu de Bird image cryptée.*

D'après les **figures 4-13, 4-14 et 4-15**, on remarque que l'histogramme de l'image originale et celui de l'image cryptée sont complètement différents. De plus, il est presque uniforme, et

par conséquent, l'image cryptée ne va pas fournir aucune information sur les caractéristiques de l'image originale, ce qui va rendre les attaques statistiques plus difficiles.

4.3.2.2 Analyse de la corrélation

Les résultats du **tableau 4-7** montrent que le coefficient de corrélation calculé à partir des couples des pixels adjacents de l'image originale est très proche du 1, en revanche, après avoir appliqué notre système de cryptage. Le coefficient de corrélation a pris des valeurs très proches de 0.

Ceux-ci expliquent que notre système de cryptage a réussi à détruire la corrélation entre les pixels adjacents.

Tableau 4-7 : *Coefficient de corrélation entre les pixels adjacents de l'image originale et l'image cryptée.*

Image	**Canal**	**Image Originale**			**Image Cryptée**		
		Horizontale	**Vertical**	**Diagonale**	**Horizontale**	**Vertical**	**Diagonale**
Lena	Rouge	0,990055	0,98678	0,971122	0,001366	-0,001379	-0,007086
	Vert	0,991068	0,979745	0,966814	0,002016	0,013436	-0,006177
	Bleu	0,98135	0,972132	0,945108	-0,003958	-0,004956	-0,007068
	Moyenne	0,987491	0,979552	0,961015	-0,000192	0,002367	-0.006777
Airplane	Rouge	0,96564	0,977083	0,937893	0,009517	0,023228	-0,002749
	Vert	0,966987	0,944003	0,914516	0,006761	0,027268	0,025129
	Bleu	0,956584	0,027268	0,932854	0,009093	0,001257	-0,002553
	Moyenne	0,963071	0,962508	0,928421	0,008457	0,017251	0,006609
Bird	Rouge	0,988108	0,991376	0,982143	-0,000706	0,000539	0,003051
	Vert	0,986715	0,990349	0,973991	0,002541	0,039796	0,00961
	Bleu	0,986889	0,984337	0,981694	-0,013427	0,031884	0,022351
	Moyenne	0,987237	0,988687	0,979276	-0,003864	0,024073	0,011671

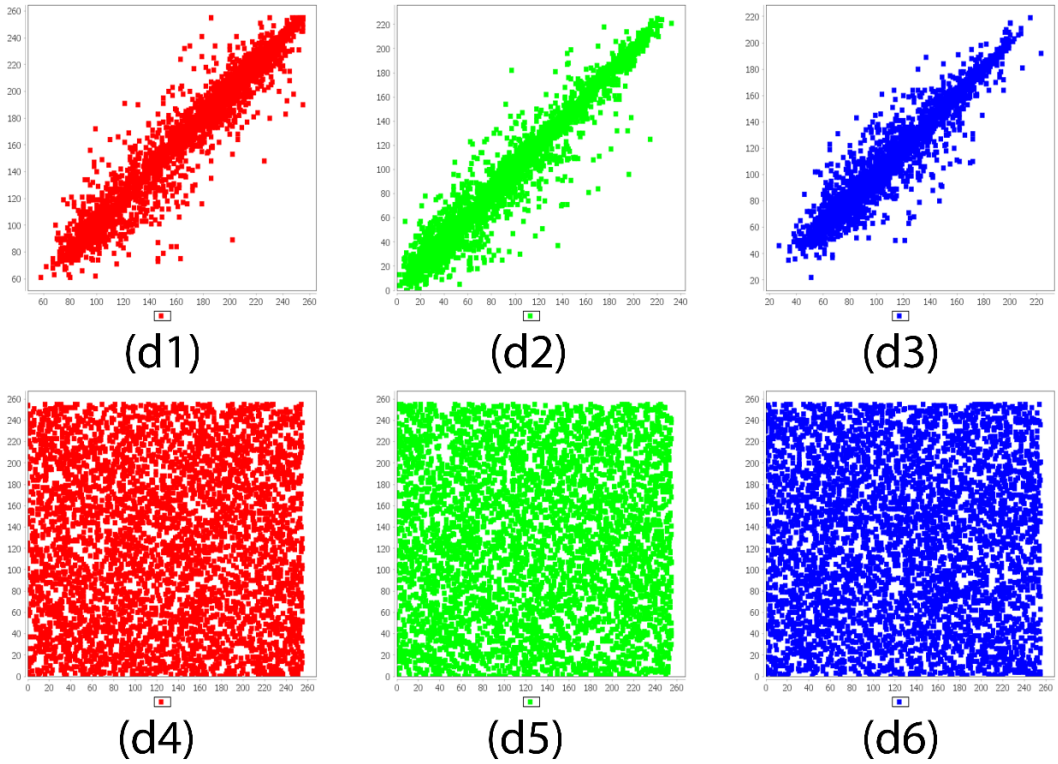


Figure 4-16 : *Distribution des pixels suivant la direction horizontale de Lena image : (d1) Canal rouge originale (d2) Canal vert originale (d3) Canal bleu originale (d4) Canal rouge cryptée (d5) Canal vert cryptée (d6) Canal bleu cryptée.*

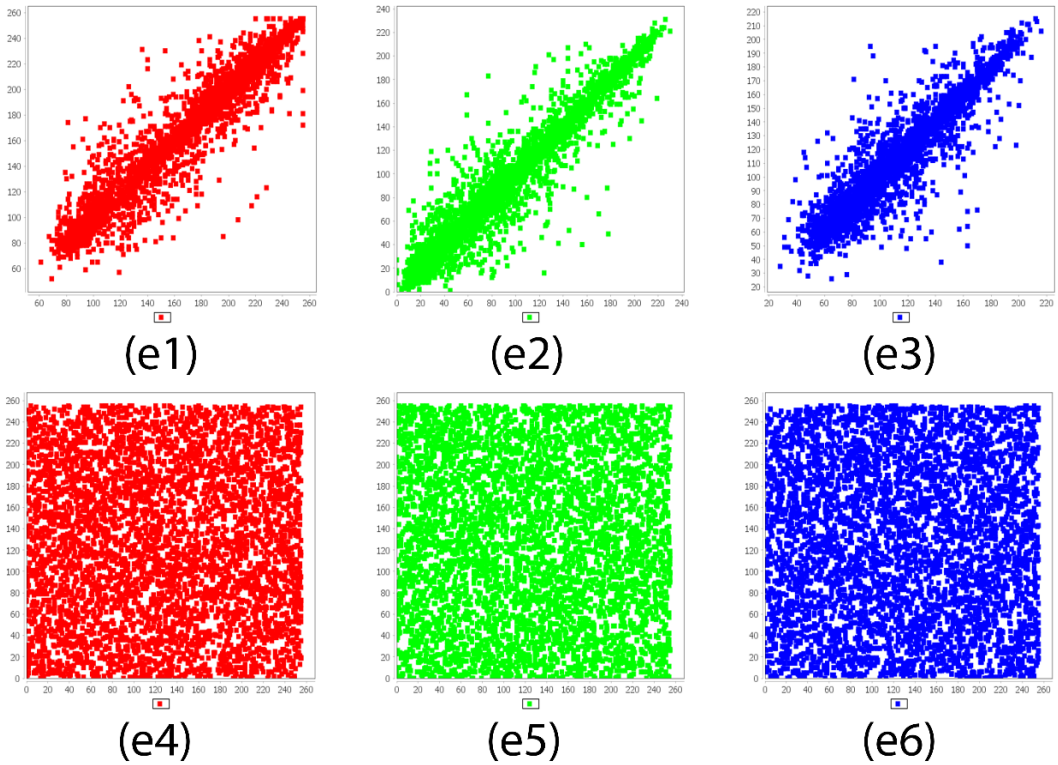


Figure 4-17 : *Distribution des pixels suivant la direction verticale de Lena image : (e1) Canal rouge originale (e2) Canal vert originale (e3) Canal bleu originale (e4) Canal rouge cryptée (e5) Canal vert cryptée (e6) Canal bleu cryptée.*

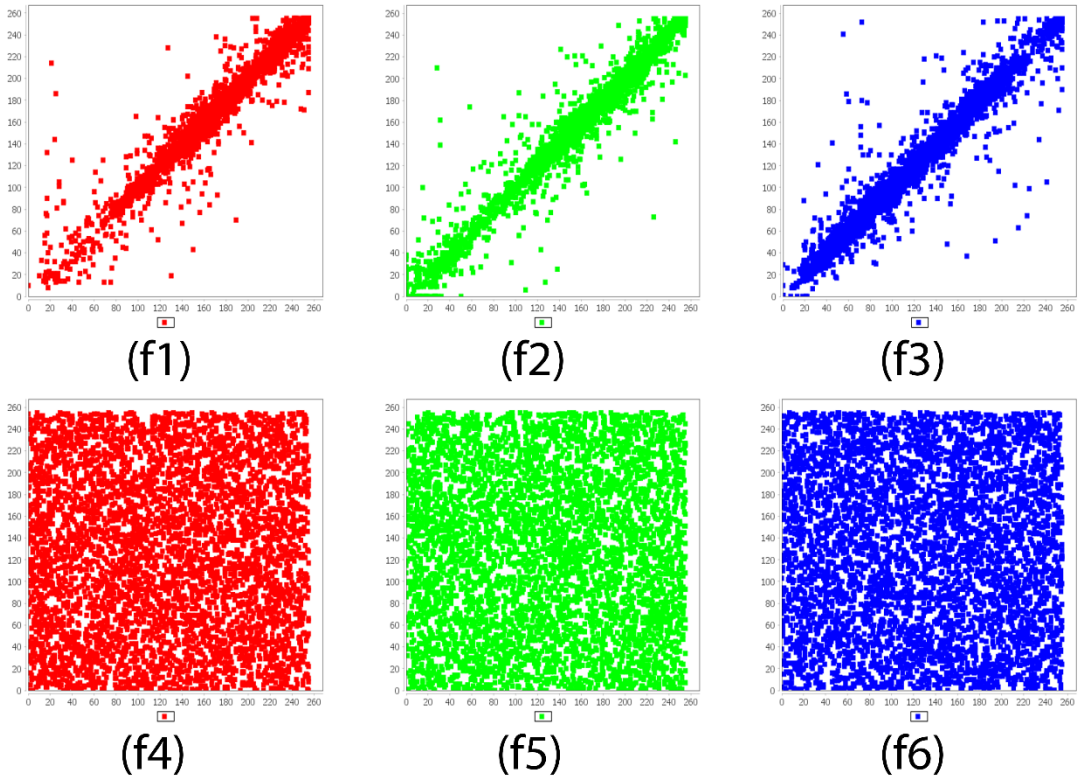


Figure 4-18: *Distribution des pixels suivant la direction diagonale de Lena image : (f1) Canal rouge originale (f2) Canal vert originale (f3) Canal bleu originale (f4) Canal rouge cryptée (f5) Canal vert cryptée (f6) Canal bleu cryptée.*

Les résultats du **tableau 4-7** montrent que le coefficient de corrélation calculé à partir des couples des pixels adjacents de l'image originale est très proche du 1, en revanche, après avoir appliqué notre système de cryptage. Le coefficient de corrélation a pris des valeurs très proche de 0.

Ceux-ci expliquent que notre système de cryptage a réussi à détruire la corrélation entre les pixels adjacents.

Les **figures 4-18, 4-19, et 4-20** montrent la distribution des pixels avant et après l'utilisation de notre algorithme pour chacun des trois canaux (rouge, vert et bleu) et suivant chacune des trois directions (horizontale, verticale et diagonale).

D'après ces figures, on peut constater que les pixels essaient de prendre une forme d'une ligne, ce qui est normal puisque le degré de dépendance entre ces pixels et très fort, mais après avoir appliqué notre algorithme, les pixels sont éparpillés partout.

Le **tableau 4-8** représente une comparaison du coefficient de corrélation des pixels adjacents obtenu par notre méthode avec celui obtenu avec d'autres méthodes concurrentes. Les résultats montrent que notre méthode performe plus que les autres.

Tableau 4-8 : *Comparaison du coefficient de corrélation des pixels adjacents avec d'autres méthodes.*

Méthode	**Directions**			**Moyenne**
	Horizontale	**Vertical**	**Diagonale**	
Es-Sabry et al. [130]	- 0.007205	0.025854	-0.009817	0.002944
Wang et al. [131]	0.0155	0.0013	0.0021	0.0063
Pak et al. [132]	-0.0038	-0.0026	0.0017	-0.00136
Xu et al. [133]	0.00529	-0.00674	-0.00674	0.00473
Kandar et al [30]	0.0009466	0.0008444	0. 0027413	0.00151076
Broumandnia et al (2DMCM) [31]	−0.0287	0.0217	0.0179	0.00363
Broumandnia et al (3DMCM) [31]	0.0269	0.0038	0.0094	0.013366
Méthode Proposée	-0,000192	0,002367	-0.006777	0.001534

Le **tableau 4-9** représente une comparaison entre notre méthode avec celle de Norouzi [129], Zhu [128] et Es-Sabry [130].

Tableau 4-9 : *Comparaison du coefficient de corrélation entre l'image originale et l'image cryptée avec d'autres méthodes.*

Image	Channel	Es-Sabry et al [130]	Norouzi et al [129]	Zhu el al [128]	Méthode Proposée
Lena	**Red**	0.007621	–	–	0.00043098
	Green	0.005257	–	–	0.00049824
	Blue	-0.007645	–	–	-0.00259657
	Average	0.001744	0.00111	0.002851	-0.00055578
Airplane	**Red**	0.004597	–	–	0.00117375
	Green	0.007441	–	–	0.00541818
	Blue	0.006604	–	–	-0.00098966
	Average	0.006214	–	–	0.00186742
Bird	**Red**	0.004597	–	–	-0.00069420
	Green	0.007441	–	–	-0.00122791
	Blue	0.006604	–	–	-0.00202655
	Average	0.006214	–	–	-0.00131622

Les valeurs obtenues après avoir calculer le coefficient de corrélation entre l'image originale et l'image cryptée, pour chacun des trois canaux (rouge, vert et bleu). Il est clair que notre méthode donne des résultats beaucoup meilleurs que ceux des autres.

4.3.2.3 Analyse des différentielles attaques

Le **tableau 4-10** montre les valeurs obtenues après avoir calculer les mesures UACI et NPCR, et les comparer avec celles d'autres méthode.

Tableau 4-10 : *Comparaison de NPCR et UACI avec d'autres méthodes.*

Image	Channel	Es-Sabry et al. [130]		Wang et al. [131]		Xu et al. [133]		Méthode Proposée	
		NPCR	**UACI**	**NPCR**	**UACI**	**NPCR**	**UACI**	**NPCR**	**UACI**
Lena	**Red**	99.662780	33.440426	–	–	–	–	99.696941	33.492887
	Green	99.686486	33.523906	–	–	–	–	99.627810	33.498154
	Blue	99.678039	33.830632	–	–	–	–	99.626826	33.468541
	Average	99.675768	33.598321	99.6009	33.4330	99.6181	33.4991	99.650525	33.486527
Airplane	**Red**	99.615242	33.455644	–	–	–	–	99.642191	33.426967
	Green	99.619550	33.436075	–	–	–	–	99.679756	33.456133
	Blue	99.601454	33.477522	–	–	–	–	99.656627	33.424502
	Average	99.610126	33.623080	–	–	–	–	99.659524	33.435867
Bird	**Red**	99.653625	33.555644	–	–	–	–	99.611183	33.490963
	Green	99.630737	33.736075	–	–	–	–	99.694384	33.414603
	Blue	99.636159	33.577522	–	–	–	–	99.637429	33.422080
	Average	99.640173	33.623080	–	–	–	–	99.647665	33.442548

Le **tableau 4-11** montre les résultats obtenus après avoir effectué un changement d'ordre 10^{-15} pour chacun des paramètres des fonctions chaotiques utilisées pour construire le cube de cryptage. Ces résultats montrent qu'un petit changement fait au niveau de la clé a causé un grand changement de plus 99.6%.

La **figure 4-19** montre l'opération de décryptage en utilisant une clé correcte :

$$\boldsymbol{\mu = 3.969480278735362}$$

$$x_0 = 0.910164784767923$$

$$r = 0.845933323162253$$

$$y_0 = 0.936427376755219$$

$$a = 2.898639715866417$$

$$z_0 = 0.984321261956951$$

et une clé fausse :

$$\boldsymbol{\mu = 3.969480278735363}$$

$$x_0 = 0.910164784767923$$

$$r = 0.845933323162253,$$

$$y_0 = 0.936427376755219$$

$$a = 2.898639715866417$$

$$z_0 = 0.984321261956951$$

Cette opération a échoué complètement même si le changement fait est très médiocre.

Tableau 4-11 : *Taux de changement en %.*

Paramètres	Changement %
$\boldsymbol{\mu}$	99.670269
$\boldsymbol{x_0}$	99.641428
$\boldsymbol{r}$	99.629901
$\boldsymbol{y_0}$	99.672460
$\boldsymbol{a}$	99.652617
$\boldsymbol{z_0}$	99.686578

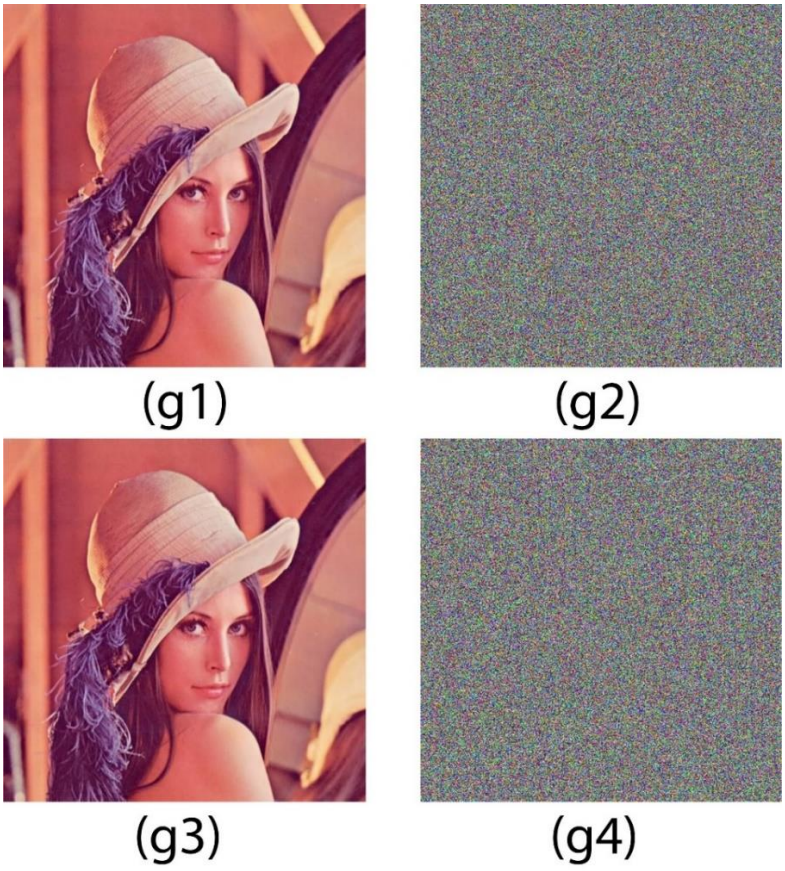


Figure 4-19 : *Sensibilité de la clé :* ***(g1)*** *Image Originale* ***(g2)*** *Image cryptée* ***(g3)*** *Image décryptée avec une clé correcte* ***(g4)*** *Image décryptée avec une clé fausse.*

4.3.2.4 Analyse de l'entropie

Le **tableau 4-12** montre les résultats obtenus après avoir calculer l'entropie de notre méthode pour chaque canal (rouge, vert et bleu) et pour différentes images ainsi qu'une comparaison est faite avec d'autres méthodes.

Tableau 4-12 : *Comparaison de l'entropie avec d'autres méthodes.*

Image	Channel	Es-Sabry et al. [130]	Saiyma et al. [134]	Shahna et al. [135]	Méthode Proposée
Lena	Red	7.997080	–	–	7.99915473
	Green	7.997886	–	–	7.99923016
	Blue	7.997364	–	–	7.99941321
	Average	7.997443	7.997407	7.99875443	7.99926603
Airplane	Red	–	–	–	7.99928567
	Green	–	–	–	7.99935600
	Blue	–	–	–	7.99930042
	Average	–	–	–	7.99931403
Bird	Red	–	–	–	7.99944480
	Green	–	–	–	7.99929115
	Blue	–	–	–	7.99939244
	Average	–	–	–	7.99937613

Les résultats du **tableau 4-12** sont très proche de la valeur idéale 8, et par conséquence, notre méthode peut résister les attaques d'entropie ainsi que ces valeurs sont meilleures que celles de Saiyma [134], Shahna [135] et Es-Sabry [130].

4.3.2.5 Analyse de MSE et PSNR

Le **tableau 4-13** montre les résultats obtenus des deux mesures qu'on a défini précédemment sur les images de la **figure 4-13, 4-14 et 4-15** pour chacun des trois canaux (rouge, vert et bleu) avant et après l'opération de cryptage.

Tableau 4-13 : *Analyse de MSE et PSNR.*

Image	Canal	Cryptage		Décryptage	
		MSE	**PSNR**	**MSE**	**PSNR**
Lena	**Rouge**	10633.40	7.864080	0	inf
	Vert	9010.86	8.092676	0	inf
	Bleu	7117.29	8.823970	0	inf
	Moyenne	8920.51	8.260242	0	inf
Airplane	**Rouge**	9902.67	7.277031	0	inf
	Vert	10660.84	7.106401	0	inf
	Bleu	10429.68	6.937806	0	inf
	Moyenne	10331.06	7.107079	0	inf
Bird	**Rouge**	9902.67	7.277031	0	inf
	Vert	10660.84	7.106401	0	inf
	Bleu	10429.68	6.937806	0	inf
	Moyenne	10331.06	7.107079	0	inf

4.3.3 Nouvelle méthode de cryptage d'images couleurs 32 bits basée sur plusieurs cartes chaotiques de type 1D et la méthode de chiffrement à quatre carrés

4.3.3.1 Histogramme

Les **figures 4-20 et 4-21** montrent une comparaison histogramme des images cryptées et originales pour chacun des quatre canaux (rouge, vert, bleu et alpha).

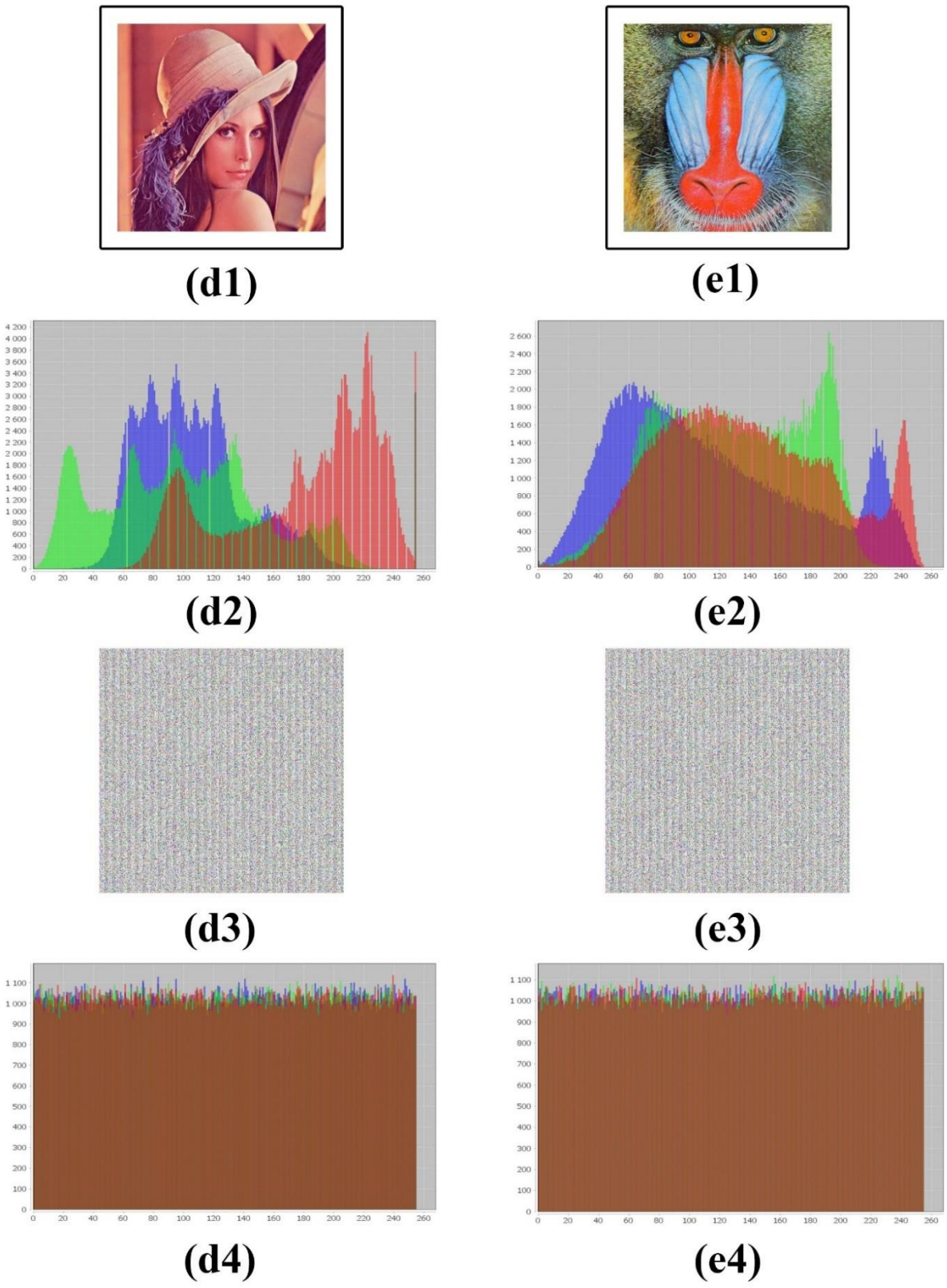


Figure 4-20 : (d1) *Image Lena* ***(d2)*** *Histogramme de l'image Lena* ***(d3)*** *Image Lena cryptée (****d4****) Histogramme de l'image Lena cryptée* ***(e1)*** *Image de babouin* ***(e2)*** *Histogramme de l'image de babouin* ***(e3)*** *Image de babouin cryptée* ***(e4)*** *Histogramme d'image de babouin cryptée.*

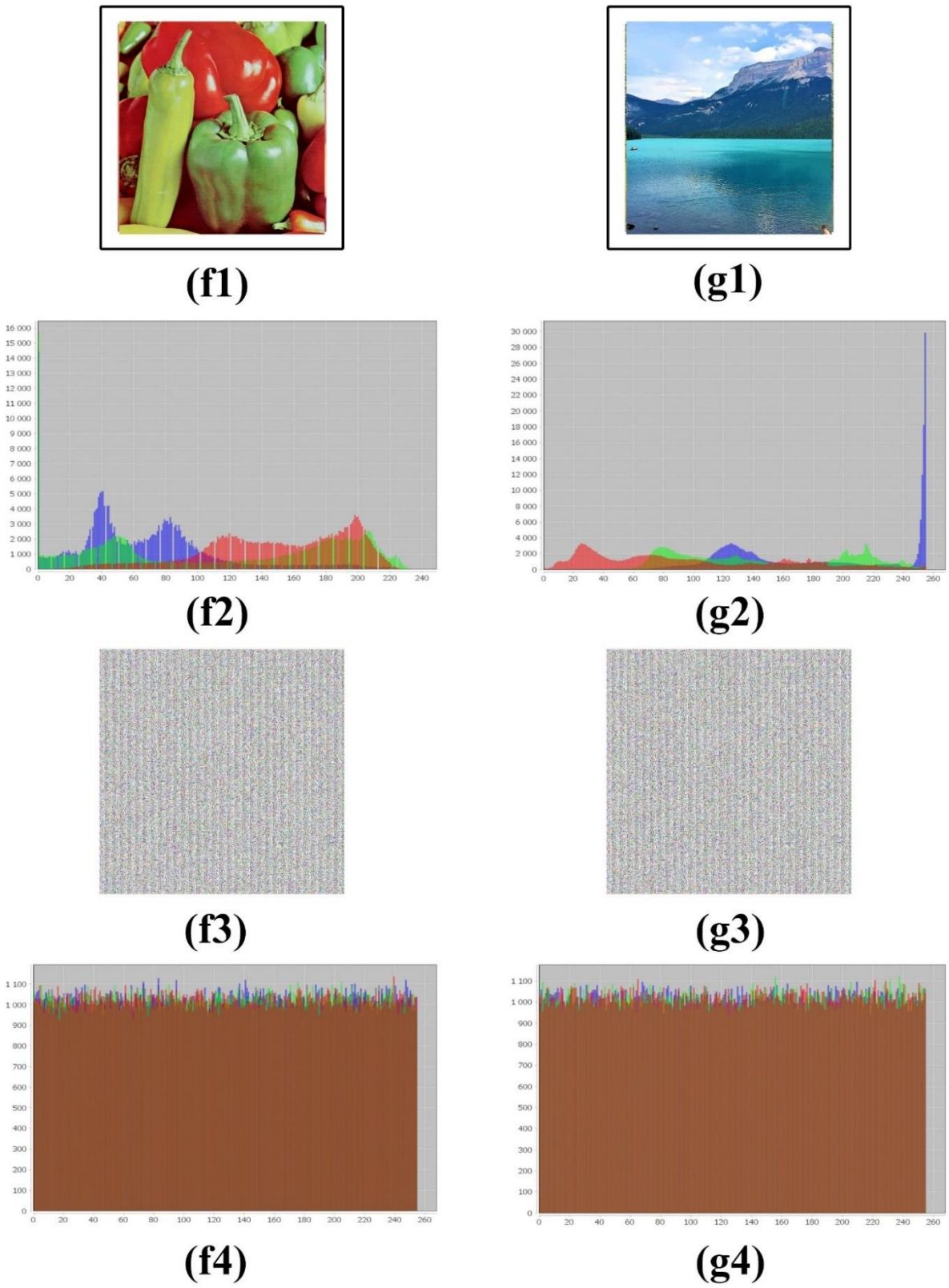


Figure 4-21 : (f1) *Image des poivrons* ***(f2)*** *Histogramme de l'image des poivrons* ***(f3)*** *Image des poivrons cryptés* ***(f4)*** *Histogramme de l'image des poivrons cryptés* ***(g1)*** *Image du lac* ***(g2)*** *Histogramme de l'image du lac* ***(g3)*** *Image cryptée du lac* ***(g4)*** *Histogramme de l'image du lac cryptée.*

Les histogrammes de l'image originale 32 bits et de l'image chiffrée sont radicalement différents, comme le montrent les **figures 4-20 et 4-21**. Il est presque uniforme, de sorte que l'image cryptée ne révélera aucune information sur les caractéristiques de l'image originale, ce qui rend les attaques statistiques plus difficiles.

4.3.3.2 Analyse de corrélation

Tableau 4-14 : *Coefficient de corrélation entre pixels adjacents du jeu de données expérimenté.*

Image	Canaux	Image originale			Image cryptée		
		H	**V**	**D**	**H**	**V**	**D**
Lena	**R**	0,982726	0,988578	0,972728	-0,001953	0,002179	-0,007736
	V	0,992783	0,972065	0,974490	0,001116	-0,018536	-0,002457
	B	0,977263	0,974761	0,980021	0,001957	0,004977	0,007068
	A	0,999847	0,990106	0,990928	0,002091	0,004011	-0,001457
	Moyenne	0,988154	0,981377	0,979541	0,000802	-0,001842	-0,000460
Baboon	**R**	0,998503	0,981491	0,982970	0,006732	0,028277	-0,003422
	V	0,976764	0,997874	0,997078	0,008296	0,022998	0,028470
	B	0,982565	0,988250	0,980910	0,000673	0,003189	-0,006057
	A	0,990623	0,990573	0,991362	0,000310	0,004890	0,008473
	Moyenne	0,987113	0,989547	0,988080	0,004002	0,014838	0,006866
Poivrons	**R**	0,986795	0,974727	0,976397	-0,008808	0,008786	0,007578
	V	0,976447	0,984192	0,977354	0,007325	0,032646	0,004817
	B	0,986065	0,981639	0,971148	-0,018612	0,036206	0,021651
	A	0,991175	0,995252	0,994414	0,008605	0,001602	0,008206
	Moyenne	0,985120	0,983952	0,979828	-0,002872	0,019810	0,010563

Le coefficient de corrélation obtenu à partir de pixels adjacents dans l'image originale 32 bits est assez proche de 1, comme le montre le **tableau 4-14**. Après avoir utilisé notre système de cryptage, le coefficient de corrélation était extrêmement proche de zéro. Ces résultats indiquent que notre technique de cryptage a détruit la corrélation entre les pixels adjacents.

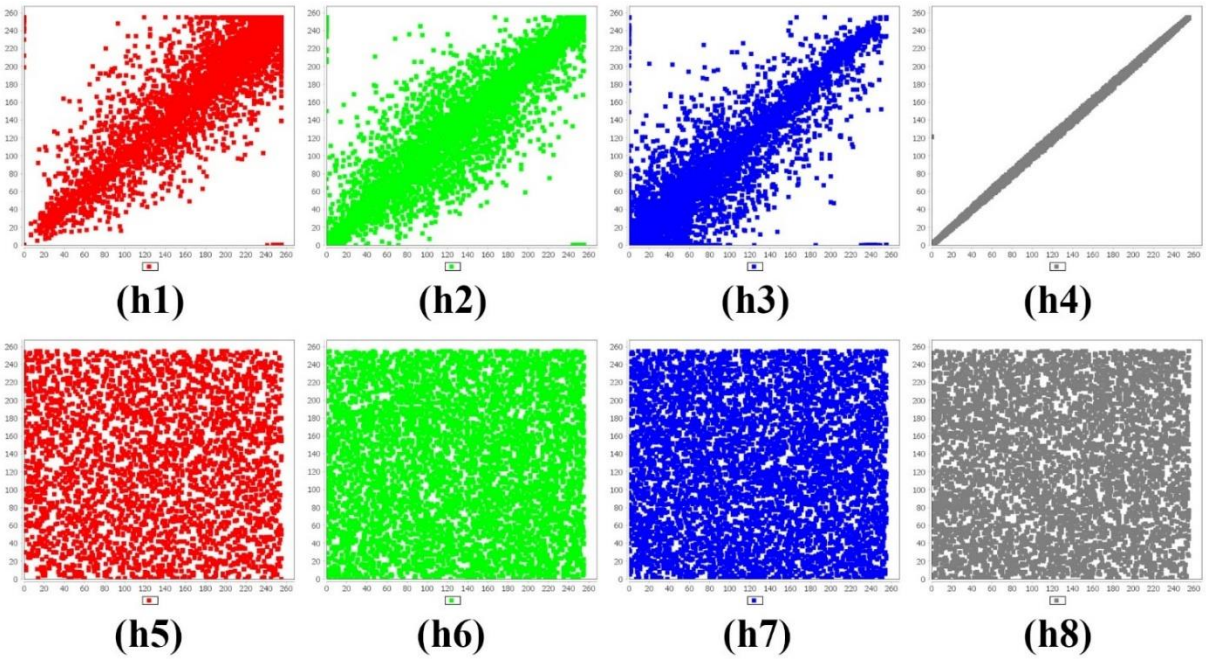


Figure 4-22 : *Distribution de la corrélation pour Lena Image (direction horizontale).*

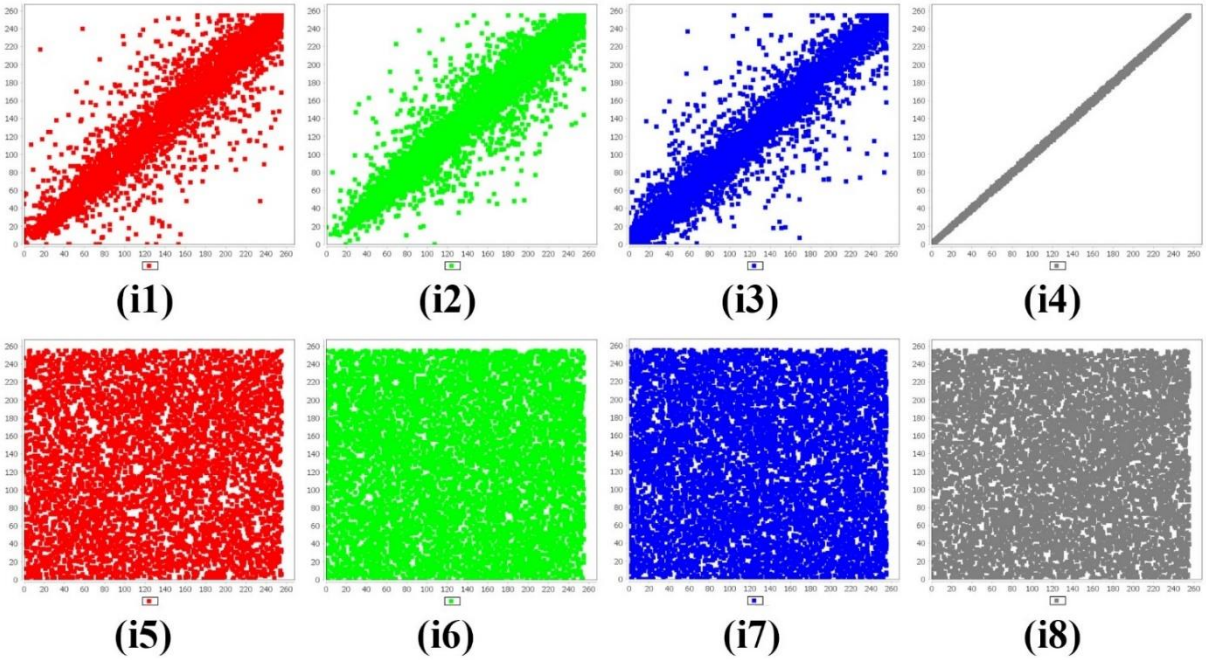


Figure 4-23 : *Distribution de la corrélation pour Lena Image (direction verticale).*

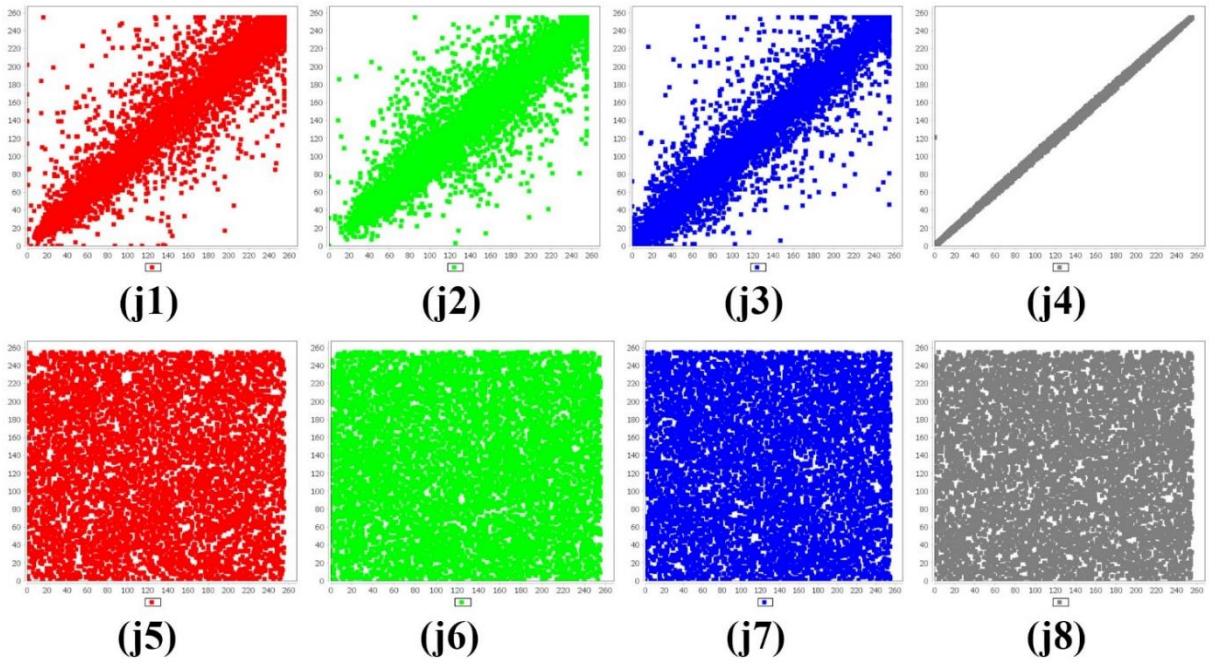


Figure 4-24 : *Distribution de la corrélation pour Lena Image (direction diagonale).*

Les **figures 4-22, 4-23 et 4-24** montrent la distribution des pixels avant et après avoir utilisé notre technique de chiffrement sur chacun des quatre canaux (rouge, vert, bleu et alpha) et suivi chacune des trois directions (horizontale, verticale et diagonale).

Ces chiffres montrent que les pixels tentent de former une ligne, ce qui est raisonnable compte tenu de la dépendance importante entre ces pixels. Cependant, après avoir utilisé notre méthode, les pixels sont dispersés partout.

Les coefficients de corrélation des pixels adjacents acquis par notre méthode sont comparés à ceux obtenus par des méthodes concurrentes dans le **tableau 4-15**. Notre Méthode surpasse les autres, selon les résultats obtenus.

Tableau 4-15 : *Résultats quantitatifs du coefficient de corrélation entre les pixels adjacents par rapport aux autres méthodes (l'image de Lena).*

Méthode	Directions			Moyenne
	H	V	D	
Es-Sabry et al. [130]	- 0.007205	0.025854	-0.009817	0.002944
Roy et al. [136]	-0.0009	-0.0017	-0.0019	-0,0015
Folifack Signing et al. [137]	-0.0031	0.0097	0.0005	0.002366
Benssalah et al. (ECCHC) [138]	0.016727	0,156755	0,009032	0.060838
Benssalah et al. (IECCHC) [138]	0,00014410	0,00052457	0,001594	0.00075422
Méthode Proposée	0,000802	-0,001842	-0,000460	0,0005

Le **tableau 4-16** compare notre approche à celles de Norouzi [129], Zhu [128] et Es-Sabry [130].

Tableau 4-16 : *Résultats quantitatifs du coefficient de corrélation entre l'image originale et cryptée par rapport à d'autres méthodes.*

Canaux	Es-Sabry et al [130]	Norouzi et al [129]	Zhu et al [128]	Méthode Proposée
R	0.007621	–	–	0,00071895
V	0.005257	–	–	0,00006067
B	-0.007645	–	–	-0,00294684
A	–	–	–	0,00003122
Moyenne	0.001744	0.00111	0.002851	-0.000534

Basant sur les résultats obtenus après le calcul du coefficient de corrélation entre l'image cryptée et l'image originale pour chacun des quatre canaux, nous pouvons conclure que notre méthode surpasse les autres.

4.3.3.3 Analyse des attaques différentielles

Les **tableaux 4-17 et 4-18** comparent les valeurs obtenues pour diverses images après le calcul des mesures UACI et NPCR à celles obtenues à l'aide d'autres approches.

Tableau 4-17 : *Valeurs NPCR et UACI de plusieurs images pour chaque canal.*

Images	Canaux	NPCR	UACI
Lena	**R**	99,642836	33,482122
	V	99,699538	33,495061
	B	99,608242	33,570241
	A	99,657437	33,550240
	Moyenne	99.652013	33.524416
Baboon	**R**	99.650336	33,487051
	V	99.653904	33.424790
	B	99.634745	33,449646
	A	99.699434	33,409734
	Moyenne	99.659604	33,442805
Poivrons	**R**	99.679909	33,554316
	V	99.613178	33,429954
	B	99.698881	33,469488
	A	99.646969	33,475464
	Moyenne	99,659734	33,482305

Tableau 4-18 : *Comparaison des valeurs UACI et NPCR pour l'image de Lena avec d'autres approches.*

Canaux	Es-Sabry et al. [130]		Wang et al. [131]		Xu et al. [133]		Méthode Proposée	
	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI
R	99.662780	33.440426	–	–	–	–	99,642836	33,482122
V	99.686486	33.523906	–	–	–	–	99,699538	33,495061
B	99.678039	33.830632	–	–	–	–	99,608242	33,570241
A	–	–	–	–	–	–	99,657437	33,550240
Moyenne	99.675768	33.598321	99.6009	33.4330	99.6181	33.4991	99.652013	33.524416

Basant sur ces comparaisons, l'approche suggérée surpasse nettement les méthodes concurrentes.

Les **tableaux 4-19 et 4-20** montrent respectivement les valeurs clés correctes et incorrectes, avec de légers changements dans l'ordre des 10^{-15} paramètres de la carte logistiqueα.

Le **tableau 4-21** montre les résultats obtenus après avoir modifié chacun des paramètres chaotiques de la carte utilisés par un ordre 10^{-15}. Même si la modification était mineure, l'opération de chiffrement a échoué.

Pour construire les quatre carrés du système de cryptage. Ces résultats montrent que de petites modifications des clés entraînent de grandes modifications de plus de 99.6%.

La **figure 4-25** illustre le résultat de la phase de déchiffrement à l'aide des clés correctes et incorrectes.

Tableau 4-19 : *Clé correcte.*

Carte logistique	Carte sinus	Carte de Chebyshev	Carte de la tente
$\alpha = 3.996283647041798$	$\beta = 0.830320676050502$	$\gamma = 2.865124029525049$	$\varepsilon = 0.795816275401810$
$x_0 = 0.900396583781812$	$y_0 = 0.912999005310166$	$z_0 = 0.91781678207741$	$t_0 = 0.969558093367123$

Tableau 4-20 : *Clé incorrecte.*

Carte logistique	Carte sinus	Carte de Chebyshev	Carte de la tente
$\alpha = 3.996283647041797$	$\beta = 0.830320676050502$	$\gamma = 2.865124029525049$	$\varepsilon = 0.795816275401810$
$x_0 = 0.900396583781812$	$y_0 = 0.912999005310166$	$z_0 = 0.91781678207741$	$t_0 = 0.969558093367123$

Tableau 4-21 : *Taux de variation en pourcentage.*

Paramètres	Taux %
α	99,697674
x_0	99,696383
β	99,692302
y_0	99,690884
γ	99,691966
z_0	99,697772
ε	99,696124
t_0	99,693163

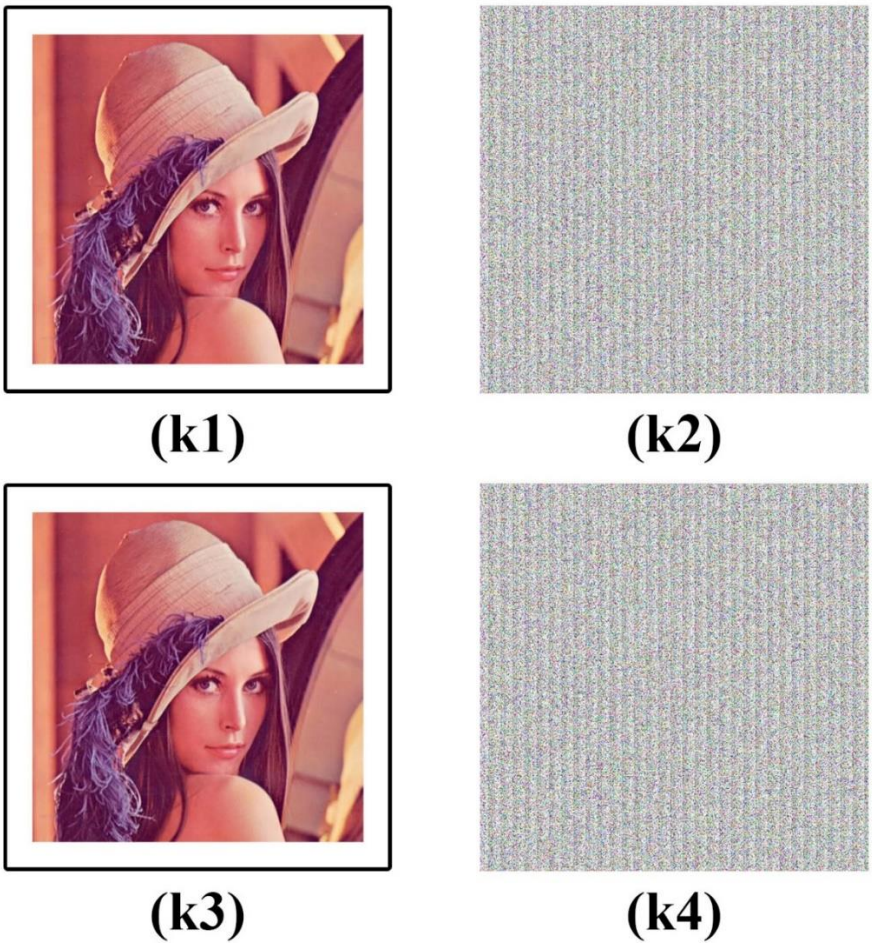


Figure 4-25 : *Sensibilité de la clé :* ***(k1)*** *Image originale* ***(k2)*** *Image cryptée* ***(k3)*** *Image décryptée avec la clé correcte* ***(k4)*** *Image décryptée avec la mauvaise clé.*

4.3.3.4 Analyse d'entropie

Les **tableaux 4-22 et 4-23** présentent les résultats du calcul de l'entropie de notre système de cryptage pour chaque canal (rouge, vert, bleu et alpha) de diverses images, ainsi qu'une comparaison avec des méthodes alternatives.

Tableau 4-22 : *Entropie de l'information de l'image originale et cryptée.*

Images	Canaux	Image originale	Image cryptée
Lena	**R**	7.35008052	7.99903665
	V	7.38568410	7.99948716
	B	7.31933801	7.99940240
	A	7.32826197	7.99968672
	Moyenne	7.34584115	7.99940323
Babouin	**R**	7.18265412	7.99895332
	G	7.16238290	7.99834511
	B	7.12126024	7.99816554
	A	7.13667185	7.99898805
	Moyenne	7.15074227	7.99861300
Poivrons	**R**	7.55445118	7.99818290
	V	7.68163139	7.99845365
	B	7.50999886	7.99844860
	A	7.54997277	7.99845424
	Moyenne	7.57401355	7.99838484

Tableau 4-23 : *Comparaison de la valeur d'entropie avec d'autres méthodes.*

Canaux	Raza et al. [134]	Es-Sabry et al. [130]	Es-Sabry et al. [139]	Shahna et al. [135]	Méthode Proposée
R	–	7.997080	7.99915473	–	7.99903665
V	–	7.997886	7.99923016	–	7.99948716
B	–	7.997364	7.99941321	–	7.99940240
A	–	–	–	–	7.99968672
Moyenne	7.997407	7.997443	7.99926603	7.99875443	7.99940323

Le **tableau 4-22** indique que les résultats sont assez proches de la valeur théorique 8, démontrant que notre approche est résistante aux attaques d'entropie. Comme le montre le **tableau 4-23**, ces valeurs surpassent celles de Raza [134], Es-Sabry [130], Shahna [139] et Es-Sabry [138].

4.3.3.5 Analyse de MSE et PSNR

Le **tableau 4-24** présente les résultats d'expérimentation des deux mesures définies précédemment sur les images des **figures 4-20 et 4-21** avant et après la phase de chiffrement pour chacun des quatre canaux (rouge, vert, bleu et alpha).

Tableau 4-24 : *Valeurs MSE et PSNR de diverses images.*

Images	Canaux	Cryptage		Décryptage	
		MSE	**PSNR**	**MSE**	**PSNR**
Lena	**R**	10728.60	7.825371	0	Inf
	V	9221.13	7.106401	0	Inf
	B	7285.10	6.937806	0	Inf
	A	21710.51	4.764102	0	Inf
	Moyenne	12236.33	7.644719	0	Inf
Baboon	**R**	17169.89	5.783127	0	Inf
	V	14517.01	6.512030	0	Inf
	B	15724.18	6.165120	0	Inf
	A	20815.31	4.946972	0	Inf
	Moyenne	17056.60	5.851812	0	Inf
Poivrons	**R**	15897.77	6.117438	0	Inf
	V	12594.69	7.128927	0	Inf
	B	14131.36	6.628961	0	Inf
	A	20719.77	4.966953	0	Inf
	Moyenne	15835.90	6.210570	0	Inf

4.4 Discussions

Sur la base des expériences et des simulations effectuées dans la section précédente, nous pouvons dire que nos contributions donnent des résultats satisfaisants par rapport aux méthodes que nous comparons. De plus, si nous comparons les trois méthodes que nous avons développées, nous pouvons conclure :

- ✓ La première contribution se base sur la génération des entiers uniques aléatoirement sans préciser la source de génération, et par suite, la clé sera les valeurs des matrices elles-mêmes, ce qui rend l'espace de clé est très grand ($2 \times 256!$). Le seul problème qu'on peut trouver à propos de cette méthode, c'est qu'il fait trois fois le parcours d'une image couleur et fusionne le résultat pour enfin donner une image couleur crypter, ce qui rend cette opération couteuse en terme du temps et des ressources.

- ✓ La deuxième contribution consiste à construire un cryptosystème basé sur les trois faces adjacentes d'un cube représentant les trois canaux d'une image couleur (rouge, vert, et bleu), et qui sont rempli à l'aide de trois cartes chaotiques (logistique, Sinus, et Chebyshev), l'opération de cryptage se fait à l'aide de la méthode d'intersection des plans à l'intérieur du cube. L'avantage de cet algorithme par rapport à notre première contribution, c'est qu'il crypte toutes les valeurs des matrices des trois canaux en même temps, ce qui permet d'optimiser les ressources ainsi que le temps d'exécution.
- ✓ La troisième contribution est une amélioration de notre première contribution, cet algorithme est construit à partir de quatre matrices qui représentent les quatre canaux d'une image couleur de 32 bits (rouge, vert, bleu, et alpha).

4.5 Conclusion

Dans ce chapitre, nous avons présenté les résultats d'expérimentation de nos trois contributions pour la cryptographie des images couleurs (24 bits, 32 bits), après avoir analysé la performance de nos méthodes contre les différentes attaques statistiques et différentielles en se basant sur plusieurs critères tels que l'histogramme, le coefficient de corrélation, l'entropie, le NPCR, le UACI et le PSNR. Les résultats ont été comparés avec ceux des autres méthodes de la littérature, et ils ont été compétitives, voire meilleurs que d'autres méthodes, ce qui nous permet de conclure que nos algorithmes sont robustes et très sécurisés.

Conclusion générale et perspectives

Conclusion générale

Au cours de cette thèse, nous avons proposé des nouvelles méthodes pour la cryptographie des images couleurs de type 24 bits ou bien 32 bits. Dans un premier temps, nous avons étudié les principes de la cryptographie et nous avons présenté les différentes techniques classiques et modernes. Ensuite, nous avons traité leurs applications limitées au chiffrement d'images et pourquoi la majorité des algorithmes utilisés dans la littérature, et qui sont prouvés leurs robustesses et leurs niveaux de sécurité très élevé, ne sont pas adéquats au chiffrement des images. Par conséquent, nous avons présenté nos méthodes et algorithmes pour le chiffrement des images couleurs en tenant compte des caractéristiques des images (taille, corrélation entre les pixels, etc…) :

La première contribution est basée sur l'utilisation de deux matrices de taille 16×16 avec des valeurs entières comprises entre 0 et 255, générées aléatoirement avec rotation cyclique de bits. Ces matrices sont utilisées pour effectuer la première étape de chiffrement. La première valeur de la première matrice est calculée à partir des pixels de chaque canal de l'image d'origine et le reste des valeurs est généré aléatoirement. La deuxième matrice a des valeurs différentes et elle est générée aléatoirement. Le cryptage des pixels bruts de l'image se fait via des digrammes (séquences de deux chiffres). Nous localisons le premier nombre dans la première matrice, le deuxième nombre dans la deuxième matrice, puis nous recherchons dans ces matrices les nombres qui complètent le rectangle.

La deuxième contribution consiste à utiliser plusieurs fonctions chaotiques (carte logistique, carte sinus et la carte de Chebyshev) permettant de remplir les trois faces adjacentes d'un cube de cryptage. Ce cube représente les trois canaux de l'image couleur (rouge, vert, bleu). La première phase de notre algorithme consiste à faire une extraction des pixels de l'image originale et chercher les valeurs correspondantes à chaque pixel dans le cube. Une rotation circulaire est faite en se basant sur la position de chaque pixel (ligne, colonne). Cette rotation permet de crypter deux pixels identiques en deux valeurs différentes. La méthode d'intersection des plans avec la face correspondante est utilisée pour crypter les pixels. Pour la partie de la

confusion, on a choisi la transformation 2D « Arnold Cat Map » afin de mélanger les pixels et changer leurs positions avec des paramètres dépendant de tous les pixels de l'image originale.

La troisième contribution introduit une méthode de cryptage d'image couleur 32 bits améliorée à l'aide de quatre cartes chaotiques 1D, la carte logistique, la carte Chebyshev, la carte Tente et la carte Sine. Ces cartes ont été utilisées pour remplir les quatre matrices de notre système de cryptage avec des entiers uniques compris entre 0 et 255. Chaque matrice a une taille de 16 × 16 disposée en carré. Les quatre canaux d'une image couleur 32 bits sont représentés par ces matrices (rouge, vert, bleu et alpha). Les matrices supérieures gauche et inférieure droite sont utilisées pour crypter respectivement les canaux rouge et alpha. Les matrices supérieures droite et inférieure gauche sont utilisées pour crypter respectivement les canaux vert et bleu. La première partie de notre méthode prend toutes les valeurs décimales des pixels de l'image originale et recherche le pixel correspondant dans les quatre matrices. Dans la deuxième partie, une rotation circulaire a été appliquée en utilisant la position de chaque pixel de l'image originale (ligne, colonne). Cette opération a été faite pour empêcher le cryptage de zones homogènes de même couleur. Ensuite, nous avons utilisé la méthode de chiffrement à quatre carrés pour chiffrer les valeurs décimales des pixels. Pour la section concernant confusion, nous avons appliqué la transformation Arnold Cat Map pour mélanger tous les pixels et changer leurs positions.

Plusieurs critères de sécurité ont été utilisés pour évaluer les performances de nos algorithmes contre les différentes attaques. De plus, plusieurs expérimentations et comparaisons de nos approches avec des méthodes, bien établies dans la littérature, ont été effectuées pour confirmer et justifier la qualité de nos différents travaux présentés dans cette thèse.

Perspectives

Dans cette section, nous présentons les perspectives et les futurs travaux sur la cryptographie des images couleurs, qui pourrait être intéressants à approfondir.

Dans nos contributions, la génération des clés est faite à partir des cartes chaotiques de type 1D. Malgré la simplicité de ce type des systèmes chaotiques, qui sont faciles à implémenter, ces systèmes finiront par devenir périodiques. Ce type de problème est pratiquement évité dans les cartes chaotiques de grande dimension, puisque la période se trouve beaucoup plus longue

que les cartes chaotiques de type 1D. Par conséquent, il est plus difficile de prédire les clés générées avec des systèmes chaotiques 1D.

Un autre axe de recherche qui parait intéressant, vu l'apparition et les améliorations continues des ordinateurs quantiques, de nombreux chercheurs rejoignent maintenant le domaine quantique et explorent de nouvelles méthodes de protection de l'information. On sait que les algorithmes actuels prendraient des millions d'années pour factoriser, par exemple, un nombre à 500 chiffres. En revanche, un ordinateur quantique entièrement fonctionnel pourrait le faire en quelques minutes, c'est la raison pour laquelle un standard cryptographie post-quantique doit être développé le plus vite possible pour surmonter ce type de problème.

Annexe A : Algorithmes des signatures électroniques

Algorithme de MD5 :

Rappelez-vous qu'une fonction de hachage calcule un résumé du texte, et ce résumé est très sensible au texte original (une petite modification du texte entraîne une grande modification du résumé). Les deux algorithmes de hachage les plus couramment utilisés sont SHA (Secure Hash Algorithm), qui calcule un résumé de 160 bits, et MD5, qui calcule un résumé de 128 bits. Nous vous présentons le dernier algorithme, qui procède en plusieurs étapes.

Début

Le message est composé de b bits $m_1 m_2 \ldots . m_6$. Nous complétons le message avec des 1 et suffisamment de 0 pour que la longueur du message étendu soit égale à $448\ mod\ 512$.

On ajoute alors la valeur de b à ce message, encodé en binaire 64 bits (on a donc b, qui peut avoir une valeur aussi élevée que 2^{64}... ce qui est énorme). On obtient ainsi des messages dont la longueur totale est un multiple de 512 bits. Nous allons parcourir chaque bloc de 512 bits.

On définit 4 buffers A, B, C et D de 32 bits, initialisés ainsi (les nombres sont en hexadécimal, c'est à dire $a = 10$, $b = 11$...).

$$A = 01234567$$
$$B = 89abcdef$$
$$C = fedcba98$$
$$D = 76543210$$

On définit aussi 4 fonctions F, G, H et I, qui prennent des arguments codés sur 32 bits, et renvoie une valeur sur 32 bits, les opérations se faisant bit à bit.

$$F(X,Y,Z) = (X\ AND\ Y)\ OR\ (\neg X\ AND\ Z)$$

$$G(X,Y,Z) = (X\ AND\ Z)\ OR\ (Y\ AND\ \neg Z)$$

$$H(X,Y,Z) = (X \oplus Y \oplus Z)$$

$$I(X,Y,Z) = Y \oplus (X\ OR\ \neg Z)$$

L'important à propos de ces 4 fonctions est que si les bits de leurs arguments X, Y et Z sont indépendants, alors les bits du résultat sont également indépendants.

Pour chaque bloc de texte de 512 bits, nous procédons comme suit :

1. Enregistrez la valeur du registre dans AA, BB, CC, DD.
2. Les nouvelles valeurs de A, B, C, D sont calculées à partir de leurs anciennes valeurs, des bits du bloc étudié et des 4 fonctions F, G, H, I.
3. Nous faisons $A = AA + A$, $B = BB + B$, $C = CC + C$, $D = DD + D$.

Le condensé de 128 bits est obtenu en concaténant les 4 tampons A, B, C et D de 32 bits de bout en bout.

Fin

Algorithme 4-1 : *Description du fonctionnement de MD5*

Algorithme de SHA :

Le 31 janvier 1992, le NIST (National Institute of Standards and Technology, États-Unis) a publié dans le Federal Register une proposition de norme de hachage sécurisé (SHS) contenant la définition de l'algorithme de hachage sécurisé (SHA). Cette fonction de hachage est destinée à être utilisée conjointement avec l'algorithme de signature numérique (DSA) proposé par la norme de signature numérique (DSS). La méthode est clairement basée sur MD4, mais elle a été améliorée. La taille des variables de chaînage est portée à 5 blocs de 32 bits, pour un total de 160 bits. Cela semble être cohérent avec la faisabilité d'une attaque d'anniversaire.

Début

Valeurs initiales : $h0 = 67452301$, $h1 = EFCDAB89$, $h2 = 98BADCFE$,

$h3 = 10325476$, $h3 = 10325476$, $h4 = C3d2E1F0$.

$A = h0$; $B = h1$; $C = h2$; $D = h3$; $E = h4$.

Pour i *allant de* 0 *jusqu'à* $t - 1$ ***faire***

Pour j *allant de* 0 *jusqu'à* 79 ***faire***

if $(j > 15)$ ***alors***

$X[i][j] = X[i][j-3] \oplus X[i][j-8] \oplus X[i][j-14] \oplus X[i][j-16];$

Fin Si

$Tmp = rol(A, 5) + F(j, B, C, D) + E + X[i][j] + K(j)$

$E = D;\ D = C;\ C = rol(B, 30);\ B = A;\ A = Tmp;$

Fin Pour

$h0 = h0 + A;\ h1 = h1 + B;\ h2 = h2 + C;\ h3 = h3 + D;\ h4 = h4 + E;$

Fin Pour

Fin

Algorithme 4-2 : *Description du fonctionnement de SHA.*

Publications

Articles publiés dans des journaux internationaux

1) M. Es-Sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « A new image encryption algorithm using random numbers generation of two matrices and bit-shift operators », **Soft Computing**, vol. 24, no 5, p. 3829-3848, mars 2020, doi: 10.1007/s00500-019-04151-8.

2) M. Es-Sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « A new color image encryption algorithm using multiple chaotic maps with the intersecting planes method », **Scientific African**, vol. 16, p. e01217, juill. 2022, doi: 10.1016/j.sciaf.2022.e01217.

3) Hjouji, A., Jourhmane, M., EL-Mekkaoui, J., Es-sabry, M. « Mixed Finite Element Approximation for Bivariate Perona–Malik Model Arising in 2D and 3D Image Denoising », **3D Research**, vol. 9, no 36, September 2018, doi: 10.1007/s13319-018-0187-6

Article accepté avec modification

1) M. Es-Sabry, N. El Akkad, M. Merras, et K. Satori, « A New 32-Bit Color Image Encryption Method Based on Multiple 1d Chaotic Maps and the Four Square Cipher », submitted to: **Journal of Mathematical Imaging and Vision**, May 2023.

Communications internationales

1) M. Es-sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « A New Color Image Encryption Algorithm Using Random Number Generation and Linear Functions », in Embedded Systems and Artificial Intelligence, Singapore, 2020, p. 581-588. doi: 10.1007/978-981-15-0947-6_55.

2) M. Es-sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « Grayscale image encryption using shift bits operations », in 2018 International Conference on Intelligent Systems and Computer Vision (ISCV), avr. 2018, p. 1-7. doi: 10.1109/ISACV.2018.8354028.

3) M. Es-Sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « A Novel Text Encryption Algorithm Based on the Two-Square Cipher and Caesar Cipher », in Big Data, Cloud and Applications, Cham, 2018, p. 78-88. doi: 10.1007/978-3-319-96292-4_7.

4) K. Karim, N. E. Akkad, K. Satori, M. ES-Sabry, M. Merras, et A. Saaidi, « Reconstruction of three-dimensional scenes from a sequence of images of which only one is calibrated », présenté à Proceedings of the Third International Conference on Computing and Wireless Communication Systems, ICCWCS 2019, April 24-25, 2019, Faculty of Sciences, Ibn Tofaïl University -Kénitra- Morocco, mai 2019. Consulté le : 14 novembre 2022. [En ligne]. Disponible sur : https://eudl.eu/doi/10.4108/eai.24-4-2019.2284139.

Bibliographie

[1] M. Es-sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « A New Color Image Encryption Algorithm Using Random Number Generation and Linear Functions », in *Embedded Systems and Artificial Intelligence*, Singapore, 2020, p. 581-588. doi: 10.1007/978-981-15-0947-6_55.

[2] M. Es-sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « Grayscale image encryption using shift bits operations », in *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, avr. 2018, p. 1-7. doi: 10.1109/ISACV.2018.8354028.

[3] M. Es-Sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « A Novel Text Encryption Algorithm Based on the Two-Square Cipher and Caesar Cipher », in *Big Data, Cloud and Applications*, Cham, 2018, p. 78-88. doi: 10.1007/978-3-319-96292-4_7.

[4] S. Gao *et al.*, « A 3D model encryption scheme based on a cascaded chaotic system », *Signal Process.*, vol. 202, p. 108745, janv. 2023, doi: 10.1016/j.sigpro.2022.108745.

[5] A. Belazi *et al.*, « Improved Sine-Tangent chaotic map with application in medical images encryption », *J. Inf. Secur. Appl.*, vol. 66, p. 103131, mai 2022, doi: 10.1016/j.jisa.2022.103131.

[6] N. Munir, M. Khan, S. S. Jamal, M. M. Hazzazi, et I. Hussain, « Cryptanalysis of hybrid secure image encryption based on Julia set fractals and three-dimensional Lorenz chaotic map », *Math. Comput. Simul.*, vol. 190, p. 826-836, déc. 2021, doi: 10.1016/j.matcom.2021.06.008.

[7] M. A. Lone et S. Qureshi, « RGB image encryption based on symmetric keys using Arnold transform, 3D chaotic map and affine hill cipher », *Optik*, vol. 260, p. 168880, juin 2022, doi: 10.1016/j.ijleo.2022.168880.

[8] C. Li et X. Yang, « An image encryption algorithm based on discrete fractional wavelet transform and quantum chaos », *Optik*, vol. 260, p. 169042, juin 2022, doi: 10.1016/j.ijleo.2022.169042.

[9] X.-H. Song, H.-Q. Wang, S. E. Venegas-Andraca, et A. A. Abd El-Latif, « Quantum video encryption based on qubit-planes controlled-XOR operations and improved logistic map », *Phys. Stat. Mech. Its Appl.*, vol. 537, p. 122660, janv. 2020, doi: 10.1016/j.physa.2019.122660.

[10] M. Tang, G. Zeng, Y. Yang, et J. Chen, « A hyperchaotic image encryption scheme based on the triple dislocation of the Liu and Lorenz system », *Optik*, vol. 261, p. 169133, juill. 2022, doi: 10.1016/j.ijleo.2022.169133.

[11] D. Wei, M. Jiang, et Y. Deng, « A secure image encryption algorithm based on hyper-chaotic and bit-level permutation », *Expert Syst. Appl.*, vol. 213, p. 119074, mars 2023, doi: 10.1016/j.eswa.2022.119074.

[12] Q. N. Natsheh, B. Li, et A. G. Gale, « Security of Multi-frame DICOM Images Using XOR Encryption Approach », *Procedia Comput. Sci.*, vol. 90, p. 175-181, janv. 2016, doi: 10.1016/j.procs.2016.07.018.

[13] X. Li *et al.*, « Multiple-image encryption via lifting wavelet transform and XOR operation based on compressive ghost imaging scheme », *Opt. Lasers Eng.*, vol. 102, p. 106-111, mars 2018, doi: 10.1016/j.optlaseng.2017.10.023.

[14] A. A. Abbasi, M. Mazinani, et R. Hosseini, « Evolutionary-based image encryption using biomolecules and non-coupled map lattice », *Opt. Laser Technol.*, vol. 140, p. 106974, août 2021, doi: 10.1016/j.optlastec.2021.106974.

[15] J. Sethi, J. Bhaumik, et A. S. Chowdhury, « Joint video compression and encryption using parallel compressive sensing and improved chaotic maps », *Digit. Signal Process.*, vol. 130, p. 103746, oct. 2022, doi: 10.1016/j.dsp.2022.103746.

[16] F. Ikbal et R. Gopikakumari, « Image block generation from block-based SMRT in colour image encryption and its performance analysis », *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, n° 10, Part A, p. 8459-8477, nov. 2022, doi: 10.1016/j.jksuci.2021.08.026.

[17] X. Wang et R. Si, « A new chaotic image encryption scheme based on dynamic L-shaped scrambling and combined map diffusion », *Optik*, vol. 245, p. 167658, nov. 2021, doi: 10.1016/j.ijleo.2021.167658.

[18] K. Ahmadi et A. Carnicer, « Optical visual encryption using focused beams and convolutional neural networks », *Opt. Lasers Eng.*, vol. 161, p. 107321, févr. 2023, doi: 10.1016/j.optlaseng.2022.107321.

[19] P. Chaudhary, R. Gupta, A. Singh, P. Majumder, et A. Pandey, « Joint image compression and encryption using a novel column-wise scanning and optimization algorithm », *Procedia Comput. Sci.*, vol. 167, p. 244-253, janv. 2020, doi: 10.1016/j.procs.2020.03.218.

[20] S. M. Basha, P. Mathivanan, et A. B. Ganesh, « Bit level color image encryption using Logistic-Sine-Tent-Chebyshev (LSTC) map », *Optik*, vol. 259, p. 168956, juin 2022, doi: 10.1016/j.ijleo.2022.168956.

[21] M. Helmy, E.-S. M. El-Rabaie, I. Eldokany, et F. E. A. El-Samie, « Proposed Hybrid Encryption Algorithm for Robust 3D Image Communication over Wireless Channels », *Optik*, p. 170205, nov. 2022, doi: 10.1016/j.ijleo.2022.170205.

[22] A. A. Alarood, E. Alsolami, M. A. Al-Khasawneh, N. Ababneh, et W. Elmedany, « IES: Hyper-chaotic plain image encryption scheme using improved shuffled confusion-diffusion », *Ain Shams Eng. J.*, vol. 13, n° 3, p. 101583, mai 2022, doi: 10.1016/j.asej.2021.09.010.

[23] F. Elazzaby, N. E. Akkad, et S. kabbaj, « Advanced encryption of image based on S-box and chaos 2D (LSMCL) », in *2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, avr. 2020, p. 1-7. doi: 10.1109/IRASET48871.2020.9092254.

[24] F. Elazzaby, N. El Akkad, et S. Kabbaj, « A New Encryption Approach Based on Four-Square and Zigzag Encryption (C4CZ) », in *Embedded Systems and Artificial Intelligence*, Singapore, 2020, p. 589-597. doi: 10.1007/978-981-15-0947-6_56.

[25] H. TOUIL, N. E. AKKAD, et K. SATORI, « Text Encryption: Hybrid cryptographic method using Vigenere and Hill Ciphers », in *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, juin 2020, p. 1-6. doi: 10.1109/ISCV49265.2020.9204095.

[26] H. Touil, N. El Akkad, et K. Satori, « H-Rotation: Secure Storage and Retrieval of Passphrases on the Authentication Process », *Int. J. Saf. Secur. Eng.*, vol. 10, nº 6, p. 785-796, déc. 2020, doi: 10.18280/ijsse.100609.

[27] H. Touil, N. E. Akkad, et K. Satori, « Secure and Guarantee QoS in a Video Sequence: A New Approach based on TLS Protocol to Secure Data and RTP to Ensure Real-time Exchanges », *WSEAS Trans. Commun.*, vol. 20, p. 52-62, 2021, doi: 10.37394/23204.2021.20.7.

[28] L. Khrissi, N. El Akkad, H. Satori, et K. Satori, « Clustering method and sine cosine algorithm for image segmentation », *Evol. Intell.*, vol. 15, nº 1, p. 669-682, mars 2022, doi: 10.1007/s12065-020-00544-z.

[29] H. Huang, S. Yang, et R. Ye, « Image encryption scheme combining a modified Gerchberg–Saxton algorithm with hyper-chaotic system », *Soft Comput.*, vol. 23, nº 16, p. 7045-7053, août 2019, doi: 10.1007/s00500-018-3345-0.

[30] S. Kandar, D. Chaudhuri, A. Bhattacharjee, et B. C. Dhara, « Image encryption using sequence generated by cyclic group », *J. Inf. Secur. Appl.*, vol. 44, p. 117-129, févr. 2019, doi: 10.1016/j.jisa.2018.12.003.

[31] A. Broumandnia, « Designing digital image encryption using 2D and 3D reversible modular chaotic maps », *J. Inf. Secur. Appl.*, vol. 47, p. 188-198, août 2019, doi: 10.1016/j.jisa.2019.05.004.

[32] B. Li, X. Liao, et Y. Jiang, « A novel image encryption scheme based on improved random number generator and its implementation », *Nonlinear Dyn.*, vol. 95, nº 3, p. 1781-1805, févr. 2019, doi: 10.1007/s11071-018-4659-2.

[33] G. Hu et B. Li, « Coupling chaotic system based on unit transform and its applications in image encryption », *Signal Process.*, vol. 178, p. 107790, janv. 2021, doi: 10.1016/j.sigpro.2020.107790.

[34] M. Zhou et C. Wang, « A novel image encryption scheme based on conservative hyperchaotic system and closed-loop diffusion between blocks », *Signal Process.*, vol. 171, p. 107484, juin 2020, doi: 10.1016/j.sigpro.2020.107484.

[35] W. Wen, K. Wei, Y. Zhang, Y. Fang, et M. Li, « Colour light field image encryption based on DNA sequences and chaotic systems », *Nonlinear Dyn.*, vol. 99, nº 2, p. 1587-1600, janv. 2020, doi: 10.1007/s11071-019-05378-8.

[36] M. Babaei, « A novel text and image encryption method based on chaos theory and DNA computing », *Nat. Comput.*, vol. 12, nº 1, p. 101-107, mars 2013, doi: 10.1007/s11047-012-9334-9.

[37] X. Yan, X. Wang, et Y. Xian, « Chaotic image encryption algorithm based on arithmetic sequence scrambling model and DNA encoding operation », *Multimed. Tools Appl.*, vol. 80, nº 7, p. 10949-10983, mars 2021, doi: 10.1007/s11042-020-10218-8.

[38] S. Suri et R. Vijay, « A Pareto-optimal evolutionary approach of image encryption using coupled map lattice and DNA », *Neural Comput. Appl.*, vol. 32, nº 15, p. 11859-11873, août 2020, doi: 10.1007/s00521-019-04668-x.

[39] R. Guesmi, M. A. B. Farah, A. Kachouri, et M. Samet, « A novel chaos-based image encryption using DNA sequence operation and Secure Hash Algorithm SHA-2 »,

Nonlinear Dyn., vol. 83, nº 3, p. 1123-1136, févr. 2016, doi: 10.1007/s11071-015-2392-7.

[40] H. Hermassi, A. Belazi, R. Rhouma, et S. M. Belghith, « Security analysis of an image encryption algorithm based on a DNA addition combining with chaotic maps », *Multimed. Tools Appl.*, vol. 72, nº 3, p. 2211-2224, oct. 2014, doi: 10.1007/s11042-013-1533-6.

[41] A. Jain et N. Rajpal, « A robust image encryption algorithm resistant to attacks using DNA and chaotic logistic maps », *Multimed. Tools Appl.*, vol. 75, nº 10, p. 5455-5472, mai 2016, doi: 10.1007/s11042-015-2515-7.

[42] X. Liu, X. Tong, Z. Wang, et M. Zhang, « A novel hyperchaotic encryption algorithm for color image utilizing DNA dynamic encoding and self-adapting permutation », *Multimed. Tools Appl.*, vol. 81, nº 15, p. 21779-21810, juin 2022, doi: 10.1007/s11042-022-12472-4.

[43] W. Yu, Y. Liu, L. Gong, M. Tian, et L. Tu, « Double-image encryption based on spatiotemporal chaos and DNA operations », *Multimed. Tools Appl.*, vol. 78, nº 14, p. 20037-20064, juill. 2019, doi: 10.1007/s11042-018-7110-2.

[44] C. Yang, I. Taralova, S. El Assad, et J.-J. Loiseau, « Image encryption based on fractional chaotic pseudo-random number generator and DNA encryption method », *Nonlinear Dyn.*, vol. 109, nº 3, p. 2103-2127, août 2022, doi: 10.1007/s11071-022-07534-z.

[45] E. Z. Zefreh, « An image encryption scheme based on a hybrid model of DNA computing, chaotic systems and hash functions », *Multimed. Tools Appl.*, vol. 79, nº 33, p. 24993-25022, sept. 2020, doi: 10.1007/s11042-020-09111-1.

[46] X. Chai, Z. Gan, K. Yuan, Y. Chen, et X. Liu, « A novel image encryption scheme based on DNA sequence operations and chaotic systems », *Neural Comput. Appl.*, vol. 31, nº 1, p. 219-237, janv. 2019, doi: 10.1007/s00521-017-2993-9.

[47] P. Zhen, G. Zhao, L. Min, et X. Jin, « Chaos-based image encryption scheme combining DNA coding and entropy », *Multimed. Tools Appl.*, vol. 75, nº 11, p. 6303-6319, juin 2016, doi: 10.1007/s11042-015-2573-x.

[48] A. N. Kengnou Telem, H. B. Fotsin, et J. Kengne, « Image encryption algorithm based on dynamic DNA coding operations and 3D chaotic systems », *Multimed. Tools Appl.*, vol. 80, nº 12, p. 19011-19041, mai 2021, doi: 10.1007/s11042-021-10549-0.

[49] Q. Zhang et J. Han, « A novel color image encryption algorithm based on image hashing, 6D hyperchaotic and DNA coding », *Multimed. Tools Appl.*, vol. 80, nº 9, p. 13841-13864, avr. 2021, doi: 10.1007/s11042-020-10437-z.

[50] A. Girdhar et V. Kumar, « A RGB image encryption technique using Lorenz and Rossler chaotic system on DNA sequences », *Multimed. Tools Appl.*, vol. 77, nº 20, p. 27017-27039, oct. 2018, doi: 10.1007/s11042-018-5902-z.

[51] J. Tian *et al.*, « A novel image encryption algorithm using PWLCM map-based CML chaotic system and dynamic DNA encryption », *Multimed. Tools Appl.*, vol. 80, nº 21, p. 32841-32861, sept. 2021, doi: 10.1007/s11042-021-11218-y.

[52] S. Stalin, P. Maheshwary, P. K. Shukla, M. Maheshwari, B. Gour, et A. Khare, « Fast and Secure Medical Image Encryption Based on Non Linear 4D Logistic Map and DNA Sequences (NL4DLM_DNA) », *J. Med. Syst.*, vol. 43, nº 8, p. 267, juill. 2019, doi: 10.1007/s10916-019-1389-z.

[53] S. Zhang et T. Gao, « An image encryption scheme based on DNA coding and permutation of hyper-image », *Multimed. Tools Appl.*, vol. 75, nº 24, p. 17157-17170, déc. 2016, doi: 10.1007/s11042-015-2982-x.

[54] X. Zhang et X. Wang, « Multiple-image encryption algorithm based on DNA encoding and chaotic system », *Multimed. Tools Appl.*, vol. 78, nº 6, p. 7841-7869, mars 2019, doi: 10.1007/s11042-018-6496-1.

[55] S. E. El-Khamy et A. G. Mohamed, « An efficient DNA-inspired image encryption algorithm based on hyper-chaotic maps and wavelet fusion », *Multimed. Tools Appl.*, vol. 80, nº 15, p. 23319-23335, juin 2021, doi: 10.1007/s11042-021-10527-6.

[56] X. Zhang et R. Ye, « A novel RGB image encryption algorithm based on DNA sequences and chaos », *Multimed. Tools Appl.*, vol. 80, nº 6, p. 8809-8833, mars 2021, doi: 10.1007/s11042-020-09465-6.

[57] X. Wu, K. Wang, X. Wang, et H. Kan, « Lossless chaotic color image cryptosystem based on DNA encryption and entropy », *Nonlinear Dyn.*, vol. 90, nº 2, p. 855-875, oct. 2017, doi: 10.1007/s11071-017-3698-4.

[58] Z. Azimi et S. Ahadpour, « Color image encryption based on DNA encoding and pair coupled chaotic maps », *Multimed. Tools Appl.*, vol. 79, nº 3, p. 1727-1744, janv. 2020, doi: 10.1007/s11042-019-08375-6.

[59] Z. Liang, Q. Qin, C. Zhou, et S. Xu, « Color Image Encryption Algorithm Based on Four-Dimensional Multi-stable Hyper Chaotic System and DNA Strand Displacement », *J. Electr. Eng. Technol.*, juill. 2022, doi: 10.1007/s42835-022-01157-5.

[60] C. Zhu, Z. Gan, Y. Lu, et X. Chai, « An image encryption algorithm based on 3-D DNA level permutation and substitution scheme », *Multimed. Tools Appl.*, vol. 79, nº 11, p. 7227-7258, mars 2020, doi: 10.1007/s11042-019-08226-4.

[61] W. El-Shafai, F. Khallaf, E.-S. M. El-Rabaie, et F. E. A. El-Samie, « Robust medical image encryption based on DNA-chaos cryptosystem for secure telemedicine and healthcare applications », *J. Ambient Intell. Humaniz. Comput.*, vol. 12, nº 10, p. 9007-9035, oct. 2021, doi: 10.1007/s12652-020-02597-5.

[62] J. Wang, X. Zhi, X. Chai, et Y. Lu, « Chaos-based image encryption strategy based on random number embedding and DNA-level self-adaptive permutation and diffusion », *Multimed. Tools Appl.*, vol. 80, nº 10, p. 16087-16122, avr. 2021, doi: 10.1007/s11042-020-10413-7.

[63] S. A. Elsaid, E. R. Alotaibi, et S. Alsaleh, « A robust hybrid cryptosystem based on DNA and Hyperchaotic for images encryption », *Multimed. Tools Appl.*, juin 2022, doi: 10.1007/s11042-022-12641-5.

[64] X. Wang, B. Li, Y. Wang, J. Lei, et J. Xue, « An efficient batch images encryption method based on DNA encoding and PWLCM », *Multimed. Tools Appl.*, vol. 80, nº 1, p. 943-971, janv. 2021, doi: 10.1007/s11042-020-09533-x.

[65] X. Wang et H. Sun, « A chaotic image encryption algorithm based on zigzag-like transform and DNA-like coding », *Multimed. Tools Appl.*, vol. 78, nº 24, p. 34981-34997, déc. 2019, doi: 10.1007/s11042-019-08085-z.

[66] X. Wang et M. Zhang, « A new image encryption algorithm based on ladder transformation and DNA coding », *Multimed. Tools Appl.*, vol. 80, nº 9, p. 13339-13365, avr. 2021, doi: 10.1007/s11042-020-10318-5.

[67] Y. Wang, C. Wu, S. Kang, Q. Wang, et V. I. Mikulovich, « Multi-channel chaotic encryption algorithm for color image based on DNA coding », *Multimed. Tools Appl.*, vol. 79, nº 25, p. 18317-18342, juill. 2020, doi: 10.1007/s11042-020-08742-8.

[68] L. Chen, H. Yin, L. Yuan, A. M. Lopes, J. A. T. Machado, et R. Wu, « A novel color image encryption algorithm based on a fractional-order discrete chaotic neural network and DNA sequence operations », *Front. Inf. Technol. Electron. Eng.*, vol. 21, nº 6, p. 866-879, juin 2020, doi: 10.1631/FITEE.1900709.

[69] F. Özkaynak et S. Yavuz, « Analysis and improvement of a novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system », *Nonlinear Dyn.*, vol. 78, nº 2, p. 1311-1320, oct. 2014, doi: 10.1007/s11071-014-1517-8.

[70] N. Nandy, D. Banerjee, et C. Pradhan, « Color image encryption using DNA based cryptography », *Int. J. Inf. Technol.*, vol. 13, nº 2, p. 533-540, avr. 2021, doi: 10.1007/s41870-018-0100-9.

[71] Y. Zhang, « The image encryption algorithm based on chaos and DNA computing », *Multimed. Tools Appl.*, vol. 77, nº 16, p. 21589-21615, août 2018, doi: 10.1007/s11042-017-5585-x.

[72] X.-Y. Wang, P. Li, Y.-Q. Zhang, L.-Y. Liu, H. Zhang, et X. Wang, « A novel color image encryption scheme using DNA permutation based on the Lorenz system », *Multimed. Tools Appl.*, vol. 77, nº 5, p. 6243-6265, mars 2018, doi: 10.1007/s11042-017-4534-z.

[73] X. Chai, Y. Chen, et L. Broyde, « A novel chaos-based image encryption algorithm using DNA sequence operations », *Opt. Lasers Eng.*, vol. 88, p. 197-213, janv. 2017, doi: 10.1016/j.optlaseng.2016.08.009.

[74] X. Wu, K. Wang, X. Wang, H. Kan, et J. Kurths, « Color image DNA encryption using NCA map-based CML and one-time keys », *Signal Process.*, vol. 148, p. 272-287, juill. 2018, doi: 10.1016/j.sigpro.2018.02.028.

[75] X. Chai, Z. Gan, K. Yang, Y. Chen, et X. Liu, « An image encryption algorithm based on the memristive hyperchaotic system, cellular automata and DNA sequence operations », *Signal Process. Image Commun.*, vol. 52, p. 6-19, mars 2017, doi: 10.1016/j.image.2016.12.007.

[76] K. C. Jithin et S. Sankar, « Colour image encryption algorithm combining Arnold map, DNA sequence operation, and a Mandelbrot set », *J. Inf. Secur. Appl.*, vol. 50, p. 102428, févr. 2020, doi: 10.1016/j.jisa.2019.102428.

[77] T. ul Haq et T. Shah, « Algebra-chaos amalgam and DNA transform based multiple digital image encryption », *J. Inf. Secur. Appl.*, vol. 54, p. 102592, oct. 2020, doi: 10.1016/j.jisa.2020.102592.

[78] J. Liu, M. Zhang, X. Tong, et Z. Wang, « Image compression and encryption algorithm based on compressive sensing and nonlinear diffusion », *Multimed. Tools Appl.*, vol. 80, nº 17, p. 25433-25452, juill. 2021, doi: 10.1007/s11042-021-10884-2.

[79] H. Huang et D. Cheng, « A secure image compression-encryption algorithm using DCT and hyperchaotic system », *Multimed. Tools Appl.*, vol. 81, nº 22, p. 31329-31347, sept. 2022, doi: 10.1007/s11042-021-11796-x.

[80] Y.-G. Yang, B.-W. Guan, Y.-H. Zhou, et W.-M. Shi, « Double image compression-encryption algorithm based on fractional order hyper chaotic system and DNA approach », *Multimed. Tools Appl.*, vol. 80, nº 1, p. 691-710, janv. 2021, doi: 10.1007/s11042-020-09779-5.

[81] D. Huo, X. Zhu, G. Dai, H. Yang, X. Zhou, et M. Feng, « Novel image compression–encryption hybrid scheme based on DNA encoding and compressive sensing », *Appl. Phys. B*, vol. 126, nº 3, p. 45, févr. 2020, doi: 10.1007/s00340-020-7397-3.

[82] J. Li, Y. Sheng, X. Di, et Y. Mu, « Range-gated laser image compression and encryption scheme based on bidirectional diffusion », *Optoelectron. Lett.*, vol. 17, nº 10, p. 630-635, oct. 2021, doi: 10.1007/s11801-021-1003-8.

[83] X. Chai, X. Fu, Z. Gan, Y. Zhang, Y. Lu, et Y. Chen, « An efficient chaos-based image compression and encryption scheme using block compressive sensing and elementary cellular automata », *Neural Comput. Appl.*, vol. 32, nº 9, p. 4961-4988, mai 2020, doi: 10.1007/s00521-018-3913-3.

[84] J. Liu, M. Zhang, X. Tong, et Z. Wang, « Image compression and encryption algorithm based on 2D compressive sensing and hyperchaotic system », *Multimed. Syst.*, vol. 28, nº 2, p. 595-610, avr. 2022, doi: 10.1007/s00530-021-00859-6.

[85] J. Xu, J. Mou, J. Liu, et J. Hao, « The image compression–encryption algorithm based on the compression sensing and fractional-order chaotic system », *Vis. Comput.*, vol. 38, nº 5, p. 1509-1526, mai 2022, doi: 10.1007/s00371-021-02085-7.

[86] Y. Song, Z. Zhu, W. Zhang, L. Guo, X. Yang, et H. Yu, « Joint image compression–encryption scheme using entropy coding and compressive sensing », *Nonlinear Dyn.*, vol. 95, nº 3, p. 2235-2261, févr. 2019, doi: 10.1007/s11071-018-4689-9.

[87] S. P. Raja, « Joint medical image compression–encryption in the cloud using multiscale transform-based image compression encoding techniques », *Sādhanā*, vol. 44, nº 2, p. 28, janv. 2019, doi: 10.1007/s12046-018-1013-9.

[88] X.-J. Tong, P. Chen, et M. Zhang, « A joint image lossless compression and encryption method based on chaotic map », *Multimed. Tools Appl.*, vol. 76, nº 12, p. 13995-14020, juin 2017, doi: 10.1007/s11042-016-3775-6.

[89] M. Gschwandtner, A. Uhl, et P. Wild, « Transmission Error and Compression Robustness of 2D Chaotic Map Image Encryption Schemes », *EURASIP J. Inf. Secur.*, vol. 2007, nº 1, p. 048179, nov. 2007, doi: 10.1155/2007/48179.

[90] X.-Z. Li, W.-W. Chen, et Y.-Q. Wang, « Quantum Image Compression-Encryption Scheme Based on Quantum Discrete Cosine Transform », *Int. J. Theor. Phys.*, vol. 57, nº 9, p. 2904-2919, sept. 2018, doi: 10.1007/s10773-018-3810-7.

[91] S. M. Darwish, « A modified image selective encryption-compression technique based on 3D chaotic maps and arithmetic coding », *Multimed. Tools Appl.*, vol. 78, nº 14, p. 19229-19252, juill. 2019, doi: 10.1007/s11042-019-7256-6.

[92] C. T. Selvi, J. Amudha, et R. Sudhakar, « Medical image encryption and compression by adaptive sigma filterized synorr certificateless signcryptive Levenshtein entropy-coding-

based deep neural learning », *Multimed. Syst.*, vol. 27, nº 6, p. 1059-1074, déc. 2021, doi: 10.1007/s00530-021-00764-y.

[93] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, et K. Ramchandran, « On compressing encrypted data », *IEEE Trans. Signal Process.*, vol. 52, nº 10, p. 2992-3006, oct. 2004, doi: 10.1109/TSP.2004.833860.

[94] A. Anil Kumar et A. Makur, « Distributed source coding based encryption and lossless compression of gray scale and color images », in *2008 IEEE 10th Workshop on Multimedia Signal Processing*, oct. 2008, p. 760-764. doi: 10.1109/MMSP.2008.4665176.

[95] R. Lazzeretti et M. Barni, « Lossless compression of encrypted grey-level and color images », in *2008 16th European Signal Processing Conference*, août 2008, p. 1-5.

[96] S.-C. Ou, H.-Y. Chung, et W.-T. Sung, « Improving the compression and encryption of images using FPGA-based cryptosystems », *Multimed. Tools Appl.*, vol. 28, nº 1, p. 5-22, janv. 2006, doi: 10.1007/s11042-006-5117-6.

[97] V. P. Kulalvaimozhi, M. G. Alex, et S. J. Peter, « A Novel Homomorphic Encryption and an enhanced DWT (NHE-EDWT) compression of crop images in agriculture field », *Multidimens. Syst. Signal Process.*, vol. 31, nº 2, p. 367-383, avr. 2020, doi: 10.1007/s11045-019-00660-9.

[98] R. Grant, M. Alsewan, et M. Devlin, « Compression/encryption technology applied to Dicom/digitised mammography », *Breast Cancer Res.*, vol. 6, nº 1, p. P41, juill. 2004, doi: 10.1186/bcr860.

[99] S. Hanis et R. Amutha, « Double image compression and encryption scheme using logistic mapped convolution and cellular automata », *Multimed. Tools Appl.*, vol. 77, nº 6, p. 6897-6912, mars 2018, doi: 10.1007/s11042-017-4606-0.

[100] Z. Zhu, Y. Song, W. Zhang, H. Yu, et Y. Zhao, « A novel compressive sensing-based framework for image compression-encryption with S-box », *Multimed. Tools Appl.*, vol. 79, nº 35, p. 25497-25533, sept. 2020, doi: 10.1007/s11042-020-09193-x.

[101] X. Lv, X. Liao, et B. Yang, « A novel scheme for simultaneous image compression and encryption based on wavelet packet transform and multi-chaotic systems », *Multimed. Tools Appl.*, vol. 77, nº 21, p. 28633-28663, nov. 2018, doi: 10.1007/s11042-018-6013-6.

[102] J.-Y. Dai, Y. Ma, et N.-R. Zhou, « Quantum multi-image compression-encryption scheme based on quantum discrete cosine transform and 4D hyper-chaotic Henon map », *Quantum Inf. Process.*, vol. 20, nº 7, p. 246, juill. 2021, doi: 10.1007/s11128-021-03187-w.

[103] K. M. Ameen Suhail et S. Sankar, « Image Compression and Encryption Combining Autoencoder and Chaotic Logistic Map », *Iran. J. Sci. Technol. Trans. Sci.*, vol. 44, nº 4, p. 1091-1100, août 2020, doi: 10.1007/s40995-020-00905-4.

[104] Z. Gan, X. Chai, J. Zhang, Y. Zhang, et Y. Chen, « An effective image compression–encryption scheme based on compressive sensing (CS) and game of life (GOL) », *Neural Comput. Appl.*, vol. 32, nº 17, p. 14113-14141, sept. 2020, doi: 10.1007/s00521-020-04808-8.

[105] X.-J. Tong, M. Zhang, Z. Wang, et J. Ma, « A joint color image encryption and compression scheme based on hyper-chaotic system », *Nonlinear Dyn.*, vol. 84, nº 4, p. 2333-2356, juin 2016, doi: 10.1007/s11071-016-2648-x.

[106] Z. Gan, X. Chai, J. Bi, et X. Chen, « Content-adaptive image compression and encryption via optimized compressive sensing with double random phase encoding driven by chaos », *Complex Intell. Syst.*, vol. 8, nº 3, p. 2291-2309, juin 2022, doi: 10.1007/s40747-022-00644-6.

[107] Z. Gan, J. Bi, W. Ding, et X. Chai, « Exploiting 2D compressed sensing and information entropy for secure color image compression and encryption », *Neural Comput. Appl.*, vol. 33, nº 19, p. 12845-12867, oct. 2021, doi: 10.1007/s00521-021-05937-4.

[108] T. Duseja et M. Deshmukh, « Image compression and encryption using chinese remainder theorem », *Multimed. Tools Appl.*, vol. 78, nº 12, p. 16727-16753, juin 2019, doi: 10.1007/s11042-018-7023-0.

[109] J. Cai, S. Xie, et J. Zhang, « Image compression-encryption algorithm based on chaos and compressive sensing », *Multimed. Tools Appl.*, juill. 2022, doi: 10.1007/s11042-022-13346-5.

[110] H. Li, C. Yu, et X. Wang, « A novel 1D chaotic system for image encryption, authentication and compression in cloud », *Multimed. Tools Appl.*, vol. 80, nº 6, p. 8721-8758, mars 2021, doi: 10.1007/s11042-020-10117-y.

[111] L. Zhang, R. Xiong, J. Chen, et D. Zhang, « Optical image compression and encryption transmission-based ondeep learning and ghost imaging », *Appl. Phys. B*, vol. 126, nº 1, p. 18, déc. 2019, doi: 10.1007/s00340-019-7362-1.

[112] J. Xiaoyong, B. Sen, Z. Guibin, et Y. Bing, « Image encryption and compression based on the generalized knight's tour, discrete cosine transform and chaotic maps », *Multimed. Tools Appl.*, vol. 76, nº 10, p. 12965-12979, mai 2017, doi: 10.1007/s11042-016-3684-8.

[113] S. Liu, A. Paul, G. Zhang, et G. Jeon, « A game theory-based block image compression method in encryption domain », *J. Supercomput.*, vol. 71, nº 9, p. 3353-3372, sept. 2015, doi: 10.1007/s11227-015-1413-0.

[114] J. Karmakar, D. Nandi, et M. K. Mandal, « A novel hyper-chaotic image encryption with sparse-representation based compression », *Multimed. Tools Appl.*, vol. 79, nº 37, p. 28277-28300, oct. 2020, doi: 10.1007/s11042-020-09125-9.

[115] R. Ponuma et R. Amutha, « Compressive sensing based image compression-encryption using Novel 1D-Chaotic map », *Multimed. Tools Appl.*, vol. 77, nº 15, p. 19209-19234, août 2018, doi: 10.1007/s11042-017-5378-2.

[116] J. Mou, F. Yang, R. Chu, et Y. Cao, « Image Compression and Encryption Algorithm Based on Hyper-chaotic Map », *Mob. Netw. Appl.*, vol. 26, nº 5, p. 1849-1861, oct. 2021, doi: 10.1007/s11036-019-01293-9.

[117] X.-J. Tong, Z. Wang, M. Zhang, et Y. Liu, « A new algorithm of the combination of image compression and encryption technology based on cross chaotic map », *Nonlinear Dyn.*, vol. 72, nº 1, p. 229-241, avr. 2013, doi: 10.1007/s11071-012-0707-5.

[118] J. Deng, S. Zhao, Y. Wang, L. Wang, H. Wang, et H. Sha, « Image compression-encryption scheme combining 2D compressive sensing with discrete fractional random transform », *Multimed. Tools Appl.*, vol. 76, nº 7, p. 10097-10117, avr. 2017, doi: 10.1007/s11042-016-3600-2.

[119] J. Ahmad, M. A. Khan, S. O. Hwang, et J. S. Khan, « A compression sensing and noise-tolerant image encryption scheme based on chaotic maps and orthogonal matrices »,

Neural Comput. Appl., vol. 28, nº 1, p. 953-967, déc. 2017, doi: 10.1007/s00521-016-2405-6.

[120] T. S. Cho, S. Avidan, et W. T. Freeman, « A probabilistic image jigsaw puzzle solver », in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, juin 2010, p. 183-190. doi: 10.1109/CVPR.2010.5540212.

[121] C. E. Shannon, « Communication theory of secrecy systems », *Bell Syst. Tech. J.*, vol. 28, nº 4, p. 656-715, oct. 1949, doi: 10.1002/j.1538-7305.1949.tb00928.x.

[122] C. E. Shannon, « A Mathematical Theory of Communication », *Bell Syst. Tech. J.*, vol. 27, nº 3, p. 379-423, 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.

[123] X. Tong, M. Cui, et Z. Wang, « A new feedback image encryption scheme based on perturbation with dynamical compound chaotic sequence cipher generator », *Opt. Commun.*, vol. 282, nº 14, p. 2722-2728, juill. 2009, doi: 10.1016/j.optcom.2009.03.075.

[124] S. Etemadi Borujeni et M. Eshghi, « Chaotic image encryption system using phase-magnitude transformation and pixel substitution », *Telecommun. Syst.*, vol. 52, nº 2, p. 525-537, févr. 2013, doi: 10.1007/s11235-011-9458-8.

[125] X. Wang et K. Guo, « A new image alternate encryption algorithm based on chaotic map », *Nonlinear Dyn.*, vol. 76, nº 4, p. 1943-1950, juin 2014, doi: 10.1007/s11071-014-1259-7.

[126] Z. Hua, Y. Zhou, C.-M. Pun, et C. L. P. Chen, « 2D Sine Logistic modulation map for image encryption », *Inf. Sci.*, vol. 297, p. 80-94, mars 2015, doi: 10.1016/j.ins.2014.11.018.

[127] X. J. Tong, Z. Wang, M. Zhang, Y. Liu, H. Xu, et J. Ma, « An image encryption algorithm based on the perturbed high-dimensional chaotic map », *Nonlinear Dyn.*, vol. 80, nº 3, p. 1493-1508, mai 2015, doi: 10.1007/s11071-015-1957-9.

[128] C. Zhu, « A novel image encryption scheme based on improved hyperchaotic sequences », *Opt. Commun.*, vol. 285, nº 1, p. 29-37, janv. 2012, doi: 10.1016/j.optcom.2011.08.079.

[129] B. Norouzi, S. M. Seyedzadeh, S. Mirzakuchaki, et M. R. Mosavi, « A novel image encryption based on hash function with only two-round diffusion process », *Multimed. Syst.*, vol. 20, nº 1, p. 45-64, févr. 2014, doi: 10.1007/s00530-013-0314-4.

[130] M. Es-Sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « A new image encryption algorithm using random numbers generation of two matrices and bit-shift operators », *Soft Comput.*, vol. 24, nº 5, p. 3829-3848, mars 2020, doi: 10.1007/s00500-019-04151-8.

[131] X. Wang, L. Feng, R. Li, et F. Zhang, « A fast image encryption algorithm based on non-adjacent dynamically coupled map lattice model », *Nonlinear Dyn.*, vol. 95, nº 4, p. 2797-2824, mars 2019, doi: 10.1007/s11071-018-4723-y.

[132] C. Pak et L. Huang, « A new color image encryption using combination of the 1D chaotic map », *Signal Process.*, vol. 138, p. 129-137, sept. 2017, doi: 10.1016/j.sigpro.2017.03.011.

[133] M. Xu et Z. Tian, « A novel image cipher based on 3D bit matrix and latin cubes », *Inf. Sci.*, vol. 478, p. 1-14, avr. 2019, doi: 10.1016/j.ins.2018.11.010.

[134] S. F. Raza et V. Satpute, « A novel bit permutation-based image encryption algorithm », *Nonlinear Dyn.*, vol. 95, nº 2, p. 859-873, janv. 2019, doi: 10.1007/s11071-018-4600-8.

[135] S. K.u. et A. Mohamed, « A novel image encryption scheme using both pixel level and bit level permutation with chaotic map », *Appl. Soft Comput.*, vol. 90, p. 106162, mai 2020, doi: 10.1016/j.asoc.2020.106162.

[136] M. Roy, S. Chakraborty, K. Mali, D. Roy, et S. Chatterjee, « A robust image encryption framework based on DNA computing and chaotic environment », *Microsyst. Technol.*, vol. 27, nº 10, p. 3617-3627, oct. 2021, doi: 10.1007/s00542-020-05120-0.

[137] V. R. Folifack Signing, T. Fozin Fonzin, M. Kountchou, J. Kengne, et Z. T. Njitacke, « Chaotic Jerk System with Hump Structure for Text and Image Encryption Using DNA Coding », *Circuits Syst. Signal Process.*, vol. 40, nº 9, p. 4370-4406, sept. 2021, doi: 10.1007/s00034-021-01665-1.

[138] M. Benssalah, Y. Rhaskali, et K. Drouiche, « An efficient image encryption scheme for TMIS based on elliptic curve integrated encryption and linear cryptography », *Multimed. Tools Appl.*, vol. 80, nº 2, p. 2081-2107, janv. 2021, doi: 10.1007/s11042-020-09775-9.

[139] M. Es-Sabry, N. El Akkad, M. Merras, A. Saaidi, et K. Satori, « A new color image encryption algorithm using multiple chaotic maps with the intersecting planes method », *Sci. Afr.*, vol. 16, p. e01217, juill. 2022, doi: 10.1016/j.sciaf.2022.e01217.