l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

UNIVERSITÉ DE NANTES

# Thèse de Doctorat

Hassan NOURA

*Mémoire présenté en vue de l'obtention*
***du grade de Docteur de l'Université de Nantes***
*Sous le label de l'Université Nantes Angers Le Mans*

***Discipline : Electronique***
***Spécialité : Traitement du signal et des images***
***Laboratoire : IETR UMR 6164***

**Soutenance prévue le 31 août 2012**

**École doctorale Sciences et Technologies de l'Information et Mathématiques (STIM)**
**Thèse N° ED503-166**

# Conception et simulation des générateurs, crypto-systèmes et fonctions de hachage basés chaos performants

**JURY**

| | |
|---|---|
| Président & Rapporteur : | **M. René LOZI**, Professeur, Université de Nice Sophia-Antipolis |
| Rapporteur : | **Mme Danièle FOURNIER-PRUNARET**, Professeur, INSA Toulouse |
| Examinateurs : | **Mme Estelle CHERRIER**, Maître de Conférences, ENSICAEN, Caen<br>**M. Olivier DEFORGES**, Professeur, INSA Rennes<br>**Mme Caroline FONTAINE**, CR CNRS/HDR, Lab-STICC/Telecom Bretagne, Brest |
| Directeur de Thèse : | **M. Safwan EL ASSAD**, Maître de Conférences/HDR, Ecole polytechnique de l'université de Nantes |

Je dédie ce travail à :

**Ma mère,**

Maman, même si je ne t'ai pas vu beaucoup durant les années de thése, sache que tes célestes yeux parcourent ces lignes, et remplissent mon cœur de joie…….

**Mon Père,**

Papa, à travers ton regard plein de sagesse, tu es pour moi une force et une source inépuisable d'inspiration…….

**Mes frères et ma sœur et leurs fils**

Vous illuminez ma vie chaque jour de plus en plus …….

***Remerciements***

Une thèse de doctorat est le fruit d'un travail collectif, pour cela je souhaite adresser mes remerciements aux personnes suivantes :

M. René Lozi, Professeur à l'université de Nice Sophia-Antipolis, qui m'a fait l'honneur de présider ce jury et d'accepter la charge de rapporteur.

Mme Danièle Fournier-Prunaret, qui a également accepté la charge de rapporteur.

M. Olivier Deforges, Professeur à l'INSA Rennes ; Mme Caroline Fontaine, Chargé de recherche au Lab STICC/ Telecom Bretagne, Brest; Mme Estelle Cherrier, Maître de conférence à l'ENSICAEN CAEN, qui ont bien voulu juger ce travail.

J'exprime mes sincères remerciements au Directeur de cette thèse, Monsieur El Assad Safwan, Maître de conférences HDR à l'école polytechnique de l'université de Nantes, pour m'avoir initié au domaine de la sécurité de l'information. Aussi pour sa compétence, sa patience et pour son temps précieux consacré aux discussions scientifiques constructives tout au long de la thèse et à la correction de la rédaction.

Monsieur le Professeur Jean François Diouris de m'avoir accueilli dans le laboratoire IETR site de Nantes.

Monsieur le Professeur Mouin Hamzé, Directeur du CNRS de LIBAN et Monsieur Charles Tabet pour avoir financé en partie ma thèse.

J'aimerais remercier aussi, Monsieur le Professeur Serge Toutain pour ses conseils, son soutien et aide tout au long de mon séjour au laboratoire.

J'aimerais également remercier mes parents, mes frères Ali et Mohammad et ma sœur Diala pour leurs aides, leurs conseils et leurs soutiens indéfectibles tout au long de mes études.

Je tiens également à remercier Madame S. Charlier, Monsieur M. Brunet et toutes les personnes que j'ai côtoyées au sein du laboratoire pour leur soutien et aide.

Je remercie ma mère qui m'a supporté ces dernières années et qui m'a donné de l'énergie pour continuer dans les moments les plus difficiles.

J'exprime aussi mes remerciements aux amis : Ali EL Trabolsi , Hussein Ammar, Khaled Chahine , Ali EL Attar et Hassan Karaky. Avec eux j'ai partagé d'excellents moments.

Enfin je remercie vivement Monsieur Fawaz Ahmed et Docteur Fawaz Zachari pour leurs soutiens tout au long de mon séjour à Nantes.

# *Introduction générale*

La protection de l'information contre l'écoute et l'échange non autorisé est vitale, en particulier pour les applications militaires, médicales, et industrielle. De nos jours, les attaques cryptographiques sont de plus en plus nombreuses et sophistiquées, et de ce fait, de nouvelles techniques efficaces et rapides de protection de l'information sont apparues ou en cours d'élaboration. Les travaux de cette thèse se situent dans le contexte de la sécurité de données transmises ou stockées.

Les différents services de la sécurité des données sont la confidentialité, l'intégrité, l'authenticité, la non répudiation et le contrôle d'accès. En informatique, la confidentialité fait partie, avec l'intégrité, l'authentification, et le contrôle d'accès, des quatre grands types de sécurité informatique. La confidentialité de l'information est, en général, atteinte par des algorithmes de chiffrement. En revanche, l'intégrité et l'authenticité sont rendues possibles par les fonctions de hachage sans et avec clé respectivement. En outre, le contrôle d'accès est réalisé par une combinaison de chiffrement et de hachage.

Dans cette thèse, pour la partie chiffrement des données, nous nous appuyons sur la technique symétrique de chiffrement ou à clé secrète, qui est nettement plus rapide (jusqu'à 1000 fois) en comparaison avec la technique asymétrique. Cette dernière, utilise deux clés, une clé publique pour le chiffrement et une clé privée pour le déchiffrement.

Le chiffrement symétrique est réalisé, selon deux modes distincts :

Chiffrement par flux, qui opère sur un flux continu de données, adapté pour la communication en temps réel et réalisé en général sur des supports matériels.

Chiffrement par bloc, qui opère sur des blocs de données de taille fixe est réalisé en logiciel et en matériel.

Le niveau de sécurité apporté par le chiffrement par bloc est plus grand comparé à celui du chiffrement par flux (sauf dans le cas du masque jetable). De ce fait, nous nous focalisons sur la conception de crypto-systèmes par bloc.

Les crypto-systèmes par bloc sont en général construits à partir de deux structures dites réseaux SPN ou Feistel. Ces crypto-systèmes sont basés sur les deux concepts de base de la sécurité, qui sont la confusion et la diffusion [Shannon, 1949]. Le degré de résistance vis-à-vis des attaques cryptographiques, dépend donc essentiellement, pour chacune des deux structures, des performances de la diffusion, réalisée par une couche de substitution non linéaire (« S-boxes », fonctions itératives non linéaires) et de diffusion (linéaire), réalisée par une couche de permutation et ou par une couche de diffusion, permettant de répartir les caractéristiques cryptographiques de la couche de substitution. Par ailleurs, la quasi-totalité des crypto-systèmes contiennent aussi une couche d'addition de clé, afin d'initialiser le processus de confusion.

La répétition des processus des différentes couches un certain nombre de fois permet d'assurer l'immunité cryptographique contre les diverses attaques, incluant les cryptanalyses linéaire et différentielle.

La majorité des crypto-systèmes actuels utilise une couche de substitution et une couche de diffusion fixes très performantes. Seule la couche d'addition des clés dépend de la clé secrète, comme dans le standard international du chiffrement, AES (Advanced Encryption Standard). La connaissance des couches de substitution et de diffusion utilisées dans un crypto-système impose d'utiliser un niveau de complexité suffisant (nombre de rounds important des couches d'addition), afin de se prémunir contre les attaques standard. De ce fait, la taille de la clé secrète doit être assez grande, au moins égale à 128 bits, pour que le nombre de rounds soit assez grand, au moins égal à 10 (dans l'algorithme AES). Ceci, augmente le temps de traitement et par conséquent, empêche l'utilisation de certaines applications nécessitant de la protection en temps réel.

Les différentes couches des crypto-systèmes proposés dans cette thèse, sont variables selon les clés dynamiques, produites par des générateurs de séquences chaotiques. De ce fait, les crypto-systèmes développés sont à la fois robustes et rapides.

Les services d'intégrité et d'authentification sont en général réalisés par des fonctions de hachage qui sont de plus en plus utilisées. Les fonctions de hachage cryptographiques font correspondre à un message de taille arbitraire, une sortie de taille fixe, appelée haché ou empreinte digitale. L'idée sous-jacente est que toute modification, même infime, du message d'entrée, doit induire des modifications importantes et imprévisibles du haché en sortie. L'une des propriétés recherchées est qu'il doit être très difficile pour un attaquant de trouver une collision, c'est-à-dire de trouver une même empreinte pour deux messages distincts.

Ces fonctions sont considérées comme faisant partie de la cryptographie symétrique pour réaliser le service de l'intégrité s'il n'y pas de secret, et l'authentification s'il y a un secret partagé (clé secrète). Grâce à leur rapidité, elles sont très utilisées en sécurité informatique pour l'intégrité et la signature numérique des données, en passant par le stockage des mots de passe et leur génération.

Dans ce mémoire, nous nous intéressons à la conception et à la réalisation des fonctions de hachage basées chaos, avec ou sans clé, et aussi à leurs performances.

Les systèmes chaotiques sont déterministes, ils produisent des signaux de caractéristiques très proches des signaux aléatoires et possèdent une forte sensibilité aux conditions initiales et aux paramètres de contrôle qui forment la clé secrète. Un infime changement dans les conditions initiales ou les paramètres de contrôle provoque un important changement dans la séquence pseudo-chaotique générée par le système.

Fonds sur ces propriétés, les crypto-systèmes et fonctions de hachage basés chaos, sont une alternative intéressante à la cryptographie et fonction de hachage standards, pour achever les services de la sécurité décrits plus haut.

Les travaux réalisés et présentés dans ce mémoire, concernent la conception, la réalisation et l'évaluation des performances des générateurs de séquences chaotiques, de crypto-systèmes et fonctions de hachage basés chaos.

L'objectif principal de la thèse est d'apporter des solutions robustes basées chaos de la sécurité, comme la confidentialité, l'intégrité des données, ainsi que l'authentification de la source de ces données. Par ailleurs, les solutions proposées ont un niveau de sécurité aussi élevé que les méthodes standard, mais elles sont plus rapides, permettant ainsi une utilisation en temps réel, et une augmentation du débit des données. D'autre part, elles sont appropriées pour implémentation en logiciel et en matériel.

Dans cette optique, le travail s'articule autour des axes suivants :

D'abord, nous nous intéressons à la conception et à la réalisation de générateurs de séquences pseudo-chaotiques performantes utilisées comme clés dynamiques dans les crypto-systèmes basés chaos proposés ou pour la production des clés secrètes.

Puis, nous développons des couches de substitution, de permutation/confusion et d'addition des clés, toutes dynamiques, c'est-à-dire, variables en fonction de la sortie de la séquence pseudo-chaotique produite par le générateur utilisé. De ce fait, même si les couches de substitution/diffusion assuraient séparément des performances un peu plus faibles que les couches utilisées par l'AES par exemple, cela permettrait des gains de temps significatifs tout en conservant une sécurité globale identique voire supérieure à celles des systèmes actuels.

Ensuite, basés sur ces couches dynamiques nous concevons et réalisons des crypto-systèmes et fonctions de hachage avec ou sans clé, basés chaos, dynamiques très performants.

Les travaux de la thèse sont structurés comme suit :

Dans le chapitre 1, nous abordons la problématique de la sécurité de l'information et ses services. Puis, nous rappelons les principales notions relatives à la cryptographie, incluant une description sommaire du standard du chiffrement par bloc, l'algorithme AES.

Ensuite, nous exposons les différentes applications d'une fonction de hachage, comprenant, la structure du code d'authentification de message avec clé le plus utilisé, le HMAC.

Enfin, nous relatons l'état de l'art sur les crypto-systèmes basés chaos, après avoir mentionné les propriétés des signaux chaotiques, et nous concluons.

Le deuxième chapitre est consacré à la réalisation des générateurs de séquences pseudo-chaotiques en précision finie performants. A cet effet, tout d'abord, nous étudions les

performances de quelques cartes chaotiques connues (Logistique PWLCM, Skew tent, Cat), mais en précision finie. Puis, nous montrons l'effet de la précision finie sur les performances obtenues, et nous décrivons la procédure de mesure des orbites chaotiques. Ensuite, nous présentons la technique de perturbation pour pallier l'inconvénient de la précision finie et nous décrivons la structure des trois générateurs proposés ainsi que leurs performances, selon une panoplie de tests signal et NIST, avant de conclure.

Dans le troisième chapitre, nous nous intéressons à la conception et à la réalisation des crypto-systèmes basés chaos robustes et rapides. A ce sujet, nous rappelons tout d'abord, la structure des crypto-systèmes, et spécialement celle de type SPN. Puis, nous développons les différentes couches dynamiques basés chaos (addition de clé, substitution, permutation, diffusion), et les couches inverses. Basés sur ces couches, nous décrivons ensuite, les deux crypto-systèmes proposés, qui sont à la fois robustes vis-à-vis des attaques cryptographiques et adéquats pour des applications de confidentialité en temps réel. Dans notre conception, nous gardons à l'esprit, la nécessité de pouvoir implanter les algorithmes développés en logiciels et en matériels (cartes FPGA, Processeurs). Nous quantifions les performances, couche par couche et globalement, en nous appuyant sur la panoplie des tests existants dans la littérature ( bijectivité, non linéarité, corrélation, indépendance des bits produits en sortie, critère d'avalanche, sensibilité à la clé, temps de calcul, etc.), que nous avons implémentés. Enfin, nous analysons, les résultats obtenus, en démontrant l'intérêt de l'approche de crypto systèmes basés chaos proposés tant en termes de gain de temps qu'en termes de sécurité, avant de conclure.

Le quatrième chapitre, présente la structure et l'étude détaillées d'une nouvelle fonction de hachage basée chaos proposée performante, avec ou sans clé. Ceci, afin de réaliser respectivement, le service de l'authentification de la source des données, ou le service de l'intégrité des données. Pour cela, nous donnons tout d'abord, des généralités sur les fonctions de hachage et leurs propriétés. Ensuite, basée sur la structure de Merkle-Damgard, nous réalisons une fonction de hachage très performante, utilisant des couches de diffusion et confusion développées dans le chapitre précédent. Finalement, nous présentons et analysons les performances de la fonction de hachage proposée, avant de conclure sur ce chapitre.

# *Chapitre 1 : Contexte de l'étude, définitions et généralités*

## 1.1 Introduction

Ce chapitre, introduit les notions nécessaires à la compréhension des travaux réalisés dans cette thèse. Nous commençons par définir la sécurité de l'information et donnons les éléments de base la concernant. Puis, nous rappelons la définition des différents services de la sécurité, à savoir : la confidentialité, l'intégrité des données, l'authentification de la source de message, et la signature numérique. Comme la confidentialité est assurée, avec la stéganographie, par les algorithmes de chiffrement par flux et par bloc, nous présentons le standard international pour le chiffrement des données par bloc, l'algorithme AES, qui servira aussi comme élément de comparaison des performances avec les crypto-systèmes basés chaos développés dans le chapitre 3. De même, nous rappelons les autres applications d'une fonction de hachage (protection des mots de passe, dérivation des clés et génération des nombres pseudo-aléatoires) et nous donnons la structure du HMAC, le code d'authentification de message avec clé le plus utilisé. Sa structure servira aussi, comme élément de comparaison des performances des fonctions de hachages basées chaos développées dans le chapitre 4.

Nous mentionnons après les caractéristiques intéressantes des signaux chaotiques du point de vue de la sécurité de l'information. Nous décrivons ensuite, l'état de l'art des crypto-systèmes basés chaos, et nous pointons les faiblesses et points forts de quelques crypto-systèmes connus. Enfin, nous citons les propriétés des crypto-systèmes basés chaos proposés, et nous concluons.

## 1.2 Sécurité de l'information

La cryptologie peut se définir comme étant l'étude des communications dans un environnement non sécurisé [Menezes et al, 1997], [Schneider, 1996], [Katz et Lindell ,2007]. Elle a pour but d'assurer certains services de sécurité de l'information tels que : la confidentialité, l'intégrité et l'authentification. La cryptologie, appelée aussi science du secret regroupe la cryptographie et la cryptanalyse. Si le rôle des cryptographes est de construire et prouver des systèmes de chiffrement ou de signature, l'objectif des cryptanalystes est de "casser" ces systèmes. De façon complémentaire à la cryptographie, la cryptanalyse est l'étude des moyens pour faire échouer ou pour briser un procédé cryptographique.

Les procédés cryptographiques existent depuis fort longtemps, cependant, il faut attendre le développement de la théorie de l'information par [Shannon, 1949] pour avoir une véritable science cryptographique.

Actuellement, plusieurs systèmes cryptographiques sont apparus pour solutionner les nouveaux problèmes de sécurité tels que l'accès à l'information, les communications entre ordinateurs, le commerce électronique, etc.

Après cette définition nous allons préciser, dans les sections suivantes, les différents outils de services de sécurité que nous utilisons dans cette thèse, tels que les algorithmes de chiffrement et les fonctions de hachage avec ou sans clé. Pour le chiffrement, nous nous focalisons sur les crypto-systèmes symétriques par bloc.

### 3.1.2 Eléments de base

La cryptographie est l'étude des méthodes permettant de transmettre des données de manière confidentielle. Afin de protéger un message, on lui applique une transformation qui le rend incompréhensible ; c'est ce qu'on appelle le chiffrement, qui, à partir d'un texte en clair, donne un texte chiffré ou cryptogramme. Inversement, le déchiffrement est l'action qui permet de reconstruire le texte en clair à partir du texte chiffré. Dans la cryptographie moderne, les transformations en question sont des fonctions mathématiques, appelées algorithmes cryptographiques, qui dépendent d'un paramètre appelé clé secrète.

La clé est le secret partagé entre deux utilisateurs, qui leur permet de chiffrer/déchiffrer l'information. Le nombre de clés doit être suffisamment grand pour échapper à la recherche exhaustive. Actuellement, on estime que la longueur minimale d'une clé d'un système de chiffrement symétrique doit être de 128 bits. L'espace des clés comprend alors au moins $2^{128}$ clés différentes. Le choix de la clé secrète partagée par deux utilisateurs doit être parfaitement aléatoire.

La sécurité d'un crypto-système doit dépendre essentiellement de la clé secrète [Kerckhoffs, 1883].

La cryptanalyse mêle une intéressante combinaison de raisonnement analytique, d'application d'outils mathématiques, de découverte de redondances, de patience, de détermination, et de la chance. Les cryptanalystes sont aussi appelés attaquants que nous pouvons diviser en deux classes :

- un attaquant passif qui se contente d'écouter sur le canal de communication.
- un attaquant actif qui ne se contente pas seulement d'écouter la communication, mais cherche à intervenir directement sur le contenu de la communication.

### 3.1.3 Services de sécurité:

Le but de la cryptographie moderne est de traiter plus généralement des problèmes de sécurité des communications et de fournir un certain nombre de services de sécurité :

- la confidentialité ou masquage des données, vise à rendre le cryptogramme inintelligible pour celui qui n'est pas en possession de la clé,
- l'authentification permet au destinataire d'un message de s'assurer de l'identité de l'émetteur,

- l'intégrité permet au récepteur de s'assurer que le contenu du message n'a pas été falsifié depuis son écriture,
- la non répudiation, est la garantie qu'aucun des deux individus ayant effectué une transaction ne pourra nier avoir reçu ou envoyé les messages,
- le contrôle d'accès, est la faculté de limiter et de contrôler l'accès à des systèmes et des applications via des maillons de communications. Pour accomplir ce contrôle, chaque entité essayant d'obtenir un accès doit d'abord être authentifiée.

## 1.3 Confidentialité

C'est le service le plus général pour protéger les données transmises entre deux utilisateurs pendant une période donnée. Ce service est normalement assuré par un algorithme de chiffrement associé à une clé.

La sécurité des données chiffrées repose sur deux éléments :

L'invulnérabilité de l'algorithme de chiffrement et la confidentialité de la clé. On distingue deux catégories d'algorithmes de chiffrement : celle à clé secrète et celle à clé publique. Dans le cadre de notre travail, nous nous intéressons seulement au chiffrement symétrique, bien adapté pour le cryptage de grandes quantités de données (1000 fois plus que dans le cas du chiffrement asymétrique).

### 3.1.4 Algorithme de chiffrement symétrique

Le chiffrement symétrique peut être réalisé par bloc, ou par flux. Pour communiquer, il faut au préalable, échanger la clé entre les deux protagonistes. La figure 1.1, donne le schéma de principe du crypto-système symétrique.

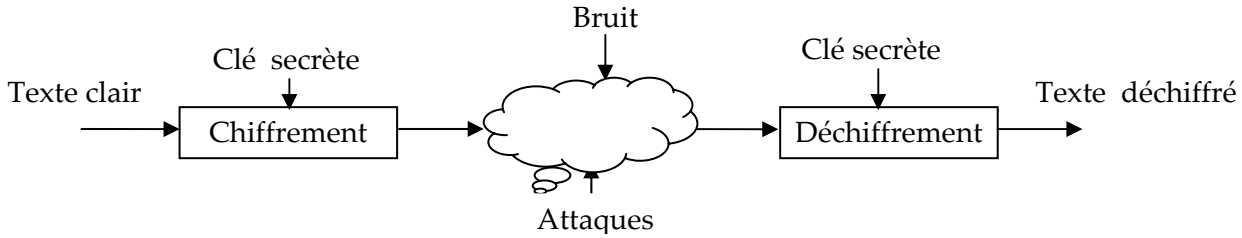



Fig. 1. 1 : Crypto-système symétrique

### 3.1.5 Chiffrement par flux

Le chiffrement par flux traite les données au fur et à mesure de leurs arrivées. Il se présente classiquement sous la forme d'un générateur de nombres pseudo-aléatoires dont la sortie est couplée via un XOR avec l'information à chiffrer. Le chiffrement se fait bit par bit octet par octet, ou bloc par bloc. Toutefois, le XOR n'est pas la seule opération possible. L'opération d'addition est également envisageable (par exemple, addition entre deux octets, modulo 256). Notons aussi, qu'un chiffrement par flux peut être réalisé par un chiffrement par bloc utilisant les modes opératoires CTR, ou OFB.

La sécurité d'un système de chiffrement par flux repose essentiellement sur les caractéristiques du générateur pseudo-aléatoire utilisé.

### 3.1.6 Types d'algorithme de chiffrement par bloc

Les algorithmes de chiffrement itératifs par blocs se répartissent essentiellement en deux grandes structures, selon l'architecture de la fonction d'itération interne. Les deux structures principalement utilisées sont le schéma de Feistel (utilisée dans le DES et le RC6), et la structure de réseau de substitution-permutation, SPN (utilisée dans l'AES). Par ailleurs, chaque structure peut être fixe (clé fixe pour chaque itération) ou variable (clé dynamique avec les itérations). La figure 1.2, résume les types d'algorithmes de chiffrement par bloc.

Il est clair que la structure variable, donc dépendante des clés d'itérations, est plus difficile à étudier par l'attaquant, et pour cela nous l'adoptons pour les crypto-systèmes basés chaos développés dans cette thèse.

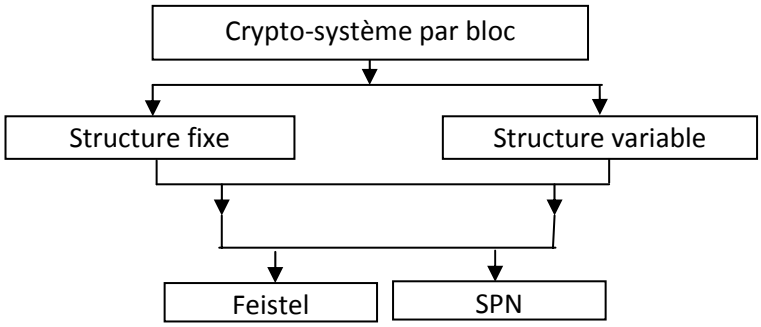



Fig. 1. 2 : Types d'algorithme de chiffrement par bloc

Comme l'algorithme AES, est le standard le plus utilisé pour le chiffrement par bloc, nous l'utilisons comme élément pour comparer les performances des algorithmes de chiffrement basés chaos proposés dans le chapitre 3. Pour cela, nous rappelons ci-dessous sa structure et son principe de fonctionnement.

### 3.1.7 Algorithme AES (Advanced Encryption Standard)

L'algorithme AES [Daemen et Rijmen, 1999], [Daemen et Rijmen, 2002] a été choisi en octobre 2000 parmi les 15 systèmes proposés en réponse à l'appel d'offre lancé par le NIST (National Institute of Standards and Technology) [NIST FIPS 197, 2001]. Cet algorithme, initialement appelé Rijndael, a été conçu par deux chercheurs belges, Rijmen et Daemen. L'algorithme se présente en deux temps, tout d'abord une procédure d'expansion de la clé (dérivations des clés des tours), puis la fonction principale de chiffrement.

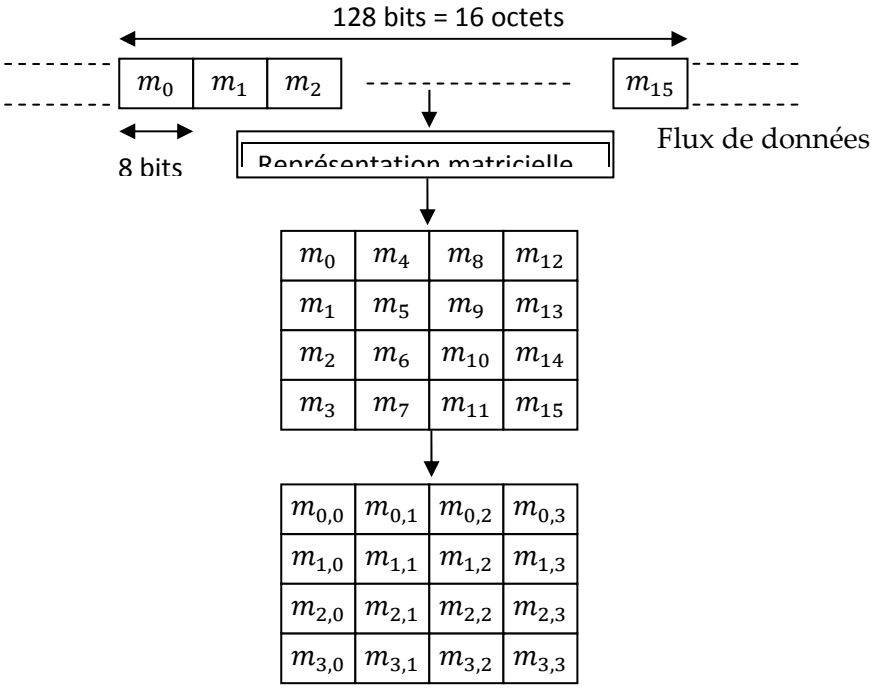



Fig. 1. 3 : Représentation matricielle des valeurs intermédiaires de l'AES

L'algorithme AES permet de chiffrer des blocs de taille 128 bits (16 octets) avec des clés de 128, 192 ou 256 bits. Le flux d'octets est organisé sous forme matricielle, selon le modèle illustré dans la figure 1.3

De même, la matrice associée à une clé aura :

- 4 lignes et 4 colonnes pour une clé de taille 128 bits,
- 4 lignes et 6 colonnes pour une clé de taille 192 bits,
- 4 lignes et 8 colonnes pour une clé de taille 256 bits.

Le nombre d'itérations $r$ dépend de la taille de clé :

- 10 itérations si la taille de la clé est de 128 bits,
- 12 itérations si la taille de la clé est de 192 bits,
- 14 itérations si la taille de la clé est de 256 bits.

Nous décrirons le principe de l'AES dans sa version à clé de 128 bits qui comprend 10 itérations et qui est actuellement la plus utilisée (voir figure 1.4).
Le chiffrement consiste en une addition initiale de clé, notée AddroundKey, suivie par $r - 1 = 9$ itérations, chacune constituée de quatre étapes [Daemen et Rijmen, 2002] :

- **SubBytes** : opération de substitution non linéaire lors de laquelle chaque octet est remplacé par un autre octet choisi dans une table particulière, une boîte-S ;

- **ShiftRows** : est une étape de transposition où chaque élément de la matrice est décalé cycliquement à gauche d'un certain nombre de colonnes ;
- **MixColumns** : effectue un produit matriciel en opérant sur chaque colonne (vue alors comme un vecteur) de la matrice ;
- **AddRoundKey** : qui combine par addition chaque octet avec l'octet correspondant dans une clé d'itération obtenue par diversification de la clé de chiffrement.

Enfin, une itération finale est appliquée, elle correspond à une itération dans laquelle l'opération MixColumns est omise.

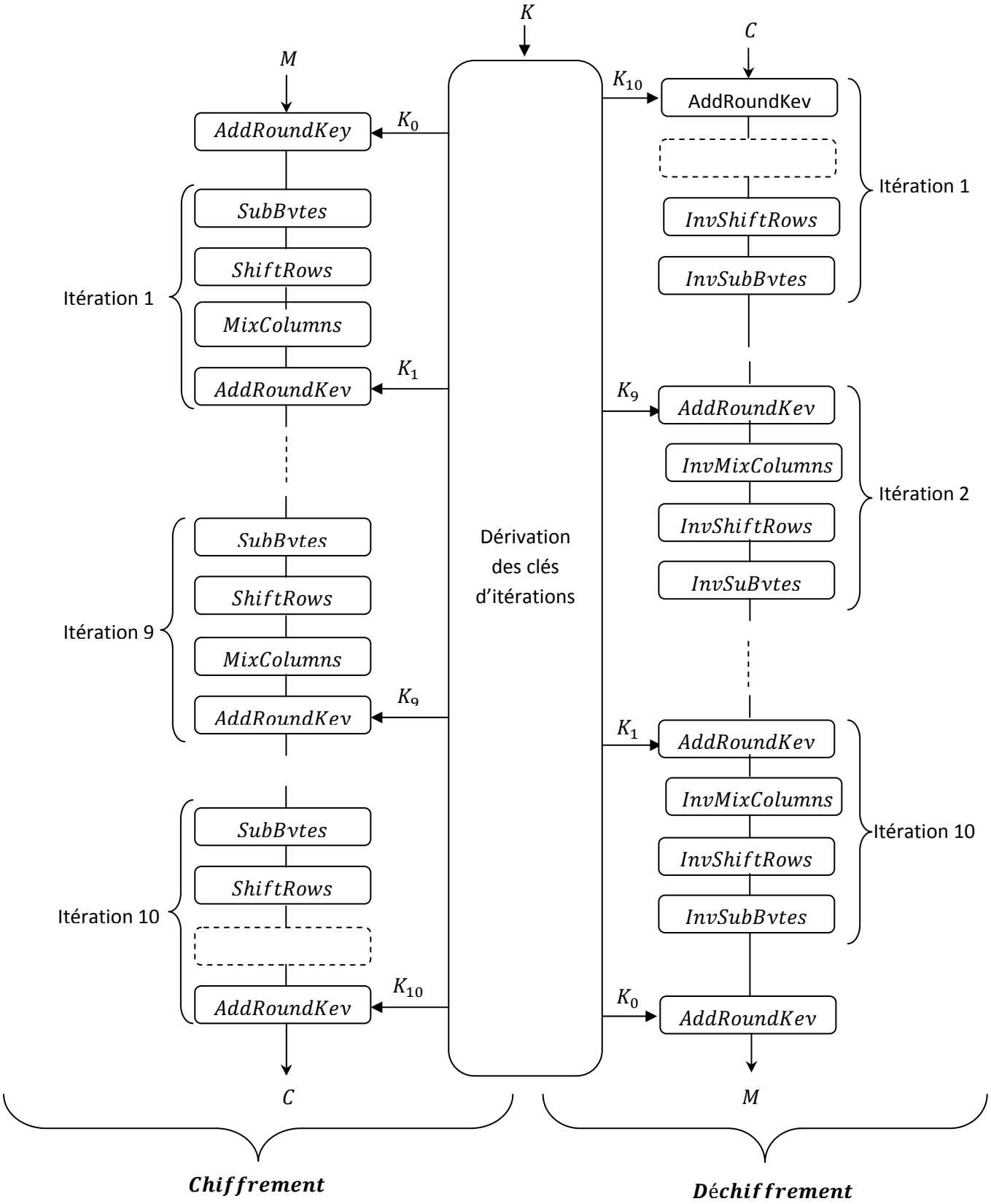



Fig. 1. 4 : Etapes de l'algorithme AES

## 1.4 Fonctions de hachage

Une fonction de hachage est un système faisant correspondre à un message $M$ de taille arbitraire, un message haché, ou empreinte digitale, de taille $n$ bits, nettement inférieure à la taille du message. En pratique, $n$ est typiquement 128, 256, 512 et 1024 bits [RFC 1321, 1992], [NIST FIPS 180-3, 2008].

Les fonctions de hachage cryptographiques sont à sens unique, et elles possèdent des nombreux domaines d'utilisation. Nous donnons ci-dessous une liste non exhaustive de ces domaines.

### 3.1.8 Intégrité des données.

Un des services de la sécurité est l'intégrité d'un message. On veut pouvoir détecter toute modification, accidentelle ou intentionnelle, des données sauvegardées ou transmises. La réponse à cette question est la fonction de hachage sans clé [Knuth, 1998]. Il s'agit de la fonctionnalité principale demandée à une fonction de hachage cryptographique. Le moindre changement dans les données transmises doit aboutir, avec une très grande probabilité, a l'obtention d'empreintes différentes.

### 3.1.9 Authentification des messages.

La propriété d'intégrité ne permet pas de se prémunir contre un adversaire actif qui essaierait d'altérer malicieusement les données. Un moyen de palier ce problème, consiste à utiliser une fonction de hachage avec clé, permettant ainsi l'authentification de la source des données (codes d'authentification de messages (MAC : Authentication Codes)), et l'intégrité des données. Une des techniques d'authentification la plus répandue est le HMAC [Bellare et al, 1196], mais on peut aussi utiliser un algorithme de chiffrement en bloc en mode CBC, noté CMAC, tel que le dernier bloc chiffré est l'empreinte digitale. Cependant, le CMAC [Dworkin, 2005 ] est nettement plus lent que le HMAC.

Un HMAC, est un code d'authentification de message avec clé. Il utilise une fonction de hachage cryptographique en combinaison avec une clé secrète. N'importe quelle fonction itérative de hachage standard, comme le SHA-256,-512, ou chaotique comme celle que nous proposons au chapitre 4, peut être utilisée dans le calcul d'un HMAC. La qualité cryptographique du HMAC dépend de la qualité cryptographique de la fonction de hachage, de la taille et la qualité de la clé.

L'équation du HMAC est donné par:

$$HMAC(K, M) = H\left((K \oplus opad) \| H\left((K \oplus ipad) \| M\right)\right) \tag{1.1}$$

avec :

$H$ : une fonction de hachage itérative, standard ou chaotique

$K$ : la clé secrète de taille identique à la taille du bloc de la fonction $H$, (si la taille de $K$ est inférieure à la taille du bloc, alors on complète par des zéros)

$M$ : le message à authentifier,

"||" désigne une concaténation et "⊕" un ou exclusif,

$ipad$ et $opad$, sont des constantes, chacune de taille identique à celle d'un bloc. Elles sont définies par : $ipad$ = *0x363636...3636* et $opad$ = *0x5c5c5c...5c5c*. Cela veut dire que, si la taille du bloc de la fonction de hachage est 1024 bits, alors $ipad$ et $opad$ sont constitués de 128 répétitions des octets, respectivement, $0x36$ et $0x5c$.

Pour plus de clarté, nous donnons ci-dessous, le schéma de réalisation du HMAC

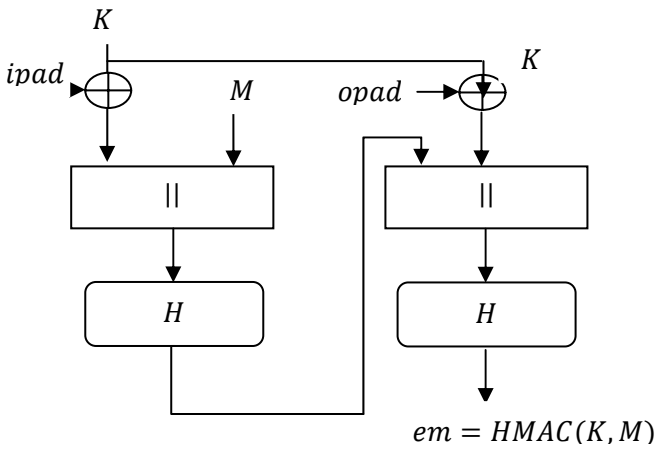



Fig. 1. 5 : Génération de l'empreinte digitale par HMAC

### 3.1.10 Signature électronique

Un schéma de signature est une application importante des fonctions de hachage. Il permet à un utilisateur de signer un message à l'aide de sa clé privée. Une signature électronique est un équivalent électronique d'une signature écrite [NIST FIPS 186-3, 2009]. Chacun peut vérifier la validité de cette signature grâce à la clé publique correspondante. Elle permet de plus, de détecter si l'information signée a été altérée après sa signature. Les opérations internes de ces primitives cryptographiques sont en général très coûteuses, car elles appartiennent à la cryptographie asymétrique. Ainsi, leur application à un très long message demande un temps de calcul trop grand dans certains cas pratiques. Les fonctions de hachage sont donc utilisées pour raccourcir le message à signer et améliorer les performances. On signe le haché du message plutôt que le message. Dans cette situation, on souhaite qu'un attaquant susceptible d'obtenir les signatures de certains messages choisis soit incapable de créer une nouvelle signature valide sans connaître la clé privée de l'utilisateur. Aussi, étant donnés plusieurs couples message/signature, il doit être

extrêmement difficile pour lui de deviner la moindre information sur la clé privée. Il est donc important que les fonctions de hachage soient résistantes à la recherche de collisions [ Rivest et al, 1978] .

### 3.1.11 Protection de mots de passe

Une autre application des fonctions de hachage cryptographiques est la protection de mots de passe. Un mot de passe est une chaine de caractères utilisée pour authentifier l'identité d'un utilisateur ou autoriser l'accès aux ressources d'un système informatique. Il est nécessaire de protéger les mots de passe afin de les stocker. Une solution courante consiste à ne stocker que leur empreinte calculée en appliquant une fonction de hachage sur les mots de passe.

Par exemple, dans un serveur, au lieu de stocker tous les mots de passe d'utilisateurs, il est préférable de stocker les hachés de ces derniers. L'authentification peut toujours avoir lieu, mais, si le serveur est compromis, l'attaquant n'a accès qu'aux hachés des mots de passe. Il ne peut donc théoriquement pas retrouver les mots de passe originaux à cause de la propriété de résistance à la recherche de préimages.

### 3.1.12 Dérivation de clé

Dans le cadre de la cryptographie symétrique, les parties partagent une clé sécrète commune. Il est alors fréquent que différentes clés supplémentaires soient nécessaires pour différentes applications. La dérivation de clés (ou diversification de clés) consiste à générer une ou plusieurs clés à partir d'une même valeur secrète. Cette utilisation des fonctions de hachage a pour bût d'empêcher un adversaire ayant obtenu une clé dérivée d'obtenir des informations sur la valeur secrète ou les autres clés dérivées.

### 3.1.13 Génération de nombres pseudo-aléatoires

Une classe de générateurs pseudo-aléatoires est basée sur une fonction de hachage cryptographique utilisant essentiellement deux modes opératoires. Le premier mode consiste à calculer de façon chaînée à partir d'une condition initiale (graine), les différentes empreintes constituant la séquence pseudo-aléatoire du générateur. Le second mode consiste à calculer les empreintes à partir de la graine et d'un compteur (mode compteur), nettement plus rapide que le premier mode.

## 1.5 Chaos et cryptographie

Au cours des dernières décennies, le comportement dynamique des systèmes non linéaires a suscité d'énormes intérêts pratiques. Le chaos peut être généré par un système dynamique non-linaire de structure assez simple. En effet, une simple équation de récurrence peut produire des dynamiques chaotiques assez complexes et riches.

Les signaux chaotiques possèdent de nombreuses caractéristiques distinctes, telles que:

- Large bande comme le bruit blanc.
- Auto-et inter corrélations proches de celles des signaux aléatoires.
- Pseudo-aléatoire et imprévisible à long terme
- Déterministe
- Haute sensibilité aux conditions initiales et aux paramètres du système (la clé secrète).
- Ergodique, c.-à-d, la plupart des orbites conduisent à la même distribution.

Plusieurs systèmes de cryptographie chaotique ont été proposés depuis [Yang et al, 1998], où, globalement, le message d'origine (en clair) est mélangé (addition, injection) avec un signal chaotique pour former le message chiffré, pour être transmis par un canal non sécurisé. Un processus inverse est placé à la réception pour déchiffrer le message chiffré et obtenir ainsi le message en clair. Dans l'application de chiffrement basé chaos, le générateur chaotique est un élément central. En effet, contrairement à la cryptographie standard, les opérations d'addition de clés, de substitution, de permutation et de diffusion se font selon des clés dynamiques, générées par le générateur de séquences pseudo-chaotiques.

## 1.6 Etat de l'art des crypto-systèmes par bloc basés chaos

La majorité des crypto-systèmes basés chaos ont été conçus en utilisant la structure de substitution-permutation proposée par [Fridrich, 1998]. Dans la quasi-totalité des crypto-systèmes basés chaos, ces couches sont réalisées par des cartes chaotiques mono et bidimensionnelles.

Après, la couche d'addition de clé, la couche de substitution, qui est une opération non linéaire (réalisée par des tables de substitution statique et dynamique, ou par des fonctions non linéaires), permet de substituer séquentiellement les valeurs des pixels, de sorte qu'un changement d'un bit dans un pixel, provoque le changement de valeurs de pratiquement 50% des pixels dans l'image chiffrée.

Dans la couche de permutation, les octets changent de positions mais pas de valeurs. Cette opération est souvent réalisée par les cartes chaotiques suivantes : Cat [Guanrong et al, 2004] , Baker [Gotz et al, 1997] , Standard [Lian et al , 2005] .

La littérature spécialisée est extrêmement riche en la matière, pour cela, nous décrivons très brièvement, quelques crypto-systèmes basés chaos qui nous semblent les plus efficaces parmi d'autres.

[Lian et al , 2005] ont indiqué qu'il existe quelques clés faibles de la carte Cat et Baker pour le processus de diffusion. De plus, l'espace de clé de ces deux cartes n'est pas assez grand comme celle de la carte standard. Ils ont suggéré d'utiliser la carte standard pour la

permutation, et la carte logistique pour la génération de clé d'addition. Pour atteindre un niveau satisfaisant de sécurité, ils ont recommandé d'exécuter au moins quatre itérations globales de substitution-diffusion. Dans chaque itération, ils itèrent une fois le processus de substitution, et 4 fois le processus de permutation. Donc, au total, pour chiffrer un bloc, l'algorithme exécute 4 itérations de substitution et 16 itérations de permutation. Donc, la vitesse de chiffrement est assez lente. En effet, la carte standard est extrêmement coûteuse en réalisation logicielle et matérielle.

Dans [Guanrong et al, 2004], [Di et al, 2007] les auteurs ont proposé l'utilisation de la carte chaotique Cat 3D pour d'abord permuter les pixels de l'image toute entière, ensuite, ils enchainent des opérations d'additions modulo et OUex sur les octets, répétées plusieurs fois pour achever le chiffrement.

Dans [Kai et al, 2005], les auteurs montrent que la méthode précédente n'est pas robuste contre l'attaque par texte clair choisi/connu. En effet, l'attaquant peut réussir à retrouver les clés d'addition, en utilisant une image de valeurs zéro.

Pour résister à l'attaque à texte clair choisi/connu, plusieurs solutions ont été proposées : [Wong et al, 2008] et [Wang et al, 2009].

La solution de [Wong et al, 2008] consiste à relier le processus de permutation de bloc sous traitement avec le bloc chiffré précédent. Le bloc chiffré devient dépendant de la clé sécrète et aussi du texte en clair.

La solution présentée dans [Wang et al, 2009], repose sur l'utilisation de paramètres de contrôle variables et non statiques. Ceci est considéré comme une version améliorée de l'architecture de substitution-permutation où les clés d'itérations deviennent dynamiques.

Dans [Socek et al, 2005], un algorithme de chiffrement d'image robuste, appelé ECKBA est proposé. La faiblesse de cet algorithme est sa rapidité, et aussi sa sensibilité en réception aux erreurs de propagation.

[Yang et al, 2010], ont proposé une architecture intéressante d'une solution permettant de réaliser à la fois le service de confidentialité et le service d'authentification. En plus, il est plus rapide que les crypto-systèmes cités plus haut. Le crypto-système en question utilise une clé de chiffrement qui dépend de la clé secrète proprement dit et du texte en clair.

L'algorithme de chiffrement proposé par [Kumar et Ghose, 2010] est constitué de deux étapes. La première étape, consiste en un mélange XOR de toute l'image avec une séquence pseudo-aléatoire $S_1$, générée par la carte standard associée au tableau de substitution de l'algorithme AES. Ensuite, une opération de permutation à base d'un registre à décalage avec retour, et une opération de décalage sur les lignes et colonnes sont réalisés.

La deuxième étape applique un XOR sur le résultat de l'étape précédente, utilisant la deuxième séquence pseudo aléatoire $S_2$ , générée de la même manière que la séquence $S_1$. En plus, l'enchainement des opérations se fait en mode CBC sur les lignes et colonnes.

La première étape est itérée $nr$ fois et la deuxième étape est itérée $nd$ fois.

## 1.7 Propriétés des crypto-systèmes proposés

Les crypto-systèmes proposés possèdent en plus de la robustesse vis-à-vis des attaques cryptographiques, les caractéristiques suivantes :

- le calcul est prévu pour être réalisé en précision finie (Valeurs entières sur $N = 32$ bits), ceci facilite la réalisation matérielle et accélère la vitesse de chiffrement/déchiffrement, en logiciel et matériel

- le traitement est fait sur des tailles flexibles de blocs : 64, 128, 256, 512 et 1024 bits, selon la contrainte de l'application envisagée.

- l'utilisation de transformation non linéaire et linéaire dans les couches de substitutions et de diffusion ne nécessite pas de mémoire, afin de tenir compte des contraintes données liées à des applications données.

- la structure d'une itération des crypto-systèmes est construite de sorte que les couches d'addition de clé et substitution soient réalisées en parallèles. Cependant, la couche de permutation de bits est limitée pour le traitement en parallèle. Pour contourner cette contrainte, une solution est présentée dans le deuxième crypto-système.

## 1.8 Conclusion

Dans ce chapitre, nous avons introduit les définitions, généralités et systèmes, permettant de situer le contexte de l'étude et comprendre la suite des travaux. Nous avons tout d'abord, rappelé les différents services de la sécurité à savoir : la confidentialité, obtenue essentiellement par les crypto-systèmes; l'intégrité des données, l'authentification de la source de message, obtenues par les fonctions de hachages et signature numérique qui nécessite des fonctions de hachage associées à des algorithmes de chiffrement asymétriques. Puis, nous avons rappelé l'essentiel sur l'algorithme AES de chiffrement par bloc, qui est le standard international. Aussi, nous avons rappelé les autres applications d'une fonction de hachage (protection des mots de passe, dérivation des clés et génération des nombres pseudo-aléatoires) et montré la structure du HMAC (un code d'authentification de message avec clé).

Ensuite, après avoir mentionné les caractéristiques des signaux chaotiques, nous avons décrit l'état de l'art des crypto-systèmes basés chaos, et pointé les faiblesses et points forts de quelques crypto-systèmes connus. Enfin, nous avons cité les propriétés des crypto-systèmes basés chaos proposés, avant de conclure.

[Bellare et al, 1996] M. Bellare, R. Canetti, H. Krawczyk, "Message authentication using hash functions: The HMAC construction", CryptoBytes, Spring 1996.

[Daemen et Rijmen, 1999] J. Daemen, V. Rijmen , "AES proposal: the Rijndael block cipher", 1999.

[Daemen et Rijmen, 2002] J. Daemen, V. Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard", Springer, ISBN 3-540-42580-2, 2002.

[Di et al, 2007] X. Di, L. Xiaofeng, W. Pengcheng, "Analysis and improvement of a chaos-based image encryption algorithm", Chaos Soliton Fract, December 2007.

[Dworkin, 2005] M. Dworkin, National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special", Publication 800-38B, May 2005.

[Fridrich, 1998] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps", International Journal Bifurcation Chaos, Vol. 8, no. 6, pp. 1259–84, 1998.

[Gotz et al, 1997] M. Gotz, K. Kelber, W. Schwarz, "Discrete-time chaotic encryption systems. I. Statistical design approach", IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications, Vol. 44, no. 10, pp. 963-970, 1997.

[Guanrong et al, 2004] C. Guanrong, M. Yaobin, C. Charles, "A symmetric image encryption scheme based on 3D chaotic cat maps", Chaos Soliton Fract, Vol 21, pp. 749–61, 2004.

[RFC 1321, 1992] R. Rivest, "The MD5 message-digest algorithm", IETF Network Working Group, RFC 1321, 1992.

[Kai et al, 2005] W. Kai, P. Wenjiang, Z. Liuhua, S. Aiguo, H. Zhenya, "On the security of 3D cat map based symmetric image encryption scheme", Phys Lett A , pp. 432–439, 2005.

[Katz et Lindell, 2007] J. Katz, Y. Lindell, "Introduction to Modern Cryptography", CRC Press, 2007.

[Kumar et Ghose, 2010] A . kumar, M. k. Ghose, "Extended substitution-diffusion based image cipher using chaotic standard map", Commun Nonlinear Sci Numer Simulat, Vol. 16, no. 1, pp. 372–382, January 2011.

[Knuth, 1998] D. Knuth, "The Art of Computer Programming, Sorting and Searching", second ed, Vol. 3 Addison-Wesley, 1998.

[Lian et al, 2005] S. Lian, J. Sun, Z. Wang, "A block cipher based on a suitable use of the chaotic standard map", Chaos Soliton Fract, Vol 26, pp. 117–29, 2005.

[Menezes et al, 1997] A.J. Menezes, P.C. van Oorschot, S. A. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1997.

[NIST FIPS 197, 2001] NIST FIPS 197, "Advanced Encryption Standard _ Federal Information Processing Standards Publication 197", 2001.

[NIST FIPS 186-3, 2009] NIST FIPS 186-3, "Digital Signature Standard (DSS)", June 2009.

[NIST SP800-90, 2007] NIST Special Publication 800-90, ”Recommendation for Random Number Generation Using Deterministic Random Bit Generators”, Revised Mars 2007.

[NIST FIPS 180-3, 2008] NIST FIPS 180-1, ”Secure Hash Standard”, Federal Information Processing Standard, Publication 180-3, National Institute of Standards and Technology, US Department of Commerce, Washington DC, 2008.

[Rivest et al, 1978] R. Rivest, A. Shamir; L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Vol. 21, no. 2, pp. 120-126, 1978.

[Schneider, 1996] B. Schneider, "Applied Cryptography", John Wiley & Sons, ISBN 0-471-12845-7, 1996

[Shannon, 1949] C. E. Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol. 28, no. 4, pp. 656-715, 1949.

[Socek et al, 2005] D. Socek, S. Li, S. Magliveras, B. Furht, “Enhanced 1-D Chaotic Key Based Algorithm for Image Encryption”, IEEE Security and Privacy for Emerging Areas in Communications Networks, pp. 406-408, 2005.

[Stinson, 2006] D. R. Stinson, “Cryptography, Theory and Practice”, Third edition. Chapman & Hall/CRC, 2006.

[Wang et al, 2009] Y. Wang, K. W. Wong, X. Liao, T. Xiang, G. Chen, “A chaos-based image encryption algorithm with variable control parameters”, Chaos, Solitons & Fractals, Vol. 41, no. 4, pp. 1773-1783, August 2009.

[Wong et al, 2008] K. Wong, S. K. Bernie, W. Law, “A fast image encryption scheme based on chaotic standard map”, Phys Lett A, Vol. 372, pp. 2645–2652, 2008.

[Yang et al, 1998] T. Yang, L.O. Chua, “Application of chaotic digital code-division multiple access (CDMA) to cable communication systems”, International Journal of Bifurcation and Chaos, Vol. 8, no. 8, pp. 1657-1669, 1998.

[Yang et al, 2010] H. Yang, K.-W. Wong, X. Liao, W. Zhang, P. Wei, “A fast image encryption and authentication scheme based on chaotic maps“, Communications in Nonlinear Science and Numerical Simulation, Vol. 15, no. 11, pp. 3507-3517, November 2010.

# *Chapitre 2 : Conception et réalisation des générateurs chaotiques performants*

## 2.2 Introduction

Les générateurs chaotiques peuvent être utilisés, entre autres, dans les applications touchant à la sécurité de l'information, pour la génération de clés secrètes dynamiques dans les algorithmes de chiffrement, de stéganographie, et tatouage numérique. Rappelons, que le chaos peut être généré par tout système dynamique non-linéaire. En effet, des simples équations de récurrence sont capables de générer des dynamiques chaotiques riches, si les paramètres de contrôle sont bien positionnés. Dans beaucoup d'équations de récurrences simples, le bon choix de ces paramètres se fait grâce au diagramme de bifurcation et à l'exposant de Lyapunov.

Dans la littérature, il y a énormément de cartes et générateurs chaotiques mono et multidimensionnels qui sont utilisés dans diverses applications telles que : sources pseudo-aléatoires, communication et sécurité de l'information, contrôle et automatique, etc…

L'implémentation pratique de ces cartes et générateurs chaotiques (calcul numérique en virgule flottante) engendre certaines faiblesses liées à la dégradation des dynamiques chaotiques telles que : périodicité, points fixes pour certaines valeurs de la clé secrète, non uniformité, temps de calcul important.

La quantification des performances des cartes et générateurs de séquences chaotiques se fait grâce à des tests au niveau du signal (plan de phase, auto et inter-corrélation, histogramme, mesure des orbites (cycles + transitoires), etc.) et NIST.

Nous proposons dans ce chapitre la conception et la réalisation de trois générateurs de séquences chaotiques très performants. Les séquences pseudo-chaotiques produites par ces générateurs ont une distribution uniforme, des orbites longues et vérifiant les tests signal et NIST. Par ailleurs, les générateurs en question sont flexibles et possèdent une clé secrète de taille très grande permettant de résister à l'attaque exhaustive.

Les générateurs proposés s'appuient sur des structures intégrant une technique de perturbation et des techniques de couplage entre des cartes chaotiques de base (Skew tent, PWLCM, Cat). Ceci permet d'atteindre de façon efficace la propriété de confusion-diffusion souhaitée par tout générateur destiné à être utilisé dans des applications de sécurité de l'information.

Le chapitre est organisé comme suit :

Dans la section 2.2, nous rappelons quelques domaines d'application des cartes et générateurs chaotiques, et dans la section 2.3, nous présentons les outils de quantification des performances des cartes et générateurs chaotiques.

Une étude des performances des cartes chaotiques discrètes unidimensionnelle (logistique, Skew tent, PWLCM,), en précision finie sur $N = 32$ bits et bidimensionnelle (Cat) est présentée dans la section 2.4.

Dans la section 2.5, nous présentons l'effet de la précision finie sur les performances, la procédure de mesure des orbites chaotiques, et nous décrivons la technique de perturbation utilisée pour pallier de façon efficace l'inconvénient de la précision finie.

En section 2.6, nous décrivons les structures des générateurs proposés et leurs performances. Chacune des structures intègre une même technique de perturbation de l'orbite chaotique, mais diffère de technique de couplage utilisée.

Le premier générateur utilise un couplage à base de fonctions booléennes non linéaires simples.

Le deuxième générateur s'appuie sur une couche de diffusion (carte Cat multidimensionnelle) globale, flexible et efficace comme technique de couplage.

Le troisième générateur utilise le couplage en parallèle de deux filtres récursifs.

Dans la section 2.7, nous présentons des résultats comparatifs pour des générateurs de séquences pseudo-aléatoires prouvés par NIST.

## 2.3 Applications des cartes chaotiques

Les cartes chaotiques sont utilisées dans différentes applications liées à la sécurité de l'information telles que : le chiffrement par flux [Li et al, 2001-1], et par bloc [Jakimoski et al, 2001], le hachage [Xiao et al, 2008], la stéganographie et le tatouage numérique [Wu et Guan, 2007], [Mooney, 2009].

Les cartes chaotiques sont des candidats potentiels en tant que générateurs de nombres pseudo aléatoires et peuvent supplanter les générateurs pseudo-aléatoires traditionnels tels que : les séquences PN à longueur maximale, les générateurs de Gold et de Kasami, etc.

En effet, les signaux pseudo-chaotiques possèdent des caractéristiques très intéressantes pour la sécurité et pour les communications numériques telles que : bonnes propriétés cryptographiques, reproductibilité à l'identique (déterministes), sensibilité extrême aux conditions initiales et aux paramètres du système (clé secrète), spectre large bande, auto et inter-corrélation similaires aux signaux pseudo-aléatoires, nombre des codes pour l'étalement des spectres très grand, ergodicité, etc.

Plusieurs générateurs à base de cartes chaotiques ont été développés dans la littérature depuis 1990. Sur le plan national, nous citons par exemple, les travaux de [Lozi, 2006], [Lozi, 2008, [Fournier, et al 2011], [Zheng et al, 2009], [Taralova et Fournier, 2002], [Grapinet et al, 2008], [Cherrier et al, 2010], [Millerioux et Guillot, 2010], etc.

Sur le plan international, la littérature est très abondante et nous ne citons que quelques travaux dont nous nous sommes inspirés pour la mise en œuvre pratique des générateurs chaotiques proposés : [Jessa et Walentynowicz, 2001], la carte Logistique [Phatak et al, 1995], la carte de Skew tent [Kocarev et Jakimoski ,2003], [Addabbo et al, 2004], la carte PWLCM [Li et al, 2001-1], [El Assad et al, 2008] et la carte Cat [Kwok et Tang, 2007].

## 2.4 Outils de mesure des performances des signaux chaotiques générés

Afin de quantifier les propriétés cryptographiques des signaux pseudo-chaotiques générés, une série de tests doit être appliquée. Chacun des tests mesure une caractéristique particulière comme l'étalement de spectre des signaux ou l'uniformité de la distribution par exemple, et l'ensemble des résultats de ces tests donne une idée du degré de l'aléatoire des signaux produits. Le comportement pseudo-aléatoire des signaux engendrés, est étroitement lié aux caractéristiques statistiques de ces signaux. Les tests du NIST [NIST SP 800-22, 2008] parmi d'autres [Marsaglia, 1996], [ENT] servent de référence pour quantifier et comparer les propriétés cryptographiques des signaux pseudo-aléatoires binaires. Par ailleurs, dans le cas des générateurs de structures simples, les tests de l'exposant de Lyapunov et du diagramme de bifurcation, permettent de fixer les valeurs optimales des paramètres pour atteindre les régions chaotiques.

Tous ces tests sont nécessaires, cependant, ils ne permettent pas d'affirmer à 100% la robustesse des signaux générés, à cause de la cryptanalyse de plus en plus efficace.

## 2.5 Cartes chaotiques

Les cartes chaotiques sont des systèmes dynamiques définis en réel par des relations de récurrence :

$$x_i(n) = f\big(x_1(n-1), x_2(n-1), \dots, x_m(n-1)\big), \qquad i = 1, 2, \cdots m \tag{2.1}$$

où $x \in S, f: S^m \to S^m$ est une fonction de $m$-dimensions, $S^m \subset [0,1]^m$ ou $[-1,1]^m$ .

Certaines cartes chaotiques monodimensionnelles, comme la carte Logistique, la carte PWLCM (Piecewise Linear Chaotic Maps), et la carte Skew tent, [Kocarev et al, 2003], [Phatak et al, 1995], [Li et al, 2001-1] et des cartes chaotiques bidimensionnelles telles que : les cartes Cat, Standard, Hénon et Lozi [Fridrich, 1998], [Lozi, 2008], [Lozi et Cherrier ,2011] sont bien étudiées dans la littérature et largement utilisées pour la conception de générateurs de nombres aléatoires et comme fonctions de substitution, de permutation, voire de diffusion, dans les différentes couches des crypto-systèmes basés chaos.

Cependant, la ou les variables dynamiques de ces cartes de la littérature évoluent dans l'intervalle [0, 1] ou [-1,1].

Les générateurs chaotiques (ainsi que les différents crypto-systèmes) proposés dans cette thèse, sont totalement numériques et sont constitués à partir des cartes chaotiques de bases suivantes : Logistique, PWLCM (Piece Wise Linear Chaotic Map), Skew tent, Cat, mais sous forme discrétisée, utilisant une précision finie $N = 32$ bits.

Par exemple la carte Cat bidimensionnelle donnée par [Fridrich, 1998] est utilisée comme couche de permutation ou couche de diffusion multidimensionnelle.

***Remarque*** : sous Matlab, le calcul est fait en double précision, mais le résultat final est donné en entier par les fonctions floor et ceil.

### 3.5.1 Tests réalisés pour la quantification des performances.

Pour les différentes cartes chaotiques discrètes utilisées, ainsi que les générateurs de nombres pseudo-chaotiques proposés, nous avons réalisé des tests signal et statistiques de NIST.

Pour le test signal, nous présentons les résultats suivants : diagramme de bifurcation, exposant de Lyapunov, variation temporelle, espace de phase 2-D, attracteur, auto et inter-corrélation, histogramme, sensibilité à la clé secrète (composée des conditions initiales et des paramètres du générateur).

L'espace de phase 2-D des cartes de base, permet de discerner le type de non linéarité (par morceaux ou polynomiale).

### 3.5.2 Etude des performances des 4 cartes chaotiques de base

Dans cette section, nous considérons les équations en réel et en discret, ainsi que les performances (en discret) des trois cartes chaotiques non-linéaires monodimensionnelles utilisées: Logistique, PWLCM, Skew tent, et la carte Cat chaotique linéaire bidimensionnelle. Ces cartes forment les éléments de base des générateurs chaotiques proposés.

***Remarque*** : seulement, dans le cas de la carte Logistique, le diagramme de bifurcation et l'exposant de Lyapunov sont donnés en réel. En effet, dans le cas discret, le paramètre de contrôle est fixé à sa valeur optimale.

L'équation sous sa forme générale en réels des cartes chaotiques monodimensionnelles s'écrit :

$$x(n) = f(x(n-1), p), \qquad n = 1,2,\cdots, \qquad x(0) \in S, \qquad p \in S_p \qquad (2.2)$$

$f : S \to S$, $S \subset \mathbb{R}$ avec $S = [0,1]$ ou $S = [-1,1]$, et $S_p$, est l'intervalle de variation du paramètre de contrôle.

Partant d'une valeur initiale $x(0) \in S$, nous obtenons, en réitérant la carte $n$ fois, la séquence suivante :

$$x(1) = f(x(0), p), \qquad x(2) = f(f(x(0), p), p), \quad \cdots, x(n) = f(\cdots f(f(x(0), p), p) \cdots, p)$$

Pour n'importe quelle valeur initiale $x(0) \in S$, la séquence de valeurs $x(0), x(1), ...., x(n)$ est appelée l'orbite (ou la trajectoire ) de la carte, produite à partir de l'état initial $x(0)$, et pour la valeur $p$ donnée du paramètre du contrôle.

#### *2.5.2.1 Carte Logistique*

A l'origine, la carte logistique est un modèle de croissance démographique publié par Pierre Verhulst en 1845 [Verhulst, 1845]. A cause de la simplicité de son équation de récurrence, en 1947 Ulam et Von Neumann l'ont utilisé en tant que générateur de nombre pseudo-aléatoire [Ulam et von Neumann, 1947]. Depuis, c'est l'une des cartes les plus utilisées dans les applications cryptographiques, l'histogramme des séquences générées n'est pas uniforme. Son équation de récurrence est donnée par :

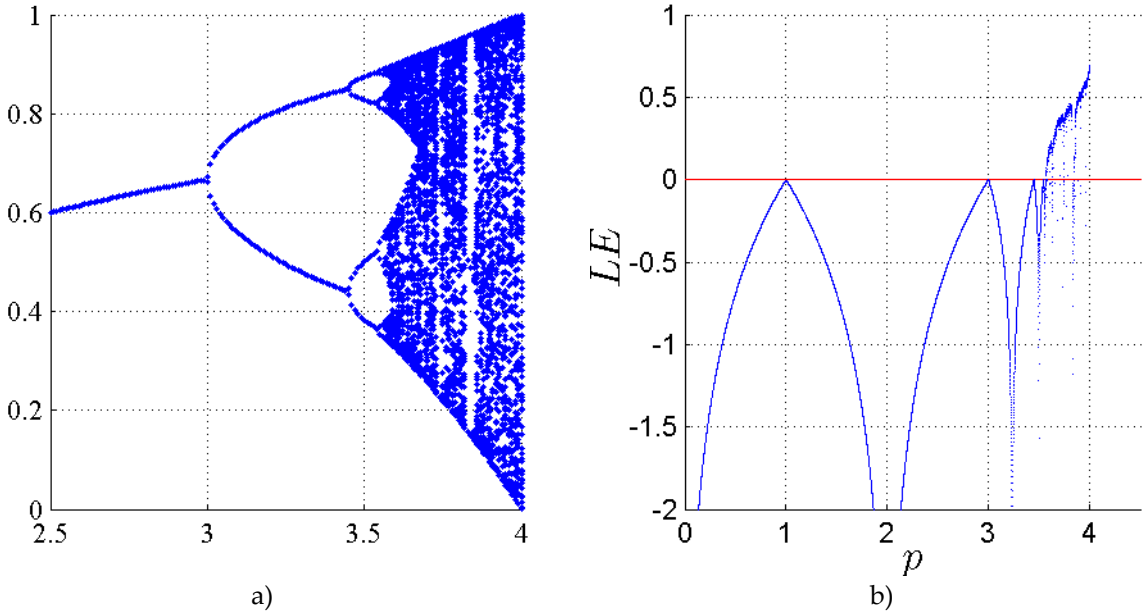
$$x(n) = f(x(n-1),\ p) = p \times x(n-1) \times \big(1 - x(n-1)\big) \tag{2.3}$$

Avec :

$$f: S \to S,\ S = [0,1]\ \ x(n) \in S, \text{et } p\ \in [1,4].$$

Dans la figure 2.1 a) et b), nous présentons respectivement les courbes connues du diagramme de bifurcation et de l'exposant de Lyapunov de la carte en réel.

Comme attendu, la région du chaos est obtenue pour $p \geq 3.57$. Cependant, la valeur $p = 4$, est optimale, car dans ce cas, l'amplitude $x(n), n = 1,2, \cdots$ couvre toute la dynamique $\in [0,1]$ de la carte. Pour cette raison, dans la version discrète de la carte, nous fixons la valeur du paramètre de contrôle à la valeur correspondant à $p = 4$.




a) b)

Fig. 2. 1 : Evolution du diagramme de bifurcation en a) et du l'exposant de Lyapunov en b) de la carte logistique en fonction de $p$

*2.5.2.2 Carte Logistique discrétisée*

L'équation de la carte Logistique discrétisée, pour le paramètre de contrôle $p$ fixé à 4, est donnée par la relation suivante [Peng et al ,2007] :

$$X(n) = F\big(X(n-1)\big) = \begin{cases} \left\lfloor \dfrac{X(n-1) \times \big(2^N - X(n-1)\big)}{2^{N-2}} \right\rfloor & si\ \ X(n-1) \neq \{0{,}3 \times 2^{N-2}, 2^N\} \\ 2^N - 1 & ailleurs \end{cases} \quad (2.4)$$

$\lfloor Z \rfloor$ (fonction Floor), dénote le plus grand entier inférieur à la valeur $Z$.

$X(n)$ prend une valeur entière $\in [0, 2^N - 1]$, et $N = 32$ bits est la précision utilisée.

Notons que le processus de discrétisation dégrade les propriétés chaotiques de la carte chaotique originale (en représentation réelle). En effet, l'opération d'approximation floor, rend différent $X(n+1)$ de $2^N \times x(n-1)$. Ainsi, les séquences produites diffèrent l'une de l'autre. La technique de perturbation décrite plus loin permet d'atténuer l'effet de la dégradation et aussi de maîtriser, en partie, la longueur de la trajectoire.

Dans les tests qui suivent, nous n'utilisons que 2000 échantillons sur les 2200 échantillons générées (afin de s'éloigner du régime transitoire). Dans la figure 2.2, nous montrons, pour $X(0) = 3890346746$, un exemple de résultats obtenus : a) variation discrète de $X(n)$ en fonction de $n$, b) espace de phase, c) attracteur, d) histogramme.

***Remarque*** : Tout au long de ce chapitre, dans certaines figures, pour plus de clarté nous ne traçons pas tous les échantillons, par exemple dans la figure 2.2 : a), b, et c) nous n'avons tracé que les 100 derniers échantillons. De même, pour les courbes de bifurcation qui utilisent un pas égal à $(2^N - 1)/100$. Par ailleurs, dans tous les histogrammes de ce chapitre, l'intervalle des valeurs possible de $X(n)$ est divisé en 20 intervalles (ou classes de valeurs).

Dans la figure 2.3-a, nous montrons, un exemple de résultats d'auto et d'inter-corrélation entre la séquence générée par la condition initiale $X(0)$ et la séquence générée par la condition initiale $X(0+\epsilon)$, qui diffère de $X(0)$ seulement dans le bit de poids faible. Ce résultat montre que les propriétés de auto et l'inter-corrélation des séquences chaotiques sont bonnes comparées à celles des générateurs pseudo-aléatoires de types : registre à décalage à réaction à longueur maximale, Gold et Kasami. Ce résultat est vrai pour toutes les cartes et générateurs chaotiques utilisés dans cette thèse.

Dans la figure 2.3-b, nous montrons la caractéristique de la sensibilité à la clé secrète (ici la condition initiale) mesurée par la distance de Hamming (en pourcentage de bits changés) entre la séquence générée par $X(0)$ et la séquence générée par $X(0+\epsilon_i), i = 1, 2, \cdots, 32$. $\epsilon_i$ indique la position du bit changé. Nous remarquons que, quelque soit la position du bit changé dans la condition initiale, ceci engendre pratiquement 50 % de

différence entre les bits de deux séquences concernées, et donc la propriété de l'avalanche est atteinte (dans le paragraphe 3.12.2 du chapitre 3, nous détaillons la procédure du test sur un exemple similaire).

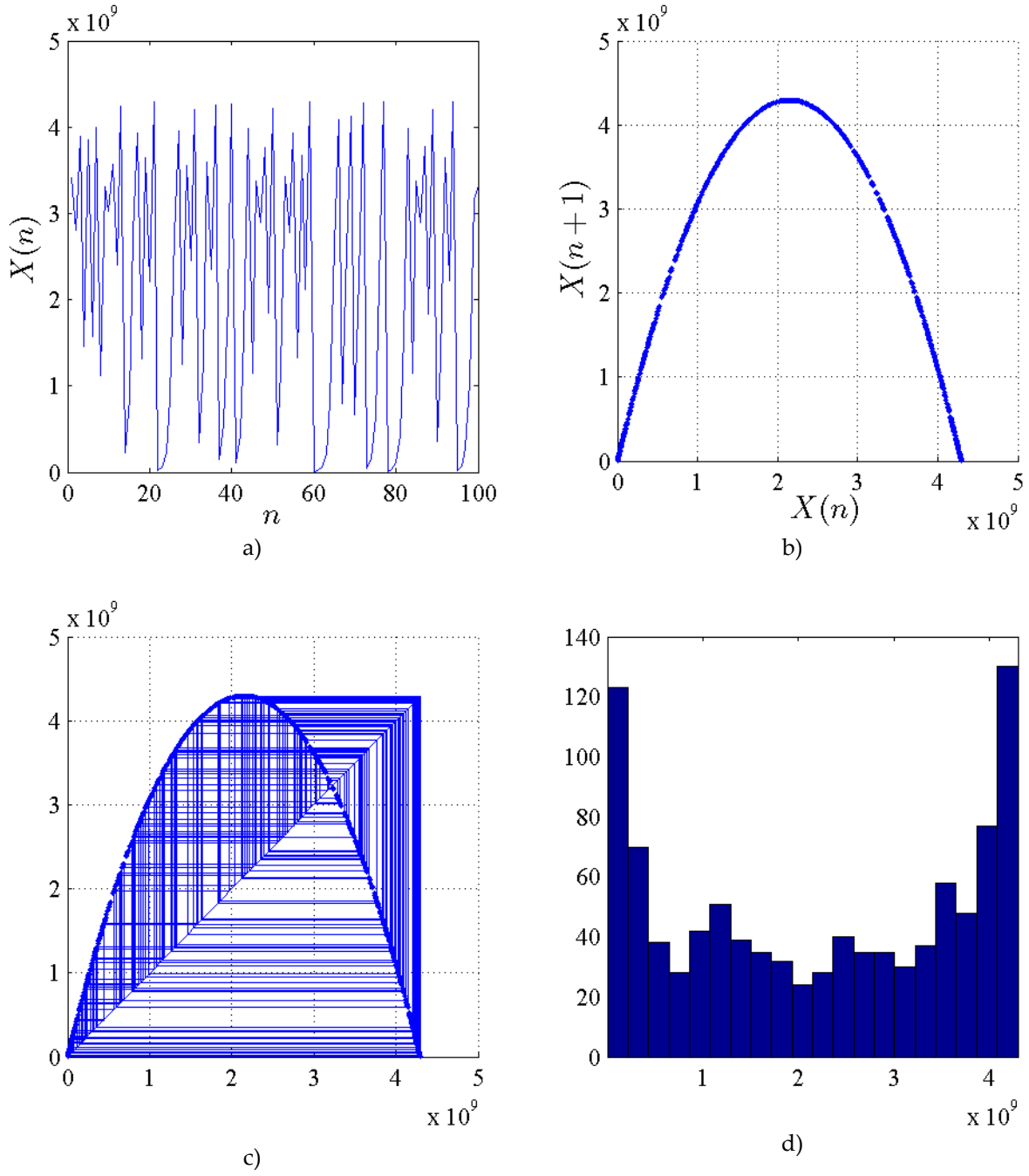



Fig. 2. 2 : Exemple de résultats de la carte Logistique discrète : a) variation discrète de *X(n)* en fonction de *n* b) espace de phase, c) attracteur, d) histogramme.

,

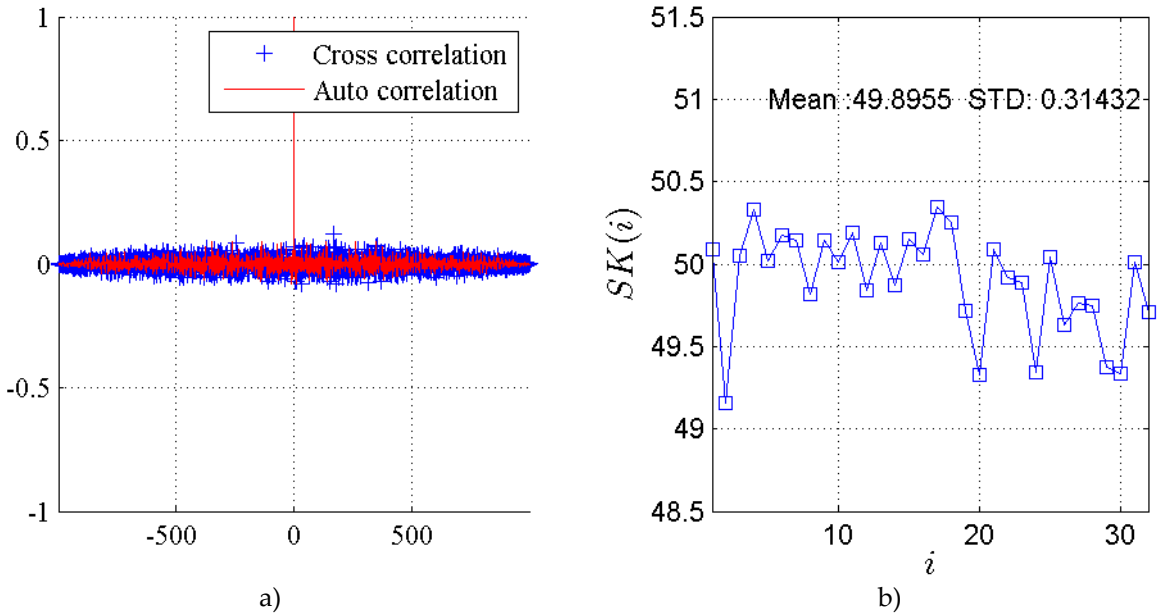



Fig. 2. 3: Exemple de résultats de la carte Logistique discrète : a) auto et inter-corrélation , b) sensibilité à la clé secrète.

*2.5.2.3 Carte Skew Tent*

La carte Skew tent est une carte linéaire par morceaux, décrite en réel par l'équation suivante :

$$x(n) = f(x(n-1), p) = \begin{cases} \dfrac{x(n-1)}{p} & si\ \ 0 \le x(n-1) \le p \\ \dfrac{1-x(n-1)}{1-p} & si\ \ p < x(n-1) \le 1 \end{cases} \qquad (2.5)$$

avec $f: S \to S, S = [0,1], x(n) \in S$

et $p$ est le paramètre de contrôle qui varie dans l'intervalle suivant : $0 < p < 1$

L'histogramme de cette carte est pratiquement uniforme comparé à celle de la carte Logistique [Billings et Bollt, 2001].

*2.5.2.4 Carte Skew Tent discrétisée*

Le carte Skew tent discrétisée est définie par la relation suivante [Masuda et al, 2006] :

$$X(n) = F(X(n-1), P) = \begin{cases} \left\lceil \dfrac{2^N \times X(n-1)}{P} \right\rceil & si\ \ 0 \le X_n \le P \\ \left\lfloor 2^N \times \dfrac{2^N - X(n-1)}{2^N - P} \right\rfloor + 1 & si\ \ P < X_n \le 2^N \end{cases} \qquad (2.6)$$

Où $\lceil Z \rceil$ (fonction ceil) dénote l'entier supérieur à $Z$, $X(n)$ prend une valeur entière appartenant à $[0, 2^N - 1]$, et $P$ le paramètre de contrôle discret est tel que : $0 < P < 2^N$-1.

L'exposant de Lyapunov en discret est donné par :

$$\lambda(P) = -\frac{P}{2^N}\ln\left(\frac{P}{2^N}\right) - \left(\frac{2^N - P}{2^N}\right)\ln\left(\frac{2^N - P}{2^N}\right) \tag{2.7}$$

Dans la figure 2.4 a) et b), nous présentons respectivement les courbes du diagramme de bifurcation et de l'exposant de Lyapunov de la carte Skew tent discrète, pour $X(0)$ =3890346746

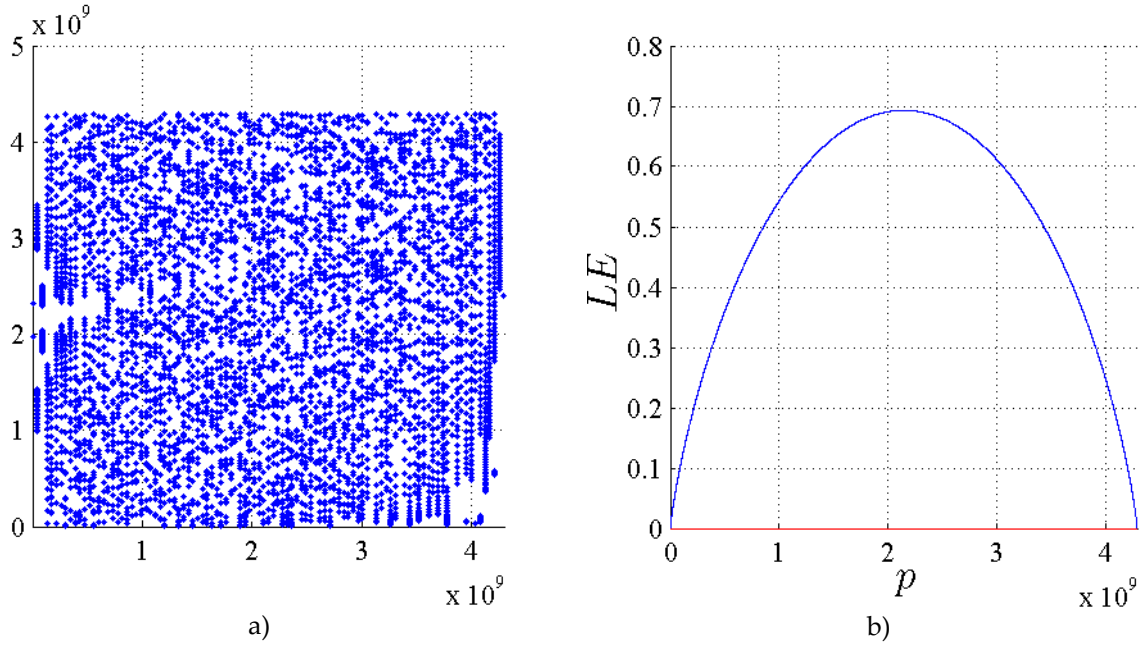



Fig. 2. 4: Evolution du diagramme de bifurcation en a) et l'exposant de Lyapunov en b) de la carte Skew tent discrète en fonction de $P$.

Le diagramme de bifurcation, nous montre que le meilleur intervalle pour le paramètre de contrôle est : $0.6 \times 10^9 \leq P \leq 3.7 \times 10^9$. Donc, il est préférable d'utiliser les valeurs de cet intervalle pour former la valeur de la clé secrète.

En se plaçant dans les mêmes conditions d'expérimentation que la carte Logistique, et pour $P = 2 \times 10^9$, nous montrons dans la figure 2.5, un exemple de résultats obtenus : a) variation discrète de $X(n)$ en fonction de $n$, b) espace de phase, c) attracteur, d) histogramme.

On remarque que l'histogramme obtenu est, en tout cas, plus uniforme que celui de la carte Logistique.

Dans la figure 2.6 a), et b) nous montrons respectivement un exemple de résultats d'auto et d'inter-corrélation et de la sensibilité à la clé secrète, formée ici par le

couple condition initiale et paramètre de contrôle, donc de taille 64 bits. A ce propos, les mêmes remarques faites dans le cas de la carte Logistique s'appliquent ici.

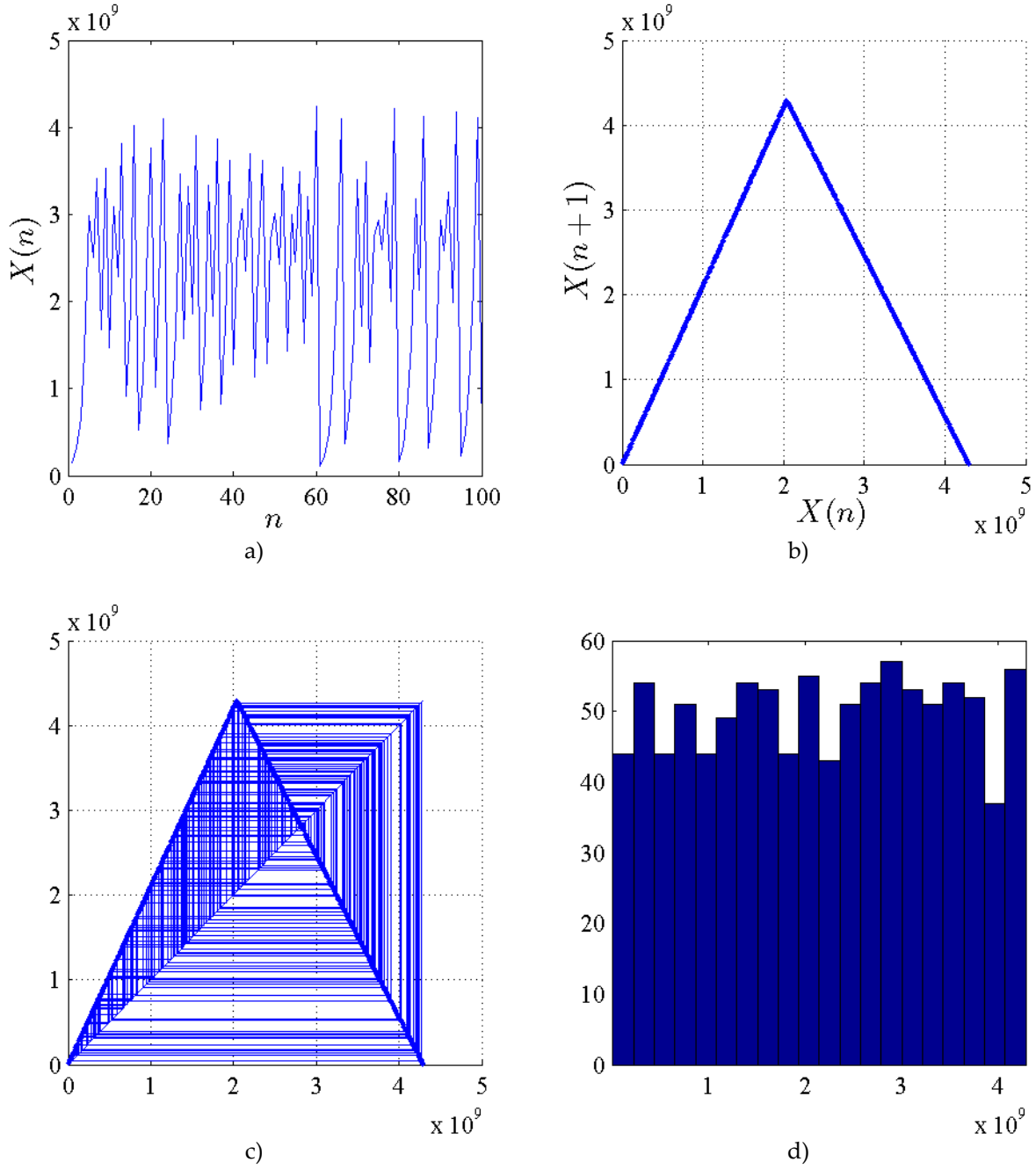
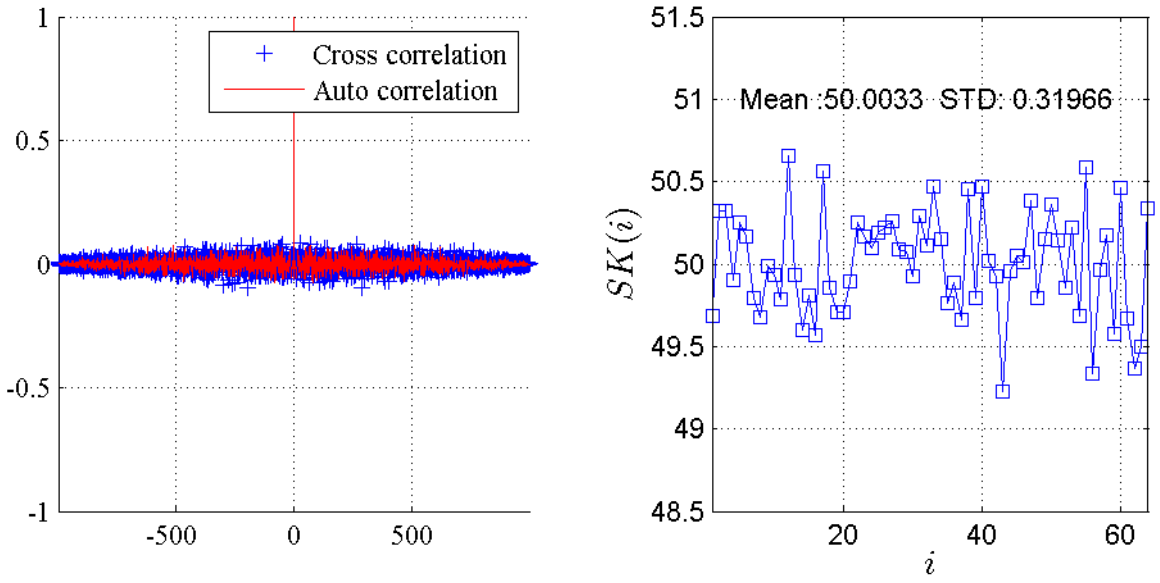



Fig. 2. 5: Exemple de résultats de la carte Skew tent discrète : a) variation discrète de *X(n)* en fonction de *n* b) espace de phase, c) attracteur, d) histogramme.

,




a) b)

Fig. 2. 6: Exemple de résultats de la carte Skew tent discrète : a) auto et inter-corrélation , b) sensibilité à la clé secrète.

***Remarque*** : Dans le chapitre 3, la carte Skew tent (après adaptation de son équation) est utilisée en tant que couche de substitution et de permutation dans les crypto-systèmes proposés.

*2.5.2.5 Carte PWLCM*

La carte PWLCM (Piecewise Linear Chaotic Maps) est une autre carte chaotique linéaire par morceaux, décrite par l'équation suivante :

$$x(n) = f(x(n-1), p) = \begin{cases} \dfrac{x(n-1)}{p} & si\ \ 0 \le x(n-1) \le p \\ \dfrac{[x(n-1)-p]}{0.5-p} & si\ \ p \le x(n-1) \le 0.5 \\ f(1-x(n-1), p) & ailleurs \end{cases} \quad (2.8)$$

$x(n) \in [0, 1]$, et $p$ le paramètre de contrôle est tel que : $0 < p < 0.5$

Grâce à ses bonnes caractéristiques cryptographiques, comparées à celles de la carte Logistique, la carte PWLCM est très utilisée dans le chiffrement des données.

La version discrète de l'équation ci-dessus est donnée par [Lian et al, 2007] :

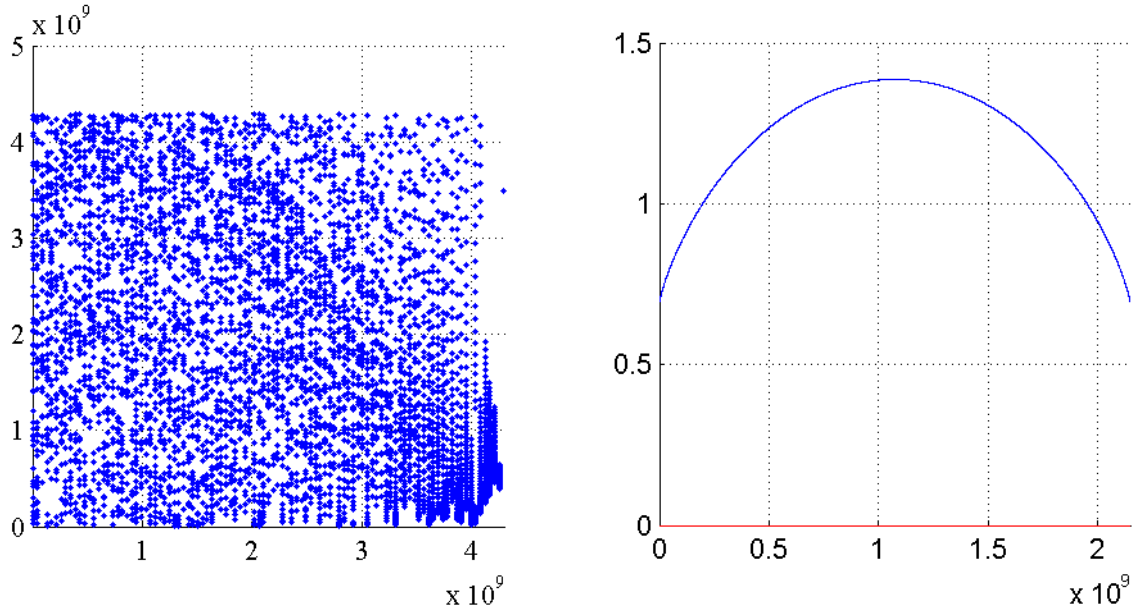
$$X(n) = F(X(n-1), P)\begin{cases} \left\lfloor 2^N \times \dfrac{X(n-1)}{P} \right\rfloor & 0 \le X(n-1) < P \\ \left\lfloor 2^N \times \dfrac{(X(n-1)-P)}{(2^{N-1}-P)} \right\rfloor & P \le X(n-1) < 2^{N-1} \\ f(2^N - X(n-1), P) & X(n-1) \ge 2^{N-1} \end{cases} \quad (2.9)$$

$X(n) \in [0, 2^N - 1]$, et $P$ le paramètre de contrôle, avec : $0 < P < 2^{N-1}$

L'exposant de Lyapunov de la carte PWLCM, se calcule de la même manière que dans le cas de la carte Skew tent, la seule différence est dans le nombre de branches linéaires. Il est donné par :

$$\lambda(P) = -\frac{P}{2^{N-1}} \ln \frac{P}{2^N} - (2^N - \frac{P}{2^{N-1}}) \ln \left(2^{N-1} - \frac{P}{2^N}\right)$$

Dans la figure 2.7 a) et b), nous présentons le diagramme de bifurcation et l'exposant de Lyapunov de la carte PWLCM discrète, pour $X(0) = 3890346746$




a) b)

Fig. 2. 7: Evolution du diagramme de bifurcation en a) et l'exposant de Lyapunov en b) de la carte PWLCM discrète en fonction de $P$.

Nous remarquons que les valeurs de l'exposant de Lyapunov sont meilleures que dans le cas de la carte Skew tent.

Pour $P = 1.2 \times 10^9$, nous montrons dans la figure 2.8, un exemple de résultats obtenus : a) variation discrète de $X(n)$ en fonction de $n$, b) espace de phase, c) attracteur, d) histogramme.

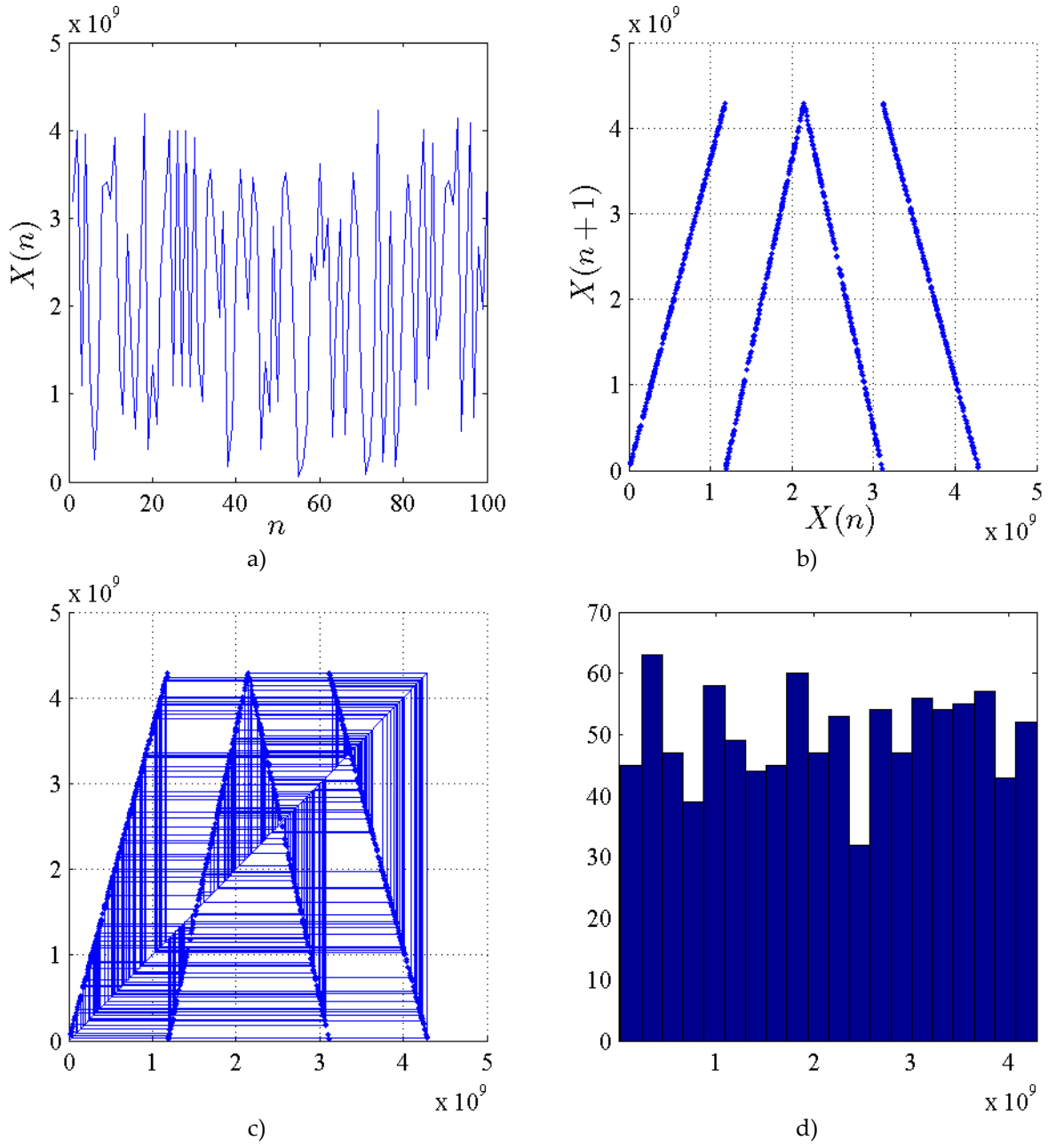
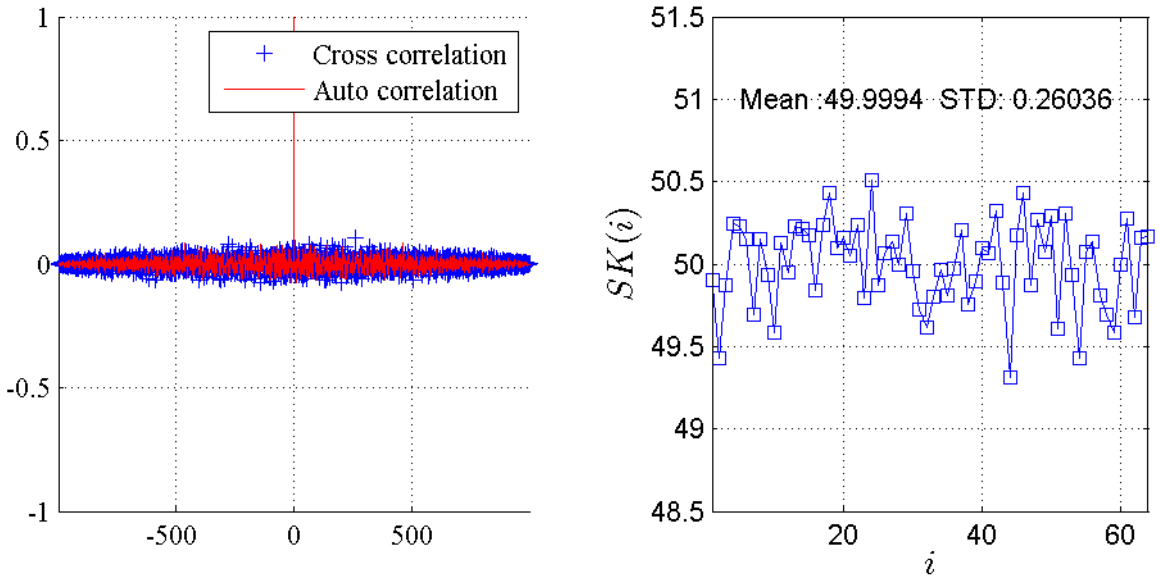



Fig. 2. 8: Exemple de résultats de la carte PWLCM discrète : a) variation discrète de *X(n)* en fonction de *n* b) espace de phase, c) attracteur, d) histogramme.

Dans la figure 2.9 a), et b) nous montrons respectivement un exemple de résultats d'auto et d'inter-corrélation et de la sensibilité à la clé secrète, de taille 64 bits. Ces résultats amènent exactement aux mêmes commentaires que le cas de la carte Skew tent.




a) b)

Fig. 2. 9: Exemple de résultats de la carte PWLCM discrète : a) auto et inter-corrélation , b) sensibilité à la clé secrète.

### 3.5.3 Carte Cat bidimensionnelle

La carte Cat en réel est décrite par l'équation suivante [Arnold et Avez, 1967] :

$$\begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} = \begin{bmatrix} 1 & u \\ v & 1+uv \end{bmatrix} \times \begin{bmatrix} x_1(n-1) \\ x_2(n-1) \end{bmatrix} mod 1 \qquad (2.10)$$

L'équation de la carte Cat montre une relation linéaire entre les échantillons (état) à l'instant $n$, $[x_1(n), x_2(n)]$ et les échantillons (état) à l'instant $(n-1)$, $[x_1(n-1), x_2(n-1)]$, avec $x_i(n) \in [0,1], i = 1,2$. Les paramètres de contrôle $u = 1, v$ =1, sont réels.

La carte Cat est bijective, puisque le déterminant de sa matrice de transformation est égal à $\pm 1$. Sous forme discrète [Fridrich, 1998], l'équation de la carte Cat discrète est donnée par :

$$\begin{bmatrix} X_1(n) \\ X_2(n) \end{bmatrix} = A \times \begin{bmatrix} X_1(n-1) \\ X_2(n-1) \end{bmatrix} mod\ 2^N \qquad (2.11)$$

La matrice $A$ peut prendre les 4 formes suivantes :

$$A_0 = \begin{bmatrix} 1 & u \\ v & 1+u\times v \end{bmatrix},\ A_1 = \begin{bmatrix} 1+u\times v & u \\ v & 1 \end{bmatrix}, A_2 = \begin{bmatrix} u & -1+u\times v \\ 1 & v \end{bmatrix}, A_3 = \begin{bmatrix} u & 1 \\ -1+u\times v & v \end{bmatrix}.$$

Et : $X_i,\ u, v \in [0, 2^N - 1],\ i = 1,2,\ N = 32$ bits.

Nous avons tracé dans la figure 2.10, pour $X_1(0)$= 3437178441, $X_2(0)$= 609397184, le diagramme de bifurcation de la carte discrète en fonction du paramètre $u$, pour un paramètre $v$ fixé. Nous remarquons d'après cette figure, que le chaos existe pour les valeurs du paramètre de contrôle $u < 2^{(N-1)}$. En outre, pour les valeurs du paramètre de

contrôle $u > 2^{(N-1)}$, nous trouvons une orbite point fixe (0) quelque soit la valeur de $v$. Nous obtenons les mêmes résultats, en inter-changeant $u$ et $v$.

En fait, pour avoir du chaos de façon certaine, il faut que le produit $u \times v \times X_i(n) < 2^{64}$, résultat observé aussi expérimentalement.

Pour ces raisons, les valeurs des paramètres de contrôle $(u, v)$ doivent être prises dans l'intervalle suivant : $0 < [u, v] < 2^{N/2}$.

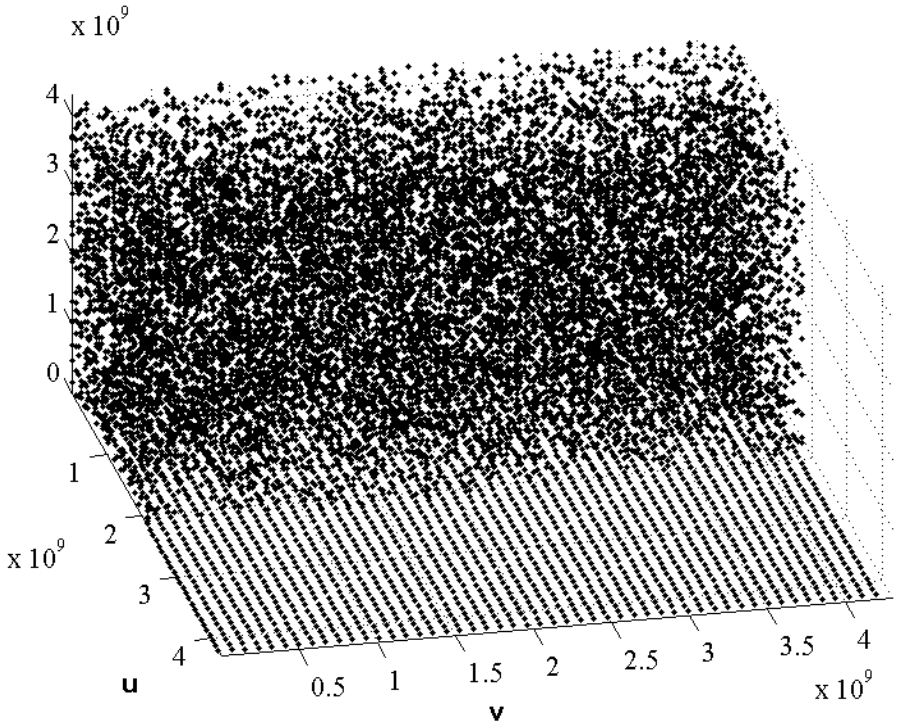



Fig. 2. 10: Diagramme de bifurcation de la carte Cat en fonction du paramètre de contrôle $u$, et $v$

Dans la figure 2.11 : a), b) c) et d), pour $X_1(0)$= 3437178441, $X_2(0)$= 609397184, $u$ = 27640, $v$= 60013, nous avons tracé respectivement, l'espace de phase pour $X_1$, l'espace de phase pour $X_2$, l'espace de phase entre $X_1$et $X_2$, l'auto-corrélation de $X_1$ et l'inter-corrélation entre $X_1$et $X_2$.

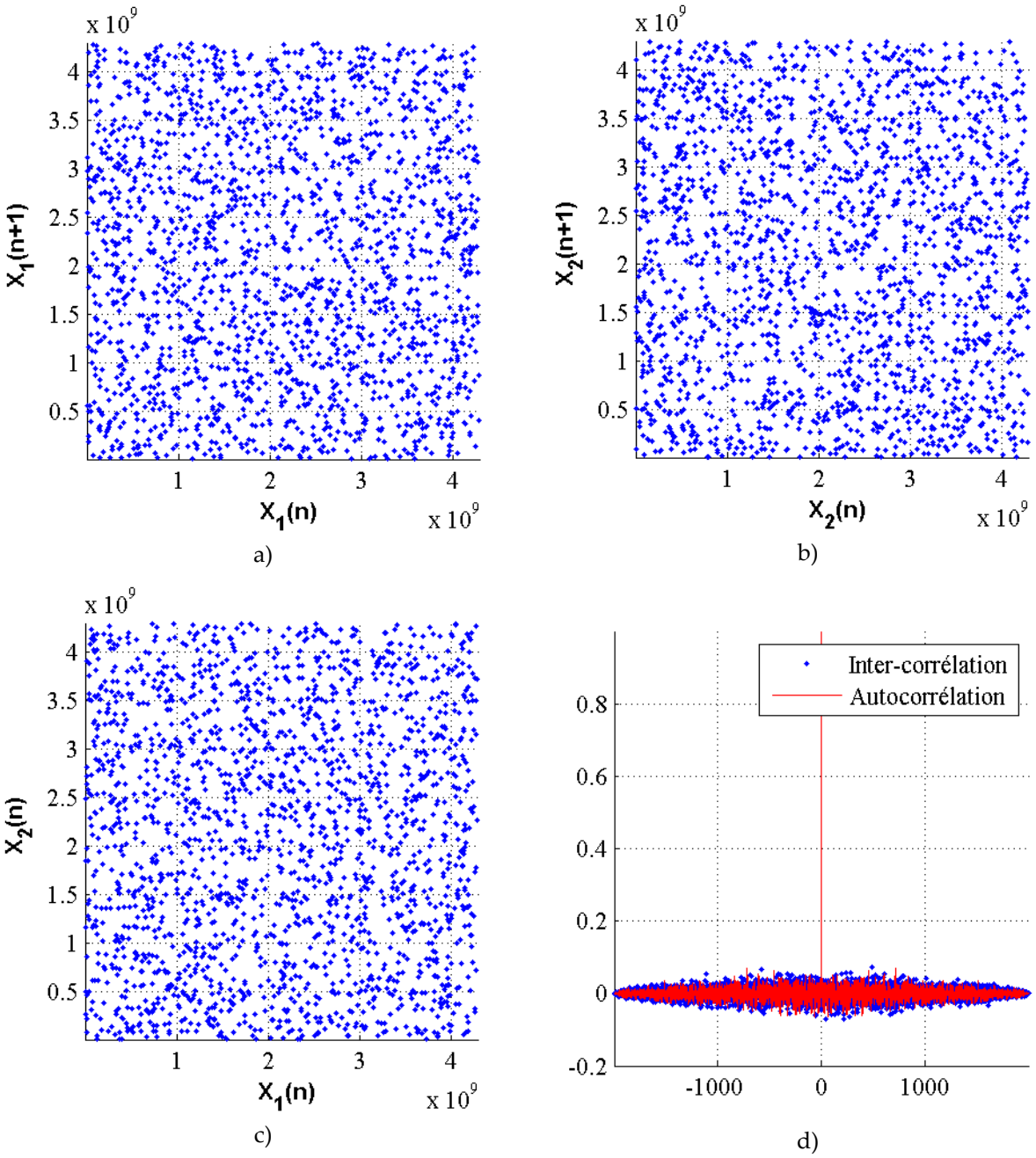



Fig. 2. 11: a) l'espace de phase pour $X_1$ ; b) espace de phase pour $X_2$; c) espace de phase entre $X_1$ et $X_2$ ; d) auto-corrélation de $X_1$ et inter-corrélation entre $X_1$ et $X_2$

Dans la figure 2.12 a) et b), nous avons tracé les histogrammes de $X_1$ et de $X_2$ respectivement.

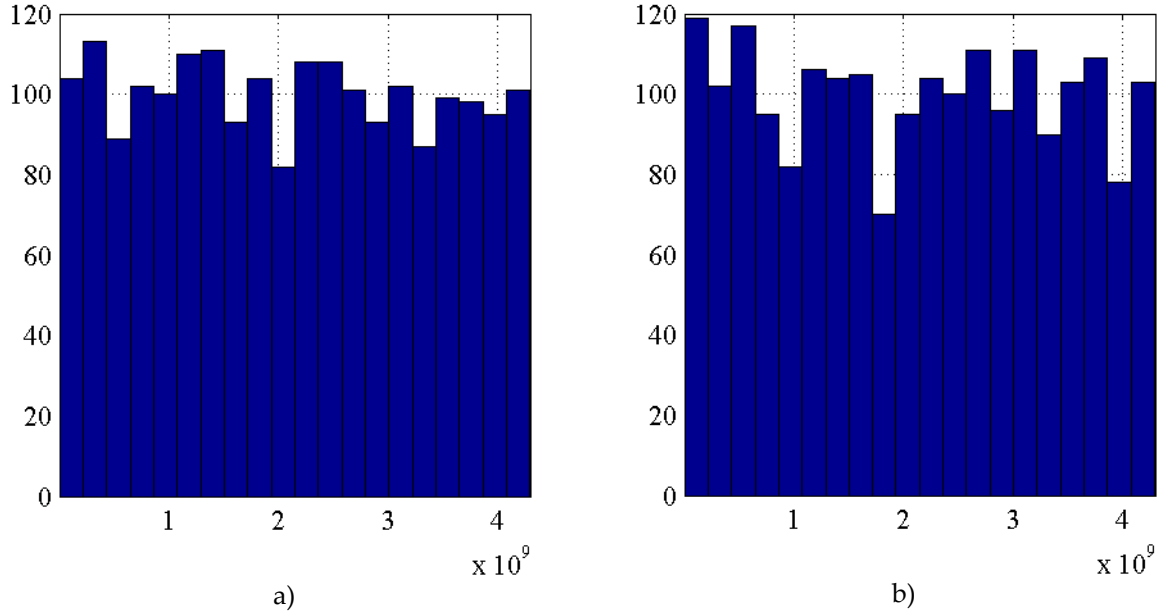



Fig. 2. 12: a) Histogramme de $X_1$ b) Histogramme de $X_2$

La carte Cat de base est utilisée dans ce chapitre, comme élément fondamental du deuxième générateur chaotique proposé et aussi utilisée en chapitre 3, en tant que couche de diffusion dynamique.

**Test de NIST (voir annexe 2.1):**

Les tests de NIST appliqués sur les 4 cartes chaotiques précédentes : Logistique, PWLCM, Skew tent, et Cat, montrent les faiblesses de ces cartes (elles ne passent pas tous les tests au nombre 188), surtout pour la carte Logistique et la carte Cat. Cependant, lorsque nous appliquons la technique de perturbation (voir plus loin) sur ces cartes, les résultats obtenus (donnés plus loin) montrent que le taux de réussite des différents tests est pratiquement de 100%. D'où l'un des intérêts majeurs de la technique de perturbation.

### 3.5.4 Sous-échantillonnage des cartes chaotiques

Les propriétés cryptographiques des cartes ou générateurs chaotiques peuvent être considérablement améliorées en appliquant une procédure de sous-échantillonnage [Zhang et al, 2000]. Dans ce cas l'équation monodimensionnelle peut s'écrire :

$$X(n) = F^s(X(n-1), P) \tag{2.12}$$

Cela veut dire qu'on itère $s$ fois la carte chaotique $F(X(n-1), P)$ pour obtenir en sortie l'échantillon $X(n)$, avec $s$ un entier positif et supérieur à 3, afin de rendre extrêmement difficile, toute opération visant à connaître la clé ou une partie de la clé secrète. Le prix à payer de cette technique est la perte de certains échantillons et le temps de calcul pour générer la séquence chaotique souhaitée.

Nous présentons dans la suite de ce chapitre notre contribution majeure, concernant la proposition de trois générateurs de structures différentes, utilisant des cartes chaotiques de base connues, et incluant chacun une technique de perturbation.

D'abord, nous présentons l'effet de la précision finie en termes de dégradation des propriétés des signaux chaotiques, et la technique de perturbation utilisée pour contourner cet effet [Tao et al, 1998 ], [El Assad et Noura, 2010-1], [El Assad et Noura, 2010-2].

## 2.6 Effet de la précision finie, mesure de l'orbite et technique de perturbation

Les dynamiques chaotiques numériques (précision finie), diffèrent des dynamiques chaotiques continues et des problèmes sérieux apparaissent comme : périodicité et longueur courte des trajectoires, distribution non uniforme, etc. Nous montrons que l'association d'une méthode de perturbation de la trajectoire (ou orbite chaotique) avec une technique de couplage permet de résoudre ce problème.

En précision finie, pour un système de $N$ bits de quantification, le nombre maximum (théorique rarement atteint) de niveaux chaotiques différents est $2^N$. Cette limitation de l'espace des valeurs (supposée infini pour le chaos analogique) entraîne des cycles périodiques des différentes orbites chaotiques. La longueur maximale des orbites est forcement inférieure à $2^N$ (propriété quasi chaotique), donc la dynamique des signaux chaotiques est dégradée [Jessa, 2006] [Li et al, 2001-2], [Li et al, 2003].

Par ailleurs, à chaque condition initiale, on a une orbite chaotique formée généralement de deux parties : une branche transitoire de longueur $l$ et un cycle de période $c$ [Grebogi et al,1998]. Notons aussi que le cas $l=0$ ou $c=1$ est possible. Lorsque $l=0$, l'orbite est un simple cycle de longueur $c$, et lorsque $c=1$, l'orbite chaotique converge vers un point fixe (voir figures 2.13 et 2.14).

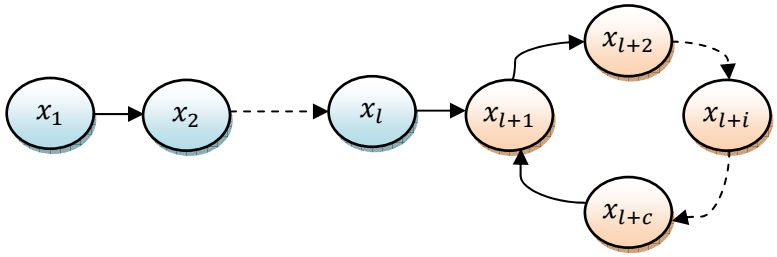



Fig. 2. 13 : Orbite chaotique de longueur $o = l + c$

La figure 2.14, montre un exemple explicatif, dans le cas de $N=4$, de deux orbites chaotiques obtenues pour deux conditions initiales différentes.

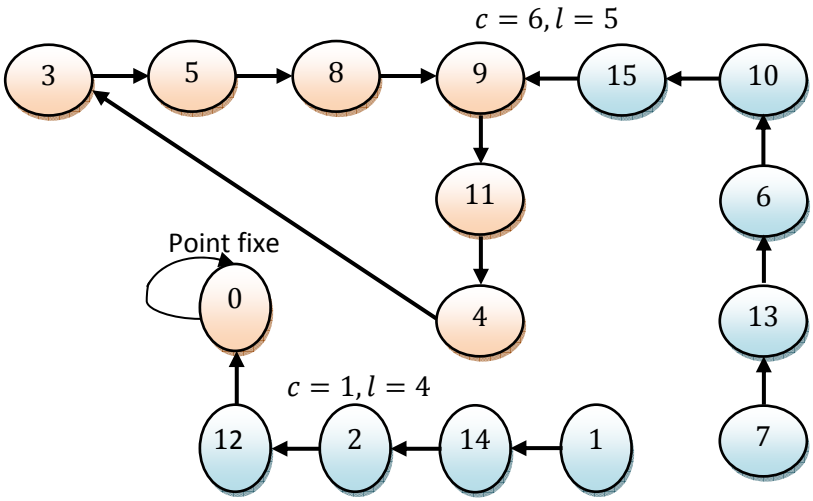



Fig. 2. 14: Deux orbites chaotiques différentes pour deux conditions initiales différentes, *N=4*

*2.6.1.1 Mesure des orbites (ou trajectoires) chaotiques : cycles et transitoires*

Dans ce paragraphe, nous exposons la méthode développée permettant la mesure de l'orbite chaotique $o = l + c$, où $c$ est la longueur du cycle et $l$ est la longueur du régime transitoire. Plusieurs résultats ont été donnés dans la littérature sur cette question et sur la simulation des nombres aléatoires, mais sans indiquer la méthode de mesure utilisée [Lanford,1998], [Grebogi et al, 1998], [Ecuyer,1990], [Lozi et Fiol, 2009], [Wang et al, 2004].

Ces études indiquent entre autres que : pour un nombre $N_0$ de conditions initiales différentes, le nombre moyen de cycles différents est:

$$\overline{N_c} = \sum_{k=1}^{N_0} \frac{1}{2k-1} \# \frac{1}{2} \ln(N_0) + 0.982, pour\ N_0 \geq 2 \qquad (2.13)$$

En effet, certaines conditions initiales ne génèrent pas de nouveaux cycles différents. Ceci est illustré schématiquement sur la figure 2.15, sur un exemple simple d'une carte chaotique de base $X(n) = F(X(n-1), P)$, dans le cas où $N_0$ =2. Deux cas de figures a) et b) sont à considérer. Pour les deux cas, évidemment, la première condition initiale génère le premier cycle (sauf le cas trivial d'un point fixe). Ensuite, dans le cas a), la deuxième condition initiale ne génère pas un nouveau cycle. Tandis dans le cas b), la deuxième condition initiale génère un nouveau cycle.

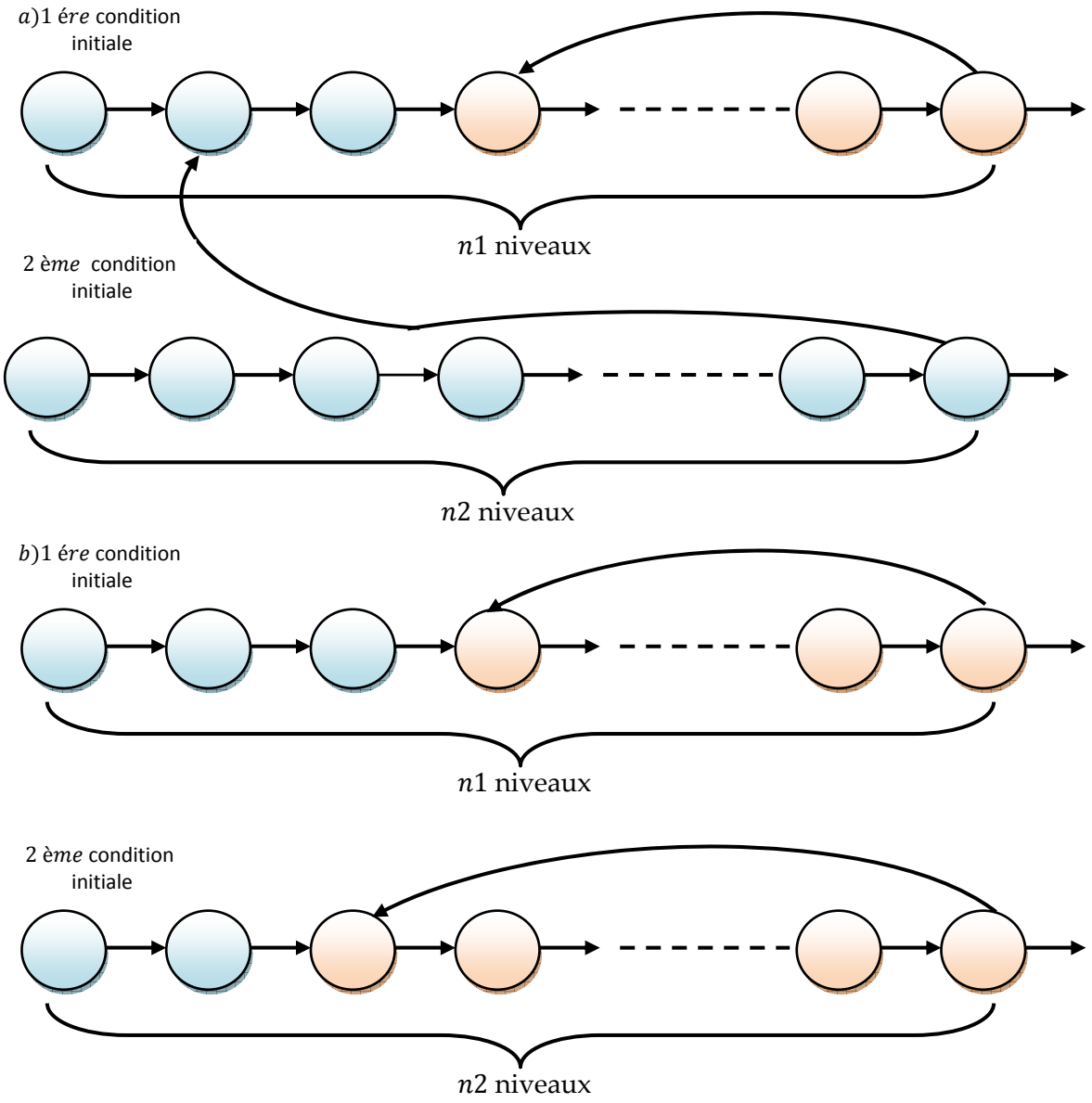



Fig. 2. 15: Illustration schématique des orbites générées dans le cas de deux conditions initiales différentes.

La procédure pour mesurer les orbites, dans le cas des cartes chaotique de base de la forme $X(n) = F(X(n-1), P)$ se déroule comme suit [El Assad et Noura, 2010-2] :

Pour chaque condition initiale et valeur(s) du ou des paramètre(s), on génère une séquence de valeurs chaotiques de longueur $N_t$, puis on procède à la mesure du cycle et du transitoire de la séquence en question (voir Brevets). Si la séquence en question ne contient aucun cycle, alors on poursuit le processus de génération jusqu'à 2x$N_t$ et on recommence la mesure, et ainsi de suite jusqu'à l'obtention dans la séquence sous test de longueur $N_c$=$W$x$N_t$, ($W$ est un entier positif) une orbite qui sera composée d'une partie transitoire et d'un cycle (qui peut être répété). Ensuite, on sauvegarde dans un tableau, les valeurs : $X_i$, $P_i$, $c$, $l$, $o$ et on recommence le processus pour une nouvelle condition initiale et valeur(s) du ou des paramètre(s), jusqu'à l'épuisement de toutes les conditions initiales au nombre $N_0$. Enfin, une

analyse statistique (courbes *c, l, o=f(N₀)* min, max, moyen) sur les orbites nous permettent de fixer la valeur de l'orbite nominale $o_{nom} = \Delta_{nom}$ avant la perturbation.

### 3.6.2 Résultats de mesure des cycles, transitoires et orbites des 3 cartes chaotiques de base

Nous donnons dans ce paragraphe les résultats de mesure des cycles, transitoires et orbites des 3 cartes chaotiques de base : Logistique, Skew tent et PWLCM. Nous verrons que les 3 cartes ont des caractéristiques (cycle, transitoire, orbite) différentes.

Pour pouvoir réaliser des mesures d'orbites chaotiques en temps raisonnable, nous avons fixé la précision à $N = 18$ bits et nous avons considéré 10.000 conditions initiales (ou clés secrètes) tirées de façon aléatoire et uniforme.

#### *2.6.2.1 Résultats de la carte Logistique :*

Sur les 10.000 conditions initiales, nous obtenons seulement deux cycles de longueurs 30 et 588, et de fréquences 52 et 9948 respectivement. Sur la 2.16 a) et b) nous présentons respectivement les longueurs du transitoire et de l'orbite (en échelle $log_2$) en fonction des conditions initiales. Nous remarquons que la plupart des orbites ont des longueurs comprises dans l'intervalle suivant : $2^{9.34} \leq o \leq 2^{9.87}$.

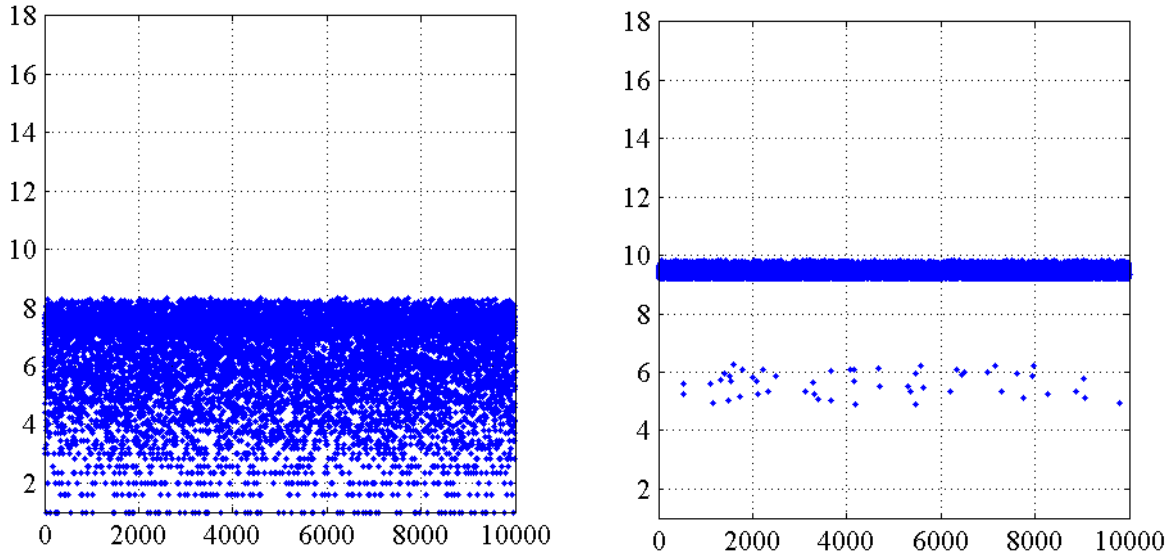
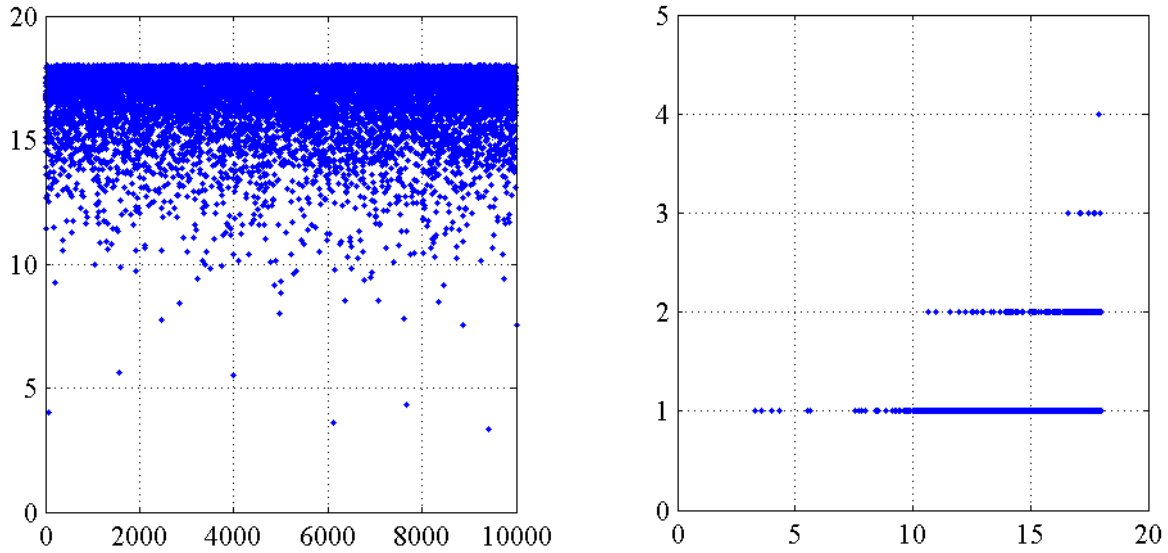



Fig. 2. 16: Variation de la longueur du transitoire en a) et de longueur de l'orbite en b) (échelle $\log_2$) en fonction des conditions initiales.

#### *2.6.2.2 Résultats de la carte Skew tent :*

Nous présentons dans la figure 2.17 a) et b), la longueur des orbites en fonction des conditions initiales et leurs fréquences. Nous remarquons que la plupart des orbites ont des

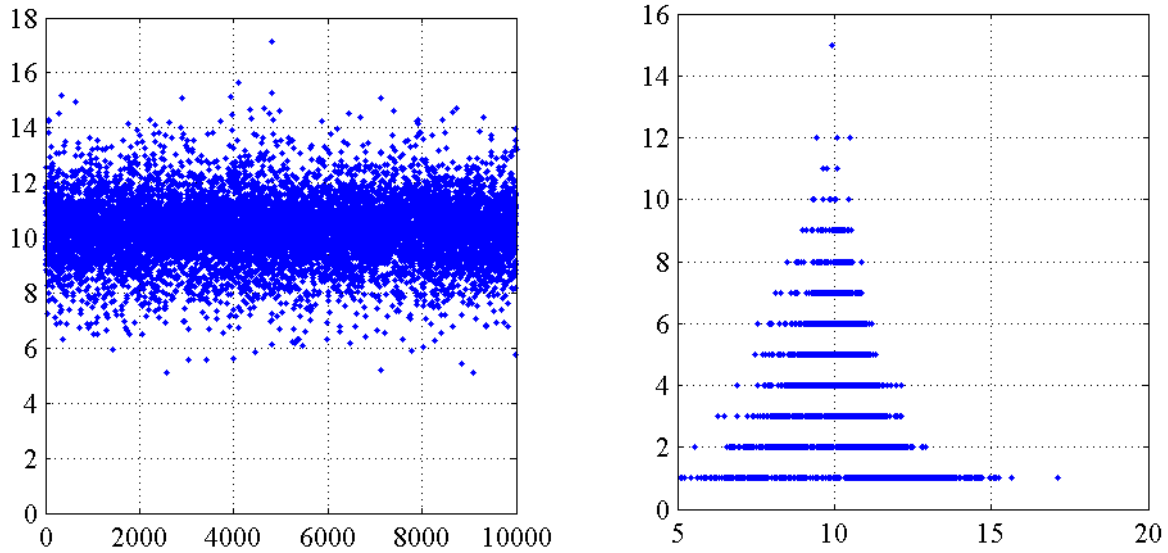
longueurs : $2^{14.63} \leq o \leq 2^{17.7}$ très grandes par rapport à celles obtenues dans le cas de la carte Logistique, et possèdent une fréquence unique. L'échelle abscisse de b) est en $log_2$.




a) b)

Fig. 2. 17: Variation de la longueur des orbites en fonction des conditions initiales en a) et leurs fréquences en b).

*2.6.2.3 Résultats de la carte PWLCM :*




a) b)

Fig. 2. 18: Variation de la longueur des orbites en fonction des conditions initiales en a) et leurs fréquences en b).

De même, nous présentons dans la figure 2.18 a) et b) la longueur des orbites en fonction des conditions initiales et leurs fréquences.

La plupart des orbites ont des longueurs : $2^9 \leq o \leq 2^{11.8}$. L'échelle abscisse de b) est en $log_2$.Dans le tableau 2.1, nous donnons des résultats statistiques comparatifs des 3 cartes :

Tableau 2. 1 : Statistique sur les longueurs des cycles, transitoires et orbites des 3 cartes.

| Carte chaotique | Logistique | Skew tent | PWLCM |
|---|---|---|---|
| Nombre de cycles différents | 2 | 9635 | 2504 |
| Nombre de transitoires différentes | 322 | 0 | 2540 |
| Nombre d'orbites différentes | 354 | 9635 | 3586 |
| Longueur minimale des cycles | 30 | 10 | 1 |
| Longueur minimale des transitoires | 0 | 0 | 0 |
| Longueur minimale des orbites | 30 | 10 | 34 |
| Longueur maximale des cycles | 588 | 262099 | 119820 |
| Longueur maximale des transitoires | 323 | - | 39028 |
| Longueur maximale des orbites | 911 | 262099 | 142267 |
| Longueur moyenne des cycles | 585.0984 | 131391.154100 | 915.619 |
| Longueur moyenne des transitoires | 115.4143 | - | 883.1205 |
| Longueur moyenne des orbites | 700.512 | 131391.154100 | 1798.74 |
| Probabilité d'avoir des orbites de longueur > $2^{10}$ | 0 | 0.9965 | 0.641 |

D'après ces résultats, nous pouvons affirmer que la carte Skew tent est la plus performante en termes de cycles et d'orbites, de longueurs très grandes comparativement aux cartes Logistique et PWLCM. La carte Logistique est la moins performante.

### 3.6.3 Perturbation des orbites chaotiques

Afin de contourner l'effet de la précision finie sur la dynamique chaotique, deux techniques sont utilisées : la technique de couplage et la technique de perturbation de l'orbite chaotique. La première technique permet effectivement une expansion de la longueur des cycles mais sans aucun contrôle. La seconde technique permet non seulement d'avoir un cycle de longueur très grande, mais aussi d'imposer une longueur minimale de cycle, dépendant directement du signal perturbateur. La méthode de perturbation trouve son fondement dans le fait qu'aucun cycle stable n'existe, c.-à-d, si le système chaotique décrit, à un moment donné, un cycle donné, il peut, par application d'une perturbation, quitter ce cycle immédiatement pour aller vers un autre cycle. La figure 2.19, montre le principe de la méthode de perturbation de l'orbite chaotique d'une carte ou d'un générateur chaotique.

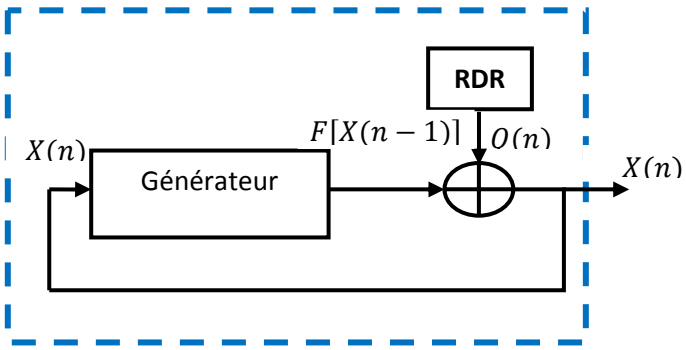



Fig. 2. 19: Principe de la méthode de perturbation de l'orbite chaotique

La structure est composée d'une carte ou d'un générateur de séquences chaotiques, d'une fonction XOR et d'un générateur perturbateur, dont le rôle est de perturber l'orbite chaotique du générateur, lui permettant ainsi d'accéder à une nouvelle orbite.

Le choix de la séquence perturbatrice est effectué selon les règles suivantes : elle devrait avoir une longue longueur de cycle contrôlable et une distribution uniforme; elle ne devrait pas dégrader les bonnes propriétés statistiques de la dynamique chaotique, donc l'amplitude du signal perturbateur doit être nettement plus petite que celle du signal chaotique, de sorte que le rapport entre les deux amplitudes maximales, soit supérieur ou égal à 40 dB :

$$SNR = 10\log\left(\frac{maximum\ amplitude\ du\ signal\ chaotique = 2^N}{maximum\ amplitude\ du\ signal\ perturbé = 2^k}\right) \geq 40\ db \quad (2.14)$$

Un bon candidat pour la génération de séquences perturbatrices est le registre à décalage avec rétroaction (RDR) à longueur maximale. En effet, ce dernier est caractérisé par : une bonne fonction d'auto-corrélation, par une distribution presque uniforme, par un cycle de longueur maximale égale à $2^k - 1$ ($k$ est le degré du polynôme primitif utilisé) et une implémentation logicielle ou matérielle facile.

Partons de l'équation du générateur de base : $X_n = F[X(n-1), P],\ \ X_n \in [0, 2^N - 1], n = 1,2, \ldots$. Chaque valeur $X(n)$ est représentée par $N$ bits :

$$X(n) = x_{N-1}(n), x_{N-2}(n), \cdots, x_i(n), \cdots, x_0(n),\ \ x_i(n) \in A_{b=}\{0,1\}, i = 0,2, \ldots, N-1$$

Soit $\Delta$ le cycle minimal du générateur de séquences chaotiques sans perturbation. La perturbation est appliquée si $n = l \times \Delta, l = 0, 1, 2, \cdots$ (c.à.d. pour $n = 0$ et tout les $\Delta$ itérations ($mod(n, \Delta) = 0$), donc $\Delta$ est l'horloge du registre RDR). La séquence perturbée s'écrit selon l'équation suivante :

$$x_i(n) = \begin{cases} F[x_i(n-1)] & k \leq i \leq N-1 \\ F[x_i(n-1)] \oplus q_i\left(\frac{n}{\Delta} + 1\right) & 0 \leq i \leq k-1 \end{cases} \quad (2.15)$$

Où $F[x_i(n-1)]$ représente le ième bit de $F[X(n-1)]$ et $q_i(n)$ représente le bit de rang $i$ de la séquence de perturbation (sortie du RDR). Le bouclage du RDR est tel que :

$$q_{k-1}^{+}\left(\frac{n}{\Delta}+1\right)=q_k\left(\frac{n}{\Delta}+1\right)=g_0q_0(\frac{n}{\Delta}+1)\oplus g_1q_1(\frac{n}{\Delta}+1)\oplus\cdots\cdots g_{k-1}q_{k-1}(\frac{n}{\Delta}+1),$$

$$n=l\times\Delta,\qquad l=0,1,2,$$

et :

$[g_0,g_1,\cdots,g_{k-1}]$ sont les coefficients du polynôme primitif du registre et $Q=[q_0,q_1,\cdots,q_{k-1}]$ représente la valeur initiale non nulle du registre.

Notons que la séquence perturbatrice est appliquée sur les $k$ bits de poids faibles de $F[X(n-1)]$ . Si $\neq l\times\Delta$ , $l=0,1,2,$ ...., la sortie du générateur de séquences chaotiques n'est pas perturbée, donc : $X(n)=F[X(n-1)]$.

La période du système chaotique perturbé est donnée par :

$$L=\sigma\times\Delta\times\left(2^k-1\right) \tag{2.16}$$

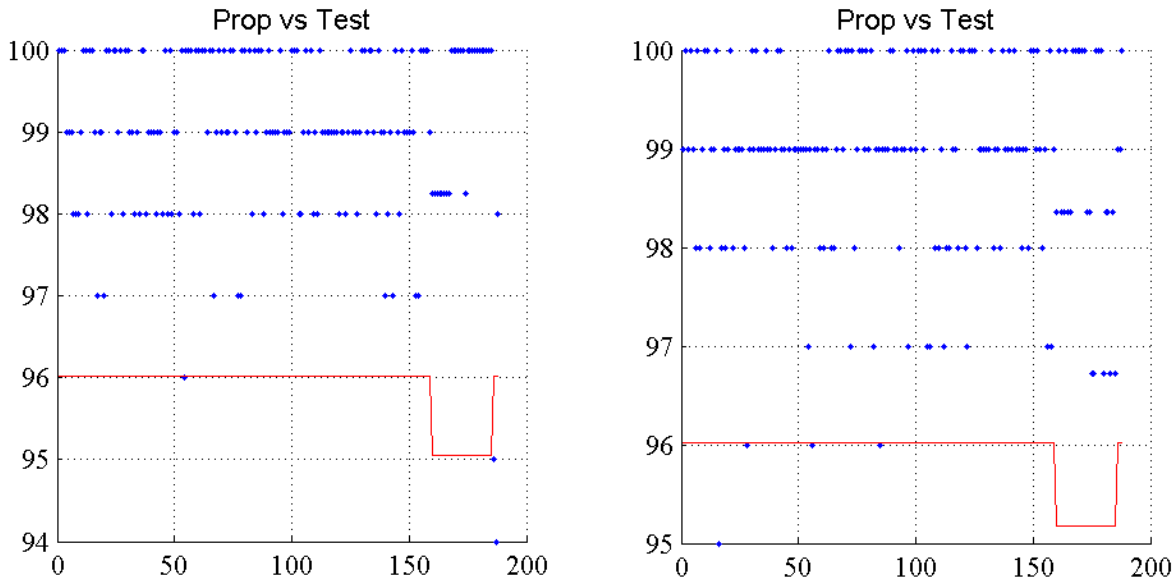
où σ est un entier positif. La période minimale est alors :

$$L=\Delta\times\left(2^k-1\right) \tag{2.17}$$

Etant donné que le premier générateur chaotique proposé utilise les cartes chaotiques perturbées Skew tent, et PWLCM, couplées par des fonctions booléennes non-linéaires, alors, il est utile de montrer d'abord les performances de ces deux cartes en termes de tests de NIST.

A ce propos, nous donnons dans la figure 2.20 a) et b) ci-dessous, les résultats de NIST sur les deux cartes chaotiques perturbées, la carte Skew tent et la carte PWLCM respectivement. Nous remarquons clairement qu'avec la technique de perturbation, les deux cartes en question passent les 188 tests de NIST.

Rappelons au passage que les 188 tests de NIST sont appliqués sur 100 séquences produites en sortie, dont chacune est composée de 1 million de bits.




a) b)

Fig. 2. 20: Proportion de réussite des 188 tests de NIST pour : en a) la carte Skew tent perturbée et en b) la carte PWLCM perturbée.

## 2.7 Générateurs chaotiques proposés

Dans ce paragraphe, nous décrivons les 3 générateurs chaotiques proposés et leurs performances.

Le premier générateur utilise une structure basée sur 4 cartes chaotiques perturbées (Skew tent, PWLCM, Skew tent, PWLCM) couplées par des fonctions booléennes non-linéaires. L'ordre des cartes chaotiques est variable.

Le deuxième générateur utilise une structure perturbée, basée sur m-cartes chaotiques couplées par une matrice de diffusion à base d'une carte Cat multidimensionnelle.

Le troisième générateur [El Assad et Noura, 2010-1], utilise une structure basée sur deux filtres récursifs contenant chacun une carte chaotique (fonction non-linéaire). Les filtres sont perturbés, couplés par ou exclusif et multiplexés.

### 3.7.1 Premier générateur chaotique proposé, basé sur des cartes chaotiques de base perturbées, couplées par des fonctions booléennes non-linéaires.

La structure du premier générateur chaotique proposé est illustrée par la figure 2.21. Il est composé de $m$ cartes chaotiques perturbées, couplées en parallèles par des fonctions booléennes non-linéaires et dispose de $m$ sorties.

Les $m$ -RDR ($m$-LFSR : Linear Feedback Shift Register) s'appuient sur $m$ polynôme primitifs de degrés $k_j$, ayant les sorties $Q_j, j = 1, \dots, m$. Les cartes chaotiques sont perturbées à l'instant initial et tous les $\Delta_j$, $j = 1, \dots, m$. ($\Delta_j$ est un entier positif) comme indiqué par l'équation suivante :

$$X_j(n) = \begin{cases} X_j(n) & si\ mod(n, \Delta_j) \neq 0 \\ X_j(n) \oplus mod\left(Q_j\left(\frac{n}{\Delta_j} + 1\right), 2^{k_j}\right) & si\ mod(n, \Delta_j) = 0 \end{cases} \quad j = 1, \cdots, m \tag{2.18}$$

L'équation ci-dessus, montre que la perturbation $Q_j$ est appliquée sur les derniers $k_j$ bits de poids faibles de $X_j(n)$, aux instants $n = l \times \Delta_j$ , $l = 0,1, \dots$, pour chacun des générateurs $j = 1, \dots, m$.

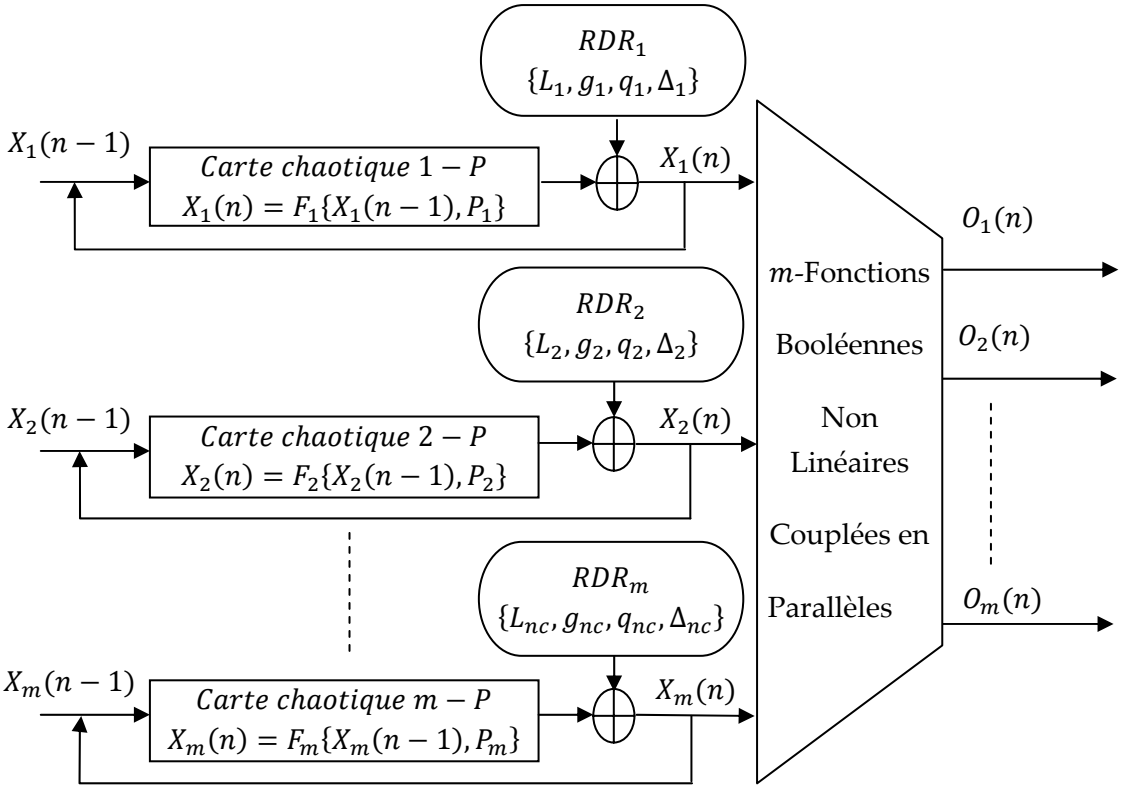



Fig. 2. 21: Structure du premier générateur chaotique proposé

Nous avons réalisé et implémenté un premier générateur chaotique basé sur la structure de la figure 2.21. Le générateur en question utilise 4 cartes chaotiques perturbées, $m = 4$ qui sont par ordre: Skew tent, PWLCM, Skew tent et PWLCM, et 4 fonctions booléennes non-linéaires, qui sont utilisées par ailleurs dans les 4 boucles de la fonction de hachage MD5, [*RFC 1321*, 1992], SHA-1 [RFC 3174, 2001]. Ces fonctions non-linéaires sont :

$$O_1 = C(X_1, X_2, X_3) = (X_1 \wedge X_2) \vee (\neg X_1 \wedge X_3)$$

$$O_2 = P(X_1, X_2, X_4) = X_1 \oplus X_2 \oplus X_4$$

$$O_3 = H(X_1, X_3, X_4) \;=\; (X_1 \wedge X_3) \vee (\neg X_4 \wedge X_3)$$

$$O_4 = Ch(X_2, X_3, X_4) \;=\; X_3 \oplus (X_2 \wedge \neg X_4)$$

Où ∧, ∨, ¬, ⊕, dénotent respectivement les operateurs logiques ET, OU, PAS et OU-exclusif.

Les degrés des polynômes primitifs utilisés dans l'implémentation sont par ordre respectif : $k_j = 21, 23, 27, 29$ bits, pour $j = 1, 2, 3\ et\ 4$ respectivement.

La taille de la clé secrète est assez grande ($> 338\ bits$) et résiste à l'attaque exhaustive. Par ailleurs, comme chaque sortie $O_j$ est couplée avec 3 sorties $X_j$, alors la longueur minimale de l'orbite $o_j$ d'une sortie $O_j$, est le plus petit commun multiple des longueurs des 3 orbites correspondants aux 3 sorties $X_j$. A titre d'exemple, la longueur minimale de l'orbite $o_2$ est donnée par :

$$o_{2min} = ppcm\left[\Delta_1 \times \left(2^{k_1} - 1\right),\ \Delta_2 \times \left(2^{k_2} - 1\right),\ \Delta_4 \times \left(2^{k_4} - 1\right)\right] \tag{2.19}$$

Par ailleurs, nous avons la relation suivante entre le $ppcm$ (le plus petit commun multiple) et le $pgcd$ (le plus grand commun diviseur) :

$$a \times b = \frac{ppcm(a, b)}{pgcd\ (a, b)} \tag{2.20}$$

Si le $pgcd\left[\Delta_1 \times \left(2^{k_1} - 1\right),\ \Delta_2 \times \left(2^{k_2} - 1\right),\ \Delta_4 \times \left(2^{k_4} - 1\right)\right]$, alors, la taille de la longueur de l'orbite minimale $o_{2min}$ est :

$$o_{2min} = \Delta_1 \times \left(2^{k_1} - 1\right) \times\ \Delta_2 \times \left(2^{k_2} - 1\right) \times\ \Delta_4 \times \left(2^{k_4} - 1\right) \tag{2.21}$$

Donc, le générateur en question, d'après les résultats du tableau 2.1 et des relations (2.21) et (2.34), possède des orbites de longueur très grande.

Nous montrons dans la figure 2.22, un exemple de résultats obtenus : a) variation discrète de $X(n)$ en fonction de $n$, b) espace de phase, c) attracteur, d) histogramme, pour la valeur de la clé secrète (paramètres de contrôle, conditions initiales des différents éléments du générateur) suivante, donnée sous forme de tableau, et utilisée pour la simulation :

| $i$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $X_i(0)$ | 2279919388 | 234840133 | 181129231 | 3.485624059 |
| $P_i$ | 2807807357 | 2546740132 | 3095434943 | 1705015261 |
| $Q(0)$ | 16192 | 64768 | 259075 | 1036302 |
| $\Delta_i$ | 1007 | 991 | 1007 | 99 |

Dans la figure 2.23, nous montrons les résultats obtenus en a) de l'auto-et l'intercorrélation de la sortie $O_1$et en b) les 188 tests de NIST. Ensuite, dans la figure 2.24, nous donnons l'espace de phase 3-D mettant en œuvre les trois sorties $O_1, O_2$ et $O_3$.

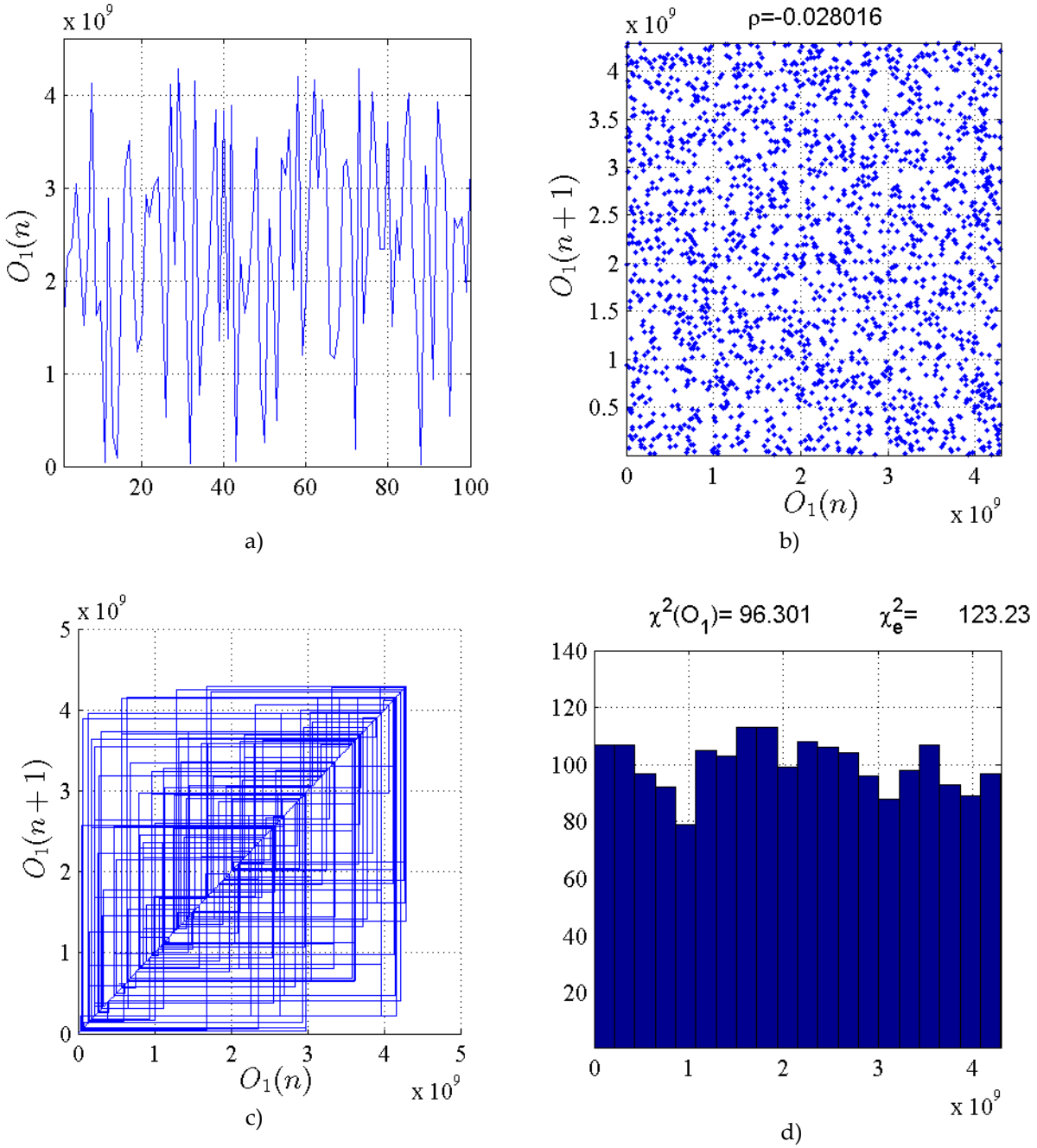



Fig. 2. 22: Exemple de résultats du premier générateur proposé : a) variation discrète de *X(n)* en fonction de *n* b) espace de phase, c) attracteur, d) histogramme.

,

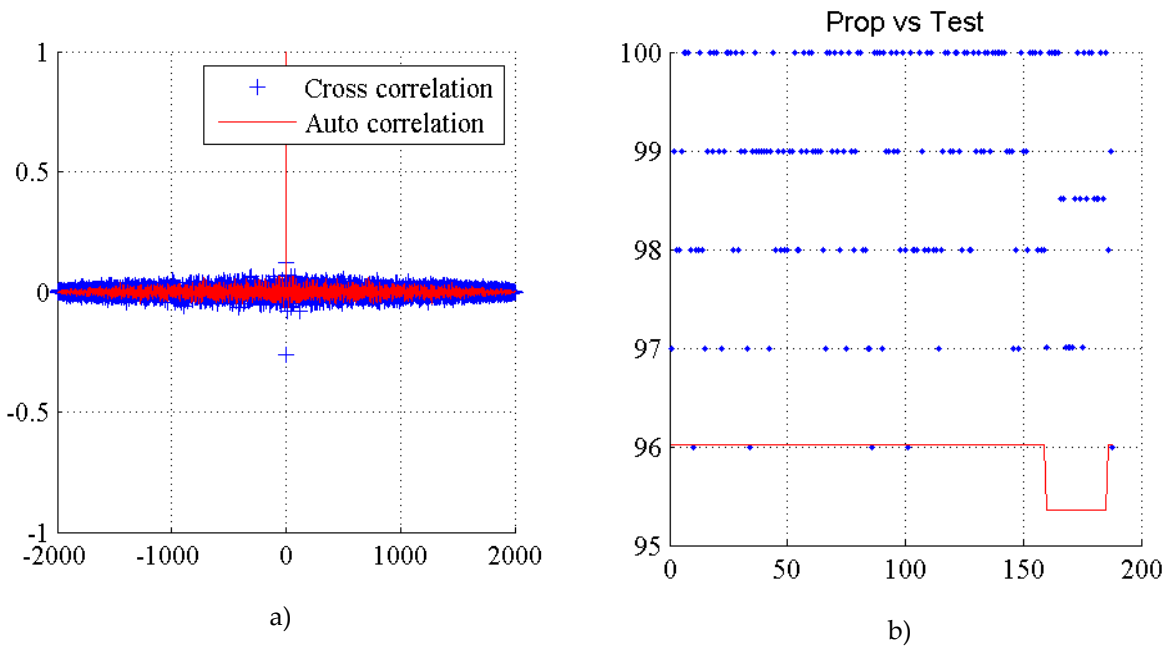



Fig. 2. 23: résultats : en a) Auto-et inter-corrélation de la sortie $O_1$ et en b) Proportion des tests de NIST appliqués sur les 100 séquences produites par la sortie $O_1$

$\rho(O_1,O_2)=$ -0.262 $\rho(O_1,O_3)=$ -0.0283 $\rho(O_2,O_3)=$ 0.00181

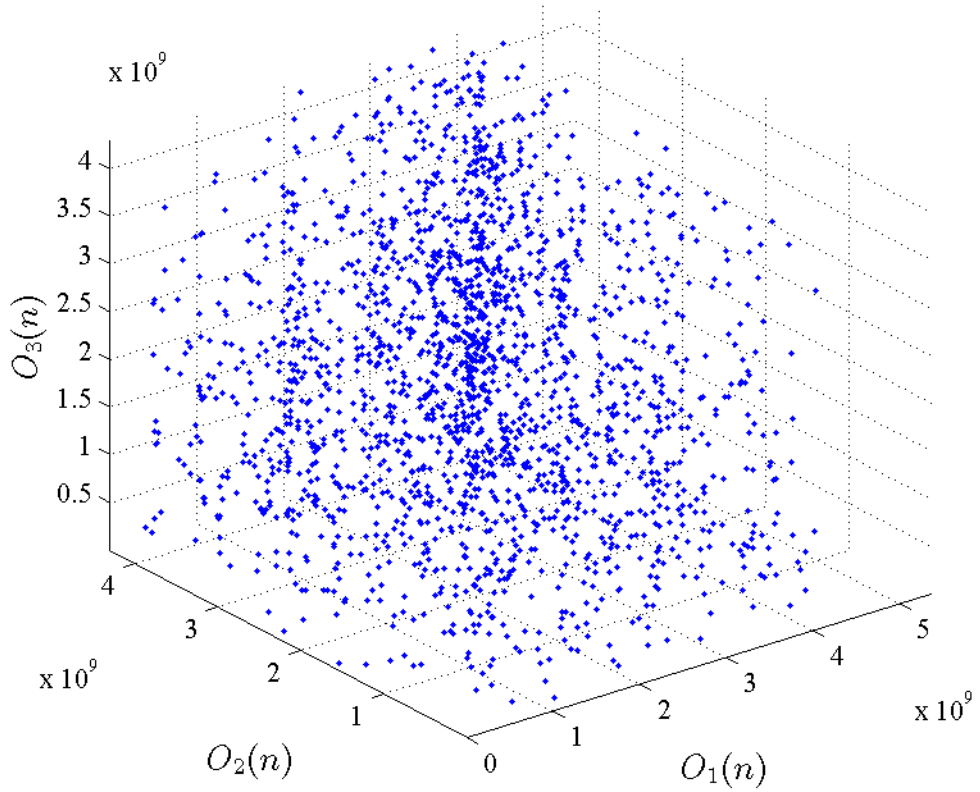



Fig. 2. 24 : Espace de phase $O_3=F(O_1,O_2)$

Tous ces résultats (qui sont complémentaires) montrent que les propriétés cryptographiques du premier générateur proposé sont bonnes. En effet, toutes les séquences chaotiques produites par ses sorties $O_j, j = 1, \cdots, 4$, passent les 188 test de NIST. Sur la figure 2.23 b), nous voyons clairement que les 188 tests de NIST de la sortie $O_1$, passent. Par ailleurs, l'espace de phase de la figure 2.24, montre clairement le caractère pseudo-aléatoire des différentes sorties. Ce générateur est apte à être utilisé dans des applications touchant à la sécurité de l'information. Nous l'avons utilisé dans les deux crypto-systèmes SPN-1 et SPN-2 proposés dans le chapitre 3, en tant que générateur de clés dynamiques des différentes couches : addition de clés, substitution, diffusion et permutation.

### 3.7.2 Deuxième générateur chaotique proposé : Confidentiel

### 3.7.3 Brevet déposé, référencé DI 03298-02 pour l'unité UMR6164

Partant de la carte Cat chaotique de base, Liu et al [Liu et al, 2008] ont développé une méthode flexible (détaillé en chapitre 3 au paragraphe 3.4.5.2) permettant d'avoir une carte Cat de dimension $m$, utilisée en tant que générateur chaotique performant. Ils ont montré qu'à partir de $m > 6$, le générateur peut avoir des propriétés cryptographiques acceptables.

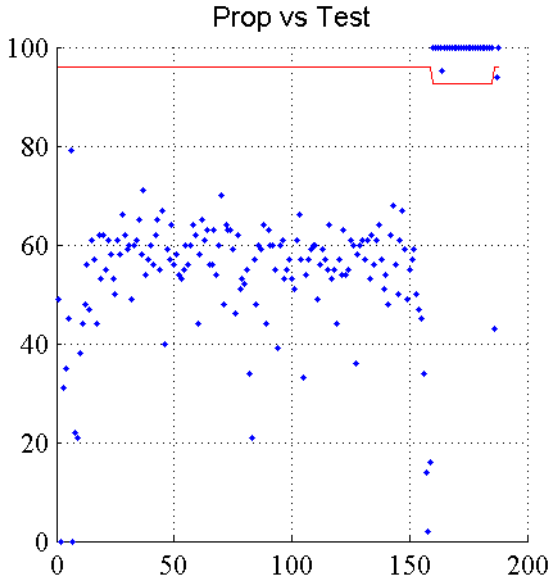



Fig. 2. 25: Résultats de NIST pour la carte cat 3-D avec perturbation

Cependant, comme la carte Cat est linéaire, l'utiliser en tant que générateur chaotique pose un problème. En effet, dans la figure 2.25, nous présentons les résultats de NIST obtenus sur la carte Cat 3 dimensions avec perturbation. Nous remarquons clairement, que la plupart des tests ne passent pas.

Fig. 2. 26: Structure du générateur du Brevet proposé

Les équations du générateur sont données par :

(3.22)

(3.23)

Fig. 2. 27: Illustration du calcul de la sortie $X_i(n)$ du générateur chaotique du Brevet

Dans la figure 2.28, nous montrons un exemple de résultats obtenus par ce générateur: a) variation discrète de $X_1(n)$ en fonction de $n$, b) espace de phase, c) attracteur, d) histogramme. Aussi un test de chi-carré appliqué sur les échantillons de la sortie $X_1$ (en prenant 100 intervalles) prouve l'uniformité des échantillons (voir valeurs obtenues sur l'histogramme).

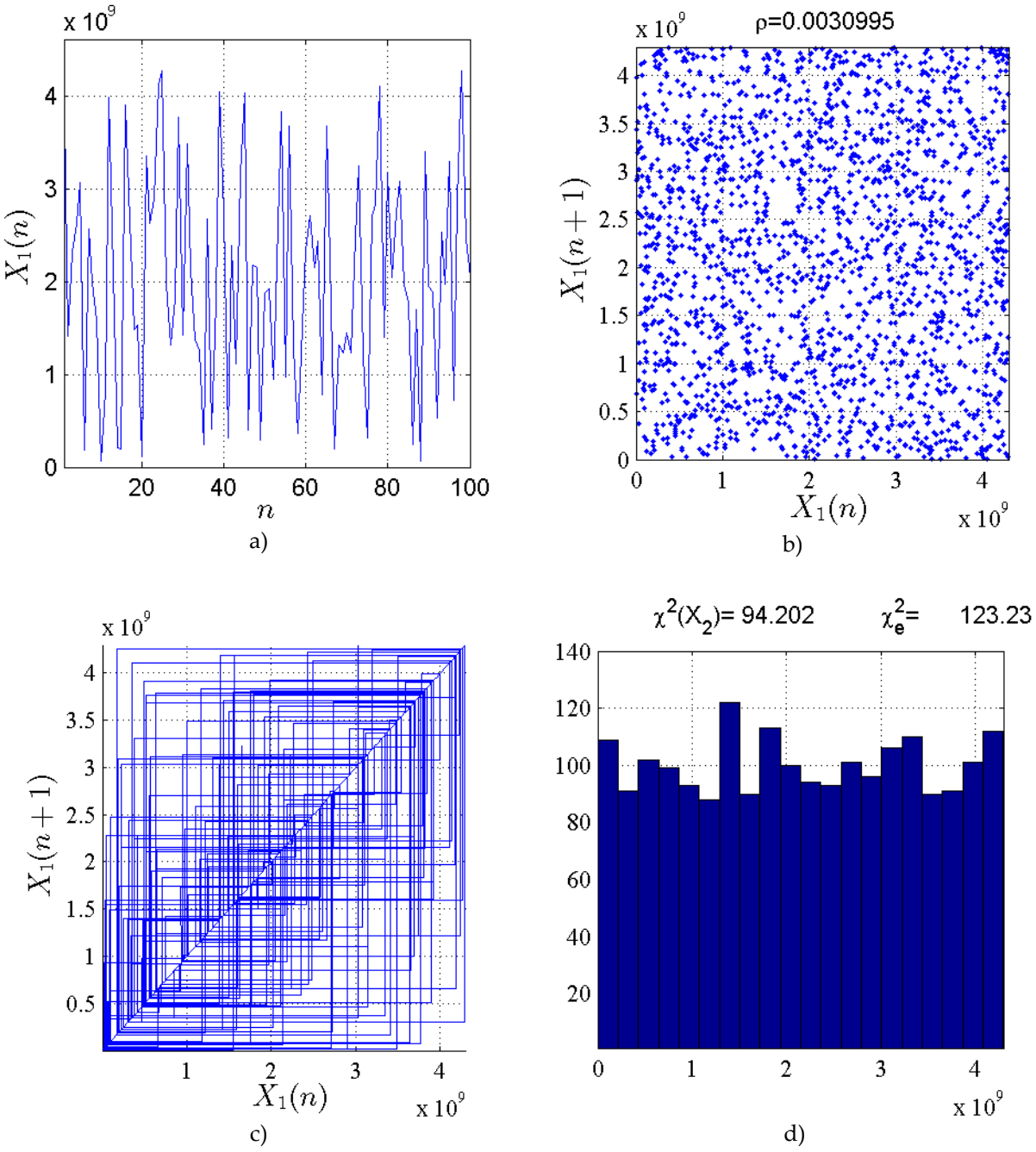



Fig. 2. 28: Exemple de résultats du deuxième générateur proposé : a) variation discrète de *X1(n)* en fonction de *n* b) espace de phase, c) attracteur, d) histogramme.

Dans la figure 2.29, nous montrons les résultats obtenus en a) de l'auto-et l'inter-corrélation de la sortie $X_1$et en b) les 188 tests de NIST. Ensuite, dans la figure 2.30, nous donnons l'espace de phase 3-D, mettant en œuvre les trois sorties$X_1, X_2$ et $X_3$.

La structure du deuxième générateur proposé est très efficace grâce à sa matrice de diffusion, ceci est confirmé par l'ensemble des résultats obtenus par tel générateur. Cependant, il est plus lent que le premier générateur.

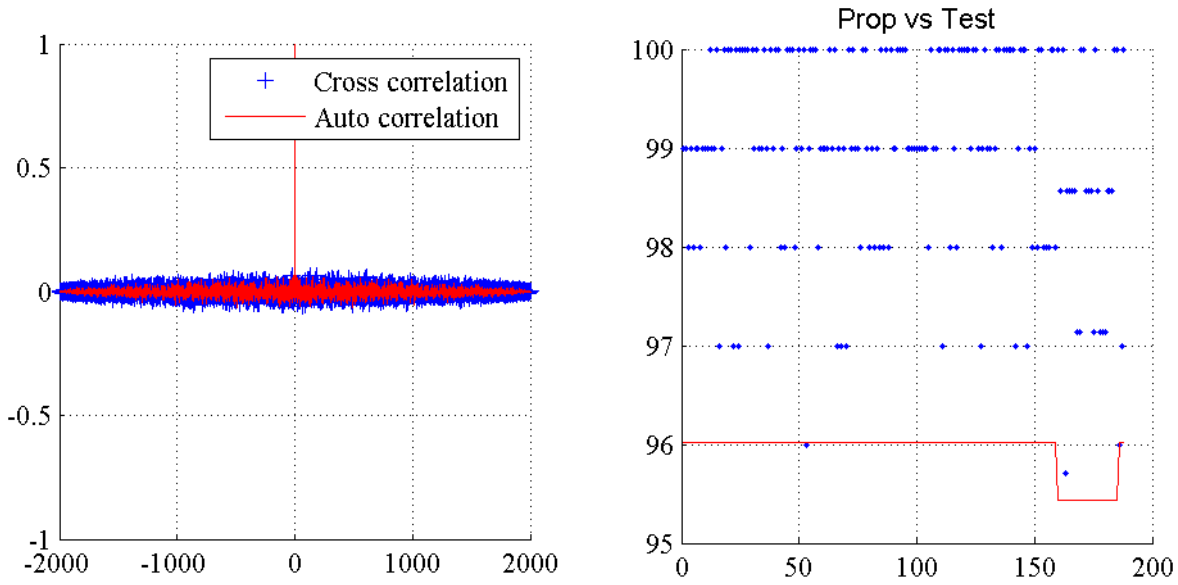



Fig. 2. 29: Résultats : en a) Auto-et inter-corrélation de la sortie $X_1(n)$ et $X_2(n)$ en b) Proportion des tests de NIST appliqués sur les 100 séquences produites par la sortie $X_1$.

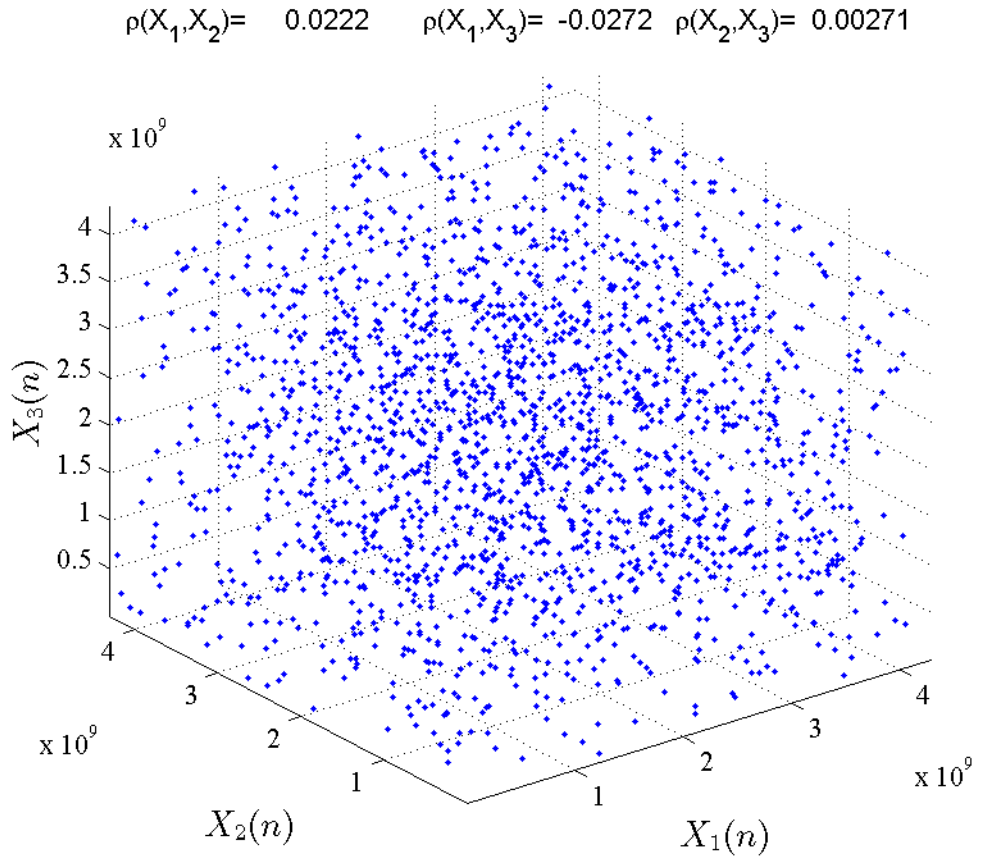



Fig. 2. 30: Espace de phase $X_3$=F($X_1$, $X_2$)

### 3.7.4 Troisième générateur proposé, basé sur une structure récursive, contenant une fonction non linéaire.

En 2001, Frank Dachselt et al, [Dachselt et al, 2001] ont proposé un système chaotique de chiffrement/déchiffrement auto-synchronisant utilisé en bande en base. La structure générale est composée d'une fonction non linéaire associée à plusieurs retards. Le chiffrement de l'information par flot est du type injection du message dans le filtre. Depuis ce système est devenu populaire et il a été utilisé dans différentes technologies : analogique, numérique, et optique [ACSCOM-2005]. Cependant, ce système est extrêmement sensible à la moindre perturbation externe et donc, ne peut pas être utilisé en pratique dans un environnement réel où des perturbations diverses existent.

Le troisième générateur proposé [El Assad et Noura, 2010-1], s'appuie sur une structure récursive perturbée, dont la fonction non linéaire utilisée, est une carte chaotique de base. Plus précisément, il est composé de deux cellules récursives non linéaires d'ordre $m$ en cascade parallèle, comme montré dans la figure 2.31. La première fonction non linéaire $NLF(\,)_1$ est la carte discrète PWLCM et la deuxième fonction non linéaire $NLF(\,)_2$ est la carte discrète Skew tent. Leurs équations ont déjà été données au début de ce chapitre.

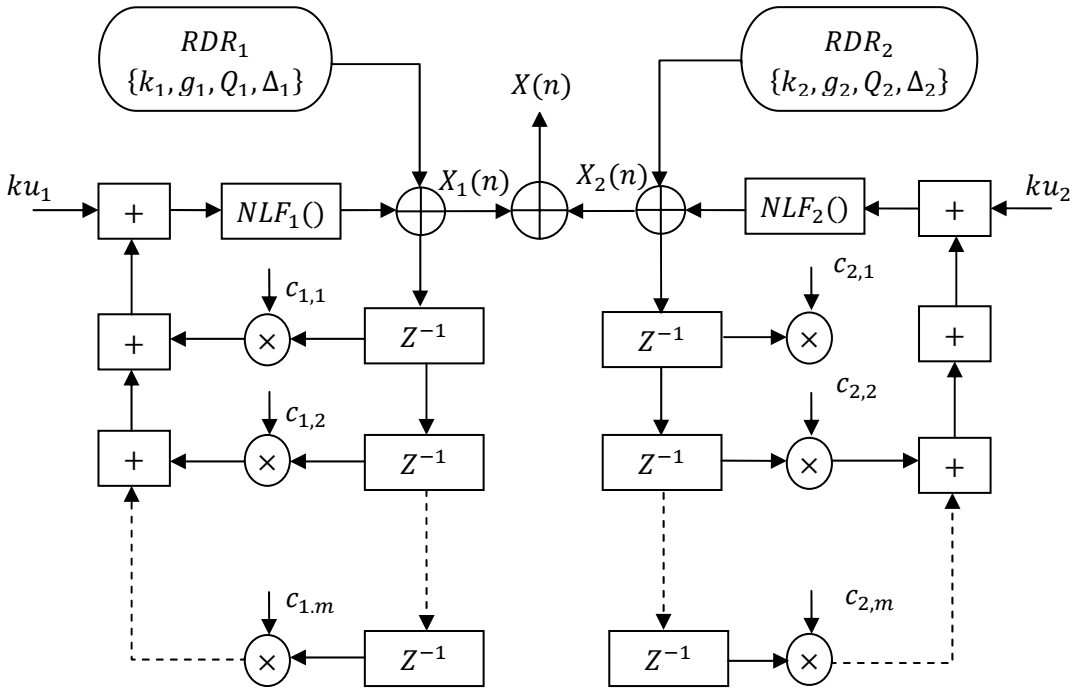



Fig. 2. 31: Structure du troisième générateur proposée.

La sortie du générateur est:

$$X(n) = X_1(n) \oplus X_2(n) \tag{2.24}$$

Avec :

$$X_1(n) = NLF_1\left(Mod\left\{\sum_{j=1}^{m} c_{1,j} \times X_1(n-j)\,, 2^N\right\}, P_1\right)$$

$$X_2(n) = NLF_2\left(Mod\left\{\sum_{j=1}^{m} c_{2,j} \times X_2(n-j)\,, 2^N\right\}, P_2\right)$$

Les deux séquences $X_1(n)$, et $X_2(n)$ sont produites par des cartes chaotiques différentes et sous deux clés secrètes de valeurs différentes, donc elles sont non corrélées.

*2.7.4.1 Taille de la clé secrète*

La taille de la clé secrète est très large, elle est formée des conditions initiales et paramètres des cartes chaotiques et des RDR et aussi des différents coefficients $c_{i,j}, i = 1,2\ et\ j = 1, 2, \cdots, m.$

Nous avons montré de façon expérimentale que la structure récursive est optimale pour un nombre de retard maximum $m = 3$.

Tous les résultats qui suivent, concernant ce générateur sont donnés pour $m = 3$. Dans ce cas la taille de la clé secrète du générateur proposée est de 547 bits au minimum, et ci-dessous, nous donnons le détail de sa composition.

| | |
|---|---|
| Cellule sans RDR<br>6 conditions initiales (6 X N bits)<br>2 paramètres de contrôle (N + N-1) bits<br>6 paramètres $c_{i,j}$, (6 X N bits)<br>2 entrées $k_u$ (2 X N bits) | RDR :<br>2 conditions initiales RDR ($k_1$ = 17 bits minimum, k2=19 bits) |

La valeur de la clé secrète utilisée pour la simulation (paramètres de contrôle, conditions initiales des différents éléments du générateur) est donnée sous forme tableau par :

| $m$ | 1 | 2 | 3 |
|---|---|---|---|
| $X_1$(m) | 2439816573 | 453304021 | 1741286952 |
| $X_2$(m) | 123183139 | 3153038243 | 993996956 |
| $c_{1,m}$ | 1443016135 | 1112813841 | 2001391952 |
| $c_{2,m}$ | 3353980715 | 2912924320 | 698331974 |

Et

$$P1 = 209460479,\ \ P2 = 1834739352\,, Q1 = 101171, Q2 = \ 444526, \qquad \Delta_1 = 99, \Delta_2 = 101$$

Les séquences générées ont des orbites assez longues. En effet, la valeur nominale $\Delta_{nom}$ de l'orbite d'une cellule sans perturbation est de l'ordre de $\Delta_{nom} = \sqrt{(2^N)^3} = 2^{3N/2} = 2^{48}\ pour\ N = 32$ [Lanford, 1998]. La longueur minimale de l'orbite générée par la sortie $X_1$

est : $o_1 \cong \Delta_{nom} \times 2^{17} = 2^{65}$, soit supérieure à $3.5 \times 10^{19}$, or l'âge de l'univers est de l'ordre de $10^{10}$.

La longueur minimale de l'orbite générée par la sortie $X_2$ est : $o_2 \cong \Delta_{nom} \times 2^{19} = 2^{67}$. La longueur minimale générée par la sortie $X$ est :

$o = ppcm(o_1, o_2)$

Si de plus (cas vérifié assez souvent) le $pgcd(o_1, o_2)$ est égal à l'unité, alors :

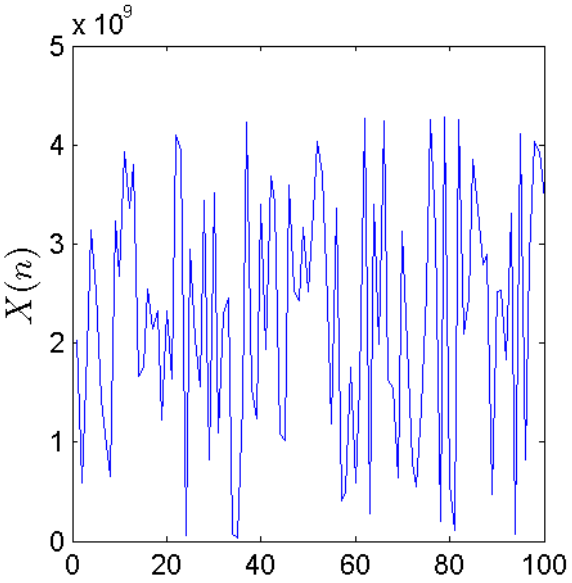
$$o = ppcm(o_1, o_2) = o_1 \times o_2 \tag{2.25}$$

Le troisième générateur proposé, permet non seulement de résoudre le problème de la dégradation des dynamiques chaotiques, mais aussi d'augmenter considérablement la longueur des cycles et d'assurer un bon niveau de sécurité.

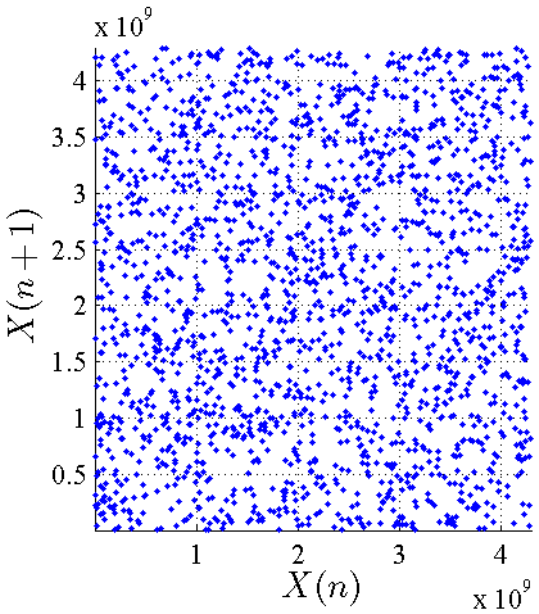
*Remarque* :

Le générateur du Brevet [El Assad et Noura, 2010-1] est composé de 28 générateurs élémentaires, dont chacun possède la même structure que le troisième générateur que nous venons de décrire. Ces 28 générateurs sont couplés par multiplexage.

Dans la figure 2.32, nous montrons un exemple de résultats obtenus par le troisième générateur: a) variation discrète de $X_1(n)$ en fonction de $n$, b) espace de phase, c) attracteur, d) histogramme.




a)




b)

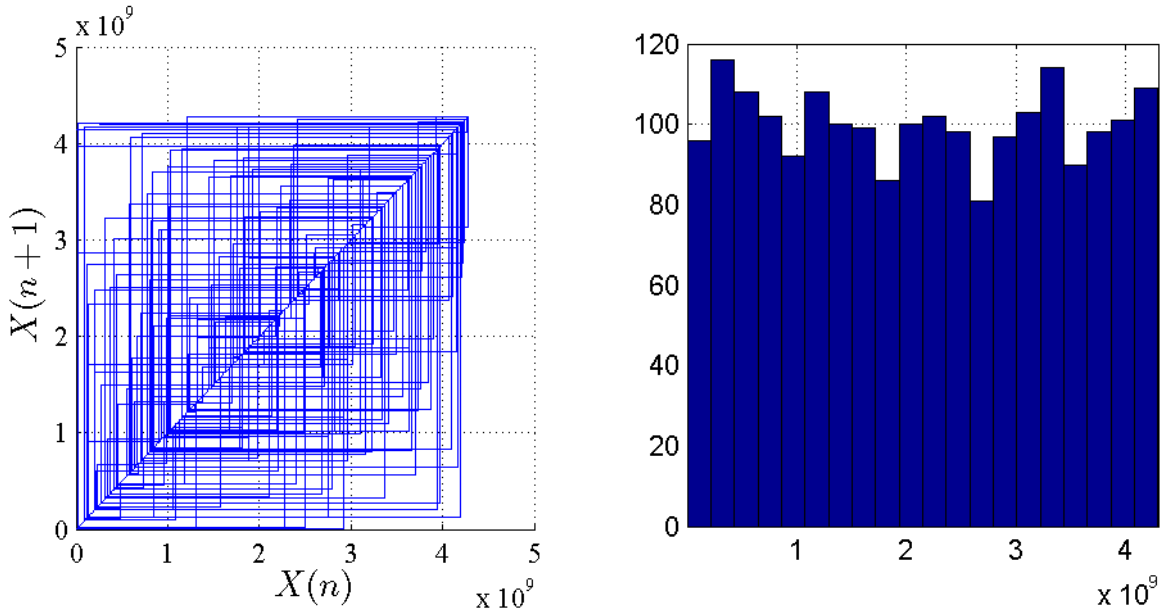



Fig. 2. 32: Exemple de résultats du troisième générateur proposé : a) variation discrète de *X(n)* en fonction de *n* b) espace de phase, c) attracteur, d) histogramme.

Dans la figure 2.33, nous montrons les résultats obtenus en a) de l'auto-et l'inter-corrélation de la sortie $X$ et en b) les 188 tests de NIST. Ensuite, dans la figure 2.34, nous donnons l'espace de phase du couple $X_1, X_2$.

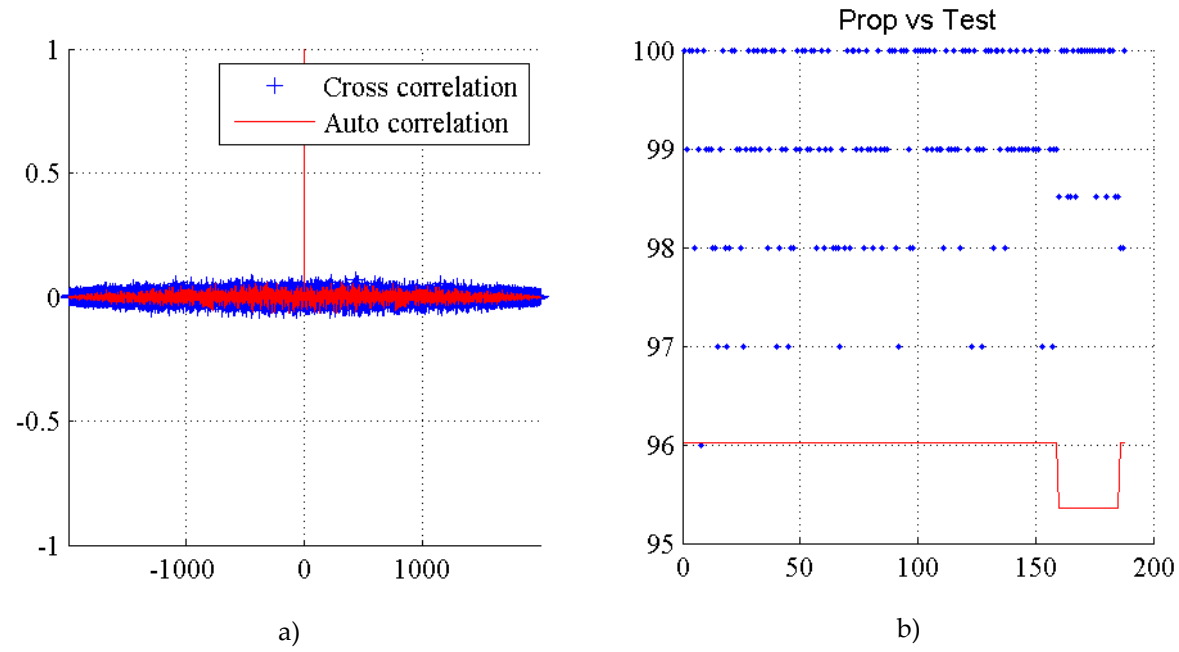



Fig. 2. 33: résultats : en a) Auto-et inter-corrélation de la sortie $X_1$ et $X_2$ en b) Proportion des tests de NIST appliqués sur les 100 séquences produites par la sortie X.

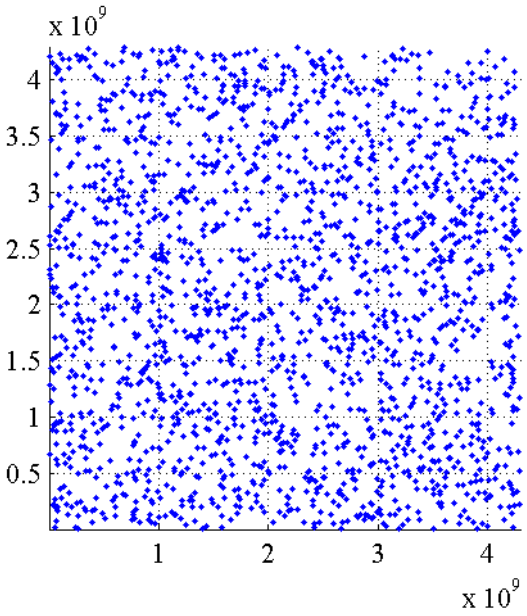



Fig. 2. 34: Espace de phase du couple ($X_1$, $X_2$).

Tous ces résultats montrent que le troisième générateur proposé, basé sur deux cellules récursives en parallèle, assure les propriétés de la confusion-diffusion et donc, il est utilisable dans les applications concernées par la sécurité de l'information.

## 2.8 Comparaison des performances avec des générateurs cryptographiques prouvés par NIST

Dans ce paragraphe, nous présentons des résultats comparatifs, basés sur les tests de NIST, obtenus par trois générateurs pseudo-aléatoires de nombres prouvés par NIST.

Le premier générateur est appelé Blum Blum Shub, ou générateur à résidu quadratique, puisque la théorie sous-jacente à ce générateur est liée aux résidus quadratiques modulo un entier $n = p \times q$, où $p$ et $q$ sont deux grands nombres premiers [Blum et al, 1986], [NIST SP 800-22, 2008]. Ce générateur est très robuste mais il est très lent comparé aux autres générateurs.

Les deuxième et troisième générateurs notés respectivement Hash-DRBG (Deterministic Random Bit Generators) et HMAC-DRBG sont basés sur une fonction de hachage SHA-512 respectivement sans clé et avec clé [NIST SP 800-90A, 2012]. Le troisième générateur est plus robuste que le deuxième générateur, mais il est moins rapide.

Les résultats de NIST des trois générateurs cités plus haut Blum Blum Shub, Hash-DRBG et HMAC-DRBG sont présentés dans les figures 2.35 et 2.36 a) et b). Comme prévu et indiqué dans la littérature ces générateurs passent les tests de NIST et sont robustes. Cependant, ces

générateurs sont moins rapides que les trois générateurs proposés et ceci est lié à la structure de la fonction de hachage SHA-512.

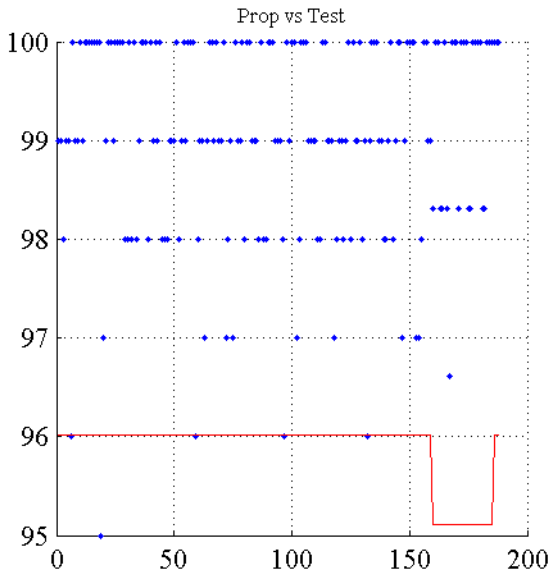



Fig. 2. 35: Proportion des tests de NIST appliqués sur les 100 séquences produites par le générateur Blum Blum Shub.

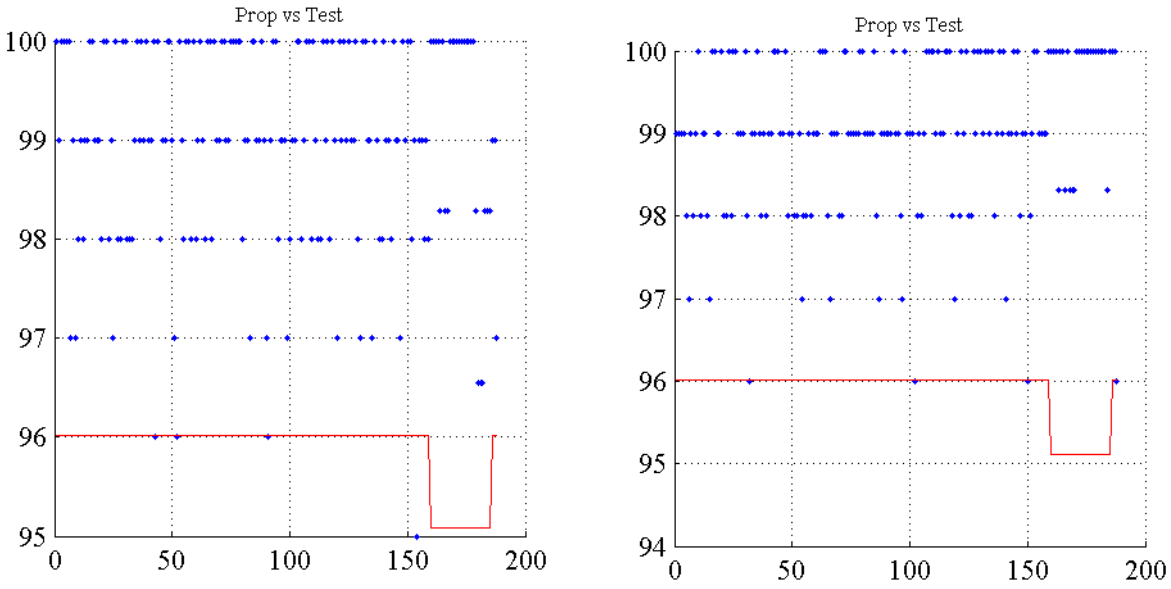



Fig. 2. 36: Proportion des tests de NIST appliqués sur les 100 séquences produites en a) par le générateur Hash-DRBG et en b) par le générateur HMAC-DRBG.

## 2.9 Conclusion :

Dans ce chapitre, nous avons tout d'abord, étudié et quantifié, selon les tests signal et NIST, les performances des cartes chaotiques de bases (Skew tent, PWLCM, Cat 2-D) en précision finie sur $N = 32$ bits. Puis, après avoir mis en évidence les problèmes posés par la précision finie, et exposé comment mesurer l'orbite chaotique, nous avons montré que la technique de perturbation associée à des techniques de couplage proposées, permettent de contourner les effets de la précision finie et ainsi d'augmenter considérablement le degré de la sécurité. Cette analyse, nous a permis de concevoir et de réaliser les trois générateurs chaotiques proposés.

Le premier générateur utilise une structure basée sur 4 cartes chaotiques perturbées (Skew tent, PWLCM, Skew tent, PWLCM) couplées par des fonctions booléennes non-linéaires. Le deuxième générateur utilise une structure perturbée, basée sur m-cartes chaotiques couplées par une matrice de diffusion à base d'une carte Cat multidimensionnelle. Le troisième générateur, utilise une structure basée sur deux filtres récursifs contenant chacun une carte chaotique (fonction non-linéaire). Les filtres sont perturbés, couplés par un ou exclusif.

Les trois générateurs proposés sont flexibles, possèdent une clé secrète très large, et satisfont les propriétés de confusion-diffusion, en en jugeant par l'analyse expérimentale des résultats obtenus lors des différents tests appliqués. Par ailleurs, ces générateurs sont plus rapides que les trois générateurs Blum Blum Shub, Hash-DRBG et HMAC-DRBG prouvés par Nist.

### *Annexe 2.1 : Procédure de création du fichier d'entrée (sortie du générateur à tester) sur lequel sont appliqués les tests de NIST*

Pour utiliser le logiciel NIST, nous devons créer un fichier binaire en format ASCII contenant $N_s = 100$ séquences, dont chacune est produite pour une clé aléatoire et composée de 1 million de bits. Donc, la taille totale du fichier est $Tf = 100 \times 10^6$ bits en format ASCII, ou octets.

La procédure de création du fichier en question se déroule comme suit :

D'abord nous générons $Ns$ =100 clés sécrètes, de façon aléatoire et uniforme.

Ensuite, pour chaque clé sécrète, nous produisons par le générateur sous test, une séquence de 31250 échantillons, chacun quantifié sur $N = 32$ bits, la précision finie du générateur. Les échantillons sont convertis en bits en format ASCII, puis stockés dans un fichier, et ainsi de suite pour les autres séquences qui seront stockées séquentiellement dans le même fichier. Le fichier binaire en format ASCII est l'entrée du logiciel NIST. Chacun des 188 tests de NIST est appliqué sur chacune des 100 séquences, produisant ainsi 100 P-values (basés sur un test de $\chi^2$). Pour un test donné, une séquence donnée passe le test si sa $P - value \geq \alpha = 0.01$, autrement la séquence ne passe pas le test ($\alpha$ est le seuil de confiance). Ainsi, pour chaque test, nous calculons la proportion des séquences qui passent le test en question, puis l'intervalle des proportions acceptables.

Ci-dessous, nous donnons le code Matlab pour la création du fichier binaire en format ASCII pour les 3 générateurs proposés.

```
N = 32;
%N : précisionfinie du générateur
m = 1.000.000;
%m : taille d'une séquence en bits
Ns = 100;
%Ns : nombre de séquences à générer
ng = 1;
%ng : type du générateur sous test parmi les 3 trois proposés
Filename = ['NIST m = '  num2str(m)  '-  NS = '  num2str(Ns)  'ng = '  num2str(ng)  '.txt']
fid = fopen(Filename,'w');
fclose(fid);
for i = 1:Ns
%géneration de séquence
if ng == 1
%premier generateur
    choix = [2 3 2 3];
% : 2 carte Skew tent
 % :  3 carte PWLCM

  [x1 y1 z1 w1] = chaoticgeneratorinteger1(N, m/N, choix);
se = x1;
```

```
elseif ng == 2
% deuxieme generateur
    choix = [2 3 2 3];
  [x1 y1 z1 w1] = chaoticgeneratorinteger2(N, m/N, choix);
se = x1;
elseif ng == 3
% troisiéme generateur
%d : nombre de retard
d = 2 ;
se = chaoticintegersequencefir(N, m/N, d) ;
end
% discrétisation
sb = dec2bin(se, N)
l = reshape(sb, 1, m);
% ecriture de sequence dans le fichier en question
fid = fopen(Filename, 'a + ');
fwrite(fid, l);
end
fclose(fid);
```

Le logiciel Nist peut être téléchargé sur le site web suivant :

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

Versions possibles du logiciel:

: version 1.8

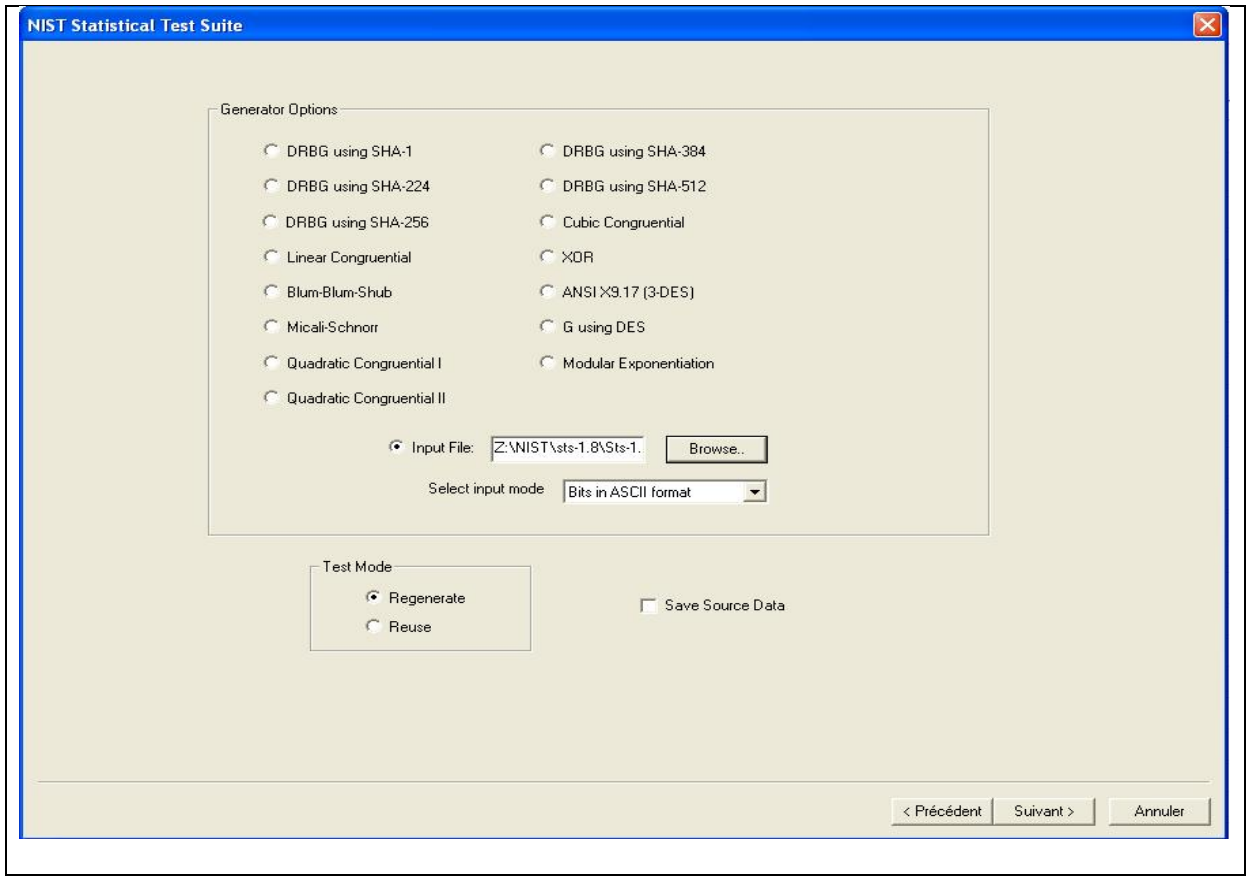
April 2010 : version sts-2.1

August 2010 : version sts-2.1.1

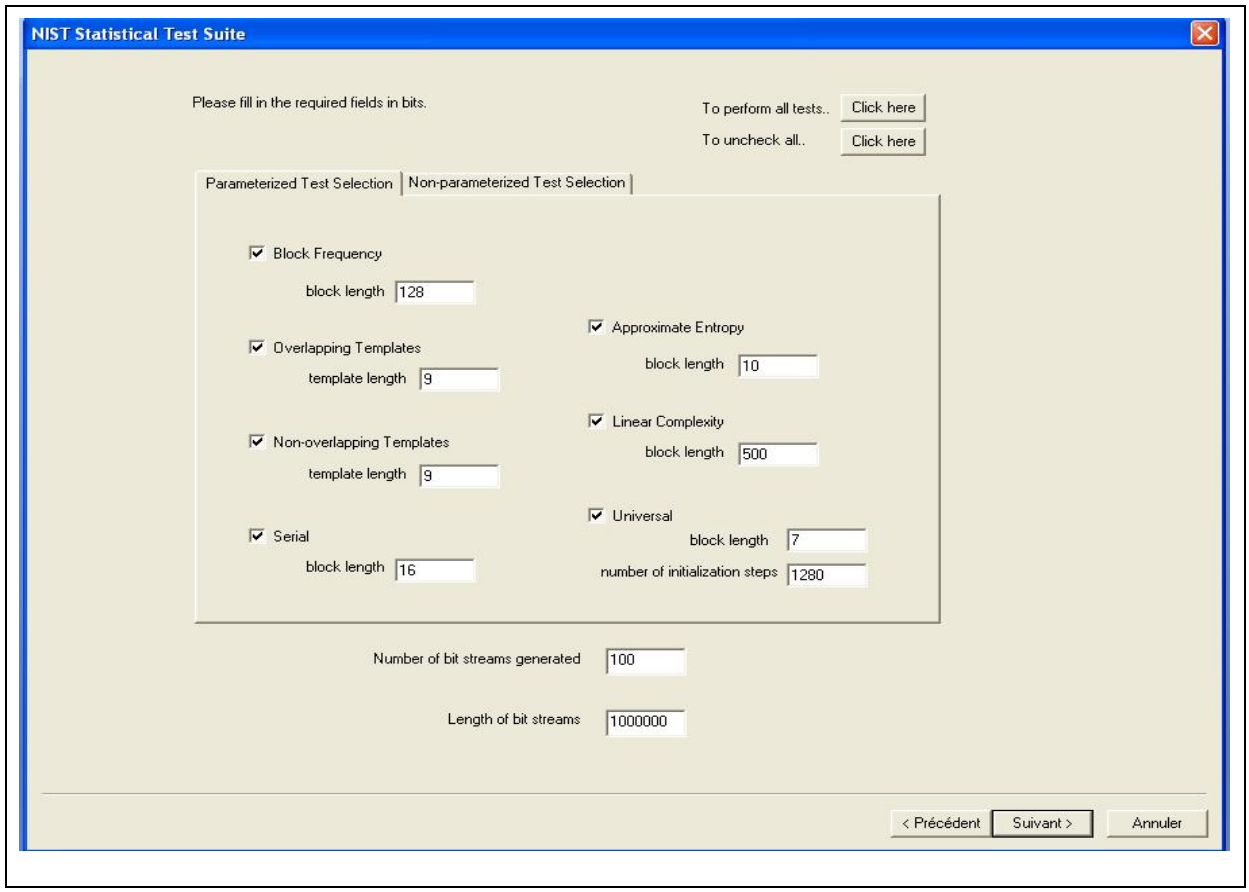
Nous avons travaillé avec la version 1.8 :

- Décompresser le fichier
- Copier le fichier mfc70.dll dans le répertoire C:\Windows\System32
- Exécuter « StsGui.exe » qui se trouve dans …\Sts-1.8\Release.

Ci-dessous, nous montrons les 2 premiers écrans permettant de savoir comment introduire le fichier binaire en format ASCII dans le logiciel.




Premier écran




Deuxième écran

[Addabbo et al, 2004] T. Addabbo, M. Alioto, S. Bernardi, A. Fort, S. Rocchi, V. Vignoli, "The digital tent map: performance analysis and optimized design as a source of pseudo-random bits", IEEE Transactions on Instrumentation and Measurement, Vol. 2, pp. 1301-1304, May 2004.

[Arnold et Avez, 1967] V. I. Arnold, A. Avez, "Ergodic Problems in Classical Mechanics" , New York, Benjamin, 1967.

[Billings et Bollt, 2001] L. Billings, E.M. Bollt, "Probability density functions of some skew tent maps", Chaos Solitons Fractals, Vol. 12, pp. 365–376, 2001.

[Blum et al, 1986] L. Blum, M. Blum, M. Shub, "A simple Unpredictable Pseudo-Random Number Generator", SIAM Journal on Computing, Vol. 15, no. 2, pp. 364-383, 1986.

[Cherrier et al, 2010] E. Cherrier, M. M'Saad, M. Farza, "High-gain observer synchronization for a class of time-delay chaotic systems, Application to secure communications", Journal of Nonlinear Systems and Applications, pp. 102-112, 2010.

[Dachselt et Schwarz, 2001] F. Dachselt, W. Schwarz, "Chaos and cryptography", IEEE Trans. Circuits and Systems–I, Vol. 48, pp. 1498–1509, 2001.

[Ecuyer, 1990] P. L'Ecuyer, "Random numbers for Simulation", Communications of the ACM, 1990, Vol. 33, no. 10, pp. 85-97.

[El Assad et al, 2008] S. El Assad, H. Noura, I. Taralova, "Design and analyses of efficient chaotic generators for crypto-systems", Advances in Electrical and Electronics Engineering- IAENG Special Edition of the World Congress on Engineering and Computer Science, Vol. I, pp. 3-12, 2008.

[El Assad et Noura, 2010-1] S. El Assad, H. Noura, "Brevet France : FR2958057 Générateur de séquences chaotiques", Mars 2010. Extension PCT.

[El Assad et Noura, 2010-2] S. El Assad, H. Noura, " Brevet France : FR2958100 Procédé de mesure d'orbite de séquences chaotiques et programme d'ordinateur correspondant", Novembre 2010. Extension PCT.

[ENT] ENT program, "A Pseudorandom Number Sequence Test Program", Fourmilab, [Online]. Available: http://www.fourmilab.ch/random/

[Fournier et al, 2006] D. Fournier, R. Lopez-Ruiz, A. K. Taha,"Route to chaos in three-dimensional maps of logistic type", Inst. für Math. Univ. Graz Grazer Math. Ber. Vol. 350, pp. 82-95, 2006.

[Fournier et al, 2011] D. Fournier, P. Chargé, L. Gardini, "Border Collision Bifurcations and Chaotic Sets in a Two-Dimensional Piecewise Linear Map", Communications in Nonlinear Science and Numerical Simulation Vol. 16, no. 2, pp. 916-927, February 2011.

[Grapinet et al, 2008] M. Grapinet, V.S. Udaltsov, L. Larger, J. M. Dudley, "Synchronisation and communication with regularly clocked optoelectronic discrete time chaos", Electronics Letters, Vol. 44, no. 12, pp. 764-766, June 2008.

[Grebogi et al, 1998] C. Grebogi, E. Ott, G. A. Yorke, "Roundoff-induced periodicity and the correlation dimension of chaotic attractors", Physical Review A, Vol. 38, no. 7, pp. 3688-3692, 1988.

[Jakimoski et al ,2001] G. Jakimoski, L. Kocarev, "Chaos and cryptography: block encryption ciphers based on chaotic maps", IEEE Transaction on Circuits and Systems I: Fundamental Theory and Applications, Vol. 48, no. 2, pp. 163-169, 2001.

[Jessa et Walentynowicz, 2001] M. Jessa, M. Walentynowicz, "Statistical properties of number sequences generated by 1D chaotic maps considered as a potential source of pseudorandom number sequences", The 8th IEEE Int. Conference on Electronics, Circuits and Systems, Vol. 1, pp. 449-455, 2001.

[Jessa, 2006] M. Jessa, "On some properties of chaotic maps implemented in finite-state machines", Proceedings International Symposium on Nonlinear Theory and its Applications, pp. 694-694, 2006.

[Khodor et al, 2010] N. Khodor, J.P Cances, V. Meghdadi, R. Quéré, "Performances of Chaos Coded Modulation Schemes Based on Mod-MAP Mapping and High Dimensional LDPC Based Mod-MAP Mapping with Belief Propagation. IJCNS, Vol. 3, pp. 495-506, 2010.

[Kocarev et Jakimoski, 2003] L. Kocarev, G. Jakimoski, "Pseudorandom bits generated by chaotic maps", IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications", Vol. 50, no. 1, pp. 123-126, 2003.

[Kwok et Tang, 2007] H.S. Kwok, W. K. Tang, "A fast image encryption system based on chaotic maps with finite precision representation", Chaos Solitons Fractals, Vol. 32, no. 4, pp. 1518-1529, May 2007.

[Lanford, 1998] O.E. Lanford III, "Informal remarks on the orbit structure of discrete approximations to chaotic maps", Experimental Mathematics, 1998, Vol. 7, no. 4, pp. 317-324, 1998.

[Li et al, 2001-1] S. Li, X. Mou, Y. Cai, "Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography", Progress in Cryptology, INDOCRYPT 2001, LNCS, Vol. 2247, pp. 316-329, 2001.

[Li et al, 2001-2] S. Li, Q. Li, W. Li, X. Mou, Y. Cai, "Statistical properties of digital piecewise linear chaotic maps and their roles in cryptography and pseudorandom coding", Proceedings of the IMA International Conference on Cryptography and Coding, Vol. 2260, pp. 205-221, 2001.

[Li et al, 2003] S. Li, X. Mou, , Y. Cai, , Z. Ji, J. Zhang, "On the security of a chaotic encryption scheme: Problems with computerized chaos in finite computing precision", Computer Physics Communications, Vol. 153, pp. 52–58, 2003.

[Lian et al, 2007] S. Lian, J. Sun, J. Wang, Z. Wang, "A chaotic stream cipher and the usage in video protection", Chaos, Solitons & Fractals, 2007, Vol. 34, no. 3, pp. 851-859, 2007.

[Lozi, 2006] R. Lozi, "Giga-periodic orbits for weakly coupled tent and logistic discretized maps", Anamaya Publishers, New Delhi, India, pp. 80-110, 2006.

[Lozi, 2008] R. Lozi, "New enhanced chaotic number generators", Indian journal of industrial and applied mathematics, Vol. 1, pp. 1–23, 2008.

[Lozi et Cherrier, 2011] R. Lozi, E. Cherrier," Noise-resisting ciphering based on a chaotic multi-stream pseudo-random number generator", IEEE, 6th International Conference for Internet Technology and Secured Transactions, ICITST-2011, pp. 91-96, Abu Dhabi, December 2011.

[Lozi et Fiol, 2009] R. Lozi, C. Fiol, "Global Orbit Patterns for One Dynamical Systems", Iteration Theory ECIT'08, 2009, pp. 1-33, 2009.

[Marsaglia, 1996] G. Marsaglia, "Diehard: a battery of tests of randomness," http://stat.fsu.edu/geo/diehard.html, 1996.

[Millerioux et Guillot, 2010] G. Millerioux, P. Guillot, "Self-synchronizing stream ciphers and dynamical systems: state of the art and open issues", International Journal of Bifurcation and Chaos Vol. 20, no. 9, pp. 2979-2991, 2010.

[Mooney, 2009] A. Mooney, "Chaos Based Digital Watermarking", Intelligent Computing Based on Chaos, Studies in Computational Intelligence, 2009, Vol. 184, pp. 315-332, 2009.

[NIST SP 800-22, 2008] NIST Special Publication 800-22 rev. 1. "A statistical test suite for random and pseudorandom number generators for cryptographic applications", August 2008.

[NIST SP 800-90A, 2012] NIST Special Publication 800-90A, "Recommendation for Random Number generation Using Deterministic Random Bit Generators", January 2012.

[Peng et al, 2007] J. Peng, M. You, Z. Yang, S. Jin, "Research on a Block Encryption Cipher Based on Chaotic Dynamical System", In Proceedings of the Third International Conference on Natural Computation , IEEE Computer Society, Vol. 5, pp. 744-748, 2007.

[Phatak et al, 1995] S. C. Phatak, S. Suresh Rao, "Logistic map: a possible random number generator", Physics Review E, Vol. 51, no. 4, pp. 3670-3678, 1995.

[*RFC 1321*, 1992] R. Rivest, "The MD5 message-digest algorithm", IETF Network Working Group, *RFC 1321*, 1992.

[RFC 3174, 2001] RFC 3174, US, "Secure Hash Algorithm 1", http://www.faqs.org/rfcs/rfc3174.html.

[Tao et al, 1998 ] S. Tao, W. Ruli, Y. Yixun, "Perturbance based algorithm to expand cycle length of chaotic key stream," Electronics Letters, Vol. 34, no. 9, 1998, pp. 873-874, 1998.

[Taralova et Fournier, 2002] I. Taralova, D. Fournier, "Dynamical study of a second order DPCM transmission system modeled by a piece-wise linear function", IEEE Transactions of Circuits and Systems I , IEEE publisher Vol. 49, November 2002.

[Ulam et Neumann, 1947] S. Ulam, J. V. Neumann, "Random ergodic theorems", Bull. Amer. Math. Soc. Vol. 51, p. 660, 1945.

[Verhulst, 1845] P. F. Verhulst, "Recherches mathématiques sur la loi d'accroissement de la population", Nouv. mém. de l'Academie Royale des Sci. et Belles-Lettres de Bruxelles, Vol. 18, pp. 1-41, 1845.

[Wang et al, 2004] S. Wang, W. Liu, H. Lu, J. Kuang, G. Hu, "Periodicity of chaotic trajectories in realizations of finite computer precisions and its implication in chaos communications", International Journal of Modern Physics B. 2004, Vol. 18, pp. 2617-2626, 2004.

[Wu et Guan, 2007] X. Wu and Z. Guan, "A novel digital watermark algorithm based on chaotic maps", Physical Letters A, Vol. 365, pp. 403–406, 2007.

[Zhang et al, 2000] H. Zhang, J. Guo, H. Wang, R. Ding, W. Chen, "Oversampled chaotic map binary sequences: definition, performance and realization", The 2000 IEEE Asia-Pacific Conf. on Circuits and Systems, pp. 618-621, 2000.

[Zheng et al, 2009] G. Zheng, D. Boutat, T. Floquet ,J-P Barbot, “Secure Communication Based on Multi-input Multi-output Chaotic System with Large Message Amplitude", Chaos Solitons & Fractal, Vol. 41, no. 3, pp. 1510-1517, August 2009.

# *Chapitre 3 : Conception et réalisation de crypto-systèmes basés chaos robustes et efficaces*

## 3.1 Introduction

Quelle est la structure d'un crypto-système chaotique permettant d'assurer la sécurité des communications avec efficacité et robustesse ? Comment peut-on construire un algorithme de chiffrement chaotique assurant un bon niveau de sécurité tout en ayant une architecture non complexe?

Comment peut on concevoir un crypto-système dynamique (donc très robuste) tout en étant compatible avec différents types de communication: réseaux mobiles, internet, etc ?

Dans ce chapitre nous nous attaquons à ces questions (problèmes) et nous proposons des solutions qui nous paraissent efficaces.

Selon le principe de Shannon, la sécurité fournie par un crypto-système dépend de la complexité de calcul. En effet, il montre que, plus on itère la fonction d'itération (round), constituée d'opérations de confusion et de diffusion, plus le crypto-système est robuste. Cependant, ceci est au prix d'un temps de calcul plus important, et donc une vitesse de chiffrement/déchiffrement non acceptable dans certaines applications nécessitant un travail en temps réel. Pratiquement, il n'y a pas de chiffrement efficace, telle qu'une seule itération soit suffisante pour assurer la sécurité contre les attaques passives et surtout actives.

Les structures assez simples des crypto-systèmes basés chaos d'une part, et les propriétés cryptographiques des signaux chaotiques d'autre part, font que ces crypto-systèmes sont assez efficaces à condition d'être bien conçus. En effet, dans la littérature, un certains nombre de crypto-systèmes basés chaos présentent des lacunes vis-à-vis de certaines attaques cryptographique et plus particulièrement l'attaque à texte en clair connu/choisi.

Dans la conception des deux crypto-systèmes basés chaos proposés de structure SPN, nous avons tenu compte de ces lacunes. La structure SPN est plus adéquate pour un calcul en parallèle que le schéma de Feistel. Cependant, le schéma de Feistel est plus flexible (en taille de bloc) et plus simple car la fonction d'itération inverse au déchiffrement peut être identique à la fonction d'itération utilisée au chiffrement.

Les deux crypto-systèmes proposés sont dynamiques et donc extrêmement sûrs. En effet, à chaque itération les clés des différentes couches, calculées à partir des sorties discrètes du générateur pseudo-chaotique, changent.

Les résultats de simulation montrent que les deux crypto-systèmes proposés sont robustes vis à vis de toutes les attaques cryptographiques connues. A ce sujet, nous avons étudié en détail et testé tous les critères utilisés au niveau des différentes couches et au niveau global, permettant de quantifier les performances des crypto-systèmes, à savoir (parmi d'autres) :

sensibilité aux textes en clair (mesure de l'effet d'avalanche), sensibilité à la clé secrète en émission et en réception, analyse statistique de la sécurité (histogramme, entropie,

corrélation, temps de calcul), bijectivité, non linéarité, distribution équiprobable et petite de la table XOR entrée-sortie, critère d'avalanche strict, critère d'indépendance des bits, nombre des branches linéaires, etc.

Par ailleurs, le deuxième crypto-système (SPN-2) est 2 fois plus rapide que le premier crypto-système (SPN-1), qui lui-même 3 fois plus rapide que l'AES. La rapidité du deuxième crypto-système découle du fait que sa couche de diffusion est réalisée par une matrice de diffusion (assez rapide) et non par une opération de permutation sur les bits (assez lente) utilisée par le premier crypto-système.

Le chapitre est organisé comme suit :

Dans la section 2, nous décrivons la structure des Crypto-systèmes basés chaos utilisant les réseaux de substitution-permutation. Dans les sections 3, et 4, nous présentons respectivement les deux crypto-systèmes proposés. Nous exposons dans la section 5, les outils utilisés pour quantifier les performances des crypto-systèmes, puis nous analysons dans la section 6, le niveau de sécurité de chacune des couches des crypto-systèmes proposés. Dans la section 7, nous présentons, analysons et comparons les performances globales des crypto-systèmes proposés avec ceux connus de la littérature. Dans la section 8, nous donnons une conclusion à ce chapitre.

## 3.2 Structure des crypto-systèmes basés chaos conçus et réalisés.

Nous décrivons dans ce paragraphe les crypto-systèmes proposés, qui sont généralement appropriés pour des mises en œuvre logicielles et matérielles. Les crypto-systèmes réalisés peuvent être utilisés pour assurer le service de confidentialité tel que : communications câblées, sans fil, et radio-mobiles, les transactions bancaires, etc.

Ces crypto-systèmes utilisent des algorithmes de chiffrement/déchiffrement par bloc de type symétrique.

Des résultats expérimentaux sont donnés pour chaque crypto-système ainsi qu'une étude comparative des performances de ces crypto-systèmes selon une batterie de tests les plus adéquats.

Tous les algorithmes de chiffrement/déchiffrement proposés sont réalisés pour tous les modes opératoires de la norme NIST (ECB, CBC, OFB, CTR, CMAC, CCM et GCM). Les trois premiers modes, à part le mode ECB, renforcent la confidentialité des crypto-systèmes et les trois derniers modes renforcent la confidentialité et réalisent l'authentification. Par ailleurs, les crypto-systèmes réalisés sont applicables à pratiquement tous les formats d'images JPEG, JPEG2000, JPSEC etc.

Nous présentons par la suite, le diagramme de chiffrement/déchiffrement des différents crypto-systèmes proposés en mode ECB, sachant que nous avons aussi, mis en œuvre les autres modes cryptographiques à savoir: CBC, OFB CTR, CCM, CMAC, GCM.

### 3.2.1 Crypto-systèmes basés chaos utilisant les réseaux de substitution-permutation

La structure moderne d'un algorithme de chiffrement par bloc, basé sur les réseaux de substitution-permutation (SPN), consiste, dans le cas où le processus de permutation est réalisé sur les octets, en quatre transformations de base, appelées aussi couches qui sont:

- la transformation d'addition de clés dynamiques
- la transformation de substitution
- la transformation de permutation
- la transformation de mélange linéaire

Chaque transformation est appliquée un certain nombre de fois (itérations) avant de passer à la transformation suivante, ceci permet d'augmenter considérablement la sécurité de l'algorithme, [Shannon, 1949]. Notons au passage que la transformation de mélange linéaire utilisée par l'algorithme AES pour assurer la diffusion, consomme pratiquement les ¾ du temps de calcul, ceci est causé par les multiples opérations de multiplication. D'où l'intérêt de proposer une couche de diffusion simple du point de vue de l'implémentation, donc rapide et aussi performante comparée à celle de l'AES.

Dans le cas où la permutation est faite sur les bits, la dernière couche (transformation de mélange linéaire) n'est pas nécessaire.

Le déchiffrement est réalisé par les transformations inverses appliquées dans l'ordre inverse du chiffrement.

En général, dans les crypto-systèmes basés chaos, les différentes couches citées plus haut sont appliquées en temps réel sur chaque bloc, utilisant des clés dynamiques comme paramètres de contrôle. Certains algorithmes utilisent la même clé dynamique d'itération pour chaque transformation, d'autres, dont font partie les deux premiers crypto-systèmes proposés, utilisent des clés dynamiques différentes pour chaque itération, ce qui augmente considérablement la robustesse de l'algorithme. Notons aussi que la sécurité d'un algorithme de chiffrement dépend grandement de la robustesse de la couche de substitution. Plus le processus de substitution est fortement non-linéaire, plus grande est la robustesse globale de l'algorithme.

Pour des applications qui ont une contrainte de ressources (capacité de mémoires et moyen de calculs) comme les réseaux mobiles, le processus de substitution peut être réalisé par des transformations non-linéaires, comme par exemple la carte chaotique skew tent qui est inversible. Le processus de permutation peut lui aussi être réalisé par la même carte chaotique skew tent légèrement modifiée (avec deux versions selon que la permutation est réalisée sur les bits ou sur les octets) ou aussi par une fonction polynomiale comme celle utilisée par l'algorithme RC6 [ Rivest et al, 1998].

Pour les applications qui nécessitent le traitement d'une importante quantité d'information et aussi la transmission à un grand débit, et tout cela en temps réel, comme la transmission des vidéo chiffrées, les couches de substitution et de permutation sont générées dynamiquement et sauvegardées dans des mémoires sûres, pour ensuite être utilisées par l'algorithme de chiffrement/déchiffrement. Dans ce cas, le troisième crypto-système proposé s'applique. A ce propos, plusieurs techniques, basées chaos, de création des boîtes de substitution (boîtes-S, S-boxes) et de permutation (boîtes-P, P-boxes) dépendantes de clés dynamiques sont proposées. Les boîtes de substitution sont conçues pour une entrée/sortie de 8 bits (traitement octet par octet).

Les boîtes de permutation sont essentiellement conçues à partir des cartes chaotiques monodimensionnelles (1-D) et bidimensionnelles (2-D) comme les cartes cat et standard qui sont bijectives mais non inversibles à cause de la fonction modulo. Notons au passage que la carte chaotique standard est extrêmement coûteuse en implémentation matérielle, car elle utilise la fonction sinus qui sera forcement calculée en virgule flottante.

Pour cette catégorie de crypto-systèmes, le gain obtenu en termes de quantité d'information traitée et de temps de traitement est balancé par la robustesse globale obtenue.

Chaque algorithme de chiffrement/déchiffrement est aussi caractérisé par :

- les composantes du SPN (ou la fonction d'itération)
- taille du bloc à chiffrer/déchiffrer en bits : 64, 256, 1024 bits
- taille de la clé secrète qui doit être supérieure à 128 bits, pour contourner les attaques exhaustives.

Par ailleurs, pour tous les algorithmes proposés, avant de commencer le processus de chiffrement, d'abord, nous lisons l'image $I$ et l'exprimons sous forme d'un tableau 1-D de taille $L \times C \times P$, où $L$ est le nombre de lignes, $C$ est le nombre de colonnes et $P$ est le nombre de plan (P=3, pour une image couleur RGB). Ensuite, nous divisons le tableau 1-D en blocs de taille fixe $Tb = 2^i$ bits, avec $i$ entier. Le nombre d'octets dans un bloc est égal à $Tb/8 = 2^{i-3}$.

## 3.3 Premier algorithme SPN-1 proposé :

Dans la figure 3.1, nous donnons le schéma de base du premier crypto-système proposé [Noura et al, 2011]. Sa structure est assez simple, mais très efficace du point de vue cryptographique comme nous le montrons plus loin. En effet, en contraste avec la plupart des crypto-systèmes les plus récents, les paramètres de contrôle utilisés dans les opérations d'addition de clés, de substitution et de permutation sont chaotiques et dynamiques pour chaque itération.

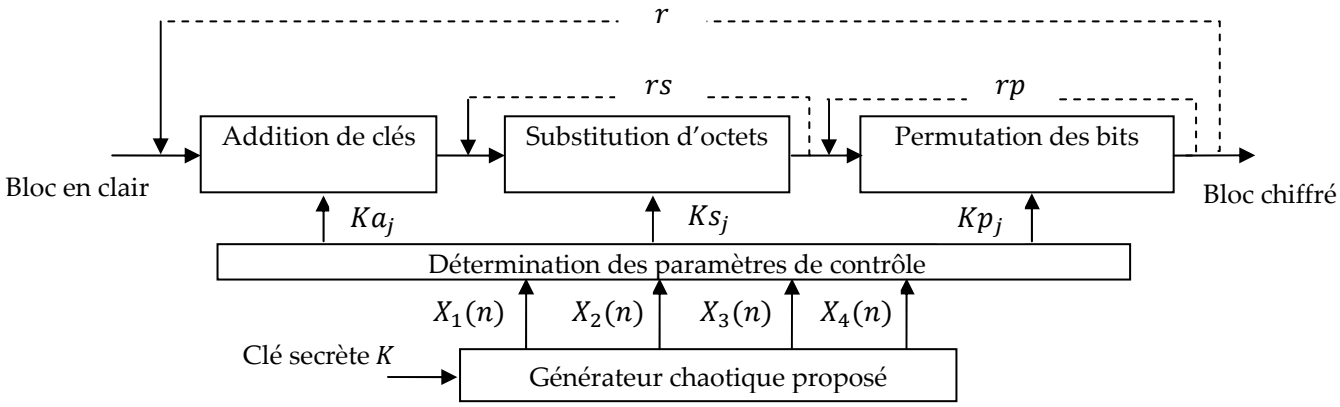



Fig. 3.1: Schéma de base du premier crypto-système proposé

Le premier crypto-système proposé utilise dans la couche d'addition, une addition de clés dynamiques modulo $2^8$ pour les itérations impaires et XOR pour les itérations paires (le nombre total r d'itérations doit être pair). Aussi, il utilise deux cartes chaotiques, la tent pour la couche de substitution, la cat ou la standard ou la tent modifiée pour la couche de permutation.

Pour chiffrer un bloc d'information, le bloc en question subit une addition de clés, suivie d'une opération de substitution sur les octets, répétée $rs$ fois, suivie par une opération de

permutation sur les bits répétée $rp$ fois. Ensuite, les trois opérations précédentes sont itérées $r$ fois, permettant ainsi de réaliser les propriétés de confusion et de diffusion du bloc sous traitement. Toutes ces opérations sont décrites en détail plus loin dans ce paragraphe.

A chaque itération globale, des clés dynamiques $Ka$, $Ks$ et $Kp$ sont produites par un générateur de séquences pseudo chaotiques de valeurs entières, assurant ainsi un haut niveau de sécurité pour le bloc chiffré.

Avant d'exposer en détail les différentes opérations d'une fonction d'itération utilisée dans les processus de chiffrement et de déchiffrement, commençons d'abord par décrire la structure de chacune des clés dynamiques $Ka$, $Ks$ et $Kp$ nécessaires à cet effet.

### Structure de la clé d'addition dynamique $Ka$

$$Ka = [Ka_1 || Ka_2 || \dots || Ka_r] \quad (3.1)$$

$$\text{Avec} \quad Ka_j = [Ka_{j,1} || Ka_{j,2} || \dots || Ka_{j,nk}],\ j = 1, 2, \dots, r \ et \ nk = Tb/N$$

La taille de $Ka_{j,i}$ est justement $N$

### Structure de la clé de substitution dynamique $Ks$

$$Ks = [Ks_1 || Ks_2 || \dots || Ks_r] \quad (3.2)$$

$$\text{Avec} \quad Ks_j = [a_{j,1} || a_{j,2} || \dots || a_{j,na}],\ j = 1, 2, \dots, r \ et \ 1 \leq a_{j,i} \leq Q$$

Où $a_{j,i}$ est le paramètre de contrôle qui varie entre 1 et $Q = 2^8$, puisque le processus de substitution se fait sur les octets. Chaque paramètre de contrôle $a_{j,i}$ $i = 1, 2, \dots, na$ est itéré, sur le bloc sous traitement de taille $Tbo = Tb/8$ octets, $rs$ fois.

### Structure de la clé de permutation dynamique $Kp$ dans le cas des cartes chaotiques cat et standard

$$Kp = [Kp_1 || Kp_2 || \dots || Kp_r] \quad (3.3)$$

$$\text{Avec} \quad Kp_j = [Kp_{j,1} || Kp_{j,2} || \dots || Kp_{j,rp}],\ j = 1,2, \dots, r.$$

Dans le cas d'utilisation de la carte Cat, la clé dynamique $Kp_{j,i}$ de permutation est composée de quatre éléments $\{u, v, ri, rj\}$, et de trois éléments $\{k, ri, rj\}$ dans le cas de la carte standard, comme montré sur la figure 3.2.

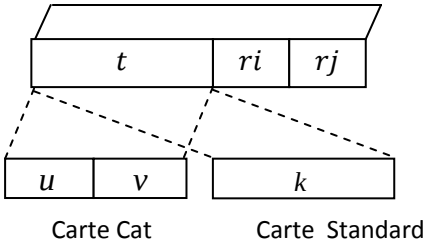



Fig. 3.2: Forme de la clé dynamique de permutation $Kp_{j\,i}$

Les paramètres de contrôle de base $u$ et $v$ de la carte cat varient de:
$0 \leq u, v, r_i, r_j \leq M - 1 = 2^q - 1$, où $q = log_2(M)$ est le nombre de bits nécessaire pour représenter $u$ et $v$. Le paramètre de contrôle de base $k$ de la carte standard varie entre :
$0 \leq k \leq M^2 - 1 = 2^{2q} - 1$. Ici $Tb = 2^i$, avec $i$ entier pair, et $M = \sqrt{Tb} = 2^{i/2}$.
Pour les deux cartes chaotiques, nous incluons un couple $ri$ , $rj$ pour augmenter l'espace de la clé dynamique. Leurs valeurs varient de 0 à $M - 1$.

### Structure de la clé de permutation dynamique $Kp_j$ dans le cas de la carte chaotique skew tent map

$$Kp_j = [b_{j,1} || b_{j,2} || \dots || b_{j,nb}], \qquad j = 1, 2, \dots, r \; et \; 1 \leq b_{j,i} \leq Q = Tb$$

Chaque paramètre de contrôle $b_{j,i}$ $i = 1, 2, \dots, nb$ est itéré sur le bloc sous traitement $rp$ fois.
La taille en bits de chacune des clés dynamiques $Ka_j$, $Ks_j$ et $Kp_j$ nécessaires pour la réalisation du processus de chiffrement pour une itération $j$ est :
$Tb$ pour $Ka_j$, $8 \times na$ pour $Ks_j$, $4q \times rp$ pour $Kp_j$ dans le cas des cartes cat et standard et $nb \times log_2(Tb) \times rp$ pour $Kp_j$ dans le cas de l'utilisation de la carte tent.
Alors, la taille en bits de chacune des clés dynamiques $Ka_j$, $Ks_j$ et $Kp_j$ pour chiffrer/déchiffrer un bloc est $r$ fois celle d'une itération $j$.

Le processus de chiffrement/déchiffrement détaillé se déroule selon le schéma de la figure 3.3

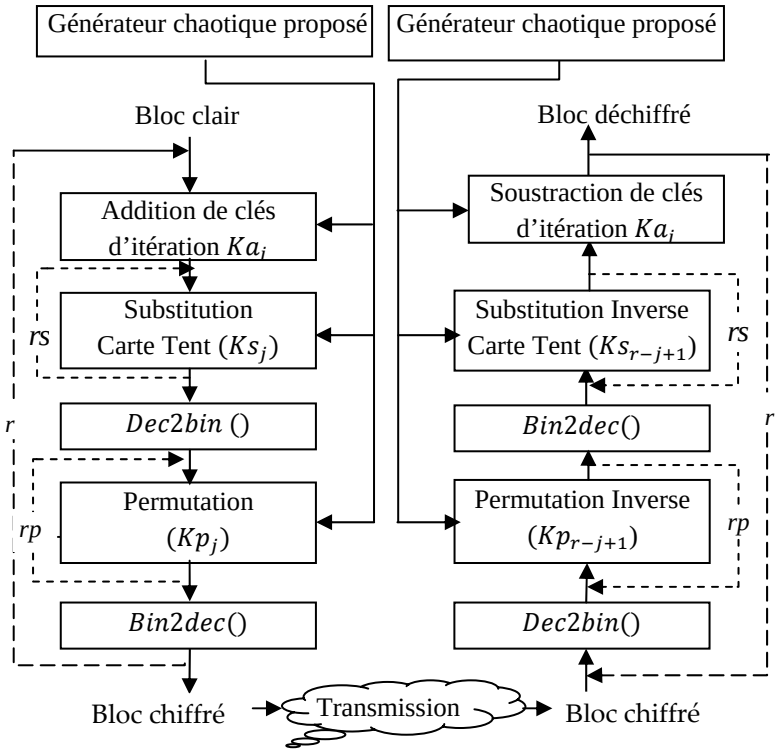



Fig. 3.3 : Processus détaillé du chiffrement/déchiffrement

### 3.3.1 Description du processus de chiffrement

Le chiffrement d'un bloc informatif s'effectue comme suit : d'abord, nous appliquons une opération d'addition de clés dynamiques, puis une opération de substitution sur les octets itérée plusieurs fois, ensuite une opération de conversion décimal/binaire (Dec2bin), suivie d'une opération de permutation sur les bits itérée un nombre de fois, et enfin une opération de conversion binaire/décimal (Bin2dec). Toutes les opérations précédentes sont répétées $r$ fois pour réaliser le chiffrement du bloc sous traitement. Nous détaillons ci-dessous les trois opérations principales mises en œuvre dans le processus de chiffrement.

#### *2.3.1.1 Couche d'addition des clés dynamiques*

Le premier crypto-système proposé utilise dans la couche d'addition, une addition de clés dynamiques modulo $2^8$ pour les itérations impaires et XOR pour les itérations paires (le nombre total $r$ d'itérations doit être pair)

La couche d'addition de clés dynamiques mélange le bloc en clair de taille $Tb/8$ octets avec une clé d'addition. L'opération de mélange s'effectue sur les octets (octet par octet) et dépend de la valeur paire/impaire de l'itération $j$. Par ailleurs, le nombre total d'itérations $j = r$ doit être pair.

Si la valeur $j$ de l'itération est impaire, on applique l'opération arithmétique modulo $2^8$, sinon, on applique l'opération logique ou exclusive, comme suit, où $O$ est un octet du bloc en clair et $K$ un octet de la clé d'addition [Socek et al, 2005 ]:

$$\begin{cases} mod[(O+K), 2^8] & si\ j\ est\ impair \\ O \oplus K & si\ j\ est\ pair \end{cases} \quad (3.4)$$

L'opération modulo est définie par : $Mod(c,d) = c - d \times \lfloor c/d \rfloor$

L'utilisation alternée addition modulo et ou exclusive permet de rendre extrêmement difficile la cryptanalyse.

Notons aussi au passage que l'opération inverse utilisée dans le processus de déchiffrement est donnée par l'équation suivante :

$$\begin{cases} mod[(O-K), 2^8] & si\ j\ est\ impair \\ O \oplus K & si\ j\ est\ pair \end{cases} \quad (3.5)$$

La seule modification est que l'opération d'addition modulo est remplacée par l'opération soustraction modulo.

Pour réaliser l'opération de mélange octet par octet de la clé dynamique $Ka_j$ avec le bloc informatif $B$, de taille $Tb/8$ octets, d'abord, nous devons extraire de chaque clé $Ka_{j,i}$ de taille $N = 32$ bits, avec $i = 1, \cdots, nk = Tb/N$ les différents octets au nombre de quatre $(N/8 = 4)$. Ensuite, nous pouvons former le tableau de la clé d'itération $Ka_j$, de taille $Tbo = Tb/8$ octets, par concaténation des différentes clés $Ka_{j,i}$, $i = 1, \cdots, nk = Tb/N$.

Deux méthodes peuvent être utilisées pour l'extraction des octets. La première est basée sur les fonctions de conversion décimal/binaire et binaire/décimale et la deuxième utilise la fonction Matlab typecast.

Soit $Ka_{j,i}^{o}$ est l' $o$ème octet de $Ka_{j,i}$, avec $o = 1, \cdots, N/8 = 4$

Méthode basée conversion :

La méthode est basée d'abord sur la conversion de la valeur entière $Ka_{j,i}$ en bits, suivie d'un groupement des bits en octets, puis une conversion en décimale :

$$\{Ka_{j,i}\}_b = dec2bin(Ka_{j,i})$$

$\{Ka_{j,i}\}_o = bin2dec\left[\{Ka_{j,i}\}_b((o-1)\times 8+1 : o\times 8)\right]$, avec $o = 1, \cdots, 4$.

Méthode basée sur la fonction Matlab typecast

$$\{Ka_{j,i}\} = Typecast(uint32(Ka_{j,i}), 'uint8')$$

$\{Ka_{j,i}\}$ est un vecteur formé de 4 octets

En Matlab, la deuxième implémentation est pratiquement 57 fois plus rapide que la première (la première méthode semble plus compliquée d'implémentation en Matlab).

L'opération d'addition de clés s'écrit alors :

$$\left.\begin{cases} BA = mod\left[\left(B + Ka_j\right), 2^8\right] & si\ j\ est\ impair \\ BA = B \oplus Ka_j & si\ j\ est\ pair \end{cases}\right\} et\ j = 1, 2, \ldots, r \qquad (3.6)$$

*2.3.1.2 Couche de substitution basée sur la carte chaotique skew tent de valeurs entières finies et inversible*

Nous avons utilisé la carte chaotique skew tent, de valeurs entières finies et inversible, en tant que fonction de substitution non linéaire et non en tant que générateur de séquences chaotiques. Les valeurs du bloc $BA$, résultat de l'opération de mélange de clés varient entre 0 et $Q - 1 = 2^8 - 1$. Pour que la carte skew tent soit inversible, chaque valeur du bloc $BA$ est augmentée de 1 pour former le bloc d'entrée de la carte de substitution skew tent, soit $BA1 = BA + 1$ avec $BA1 \in \{1, \cdots, Q\}$. Puis, une fois l'opération de substitution est terminée, nous retranchons 1 de chaque valeur du bloc $BA1$ pour former le bloc $BS = BA1 - 1$, entrée de la couche de permutation [Masuda et al, 2006]. L'équation de substitution est alors :

$$BS = S_{a_{j,i}}(BA1) = \begin{cases} \left\lceil \frac{Q}{a_{j,i}} \times BA1 \right\rceil & 1 \leq BA1 \leq a_{j,i} \\ \left\lfloor \frac{Q}{Q - a_{j,i}} (Q - BA1) \right\rfloor + 1 & a_{j,i} < BA1 \leq Q \end{cases} \quad (3.7)$$

Avec, $BS \in \{1, \cdots, Q\}$, $a_{j,i} \in \{1, \cdots, Q\}$ $et$ $i = 1, 2, \ldots, na$, où $na$ est le nombre de paramètres de contrôle utilisé dans la fonction de substitution. Pour chaque paramètre de contrôle $a_{j,i}$ $i = 1, 2, \ldots, na$, la fonction de substitution est itérée $rs$ fois, donc au total la fonction de substitution est exécutée $na \times rs$ fois. L'extraction des paramètres $a_{j,i}$ nécessaires, à partir des échantillons (chacun quantifié sur 32 bits) du générateur chaotique utilisé, se fait de la même manière que dans le cas d'addition de clés. Ensuite, à chaque octet extrait on rajoute +1 afin de satisfaire la condition $a_{j,i} \in \{1, \cdots, Q\}$.

Sous forme algorithmique l'opération de substitution réalisée par l'équation ci-dessus se décrit comme suit :

**Entrée** : $BA1$, $Ks_j = [a_{j,1} || a_{j,2} || \ldots || a_{j,na}]$, $j = 1, 2, \ldots, r$ $et$ $1 \leq a_{j,i} \leq Q = 256, rs$
**Sortie** : $BS$
Pour $ina$ allant $de$ 1 $à$ $na$
  Pour $ito$ allant $de$ 1 $à$ $Tbo$
    Pour $itr$ allant $de$ 1 $à$ $rs$
      *Si* $BA1(ito) \leq a(ina)$
        $BA1(ito) = \left\lceil \frac{Q}{a(ina)} \times BA1(ito) \right\rceil$
      *Sinon*
        $BA1(ito) = \left\lfloor \frac{Q}{Q - a(ina)} (Q - BA1(ito)) \right\rfloor + 1$
      Fin *Si*
    *Fin itr*
  *Fin ito*
*Fin ina*
*BS= BA1-1*

Fig. 3.4: Pseudo code de l'opération de substitution

En langage Matlab, l'opération de substitution précédente est réalisée efficacement comme suit :

```
Q = 256;
rs = 4;
for ina = 1:na
    a = Ks_j(ina)
    BA1 = subboxvector(BA1, Q, a, rs);
End
Bs = BA1 − 1;
```

La fonction Matlab $subboxvector$ est décrite sous-dessous :

$$
\begin{aligned}
&function\ S = subboxvector(E, Q, a, rs)\\
&\quad for\ w = 1{:}rs\\
&\quad\ dinf = find\ (E <= a);\\
&\quad\ dsup = find\ (E > a);\\
&\quad\ S(dinf) = ceil((Q * E(dinf))/a);\\
&\quad\ S(dsup) = floor((Q * (Q - E(dsup)))/(Q - a)) + 1;\\
&\quad\ E = S;\\
&end
\end{aligned}
$$

Nous illustrons ci-dessous (voire figure 3.5) l'opération de substitution (sous Matlab) sur un exemple d'un bloc $BA1$ de taille 8 octets et dont les valeurs entières sont de 1 à 8 par ordre successif (ce choix est judicieux pour bien illustrer la substitution). Nous avons pris $rs = 1$, et nous avons utilisé 4 paramètres de contrôle $a$, comme indiqué ci dessous:

$$
\begin{aligned}
&a = [16\ , 31, 129, 5];\\
&Q = 256;\\
&for\ ina = 1{:}na\\
&\qquad BA1 = supboxvector(BA1, 256, a(ina), 1);\\
&end\\
&BS = BA1 - 1
\end{aligned}
$$

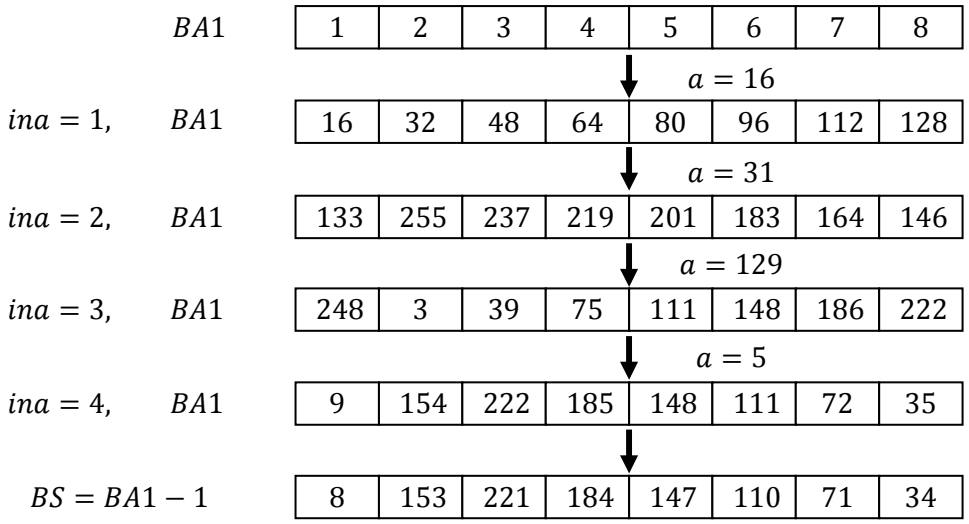



Fig. 3.5: Exemple d'application de l'opération de substitution sur un bloc de 8 octets.

*2.3.1.3 Couche de permutation basée sur la carte chaotique skew tent.*

L'équation de la carte chaotique skew tent est reformulée ici pour être utilisée en tant que fonction de permutation sur les bits du bloc $BS$ de taille $Tb$ (ceci suppose qu'une opération de conversion décimal binaire « dec2bin » a eu lieu avant sur le bloc $BS$).
L'équation devient alors :

$$mn = P_{b_{j,i}}(m) = \begin{cases} \left\lceil \frac{Tb}{b_{j,i}} \times m \right\rceil & 1 \leq m \leq b_{j,i} \\ \left\lfloor \frac{Tb}{Tb - b_{j,i}} (Tb - m) \right\rfloor + 1 & b_{j,i} < m \leq Tb \end{cases} \quad (3.8)$$

où $m$ et $mn$ sont les positions originales et permutées des bits du bloc $BS$, avec :
$mn \in \{1, \cdots, Tb\}$, $b_{j,i} \in \{1, \cdots, Tb\}$ $et$ $i = 1, 2, \ldots, nb$, où $nb$ est le nombre de paramètres de contrôle utilisés dans la fonction de permutation. Pour chaque paramètre de contrôle $b_{j,i}$ $i = 1, 2, \ldots, nb$, la fonction de permutation est itérée $rp$ fois, sur le bloc sous traitement, donc au total la fonction de permutation est exécutée $nb \times rp$ fois. L'extraction des paramètres $b_{j,i}$ nécessaires, à partir des échantillons du générateur chaotique utilisé, se fait de la même manière que dans le cas d'addition de clés.
Sous forme algorithmique, l'opération de permutation réalisée par l'équation ci-dessus se décrit comme suit :

**Entrée** : $Kp_j = [b_{j,1} || b_{j,2} || \ldots || b_{j,nb}]$, $j = 1, 2, \ldots, r$ $et$ $1 \leq b_{j,i} \leq Tb, rp$
**Sortie** : $mn$
$Pour\ itb$ allant $de$ 1 $à$ $Tb$
  $mn(itb) = itb$
    $Pour\ inb$ allant $de$ 1 $à$ $nb$
      $Pour\ itr$ allant $de$ 1 $à$ $rp$
        $Si\ mn(itb) \leq b(inb)$
          $mn(itb) = \left\lceil \frac{Tb}{b(inb)} \times mn(itb) \right\rceil$
        $Sinon$
          $mn(itb) = \left\lfloor \frac{Tb}{Tb - b(inb)} (Tb - mn(itb)) \right\rfloor + 1$
        $Fin\ Si$
      $Fin\ itr$
    $Fin\ inb$
$Fin\ itb$

Fig. 3.6: Pseudo code de l'opération de permutation

En langage Matlab, l'opération de permutation précédente s'effectue comme dans le cas de la substitution d'une manière efficace comme suit :
Etant donnée la taille du bloc $Tb$, par exemple $Tb = 256$;

```
rp = 1;
mn = 1 :Tb
for inb = 1: nb
  b = Kp_j(inb)
  mn = permboxvector(mn, Tb, b, rp);
end
```

La fonction Matlab $permboxvector$ est décrite ci-dessous :

$$
\begin{aligned}
&function\ mn = permboxvector(mn, Tb, b, rp)\\
&\ for\ w = 1{:}rp\\
&\quad dinf = find\ (mn <= b);\\
&\quad dsup = find\ (mn > b);\\
&\quad mn(dinf) = ceil((Tb * mn(dinf))/b);\\
&\quad mn(dsup) = floor((Tb * (Tb - mn(dsup)))/(Tb - b)) + 1;\\
&\ end
\end{aligned}
$$

Nous illustrons ci-dessous (voire figure 3.7) l'opération de permutation (sous Matlab) sur un exemple d'un bloc $BS$ exprimé en bits de taille $Tb = 8$ bits et dont les indices varis par ordre de 1 à 8 (ce choix est judicieux pour bien illustrer la permutation). Nous avons pris $rp = 1$, et nous avons utilisé 4 paramètres de contrôle $b$, comme indiqué ci dessous:

$b = [6\ , 1, 5, 7];$
$Tb = 8;$
$mn = 1{:}Tb$
$for\ inb = 1{:}nb$
$b = Kp_j(inb)$
$mn = permboxvector(mn, Tb, b, 1);$
$end$

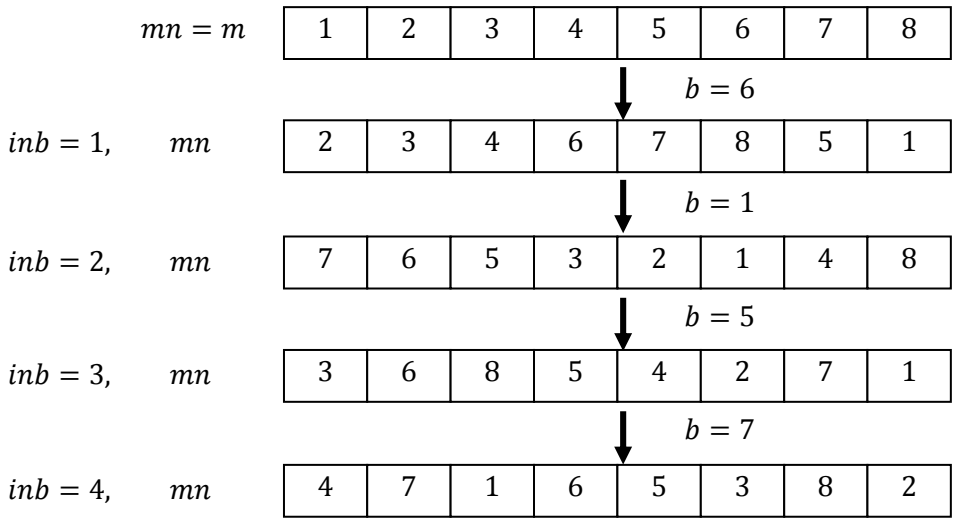



Fig. 3.7: Exemple d'application de l'opération de permutation sur un bloc de 8 bits

*2.3.1.4 Couche de permutation basée sur la carte chaotique 2-D cat.*

Nous donnons ci-dessous l'équation courante de la carte chaotique 2-D cat, utilisée pour l'opération de permutation [Fridrich, 1998], ensuite, nous présentons l'équation modifiée de cette même carte qui permet une implémentation en Matlab très efficace (pratiquement mille fois plus rapide) par rapport à l'implémentation courante. Ce résultat est valable aussi pour toutes les cartes chaotiques 2-D (par exemple la carte chaotique standard utilisée par la suite).

L'équation de permutation utilisant la carte chaotique courante cat, définie par la matrice carrée $A_0$ de taille $M \times M$, est donnée par:

$$\begin{bmatrix} in \\ jn \end{bmatrix} = Mod\left(A_0 \times \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} ri + rj \\ rj \end{bmatrix}, \begin{bmatrix} M \\ M \end{bmatrix}\right) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad (3.9)$$

où $(i, j)$ et $(in, jn)$ sont les positions originales et permutées des éléments de la matrice carrée $BS$ exprimée en bits de taille $M \times M$, et de forme

$$BS = \begin{pmatrix} bs_{1,1} & \cdots & bs_{1,M} \\ \vdots & \ddots & \vdots \\ bs_{M,1} & \cdots & bs_{M,M} \end{pmatrix} \quad (3.10)$$

Le rajout de $+\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ dans l'équation cat, vient du fait que l'opération modulo peut engendrer la valeur (0, 0), ceci pose un problème dans l'implémentation Matlab car l'indice (0, 0) n'existe pas.

La matrice $A_0$ définie par :

$$A_0 = \begin{bmatrix} 1 & u \\ v & 1 + u \times v \end{bmatrix} \quad (3.11)$$

a pour déterminant la valeur 1, et donc la carte cat est bijective, c'est-à-dire, la position de chaque élément permuté est unique. Cependant, la carte cat n'est pas une fonction inversible à cause de l'opération modulo mais elle est réversible.

Nous donnons ci-dessous sous forme algorithmique le calcul standard de la permutation:

La permutation des valeurs des bits de la matrice $BS$ se fait comme suit :

Pour $i$ allant $de$ 1 $à$ $M$

Pour $j$ allant $de$ 1 $à$ $M$

Calcul de la nouvelle position $(in, jn)$ donnée par l'équation ci-dessus (3.11)

Echange (Swap) du contenu de l'élément $(i, j)$ par celui de $(in, jn)$.

Ceci se résume ainsi :

```
Pour i allant de 1 à M
    Pour j allant de 1 à M
        (in, jn) = cat(i, j, u, v, ri, rj)
        Tamp = BS(i, j)
        BS(i, j) = BS(in, jn)
        BS(in, jn) = Tamp
    Fin j
Fin i
```

*2.3.1.5 Equation modifiée de la carte cat*

Afin d'accélérer d'une façon considérable le processus de permutation sur les bits de la matrice $BS$ (# 1000 fois, en Matlab), nous proposons de mettre l'équation de la carte cat sous la forme suivante :

$$\begin{bmatrix} Mln \\ Mcn \end{bmatrix} = Mod\left(A_0 \times \begin{bmatrix} Ml \\ Mc \end{bmatrix} + \begin{bmatrix} rl + rc \\ rc \end{bmatrix}, \begin{bmatrix} M \\ M \end{bmatrix}\right) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad (3.12)$$

où $Ml$, et $Mc$ sont les matrices carrées, respectivement ligne et colonne données par :

$$Ml = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ M & \cdots & M \end{bmatrix}, \qquad Mc = \begin{bmatrix} 1 & \cdots & M \\ \vdots & \ddots & \vdots \\ 1 & \cdots & M \end{bmatrix}$$

Les nouvelles matrices ligne $Mln$ et colonne $Mcn$ permettent de calculer la matrice $ind$ (indices linéaires : valeurs entières comprises entre 1 et $M$) indiquant les positions des bits à permuter de la matrice $BS$. La matrice $ind$ est donnée par la fonction Matlab suivante :

$ind = sub2ind([M\ M], Mln, Mcn)$, soit $ind = (Mln - 1) + (Mcn - 1) \times M$+1.

Enfin, la matrice $BP$ des bits permutés s'obtient par l'équation suivante : $BP = BS(ind)$.

Afin d'illustrer la méthode, nous donnons ci-dessous (voire figure 3.8) un exemple sous Matlab, dans le cas $M = 4,\ u = 3,\ v = 2,\ rl = 2,\ rc = 3,\ rp = 1$, les valeurs des différentes matrices :

$Mln, Mcn$ résultats du processus de permutation des matrices initiales $Ml, Mc,$ puis la matrice $ind$, ensuite, la matrice $BP$, résultat, de la permutation des bits de la matrice $BS$.

Sur la figure 3.9 suivante, nous représentons le temps moyen de calcul (en secondes), pour 1000 clés dynamiques différentes et pour différentes taille $M$ des matrices carrées, de l'opération de permutation par la carte cat usuelle et celle modifiée (ainsi que pour la carte standard et sa version modifiée présentées plus loin dans le 3ème crypto-système).

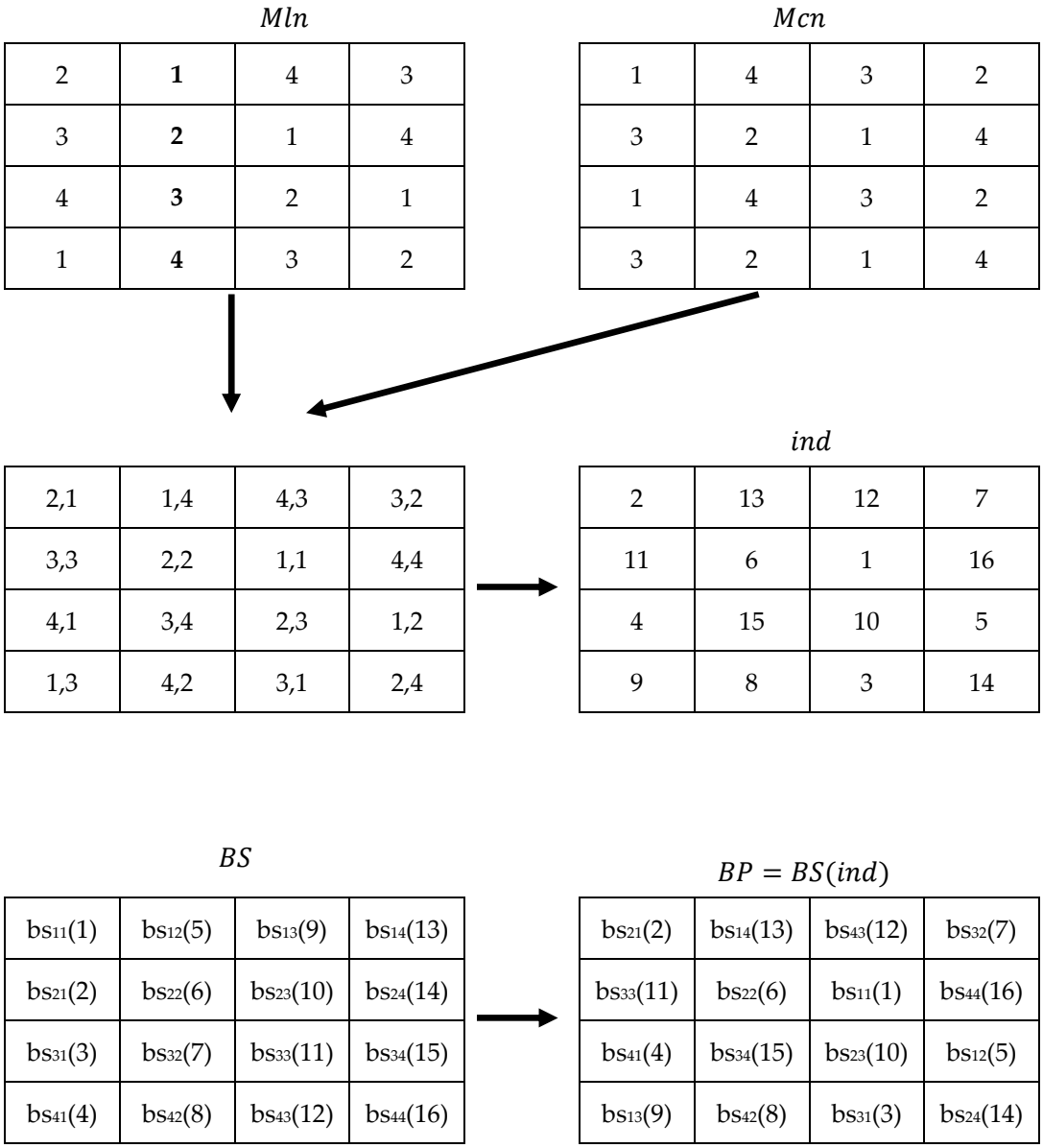



Fig. 3.8: Exemple des valeurs des différentes matrices impliquées dans le processus de permutation.

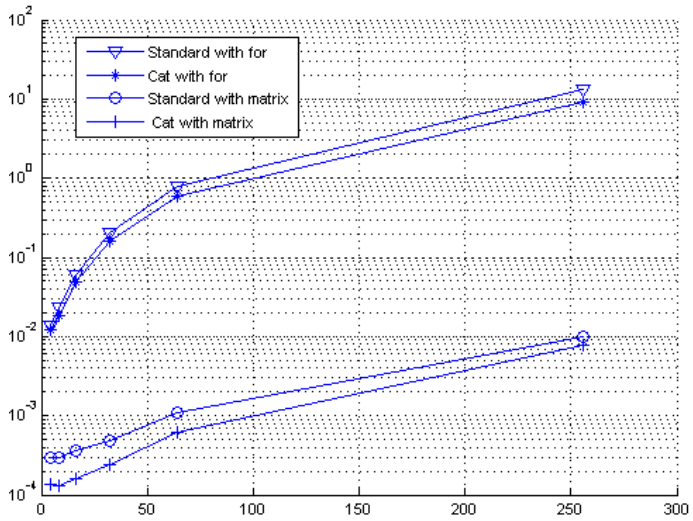



Fig. 3.9: Temps moyen de calcul en secondes, pour 1000 clés différentes, de l'opération de permutation de deux cartes chaotiques cat et standard en fonction de la taille *M*

### 3.3.2 Description du processus de déchiffrement

Dans le processus de déchiffrement nous réalisons les transformations inverses appliquées sur le bloc chiffré, dans l'ordre inverse du chiffrement, afin de retrouver l'information en claire. Nous détaillons ci-dessous ces différentes transformations inverses.
Remarque : L'influence des erreurs de transmission dans le bloc chiffré après transmission sur le processus de déchiffrement dépend, rappelons le, du mode cryptographique (CBC, CFB, OFB, CTR, etc.) utilisé.

#### *2.3.2.1 Couche de permutation inverse basée sur l'équation inverse de la carte chaotique skew tent.*

L'équation de permutation inverse de la carte chaotique skew tente appliquée sur les indices du bloc chiffré de taille $Tb$ (ceci suppose qu'une opération de conversion décimal binaire « dec2bin » a eu lieu avant sur le bloc chiffré) est donnée par [Tang et al, 2010] :

$$imn = {P^{-1}}_{b_{j,i}}(m) = \begin{cases} \zeta 1 & si \;\; \theta(m) = m \;\; et \left(\frac{\zeta 1}{b_{j,i}}\right) > \left(\frac{Tb - \zeta 2}{Tb - b_{j,i}}\right) \\ \zeta 2 & si \;\; \theta(m) = m \;\; et \left(\frac{\zeta 1}{b_{j,i}}\right) \leq \left(\frac{Tb - \zeta 2}{Tb - b_{j,i}}\right) \\ \zeta 3 = \zeta 1 & si \;\; \theta(m) = m + 1 \end{cases} \quad (3.13)$$

où :

$$\zeta 1 = ceil\left[\left(\frac{b_{j,i}}{Tb}\right) \times m\right], \quad \zeta 2 = floor\left[\left(\frac{b_{j,i}}{Tb} - 1\right) \times m + Tb\right]$$

$$\zeta 3 = floor[\left(\frac{b_{j,i}}{Tb}\right) \times m], \;\; \theta(m) = m + \zeta 1 - \zeta 3 + 1$$

Avec : $imn \in \{1, \cdots, Tb\}$ est l'indice inverse de permutation de l'indice de permutation $mn$ du chiffrement, $m \in \{1, \cdots, Tb\}$ est l'indice d'entrée, $b_{j,i} \in \{1, \cdots, Tb\}\ et\ i = nb, nb-1, \cdots, 2, 1$ où, $nb$ est le nombre de paramètres de contrôle utilisés dans la fonction inverse de permutation.

Pour chaque paramètre de contrôle $b_{j,i}\ \ i = nb, nb-1, \cdots, 2, 1,$ l'équation inverse de permutation est itérée $rp$ fois, sur le bloc sous traitement, donc au total pour une itération $j$, l'équation inverse de permutation est exécutée $nb \times rp$ fois.

Sous forme algorithmique, l'opération inverse de permutation réalisée par l'équation ci-dessus se décrit comme suit :

**Entrée** : $Kp_j = [b_{j,1} || b_{j,2} || \ldots || b_{j,nb}],\ j = r, r-1, \ldots, 1\ et\ 1 \leq b_{j,i} \leq Tb, rp$
**Sortie** : $imn$
Pour $itb$ allant $de\ 1\ à\ Tb$
$m(itb) = itb$
Pour $inb$ allant $de\ nb\ à\ 1$
Pour $itr$ allant $de\ 1\ à\ rp$
$\zeta 1 = \left\lceil \left( \frac{b(inb)}{Tb} \right) \times m \right\rceil$
$\zeta 2 = \left\lfloor \left( \frac{b(inb)}{Tb} - 1 \right) \times m + Tb \right\rfloor$
$\zeta 3 = \left\lfloor \left( \frac{b(inb)}{Tb} \right) \times m \right\rfloor$
$\theta(m) = m + \zeta 1 - \zeta 3 + 1$
$Si\ \ \theta(m) = m\ et\ \left( \frac{\zeta 1}{b(inb)} \right) > \left( \frac{Tb - \zeta 2}{Tb - b(inb)} \right)$
$imn(itb) = \zeta 1$
$Si\ non\ \ \theta(m) = m\ et\ \left( \frac{\zeta 1}{b(inb)} \right) \leq \left( \frac{Tb - \zeta 2}{Tb - b(inb)} \right)$
$imn(itb) = \zeta 2$
$Si\ non\ \ \theta(m) = m + 1$
$imn(itb) = \zeta 3$
$Fin\ si$
$Fin\ itr$
$Fin\ inb$
$Fin\ itb$

Fig. 3.10: Pseudo code de l'opération inverse de permutation

En langage Matlab, l'opération inverse de permutation s'effectue d'une manière efficace comme suit :

Etant donnée la taille du bloc$Tb$, par exemple $Tb = 256$;

```
rp = 1;
m = 1 :Tb
for  inb = nb: −1: 1
       b = Kp_j(inb)
      imn = inversepermboxvector(m, Tb, b, rp);
End
```

La fonction Matlab $inversepermboxvector$ est décrite sous-dessous :

```
function imn = inversepermboxvector(m,Tb,b,rp)
  for itr = 1:rp
    sigma1 = floor((b * m)/Tb);
    sigma2 = ceil(((b/Tb) - 1) * m + Tb);
    sigma3 = ceil((b * m)/Tb);
    s = m + sigma1 - sigma3 + 1;
    d1 = find(s == m);
    d2 = find(sigma1(d1)/b > (Tb - sigma2(d1))/(Tb - b));
    imn(d1(d2)) = sigma1(d1(d2));
    d3 = find(sigma1(d1)/b <= (Tb - sigma2(d1))/(Tb - b));
    imn(d1(d3)) = sigma2(d1(d3));
    d4 = find(s~ = m);
    imn(d4) = sigma3(d4) ;
    m = imn;
end
```

Nous illustrons ci-dessous (voire figure 3.11) le calcul de l'indice inverse de permutation $imn$ (sous Matlab) sur un exemple d'un bloc de taille $Tb = 8$ bits et dont les indices de départ $m$ varis par ordre de 1 à 8. Evidemment, on se place dans les mêmes conditions que dans l'exemple du calcul de l'indice $m$ de permutation.

```
b = [6 , 1, 5, 7];
Tb = 8;
imn = 1:Tb
for inb = nb:-1:1
imn = inversepermboxvector(m,Tb,b(inb),1);
end
```

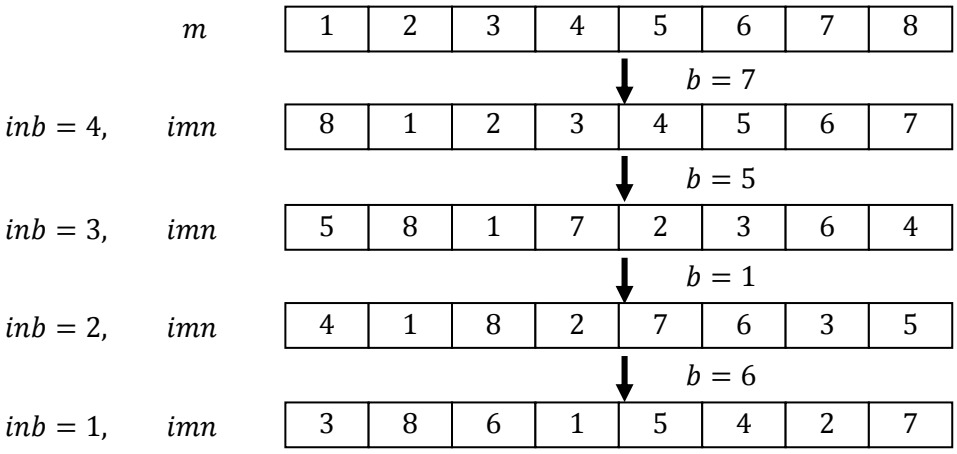



Fig. 3.11: Exemple de calcul de l'indice inverse de permutation sur un bloc de 8 bits

Ci-dessous, dans la figure 3.12 en combinant les résultats de deux exemples précédents, nous montrons l'exactitude des procédures de permutation et de permutations inverse réalisées sous Matlab.

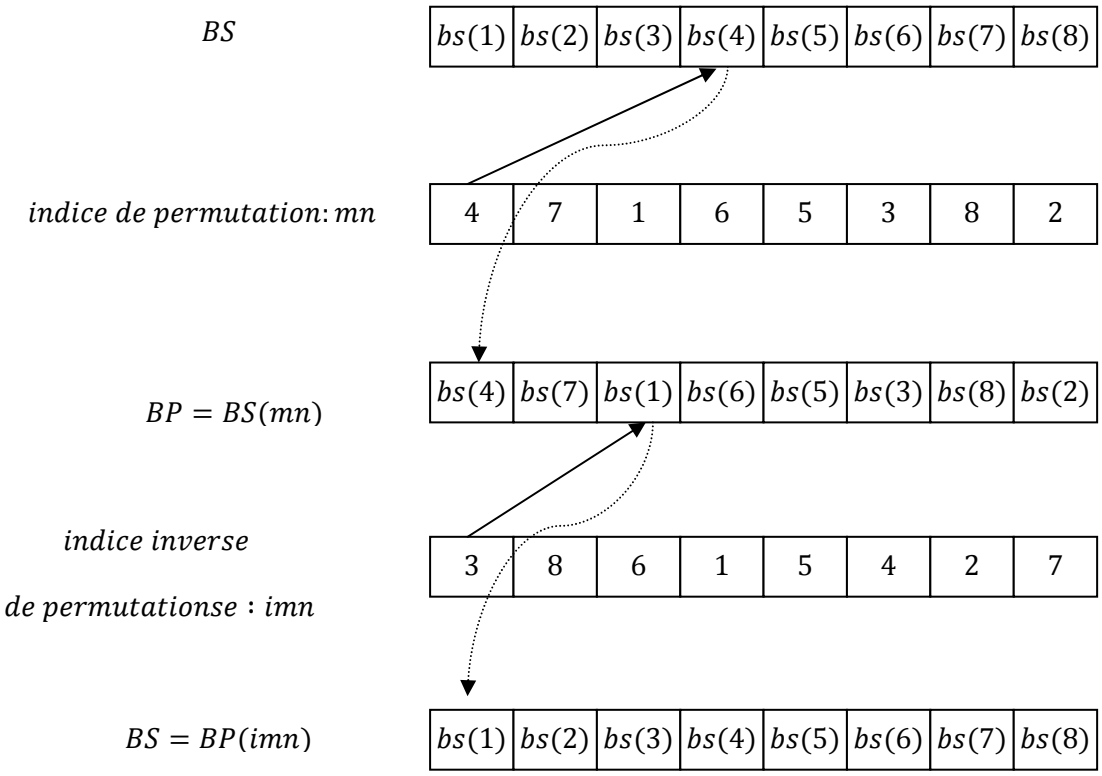



Fig. 3.12: Exemple de réalisation de l'opération de permutation et permutation inverse sur un bloc de 8 bits

*2.3.2.2 Couche de permutation réversible basée sur la carte chaotique 2-D cat usuelle.*

Nous avons déjà signalé que la carte chaotique 2-D cat n'est pas une fonction inversible à cause de l'opération modulo mais elle est retournable. Donc, pour réaliser la permutation inverse par l'équation usuelle de la carte chaotique 2-D cat, il suffit d'utiliser la même équation mais de commencer le calcul des indices $(in, jn)$ à partir des indices $(i, j)$ pour $i$ *allant de* $M$ à 1, et $j$ *allant de* $M$ à 1.

Nous donnons ci-dessous sous forme algorithmique le calcul de la permutation retournable (cas où : $rp = 1$)

La permutation inverse des valeurs des bits de la matrice $BP$ se fait comme suit :

Pour $i$ *allant de* $M$ à 1

Pour $j$ *allant de* $M$ à 1

Calcul de la nouvelle position $(in, jn)$ donnée par l'équation de la carte cat-2D usuelle

Echange (Swap) du contenu de l'élément $(i, j)$ par celui de $(in, jn)$.

Ceci se résume ainsi :

$Pour\ i\ \ allant\ de\ M\ à\ 1$
    $Pour\ j\ allant\ de\ M\ à\ 1$
        $(in, jn) = cat(i, j, u, v, ri, rj)$
        $Tamp = BP(i, j)$
        $BP(i, j) = BP(in, jn)$
        $BP(in, jn) = Tamp$
    $Fin\ i$
$Fin\ j$

Ci-dessous, dans la figure 3.13 nous montrons sur un exemple numérique (sous Matlab) d'abord, le résultat *BP* de l'opération de permutation sur la matrice *BS* (cas où *M=2*), puis le résultat *BS* de la permutation inverse (retournable) sur la matrice *BP*. On voit clairement que, en fin de l'opération de permutation retournable, on retrouve la matrice *BS*.

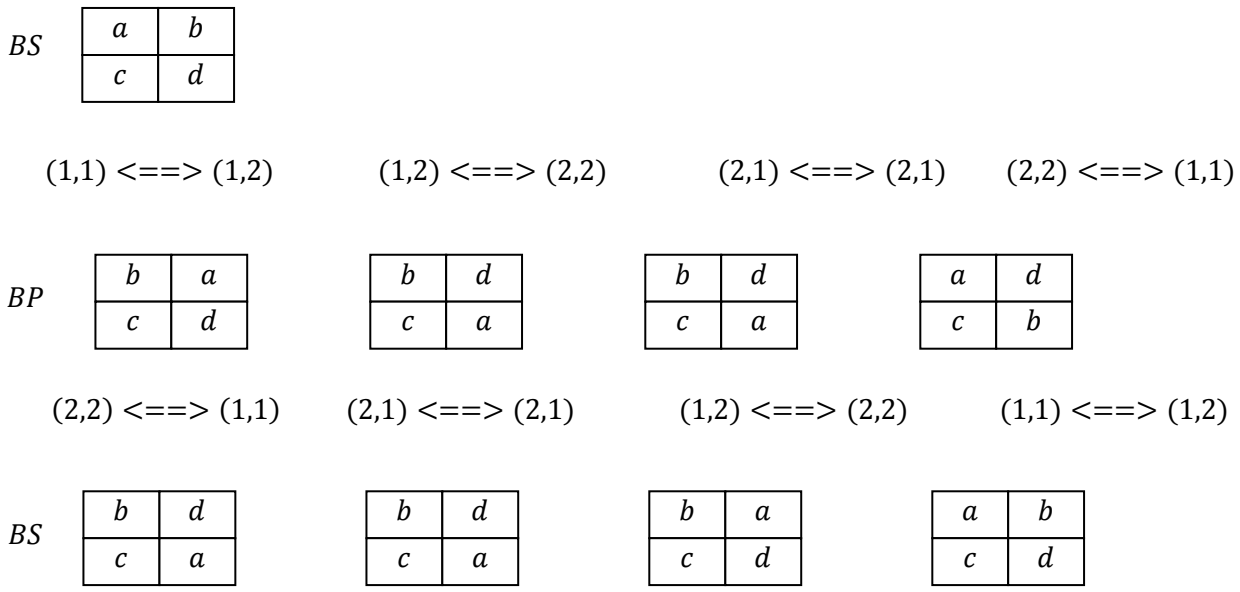



Fig. 3.13: Exemple de réalisation de l'opération de permutation et permutation retournable utilisant la carte chaotique 2-D usuelle sur une matrice carrée de taille *M=2*.

*2.3.2.3 Equation modifiée de la carte cat-2D pour le processus de permutation inverse :*

D'abord, nous calculons la matrice carrée $ind$ de la même manière et dans les mêmes conditions que dans le cas de la permutation directe. Ensuite, nous calculons, à partir de la matrice $ind$, la matrice indice de permutation inverse $indinv$ par : $indinv[ind(i)] = i$ qui nous permet de retrouver la matrice des bits substitués $BS$ par : $BS = BP(indinv)'$ ($X'$ signifie une opération de transposition sur $X$).

Sous Matlab, les différents calculs se réalisent comme suit :

% Conversion de la matrice carrée $ind$ en un vecteur ligne de taille $(1, M\text{^}2)$
$ind = reshape(ind, 1, M\text{^}2);$
% Calcul du vecteur ligne indice de permutation inverse $indinv$
$for\ i\ =\ 1:1:(M\text{^}2)$
$indinv(ind(i))\ =\ i\ ;$
$end$
% Conversion du vecteur ligne indice de permutation
%inverse $indinv$ en une matrice carrée de taille $(M, M)$
$indinv = reshape(indinv, M, M);$
% Calcul de la matrice $BS$ des bits substitués par

$BS\ =\ BP(indinv)';$

end

Afin d'illustrer le processus de permutation et de permutation inverse réalisé par l'équation de la carte cat 2-D modifiée, nous donnons ci-dessous (voire figure 3.14) un exemple sous Matlab, dans le cas $M = 4,\ u = 3,\ v = 2,\ ri = 2,\ rj = 3,\ rp = 1$, les valeurs des différentes matrices :

| $indice\ initiale$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $ind$ | 2 | 11 | 4 | 9 | 13 | 6 | 15 | 8 | 12 | 1 | 10 | 3 | 7 | 16 | 5 | 14 |
| $indinv$ | 10 | 1 | 12 | 3 | 15 | 6 | 13 | 8 | 4 | 11 | 2 | 9 | 5 | 16 | 7 | 14 |

Matrice initiale

| | | | |
|---|---|---|---|
| 1 | 5 | 9 | 13 |
| 2 | 6 | 10 | 14 |
| 3 | 7 | 11 | 15 |
| 4 | 8 | 12 | 16 |

$ind$

$BS$

| | | | |
|---|---|---|---|
| $bs_{11}(1)$ | $bs_{12}(5)$ | $bs_{13}(9)$ | $bs_{14}(13)$ |
| $bs_{21}(2)$ | $bs_{22}(6)$ | $bs_{23}(10)$ | $bs_{24}(14)$ |
| $bs_{31}(3)$ | $bs_{32}(7)$ | $bs_{33}(11)$ | $bs_{34}(15)$ |
| $bs_{41}(4)$ | $bs_{42}(8)$ | $bs_{43}(12)$ | $bs_{44}(16)$ |

$BP = BS(ind)$

| | | | |
|---|---|---|---|
| 2 | 13 | 12 | 7 |
| 11 | 6 | 1 | 16 |
| 4 | 15 | 10 | 5 |
| 9 | 8 | 3 | 14 |

$indinv$

| | | | |
|---|---|---|---|
| $bs_{21}(2)$ | $bs_{14}(13)$ | $bs_{43}(12)$ | $bs_{32}(7)$ |
| $bs_{33}(11)$ | $bs_{22}(6)$ | $bs_{11}(1)$ | $bs_{44}(16)$ |
| $bs_{41}(4)$ | $bs_{34}(15)$ | $bs_{23}(10)$ | $bs_{12}(5)$ |
| $bs_{13}(9)$ | $bs_{42}(8)$ | $bs_{31}(3)$ | $bs_{24}(14)$ |

$BS = BP(indinv)'$

| | | | |
|---|---|---|---|
| 10 | 15 | 4 | 5 |
| 1 | 6 | 11 | 16 |
| 12 | 13 | 2 | 7 |
| 3 | 8 | 9 | 14 |

| | | | |
|---|---|---|---|
| $bs_{11}(1)$ | $bs_{12}(5)$ | $bs_{13}(9)$ | $bs_{14}(13)$ |
| $bs_{21}(2)$ | $bs_{22}(6)$ | $bs_{23}(10)$ | $bs_{24}(14)$ |
| $bs_{31}(3)$ | $bs_{32}(7)$ | $bs_{33}(11)$ | $bs_{34}(15)$ |
| $bs_{41}(4)$ | $bs_{42}(8)$ | $bs_{43}(12)$ | $bs_{44}(16)$ |

Fig. 3.14: Exemple de calcul, par l'équation la carte cat modifiées, des différentes matrices impliquées dans le processus de permutation et de permutation inverse

*2.3.2.4 Couche de substitution inverse basée sur la carte chaotique skew tent*

Avant d'appliquer l'équation de substitution inverse de la carte chaotique skew tente, nous procédons à une opération de conversion binaire décimal «bin2dec» sur le bloc *BS=(imn)* puis nous augmentons de 1 chaque valeur du bloc $BS$, soit $BS1 = BS + 1$, avec $BS1 \in \{1, \cdots, Q\}$, pour former le bloc d'entrée de l'équation de substitution inverse. Puis, une fois l'opération de substitution inverse est terminée, nous retranchons 1 de chaque valeur du bloc résultat, qui n'est autre que le bloc $BA1$, pour former le bloc $BA = BA1 - 1$, entrée de la couche d'addition inverse.

L'équation de substitution inverse est donnée par :

$$BA1 = S_{a_{j,i}}^{-1}(BS1) = \begin{cases} \zeta 1 & si \quad \theta(BS1) = BS1 \quad et \quad \left(\frac{\zeta 1}{a_{j,i}}\right) > \left(\frac{Q - \zeta 2}{Q - a_{j,i}}\right) \\ \zeta 2 & si \quad \theta(BS1) = BS1 \quad et \quad \left(\frac{\zeta 1}{a_{j,i}}\right) \leq \left(\frac{Q - \zeta 2}{Q - a_{j,i}}\right) \\ \zeta 3 = \zeta 1 & si \quad \theta(BS1) = BS1 + 1 \end{cases} \tag{3.14}$$

où :

$$\zeta 1 = ceil\left[\left(\frac{a_{j,i}}{Q}\right) \times BS1\right], \zeta 2 = floor\left[\left(\frac{a_{j,i}}{Q} - 1\right) \times BS1 + Q\right], \ \zeta 3 = floor[\left(\frac{a_{j,i}}{Q}\right) \times BS1]$$
$$\theta(BS1) = BS1 + \zeta 1 - \zeta 3 + 1$$

Et, $a_{j,i} \in \{1, \cdots, Q\}\ et\ i = na, na - 1 \ldots 2,1$ ($na$, rappelons le est le nombre de paramètres de contrôle utilisé dans la fonction de substitution inverse). Aussi pour chaque paramètre de contrôle $a_{j,i}$ $i = na, na - 1, \ldots, 2,1$, l'équation de substitution inverse est itérée $rs$ fois, donc au total, pour une itération $j = r, \cdots, 2,1$ donnée, la fonction de substitution inverse est exécutée $na \times rs$ fois.

Sous forme algorithmique l'opération de substitution inverse réalisée par l'équation ci-dessus se décrit comme suit :

**Entrée** : $BS1$, $Ks_j = \left[a_{j,1}|| \ldots ||a_{j,2}||a_{j,na}\right]$, $j = r, r-1, \cdots, 2,1$ $et$ $1 \leq a_{j,i} \leq Q$, $rs$
**Sortie** : $BA$

Pour $ina$ allant $de$ $na$ $à$ 1
  Pour $ito$ allant $de$ 1 $à$ $Tbo$
    Pour $itr$ allant $de$ 1 $à$ $rs$
      $\zeta 1 = \left\lfloor \left(\frac{a(ina)}{Q}\right) \times BS1(ito) \right\rfloor$
      $\zeta 2 = \left\lfloor \left(\frac{a(ina)}{Q} - 1\right) \times BS1(ito) + Q \right\rfloor$
      $\zeta 3 = \left\lfloor \left(\frac{a(ina)}{Q}\right) \times BS1(ito) \right\rfloor$
      $\theta(BS1(ito)) = BS1(ito) + \zeta 1 - \zeta 3 + 1$
        ***Si*** $\theta(BS1(ito)) = BS1(ito)$ $et$ $\left(\frac{\zeta 1}{a(ina)}\right) > \left(\frac{Q - \zeta 2}{Q - a(ina)}\right)$
          $BS1(ito) = \zeta 1$
        ***Si non*** $\theta(BS1(ito)) = BS1(ito)$ $et$ $\left(\frac{\zeta 1}{a(ina)}\right) \leq \left(\frac{Q - \zeta 2}{Q - b(ina)}\right)$
          $BS1(ito) = \zeta 2$
        ***Si non*** $\theta(BS1(ito)) = BS1(ito) + 1$
          $BS1(ito) = \zeta 3$
        ***Fin Si***
    $Fin\ itr$
  $Fin\ ito$
$Fin\ ina$
$BA = BS1 - 1$

Fig. 3.15: Pseudo code de l'opération de substitution inverse

En langage Matlab, l'opération de substitution inverse précédente est réalisée efficacement comme suit :

```
Q = 256;
rs = 4;
for ina = na:-1:1
a = Ks_j(ina)
BS1 = subboxvectorinverse(BS1, Q, a, rs);
end
BA = BS1 - 1;
```

La fonction Matlab $subboxvectorinverse$ est décrite sous-dessous:

```
function S = inversesubboxvector(E, Q, a, rs)
  for itr = 1:rs
    sigma1 = floor((a * E)/Q);
    sigma2 = ceil(((a/Q) - 1) * E + Q);
    sigma3 = ceil((a * E)/Q);
    res = E + sigma1 - sigma3 + 1;
    d1 = find(res == E);
    d2 = find(sigma1(d1)/a > (Q - sigma2(d1))/(Q - a));
    S(d1(d2)) = sigma1(d1(d2));
    d3 = find(sigma1(d1)/a <= (Q - sigma2(d1))/(Q - a));
    S(d1(d3)) = sigma2(d1(d3));
    d4 = find(res~ = m);
    S(d4) = sigma3(d4) ;
    E = S;
end
```

Nous illustrons ci-dessous (voire figure 3.16) l'opération de substitution inverse (sous Matlab) sur un exemple d'un bloc $BS1$ de taille 8 octets et dont les valeurs entières sont le résultat de l'opération de substitution décrit en chiffrement. Pour cela, on se place dans les mêmes conditions lors de l'opération de substitution, soit :

$rs = 4$
$a = [16\ , 31, 129, 5];$
$Q = 256;$
$for\ ina = na{:}\ {-1}{:}\ 1$
$BS1 = supboxvectorinverse(BS1, 256, a(ina), 1);$
$end$
$BA = BS1 - 1$

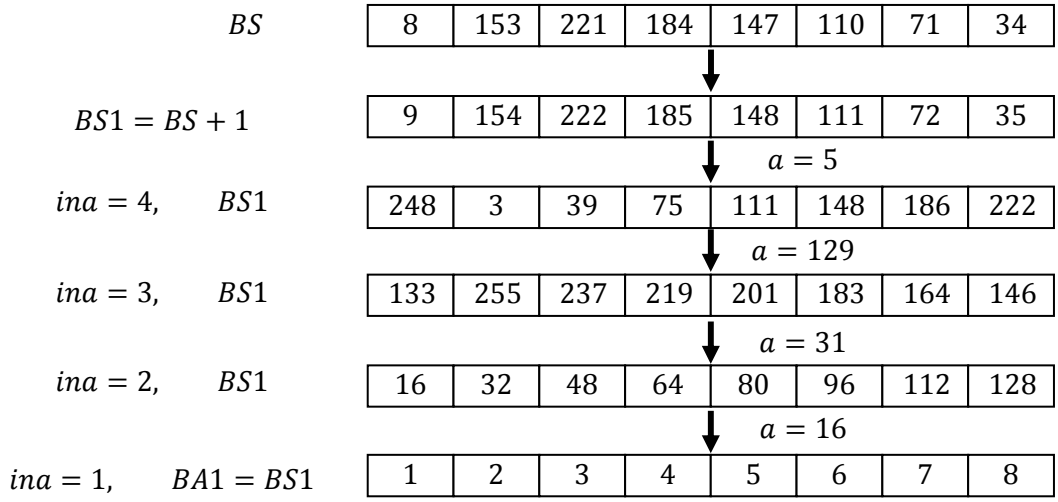



Fig. 3.16: Exemple d'application de l'opération de substitution inverse sur un bloc de 8 octets.

Ci-dessous, (voire figure 3.17) nous montrons l'exactitude des procédures de substitution et de substitution inverse réalisées sous Matlab.

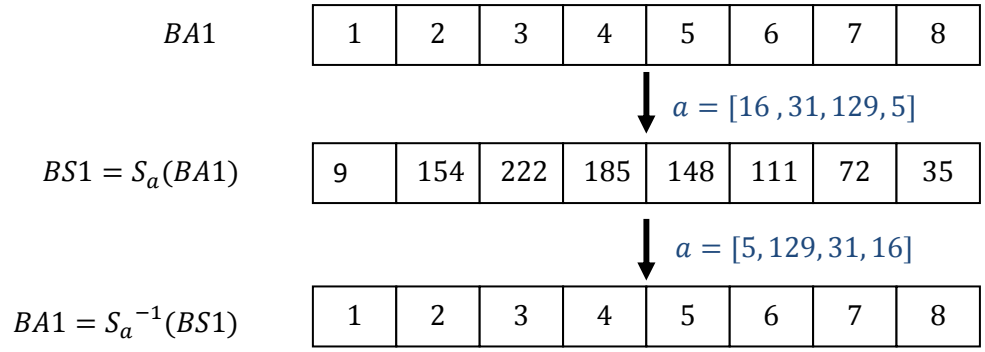



Fig. 3.17: Exemple de réalisation de l'opération de substitution et substitution inverse sur un bloc de 8 octets.

*2.3.2.5 Couche d'addition inverse des clés dynamiques*

Comme nous l'avons déjà signalé, l'opération inverse utilisée dans le processus de déchiffrement pour le mélange des clés dynamiques est donnée par l'équation suivante :

$$\begin{cases} mod[(O-K), 2^8] & si\ it\ est\ impair \\ O \oplus K & si\ it\ est\ pair \end{cases} \quad (3.15)$$

### 3.3.3 Générateur des clés dynamiques

Le générateur chaotique utilisé dans ce premier crypto-système est celui décrit de façon générale dans le chapitre 2, fig. 2.26 , Il dispose de 4 sorties en parallèles $X_i,\ i = 1, \cdots, 4$. A partir de ces 4 sorties, nous calculons le nombre d'itérations nécessaires du générateur pour chiffrer un bloc de taille $Tb$ bits.

A cet effet, définissons d'abord, les itérations nécessaires du générateur pour le régime transitoire et les trois processus impliqués dans le chiffrement/déchiffrement pour chaque itération $j$.

soit :

$tr$ : Le nombre d'itérations transitoires du générateur pour s'assurer de la sensibilité de la clé secrète.

$nita = Tb/(4 \times N)$ : Le nombre d'itérations nécessaires au générateur pour le processus d'addition des clés dynamiques.

$nits$ : Le nombre d'itérations nécessaires au générateur pour le processus de substitution.

$nitp$ : Le nombre d'itérations nécessaires au générateur pour le processus de permutation, dans le cas de l'utilisation de la carte skew tent.

$nitp$ : le nombre d'itérations nécessaires au générateur pour le processus de permutation, dans le cas de l'utilisation de la carte cat (ou standard).

Dans le processus de chiffrement, initialement (pour le premier bloc à traiter) on fait marcher le générateur : $tr + nita + nits + nitp$ itérations, puis arrêt. Les échantillons générés servent alors à avoir les 3 clés dynamiques nécessaires pour commencer le processus d'addition de clés, puis le processus de substitution et enfin le processus de permutation sur le premier bloc sous test. Une fois le dernier processus terminé (celui de la permutation), on fait marcher à nouveau le générateur à partir de son point d'arrêt, durant $nita + nits + nitp$ itérations et on recommence les trois processus comme précédemment sur le bloc déjà traité.

Les trois opérations précédentes sont répétées $r$ fois (à chaque fois le générateur tourne durant $nita + nits + nitp$) pour avoir le premier bloc chiffré. Donc, le nombre d'itérations $nit$ nécessaires au générateur pour chiffrer le premier bloc de taille $Tb$ bits est : $nit1 = tr + r \times nit$ avec : $nit = (nita + nits + nitp)$. Pour les autres blocs de l'image, le fonctionnement du générateur est le même et donc le nombre d'itérations nécessaires pour chiffrer chaque bloc est $r \times nit$.

Dans le processus de déchiffrement, le nombre d'itérations nécessaires au générateur pour déchiffrer le premier bloc de taille $Tb$ bits est aussi $nit1$. La seule différence est qu'au déchiffrement le générateur doit fonctionner durant $nit1$ avant de commencer le processus de déchiffrement du premier bloc, ensuite de marcher $r \times nit$ pour déchiffrer le deuxième bloc, et ainsi de suite.

Les différents paramètres de contrôle sont calculés comme suit :

#### *2.3.3.1 Paramètres d'addition*

Les paramètres nécessaires sont $Ka_{j,i}$ , $i = 1, \cdots, nk$ avec $nk = Tb/N$.

Par exemple pour $Tb = 256$, le nombre de paramètres nécessaires est $nk = 8$, et donc , il faut $nita = Tb/(4 \times N) = 2$ itérations du générateur pour calculer les 8 paramètres comme suit : la première itération permet de calculer : $Ka_{j,1} = X_1$, $Ka_{j,2} = X_2$, $Ka_{j,3} = X_3$, $Ka_{j,4} = X_4$ et la deuxième itération permet de calculer le reste : $Ka_{j,5} = X_1$, $Ka_{j,6} = X_2$, $Ka_{j,7} = X_3$, $Ka_{j,8} = X_4$.

#### *2.3.3.2 Paramètres de substitution*

Pour le processus de substitution, les paramètres nécessaires sont $a_{j,i}$ , $i = 1, \cdots, na$ avec $1 \leq a_{j,i} \leq 2^8$ et $na = 4$ (ce choix est un compromis entre l'efficacité de la substitution et le temps de calcul nécessaire). Une itération du générateur $nits = 1$ est suffisante.

Les quatre paramètres sont calculés comme suit :

$$a_{j,1} = O_1 \oplus O_2 \oplus O_3 \oplus O_4$$
$$a_{j,2} = (O_1 \wedge O_2) \vee [(\neg O_1 \wedge O_3) \oplus O_4]$$
$$a_{j,3} = (O_1 \wedge O_3) \vee (O_2 \wedge \neg O_3) \oplus O_4$$
$$a_{j,4} = (O_2 \wedge O_4) \oplus (O_1 \vee \neg O_3)$$

Avec

$$O_1 = \oplus_{l=1}^{4} T_1(l)$$
$$O_2 = \oplus_{l=1}^{4} T_2(l)$$
$$O_3 = \oplus_{l=1}^{4} T_3(l)$$
$$O_3 = \oplus_{l=1}^{4} T_4(l)$$
$$T_i = Typecast(uint32(X_i(n)), 'uint8'), \; i = 1, 2, 3, 4$$

Où ∧ est l'opération logique ET bit à bit, ¬ est l'opération logique de complément à 1 bit à bit, ∨ est l'opération OU bit à bit et ⊕ est l'opération logique OU exclusif bit à bit (XOR).

La fonction $typecast$ permet d'extraire 4 octets à partir de chaque échantillon $X_i$, donc, $T_i$ est un vecteur composé de 4 octets.

Rappelons-le, chaque paramètre est itéré $rs = 4$ fois, aussi, ce choix s'inscrit dans la remarque précédente et résulte d'une étude expérimentale (voir ci-dessous).

*2.3.3.3 Paramètres de permutation:*

**Cas de la carte chaotique cat :**

Les paramètres dynamiques $Kp_{j,i}, i = 1, \cdots, rp$ nécessaires pour la permutation sont pour chaque $i$ : $\{u,\ v,\ ri,\ rj\}$, avec $0 \leq u, v, ri, rj \leq M - 1 = 2^q - 1$, où $q = log_2(M)$ est le nombre de bits nécessaire pour représenter $u, v, ri, rj$. Ici $Tb = 2^i$, avec $i$ entier pair, et $M = \sqrt{Tb} = 2^{i/2}$.

Pour les différents crypto-systèmes réalisés, nous avons fixé $rp = 1$, ceci est suffisant, d'un point de vue de la sécurité, pour ce processus.

Les différents paramètres, pour une itération $j$, sont calculés comme suit :

$$u_j = mod(((O_1 \wedge O_2) \vee (\neg O_1 \wedge O_3)) \oplus O_4, M)$$
$$v_j = mod((O_1 \oplus O_2 \oplus O_3) \vee O_4), M)$$
$$ri_j = mod((O_1 \wedge O_2) \vee (O_1 \wedge O_3) \vee (O_2 \wedge O_4), M)$$
$$rj_j = mod((O_2 \oplus O_1) \vee (O_3 \wedge \neg O_4), M)$$

Où $Mod$ est l'opération modulo définie par : $Mod\ (a, b) = a - b \times floor\ (a/b)$.

Par exemple pour $Tb = 256$ bits, $M = 16$, alors une itération du générateur $nitp = 1$ est suffisante.

**Cas de la carte chaotique skew tent :**

Les paramètres nécessaires sont $b_{j,i}, i = 1, \cdots, nb$ avec $1 \leq b_{j,i} \leq Tb$ et $nb = 4$ (ce choix est un compromis entre l'efficacité de la permutation et le temps de calcul nécessaire). Une itération du générateur $nitp = 1$ est suffisante.

Les quatre paramètres sont calculés comme suit :

$$b_{j,1} = mod((\neg O_1 \wedge O_2) \vee (\neg O_1 \wedge O_3) \oplus O_4, Tb)$$
$$b_{j,2} = mod(O_2 \oplus (O_3 \wedge \neg O_4), Tb)$$
$$b_{j,3} = mod((\neg O_2 \wedge O_3) \vee (\neg O_2 \wedge O_4) \oplus O_1, Tb)$$
$$b_{j,4} = mod(O_2 \oplus (O_3 \wedge \neg O_4), Tb)$$

Chaque paramètre de contrôle $b_{j,i}$ $i = 1, 2, \ldots, nb$ est itéré sur le bloc sous traitement $rp$ fois.

## 3.4 Deuxième algorithme SPN-2 proposé :

Les résultats obtenus sur le temps de calcul pour le chiffrement/déchiffrement du premier algorithme SPN-1 ont montré que les opérations de conversion décimal vers binaire et binaire vers décimal consomment pratiquement 52 % du temps de calcul global (). Plus précisément, l'opération de conversion décimal vers binaire (Dec2bin) consomme pratiquement 45% du temps de chiffrement/déchiffrement et la conversion binaire vers décimal (Bin2dec) consomme approximativement 7%. Afin d'éviter ces opérations de conversion dans la structure des crypto-systèmes, il suffit d'appliquer la permutation sur les octets et non sur les bits, comme cela est fait dans l'algorithme AES. Cependant, la permutation sur les octets seulement ne permet pas de réaliser l'effet d'avalanche et par conséquent la sécurité du crypto-système devient faible. En effet, dans ce cas, le changement d'un bit dans un octet du texte en clair provoque seulement un changement des bits dans le même octet après plusieurs opérations de substitution/permutation. Pour cela, et afin de réaliser l'effet d'avalanche, le crypto-système doit contenir une couche de diffusion après la couche de substitution. Par ailleurs, l'équation de diffusion peut se réaliser en parallèle, donc la diffusion est plus rapide que celle obtenue par une couche de permutation sur les bits.

Dans l'algorithme AES, la couche de diffusion est réalisée par la technique MDS (Maximal Distance Separable) très couteuse en temps de calcul, puisqu'elle consomme en elle-même pratiquement 94% du temps de calcul global. Pour cela, dans l'algorithme SPN-2, nous proposons l'utilisation de deux candidats (un à la fois) en tant que couche de diffusion qui sont plus rapides que la MDS et capables de fournir un nombre de branches linéaires plus grand que 5, valeur obtenue par la matrice MDS. Ainsi, le nombre $r$ d'itérations nécessaires pour assurer la diffusion (et donc la sécurité) est plus petit.

Plus le nombre de branches est grand, plus les performances de la couche de diffusion sont grandes. Le nombre de branches linéaires s'obtient expérimentalement à partir de la matrice de diffusion utilisée.

Ces deux candidats sont :

- Une matrice binaire, ayant une structure statique et utilisant seulement l'opération logique ou exclusif [Kwon et al, 2003 ], [Aoki et al, 2001 ], [Koo et al, 2003],[Koo et al, 2006]

- Une carte chaotique cat à grande dimension possédant une structure dynamique et utilisant des opérations arithmétiques de multiplication et d'addition modulo [ Xiao et al, 2008].

Dans les deux cas de figures le calcul se fait en parallèle.

Nous notons par SPN-2-1, l'algorithme SPN-2, utilisant la matrice binaire comme couche de diffusion et par SPN-2-2, l'algorithme SPN-2, utilisant la carte chaotique cat à grande dimension comme couche de diffusion.

### 3.4.1 Description du crypto-système proposé

La figure 3.18, présente la structure du crypto-système SPN-2 proposé. Dans cette structure, nous utilisons les mêmes couches d'addition de clé, de substitution et de permutation, à part que la permutation dans ce crypto-système est réalisée sur les octets et non sur les bits comme c'est le cas du crypto-système SPN-1. Ceci afin d'éviter l'opération de conversion binaire vers décimal, Bin2dec, consommatrice de temps de calcul. Ces différents processus ont été déjà détaillés dans le précédent crypto-système.

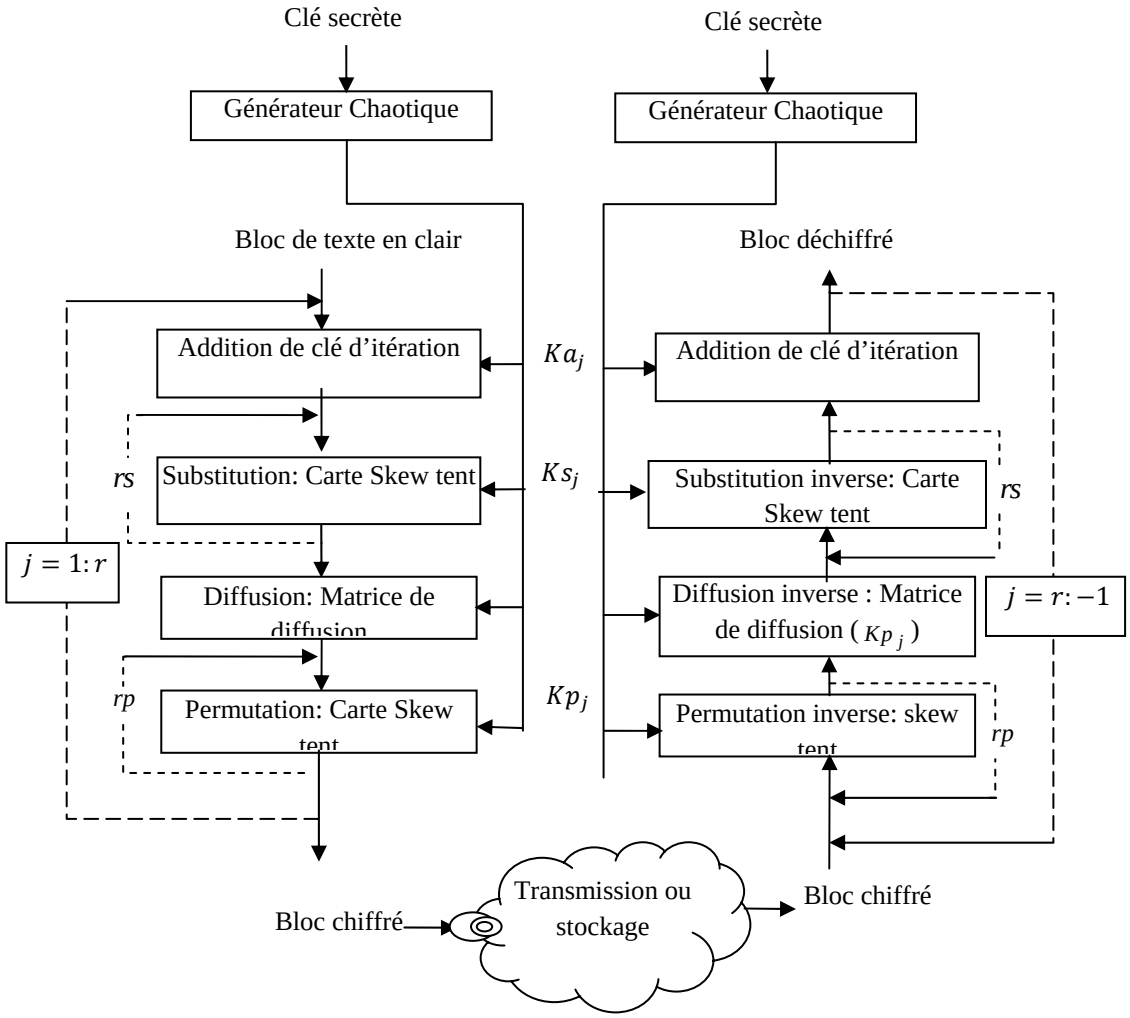



Fig. 3.18: Processus de chiffrement/déchiffrement du SPN-2

La différence majeure du SPN-2, par rapport au SPN-1 est dans la couche de diffusion que contient SPN-2. Pour cela, ce crypto-système présente une forte résistance à la cryptanalyse linéaire et différentielle comme nous le verrons plus loin. Différentes couches

de diffusion sont étudiées et proposées, mais d'abord nous décrivons la structure de la clé de diffusion dynamique $Kd$, dans les différents cas de figures.

### 3.4.2 Structure de la clé de diffusion dynamique $Kd$

La couche de diffusion est construite au moyen d'une matrice de diffusion binaire ou à base d'une carte chaotique cat multidimensionnelle. Dans ce dernier cas, la longueur de la clé de la couche de diffusion est assez large et possède une haute sensibilité. La diffusion est réalisée sur les octets.
Dans le cas de l'utilisation d'une matrice de diffusion binaire statique, il n'y a pas de clé et la présence d'une couche de permutation à la suite de la couche de diffusion est indispensable. Dans le cas d'une couche de diffusion dynamique basée sur une carte cat multidimensionnelle, la présence d'une couche de permutation suite à la couche de diffusion n'est pas primordiale.

Le bloc d'entrée de taille $Tb$ est divisé en sous blocs de taille 8 bits, 16 bits ou 32 bits. Plus la taille du sous bloc est grande, moins grande est la dimension de la carte cat à construire.
La structure de la clé de diffusion est distincte pour chaque méthode étudiée.

#### *2.4.2.1 Première technique I : [Chen et al, 2004], [Xiao et al, 2008]*

La première technique consiste à réaliser une multiplication de matrices carrées de taille $n \times n$ et dont les éléments de base sont ceux de la carte cat de deux dimensions.
Dans le cas d'utilisation d'une structure de base de la carte cat (les équations des 4 structures de base de la carte cat sont données par les relations (3.28) à (3.31)), nous avons alors besoin de $n \times (n-1)/2$ matrices dont chacune utilise un couple $(u_{i,l}, v_{i,l})$, $i, l = 1, \dots, n$ et $i \neq l$. Chaque élément du couple est représenté par un bit ou 2 bits ($pr = 1\ bit\ ou\ 2\ bits$). La taille de la clé de diffusion est alors $|Kd| = n \times (n-1)/2 \times 2 \times pr$ bits. Dans cette première technique, nous prenons *n = 8,* donc *si pr =1,* alors $|Kd| = 56$ bits et pour $pr = 2$, $|Kd| =$ 112 bits.
La génération de $|Kd|$ à partir de quatre sorties du générateur chaotique, se fait en deux étapes. Dans la première étape, on convertit chaque échantillon $X_i$ du générateur en 4 octets placés dans $T_i$, puis on forme par concaténation le vecteur ligne $T$ de taille 16 octets comme suit :
$T_i = Typecast(uint32(X_i(n)), 'uint8'), i = 1, 2, 3, 4$
$T = [T_1\ T_2\ T_3\ T_4]$ ;
Dans la deuxième étape, nous calculons $|Kd|$ à partir du vecteur $T$ comme suit :
pour $|Kd| = 112$ bits (14 octets), on élimine 2 octets d'indices $x$ et $y$ calculés de façon aléatoire comme décrit ci-dessous :
le premier octet à éliminer d'indice $x$ est choisi à partir des 4 LSBs bits du premier octet de $T$, soit :

$$x = Mod(T(1), 16) + 1$$

$$T(x) = [\,];$$

Cette dernière relation a pour effet d'éliminer l'octet d'indice $x$ du vecteur $T$.
Le deuxième octet à éliminer d'indice $y$ est choisi selon la valeur des 4 $LSBs$ bits du dernier octet de $T$, soit :

$$y = Mod(T(15), 15) + 1$$
$$T(y) = [\,];$$

Pour $|Kd| = 56$ bits (7 octets), nous prenons les 7 premiers octets de $T$.

***Remarque*** : dans le cas de construction de la couche de diffusion à partir de plusieurs structures cat au nombre de quatre au maximum, la taille de la clé dynamique de diffusion sera alors celle d'une seule structure multipliée par le nombre des structures utilisées. Cependant, on peut utiliser la même clé pour les différentes structures et dans ce cas la taille de la clé globale ne change pas.

**Affectation des paramètres de la carte cat multidimensionnelle à partir de $Kd$.**

L'affectation des paramètres se fait à partir du vecteur $A$ construit comme suit :

$$Temp = dec2bin(T, 8)$$
$$A = bin2dec\big(reshape(\,Temp, length(Temp)/pr\,, pr)\big)$$

Le vecteur $A$ contient 56 lignes dont les valeurs sont 0 ou 1 si $pr = 1$ ou bien {0, 1, 2, 3} si $pr = 2$. L'affectation des paramètres de la carte cat sera faite par ordre croissant : $u12, v12, u13, v13, , \ldots., u18, v18, u23, v23, \ldots, u28, v28, \ldots, u56, v56, \ldots, u58, v58, u67,$
$v67, u68, v68, u78, v78.$

*2.4.2.2 Deuxième technique [Liu et al, 2008]*

Dans la deuxième technique, deux cas de construction de la carte cat à grande dimension sont réalisés. La taille de la clé dynamique dépend d'un paramètre $m = n - l$ que nous fixons pour un $n$ donné (voir plus loin). Les matrices $Mu$ et $Mv$ sont de tailles $m \times l$ et $l \times m$ respectivement.

- **Deuxième technique : 1er cas de construction II-a :**

Dans le premier cas de construction, les matrices $Mu$ et $Mv$ sont fixées pour les quatre structures (présentées plus loin), et sont données par l'équation suivante :

$$Mu = (O_1 \wedge O_2) \oplus (\neg O_3 \wedge O_4)$$
$$Mv = (\neg O_1 \wedge O_2) \oplus (\neg O_2 \wedge O_3) \oplus (\neg O_3 \wedge O_1)$$

Avec :

$$O_1 = \bigoplus_{l=1}^{2} T_1(l)$$
$$O_2 = \bigoplus_{l=1}^{2} T_2(l)$$
$$O_3 = \bigoplus_{l=1}^{2} T_3(l)$$
$$O_4 = \bigoplus_{l=1}^{2} T_4(l)$$
$$T_i = Typecast(uint32(X_i(n)), 'uint16'), i = 1, 2, 3, 4$$

Pour $n = 4, 8$, nous fixons $m = 2, 4$ respectivement. Ce choix a pour but d'obtenir dans chaque cas, une clé dynamique de taille 32 bits. $Mu$ et $Mv$ sont deux valeurs, dont chacune est quantifiée sur 16 bits, d'après l'équation ci-dessus.
Pour les deux valeurs de $n =$ 8, les valeurs $Mu$ et $Mv$ sont transformées en matrices de taille 4 x 4 bits.

$$Mu = reshape(dec2bin(Mu, 16), 4, 4)$$
$$Mv = reshape(dec2bin(Mv, 16), 4, 4)$$

Ensuite, pour $n = 4$, les matrices $Mu$ et $Mv$ sont transformées en matrices carrées de taille 2 x 2 (puisque dans ce cas on a $m = l = 2$) dont les éléments sont des valeurs entières comprises entre 0 et 15.

$$Mu = reshape\{bin2dec\ \{Mu\}, 2, 2\}$$
$$Mv = reshhape\{bin2dec\ \{Mv\}, 2, 2\}$$

Pour $n = 16, 32$, on fixe $m = 8$, afin d'obtenir une clé dynamique de taille 128 bits. Dans le cas de $n = 16$, on a $m = l = 8$, et les matrices $Mu$ et $Mv$ sont construites comme suit : d'abord, on construit, à partir des quatre sorties du générateur, un vecteur $V$ à deux éléments, dont chacun est quantifié sur 64 bits. Ensuite, le premier élément est utilisé pour créer la matrice binaire $Mu$ de taille 8 X 8 bits, et le second élément est utilisé pour construire la matrice binaire $Mv$ de taille 8 x 8 bits aussi.

$$Mu = reshape(dec2bin(V(1), 64), 8, 8)$$
$$Mv = reshape(dec2bin(V(2), 64), 8, 8)$$

où :

$$V = Typecast(uint32([X_1(n), X_2(n), X_3(n), X_4(n)]), 'uint64'),$$

Dans le cas de $n = 32$, la taille de la matrice $Mu$ est $8 \times 24$ et la taille de la matrice $Mv$ est $24 \times 8$. La construction de ces matrices se fait à partir des matrices précédentes de taille 8 x 8 que nous notons ici par $Mu_8$ et $Mv_8$, comme suit :

$$Mu = zeros(8,24)$$
$$Mv = zeros(24,8)$$
$$Mu(1{:}8,1{:}8) = Mu_8$$
$$Mv(1{:}8,1{:}8) = Mv_8$$

*2.4.2.3 Deuxième technique : 2ème cas de construction II-b :*

Dans le deuxième cas de construction, les matrices $Mu$ et $Mv$ sont spécifiques pour chacune des quatre structures utilisées, leurs expressions sont données par les équations suivantes:

$$Mu_1 = (O_1 \wedge O_2) \oplus (\neg O_3 \wedge O_4)$$
$$Mv_1 = (\neg O_1 \wedge O_2) \oplus (\neg O_2 \wedge O_3) \oplus (\neg O_3 \wedge O_1)$$
$$Mu_2 = O_1 \oplus O_2 \oplus O_3 \oplus O_4$$
$$Mv_2 = (O_1 \wedge O_2) \vee [(\neg O_1 \wedge O_3) \oplus O_4]$$
$$Mu_3 = (O_1 \wedge O_3) \vee (O_2 \wedge \neg O_3) \oplus O_4$$
$$Mv_3 = (O_2 \wedge O_4) \oplus (O_1 \vee \neg O_3)$$
$$Mu_4 = mod(O_1 + O_3, 2^{16}) \oplus O_4$$
$$Mv_4 = mod(O_2 + (O_1 \vee \neg O_3), 2^{16})$$

Où les différents $O_i, i = 1, \cdots, 4$ sont déjà donnés plus haut, dans le premier cas de construction (Deuxième technique : 1er cas de construction II-a).

La création des différentes matrices $Mu_i$ et $Mv_i$ pour $i = 1,2,3,4$ est similaire au premier cas de construction à savoir :

Pour les deux valeurs de $n = 8$, les valeurs $Mu_i$ et $Mv_i$ sont transformées en matrices de taille 4 x 4 bits.

$$Mu_i = reshape(dec2bin(Mu_i, 16), 4,4)$$
$$Mv_i = reshape(dec2bin(Mv_i, 16), 4,4), i = 1,2,3,4$$

Ensuite, pour $n = 4$, les matrices $Mu_i$ et $Mv_i$ sont transformées en matrices carrées de taille 2 x 2 dont les éléments sont des valeurs entières comprises entre 0 et 15.

$$Mu_i = reshape\{bin2dec\{Mu_i\}, 2,2\}$$
$$Mv_i = reshhape\{bin2dec\{Mv_i\}, 2,2\}, i = 1,2,3,4$$

Pour $n = 16, 32$, on fixe $m = 8$, afin d'obtenir une clé dynamique de taille 4 x 128 bits.

Dans le cas de $n = 16$, on a $m = l = 8$, et les matrices $Mu_i$ et $Mv_i, i = 1,2,3,4$ sont construites comme suit : d'abord, on construit, à partir de quatre itérations du générateur chaotique, quatre vecteurs $V_1, V_2, V_3, V_4$ à deux éléments chacun (chaque élément est quantifié sur 64 bits), ensuite, le premier élément de chaque vecteur $V_i, i = 1,2,3,4$ est utilisé pour créer la matrice binaire $Mu_i$ de taille 8 X 8 bits, et le second élément est utilisé pour construire la matrice binaire $Mv_i$ de taille 8 x 8 bits.

$$Mu_i = reshape(dec2bin(V_i(1), 64), 8, 8)$$
$$Mv_i = reshape(dec2bin(V_i(2), 64), 8, 8), \; i = 1,2,3,4$$

où :

$$V_i = Typecast(uint32([X_1(n+i-1), X_2(n+i-1), X_3(n+i-1), X_4(n+i-1)]), 'uint64')$$
$$i = 1,2,3,4$$

Dans le cas de $n = 32$, la taille de la matrice $Mu_i$ est $8 \times 24$ et la taille de la matrice $Mv_i$ est $24 \times 8$. La construction de ces matrices se fait à partir des matrices précédentes de taille 8 x 8 que nous notons ici par $Mu_{i,8}$ et $Mv_{i,8}$, comme suit :

$$Mu_i = zeros(8, 24)$$
$$Mv_i = zeros(24, 8)$$
$$Mu_i(1{:}8, 1{:}8) = Mu_{i,8}$$
$$Mv_i(1{:}8, 1{:}8) = Mv_{i,8}$$

***Remarque*** :

Dans le cas de la deuxième technique : 2ème cas de construction II-b, la taille de la clé dynamique $|Kd_j|$ dépend aussi de la taille $Tb$ et de la taille $Tsb$ (des sous blocs) choisi, avec :

$Tb = n \times Tsb$. Dans notre étude, nous avons fixé $Tsb = 8$ bits, mais nous pouvons aussi travailler avec $Tsb = 32$ bits.

En résumé :

Sachant que la taille des clés dynamiques $Ka_j$, $Ks_j$, et $Kp_j$ d'une ronde $j$ est $Tb$, $na \times 8$ et $nb \times log2(Tb/8)$ bits respectivement, alors, la taille totale de la clé dynamique d'itération est la somme de diverses clés dynamiques, c'est-à-dire $|Ka_j| + |Ks_j| + |Kd_j| + |Kp_j|$.

Dans le tableau suivant, nous donnons la taille des différentes clés et la taille totale en bits.

Tableau 3.1. Taille dynamique de chaque couche et taille totale pour les diverses versions du crypto-système SPN-2, pour *Tb* = 128 bits, et *Tb* = 256 bits, *na* = *nb* = 4

| Méthode | $Tb$ | $\|Ka_j\|$ | $\|Ks_j\|$ | $\|Kd_j\|$ | $\|Kp_j\|$ | totale |
|---|---|---|---|---|---|---|
| I | 128 | 128 | 32 | 0 | 16 | 176 |
| I | 256 | 256 | 32 | 0 | 20 | 308 |
| II-a | 128 | 128 | 32 | {56,112} | 16 | {232,288} |
| II-a | 256 | 256 | 32 | {56,112} | 20 | {364, 420} |
| II-b | 128 | 128 | 32 | {32,128} | 16 | {208, 304} |
| II-b | 256 | 256 | 32 | {32,128} | 20 | {340, 436} |

Par la suite, dans le cas II-b, la construction de la matrice de diffusion est faite pour $Tsb = 8$ bits. La généralisation au cas où $Tsb = 32$ bits est immédiate.

### 3.4.3 Couche de diffusion

Nous présentons ci-dessous les deux méthodes permettant de construire une couche de diffusion ainsi que leurs performances comparatives, et nous pointons notre apport dans chaque cas de figure.

La première méthode consiste à construire une matrice de diffusion binaire statique utilisant une transformation linéaire logique basée sur l'opérateur ou exclusif.

La deuxième méthode s'appuie sur la carte cat 2-D, pour construire des matrices de diffusion dynamique multidimensionnelles. Cette méthode a un pouvoir de diffusion plus grand que la première méthode mais elle est plus lente. En effet, dans ce cas, nous avons besoin de calculer l'inverse de la matrice de diffusion, ce qui n'est pas le cas de la première méthode.

Dans la deuxième méthode, deux approches sont présentées : la première approche s'appuie sur la multiplication des cartes cat [ Chen et al, 2004], et la deuxième approche, plus générale, utilise une transformation linéaire [Liu et al, 2008 ].

### 3.4.4 Couche de diffusion statique à base d'une matrice binaire :

Le nombre de branches de la couche de diffusion produit par une matrice statique MDS [**AES, 1997**] de l'algorithme AES est 5. Afin d'améliorer les performances en termes de degré de sécurité et de temps de calcul de la couche de diffusion de l'algorithme standard AES, Koo [**Koo et al, 2003**] a proposé une méthode de construction statique de matrice binaire 16×16 dont le nombre de branches résultant est 8. Puis, il a étendu la méthode précédente de construction pour une matrice binaire de 32 × 32 produisant un nombre de branches égale à 10 [**Koo et al, 2006** ]. Cependant, la matrice de diffusion 32 x 32 de Koo ne vérifie pas la relation $D^{-1} = D$ et donc son implémentation n'est pas possible.

A ce sujet, nous proposons une matrice de diffusion 32 x 32 qui vérifie la relation $D^{-1} = D$ mais dont le nombre de branches est 8 (identique à celui d'une matrice binaire 16 x 16). Cependant, nous ne donnons pas sa composition car elle fait partie d'un Brevet en cours de rédaction.

La résistance contre l'attaque différentielle et l'attaque linéaire dépend respectivement de la probabilité d'approximation différentielle ($Dp$) et de la probabilité d'approximation linéaire ($Lp$) (résultant des performances de la couche de substitution), du nombre de branches de la couche de diffusion $bd$ (résultant des performances de la couche de diffusion) et du nombre d'itérations $r$.

La complexité d'attaque pour $r$ itérations SPN est inversement proportionnelle aux probabilités maximales suivantes :

$$P_D \leq Dp^{bd \times r/2} \tag{3.16}$$

$$P_L \leq Lp^{bd \times r/2} \tag{3.17}$$

Pour les crypto-systèmes dont les couches de substitution, de diffusion et de permutation sont indépendantes de la clé secrète $K$, tels que : l'AES, le RC6, CAMELIA, ARIA, nous pouvons déterminer le nombre minimal d'itérations nécessaires $r_{min}$ pour assurer la sécurité exigée. En effet, dans le cas de l'AES par exemple, nous avons $Dp = Lp = 2^{-6}$ , $bd = 5$, et $K = 128, ou\ 196, ou\ 256$ (tailles utilisées pour la clé secrète), alors :

$$P_L = P_D \leq 2^{-6 \times 5 \times r/2} \tag{3.18}$$

La valeur $r_{min}$ est telle que :

$$^1/_{P_D} \geq 2^K, \quad soit : \ 2^{6 \times 5 \times r/2} \geq 2^K \Rightarrow r_{min} = ceil\ (^K/_{15})$$

Soit :

$$\begin{cases} r_{min} = 10, & pour\ K = 128 \\ r_{min} = 12, & pour\ K = 196 \\ r_{min} = 14, & pour\ K = 256 \end{cases}$$

Pour notre crypto-système, la complexité d'attaque est plus grande ( $2^{240}$) que celle de l'AES ( $2^{150}$). Dans ce calcul, nous avons pris les mêmes valeurs des paramètres suivants ($r = 10, DP = 2^{-6}$) que l'AES, à part le nombre de branches qui est 8 au lieu de 5.

#### *2.4.4.1 Matrice de diffusion binaire : préliminaires*

La transformation linéaire réalisée par une matrice binaire de diffusion $D$ est définie par :

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \cdots & \ddots & \vdots \\ 1 & 1 & 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \odot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 = x_1 \oplus x_3 \\ y_2 = x_1 \oplus x_4 \oplus 0 \oplus \cdots \oplus 0 \oplus x_n \\ \vdots \\ \vdots \\ y_n = x_1 \oplus x_2 \oplus x_4 \end{bmatrix} \tag{3.19}$$

$(x_1, x_2, \cdots, x_n)^t$ est l'entrée de la couche de diffusion (sortie de la couche de substitution) dont chaque élément $x_i$ est représenté sur un octet, et $(y_1, y_2, \cdots, y_n)^t$ est la sortie de la couche de diffusion (résultat de la transformation linéaire).

Il est souhaitable de construire des matrices de diffusion binaires tel que : $D = D^{-1}$, car ceci facilite l'opération d'implémentation dans le déchiffrement. A cet effet, nous construisons de telles matrices de diffusion binaires par concaténation de sous matrices vérifiant la relation précédente à savoir $A = A^{-1}$. Par exemple, la sous matrice suivante de taille 4x4 vérifie la relation précédente [Koo et al, 2003]:

$$A = A^{-1} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Le nombre de branche linéaires de cette matrice est 4.

Il est important de bâtir une technique de construction de matrice binaire de diffusion multidimensionnelle, de sorte que le nombre de branches linéaires soit plus grand que cinq.

Kanda et al, [Kanda et al, 1999], ont proposé une matrice de diffusion binaire 8x8, dont le nombre maximum de branches est 5 (d'après les simulations numériques) qui est identique au nombre de branches de la matrice de diffusion MDS utilisée par l'algorithme AES.

Dans Camellia [Aoki et al, 2001], la matrice de diffusion binaire utilisée est de taille 8 x 8, et le nombre de branches obtenu est 5.

Le traitement au niveau de la matrice de diffusion binaire se fait en octets, alors la taille de la matrice dépend de la taille du bloc sous traitement. Pour des tailles de bloc de 128 et 256 par exemple, nous avons besoin de construire de matrices de diffusion binaire de taille 16*16 et 32x32 respectivement.

[Koo et al, 2006] a proposé une technique de construction d'une matrice de diffusion binaire de taille 16x16 dont le nombre de branches est 8, puis il a étendu la méthode pour la construction d'une matrice de diffusion binaire de taille 32x32 ayant un nombre de branches égale à 10, mais la matrice en question n'est pas inversible.

Dans la suite, nous donnons la forme des matrices de diffusion binaire 4 x 4, et 8 x8 (celle de Camellia) et les équations de diffusion correspondantes. Puis nous exposons la technique de Koo pour la construction d'une matrice binaire dynamique 16 × 16 dont le nombre de branches est 8, et nous montrons comment obtenir l'équation réduite de diffusion correspondante. Ensuite, nous proposons d'autres formes de matrices de base pour la construction d'une matrice binaire 32 x 32 inversible mais ayant un nombre de branche égale à 8.

Les matrices de diffusion 4 x 4 et 8 x 8 sont utilisées dans les crypto-systèmes basés sur le schéma de Feistel, développés plus loin, et les matrices de diffusion 16 x 16 et 32 x 32 sont utilisées dans le crypto-système SPN-2 proposé.

#### *2.4.4.2 Matrice de diffusion utilisée 4 x 4*

La matrice de diffusion $D$ utilisée est donnée par l'équation ci-dessous, elle vérifie la relation $D = D^{-1}$, et le nombre de branches obtenu est 4.

L'opération de diffusion est définie comme suit :

$$\begin{bmatrix} BS1d \\ BS2d \\ BS3d \\ BS4d \end{bmatrix} = D \odot \begin{bmatrix} BS1 \\ BS2 \\ BS3 \\ BS4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \odot \begin{bmatrix} BS1 \\ BS2 \\ BS3 \\ BS4 \end{bmatrix} \quad (3.20)$$

où $BSi$ et $BSid$, $i = 1, 2, 3, 4$, représentent les $i$èmes octets du bloc avant et après la diffusion.

Dans l'équation ci-dessus, la taille de chaque sous bloc d'entrée est de 32 bits, soit 4 octets.

L'équation de diffusion est implémentée comme suit :

$$BS1d = BS1 \oplus BS2 \oplus BS3$$
$$BS2d = BS1 \oplus BS2 \oplus BS4$$
$$BS3d = BS1 \oplus BS3 \oplus BS4$$
$$BS4d = BS2 \oplus BS3 \oplus BS4$$

L'opération de diffusion inverse est ($D^{-1} = D$) :

$$\begin{bmatrix} BS1 \\ BS2 \\ BS3 \\ BS4 \end{bmatrix} = D^{-1} \odot \begin{bmatrix} BS1d \\ BS2d \\ BS3d \\ BS4d \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \odot \begin{bmatrix} BS1d \\ BS2d \\ BS3d \\ BS4d \end{bmatrix} \quad (3.21)$$

L'équation de diffusion inverse est :

$$BS1 = BS1d \oplus BS2d \oplus BS3d$$
$$BS2 = BS1d \oplus BS2d \oplus BS4d$$
$$BS3 = BS1d \oplus BS3d \oplus BS4d$$
$$BS4 = BS2d \oplus BS3d \oplus BS4d$$

#### *2.4.4.3 Matrice de diffusion utilisée 8 x 8 (Camellia)*

La matrice de diffusion $D$ de Camellia est donnée par l'équation ci-dessous, et le nombre de branches obtenu par cette matrice est maximal et égale à 5 [Aoki et al, 2001 ].

L'opération de diffusion est définie comme suit :

$$\begin{bmatrix} BS1d \\ BS2d \\ BS3d \\ BS4d \\ BS5d \\ BS6d \\ BS7d \\ BS8d \end{bmatrix} = D \odot \begin{bmatrix} BS1 \\ BS2 \\ BS3 \\ BS4 \\ BS5 \\ BS6 \\ BS7 \\ BS8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \odot \begin{bmatrix} BS1 \\ BS2 \\ BS3 \\ BS4 \\ BS5 \\ BS6 \\ BS7 \\ BS8 \end{bmatrix} \quad (3.22)$$

La taille des sous blocs est 64 bits, soit 8 octets.

L'équation de diffusion est alors:

$$BS1d{=}BS1 \oplus BS3 \oplus BS4 \oplus BS6 \oplus BS7 \oplus BS8$$
$$BS2d{=}BS1 \oplus BS2 \oplus BS4 \oplus BS5 \oplus BS7 \oplus BS8$$
$$BS3d{=}BS1 \oplus BS2 \oplus BS3 \oplus BS5 \oplus BS6 \oplus BS8$$
$$BS4d{=}BS2 \oplus BS3 \oplus BS4 \oplus BS5 \oplus BS6 \oplus BS7$$
$$BS5d{=}BS1 \oplus BS2 \oplus BS6 \oplus BS7 \oplus BS8$$
$$BS6d{=}BS2 \oplus BS3 \oplus BS5 \oplus BS7 \oplus BS8$$
$$BS7d{=}BS3 \oplus BS4 \oplus BS5 \oplus BS6 \oplus BS8$$
$$BS8d{=}BS1 \oplus BS4 \oplus BS5 \oplus BS6 \oplus BS7$$

L'opération de diffusion inverse est :

$$\begin{bmatrix} BS1 \\ BS2 \\ BS3 \\ BS4 \\ BS5 \\ BS6 \\ BS7 \\ BS8 \end{bmatrix} = D^{-1} \odot \begin{bmatrix} BS1d \\ BS2d \\ BS3d \\ BS4d \\ BS5d \\ BS6d \\ BS7d \\ BS8d \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \odot \begin{bmatrix} BS1d \\ BS2d \\ BS3d \\ BS4d \\ BS5d \\ BS6d \\ BS7d \\ BS8d \end{bmatrix} \quad (3.23)$$

Et l'équation de diffusion inverse est:

$$BS1{=}BS2d \oplus BS3d \oplus BS4d \oplus BS6d \oplus BS7d \oplus BS8d$$
$$BS2{=}BS1d \oplus BS3d \oplus BS4d \oplus BS5d \oplus BS7 \oplus BS8\text{d}$$
$$BS3{=}BS1d \oplus BS2d \oplus BS4d \oplus BS5d \oplus BS6d \oplus BS8d$$
$$BS4{=}BS1d \oplus BS2d \oplus BS3d \oplus BS5d \oplus BS6d \oplus BS7d$$
$$BS5{=}BS1d \oplus BS2d \oplus BS5d \oplus BS7d \oplus BS8d$$
$$BS6{=}BS2d \oplus BS3d \oplus BS5d \oplus BS6d \oplus BS8d$$
$$BS7{=}BS3d \oplus BS4d \oplus BS5d \oplus BS6d \oplus BS7d$$
$$BS8{=}BS1d \oplus BS4d \oplus BS6d \oplus BS7d \oplus BS8d$$

#### *2.4.4.4 Construction de la matrice de diffusion binaire 16 x 16*

La construction de la matrice de diffusion binaire 16 x 16 est basée sur la définition de deux types de matrices $L$ et $M$ de même taille 16 x 16. Le nombre de branches obtenu par cette matrice est 8 et elle est donnée par l'équation suivante [Koo et al, 2003] :

$$D = Mod\ (L_l \times M_m \times L_l^t, 2) \quad (3.24)$$

Où $L_l^t = L_l$.

Les matrices $L$ et $M$ sont définies comme suit.

$$L_l = \begin{bmatrix} B_{1,1} & B_{1,2} & B_{1,3} & B_{1,4} \\ B_{2,1} & B_{2,2} & B_{2,3} & B_{2,4} \\ B_{3,1} & B_{3,2} & B_{3,3} & B_{3,4} \\ B_{4,1} & B_{4,2} & B_{4,3} & B_{4,4} \end{bmatrix}, \text{ où } B_{i,j} = \begin{cases} I_{4\times4}\ si\ h_{i,j} = 1, \ \ h_{i,j} \in H_l \\ O_{4\times4}\ si\ h_{i,j} = 0, \end{cases}$$

Avec :

$$H_l = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} = H_l^t, \quad I_{4\times4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad O_{4\times4} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

D'où la forme de la matrice $L_l$ :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

La matrice $M_m$ est une matrice composée de sous matrices sur sa diagonale et zéro ailleurs :

$$M_m = \begin{bmatrix} H_{m1} & O_{4\times4} & O_{4\times4} & O_{4\times4} \\ O_{4\times4} & H_{m2} & O_{4\times4} & O_{4\times4} \\ O_{4\times4} & O_{4\times4} & H_{m3} & O_{4\times4} \\ O_{4\times4} & O_{4\times4} & O_{4\times4} & Hm_4 \end{bmatrix} \tag{3.25}$$

Avec :

$$H_{m_1} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \; H_{m_2} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}, H_{m_3} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}, H_{m_4} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

La matrice $M_m$ est alors donnée par :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

D'où la forme de la matrice $D = Mod(L_l \times M_m \times L_l^t, 2) = D^{-1}$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

L'opération de diffusion est alors : $[BSid] = D \odot [BSi], i = 1, \cdots 16$

Pour alléger la notation, nous posons $y_i = BSid,\ et\ x_i = BSi,$ l'équation de diffusion s'écrit alors :

$y_1 = x_4 \oplus x_5 \oplus x_7 \oplus x_9 \oplus x_{10} \oplus x_{14} \oplus x_{15}$ $\quad$ $y_9 = x_1 \oplus x_2 \oplus x_5 \oplus x_8 \oplus x_{11} \oplus x_{14} \oplus x_{16}$

$y_2 = x_3 \oplus x_6 \oplus x_8 \oplus x_9 \oplus x_{10} \oplus x_{13} \oplus x_{16}$ $\quad$ $y_{10} = x_1 \oplus x_2 \oplus x_6 \oplus x_7 \oplus x_{12} \oplus x_{13} \oplus x_{15}$

$y_3 = x_2 \oplus x_5 \oplus x_7 \oplus x_{11} \oplus x_{12} \oplus x_{13} \oplus x_{16}$ $\quad$ $y_{11} = x_3 \oplus x_4 \oplus x_6 \oplus x_7 \oplus x_9 \oplus x_{14} \oplus x_{16}$

$y_4 = x_1 \oplus x_6 \oplus x_8 \oplus x_{11} \oplus x_{12} \oplus x_{14} \oplus x_{15}$ $\quad$ $y_{12} = x_3 \oplus x_4 \oplus x_5 \oplus x_8 \oplus x_{10} \oplus x_{13} \oplus x_{15}$

$y_5 = x_1 \oplus x_3 \oplus x_6 \oplus x_9 \oplus x_{12} \oplus x_{15} \oplus x_{16}$ $\quad$ $y_{13} = x_2 \oplus x_3 \oplus x_7 \oplus x_8 \oplus x_{10} \oplus x_{12} \oplus x_{13}$

$y_6 = x_2 \oplus x_4 \oplus x_5 \oplus x_{10} \oplus x_{11} \oplus x_{15} \oplus x_{16}$ $\quad$ $y_{14} = x_1 \oplus x_4 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{11} \oplus x_{14}$

$y_7 = x_1 \oplus x_3 \oplus x_8 \oplus x_{10} \oplus x_{11} \oplus x_{13} \oplus x_{14}$ $\quad$ $y_{15} = x_1 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_{10} \oplus x_{12} \oplus x_{15}$

$y_8 = x_2 \oplus x_4 \oplus x_7 \oplus x_9 \oplus x_{12} \oplus x_{13} \oplus x_{14}$ $\quad$ $y_{16} = x_2 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_9 \oplus x_{11} \oplus x_{16}$

Il utilise 96 XOR.

Nous avons simplifié l'implémentation du processus de diffusion en réduisant le nombre de XOR utilisé à 56. En effet, dans les différentes expressions $y_i, i = 1, \cdots, 16$, nous avons cherché les termes en commun qui contiennent chacun un nombre d'éléments maximum. Nous avons trouvé 8 termes dont chacun est formé de 4 éléments comme suit:

$t1 = x_2 \oplus x_7 \oplus x_{12} \oplus x_{13}$ ; $t2 = x_3 \oplus x_8 \oplus x_{10} \oplus x_{13}$ ; $t3 = x_3 \oplus x_6 \oplus x_9 \oplus x_{16}$ ;

$t4 = x_2 \oplus x_5 \oplus x_{11} \oplus x_{16}$; $t5 = x_1 \oplus x_6 \oplus x_{12} \oplus x_{15}$ ; $t6 = x_4 \oplus x_5 \oplus x_{10} \oplus x_{15}$ ;

$t7 = x_4 \oplus x_7 \oplus x_9 \oplus x_{14}$; $t8 = x_1 \oplus x_8 \oplus x_{11} \oplus x_{14}$

L'équation de diffusion se simplifie et est donnée par :

$y_1 = x_4 \oplus t6 \oplus t7$ $\quad$ $y_9 = x_{11} \oplus t4 \oplus t8$

$y_2 = x_3 \oplus t2 \oplus t3$ $\quad$ $y_{10} = x_{12} \oplus t1 \oplus t5$

$y_3 = x_2 \oplus t1 \oplus t4$ $\quad$ $y_{11} = x_9 \oplus t3 \oplus t7$

$y_4 = x_1 \oplus t5 \oplus t8$ $\quad$ $y_{12} = x_{10} \oplus t2 \oplus t6$

$y_5 = x_6 \oplus t5 \oplus t3$ $\quad$ $y_{13} = x_{13} \oplus t1 \oplus t2$

$y_6 = x_5 \oplus t4 \oplus t6$ $\quad$ $y_{14} = x_{14} \oplus t7 \oplus t8$

$y_7 = x_8 \oplus t2 \oplus t8$ $\quad$ $y_{15} = x_{15} \oplus t5 \oplus t6$

$y_8 = x_7 \oplus t1 \oplus t7$ $\quad$ $y_{16} = x_{16} \oplus t_3 \oplus t4$

L'opération de diffusion inverse et l'équation de diffusion inverse sont obtenues respectivement à partir de l'opération de diffusion et l'équation de diffusion, en remplaçant $y_i \, par \, x_i$ , $i = 1, \cdots 16$ et vice versa dans les relations et équations de diffusion.

Nous donnons ci-dessous les codes Matlab de calcul des différents termes $ti$, $i = 1, \cdots 8$, et de la matrice de diffusion binaire 16x16.

Le code Matlab de l'algorithme donnant les termes en commun est donné ci-dessous :

```
% Mettre les indices indiqaunt la présence de 1 dans une cellule
for i = 1:size(D,1)
  x{i} = (find(D(i,:) == 1));
end
% Chercher les indices communs entre toutes les lignes prises deux à deux
for i = 1:size(D)
for j = i + 1:size(D)
  C = intersect(x{i}, x{j})
  c{i,j} = C;
  l(i,j) = length(C);
end
end
```

Le code Matlab pour le calcul de la matrice de diffusion est :

```
I4 = eye(4,4);
o4 = zeros(4,4);
HL = logical([1 1 1 0; 1 0 1 1; 1 1 0 1; 0 1 1 1]);
d = find(HL == 1);
LL = zeros(16,16);
for it1 = 1:size(HL,1)
    for it2 = 1:size(HL,2)
        if HL(it1,it2) == 1
           LL((it1-1)*4+1:it1*4,(it2-1)*4+1:it2*4) = I4;
        end
    end
end
LL = LL;
Hm1 = not(eye(4,4))
Hm4 = fliplr(Hm1)
Hm2 = [1 0 1 1; 0 1 1 1; 1 1 1 0; 1 1 0 1]
Hm3 = [1 1 0 1; 1 1 1 0; 0 1 1 1; 1 0 1 1]
MM = [Hm1 o4 o4 o4; o4 Hm2 o4 o4; o4 o4 Hm3 o4; o4 o4 o4 Hm4];
D = LL * MM * LL;
D = mod(D,2)
```

### 3.4.5 Couche de diffusion à base de la carte chaotique cat de haute dimension

Pour construire une carte cat de haute dimension, nous utilisons deux approches : la première approche s'appuie sur la multiplication des cartes cat [Chen et al, 2004], et la deuxième approche utilise une transformation linéaire [Liu et al, 2008].

*2.4.5.1 Première approche*

La carte chaotique 2-D cat est utilisée ici en tant que couche de diffusion et est donnée par la relation suivante:

$$\begin{bmatrix} BS1d \\ BS2d \end{bmatrix} = Mod\left\{A \times \begin{bmatrix} BS1 \\ BS2 \end{bmatrix}, 2^{Tb/2}\right\} \tag{3.27}$$

Avec, $\{BS1,\ BS2\}$ sont deux sous blocs du bloc $BS$, sortie de la couche de substitution, et $\{BS1d,\ BS2d\}$ sont les deux sous blocs résultats de l'équation ci-dessus. Les valeurs de chacun de ces sous blocs varient entre $\left[0,\ 2^{\frac{Tb}{2}} - 1\ \right]$.

La carte cat est bijective puisque son déterminant est $|A| = \pm 1$.

Nous rappelons ci-dessous les quatre formes de la carte cat que nous utilisons dans la suite de ce chapitre.

$$A0 = \begin{bmatrix} 1 & u \\ v & \pm 1 + uv \end{bmatrix}, |A0| = \pm 1 + uv - uv = \pm 1 \tag{3.28}$$

$$A1 = \begin{bmatrix} u & 1 \\ \pm 1 + uv & v \end{bmatrix}, |A1| = uv - uv \pm 1 = \pm 1 \tag{3.29}$$

$$A2 = \begin{bmatrix} u & \pm 1 + uv \\ 1 & v \end{bmatrix}, |A2| = \pm 1 + uv - uv = \pm 1 \tag{3.30}$$

$$A3 = \begin{bmatrix} \pm 1 + uv & v \\ u & 1 \end{bmatrix}, |A3| = \pm 1 + uv - uv = \pm 1 \tag{3.31}$$

Les paramètres $u, v \in \left[0, 2^{Tb/4}\right]$ pour éviter les points fixes. En effet, les valeurs grandes de $u, v$ produisent des points fixes.

Nous pouvons augmenter le pouvoir de diffusion entre les éléments d'un bloc, par multiplication des différentes cartes cat 2D (augmentation du nombre de branches linéaires). Ceci est possible, grâce à la propriété mathématique suivante :

Le déterminant du produit de deux matrices carrées $A$ et $B$ de taille $(n \times n)$, est égale au produit des déterminants de deux matrices. Donc, si $a$ et $b$ sont respectivement, les déterminants de la matrice $A$ et de la matrice $B$, alors le déterminant de la matrice de multiplication $A \times B$ est $a \times b$.

Donc, la multiplication de deux matrices cat parmi les quatre possibles forme une carte chaotique cat avec 4 paramètres. La multiplication des quatre matrices possibles, c'est-à-dire $A0 \times A1 \times A2 \times A3$ forme une matrice avec 8 paramètres. De cette manière, on augmente la taille de la clé dynamique de la couche de diffusion, cependant, on ne peut pas assurer que le résultat de la multiplication ne donne pas des paramètres de contrôle bloquants. Afin de contourner ce problème, nous proposons de limiter la précision des paramètres de contrôle

selon le nombre de matrices qui entre dans la multiplication, nous avons pris : $u, v \in \left[0, 2^{Tb/8}\right]$ dans le cas du produit de 3 matrices et $u, v \in \left[0, 2^{Tb/16}\right]$ dans le cas du produit de 4 matrices.

La construction d'une carte cat de dimension supérieure à 2 se fait comme suit :

- Génération de $C_2^n = n \times (n-1)/2$ matrices $A_{ij}$ de taille $n \times n$ à partir d'une structure donnée de la carte cat à deux dimensions (comme expliqué ci-dessous, utilisant la structure $A0$)
- Construction de la carte cat étendue de dimension $n \times n$, par multiplication des différentes matrices générées.

$$A0n = \prod_{i=1}^{n-1} \prod_{j=2,\ j>i}^{n} A_{i,j} \tag{3.32}$$

Les deux étapes précédentes sont répétées pour chaque structure (au nombre de quatre) de la carte cat à deux dimensions.

La matrice de diffusion finale de taille 8 X 8, est obtenue par $A08$ (matrice de taille 8 X 8) ou par multiplication de deux matrices $A08 \times A18$, ou par multiplication de 3 matrices $A08 \times A18 \times A28$ ou par multiplication de 4 matrices $A08 \times A18 \times A28 \times A38$.

**Génération des matrices de tailles 3×3 et 4×4**

Générons la carte chaotique cat étendue à trois et quatre dimension à partir de la structure $A0$ [Fridrich, 1998]. Le même calcul est fait pour les autres structures.

Notons aussi que la procédure de génération utilisée, est facilement applicable pour la génération des matrices de taille $n \times n$, avec $n > 4$.

La carte chaotique $A03$ à 3 dimensions est définie comme suit:

$$\begin{bmatrix} BS1d \\ BS2d \\ BS3d \end{bmatrix} = Mod\left\{ A03 \times \begin{bmatrix} BS1 \\ BS2 \\ BS3 \end{bmatrix}, 2^{Tb/3} \right\} \tag{3.33}$$

Avec :

$$A03 = A_{12} \times A_{13} \times A_{23} \tag{3.34}$$

Le choix des différentes matrices $A_{ij}$ se fait pour i ≠ j et j > i

Les différentes matrices $A_{12}, A_{13}, A_{23}$ sont construites à partir de la matrice $A0$ comme suit :

$$A_{12} = \begin{bmatrix} 1 & u_{12} & 0 \\ v_{12} & \mp 1 + u_{12}v_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix}, A_{13} = \begin{bmatrix} 1 & 0 & u_{13} \\ 0 & 1 & 0 \\ v_{13} & 0 & \mp 1 + u_{13}v_{13} \end{bmatrix}, A_{23} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & u_{23} \\ 0 & v_{23} & \mp 1 + u_{23}v_{23} \end{bmatrix}$$

Alors (résultat obtenu avec Maple) :

$$A03 = \begin{bmatrix} 1 & u_{12} + u_{13}v_{23} & u_{12}u_{23} + u_{13}(\pm 1 + u_{23}v_{23}) \\ v_{12} & \pm 1 + u_{12}v_{12} + u_{13}v_{12}v_{23} & (\pm 1 + u_{12}v_{12})u_{23} + v_{12}u_{13}(\pm 1 + u_{23}v_{23}) \\ v_{13} & (\pm 1 + u_{13}v_{13})v_{23} & (\pm 1 + u_{13}v_{13})(\pm 1 + u_{23}v_{23}) \end{bmatrix} \quad (3.35)$$

La carte chaotique $A04$ à 4 dimensions est définie comme suit :

$$\begin{bmatrix} BS1d \\ BS2d \\ BS3d \\ BS4d \end{bmatrix} = Mod\left\{ A04 \times \begin{bmatrix} BS1 \\ BS2 \\ BS3 \\ BS4 \end{bmatrix}, 2^{Tb/4} \right\} \quad (3.36)$$

où

$$A04 = A_{12} \times A_{13} \times A_{14} \times A_{23} \times A_{24} \times A_{34} \quad (3.37)$$

Avec:

$$A_{12} = \begin{bmatrix} 1 & u_{12} & 0 & 0 \\ v_{12} & \pm 1 + u_{12}v_{12} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_{13} = \begin{bmatrix} 1 & 0 & u_{13} & 0 \\ 0 & 1 & 0 & 0 \\ v_{13} & 0 & \pm 1 + u_{13}v_{13} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{14} = \begin{bmatrix} 1 & 0 & 0 & u_{14} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ v_{14} & 0 & 0 & \pm 1 + u_{14}v_{14} \end{bmatrix}$$

$$A_{23} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & u_{23} & 0 \\ 0 & v_{23} & \pm 1 + u_{23}v_{23} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \; A_{24} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & u_{24} \\ 0 & 0 & 1 & 0 \\ 0 & v_{24} & 0 & \pm 1 + u_{24}v_{24} \end{bmatrix}$$

$$A_{34} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & v_{34} & \pm 1 + u_{34}v_{34} \end{bmatrix}.$$

Nous obtenons par Maple:

$$A04 = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}$$

Avec :

$a_{1,1} = 1$
$a_{1,2} = a12 + a13 * b23 + a14 * b24,$
$a_{1,3} = a12 * a23 + a13 * (1 + a23 * b23) + ((a12 + a13 * b23) * a24 + a14 * (1 + a24 * b24)) * b34$
$a_{1,4} = (a12 * a23 + a13 * (1 + a23 * b23)) * a34 + ((a12 + a13 * b23) * a24 + a14 * (1 + a24 * b24)) * (1 + a34 * b34)$
$a_{2,1} = b12$
$a_{2,2} = 1 + a12 * b12 + b12 * a13 * b23 + b12 * a14 * b24$
$a_{2,3} = (1 + a12 * b12) * a23 + b12 * a13 * (1 + a23 * b23) + ((1 + a12 * b12 + b12 * a13 * b23) * a24 + b12 * a14 * (1 + a24 * b24)) * b34$
$a_{2,4} = ((1 + a12 * b12) * a23 + b12 * a13 * (1 + a23 * b23)) * a34 + ((1 + a12 * b12 + b12 * a13 * b23) * a24 + b12 * a14 * (1 + a24 * b24)) * (1 + a34 * b34)$
$a_{3,1} = b13$
$a_{3,2} = (1 + a13 * b13) * b23 + b13 * a14 * b24$
$a_{3,3} = (1 + a13 * b13) * (1 + a23 * b23) + ((1 + a13 * b13) * b23 * a24 + b13 * a14 * (1 + a24 * b24)) * b34$
$a_{3,4} = (1 + a13 * b13) * (1 + a23 * b23) * a34 + ((1 + a13 * b13) * b23 * a24 + b13 * a14 * (1 + a24 * b24)) * (1 + a34 * b34)$
$a_{4,1} = b14$
$a_{4,2} = (1 + a14 * b14) * b24$
$a_{4,3} = (1 + a14 * b14) * (1 + a24 * b24) * b34$
$a_{4,4} = (1 + a14 * b14) * (1 + a24 * b24) * (1 + a34 * b34)]$

Ainsi, selon ce qui précède, la matrice $A08$ de taille 8 X 8, est obtenue comme suit :

$$A08 = \prod_{i=1}^{7} \prod_{j=2,\ j>i}^{8} A_{i,j} = A_{1,2} \times \cdots \times A_{1,8} \times A_{2,1} \cdots \times A_{2,8} \times \cdots \times A_{7,8} \qquad (3.38)$$

Pour plus de clarté, nous donnons ci-dessous la forme des 3 matrices $A_{1,2}$, $A_{3,5}$ et $A_{7,8}$ entrant dans le calcul de $A08$

$$A_{1,2} = \begin{bmatrix} 1 & u_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ v_{12} & \pm 1 + u_{12}v_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{3,5} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & u_{35} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & v_{35} & 0 & \pm 1 + u_{35}v_{35} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{7,8} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & u_{78} \\ 0 & 0 & 0 & 0 & 0 & 0 & v_{78} & \pm 1 + u_{78}v_{78} \end{bmatrix}$$

Dans le tableau ci-dessous, nous donnons un exemple de résultat de la matrice $A08$ et de son inverse. A cet effet, nous avons généré 28 couples aléatoires $\{u, v\} \in \{0, 1, 2, 3\}$ afin de former les 28 matrices $A_{i,j}$ permettant le calcul de $A08$.

Tableau 3.1. Exemple de résultat obtenu pour la matrice $A08$ et $A08^{-1}$.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 9 | 16 | 35 | 43 | 104 | 165 |
| 2 | 7 | 21 | 38 | 83 | 103 | 247 | 391 |
| 2 | 5 | 16 | 28 | 61 | 74 | 181 | 288 |
| 2 | 2 | 4 | 5 | 12 | 11 | 34 | 57 |
| 1 | 2 | 4 | 4 | 11 | 12 | 32 | 52 |
| 2 | 3 | 5 | 6 | 13 | 12 | 38 | 64 |
| 2 | 1 | 3 | 0 | 6 | 0 | 12 | 23 |
| 2 | 0 | 3 | 0 | 6 | 0 | 12 | 24 |

**Matrice : $A08$**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 81 | -27 | -9 | 0 | 0 | -3 | 0 | -1 |
| 57 | -17 | -9 | -2 | -1 | 0 | 0 | 0 |
| 260 | -84 | -32 | -8 | -2 | -4 | 0 | -1 |
| 304 | -97 | -39 | -9 | -4 | -4 | 0 | -1 |
| 535 | -169 | -71 | -16 | -7 | -7 | 0 | -1 |
| 285 | -90 | -38 | -8 | -3 | -4 | -1 | 0 |
| -460 | 144 | 63 | 14 | 6 | 5 | 2 | -1 |
| 57 | -17 | -9 | -2 | -1 | 0 | -1 | 1 |

**Matrice : $A08^{-1}$**

Notons au passage que ces matrices de valeurs entières et inversibles sont aussi utilisées dans la réalisation de fonctions de hachage reliant ainsi les bits de sortie en fonction de tous les bits d'entrée.

La construction d'une carte cat multidimensionnelle de taille $8 \times 8$, à partir des quatre structures de base de la carte cat 2-D est alors donnée par :

$$A8 = \mathrm{A08} \times \mathrm{A18} \times \mathrm{A28} \times \mathrm{A38} \tag{3.39}$$

Où les différentes matrices A18, A28 et A38 sont calculées de la même manière que la matrice A08.

*2.4.5.2 Deuxième approche*

La deuxième approche utilisée pour construire une carte cat multidimensionnelle en tant que couche de diffusion, s'appuie sur la méthode décrite par [Liu et al, 2008] et dont l'objectif initial était de réaliser un générateur de séquences chaotiques.

Cette méthode est déjà décrite en détail dans le chapitre 2. Elle est basée sur une structure matricielle particulière, simple à réaliser et performante en termes de vitesse de calcul.

Le processus de diffusion est fait au niveau octets ou au niveau des mots de 32 bits. La dimension $n \times n$ de la matrice de diffusion $D$ est flexible, avec $n = 4, 8, 16, 32, 64$ et $n = Tb/Tsb$, avec $Tsb = 8, ou\ 32$ bits.

Dans cette formulation rappelée ci-dessous, les paramètres de contrôle $u$ et $v$ sont remplacés par des sous matrices $M$ afin de former la matrice de diffusion :

$$\begin{bmatrix} BS1d \\ BS2d \\ \cdots \\ BSnd \end{bmatrix} = D \times \begin{bmatrix} BS1 \\ BS2 \\ \cdots \\ BSn \end{bmatrix} mod\ 2^{Tsb} \qquad (3.40)$$

Afin, d'augmenter la taille de la clé dynamique et assurer plus de diffusion, nous utilisons les quatre structures de la carte cat 2-D, au lieu d'une seule structure proposée par [Liu et al, 2008].

La forme finale de la matrice de diffusion dépend du nombre ns des structures utilisées.

Si $ns = 1, D = M_0$

Si $ns = 2, D = M_0 \times M_1$

Si $ns = 3,\ D = M_0 \times M_1 \times M_2$

Si $ns = 4, D = M_0 \times M_1 \times M_2 \times M_3$

où $M_0,\ M_1,\ M_2,$ et $M_3,$ sont quatre matrices construites respectivement à partir des 4 structures $A0,\ A1,\ A2,$ et $A3$ .
avec :

$$M_0 = \begin{bmatrix} I_m & M_{u0} \\ M_{v0} & I_l + M_{v0}M_{u0} \end{bmatrix},\quad M_1 = \begin{bmatrix} I_l + M_{v1}M_{u1} & M_{v1} \\ M_{u1} & I_m \end{bmatrix},$$

$$M_2 = \begin{bmatrix} M_{v2} & I_l + M_{v2}M_{u2} \\ I_m & M_{u2} \end{bmatrix},\quad M_3 = \begin{bmatrix} M_{u2} & I_m \\ I_l + M_{v2}M_{u2} & M_{v2} \end{bmatrix}$$

$I_m$ et $I_l$ sont deux matrices identité de taille $m$ et $l$ respectivement. Les éléments des sous matrices $M_{ui}$ et $M_{vi}$ forment la clé dynamique.

$M_{ui}$ et $M_{vi}$ sont deux sous matrices non-zéro de taille $m \times l$ et $l \times m$ pour la structure $i$, avec $l = n - m$. Elles peuvent être identiques pour les différentes structures ou différentes pour chaque structure.

Dans notre simulation, nous avons utilisé les même $M_{ui}$ et $M_{vi}$ pour les différentes structures. Ceci permet d'avoir une taille de la clé de diffusion comprise entre 32 bits minimum et 128 bits maximum.

Les déterminants des différentes matrices $M_i, i = 0, 1, 2, 3$ sont égaux à l'unité, donc ces matrices sont réversibles. En effet, si

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

alors, le déterminant de $M$ est donné par [Brookes, 2005] :

$det\ (M)\ =\ det\ (A)\ \times\ Det\ (D\ -\ CA^{-1}B)$ , où $A$ et D sont des matrices carrées

Par exemple pour $M = M_0$, le déterminant de $M_0$ est:

$$\det(M_0) = \det(I_m) \times \det\left(I_l + M_{v0}M_{u0} - M_{v0}{I_m}^{-1}M_{u0}\right)$$

$$= \det(I_m) \times \det(I_l + M_{v0}M_{u0} - M_{v0}M_{u0})$$

$$= \det(I_m) \times \det(I_l) = 1$$

## Exemple de calcul de la matrice de diffusion *D*

Pour $n = 8$, on peut choisir arbitrairement les matrices $M_u$ et $M_v$ (normalement calculées à partir de la clé de diffusion dynamique), par exemple, si on fixe le paramètre $m = 4$, alors $l = n - m = 4$, et :

$$M_u = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad M_v = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Donc, si nous utilisons les mêmes matrices $M_u$, et $M_v$ dans le calcul des différentes matrices $M_i, i = 0, 1, 2, 3,$ nous obtenons :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 3 | 2 | 2 | 3 |
| 1 | 0 | 1 | 1 | 2 | 3 | 1 | 3 |
| 0 | 0 | 1 | 0 | 1 | 1 | 2 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

$M_1$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 3 | 1 | 1 | 1 | 0 |
| 2 | 3 | 1 | 3 | 1 | 0 | 1 | 1 |
| 1 | 1 | 2 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

$M_2$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 3 | 2 | 2 | 3 | 1 | 1 | 1 | 0 |
| 2 | 3 | 1 | 3 | 1 | 0 | 1 | 1 |
| 1 | 1 | 2 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

$M_3$

Pour $ns = 1,\ D_1 = M_0$ par exemple (on aurait pu prendre n'importe quelle matrice $M_i, i = 0, 1, 2, 3$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 3 | 2 | 2 | 3 |
| 1 | 0 | 1 | 1 | 2 | 3 | 1 | 3 |
| 0 | 0 | 1 | 0 | 1 | 1 | 2 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Matrice de diffusion $D_1$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 0 | 1 | 1 | 0 | -1 | 0 | -1 |
| 1 | 2 | 2 | 0 | -1 | 0 | -1 | -1 |
| 2 | 1 | 4 | 1 | -1 | -1 | -1 | -1 |
| 1 | 1 | 1 | 1 | -1 | 0 | 0 | -1 |
| -1 | -1 | -1 | 0 | 1 | 0 | 0 | 0 |
| -1 | 0 | -1 | -1 | 0 | 1 | 0 | 0 |
| 0 | 0 | -1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Matrice de diffusion inverse $D_1^{-1}$

Pour $ns = 2,\ D_2 = M_0 \times M_1$ par exemple

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 1 | 5 | 2 | 3 | 5 |
| 2 | 0 | 2 | 2 | 4 | 5 | 2 | 6 |
| 1 | 1 | 2 | 1 | 4 | 3 | 4 | 5 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 3 |
| 5 | 3 | 5 | 4 | 13 | 11 | 9 | 17 |
| 3 | 4 | 3 | 3 | 11 | 6 | 8 | 14 |
| 1 | 1 | 3 | 1 | 5 | 4 | 5 | 6 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

Matrice de diffusion $D_2$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -4 | -5 | -6 | -1 | 4 | 0 | 2 | 3 |
| -8 | -3 | -11 | -5 | 3 | 4 | 3 | 3 |
| -9 | -6 | -15 | -4 | 5 | 3 | 6 | 5 |
| -5 | -2 | -5 | -3 | 2 | 2 | 1 | 3 |
| 4 | 1 | 4 | 2 | -1 | -2 | -1 | -1 |
| 2 | 3 | 4 | 0 | -2 | 0 | -2 | -2 |
| 2 | 1 | 5 | 1 | -1 | -1 | -2 | -1 |
| 1 | 1 | 1 | 1 | -1 | 0 | 0 | -1 |

Matrice de diffusion inverse $D_2^{-1}$

Pour $ns = 3,\ D_3 = M_0 \times M_1 \times M_3$ par exemple

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 18 | 17 | 11 | 26 | 8 | 3 | 7 | 7 |
| 21 | 12 | 15 | 27 | 6 | 7 | 6 | 6 |
| 19 | 15 | 14 | 25 | 6 | 4 | 8 | 6 |
| 7 | 3 | 3 | 9 | 2 | 2 | 1 | 3 |
| 63 | 46 | 43 | 83 | 21 | 16 | 22 | 20 |
| 48 | 40 | 30 | 66 | 18 | 9 | 18 | 18 |
| 23 | 18 | 18 | 30 | 7 | 5 | 10 | 7 |
| 1 | 1 | 0 | 3 | 1 | 0 | 0 | 1 |

Matrice de diffusion $D_3$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -12 | -10 | -19 | -4 | 8 | 3 | 7 | 7 |
| -15 | -6 | -21 | -9 | 6 | 7 | 6 | 6 |
| -11 | -7 | -20 | -5 | 6 | 4 | 8 | 6 |
| -5 | -2 | -5 | -3 | 2 | 2 | 1 | 3 |
| 24 | 9 | 30 | 14 | -9 | -11 | -8 | -10 |
| 30 | 22 | 48 | 12 | -18 | -9 | -18 | -18 |
| 45 | 26 | 70 | 22 | -23 | -17 | -24 | -23 |
| 18 | 13 | 25 | 8 | -11 | -5 | -8 | -11 |

Matrice de diffusion inverse $D_3^{-1}$

Pour $ns = 4,\ D_4 = M_0 \times M_1 \times M_3 \times M_4$ par exemple

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 91 | 61 | 61 | 119 | 29 | 25 | 29 | 29 |
| 92 | 75 | 58 | 126 | 34 | 18 | 34 | 34 |
| 88 | 65 | 61 | 117 | 29 | 21 | 32 | 29 |
| 26 | 21 | 14 | 38 | 11 | 5 | 8 | 11 |
| 289 | 218 | 191 | 388 | 100 | 67 | 102 | 99 |
| 226 | 159 | 151 | 301 | 75 | 58 | 75 | 75 |
| 107 | 80 | 75 | 142 | 35 | 25 | 40 | 35 |
| 7 | 3 | 3 | 9 | 2 | 2 | 1 | 3 |

Matrice de diffusion $D_4$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 62 | 32 | 90 | 32 | -29 | -25 | -29 | -29 |
| 58 | 41 | 92 | 24 | -34 | -18 | -34 | -34 |
| 56 | 33 | 90 | 27 | -29 | -21 | -32 | -29 |
| 18 | 13 | 25 | 8 | -11 | -5 | -8 | -11 |
| -88 | -64 | -136 | -36 | 53 | 26 | 49 | 52 |
| -151 | -84 | -226 | -76 | 75 | 58 | 75 | 75 |
| -205 | -126 | -317 | -96 | 109 | 73 | 111 | 109 |
| -85 | -47 | -120 | -43 | 42 | 32 | 38 | 43 |

Matrice de diffusion inverse $D_4^{-1}$

Ci-dessous (voir figure 3.19), nous donnons le pseudo code Matlab (fonction $Catdimension$) permettant la construction de la matrice de diffusion.

```
function R = couchedediffusioncat(B, Mu, Mv, ns, n, d)
% B est le bloc d'entrée en octets
D = Catdimension(length(B), n, double(Mu), double(Mv), ns);
If d == 0
% CHIFFREMENT
R = mod((D * B'), 2^8);
else
%DECHIFFREMENT
invD = round(inv(D));
R = mod(invD * (B), 2^8);
end
```

Fig. 3.19: Pseudo code Matlab pour la construction de la couche de diffusion

Rappelons que la matrice de diffusion $D$ est utilisée dans le processus de chiffrement et la matrice de diffusion inverse $D^{-1} = inv(D)$ est utilisée dans le processus de déchiffrement.

Dans le cas où les matrices $M_u, M_v$ sont identiques pour les différentes structures, alors le pseudo code Matlab (voir figure 3.20) pour le calcul de la matrice $D$ est :

```
function D = Catdimension(m, n, Mu, Mv, ns)
% n = taille de la matrice carrée pr exemple n = 16 pour Tb = 128;
% m = taille de la matrice identite;
% Mu, Mv deux matrices générées aléatoirement ;
l = m - n;
Im = eye(m);
Il = eye(l);
M0 = [Im Mu; Mv Il + Mv * Mu];
M1 = [Mv Il + Mv * Mu; Im Mu; ];
M2 = [Il + Mv * Mu Mv ; Mu Im; ];
M3 = [Mu Im ; Il + Mv * Mu Mv ];
    if (ns == 1)
        D = M0;
   elseif(ns == 2)
        D = M0 * M1;
   elseif (ns == 3)
        D = M0 * M1 * M2;
  elseif(ns == 4)
        D = M0 * M1 * M2 * M3;
end
```

Fig. 3.20: Pseudo code Matlab pour la construction de la couche de diffusion dans le cas où les matrices sont identiques.

Dans le cas d'utilisation de matrices $M_u, M_v$ différentes pour chaque structure, la procédure de création de la matrice de diffusion est présentée par le code Matlab suivant :

```
function D = Catdimensionupdate2(m, n, Mu1, Mu2, Mu3, Mu4, Mv1, Mv2, Mv3, Mv4, ns)
% n =  taille de la matrice carrée pr exemple n = 16 pour Tb = 128;
% m =  taille de la matrice identité;
% Mu, Mv deux matrices générées aléatoirement ;
l = m − n;
Im = eye(m);
Il = eye(l);
M1 = [Im      Mu1;   Mv1      Il + Mv1 ∗ Mu1];
M2 = [Mv2    Il + Mv2 ∗ Mu2;   In2         Mu2; ];
M3 = [Il + Mv3 ∗ Mu3      Mv3 ; Mu3      Im; ];
M4 = [Mu4    Im ;   Il + Mv4 ∗ Mu4    Mv4 ];
        if (ns == 1)
            D = M1;
        elseif(ns == 2)
            D = M1 ∗ M2;
        elseif (ns == 3)
            D = M1 ∗ M2 ∗ M3;
        elseif(ns == 4)
            D = M1 ∗ M2 ∗ M3 ∗ M4;
end
```

Fig. 3.21: Pseudo code Matlab pour la construction de la couche de diffusion dans le cas où les matrices sont identiques.

### 3.4.6 Couche de permutation basée sur la carte chaotique skew tent.

L'opération de permutation, réalisée par la carte chaotique skew tent, est faite sur les octets du bloc $BS$ de taille $Tbo$, avec $Tbo = Tb/8$ octets

L'équation est alors :

$$mn = P_{b_{j,i}}(m) = \begin{cases} \left\lceil \frac{Tbo}{b_{j,i}} \times m \right\rceil & 1 \leq m \leq b_{j,i} \\ \left\lfloor \frac{Tbo}{Tbo - b_{j,i}} (Tbo - m) \right\rfloor + 1 & b_{j,i} < m \leq Tbo \end{cases} \tag{3.41}$$

où $m$ et $mn$ sont les positions originales et permutées des octets du bloc $BS$, avec :
$mn \in \{1, \cdots, Tbo\}$, $b_{j,i} \in \{1, \cdots, Tbo\}$ $et$ $i = 1, 2, \ldots, nb$, où $nb$ est le nombre de paramètres de contrôle utilisés dans la fonction de permutation. Pour chaque paramètre de contrôle $b_{j,i}$ $i = 1, 2, \ldots, nb$, la fonction de permutation est itérée $rp$ fois, sur le bloc sous traitement, donc au total la fonction de permutation est exécutée $nb \times rp$ fois.

Les paramètres nécessaires sont $b_{j,i}$ , $i = 1, \cdots, nb$ avec $1 \leq b_{j,i} \leq Tbo$ et $nb = 4$. Ils sont produits comme suit :

$$b_{j,1} = mod((\neg O_1 \wedge O_2) \vee (\neg O_1 \wedge O_3) \oplus O_4, Tbo)$$
$$b_{j,2} = mod(O_2 \oplus (O_3 \wedge \neg O_4), To)$$
$$b_{j,3} = mod((\neg O_2 \wedge O_3) \vee (\neg O_2 \wedge O_4) \oplus O_1, Tbo)$$
$$b_{j,4} = mod(O_2 \oplus (O_3 \wedge \neg O_4), Tbo)$$

Avec :

$$O_1 = \oplus_{l=1}^{4} T_1(l)$$
$$O_2 = \oplus_{l=1}^{4} T_2(l)$$
$$O_3 = \oplus_{l=1}^{4} T_3(l)$$
$$O_3 = \oplus_{l=1}^{4} T_4(l)$$
$$T_i = Typecast(uint32(X_i(n)), 'uint8'), i = 1, 2, 3, 4$$

### 3.4.7 Couche de permutation inverse basée sur l'équation inverse de la carte chaotique skew tent.

L'équation de permutation inverse de la carte chaotique skew tente, appliquée sur les indices du bloc chiffré de taille $Tbo$ est donnée ci-dessous.

$$imn = P^{-1}{}_{b_{j,i}}(m) = \begin{cases} \zeta 1 & si \quad \theta(m) = m \ et \left(\frac{\zeta 1}{b_{j,i}}\right) > \left(\frac{Tbo - \zeta 2}{Tbo - b_{j,i}}\right) \\ \zeta 2 & si \quad \theta(m) = m \ et \left(\frac{\zeta 1}{b_{j,i}}\right) \leq \left(\frac{Tbo - \zeta 2}{Tbo - b_{j,i}}\right) \\ \zeta 3 = \zeta 1 & si \quad \theta(m) = m + 1 \end{cases} \tag{3.42}$$

où :

$$\zeta 1 = ceil\left[\left(\frac{b_{j,i}}{Tbo}\right) \times m\right], \quad \zeta 2 = floor\left[\left(\frac{b_{j,i}}{Tbo} - 1\right) \times m + Tbo\right]$$
$$\zeta 3 = floor[\left(\frac{b_{j,i}}{Tbo}\right) \times m], \quad \theta(m) = m + \zeta 1 - \zeta 3 + 1$$

Avec : $imn \in \{1, \cdots, Tbo\}$ est l'indice inverse de permutation de l'indice de permutation $mn$ du chiffrement, $m \in \{1, \cdots, Tbo\}$ est l'indice d'entrée, $b_{j,i} \in \{1, \cdots, Tbo\}$ $et$ $i = nb, nb - 1, \cdots, 2, 1$ où, $nb$ est le nombre de paramètres de contrôle utilisés dans la fonction inverse de permutation.

Pour chaque paramètre de contrôle $b_{j,i}$ $i = nb, nb - 1, \cdots, 2, 1$, l'équation inverse de permutation est itérée $rp$ fois, sur le bloc sous traitement, donc au total pour une itération, l'équation inverse de permutation est exécutée $nb \times rp$ fois.

Chaque paramètre de contrôle $b_{j,i}$ $i = 1, 2, \ldots, nb$ est itéré sur le bloc sous traitement $rp$ fois.

## 3.5 Analyse de la sécurité des crypto-systèmes et résultats expérimentaux

Les crypto-systèmes SPN et Feistel sont composés de plusieurs couches consécutives. La façon de réaliser et d'enchaîner ces couches, détermine les performances des crypto-systèmes en termes de rapidité d'exécution et de robustesse vis-à-vis des attaques cryptographiques. Dans cette partie nous nous proposons d'étudier et de quantifier les performances des fonctions chaotiques (cartes chaotiques) utilisées dans les différentes couches afin de quantifier les performances des crypto-systèmes proposés. A ce propos, nous nous appuyons sur les outils d'analyse des performances des crypto-systèmes de la littérature spécialisée.

D'après Shannon, si les différentes couches d'un crypto-système donné sont performantes, alors le crypto-système en question est performant.

Nous commençons par introduire tous les outils nécessaires à l'analyse de la sécurité, puis nous considérons les performances :

- de la couche de substitution, réalisée par la carte chaotique Skew tent,
- de la couche de permutation, réalisée par la carte chaotique Skew tent, puis par la carte chaotique 2-D,
- de la couche de diffusion, réalisée par la matrice binaire statique, et par la carte cat dynamique multidimensionnelle.

Enfin, nous évaluons les performances des crypto-systèmes proposés aux différentes attaques cryptographiques connues :

- sensibilité aux textes en clair (mesure de l'effet d'avalanche)
- sensibilité à la clé secrète en émission et en réception
- analyse statistique de la sécurité (chi-carré, entropie, histogramme, corrélation), temps de calcul.

### 3.5.1 Outils d'analyse de la sécurité

**Préliminaires :**

#### *2.5.1.1 Attaques statistiques*

Shannon [Shannon, 1949] a dit "il est possible de casser un bon nombre de crypto-systèmes par une analyse statistique de ces derniers". Pour résister à l'attaque statistique il faut que :

- l'histogramme de l'image chiffrée soit quasi uniformément distribué, de cette manière l'attaquant ne peut pas avoir de l'information sur l'image en clair.
- les coefficients de corrélation des pixels adjacents de l'image chiffrée dans toutes les directions soient proches de zéros.

#### *2.5.1.2 Attaques différentielles et linéaires*

Les attaques les plus puissantes et les plus courantes sur les algorithmes de chiffrement/déchiffrement par bloc sont les attaques linéaire [Matsui, 1993] et différentielle [Biham et Shamir, 1991]. Selon [Koo et al, 2003], la résistance contre ces deux types d'attaques, dépend essentiellement de la robustesse de la couche de substitution et de la couche de diffusion. La couche de substitution doit assurer que le maximum des probabilités d'approximation linéaire et différentielle soit le plus faible possible ($2^{-6}$ dans le cas de l'AES, et entre $2^{-5,8}$ et $2^{-5}$ pour nos crypto-systèmes). La couche de diffusion doit assurer un nombre de branches linéaires grand (5 pour l'AES, et entre 8 et 10 pour les crypto-systèmes proposés).
Les couches de confusion/diffusion utilisées dans les crypto-systèmes proposés sont dynamiques, car les clés utilisées dans ses couches varient pour chaque itération, par conséquent les crypto-systèmes proposés résistent à ces deux types d'attaques et possèdent une complexité supplémentaire contre la cryptanalyse par rapport à l'utilisation des couches statiques.

#### *2.5.1.3 Attaques à texte en clair/chiffré connu ou choisi*

Les crypto-systèmes proposés assurent l'effet d'avalanche. En effet, quelle que soit la position d'un bit changé dans un bloc de texte en clair ou chiffré, ceci provoque pratiquement le changement de 50% des bits entre les deux blocs chiffré ou déchiffrée respectivement (effet d'avalanche).
La mesure de l'effet d'avalanche doit être faite au niveau du bloc et non au niveau de l'image. Par ailleurs, pour réaliser l'effet d'avalanche, certains crypto-systèmes basés chaos relient le processus de génération des clés dynamiques ou de permutation avec le bloc chiffré précédent, ceci entraine des erreurs de propagation.
Parmi, les différents modes opératoires cryptographiques, le mode CBC est le mode qui réalise le plus l'effet d'avalanche dans un sens.

#### *2.5.1.4 Attaque exhaustive de la clé secrète*

Pour résister à l'attaque exhaustive de la clé secrète, un crypto-système devrait avoir une clé secrète de taille assez grande ($\geq$ 128 bits). Dans les crypto-systèmes basés chaos, la clé secrète est celle du générateur des séquences chaotiques utilisé. Elle est formée des conditions initiales et des paramètres de contrôle du générateur en question.
Les générateurs chaotiques proposés et utilisés dans nos crypto-systèmes, possèdent un espace de clé secrète plus grand que $2^{128}$ bits, et donc ils résistent à l'attaque exhaustive.

#### *2.5.1.5 Sensibilité à la clé secrète en émission/réception*

Un crypto-système est dit sensible à la clé secrète en émission/réception, si :

- le chiffrement à l'émission d'une même image avec deux clés qui diffèrent d'un seul bit, donne deux images chiffrées qui diffèrent de 50 % en bits ;
- le déchiffrement à la réception d'une même image chiffrée mais avec deux clés qui diffèrent d'un bit, fournit deux images déchiffrées qui diffèrent de 50% en bits.

Cette sensibilité est classiquement mesurée par deux paramètres qui sont le NPCR (Number of Pixel Change Rate) et le UACI (Unified Average Changing Intensity) définis plus loin.
Les séquences pseudo-chaotiques générées, sont utilisées pour former les clés dynamiques des différentes couches des crypto-systèmes proposés. Le moindre changement de la clé secrète du générateur engendre un changement quasi-chaotique dans les clés dynamiques et donc les crypto-systèmes proposés assurent la sensibilité à la clé secrète en émission/réception.

#### *2.5.1.6 Attaque sur l'implémentation*

Dans les crypto-systèmes basés sur la génération de séquences chaotiques, en réception, le déchiffrement d'un bloc nécessite d'abord la génération de tous les échantillons nécessaires au calcul des différentes clés dynamiques avant de procéder au déchiffrement. Ceci nécessite donc, de stocker ces échantillons pour une itération j, dans une mémoire protégée contre ce type d'attaque.

## 3.6 Outils de mesure des performances des couches des crypto-systèmes et évaluation de ces performances.

Dans ce paragraphe, nous introduisons les outils nécessaires à la quantification des performances des différentes couches, et nous évaluons expérimentalement les performances de chaque couche des crypto-systèmes proposés.

### 3.6.1 Couche de substitution

Une couche de substitution $n \times n$ d'un système de chiffrement par bloc itératif, doit être robuste vis-à-vis des cryptanalyses linéaire et différentielle. Une haute non-linéarité fournit une résistance contre les attaques linéaires et une distribution uniformément petite de la table XOR entrée-sortie fournit une résistance contre les attaques différentielles.
Les cinq critères principaux que doit vérifier une couche de substitution sont [4-8]:

- bijectivité,
- non linéarité,
- distribution équiprobable et petite de la table XOR entrée-sortie,
- critère d'avalanche strict (Strict Avalanche Criterion « SAC »),
- critère d'indépendance des bits (Bits Independence Criterion « BIC »).

Nous présentons, et évaluons quantitativement ces critères, puis nous comparons les résultats obtenus avec ceux obtenus par plusieurs techniques de construction de couche de substitution de la littérature. La substitution est réalisée sur les octets, donc $n = 8$.

*2.6.1.1 Bijectivité :*

Une couche de substitution est bijective si elle vérifie la relation suivante :

$$\sum_{j=1}^{2^n} \{A(i) \odot F(j)\} = 2^{n-1}, \forall i \in [0, n-1] \qquad (3.43)$$

où :

$$A(i) \odot F(j) = a(i)_1 \wedge F(j)_1 \oplus \cdots \oplus a(i)_n \wedge F(j)_n$$

Avec :

$F(j) \in [0, 2^n - 1]$ est un élément de $F = [F(1), F(2), \dots, F(2^n)]$,

$F(j) = [f(j)_1, f(j)_2, \dots, f(j)_n]$, et $A(i) = [a(i)_1, a(i)_2, \dots, a(i)_n] = 2^i,\ i = 0,1, \dots, n-1$ .

Nous pouvons vérifier autrement et plus facilement la bijectivité d'une couche de substitution, en calculant simplement le nombre des éléments différents de la sortie de cette couche de substitution $Y = F(X)$ pour une entrée variant de $X = 0, 1, \cdots 2^n - 1$. Si le nombre d'éléments est égal a $2^n$=256 (pour $n = 8$) alors, la couche de substitution est bijective, sinon, elle n'est pas bijective.

*2.6.1.2 Non linéarité*

La non linéarité est évaluée par deux approches, celle du spectre de Walsh et celle de la probabilité d'approximation linéaire.

**Approche de Walsh :**

Soit : $Y = F(X)$, la sortie d'une couche de substitution pour une entrée $X = [1, 2, \cdots, 256]$ (tel que $X(i) = i$). La non linéarité $NLF$ de $F(X)$ (une fonction Booléenne) peut être évaluée par le spectre de Walsh ($WS$) :

$$NLF = 2^{n-1} \times \left(1 - \left[2^{-n} \times Max_{i \in [1,\, 2^n]} |WS(i)|\right]\right) \qquad (3.44)$$

Où chaque élément du spectre de Walsh est calculé par :

$$WS(i) = \sum_{j=1}^{2^n} (-1)^{F(j) + i \odot j}\ , i \in [1,\ 2^n] \qquad (3.45)$$

Avec :

$F(j) \in [\ 0,\ 2^n - 1]$ et $i \odot j = i_1 \wedge j_1 \oplus i_2 \wedge j_2 \oplus \cdots \oplus i_n \wedge j_n$, avec

$i = [i_1, i_2, \dots, i_n]$, $j = [j_1, j_2, \dots, j_n]$, et $i_k, j_k$ représentent le kième bit de $i, j$

Ci-dessous nous donnons l'algorithme de calcul du spectre de Walsh et du $NLF$ dans la figure 3.22.

***Entr***é : $F, n$
***Sortie*** : $Walsh, NLF$
$--$ $n$ *est la précision de la couche de substitution, (nombre de bits)avec* $n = 8$
$--$ $F$ *est générée sous forme d'une ligne de taille* $2^n$ *éléments*
*Pour i allant de* 1 à $2^n$
$count = 0$
*Pour j allant de* 1 à $2^n$
$O = F(j)$
$temp = dec2bin\ (\{(i-1) \wedge (j-1)\}, n)$
$reg = 0$ ;
*Pour l allant de* 1 à $n$
$reg = reg \oplus temp(l)$
*Fin l*
$count = count + (-1)^{O+reg}$
*Fin j*
$Walsh(i) = count$
*Fin i*
$NLF = 2^{n-1} \times (1 - max(abs(Walsh))/2^n)$

Fig. 3.22: Algorithme de calcul du spectre de Walsh et du *NLF*

La fonction $dec2bin(z, n)$ retourne la représentation binaire de la valeur $z$ sur n bits.
Les valeurs de $NLF \geq 100$ indiquent que l'objectif de la non-linéarité est atteint.

**Approche de la probabilité d'approximation linéaire :**

La deuxième approche pour évaluer la non linéarité de la couche de substitution est basée sur le calcul de la probabilité d'approximation linéaire d'une fonction Booléenne.
Pour une couche de substitution $F: [\,0,\ 2^n - 1] \rightarrow [\,0,\ 2^n - 1]$, la probabilité linéaire approximative, notée $LP_F$ est définie par :

$$LP_F = Max_{\alpha,\beta \neq 0}[LP_F(\alpha, \beta)] = Max_{\alpha,\beta \neq 0}\left[\frac{\text{card}\{i/i \odot \alpha = F(i) \odot \beta\} - 2^{n-1}}{2^{n-1}}\right]^2 \qquad (3.46)$$

Où $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ , $\beta = [\beta_1, \beta_2, \ldots, \beta_n]$, $\alpha, \beta \in [\,1,\ 2^n - 1]$, card, dénote le cardinal, et $F(i) \odot \beta = f(i)_1 \wedge \beta_1 \oplus f(i)_2 \wedge \beta_2 \oplus \cdots \oplus f(i)_n \wedge \beta_n$ ; $i \odot \alpha = i_1 \wedge \alpha_1 \oplus i_2 \wedge \alpha_2 \oplus \cdots \oplus i_n \wedge \alpha_n$
Théoriquement :

$$LP_F(\alpha, \beta) \# \frac{1}{2^n - 1} \qquad (3.47)$$

Par ailleurs :

$$\sum_{\alpha=1}^{2^n-1} LP_F(\alpha, \beta) = 1\ , \forall\ \beta \quad \text{et} \quad \sum_{\beta=1}^{2^n-1} LP_F(\alpha, \beta) = 1\ , \forall\ \alpha$$

Ce qui implique que l'immunité d'une couche de substitution à la cryptanalyse linéaire dépend de l'uniformité de $LP_F(\alpha, \beta)$ dans $\alpha$ pour un $\beta$ fixé et vise versa.

Plus $LP_F$ est faible, plus la complexité de l'attaque linéaire est grande.
Par exemple, le $LP_F$ de la couche de substitution statique de l'algorithme AES est égal :
$\left[\frac{144-128}{128}\right]^2 = 2^{-6} = 0.015625$.
Notons que le $LP_F$ pour une couche de substitution dynamique varie de $2^{-4}$ à $2^{-6}$.
La relation entre le $NLF$ et le $LP_F$ est donnée par [Knudsen et al, 1998] :

$$LP_F \leq \left[\frac{2^{n-1} - \mathrm{NLF}}{2^{n-1}}\right]^2 \tag{3.48}$$

Ci-dessous nous donnons l'algorithme de calcul du $LP_F$ dans la figure 3.23.

***Entr***é : $F, n$
***Sortie*** : $LP_F$
*Pour* $\alpha$ *allant de* 1 *jusqu'*à $2^n - 1$
  *Pour* $\beta$ *allant de* 1 *jusqu'*à $2^n - 1$
    *Pour* $i$ *allant de* 0 *jusqu'*à $2^n - 1$
*Si* $i \odot \alpha = F(i) \odot \beta$
*incrémenter* $LPT(\alpha, \beta)$
*Fin Si*
    *Fin* $i$
  *Fin* $\beta$
*Fin* $\alpha$
$LP_F = \left(\frac{max(max(LPT)) - 2^{n-1}}{2^{n-1}}\right)^2$

Fig. 3.23: Algorithme de calcul de la probabilité d'approximation linéaire $LP_F$

### *2.6.1.3 Distribution équiprobable et petite de la table XOR entrée-sortie (Probabilité d'approximation différentielle de F)*

La cryptanalyse différentielle [Biham et Shamir,1991] est une attaque à texte en clair choisi pour trouver la clé secrète d'un crypto-système. La cryptanalyse différentielle exploite la haute probabilité des certaines occurrences se trouvant dans la différence de textes en clairs. La couche de substitution devrait idéalement avoir l'uniformité différentielle. C'est-à-dire, à une entrée différentielle $\Delta i_k$ donnée, doit correspondre une sortie différentielle $\Delta f_k = F(i) \oplus F(i \oplus \Delta i_k)$ unique. La probabilité d'approximation différentielle d'une couche de substitution $DP_F$, est une mesure de l'uniformité différentielle et est définie par :

$$DP_F = Max_{\Delta i \neq 0,\ \Delta F}\ [\ DP_F(\Delta i,\ \Delta F)] \tag{3.49}$$

$$DP_{F(\Delta i, \Delta f)} = \frac{\mathrm{card}\{i / F(i) \oplus F(i \oplus \Delta i) = \Delta f\}}{2^n} \tag{3.50}$$

Où $\Delta i \in [\ 1,\ 2^n - 1]$, $F(i),\ \Delta f \in [\ 0,\ 2^n - 1]$

$DP_F$ est la probabilité maximale de la sortie différentielle $\Delta f$, quand l'entrée différentielle est $\Delta i$.

Ci-dessous dans la figure 3.24, nous donnons l'algorithme de calcul de la probabilité d'approximation différentielle $DP_F$

***Entr***é : $F, n$
***Sortie*** : $DP_F$
*Pour* $\Delta i$ *allant de* 1 *jusqu'*à $2^n - 1$ *faire*
  *Pour* $i$ *allant de* 0 *jusqu'*à $2^n - 1$ *faire*
    $\Delta f = F(i) \oplus F(i \oplus \Delta i)$
      *incremente* $DDT(\Delta i, \Delta f)$
  *Fin Pour* $i$
*Fin pour* $\Delta i$
$DP_F = \dfrac{max(max(DDT))}{2^n}$

Fig. 3.24: Algorithme de calcul de la probabilité d'approximation différentielle $DP_F$

*2.6.1.4 Critère d'avalanche strict*

Un bon crypto-système doit réaliser l'effet d'avalanche, c.à.d, satisfaire le critère d'avalanche strict ($SAC$). Ce critère mesure l'influence du changement des bits d'entrée sur la sortie de la couche de substitution. Selon ce critère, si un bit de texte en clair change, en moyenne, la moitié des bits dans le texte chiffré changent. L'évaluation du $SAC$ d'une fonction Booléenne peut être réalisée par les trois étapes suivantes [Webster el al, 1986] :

Étape 1 : Produire un vecteur de texte en clair n-bits $i \in [0, 2^n - 1]$ et son texte substitué $F(i) \in [0, 2^n - 1]$ correspondant.

Étape 2 : Pour chaque $i \in [0, 2^n - 1]$ créer

- Un ensemble de $n$ vecteurs $I = \{i1, i2, \cdots, in\}^t$, tel que $i$ et $ik$ diffèrent seulement dans le bit $k$ : $i = [i_1, \cdots, i_k, \cdots, i_n]$ , $ik = [i_1, \cdots, \overline{i_k}, \cdots, i_n]$
- un ensemble de $n$ vecteurs : $\mathcal{F} = [F1, \cdots, Fk, \cdots, Fn]$ avec $Fk = F(ik)$
- un ensemble de $n$ vecteurs $V = [V1, \cdots, Vk, \cdots, Vn]$ avec $Vk = F(i) \oplus F(ik)$ .

Étape 3 : Appliquer la relation suivante : $a_{j,k} = a_{j,k} + v_{j,k}$, $j, k = 1, \cdots, n$

où $v_{j,k}$ est le jième bit du vecteur $Vk$ après sa conversion en binaire sur $n$ bits et $a_{j,k}$ est un élément de la matrice de dépendance $A$ de taille $n \times n$.

Initialement, tous les éléments de la matrice A sont nuls.

Chaque $a_{j,k}$ indique la relation entre le bit du texte en clair $k$ et le bit de texte substitué $j$.

La matrice $SAC$ est obtenue en divisant la matrice $A$ par $2^n$, soit par 256, pour n = 8.

Une couche de substitution est performante, si chaque élément de la matrice $SAC$ est proche de 0.5. Autrement dit, la valeur moyenne de la matrice $SAC$ est proche de 0.5 et l'écart type est proche de zéro.

Ci-dessous (voir figure 3.25), nous donnons l'algorithme de calcul du critère d'avalanche strict $SAC$

$\boldsymbol{Entreé}: F, n$
$\boldsymbol{Sortie}:\ a$
$Pour\ \ i\ allant\ de\ \ 1\ jusqu'à\ 2^n\ \ faire$
$\quad Pour\ k\ allant\ de\ 1\ jusqu'à\ n\ \ faire$
$\qquad \mathcal{F}(k) = F(i \oplus 2^{k-1})$
$\qquad V(k) = \ \ F(i) \oplus \mathcal{F}(k)$
$\qquad Vb(k, 1{:}n) = dec2bin(V(k)\ , n)$
$\quad Fin\ Pour\ k$
$a = a + \ Vb$
$Fin\ pour\ i$
$a = a/2^n$

Fig. 3.25: Algorithme de calcul du critère d'avalanche strict *SAC*

*2.6.1.5 Indépendance des bits produits en sortie « BIC »*

C'est une autre propriété désirable pour une couche de substitution. Ce critère mesure la dépendance entre les bits produits en sortie par une couche de substitution. Il est relié au critère *SAC*. En effet, pour évaluer le *BIC,* nous utilisons les mêmes étapes de l'évaluation du $SAC$ à part la relation suivante de l'étape 2 : $V(j,k) = \mathcal{F}(j) \oplus \mathcal{F}(k)$ pour $j, k \in \{1, 2, \dots, n\}\ et\ j \neq k$ et la relation suivante de l'étape 3 : $b_{j,k} = b_{j,k} + d_{j,k}, \quad j, k = 1, \cdots, n,$ où $d_{j,k}$ est la distance de Hamming $V(j,k)$ après sa conversion en binaire sur $n\ bits, et\ \ b_{j,k}$ est un élément de la matrice de dépendance $B$ de taille $n \times n$.

Initialement, tous les éléments de la matrice $B$ sont nuls.
Chaque $b_{j,k}$ indique la relation entre le bit du texte substitué $k$ et le bit de texte substitué $j$.
La matrice *BIC* est obtenue en divisant la matrice $B$ par $2^n$, soit par 256, car $n = 8$.
Une couche de substitution est performante, si chaque élément de la matrice *BIC,* à part la diagonale (qui est vide) est proche de 0.5. Autrement dit, la valeur moyenne de la matrice *BIC* est proche de 0.5 et l'écart type est proche de zéro.

Ci-dessous (voir figure 3.26), nous donnons l'algorithme de calcul de l'indépendance des bits produits en sortie *BIC.*

$\boldsymbol{Entreé}: F, n$
$\boldsymbol{Sortie}: b$
$Pour\ \ i\ allant\ de\ 1\ jusqu'à\ 2^n\ faire$
$\quad Pour\ j\ allant\ de\ 1\ jusqu'à\ n\ \ faire$
$\mathcal{F}(j) = F(i \oplus 2^{j-1})$
$\quad Fin\ Pour\ j$

$\quad Pour\ j\ allant\ de\ 1\ jusqu'à\ n\ faire$
$\qquad Pour\ k\ allant\ de\ 1\ jusqu'à\ n\ \ faire$
$\qquad\quad Si\ j \neq k$
$\qquad\qquad V(j,k) = \ \mathcal{F}(j) \oplus \mathcal{F}(k)$
$\qquad\qquad d(j,k) = sum(dec2bin(V(j,k)\ , n))$
$\qquad\quad Fin\ \ Si$
$\qquad Fin\ Pour\ k$
$\quad Fin\ Pourj$
$b = b + d$
$Fin\ pour\ i$
$b = b/(n \times 2^n)$

Fig. 3.26: Algorithme de calcul de l'Indépendance des bits produits en sortie *BIC*

### 3.6.2 Evaluation et analyse des performances des différentes cartes chaotiques.

#### *2.6.2.1 Performances de la carte Skew tent utilisant des paramètres de contrôle fixes en tant que couches de substitution et de permutation*

Dans ce paragraphe, nous pointons les points faibles de la carte Skew tent utilisant des paramètres de contrôle fixes, à savoir : coefficient de corrélation grand, non-linéarité faible, faible période. A ce propos, nous mesurons la nonlineairité *NLF*, le coefficient de corrélation (défini plus loin dans le paragraphe sur la sensibilité de la clé dynamique de la couche de substitution) et la périodicité.
Ensuite, nous proposons les solutions adéquates pour éliminer ces faiblesses.
Dans le cas de l'utilisation de la carte skew tent en tant que couche de substitution ($Q = 2^8 = 256$) le coefficient de corrélation $\rho$ est calculé pour chaque paramètre $a \in [1, 256]$, en fonction du nombre d'itération $rs$. Le calcul d'une valeur de $\rho$ est fait comme suit :
Pour un vecteur d'entrée $X = [1, 2, \cdots, 256]$ (variation linéaire de l'entrée $X(i) = i$, le pire des cas comme vecteur d'entrée) nous évaluons la fonction de substitution $F(X)$. Puis à partir de $F(X)$, nous formons deux fonctions $F1(X)$ et $F2(X)$ de taille 255 éléments chacune, telles que $F1(X) = \ F(X)$, pour les éléments $[1, 2, \cdots, 255]$ et $F2(X) = \ F(X)$, pour les éléments $[2, \cdots, 256]$. Enfin, nous calculons le coefficient de corrélation entre $F1(X)$ et $F2(X)$.
Dans la figure 3.27, nous montrons, pour chaque paramètre $a \in [1, 256]$ de la carte Skew tent, la variation du coefficient de corrélation $\rho$ en code de couleur en fonction du nombre d'itérations $rs$.

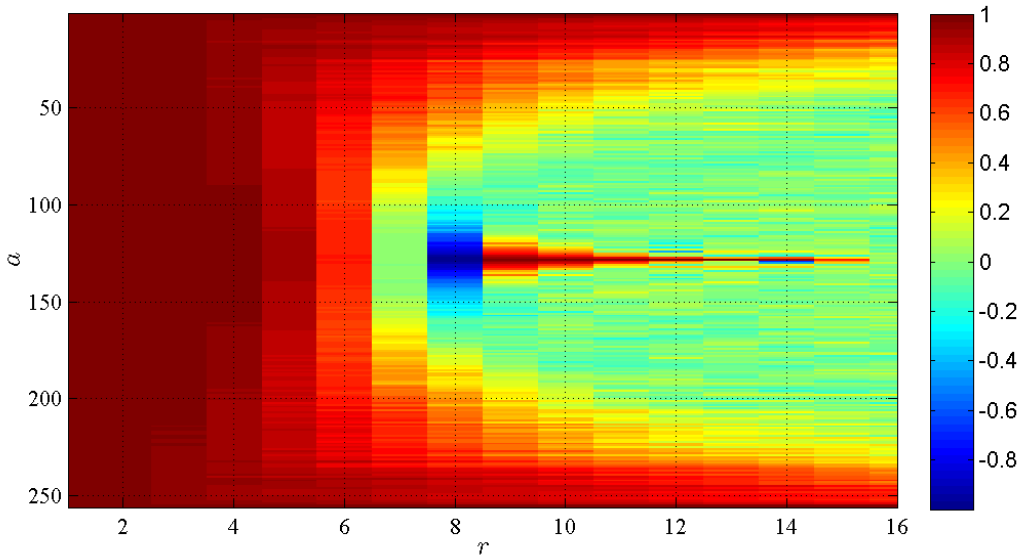



Fig. 3.27: Variation de ρ en code de couleur, en fonction de ***rs*** pour chaque valeur du paramètre a ∈ [1,256]

D'après cette figure, nous pouvons faire les remarques suivantes :

1- quelque soit la valeur du paramètre de contrôle $a \in [1, 256]$, le coefficient de corrélation $\rho$ est grand (c.à.d, un comportement linéaire de la carte Skew tent) pour le nombre d'itérations $rs$ inferieur ou égale a 6.

2- Quelle que soit la valeur de $rs$ (incluant les $rs > 6$), le coefficient de corrélation est grand pour les paramètres de contrôle proches des intervalles des extrémités [1, 32], [224, 256] et aussi pour certaines valeurs des paramètres de contrôle proches de l'intervalle du milieu [120,130]. Cependant pour $rs > 6$, certaines valeurs des paramètres de contrôle, qui sont proches de l'intervalle du milieu, permettent de converger plus rapidement vers les zones non-linéaires (donc de faibles coefficients de corrélation).

3- A partir de $rs > 8$, les paramètres de contrôle qui sont proches de la valeur 100 et de la valeur 150 conduisent à la non-linéarité et donnent des résultats symétriques de $\rho$ par rapport à la valeur zéro.

Dans la figure 3.28, nous montrons, pour chaque paramètre $a \in [1, 256]$ de la carte Skew tent, la variation de la non linéarité *NLF* en code de couleur en fonction du nombre d'itérations $rs$. La fonction est considérée non-linéaire pour les valeurs $NLF \geq 100$.
Nous pouvons faire les remarques suivantes :

1- Quelle que soit la valeur du paramètre de contrôle $a \in [1, 256]$, la non-linéarité *NLF* est faible (c.à.d, un comportement linéaire de la carte Skew tent) pour le nombre d'itérations $rs$ inferieur ou égale a 6.

2- Quelle que soit la valeur de $rs$ (incluant les $rs > 6$), la non-linéarité *NLF* est faible pour les paramètres de contrôle proches des intervalles des extrémités [1, 32], [224, 256] et aussi pour certaines valeurs des paramètres de contrôle proches de l'intervalle du milieu [120,130]. En dehors de ces intervalles, nous obtenons un bon *NLF* pour les $rs > 8$.

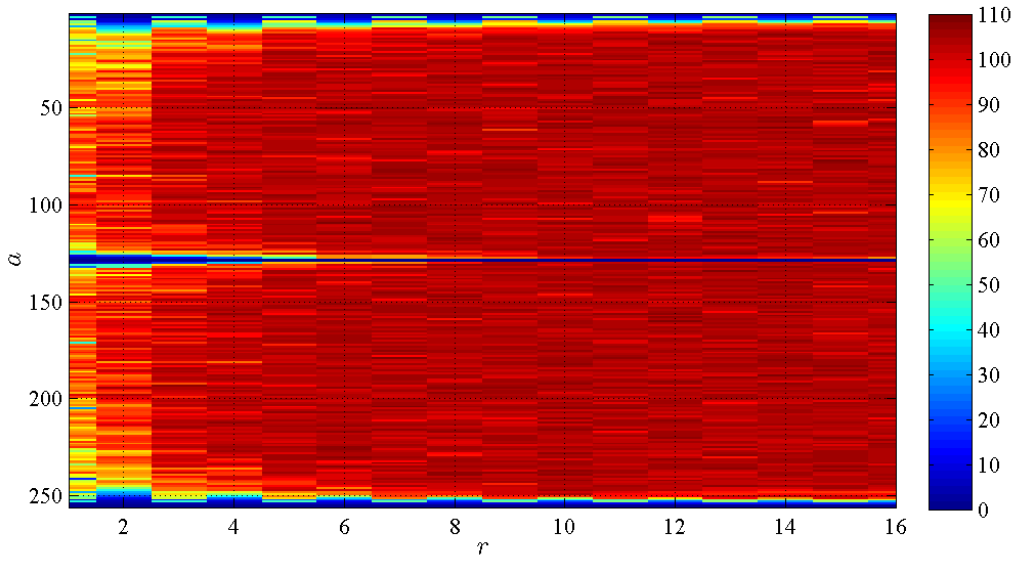



Fig. 3.28: Variation du *NLF* en code de couleur, en fonction de *rs* pour chaque valeur du paramètre $a \in [1, 256]$

Nous présentons dans la figure 3.29, la valeur du *NLF* en fonction des paramètres de contrôle $a$, pour $rs = 6$ (une coupe de la figure 3.28). Cette figure montre clairement, que seulement quelques valeurs du paramètre de contrôle $a$, conduisent à une non-linéarité acceptable. Donc, $rs$ doit être supérieur à 6.

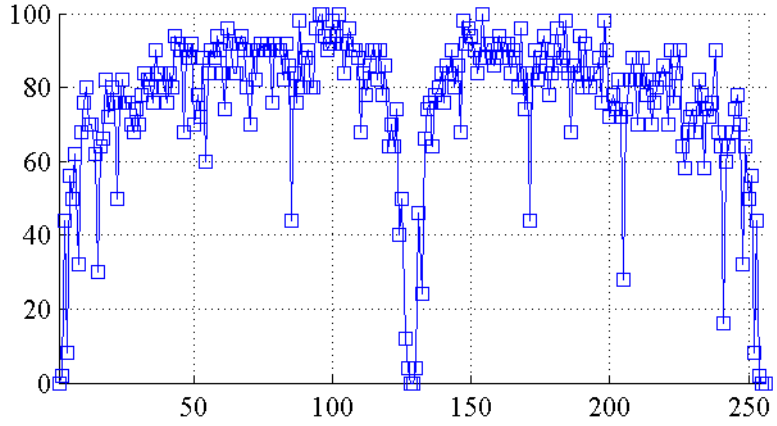



Fig. 3.29: Variation du *NLF* en fonction des paramètres de contrôle *a* pour *rs=6*

Ci-dessous dans la figure 3.30, nous donnons le pseudo-code réalisant les tests des figures 3.27, 3.28

***Entrée*** *: $n, rs$*
***Sortie*** *:NLF et $\rho$*
$Q = 2^n$
*//Remplir le tableau initial, avec n=8*
*Pour $it$ allant de 1 à $Q$*
$Temp(it) = it$
*Fin $it$*

*-- procédure de calcul du coefficient de corrélation linéaire et de la non linéarité, --de la fonction de substitution produite en fonction de paramètre de contrôle $a$ --et du nombre d'itération $rs$ .*

*Pour $a$ allant de 1 à $Q$*
$X = temp$
*Pour $it$ allant de 1 à $rs$*
$X = supboxvector(X, Q, a, 1);$
$NLF(a, it) = Nonlineairity(X)$
$\rho(a, it) = corrcoef\left(X(1: length(X) - 1), X(2: length(X))\right)$
*Fin $it$*
*Fin $a$*

Fig. 3.30: Pseudo-code réalisant les tests des figures 3.27 et 3.28

### *2.6.2.2 Périodicité de la carte Skew tent :*

Dans ce paragraphe, nous mesurons la périodicité de la carte Skew tent pour chaque paramètre $a \in [1, 2, \cdots, 256]$. A cet effet, pour chaque paramètre de contrôle, nous évaluons la fonction de substitution $F(X)$ pour un vecteur d'entrée $X = [1, 2, \cdots, 256]$ (variation linéaire de l'entrée $X(i) = i$, le pire des cas comme vecteur d'entrée) et nous testons la condition $F(X) = X$.

Si cette condition est non satisfaite, alors les valeurs obtenues de $F(X)$ servent comme entrée de la fonction $F(X)$ pour la prochaine itération et ainsi de suite jusqu'à satisfaction de la condition en question. Dans ce cas, le nombre d'itérations mis en œuvre est la périodicité de la carte.

Sur la figure 3.31, nous présentons les résultats obtenus pour la mesure de la périodicité donnée en échelle logarithmique. Nous remarquons que 10% des valeurs du paramètre de contrôle $a$, conduisent à une périodicité supérieure à $10^7$. Pour le reste des valeurs, la périodicité varie entre $10^3$ et $10^7$. Notons aussi la présence d'une périodicité maximale égale pratiquement à $10^{10}$ pour $a = 96$ et d'une périodicité égale à 1 (point fixe), pour $a = 256$.

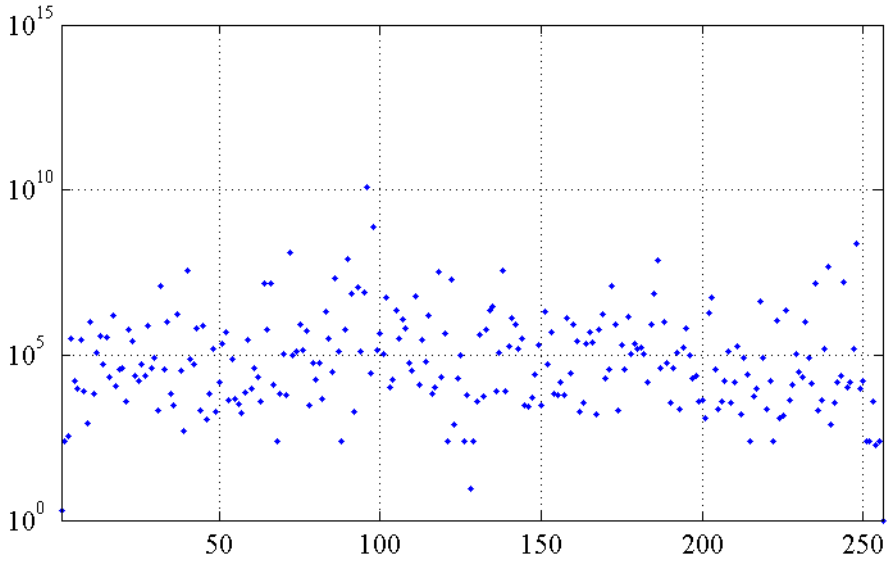



Fig. 3.31: Variation de la périodicité en fonction du paramètre de contrôle *a*

Ci-dessous dans la figure 3.32, nous donnons le pseudo-code réalisant la mesure de la périodicité.

**Entrée** : n
**Sortie** : $P$
$Q = 2^n$
--$P$ est un tableau de taille $Q$ dont chaque élément contient la périodicité du paramètre de contrôle correspondant à son indice.
-- Remplir le tableau initial
$Pour\ it$ allant de 1 à $Q$
$Temp(it) = it$
$fin\ \ it$
--Procédure de calcul de la périodicité de chaque paramètre de
--contrôle $a$
$Pour\ a$ allant de 1 à $Q$
$X = temp$
Pour $io$ allant de 1 à l'infini
$X = supboxvector(X, Q, a, 1);$
Si $(X = Temp)$
$P(a) = io$
$Sortie$ de la boucle $io$
$Fin\ Si$
$Fin\ io$
$Fin\ a$

Fig. 3.32: Pseudo-code pour la mesure de la périodicité en fonction du paramètre de contrôle

### 3.6.3 Performances de la carte skew tent avec paramètres de contrôle dynamique en tant que couches de substitution et de permutation

Nous montrons dans ce paragraphe que l'utilisation des paramètres de contrôle dynamiques, permet de remédier aux faiblesses (coefficient de corrélation grand, non-linéarité faible, périodicité petite) de la carte avec paramètres de contrôle fixes, d'augmenter la taille de la clé dynamique et ainsi de renforcer le degré de la sécurité.

#### *2.6.3.1 Performances en termes de NLF, $\boldsymbol{DP_F}$ et $\boldsymbol{\rho}$ : Choix de la valeur adéquate de $\boldsymbol{na}$ et de $\boldsymbol{rs}$*

Deux cas de figures sont pris en compte. Dans le premier cas, nous fixons $rs = 1$, et nous cherchons le nombre $na$ de paramètres de contrôle nécessaire pour réaliser les performances exigées. Dans le deuxième cas de figure, nous fixons $na = 4$ ou 8, (pour avoir une taille de la clé dynamique de 32 ou 64 bits) et nous cherchons le nombre d'itérations $rs$ nécessaires permettant d'atteindre les objectifs souhaités.

**Premier cas de figure** :
Nous avons réalisé les opérations suivantes :

D'abord, nous avons généré 1000 clés dynamiques, chacune de façon aléatoire et uniforme. Pour un même vecteur entrée $X = [1, 2, \cdots, 256]$ (tel que $X(i) = i$), et pour chaque paramètre de la clé $Ks_j = [\, a_{j,1} | a_{j,2} | \cdots a_{j,na}]$, avec $na = 16$, nous avons calculé la fonction de substitution $F(X)$ de façon itérative. C'est-à-dire le résultat du calcul de $F(X)$ pour le premier paramètre $a_{j,1}$ est utilisé comme vecteur d'entrée pour le calcul de $F(X)$ utilisant le deuxième paramètre $a_{j,2}$ et ainsi de suite, jusqu'à épuisement du dernier paramètre, soit $a_{j,na}$. Sur chaque fonction $F(X)$ calculée, nous appliquons immédiatement les tests de *NLF*, $DP_F$ et $\rho$ avant de passer au calcul de la fonction $F(X)$ suivante. Au final, nous obtenons 3 tableaux de 1000 lignes et de 16 colonnes contenant les résultats du *NLF*, $DP_F$ et $\rho$ respectivement.

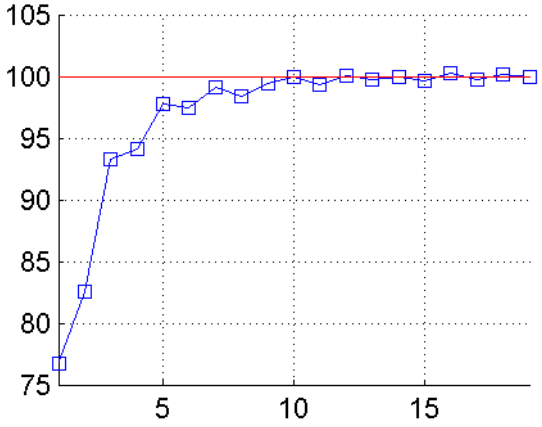



Fig. 3.33: Evolution de la moyenne *NLF* en fonction des paramètres de contrôle

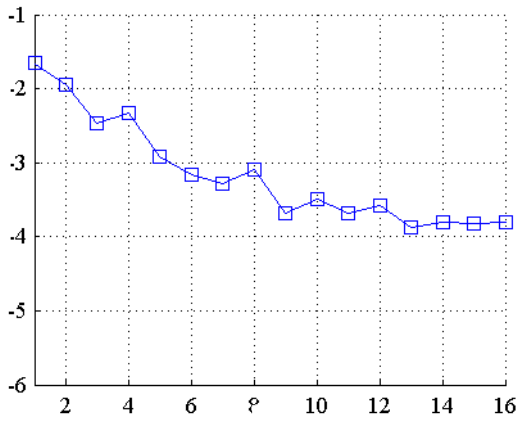



Fig. 3.34: Evolution de la moyenne *DPF* en fonction des paramètres de contrôle

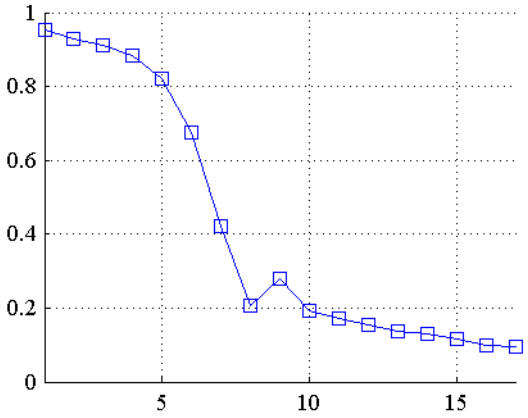



Fig. 3.35: Evolution de la moyenne *p* en fonction des paramètres de contrôle

Dans les figures 3.33, 3.34 et 3.35, nous avons tracé, la valeur moyenne (sur les 1000 clés), du coefficient *NLF*, du $DP_F$ et de $\rho$ respectivement, en fonction des paramètres de

contrôle. L'échelle ordonnée de la figure 3.34 est en $log_2$, par exemple -3 sur cet échelle, représente en fait la valeur $2^{-3}$.

D'après les trois figures précédentes, nous pouvons remarquer que les objectifs fixés sont atteints à partir de l'utilisation de 10 paramètres de contrôle.

**Deuxième cas de figure** :

Ici, nous fixons le nombre de paramètres de contrôle $na$ à 4. Comme dans le premier cas de figure, d'abord, nous générons 1000 clés dynamiques de taille $na \times n$ bits, de façon aléatoire et uniforme. Pour un même vecteur entrée $X = [1, 2, \cdots, 256]$ (tel que $X(i) = i$), le calcul $F(X)$ en fonction de $rs$ se fait comme suit :

Pour $rs = 1$, on calcule la fonction de substitution $F(X)$ de façon itérative sur les quatre paramètres comme dans le premier cas de figure. Pour $rs = 2$, on recommence le même calcul itératif de $F(X)$, mais cette fois ci, chaque paramètre est itéré 2 fois avant de passer au paramètre suivant. Ce procédé est répété pour la suite des valeurs de $rs$, jusqu'à la dernière valeur $rs = 16$. Aussi, comme dans le premier cas de figure, sur chaque fonction $F(X)$ calculée, nous appliquons immédiatement les testes de *NLF*, $DP_F$ et $\rho$ avant de passer au calcul de la fonction $F(X)$ suivante. Au final, nous obtenons 3 tableaux de 1000 lignes et de 8 colonnes contenant les résultats du *NLF*, $DP_F$ et $\rho$ respectivement.

Dans les figures 3.36, 3.37 et 3.38, nous avons tracé, en fonction de $rs$, la valeur moyenne (sur les 1000 clés), du coefficient *NLF*, du $DP_F$ et de $\rho$ respectivement. L'échelle ordonnée de la figure 3.37 est en $log_2$.

D'après les trois figures 3.36, 3.37 et 3.38, nous pouvons remarquer que les objectifs fixés sont atteints à partir de $rs = 4$.

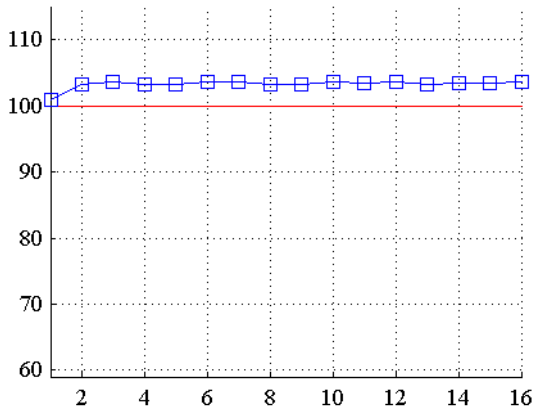



Fig. 3.36: Evolution de la moyenne *NLF* en fonction de *rs*

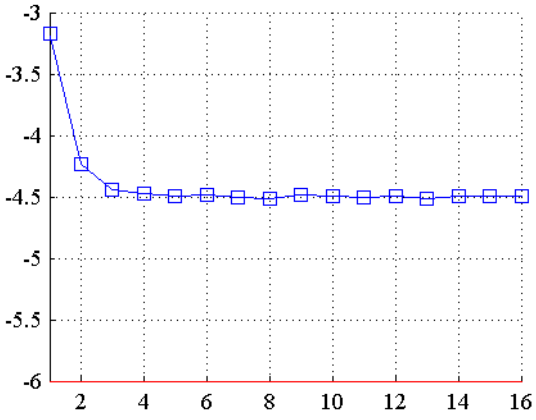



Fig. 3.37: Evolution de la moyenne *DPF* en fonction de *rs*

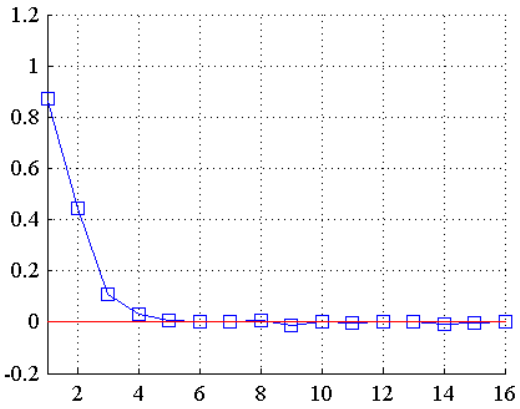



Fig. 3.38: Evolution de la moyenne $\rho$ en fonction de *rs*

Ci-dessous dans la figure 3.39 nous donnons le pseudo-code réalisant les tests des figures 3.33-3.38.

**Entrée** : $n, na, nk, rs$ // *nk est le nombre des clés aléatoires ; n=8*
**Sortie** : $NLF$ , $DP_F$ *et* $\rho$
$Q = 2^n$
//*Remplir le tableau initial*
$Pour\ \ it$ *allant de 1 à* $\ Q$
$Temp(it) = it$
$Fin\ \ it$
//*Création de* $nk$ *clés dynamiques, chacune de taille :* $\ na \times n$ *bits*
$Pour\ \ itk$ *allant de 1 à* $\ nk$
$Pour\ \ ita$ *allant de 1 à* $\ na$
$a(itk, ita) = \lfloor rand \times 256 \rfloor$
$Fin\ \ itk$
$Fin\ \ ita$
*-- Calcul de :* $NLF, DP_F$ *et* $\rho$ *produite en fonction de paramètre de contrôle* $a$ *et du nombre d'itération* $rs$ *.*
---------- Premier cas de figure ----------
$Pour\ itk$ *allant de 1 à* $\ nk$
$X = temp$
$Pour\ ita$ *allant de 1 à* $\ na$
$X = supboxvector(X, Q, a, 1);$
$NLF\_Mat(nk, ita) = Nonlineairity(X)$
$DP_F\ \_Mat(nk, ita) = probabilité\_differentielle(X)$
$\rho\_Mat(nk, ita) = corrcoef\left(X(1: length(X) - 1), X(2: length(X))\right)$
$Fin\ \ ita$
$Fin\ \ itk$
----------Deuxième cas de figure ----------
$Pour\ itk$ *allant de 1 à* $\ nk$
$X = temp$
$Pour\ itr$ *allant de 1 à* $\ rs$
$Pour\ ita$ *allant de 1 à* $\ na$
$X = supboxvector(X, Q, a, itr);$
$Fin\ \ ita$
$NLF\_Mat(nk, itr) = Nonlineairity(X)$
$DP_F\ \_Mat(nk, itr) = probabilité\_differentielle(X)$
$\rho\_Mat(nk, itr) = corrcoef\left(X(1: length(X) - 1), X(2: length(X))\right)$
$Fin\ \ itr$
$Fin\ \ itk$
---
$NLF = mean(\ NLF\_Mat)$
$DP_F = mean(\ DP_F\_Mat)$
$\rho\ = mean(\rho\_Mat)$

Fig. 3.39: Pseudo-code réalisant les tests des figures 3.33-3.38

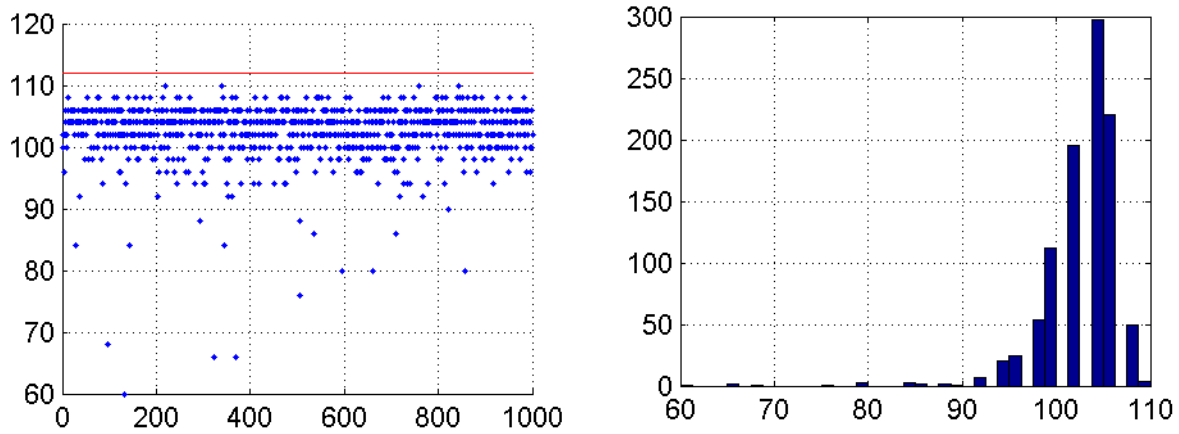
### 3.6.4 Performances de la couche de substitution selon les cinq critères énoncés:

Les différents tests réalisés précédemment, nous ont permis de choisir le bon couple $\{rs = 6, na = 4\}$ permettant d'avoir les meilleures performances de la couche de substitution. Partant de ce couple de valeurs, nous évaluons ici les cinq critères et nous comparons les résultats obtenus avec ceux obtenus par différentes couches de substitution de la littérature.

A ce propos, nous générons 1000 clés dynamiques $Ks_j$ de taille $na \times n$ bits (avec $na = 4, \text{et } n = 8$), de façon aléatoire et uniforme. Pour chaque clé, et pour un même vecteur d'entrée $X = [1, 2, \cdots, 256]$ (tel que $X(i) = i$), le calcul de la fonction de substitution $F(X)$ se fait comme suit :
Pour le premier paramètre de contrôle, on calcule la fonction de substitution $F(X)$ de façon itérative $rs$ fois. Le résultat obtenu de $F(X)$ est utilisé comme vecteur d'entrée permettant d'itérer à nouveau $F(X)$ $rs$ fois, pour le deuxième paramètre de contrôle, et ainsi de suite, jusqu'à épuisement du dernier paramètre.
Sur chaque fonction $F(X)$ calculée, nous calculons immédiatement les cinq critères *NLF*, $LP_F$, $DP_F, SAC, \text{et } BIC$ avant de passer au calcul de la fonction $F(X)$ suivante. Au final, nous obtenons 5 tableaux de 1000 lignes et de 1 colonne contenant les cinq critères.

#### *2.6.4.1 Résultats du critère NLF*

Dans les figures 3.40-a, 3.40-b, nous présentons l'évolution du *NLF* en fonction de la clé dynamique et sa distribution. Sur ces figures, nous remarquons que 93.2% de clés dynamiques utilisées (donc de couches de substitution) achèvent des non-linéarités $NLF \geq 96$. En Matlab ce calcul se fait par : $(length(find(NLF >= 96))/length(NLF)) \times 100$.
Ceci correspondant à une probabilité d'approximation linéaire $LP_F \leq 2^{-4}$.
Nous observons aussi que 6.8% des clés dynamiques aboutissent à des non-linéarités faibles $NLF < 96$, et donc un $LP_F > 2^{-4}$. Notons que toutes les techniques de création de couches de substitution dynamiques possèdent un certain nombre de clés dites faibles.
Les maximum, minimum, moyenne et écart type obtenus de la non linéarité sont : 110, 60, 102.62, et 4.5 respectivement. La probabilité d'approximation linéaire moyenne est de $LP_F = 2^{-4.7}$. Ces résultats nous permettent de conclure qu'en moyenne, l'objectif de la non-linéarité est atteint.
Rappelons au passage que la couche de substitution statique de l'algorithme AES a été conçue pour obtenir $Max_{j \in [0,\, 2^n - 1]} |WS(j)| = 32$, soit une non-linéarité maximale $NLF = 112$, ce qui correspondant à un $LP_F = 2^{-6}$.

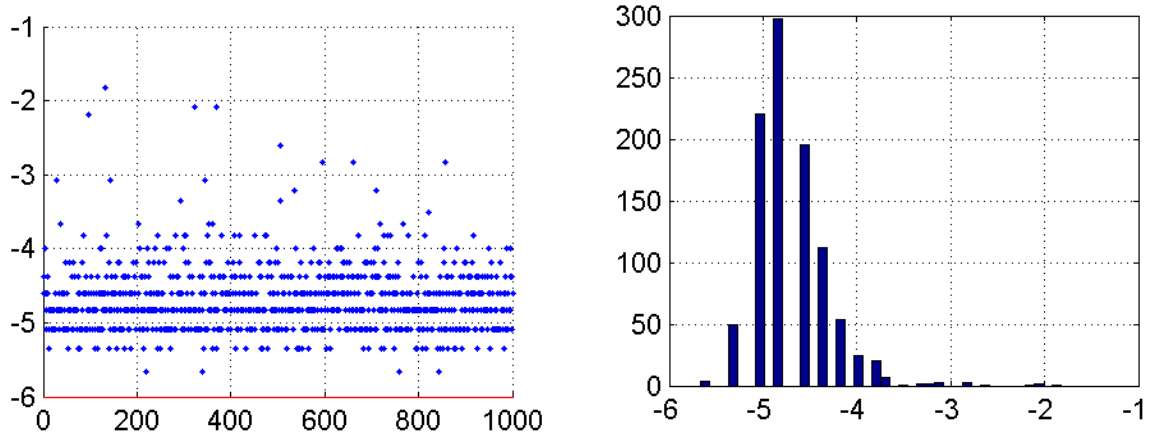



(a) (b)

Fig. 3.40:-a. Evolution du *NLF* en fonction de la clé dynamique, -b. Distribution du *NLF*

### *2.6.4.2 Résultats du critère* $\boldsymbol{LP_F}$

Dans les figures 3.41-a, 3.41-b, nous présentons l'évolution du $LP_F$ en fonction de la clé dynamique et sa distribution (l'échelle ordonnée de la figure 20-a, et l'échelle abscisse de la figure 20-b sont des échelles en $log_2$). Nous pouvons faire les remarques suivantes :
93.2% de clés dynamiques utilisées donnent des $LP_F \leq 2^{-4}$, 27.4% des couches de substitution possèdent des $LP_F$ tel que $2^{-6} \leq LP_F \leq 2^{-5}$ et seulement 0.008% des couches de substitution possèdent des $LP_F > 2^{-3}$.
Par ailleurs, les maximum, minimum, moyenne et écart-type obtenus du $LP_F$ sont : $2^{-1.82}$, $2^{-5.66}$, $2^{-4.7042}$ et 0.0189.
Tous ces résultats sont en cohérence totale avec les résultats obtenus par le critère *NLF*.




(a) (b)

Fig. 3.41: -a. Evolution du *LPF* en fonction de la clé dynamique, -b) Distribution du *LPF*

Les résultats obtenus par les critères *NLF* et $LP_F$ montrent clairement que la majorité des couches de substitution (93.2%) sont résistantes vis-à-vis de l'attaque linéaire.

### 2.6.4.3 *Résultats du critère $DP_F$*

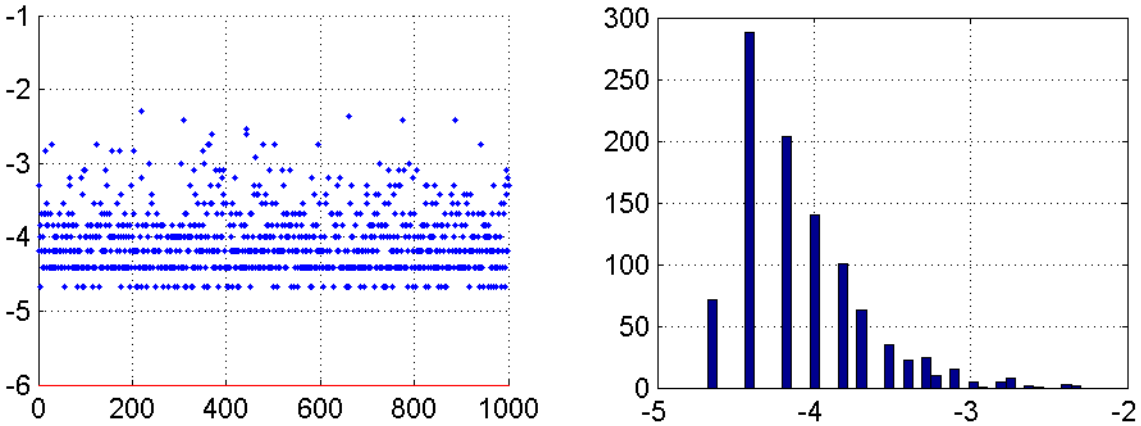
La probabilité d'approximation différentielle $DP_F$ a un comportement similaire à la probabilité d'approximation linéaire et possède les mêmes ordres de grandeur en fonction des clés dynamiques. Rappelons aussi, que la couche de substitution statique de l'algorithme AES possède un $DP_F = LP_F = 2^{-6}$.

Dans les figures 3.42-a, 3.42-b, nous présentons l'évolution du $DP_F$ en fonction de la clé dynamique et sa distribution (l'échelle ordonnée de la figure 3.42-a, et l'échelle abscisse de la figure 3.42-b sont des échelles en $log_2$). Nous pouvons faire les remarques suivantes :

90.2% de clés dynamiques utilisées donnent des $DP_F \leq 2^{-3.5}$, 70.3% des couches de substitution possèdent des $DP_F$ tel que $2^{-5} \leq DP_F \leq 2^{-4}$ et seulement 0.022% des couches de substitution possèdent des $DP_F > 2^{-3}$.

Par ailleurs, les maximum, minimum, moyenne et écart-type obtenus du $DP_F$ sont : $2^{-2.23}$, $2^{-4.678}$, $2^{-4.067}$ et 0.0228.

Ces résultats montrent que la majorité des couches de substitution résiste à l'attaque différentielle.



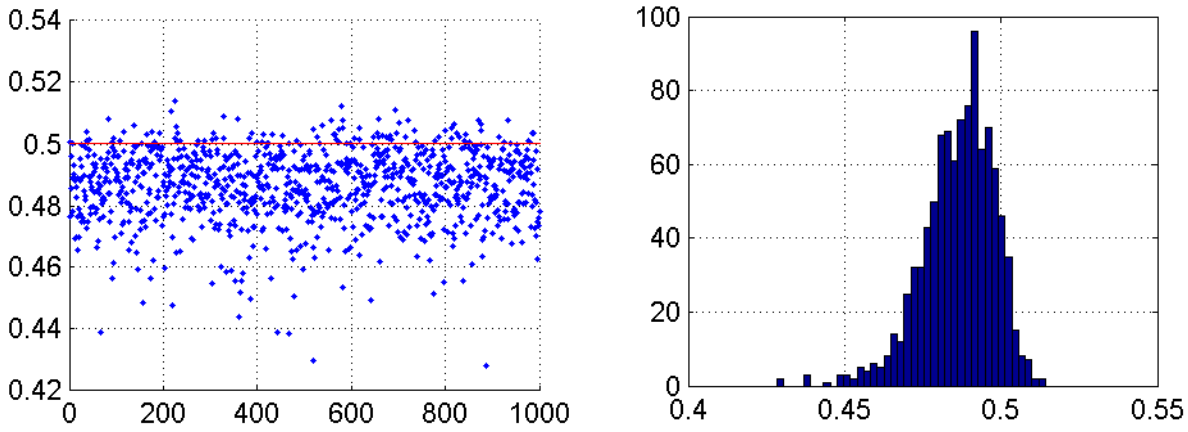

(a) (b)

Fig. 3.42:-a. Evolution du $DP_F$ en fonction de la clé dynamique, -b. Distribution du $DP_F$

### 2.6.4.4 *Résultats du critère SAC*

Dans les figures 3.43-a, 3.43-b, nous présentons l'évolution de la moyenne du $SAC$ en fonction de la clé dynamique et sa distribution. Nous pouvons faire les remarques suivantes : 97.2% de clés dynamiques utilisées donnent des valeurs moyennes des $SAC > 0.46$, 74. 90% des couches de substitution possèdent des moyennes $SAC$ tel que $0.48 \leq SAC \leq 0.5142$ et seulement 0.028% des couches de substitution possèdent des moyennes $SAC < 0.46$.

Par ailleurs, les maximum, minimum, moyenne et écart-type obtenus des $SAC$ moyennes sont : 0.5142, 0.4282, 0.48638 et 0.0117. La majorité des valeurs moyennes des $SAC$ sont autour la valeur optimale de 0.5.




(a) (b)

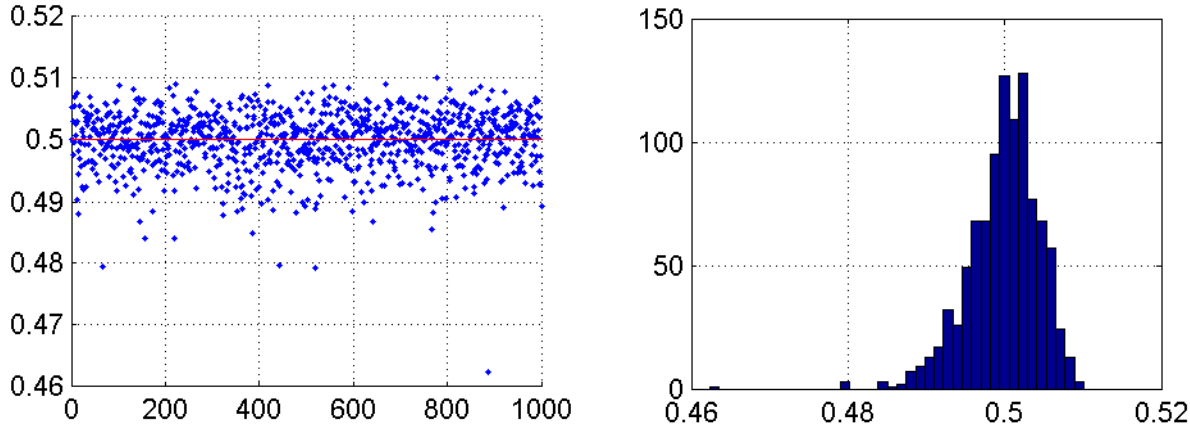
Fig. 3.43 :-a. Evolution de la moyenne des *SAC* -b. Distribution de la moyenne des *SAC* en fonction de la clé dynamique

### *2.6.4.5 Résultats du critère BIC*

Dans les figures 3.44-a, 3.44-b, nous présentons l'évolution de la moyenne du $BIC$ en fonction de la clé dynamique et sa distribution. Rappelons que la diagonale de la matrice *BIC* est vide. Nous pouvons faire les remarques suivantes :

97. 1% des couches de substitution possèdent des moyennes $BIC$ tel que $0.49 \leq BIC \leq 0.51$ et seulement 0.029% des couches de substitution possèdent des moyennes $BIC < 0.49$.

Par ailleurs, les maximum, minimum, moyenne et écart-type obtenus des $BIC$ moyennes sont : 0.5101, 0.4623, 0.4999 et 0.0046. La majorité des valeurs moyennes des *BIC* sont autour la valeur optimale de 0.5.




(a) (b)

Fig. 3.44:-a. Evolution de la moyenne des *BIC* -b. Distribution de la moyenne des *BIC* en fonction de la clé dynamique.

Les résultats des figures 3.43 et 3.44, montrent clairement que la majorité des couches de substitution satisfont les critères *SAC* et *BIC*. Donc, les couches de substitution en question résistent à l'attaque à texte clair/chiffré connu et choisi.

### 3.6.5 Sensibilité à la clé dynamique de la couche de substitution

Après validation des performances des couches de substitution selon les cinq critères de base, nous examinons ici la sensibilité de ces couches aux clés dynamiques afin de les valider.

La sensibilité est examinée selon deux critères :

- le premier critère est le coefficient de corrélation,
- le deuxième critère dépend de la distance de Hamming mesurée (en bits) entre deux couches de substitution résultats de deux clés qui diffèrent d'un seul bit en position $i$.

Le coefficient de corrélation permet de mesurer le degré de similarité entre ces deux couches de substitutions et la distance de Hamming permet de mesurer la différence en bits entre les deux couches.

Pour réaliser ces tests, nous utilisons 1000 clés dynamiques $Ks_j$, chacune de taille $l = na \times n$ bits (n=8 bits).

Pour chaque clé $Ks_j$, $j = 1,2, \ldots 1000$, et pour un même vecteur entrée $X = [1, 2, \cdots, 2^n]$ (tel que $X(i) = i$), nous changeons de façon aléatoire un bit de $Ks_j$ afin de former $Ks_j'$ comme suit $\Delta Ks_j = 2^k$ (bit à changer se trouve en position $k$) et $Ks_j' = Ks_j \oplus \Delta Ks_j = Ks_j \oplus 2^k$, puis nous calculons :

$F_{Ks_j} = \left\{F_{Ks_j}(1), F_{Ks_j}(2), \cdots, F_{Ks_j}(2^n)\right\}$ et $F_{Ks_j}{}' = \left\{F_{Ks_j}{}'(1), F_{Ks_j}{}'(2), \cdots, F_{Ks_j}{}'(2^n)\right\}$

Ensuite, à partir des $F_{Ks_j}$ et $F'_{Ks_j}$, nous calculons les deux critères en questions.

**Critère de coefficient de corrélation $\rho$ entre $F_{Ks_j}$ et $F'_{Ks_j}$**

Par la suite et afin de simplifier la notation, remplaçons $F_{Ks_j}$ et $F'_{Ks_j}$ par $F$ et $F'$. Le coefficient de corrélation entre $F$ et $F'$ est donné par l'équation suivante :

$$\rho_{F,F'} = \frac{cov(F,\ F')}{\sqrt{D(F)}\sqrt{D(F')}} \tag{3.51}$$

Où $cov(F,\ F')$ est la covariance entre $F$ et $F'$ définie par :

$$cov(F,\ F') = E([F - E(F)][F' - E(F')]) = \frac{1}{2^n}\sum_{i=1}^{2^n}[F(i) - E(F)][F'(i) - E(F')]$$

et $E(F)$ et $E(F')$ sont les moyennes de $F$ et $F'$ données par :

$$E(F) = \frac{1}{2^n}\sum_{i=1}^{2^n}F(i) \quad , \quad E(F') = \frac{1}{2^n}\sum_{i=1}^{2^n}F'(i)$$

avec $D(F)$ et $D(F')$ sont les variances de $F$ et $F'$ données par :

$$D(F) = \frac{1}{2^n}\sum_{i=1}^{2^n}[F(i) - E(F)]^2, \ D(F') = \frac{1}{2^n}\sum_{i=1}^{2^n}[F'(i) - E(F')]^2$$

Le coefficient de corrélation est compris entre -1 et 1. Un coefficient de corrélation proche de $\pm 1$, signifie la présence d'une forte corrélation linéaire entre les deux fonctions (ou vecteurs)

en jeu, un coefficient de corrélation proche de zéro, signifie pratiquement l'absence de corrélation entre le deux fonctions correspondantes.

### *2.6.5.1 Critère dépendant de la distance de Hamming*

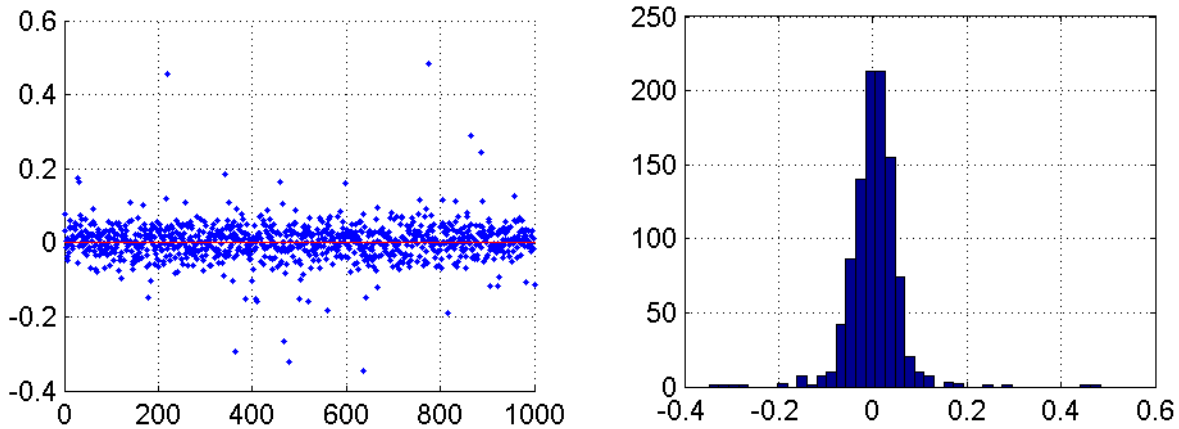
Il s'agit de calculer la distance de Hamming en bits entre les deux couches de substitution $F\ et\ F'$. Pour cela, on convertit en bit les $2^n$ éléments de chaque fonction. Soit $Fb_j\ et\ Fb_j{}'$ les deux vecteurs lignes obtenus de taille $n \times 2^n$ bits chacun.
La distance Hamming en bits entre les deux couches de substitution $Fb_j\ et\ Fb_j{}'$ est :

$$DH(Fb_j\ , Fb_j{}') = \sum_{m=1}^{n\times 2^n} Fb_j(\,m\,) \oplus Fb_j{}'(\,m\,) \qquad (3.52)$$

Le pourcentage de différence en bits entre les deux couches de substitution $Fb_j\ et\ Fb_j{}'$ est :

$$PDH(Fb_j\ , Fb_j{}') = \frac{d_H(Fb_j\ , Fb_j{}')}{n \times 2^n} \times 100 \qquad (3.53)$$

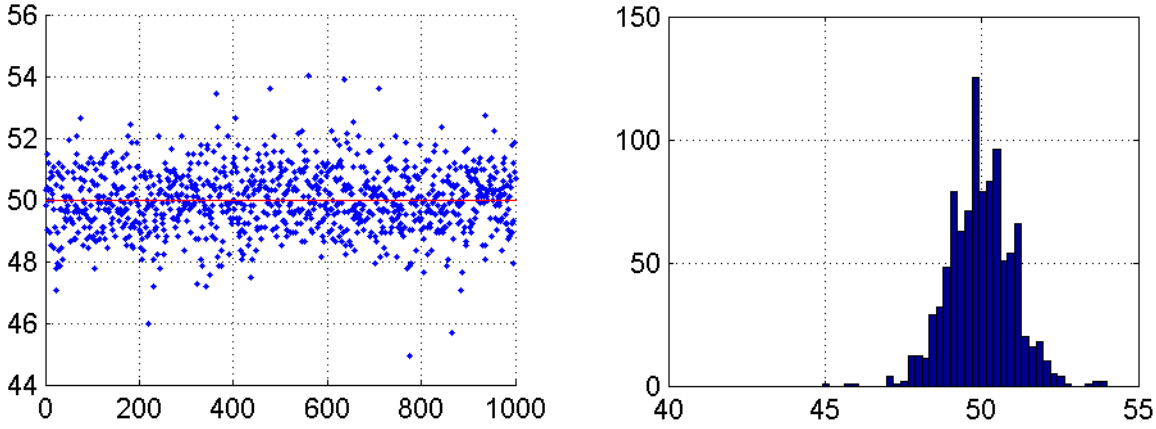
Dans les figures 3.45-a, 3.45-b, nous présentons l'évolution du coefficient de corrélation $\rho_{F,F'}$ en fonction de la clé dynamique et sa distribution. Nous pouvons faire les remarques suivantes :
95.9% des couches de substitution possèdent des $\rho_{F,F'}$ tel que $\rho_{F,F'} \leq \pm 0.1$ et seulement 0.008% des couches de substitution possèdent des $\rho_{F,F'} \geq \pm 0.2$.
Par ailleurs, les maximum, minimum, moyenne et écart-type obtenus du $\rho_{F,F'}$ sont : 0.4832, -0.3479, 0.0029 et 0.0525.




(a) (b)

Fig. 3.45: -a. Evolution du $p_{F,F'}$ en fonction de la clé dynamique ,-b) Distribution du $p_{F,F'}$

Dans les figures 3.46-a, 3.46-b, nous présentons l'évolution du pourcentage de différence en bits entre les deux couches de substitution $PDH$ en fonction de la clé dynamique et sa distribution. Nous pouvons faire les remarques suivantes :
97.4 % des couches de substitution possèdent des $PDH$ supérieurs à 48 % et seulement 0.026% des couches de substitution possèdent des $Pd_H$ inférieurs à 48 %.
Par ailleurs, les maximum, minimum, moyenne et écart-type obtenus du $PDH$ sont : 54.003, 44.921, 49.977 et écart type 1.0224.




(a) (b)

Fig. 3.46:-a. Evolution du *PDH* en fonction de la clé dynamique, -b. Distribution du *PDH*

Basé sur les résultats des figures 3.45 et 3.46, nous pouvons conclure que la majorité des couches de substitution satisfont le critère de la sensibilité à la clé dynamique.

Finalement l'analyse de tous les résultats obtenus, selon les différents critères, des différentes couches de substitution produites prouvent que ces dernières (en large majorité) possèdent des bonnes propriétés cryptographiques contre la cryptanalyse.

A titre comparatif, nous donnons ci-dessous, les résultats obtenus selon 6 critères par les deux couches de substitution statiques utilisées respectivement par les algorithmes AES et RC6.

Tableau 3.2 : Étude comparatifs de tableau de substitution utilisés dans l'algorithme AES et RC6

| | $NLF$ | $LP_F$ | $DP_F$ | $SAC$ | $BIC$ | $\rho$ |
|---|---|---|---|---|---|---|
| RC6 | 80 | $-2^{-3}$ | $2^{-3}$ | 0.4661 | 0.4362 | 0,153 |
| AES | 112 | $2^{-6}$ | $2^{-6}$ | 0.4998 | 0.4998 | -0.0609 |

Ci-dessous nous donnons le pseudo-code réalisant les tests des figures 3.45 et 3.46

***Entrée*** *:* $nk, n, rs, na$
***Sortie*** *:* $\rho$, $Pd_H$
$Q = 2^n$
*//Remplir le tableau initial, avec n=8*
$Pour\ \ it$ *allant de 1 à* $\ Q$
$Temp(it) = it$
$Fin\ \ it$
*-- Procédure de calcul du* $\rho$ *et du* $\ Pd_H$ *mettant en jeu deux fonction de substitution produites en fonction de deux clés qui diffèrent d'un seul bit dont la position est tirée aléatoirement.*
$Pour\ itk$ *allant de 1 à* $\ nk$
$a = floor(Q \times rand(1, na))$
$X = supboxvector(temp, Q, a, rs);$
$Si\ na = 4$
$a' = typecast(uint8(a), 'uint32')$
$Si\ non\ \ na = 8$
$a' = typecast(uint8(a), 'uint64')$
$Fin\ Si$
$i = \text{ceil}(rand \times na \times \text{n})$
$a'=a' \oplus 2^i$
$X' = supboxvector(temp, Q, a', rs);$
$Xb = dec2bin(X, Q \times n)$
$Xb' = dec2bin(X', Q \times n)$
$\rho(itk) = corrcoef(X, X')$
$W(itk) = sum(Xb \oplus Xb')$
$Pd_H(itk) = (W(itk)/(\text{n} \times \text{Q})) \times 100$
$Fin\ \ itk$

Fig. 3.47: Pseudo-code réalisant les tests des figures 3.45, et 3.46

### 3.6.6 Performances en termes de périodicité :

Pour tester les performances des cartes Skew tent et Cat en termes de périodicité, nous avons calculé pour chacune d'elle l'orbite en fonction des paramètres utilisés (voir pseudo-code de la figure 3. 48), nous obtenons les résultats suivants des orbites $O$ :
Carte Skew tent :

| Tapez une équation ici. | $a_1$ =109 | $a_2$=228 | $a_3$ =96 | $a_4$ =34 |
|---|---|---|---|---|
| $O(a_i), i = 1,2,3,4$ | 59220 | 13135 | 12644089920 | 1045152 |
| $O(a_1, a_2)$ | | 174846 | | |
| $O(a_1, a_2, a_3)$ | | | 133920 | |
| $O(a_1, a_2, a_3, a_4)$ | | | | 2600 |

Dans le cas de l'utilisation plusieurs paramètres, le calcul de l'orbite se fait de façon récursive.

**Carte Cat**

| Tapez une équation ici. | $\boldsymbol{a_1} = \{\boldsymbol{u}=\mathbf{1},\boldsymbol{v}=\mathbf{1}\}$ | $\boldsymbol{a_2} = \{\boldsymbol{u}=\mathbf{2},\boldsymbol{v}=\mathbf{5}\}$ | $\boldsymbol{a_3} = \{\boldsymbol{u}=\mathbf{7},\boldsymbol{v}=\mathbf{3}\}$ | $\boldsymbol{a_4} = \{\boldsymbol{u}=\mathbf{5},\boldsymbol{v}=\mathbf{2}\}$ |
|---|---|---|---|---|
| $\boldsymbol{O(a_i), i=1,2,3,4}$ | 12 | 6 | 16 | 8 |
| $\boldsymbol{O(a_1, a_2)}$ | | 12 | | |
| $\boldsymbol{O(a_1, a_2, a_3)}$ | | | 48 | |
| $\boldsymbol{O(a_1, a_2, a_3, a_4)}$ | | | | 48 |

Dans le cas de la carte Cat, on a la relation suivante entre les différentes orbites $O_i$:

$$O_t = ppcm(O_1, \dots O_k), k = 2,3,4$$

***Remarque*** : la longueur de l'orbite dans le cas de la carte Skew tent est nettement plus grande que celle de la carte Cat. En effet, la trajectoire de la première carte est non linéaire, tandis que la trajectoire de la deuxième carte est linéaire, et donc la périodicité est atteinte plus vite. Ci-dessous dans la figure 3.48, nous donnons le pseudo-code réalisant la mesure de la périodicité pour les deux cartes chaotique Skew tent et Cat.

**Entrée** : $Tb, na, nk, rp, ch$ ;
**Sortie** : $P$
--$P$ est un tableau de taille nk dont chaque élément contient la périodicité de la clé formée de $na$ paramètres de contrôle.
-- Remplir le tableau initial
$Pour\ it$ allant de 1 à $Tb$
    $Temp(it) = it$
$fin\ \ it$
--Procédure de calcul de la périodicité
  -- paramètres de contrôle
Pour $i$ allant de 1 jusqu'à $nk$
    $Kp = floor(rand(1, na) \times Tb)$
    $X = Temp$
    Pour $io$ allant de 1 à l'infini
        $Pour\ ita$ allant de 1 à $na$
            $Si\ ch = 0$
  $/Permutation\ par\ \ la\ carte\ skew\ tent$
                $X = permboxvector(X, Tb, Kp(ita), rp);$
              $Si\ non\ ch = 1$
              $/Permutaion\ par\ la\ carte\ Cat$
               $X = permcat(X, Tb, Kp(ita\ ), rp);$
            $Fin\ Si$
    $Fin\ ita$
    $Si\ (X = Temp)$
            $P(i) = io$
            $Sortie$ de la boucle $ito$
    $Fin\ Si$
      $Fin\ io$
$Fin\ i$

Fig. 3.48: Pseudo-code pour la mesure de la périodicité pour les cartes Skew tent et Cat

### 3.6.7 Performances de la couche de permutation

Dans ce paragraphe, nous étudions les performances de deux cartes chaotiques, la carte Skew tent et la carte Cat utilisées en tant que couche de permutation. L'effet d'avalanche finale produite par un crypto-système qui n'utilise pas une couche de diffusion proprement dit, résulte de l'association de l'opération de substitution suivie de l'opération de permutation sur les bits. Pour n'importe quel bit changé dans un octet de texte en clair, une bonne couche de substitution doit provoquer un changement sur pratiquement la moitié des bits (# 4 bits) de l'octet en question après l'opération de substitution. Une bonne couche de permutation sur les bits doit diffuser aléatoirement et uniformément les bits de l'octet substitué sur l'ensemble des bits du bloc.

Notons au passage que la couche d'addition des clés d'itérations n'a aucune influence sur les performances de l'association substitution-permutation. Elle sert seulement à masquer le contenu du bloc.

Pour quantifier le degré de diffusion du changement d'un bit dans un bloc, par l'association substitution-permutation, nous réalisons l'expérience suivante :

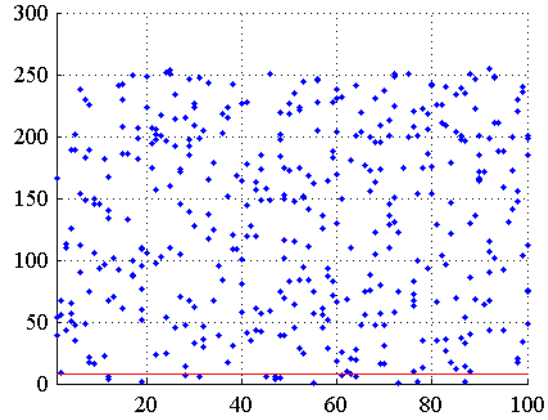
D'abord, nous générons de façon aléatoire et uniforme 1000 clés dynamiques pour l'opération de substitution, suivi de 1000 autres clés pour la couche de permutation. Puis, nous formons deux blocs de texte en clair de taille $Tbo$ octets chacun : $B1 = [0, 0, \cdots 0, 0]$ et $B2 = [0, 0, \cdots 0, 1]$, le deuxième bloc diffère du premier bloc seulement dans le premier octet de poids faible qui a la valeur 1. Ensuite, pour chaque clé de substitution et chaque clé de permutation, nous effectuons les opérations suivantes :

$$
\begin{aligned}
&R1_i = P_{Kp_i}\left[dec2bin\left\{S_{Ks_i}[B1], 8\right\}\right] \\
&R2_i = P_{Kp_i}\left[dec2bin\left\{S_{Ks_i}[B2], 8\right\}\right] \\
&\delta_i = find[R1_i \neq R2_i] \\
&Nbd_i = length\ (\delta_i) \\
&R1o_i = bin2dec(R1_i, 8) \\
&R2o_i = bin2dec(R2_i, 8) \\
&\delta o_i = find[R1o_i(1{:}Tbo-1) \\
&\qquad\qquad \neq R2o_i(1{:}Tbo-1)] \\
&Nod_i = length\ (\delta o_i)
\end{aligned}
$$

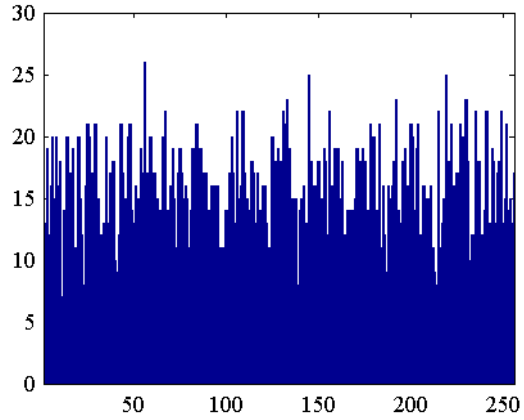
$\delta_i$ et $Nbd_i$ donnent les indices de (1 à $Tb = 256$) des bits différents entre les blocs $R1_i$ et $R2_i$ et leur nombre. De même $\delta o_i$ et $Nod_i$ donnent les indices de (1 à $Tbo - 1$) des octets différents entre les blocs $R1o_i$ et $R2o_i$et leur nombre.

Dans les figures 3.49-a, et 3.50-a nous présentons l'évolution des indices $\delta_i$ en fonction des clés (pour plus de clarté nous avons figuré seulement 100 clés) respectivement pour la carte Skew tent et la carte Cat. On voit clairement que la répartition des indices est aléatoire dans les deux cas de figures. Dans les figures 3.49-b, et 3.50-b, nous présentons l'histogramme des indices (1 à $Tb$). Les résultats obtenus montrent que l'histogramme des indices de la carte Skew tent, semble plus uniforme que celui de la carte Cat.

Le nombre de bits égaux à 1, produit par les 1000 couches de substitution est 3951, soit 3.951 bits en moyenne par couche. Ceci est proche de la valeur optimale 4 bits et confirme encore une fois la performance de la carte Skew tent en tant que couche de substitution.
Par ailleurs, afin de quantifier le nombre des bits qui ne participent pas au processus de diffusion, nous avons construit un tableau contenant les indices $\delta_i$ des 1000 clés, soit 3951 indices et nous avons cherché dans ce tableau tous les indices supérieurs à 248, et leur fréquence. Nous avons trouvé 133 indices, soit 3.3662% pour la carte Skew tent et 119 indices, soit 3.0119% dans le cas de la carte Cat, qui restent dans l'octet sous test.
Les résultats obtenus par les différentes couches de permutation montrent que le degré de diffusion de ces couches est globalement largement satisfaisant.



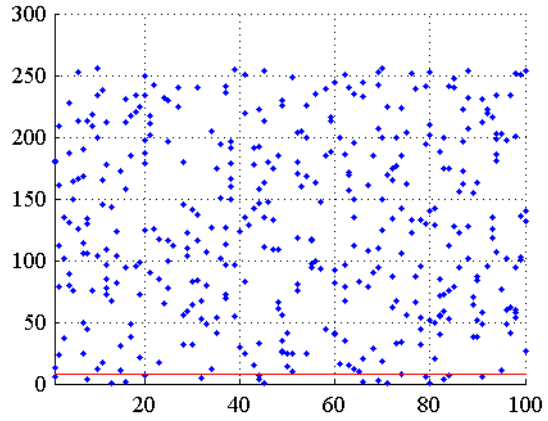

(a)



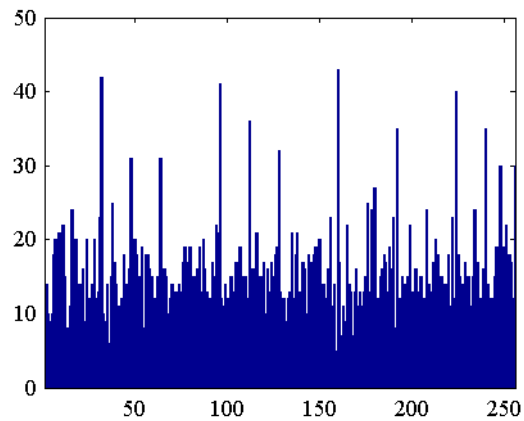

(b)

Fig. 3.49:-a. Evolution des indices $\delta_i$ de la carte Skew tent en fonction des clés.

-b. Histogramme des indices $\delta_i$ (1 à *Tb)* de la carte carte Skew tent




(a)

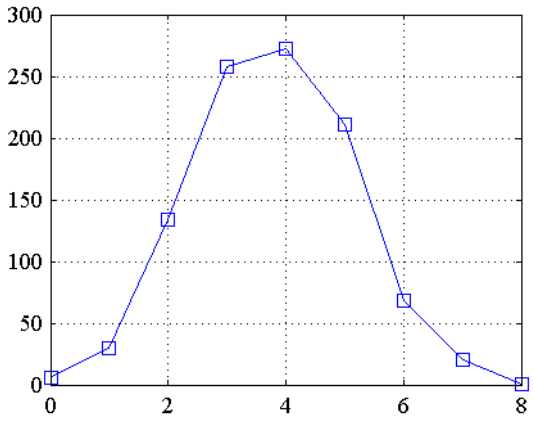



(b)

Fig. 3.50: -a. Evolution des indices $\delta_i$ de la carte Cat en fonction des clés.

-b. Histogramme des indices (1 à *T*b) de la carte carte Cat

Par ailleurs, afin de quantifier le degré de la diffusion au niveau des octets, nous avons construit un tableau contenant les nombres $Nod_i$ des 1000 clés, et nous avons calculé leur fréquence. Dans les figures 3.51-a et 3.51-b, nous présentons la fréquence des nombres d'octets modifiés $Nod$, respectivement pour la carte Skew tent et la carte Cat.
Nous remarquons que $Nod_{max} = 8$. En effet, ce cas très rare, suppose que dans la couche de substitution, les 8 bits de l'octet modifié ont changé de valeur, et que ces 8 bits sont diffusés sur 8 octets différents parmi les 32 octets du bloc ($Tbo$ =32 octets).



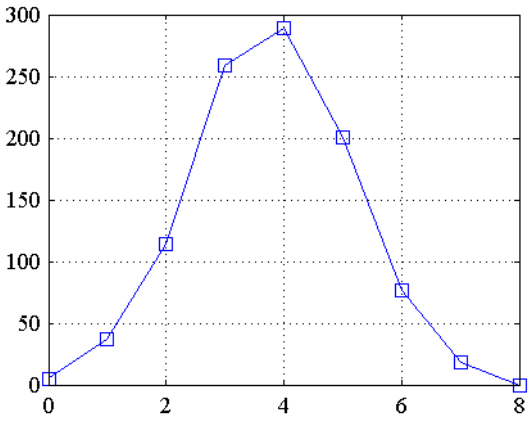




Fig. 3.51 :-a. Fréquence des nombres d'octets modifiés $Nod$ pour la carte Skew tent

-b. Fréquence des nombres d'octets modifiés $Nod$ pour la carte Cat

Nous constatons que 74.1% des $Nod$ sont tels que : $3 \leq Nod \leq 5$ dans le cas de la carte Skew tent, et 74.9% dans le cas de la carte Cat. Aussi, la fréquence maximale dans les deux cas de cartes chaotique se situe pour $Nod = 4$.
Le tableau ci-dessous, résume la fréquence des $Nod$ pour les deux cartes.

Tableau 3.3 : Fréquence des *Nod* pour les deux cartes

| $Nod$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Tente | 6 | 30 | 134 | 258 | 272 | 211 | 68 | 20 | 1 |
| Cat | 5 | 37 | 114 | 259 | 289 | 201 | 77 | 18 | 0 |

Tous ces résultats, montrent que les deux cartes chaotiques Skewt tent et Cat possèdent des performances similaires en termes de pouvoir de diffusion.
Ci-dessous dans la figure 3.52, est donné le pseudo code des tests permettant d'avoir les résultats des figures 3.49-3.51.

$\boldsymbol{Entré}: nk, n, ch$
$\boldsymbol{Sortie}:\ \delta_{itk},\ Nbd_{itk},\ \delta o_{itk},\ Nod_{itk}$
$/n\ est\ la\ précision\ (nombre\ de\ bits)\ de\ la\ couche\ de\ substitution$
$Q = 2^n$
*//Remplir le tableau initial, avec n=8*
$Pour\ \ its$ *allant de 1 à* $\ Q$
$\quad temps(it) = its$
$Fin\ \ its$
$Pour\ \ itp$ *allant de 1 à* $\ Tb$
$\quad tempp(it) = itp$
$Fin\ \ itp$
*/B1 et B2 deux tableaux, chacun de 32 octets*
$B1 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$ ;
$B2 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$ ;

$Pour\ \ itk\ allant\ de\ 1\ à\ \ nk$
$Kp(\ itk\ ) = floor(rand(1, nb) \times 2^{log2(Tb)})$
$Ks(\ itk\ ) = floor(rand(1, na) \times 2^n)$
$S_{Ks_{itk}} = subboxvector(temps, Q, Ks(\ itk\ ), 1);$
$\quad Si\ ch == 0$
/cas de carte tente comme couche de permutation
$\quad P_{Kp_{itk}} = permboxvector(tempp, Tb, Kp(itk\ ), 1);$
$\quad Si\ non\ \ ch == 1$
/cas de carte Cat comme couche de permutation
$\quad P_{Kp_{itk}} = permcat(tempp, Tb, Kp(itk\ ), 1);$
$\quad Fin\ Si$
$\quad R1_{itk} = P_{Kp_{itk}}\left[dec2bin\left\{S_{Ks_{itk}}[B1], n\right\}\right]$
$\quad R2_{itk} = P_{Kp_{itk}}\left[dec2bin\left\{S_{Ks_{itk}}[B2], n\right\}\right]$
$\quad \delta_{itk} = find[R1_{itk} \neq R2_{itk}]$
$\quad Nbd_{itk} = length\ (\delta_{itk})$
$\quad R1o_{itk} = bin2dec(R1_{itk}, n)$
$\quad R2o_{itk} = bin2dec(R2_{itk}, n)$
$\quad \delta o_{itk} = find[R1o_{itk}(1{:}Tbo - 1) \neq R2o_{itk}(1{:}Tbo - 1)]$
$\quad Nod_{itk} = length\ (\delta o_{itk})$

$\quad Fin\ pour\ \ itk$

Fig. 3.52: Pseudo code des tests permettant d'avoir les résultats des figures 3.49-3.51

### 3.6.8 Performances de la couche de diffusion basée sur la carte chaotique Cat multidimensionnelle

Normalement, le critère utilisé pour quantifier les performances d'une couche de diffusion d'un crypto-système est le nombre de branches linéaires produit par cette couche. Plus ce nombre est grand, meilleure est la diffusion.

Nous décrivons dans ce paragraphe la méthode de mesure du nombre des branches linéaires et nous quantifions aussi les performances selon : le nombre d'éléments différents $Ned$, et leur pourcentage $P_{Ned}$, la somme de valeur absolue de la différence entre les éléments $UACI$, et le pourcentage du nombre de bits différents pour :

1) la couche de diffusion binaire statique proposée 32 x 32
2) la couche de diffusion à base de la carte cat de haute dimension obtenue par multiplication des matrices, et nous considérons les quatre matrices de diffusion suivantes : $D_1 = A08,\ D_2 = A08 \times A18,\ D_3 = A08 \times A18 \times A28\ et\ D_4 = A08 \times A18 \times A28 \times A38$. La taille des matrices est $n \times n,\ avec\ n = 8$
3) la couche de diffusion à base de la carte cat de haute dimension obtenue à partir de matrices ayants une structure particulière, et nous considérons les quatre matrices de diffusion suivantes : $D_1 = M_0,\ D_2 = M_0 \times M_1,\ D_3 = M_0 \times M_1 \times M_2\ et\ D_4 = M_0 \times M_1 \times M_2 \times M_3$. La taille des matrices est $n \times n,\ avec\ n = 4, ou\ 8, ou\ 16, ou\ 32$ correspondants aux tailles des blocs égales à 32, 64, 128, 256 bits respectivement, si les $n$ sous blocs sont des octets, et 128, 256, 512, 1024 bits respectivement si les $n$ sous blocs sont des mots de 32 bits.

#### *2.6.8.1 Mesure du nombre des branches linaires de la couche de diffusion de la méthode 3*

La méthode 3), est plus générale que la méthode 2), et donne des résultats similaires. Pour cela, nous réalisons l'expérience et présentons les résultats selon la méthode 3), comme suit :

D'abord, nous créons de façon aléatoire et uniforme 1000 blocs $BS_i$ dont les valeurs varient de 1 à $2^{Tsb}$ :

$$BS_i = floor\left(rand(1,n) * 2^{Tsb}\right), i = 1,2,\cdots,nB, avec\ nB = 1000$$

L'utilisation à l'entrée de la couche de diffusion des blocs $BS_i$ dont les éléments sont générés de façon aléatoire et uniforme, découle du fait que l'entrée de cette couche résulte d'une opération d'addition des clés qui apporte déjà l'aspect aléatoire et uniforme, en plus suivi de l'opération de substitution.

Ensuite, pour chaque structure : $ins = 1$ à 4,

pour chaque clé de diffusion dynamique $Kd_k, k = 1,2, \ldots nk = 1000$,

nous générons :

une matrice de diffusion ${D^{ins}}_{Kd_k}$ correspondante : ${D^{ins}}_{Kd_k} = Catdimension(Kd_k, ins, n)$,

puis, pour chaque bloc $BS_i$, $i = 1,2,\cdots,nB$, nous calculons :

- les blocs de textes en clair diffusés $BSd_{i,k}$ :

$$Bsd_{i,k} = mod\{(D^{ins}{}_{Kd_k} \times BS_i{}'), \qquad 2^{Tsb}\}'$$

- la matrice du nombre d'éléments différents entre les blocs $BS_i$ et $Bsd_{i,k}$:

$$Ned_{i,k} = length\left(find\left(Bsd_{i,k} \sim = BS_i\right)\right)$$

- la matrice pourcentage du nombre d'éléments différents entre les blocs $BS_i$ et $Bsd$

$$P_{Ned_{i,k}} = \{Ned_{i,k}/n\} \times 100$$

- la matrice somme de valeur absolue de la différence entre les éléments des blocs $BS_i$ et $Bsd$.

$$UACI_{i,k} = \{sum(abs(Bsd_{i,k} - BS_i)/(2\hat{}Tsb - 1))/n\} \times 100$$

- la matrice nombre de branches linéaires :

$$NB_{i,k} = PH(BS\_b_i) + PH(Bsd\_b_{i,k})$$

Où $PH$ désigne le poids de Hamming, opèrant sur les bits des blocs $BS_i$ et $Bsd_{i,k}$, donc, il faut convertir les éléments des blocs $BS_i$ et $Bsd_{i,k}$ en bits, notés $BS\_b_i$ et $Bsd\_b_{i,k}$.
En Matlab, ce calcul se fait comme suit :

$$BS\_b_i = reshape(dec2bin[BS_i, Tsb], 1, Tsb \times n);$$

$$Bsd\_b_{i,k} = reshape\left(dec2bin\left[Bsd_{i,k}, Tsb\right], 1, Tsb \times n\right);$$

$$NB_{i,k} = \left[length(find(BS\_b_i == '1')) + length(find(Bsd\_b_{i,k} == '1'))\right]/n;$$

- la matrice pourcentage du nombre de bits différents entre les blocs $BS\_b_i$ et $Bsd\_b_{i,k}$

$$P_{Nbd_{i,k}} = length(find(Bsd\_b_{i,k} \sim = BS\_b_i))/n \times 100$$

Fin $i$

Fin $k$

A partir des différentes matrices 1000 x 1000 ci-dessus, nous calculons pour chacun des critères suivants : a) $Ned_{i,k}$, b) $P_{Nbd_{i,k}}$, c) $NB_{i,k}$ et d) $UACI_{i,k}$, les valeurs minimale, maximale et moyenne.

Fin $ins$.

Nous présentons dans les figures 3.53, 3.54 et 3.55 correspondants respectivement à $n = 8, 16\ et\ 32$, les valeurs minimale, maximale et moyenne pour les 4 matrices de diffusion $D_1$ à $D_4$, de chacun des critères suivants : a) $Ned_{i,k}$, b) $P_{Nbd_{i,k}}$, c) $NB_{i,k}$ et d) $UACI_{i,k}$.

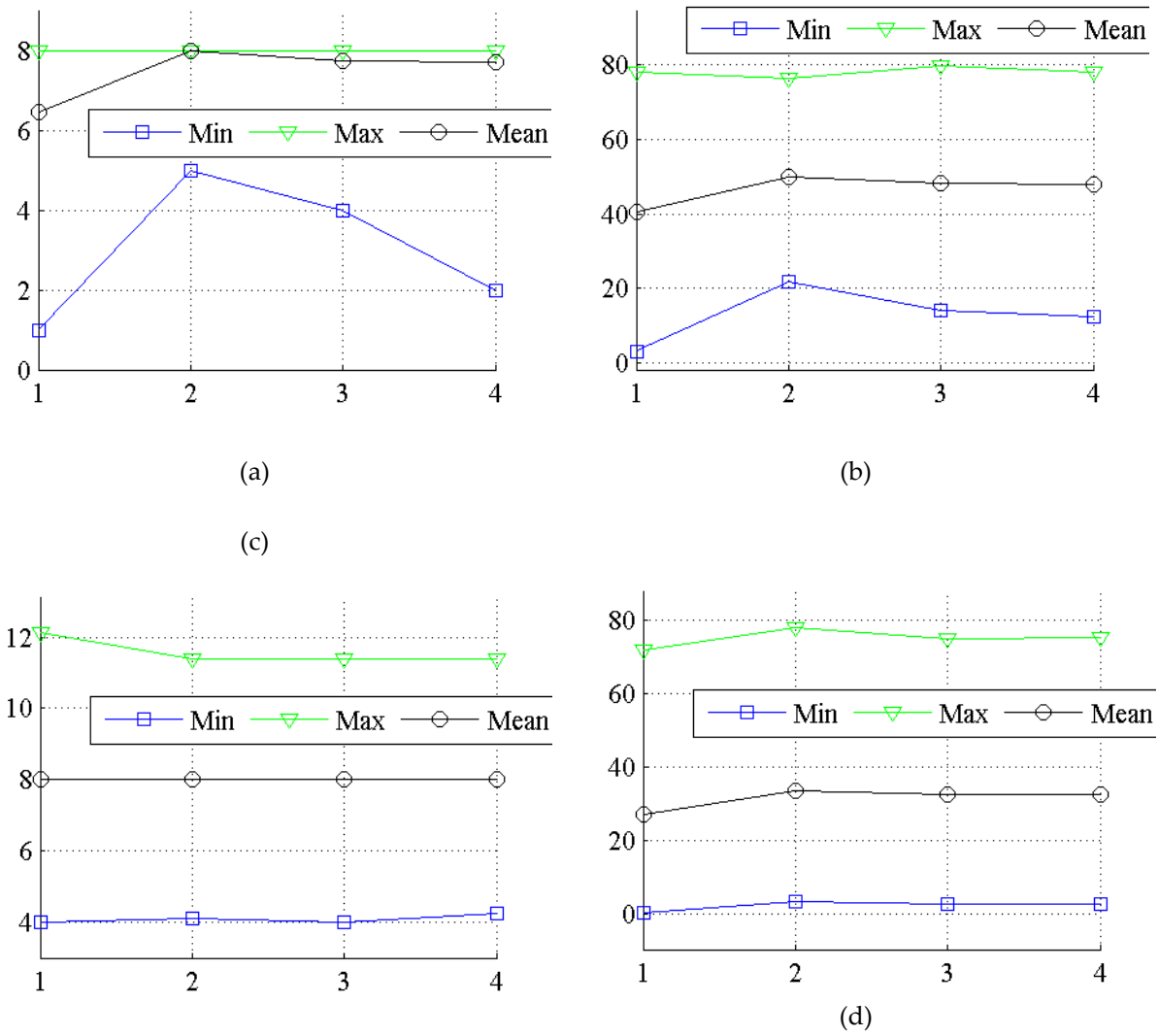



Fig. 3.53: Valeurs minimale, maximale et moyenne des :

a) $Ned_{i,k}$, b) $P_{Nbd_{i,k}}$, c) $NB_{i,k}$ et d) $UACI_{i,k}$, des 4 matrices de diffusion, dans le cas $n = 8$.

Nous pouvons faire les remarques suivantes : le nombre des branches linéaires est égale à 4 quelle que soit la matrice de diffusion utilisée.

A partir de la deuxième matrice de diffusion, les performances de la diffusion sont atteintes, car la valeur moyenne des 3 autres critères est optimale : $Ned = 8$, $P_{Nbd} = 50\ \%$, et $UACI = 33.3\%$.

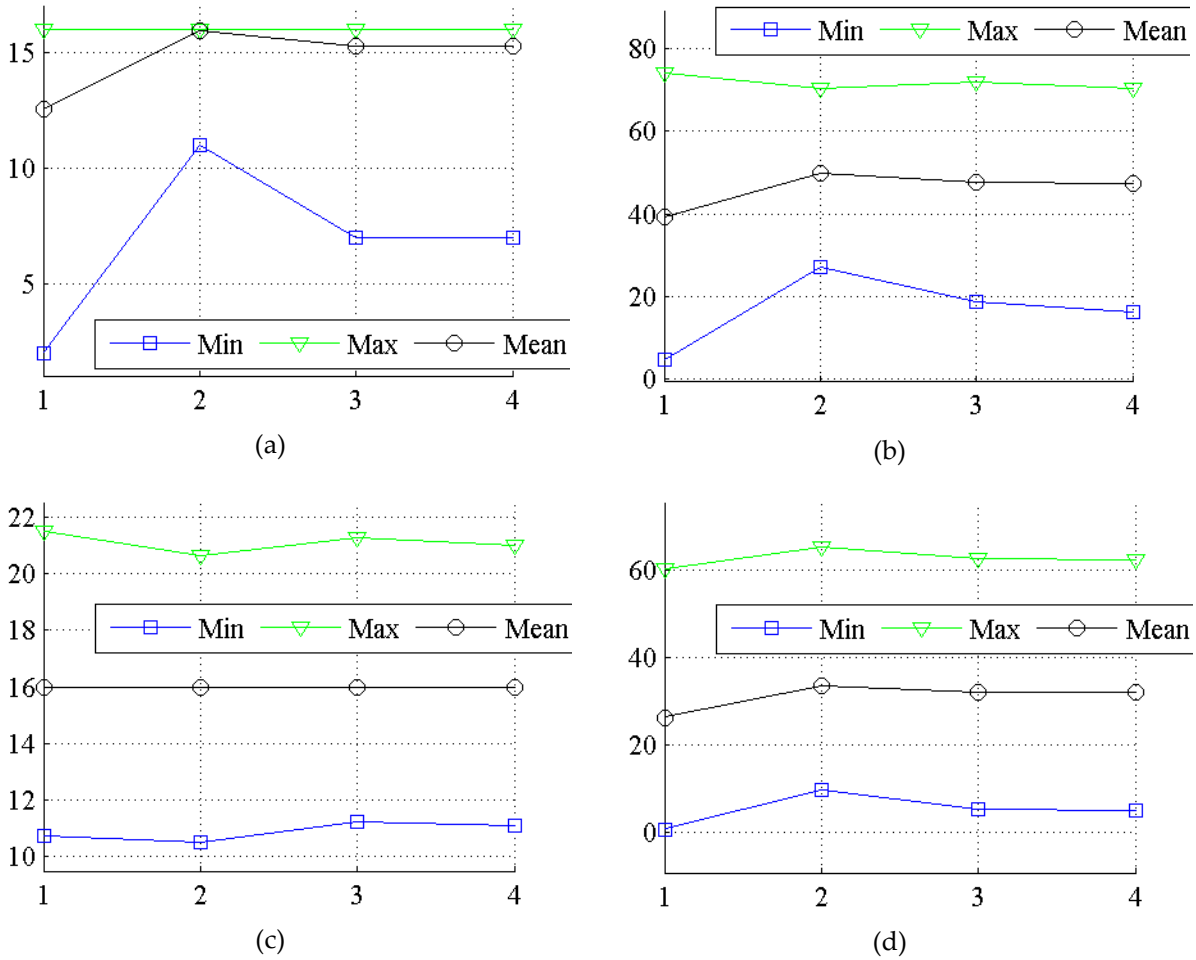



Fig. 3.54: Valeurs minimale, maximale et moyenne des :

a) $Ned_{i,k}$, b) $P_{Nbd_{i,k}}$, c) $NB_{i,k}$ et d) $UACI_{i,k}$, des 4 matrices de diffusion, dans le cas $n = 16$.

Nous constatons que : le nombre des branches linéaires est compris entre 10 et 11 en fonction de la matrice de diffusion utilisée. Ce nombre dépend en fait de la valeur de $n$.

A partir de la deuxième matrice de diffusion, les performances de la diffusion sont atteintes, car la valeur moyenne des 3 autres critères est optimale : $Ned = 16$, $P_{Nbd} = 50\ \%$, et $UACI = 33.3\%$.

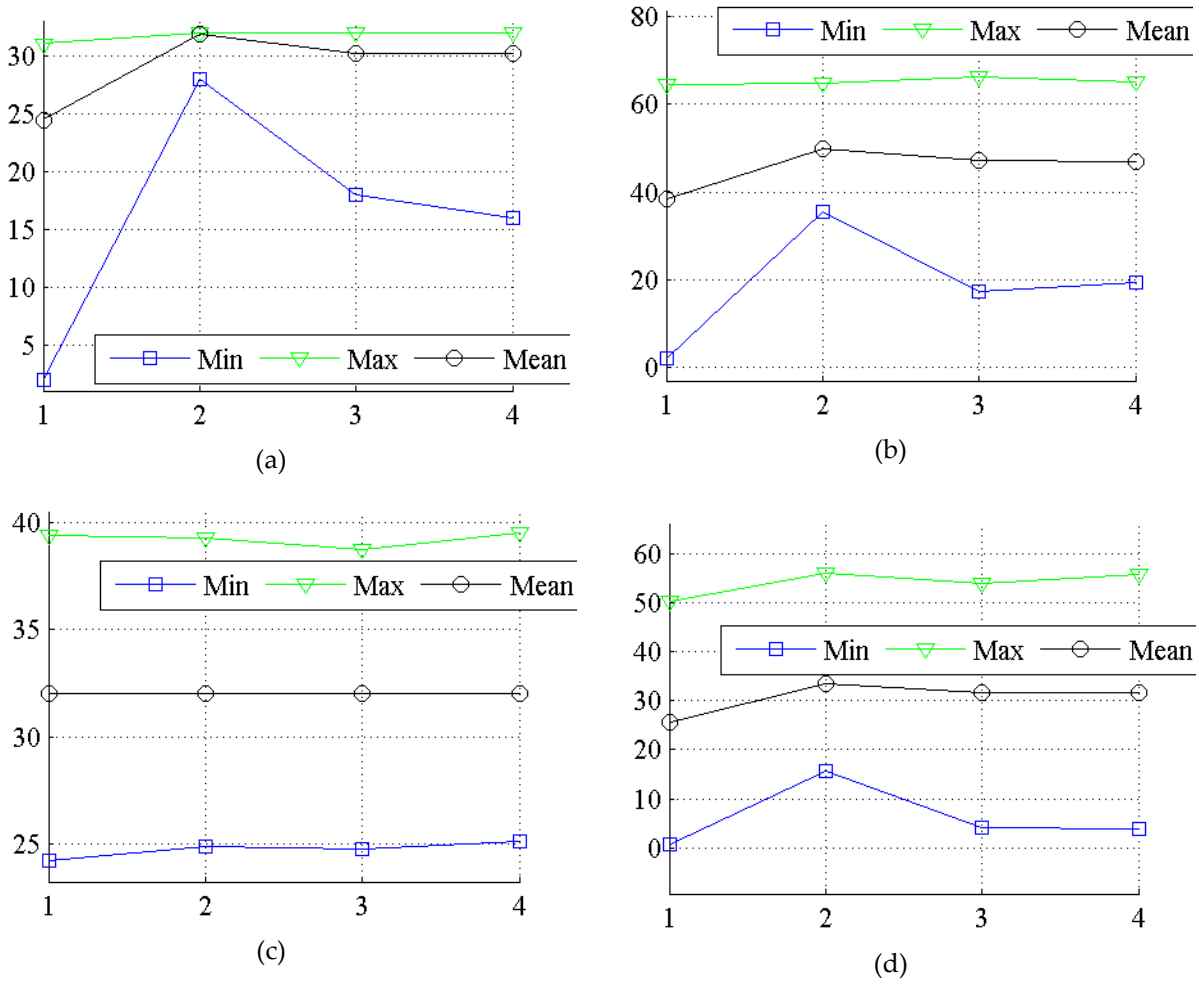



Fig. 3.55: Valeurs minimale, maximale et moyenne des :

a) $Ned_{i,k}$, b) $P_{Nbd_{i,k}}$, c) $NB_{i,k}$ et d) $UACI_{i,k}$, des 4 matrices de diffusion, dans le cas $n = 32$.

Nous remarquons que : le nombre des branches linéaires est compris entre 24 et 26 en fonction de la matrice de diffusion utilisée. A partir de la deuxième matrice de diffusion, les performances de la diffusion sont atteintes, car la valeur moyenne des 3 autres critères est optimale : $Ned = 32$, $P_{Nbd} = 50$ %, et $UACI = 33.3\%$.

### *2.6.8.2 Temps de génération de la couche de diffusion*

Dans ce paragraphe, nous donnons le temps moyen nécessaire (en micro seconde) pour la génération de chacune des matrices de diffusion $D_i, i = 1, \cdots, 4$, de la 3ème méthode (basée sur la carte cat multidimensionnelle). La simulation est faite en Matlab utilisant le même micro-ordinateur dont les caractéristiques sont décrites plus loin.

Tableau 3.4. Temps moyen de calcul (micro-seconds) des quatre matrices de diffusion en fonction de $n$

| ns \dimension n | 4 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|
| 1 | 0,028125 | 0,0328125 | 0,0421875 | 0,0546875 | 2,0796875 | 16,525 |
| 2 | 0,0359375 | 0,0359375 | 0,0421875 | 0,0765625 | 2,4703125 | 18,534375 |
| 3 | 0,0390625 | 0,040625 | 0,046875 | 0,0875 | 3,3984375 | 18,8515625 |
| 4 | 0,040625 | 0,0453125 | 0,05 | 0,1015625 | 3,9 | 21,515625 |

Ce temps augmente en fonction de $n\ et\ ns$.

## 3.7 Performances globales

### 3.7.1 Effet d'avalanche : déduction du nombre d'itérations $r$ nécessaires

Un des critères les plus importants pour quantifier la robustesse des crypto-systèmes, est le critère achevant l'effet d'avalanche globale à atteindre. Ce critère dépend du nombre $r$ d'itérations nécessaires pour atteindre l'objectif fixé. Lorsque l'effet d'avalanche est réalisé par un crypto-système donné, alors, ce dernier peut résister contre les attaques linéaire et différentielle et contre l'attaque de texte en clair/chiffré connu et choisi.

L'effet d'avalanche est atteint, si une différence d'un seul bit (pire des cas) dans deux textes en clair/chiffré aboutit à une différence de 50% dans les bits des textes chiffré/déchiffré respectivement. La mesure de l'effet d'avalanche est alors faite par la distance de Hamming ou par le pourcentage des bits qui ont changé. A cet effet, nous réalisons le test suivant (sur deux textes en clair choisi) :

Générations de 1000 clés secrètes (du générateur chaotique) $K_j, j = 1, \cdots, nk = 1000$

Choix d'un bloc de texte en clair de taille $Tb$ bits ne contenant que des zéros octets: $P1 = [0, 0, \cdots, 0]$

Pour $i = 1$ à $Tb$,

calcul du texte en clair $P2i$ qui ne diffère que d'un seul bit en position $i$ par rapport au texte en clair $P1$: $P2\_b = reshape\ (dec2bin(P1,8), 1, Tb)$

$P2\_b(i) = P2\_b(i) \oplus 1$

$P2i = bin2dec\ (reshape(P2\_b, length(P2\_b)/8, 8))$

$/P1$ et $P2$ sont deux blocs d'octets

Pour chaque clé sécrète $K_j, j = 1, \cdots, nk = 1000,$

Pour $it = 1$à $r = 10$

Nous réalisons les opérations suivantes :

Chiffrement de $P1 : C1^{it} = E_{K_j}^{it}(P1)$

Chiffrement de $P2_i : C2i^{it} = E_{K_j}^{it}(P2i)$

$/C1^{it}$ et $C2i^{it}$ sont deux bloc d'octets chiffré

Calcul de la distance de Hamming :

$DH(i, it, j) = sum\left\{reshape\left(dec2bin\left(\ C1^{it} \oplus C2i^{it}\right), 1, Tb\right)\right\}$

Fin $it$

Fin $K_j$

Fin $i$

Finalement, la matrice 3-D, $DH(i,it,j)$ contient 1000 matrices 2-D, $DH(i,it)$ dont chacune est formée de $Tb$ lignes et $r$ colonnes. Le pourcentage en bits de la distance de Hamming est donné par :

$$PDH(i,it,j) = \frac{DH(i,it,j)}{Tb} \times 100$$

À partir de la matrice $PDH\ (i,it,j)$ ou la matrice $DH(i,it,j)$, nous calculons la matrice de valeurs moyennes sur les 1000 clés (indice $(K_j)$, soit :

$$M\_DH(i,it) = mean(DH\ (i,it,:))$$
$$M\_PDH(i,it) = mean(PDH\ (i,it,:))$$

Pour le premier crypto-système SPN-1, nous illustrons dans la figure 3.56 (cas de l'utilisation de la carte Skew tent), et dans la figure 3.57 (cas de l'utilisation de la carte Cat), pour chaque position $i$ (indiquant la position du bit changé du texte en clair), l'évolution du pourcentage $M\_PDH(i,it)$ en fonction du nombre d'itérations $r$. Nous avons pris $Tb = 256$ bits.

Nous remarquons que, quelle que soit la position du bit changé du texte en clair, l'effet d'avalanche est achevé, $M\_PDH(i,it) \cong 50\%$, au bout de $r = 6$ itérations, dans les deux cas de figures. Pour $r > 6$, le crypto-système est plus robuste, mais au prix d'un temps de calcul plus important.

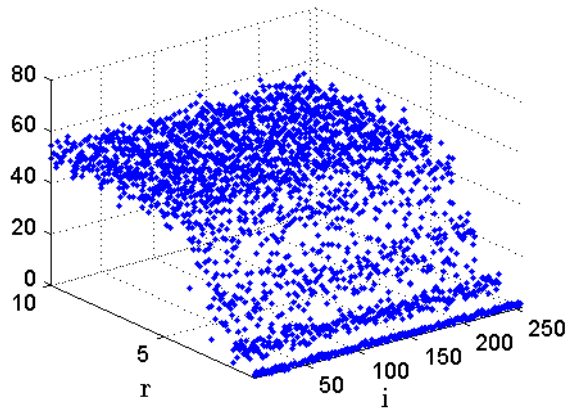



Fig. 3.56: Evolution du M_PDH en fonction du nombre d'itérations r et ceci pour chaque position $i$ du bit changé du texte en clair texte en clair, cas de la carte Skew tent

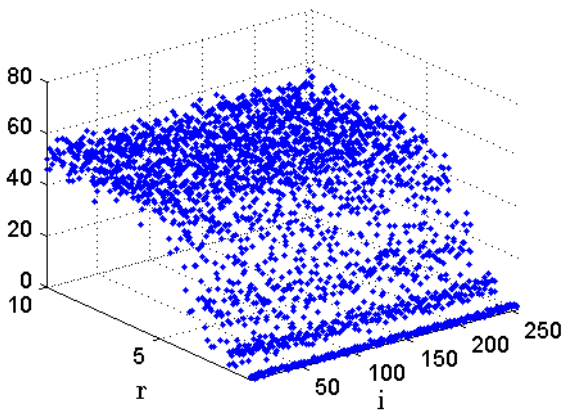



Fig. 3.57 : Evolution du $M\_PDH$ en fonction du nombre d'itérations $r$ et ceci pour chaque position $i$ du bit changé du texte en clair, cas de la carte Cat

Pour plus de clarté, nous avons tracé respectivement, dans les figures 3.58 et 3.59, une coupe des figures 3.56 et 3.57, dans le cas où la position du bit changé du texte en clair se trouve à

$i = 256$ correspondant au bit de poids faible du bloc sous traitement. Aussi, nous avons tracé dans les figures 3.58 et 3.59, la distance de Hamming correspondant aux figures 3.60 et 3.61.

D'après les résultats de ces différentes figures, nous voyons clairement que l'effet d'avalanche est obtenu à partir de $r = 6$, pour les deux variantes du premier crypto-système SPN-1.

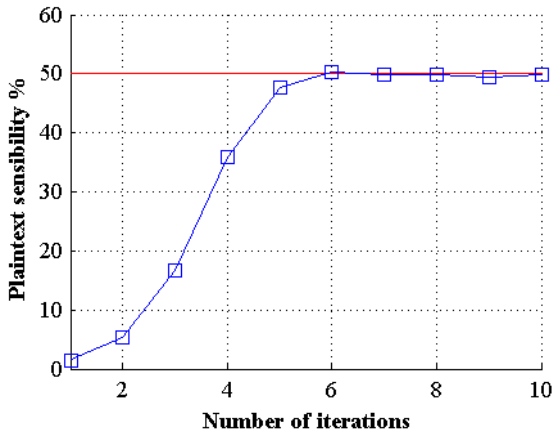



Fig. 3.58: Evolution du *M_PDH* en fonction du nombre d'itérations *r*, pour *i=256*, cas de la carte Skew tent

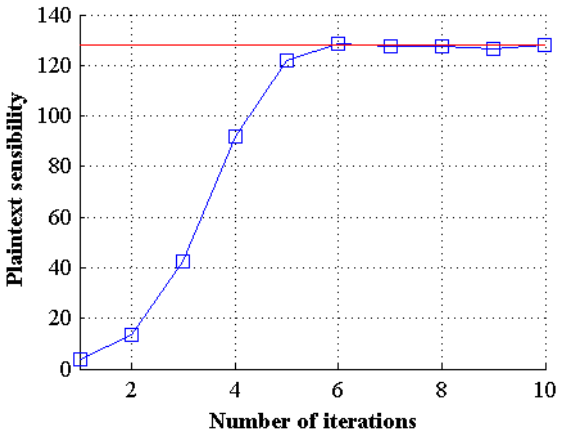



Fig. 3.59: Evolution de *M_DH* en fonction du nombre d'itérations *r* pour *i* =256, cas de la carte Skew tent

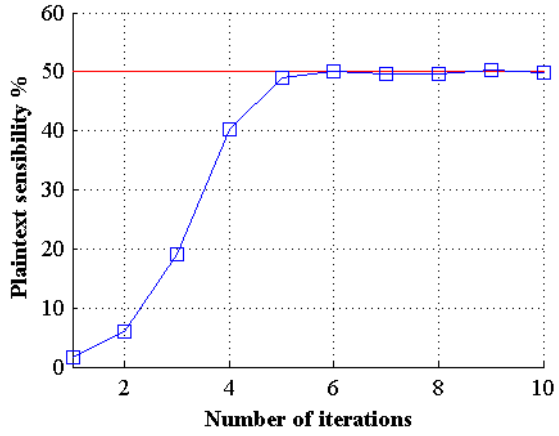



Fig. 3.60: Evolution du *M_PDH* en fonction du nombre d'itérations *r*, pour i=256, cas de la carte Cat D-2

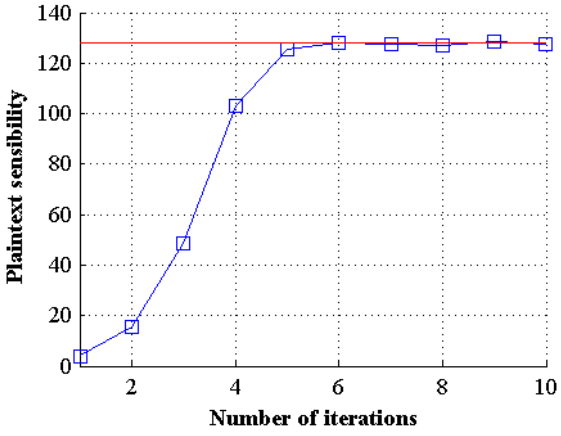



Fig. 3.61: Evolution de *M_DH* en fonction du nombre d'itérations r, pour i=256, cas de la carte Cat D-2

***Remarque*** : les résultats obtenus dans le cas où $Tb = 128$, permettent de statuer que l'effet d'avalanche est achevé aussi à partir de $r = 6$ itérations.

Pour le deuxième crypto-sytème SPN-2, nous illustrons dans la figure 3.62 (cas de l'utilisation de la couche de diffusion binaire 32 x 32), et dans la figure 3.64 (cas de l'utilisation de la carte cat de taille 32 x32 comme couche de diffusion, basée sur la matrice de diffusion $D_2$), pour la position $i = 256$, l'évolution du pourcentage $M\_PDH$ en fonction du nombre d'itérations $r$.

Dans les figures 3.63 et 3.65 nous présentons les $M\_DH$ correspondants aux figures 3.62 et 3.64 respectivement. Nous remarquons que l'effet d'avalanche est achevé $M\_PDH \cong 50\%$, au bout de $r = 2$ itérations dans le cas de la matrice binaire statique et au bout $r = 3$ itérations dans le cas de la matrice cat multidimensionnelle. Cependant, dans ce dernier cas, la clé de diffusion est dynamique.

Ces résultats montrent clairement que le deuxième crypto-système SPN-2, est plus rapide que le premier crypto-système SPN-1, tout en étant aussi robuste vis-à-vis des attaques cryptographiques.

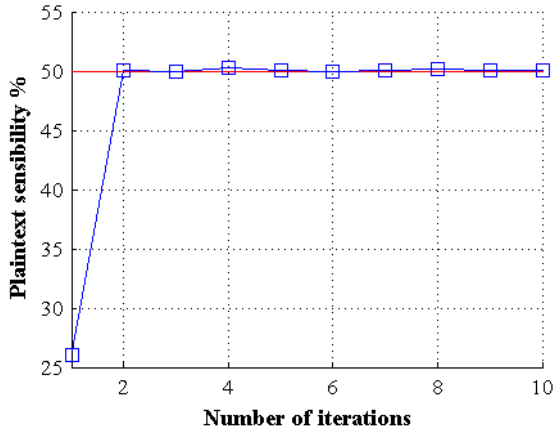



Fig. 3.62 : Evolution du *M_PDH* en fonction du nombre d'itérations *r*, pour i=256 cas de la matrice de diffusion statique

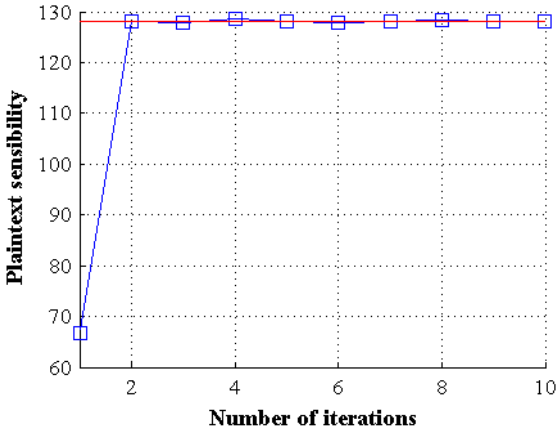



Fig. 3.63: Evolution du *M_DH* en fonction du nombre d'itérations *r*, pour *i*=256, cas de la matrice de diffusion statique

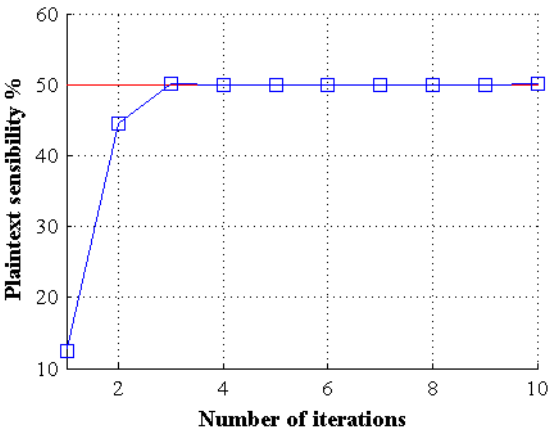



Fig. 3.64 : Evolution du *M_PDH* en fonction du nombre d'itérations *r*, pour i=256 cas de la carte Cat multidimensionnelle.

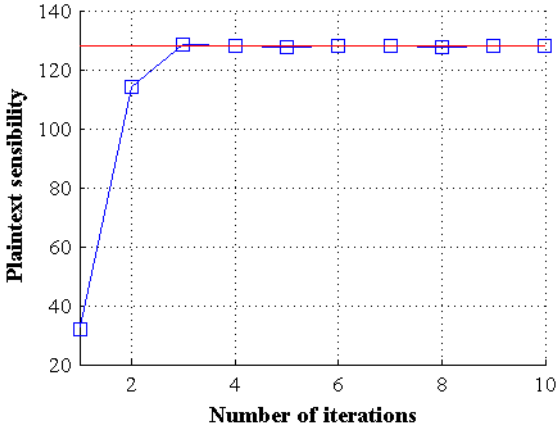



Fig. 3.65: Evolution du *M_DH* en fonction du nombre d'itérations *r*, pour *i*=256, cas de la carte Cat multidimensionnelle.

### 3.7.2 Sensibilité à la clé secrète en émission/réception

Nous quantifions ici la sensibilité à la clé secrète au chiffrement (ou au déchiffrement) des crypto-systèmes proposés. Le changement d'un seul bit même du poids faible de la clé secrète au chiffrement (ou au déchiffrement) doit entrainer une image chiffrée (ou une image déchiffrée) complètement différente. Cette sensibilité est classiquement mesurée par deux paramètres qui sont le NPCR (Number of Pixel Change Rate) et le UACI (Unified Average Changing Intensity). En plus, de ces deux paramètres qui opèrent sur les octets, nous utilisons la distance de Hamming qui opère sur les bits qui à notre avis est plus significatif.
Le test de la sensibilité à la clé sécrète au chiffrement (celui du déchiffrement est identique) est réalisé comme suit :
Générations de 1000 clés secrètes (du générateur chaotique) $K_j, j = 1, \cdots, nk = 1000$
/ Calcul de la taille $To$ en octets de l'image $I$

$$To = size(I,1) \times size(I,2) \times size(I,3)$$

/$T_K$ est la taille de la clé sécrète
Nous réalisons les opérations suivantes :
Pour chaque clé secrète $K_j, j = 1, \cdots, nk = 1000$, nous formons la clé correspondante $Ki_j$ en complémentant la valeur d'un bit de la clé $K_j$ à la position $i$ tirée de façons aléatoire :

$$i = ceil(rand \times T_K)$$

$$Ki\_b_j = reshape\left(dec2binkey(K_j, Z), 1, T_K\right)$$

/ La fonction $dec2binkey$ (voir annexe chap 2) convertit chaque élément de $K_j$ selon la précision donnée par $Z$ de cet élément.

$$Ki\_b_j(i) = Ki\_b_j(i) \oplus 1$$

$$Ki_j = bin2deckey\ (Ki\_b_j, Z)$$

/ La fonction $bin2deckey$ (voir annexe chap 2) charge chaque élément du vecteur $Z$ qui indique la précision en bit de l'élément de la clé, puis le convertit en décimal.

Ensuite, nous chiffrons la même image $I$ avec les deux clés $K_j$ et $Ki_j$ et nous les convertissons en vecteurs de taille $To$ :

$$C1_j = E_{K_j}\ (I)$$
$$C2_j = E_{Ki_j}\ (I)$$
$$C1_j = reshape(C1_j, 1, To)$$
$$C2_j = reshape(C2_j, 1, To)$$

Puis, nous calculons les paramètres NPCR et UACI :

$$NPCR = \frac{\sum_{k=1}^{To} D(k)}{To} \times 100\% \tag{3.54}$$

Avec :

$$D(k) = \begin{cases} 1\ si\ C1_j(k) \neq C2_j(k) \\ 0\ \ si\ non \end{cases}$$

Le paramètre $NPCR$, mesure le pourcentage des valeurs du niveau de gris différents entre les deux images chiffrées $C1_j$ et $C2_j$

$$UACI = \frac{1}{To}\frac{\sum_{k=1}^{To}\left|C1_j(k) - C2_j(k)\right|}{2^8 - 1} \times 100\% \tag{3.55}$$

Le paramètre $UACI$ mesure la moyenne de la différence des intensités des niveaux de gris entre $C1_j$ et $C2_j$.

En Matlab, ce calcul est donné par :

$$NPCR = \frac{length(find(C1_j \sim = C2_j))}{To} \times 100\%$$

$$UACI = \left\{sum\left(abs\left(\ C1_j - C2_j\right)/255\right)/To\ \right\} \times 100\%$$

Puis, nous calculons la distance de Hamming et du pourcentage en bits, après conversion de $C1_j$ et $C2_j$ en vecteurs de bits $C1\_b_j$ et $C2\_b_j$ :

$$C1\_b_j = \text{reshape}\left(dec2bin\left(C1_j, 8\right), 1, To \times 8\right)$$
$$C2\_b_j = \text{reshape}\left(dec2bin\left(C2_j, 8\right), 1, To \times 8\right)$$
$$DH(k) = sum\left\{C1\_b_j \oplus C2\_b_j\right\}$$
$$PDH(k) = \frac{DH(k)}{To \times 8} \times 100\%$$

Fin $K_j$

**Remarque** :

Les deux paramètres $NPCR, UACI$ ont chacun un majorant. En effet, dans le cas de deux images aléatoires uniformément distribuées et non corrélées, les valeurs attendues du $NPCR, et\ UACI$ sont données par :

$$NPCR_{Max} = (1 - 2^{-8}) \times 100\% = 99.609375\%$$
$$UACI_{Max} = \frac{\sum_{k=1}^{2^8-1} k(k+1)}{2^8(2^8 - 1)} \times 100\% = 33.46354\%$$

Nous présentons sur les figures 3.66, 3.67 et 3.68, l'évolution du $NPCR, UACI$ et $PDH$ en fonction de la clé, dans le cas de la matrice de diffusion statique 32 x 32. L'image $I$ sous teste est celle de Lena 128 x 128x3. Pour plus de visibilité, nous avons tracé la variation seulement

pour 100 clés différentes et non 1000 clés. Des résultats similaires sont obtenus dans le cas de la matrice Cat multidimensionnelle.

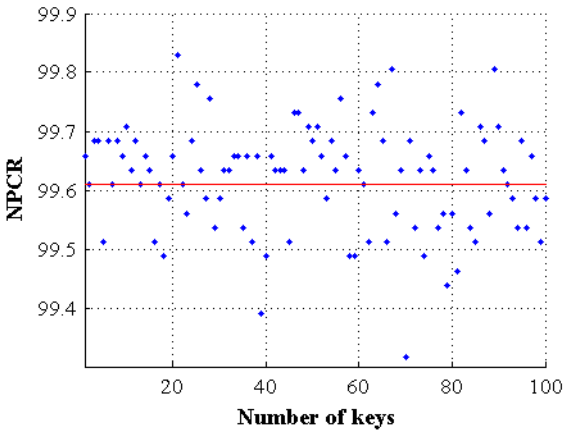



Fig. 3.66: Evolution du *NPCR* en fonction de la clé secrète

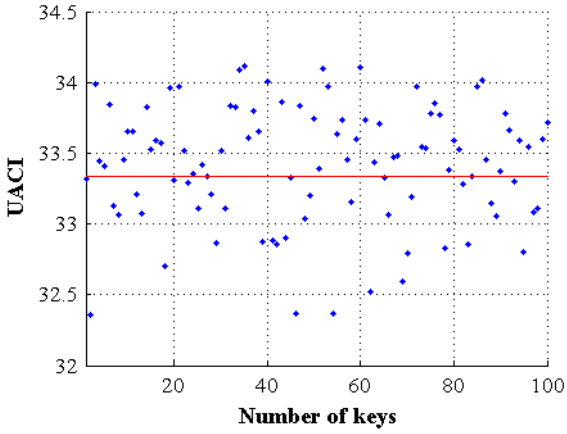



Fig. 3.67: Evolution du *UACI* en fonction de la clé secrète

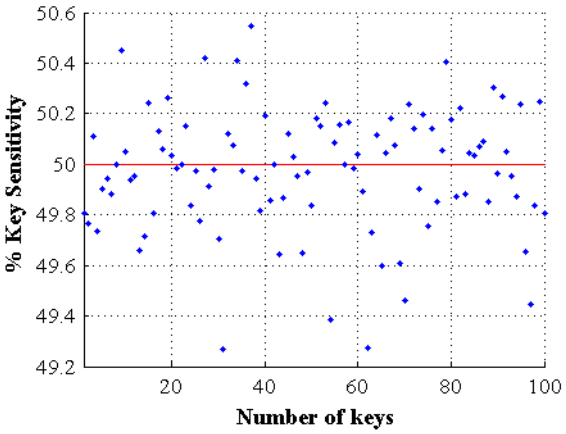



Fig. 3.68 : Evolution du *M_PDH* en fonction de la clé secrète

Dans le tableau 3.5, suivant, nous donnons la valeur moyenne et l'écart type de ces trois paramètres.

Tableau 3.5 : Moyenne et écart type des paramètres *NPCR*, *UACI* et *PDH* de l'image chiffrée Lena en mode CBC

| | *NPCR* | *UACI* | *PDH* |
|---|---|---|---|
| Moyenne | 99.6191 | 33.4187 | 49.9772 |
| Ecart type | 0.0909 | 0.4152 | 0.2423 |

Tous ces résultats montrent que le critère de la sensibilité de la clé secrète est atteint. Ci-dessous, à titre comparatif, nous donnons la valeur moyenne des paramètres $NPCR, UACI$ de

quelques crypto-systèmes basés chaos connus de la littérature, et aussi de l'AES dans le tableau 3.6.

Tableau 3.6 : Comparaison du *NPCR, UACI* de l'image chiffrée Lena en mode CBC.

| **Sensitivity of the secret key** | **NPCR** | **UACI** |
|---|---|---|
| ECKBA | 99.598 | 33.444 |
| Kumar | 99.609 | 33.461 |
| Yang | 99.61 | 33.467 |
| AES | 99.61 | 33.432 |

Les résultats obtenus sont comparables.

### 3.7.3 Analyse statistique :

Dans ce paragraphe, nous réalisons les tests de chi-carré, entropie et coefficient de corrélation des pixels adjacents sur les crypto-systèmes proposés.

#### *2.7.3.1 Uniformité de l'image chiffrée*

L'histogramme de l'image chiffrée est une caractéristique importante dans l'analyse des performances statistiques du processus de chiffrement. L'histogramme illustre comment les niveaux de gris des pixels dans une image sont distribués.
Pour résister à l'attaque statistique il faut que l'histogramme de l'image chiffrée soit très proche d'une distribution uniforme.
Pour mesurer cette uniformité, nous appliquons deux tests, le première test est le chi-carré et le deuxième test est l'entropie.

**Test de Chi-carré**

Le test chi-carré est donné ci-dessous :

$$\chi^2_{exp} = \sum_{i=0}^{Q-1} \frac{(o_i - e_i)}{e_i} \tag{3.56}$$

où $Q = 2^8 = 256$, est le nombre de niveaux de gris, $o_i$ est la fréquence d'occurrence observée de chaque niveau de gris $i \in [0, 255]$ dans l'histogramme de l'image chiffrée, et $e_i$ est la fréquence d'occurrence théorique de la distribution uniforme, ici $e_i = To/Q$, ou $To$ est la taille de l'image en octets.
La distribution de l'histogramme sous test est uniforme s'il vérifie la condition suivante : $\chi^2_{\text{exp}} \leq \chi^2_{th\ Q-1,\ \alpha} = 273{,}12$ pour un seuil $\alpha$ fixé à $\alpha = 0.05$ dans notre expérimentation.

**Test de l'Entropie**

Le deuxième test de l'uniformité est l'entropie définie par Shannon : Soit une source $S$ comportant $Q$ symboles (niveaux de gris), et $p_i$ la probabilité d'apparition du symbole $i = 1, 2, \cdots, Q$, (niveau) alors l'entropie $H$ de la source $S$ est définie par :

$$H(S) = -\sum_{i=1}^{Q} p_i \times log_2(p_i) \tag{3.57}$$

avec $p_i = o_i / To$

Dans le cas d'une distribution uniforme, l'entropie $H(S)$ est maximale et est donnée par :

$$H_{Max} = log_2(Q) = 8 \tag{3.58}$$

Donc, si la valeur de l'entropie est très proche de $H_{Max}$, cela signifie que les données de la source possèdent une distribution quasi uniforme et le crypto-système fournissant de telles données peut résister à l'attaque statistique.

Afin que ce test soit significatif, d'abord, nous décomposons l'image originale/chiffrée en sous blocs de taille 16 x 16 pixels puis nous calculons l'entropie de chaque sous bloc, et l'entropie moyenne des différentes sous blocs.

Nous donnons dans le tableau 3.7, les résultats obtenus des tests de chi-carré et de l'entropie moyenne, pour différents mode cryptographique, sur l'image Lena 512x512x3, par le premier crypto-système SPN-1, dans les deux cas de figures de la couche de permutation : carte Skew tent (T) et carte Cat (C). La taille du bloc est fixé à $Tb$ =256. Des résultats similaires sont obtenus par le deuxième crypto-système SPN-2.

Tableau 3.7 : Résultats statistique de chi-carré et de l'entropie moyenne, pour différents modes cryptographiques, pour l'image Lena 512x512x3 et pour une taille de bloc *Tb*=256

| Mode | Test chi-carré | Entropie de l'image originale | Entropie de l'image chiffrée |
|---|---|---|---|
| ECB-T | 244.49 | 5.6822 | 7.9998 |
| CBC-T | 245.44 | - | 7.9998 |
| OFB-T | 243.48 | - | 7.9998 |
| CTR-T | 245.72 | - | 7.9997 |
| ECB-C | 225.92 | - | 7.9998 |
| CBC-C | 234.27 | - | 7.9998 |
| OFB-C | 222.18 | - | 7.9997 |
| CTR-C | 223.84 | - | 7.9998 |

Enfin, afin de consolider les résultats précédents, nous donnons dans le tableau 3.8 les résultats des tests chi-carré et entropie moyenne sur différentes images chiffrées.

Tableau 3.8 : Chi-carré et entropie moyenne sur différentes images chiffrées

| Image chiffrée | Test chi-carré | Entropie |
|---|---|---|
| Boat | 243.6 | 7.9998 |
| Finger | 251.4 | 7.9998 |
| Baboon | 224.8 | 7.9998 |
| Barbara | 251.9 | 7.9998 |
| Goldhill | 244.1 | 7.9998 |
| Pepers | 214.5 | 7.9999 |

Tous ses résultats montrent que le crypto-système utilisé, résiste contre les attaques statistiques.

Sur l'image Lena de la figure 3.69, nous donnons dans les figures 3.70, 3.71 et 3.72, un exemple de résultats respectivement : image chiffrée par SPN-1 utilisant la carte Skew tent, histogramme de l'image claire de Lena et histogramme de l'image chiffrée de Lena.

Fig. 3.69: Image claire de Lena 512x512x3

Fig. 3.70 : Image chiffrée de Lena

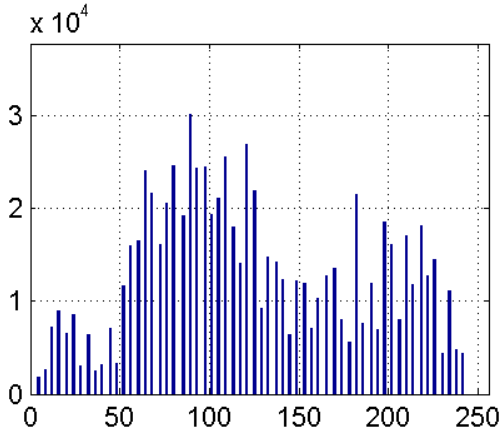



Fig. 3.71 : Histogramme image claire de Lena

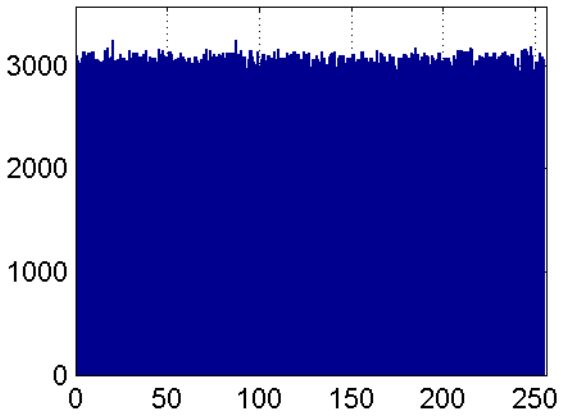



Fig. 3.72: Histogramme image chiffrée de Lena.

*2.7.3.2 Coefficient de corrélation des pixels adjacents*

En général, dans une image originale, chaque pixel est fortement corrélé avec ses pixels adjacents dans les directions horizontale, verticale et diagonale. Un crypto-système idéal devrait produire des images chiffrées sans aucune corrélation entre les pixels adjacents. Nous avons calculé les coefficients de corrélation des pixels adjacents pour les directions horizontale, verticale et diagonale, respectivement. A ce propos, nous avons pris de façon aléatoire dans l'image claire et l'image chiffrée, 2000 paires de pixels adjacents, pour chacune des trois directions comme suit :

Pour chaque pixel choisi (au nombre 2000) de coordonnées $(i, j)$ nous formons 4 vecteurs $V, V_H, V_V, V_D$, contenants les niveaux de gris des pixels qui se trouvent aux positions $(i, j), (i+1, j), (i, j+1), (i+1, j+1)$ respectivement. Les coefficients de corrélation dans les 3 directions sont : $\rho_{V,V_H}$, $\rho_{V,V_V}$ et $\rho_{V,V_D}$.

Dans le tableau 3.9 suivant, nous donnons les valeurs des coefficients de corrélation des images en clair de Lena et de Baboon

Tableau 3.9 : Coefficients de corrélation des images en claires de Lena et de Baboon

| | **Coefficient de corrélation** | | |
|---|---|---|---|
| **Direction** | **Horizontale** | **Verticale** | **Diagonale** |
| Lena | 0.97411 | 0.98544 | 0.96263 |
| Baboon | 0.9662 | 0.984 | 0.9497 |

Dans le tableau 3.10, nous présentons les valeurs des coefficients de corrélation de l'image chiffrée de Lena 512x512x3, obtenus par le premier crypto-système SPN-1 dans les deux cas de figures de la couche de permutation : carte Skew tent (T) et carte Cat (C). La taille du bloc est fixée à $Tb$ =256. Des résultats similaires sont obtenus par le deuxième crypto-système SPN-2.

Tableau 3.10 : Résultats de la corrélation horizontale, verticale et diagonale, pour différents mode cryptographique de l'image chiffrée Lena 512x512x3, avec *Tb*=256

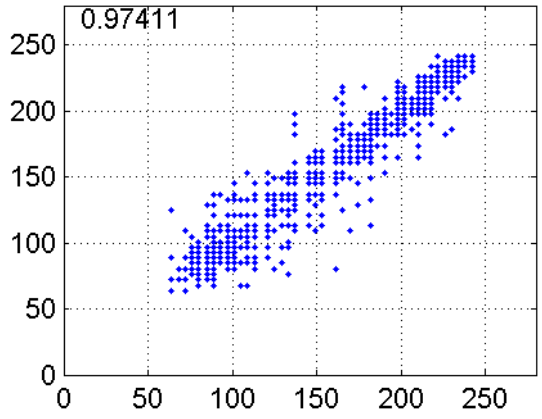
| | **Coefficient de corrélation** | | |
|---|---|---|---|
| **Direction** | **Horizontale** | **Verticale** | **Diagonale** |
| Mode -ECB-T | 0.00395 | 0.0232 | 0.0173 |
| Mode -CBC-T | 0.02485 | 0.01052 | 0.0033 |
| Mode -OFB-T | 0.01991 | 0.038536 | 0.01211 |
| Mode -CTR-T | -0.0079 | -0.0455 | -0.0081 |
| Mode -ECB-C | -0.0191 | -0.000965 | 0.01194 |
| Mode-CBC-C | -0.0313 | -0.00812 | -0.0210 |
| Mode-OFB-C | -0.01478 | -0.00817 | -0.0263 |
| Mode -CTR-C | -0.036674 | 0.065147 | 0.052871 |

A titre comparatif, nous donnons dans le tableau 3.11, les résultats de la corrélation des différents algorithmes sur l'image Lena.

Tableau 3.11 : Coefficients de corrélation des différents algorithmes sur l'image Lena. Ici, $Tb$=128 bits.

| | Coefficient de Correlation | | |
|---|---|---|---|
| **Algorithme** | **Horizontal** | **Vertical** | **Diagonal** |
| SPN-1-T | 0.024853 | 0.01052 | 0.003305 |
| AES | -0.015 | 0.028 | -0.027 |
| ECKBA | -0.0119 | -0.019 | -0.038 |
| Kumar | 0.00499 | -0.002 | 0.0083 |
| Yang | -0.003 | -0.016 | 0,0178 |

On voit clairement d'après tous ces résultats, que la corrélation dans les trois directions de l'image chiffrée est proche de la valeur zéro, quel que soit l'algorithme utilisé. Le résultat de la corrélation dans les 3 directions, pour l'image en clair et l'image chiffrée correspondante, peut aussi être évalué visuellement, en traçant les couples de niveaux de gris : $\{V, V_H\}, \{V, V_V\}$, et $\{V, V_D\}$. On voit clairement d'après les figures 3.73-b, 3.74-b et 3.75-b, que la distribution des niveaux de gris dans l'image chiffrée dans les 3 directions ne présente pas de corrélation. Ce n'est pas le cas de l'image en clair où la distribution des niveaux de gris montre nettement l'existence d'une corrélation forte, voir figures 3.73-a, 3.74-a et 3.75-a.



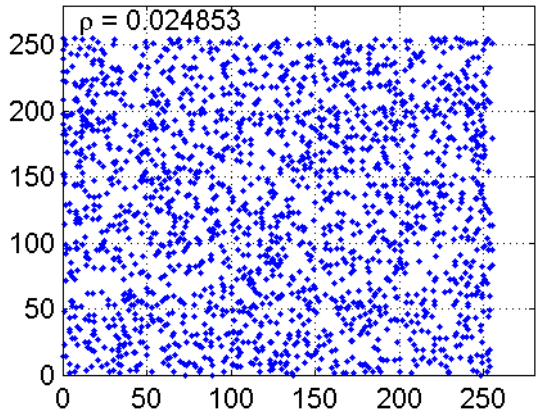




Fig. 3.73:-a. Distribution $V_H$ en fonction de $V$ dans l'image en clair

-b. Distribution $V_H$ en fonction de $V$ dans l'image chiffrée

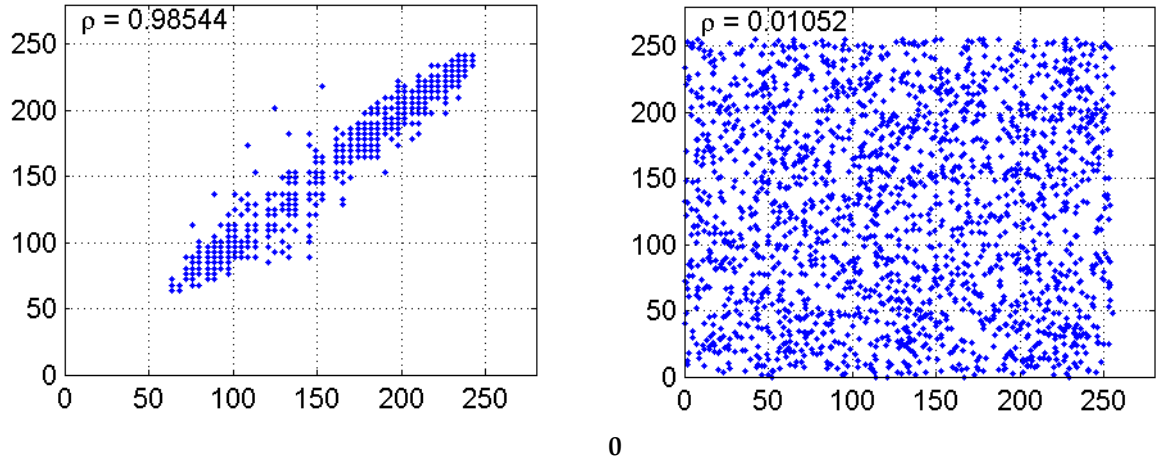



Fig. 3.74: -a. Distribution $V_V$ en fonction de $V$ dans l'image en claire -b. Distribution $V_V$ en fonction de $V$ dans l'image chiffrée

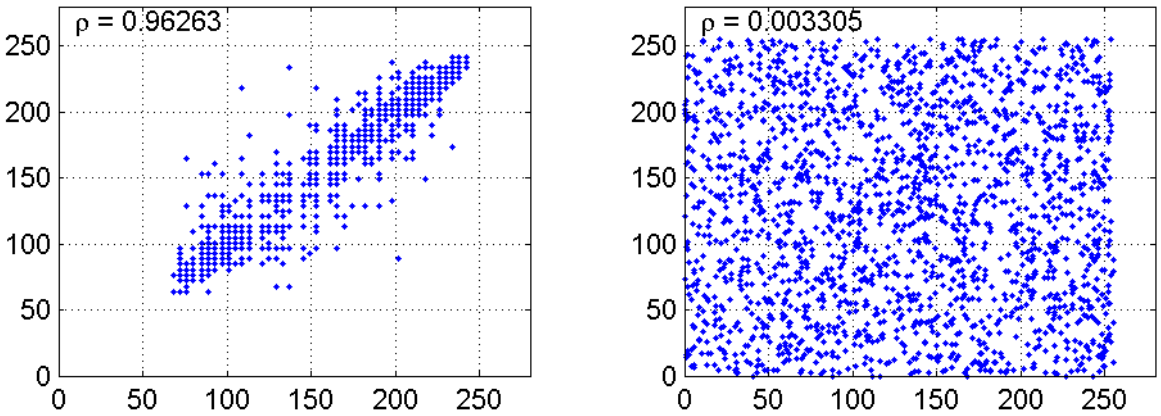



Fig. 3.75: -a. Distribution $V_D$ en fonction de $V$ dans l'image en claire -b. Distribution $V_D$ en fonction de $V$ dans l'image chiffrée

**Remarque :**

Dans la littérature sur les crypto-systèmes basés chaos, certains auteurs utilisent les paramètres NPCR et UACI pour quantifier les performances entre :

- image en clair et image chiffrée
- deux images chiffrées dont les images en clair diffèrent d'un seul bit. Dans notre test, nous avons changé le bit de poids faible du premier bloc de l'image en claire (les résultats sont donnés par NPCR-PS et UACI-PS voit tableau 3.12).

Le tableau suivant montre les résultats obtenus pour les différents modes cryptographiques. On voit clairement d'après les valeurs du NPCR-PS et UACI-PS, que seul le mode chaînage CBC, semble efficace. En fait, ce n'est pas vrai, car ce genre de test doit se faire sur les blocs et non sur les images, puisque la sécurité est au niveau bloc et non au niveau de l'image.

Quel que soit le mode utilisé sur les blocs, un bit changé dans un bloc en clair, engendre 50 % de changement de bits dans le bloc chiffré correspondant (effet d'avalanche).

Tableau 3.12 : Résultats de la sensibilité au texte en clair pour l'image Lena 512x512x3, avec une taille de bloc égale à 256 bits

| MODE | NPCR | UACI | NPCR-PS | UACI-PS |
|---|---|---|---|---|
| ECB-T | 99.617 | 30.519 | 0.00406 | 0.00138 |
| ECB-C | 99.616 | 30.493 | 0.00101 | 0.00032 |
| CBC-T | 99.606 | 30.523 | 99.603 | 33.459 |
| CBC-C | 99.612 | 30.508 | 99.597 | 33.372 |
| OFB-T | 99.618, | 30.507 | 1.27 e-4 | 4.986e-7 |
| OFB-C | 99.622, | 30.496, | 1.27 e-4 | 4.986e-7 |
| CTR-T | 99.618 | 30.488 | 1.27 e-4 | 4.986e-7 |
| CTR-C | 99.621 | 30.522 | 1.27 e-4 | 4.986e-7 |

### 3.7.4 Performances en temps de calcul :

La vitesse de chiffrement/déchiffrement d'un crypto-système, est l'une des caractéristiques fondamentale (avec la robustesse) pour quantifier et comparer les performances. Dans ce paragraphe nous donnons, pour les crypto-systèmes proposés, le temps de calcul moyen (sur 10000 itérations) pour une itération de chiffrement/déchiffrement sur un bloc de taille $Tb$, ainsi que le temps moyen pris par les processus des différentes couches. Aussi, nous donnons le temps moyen (sur 10000 itérations), estimé à part, de génération de la clé dynamique globale. Ces calculs sont donnés dans l'environnement logiciel et matériel suivants : Matlab 2008 et micro ordinateur Intel Core 2 Duo 2.1 GHz CPU avec 2 GB RAM Intel, sous Windows XP.

Dans le tableau 3.13 suivant, nous donnons les performances en temps (en milli seconde) du premier crypto-système SPN-1 dans ces deux variantes, selon la couche de permutation utilisée (T pour Skew tent, et C pour Cat), pour ($rs = 4, rp = 1, nb = na = 4, Tb = 256$). Dans ce tableau les lettres : $G, A, S, P$ signifient respectivement, processus de génération de la clé dynamique globale, addition des clés d'itérations, substitution, et permutation.

Tableau 3.13 : Performances en temps *(ms)* du premier crypto-système SPN-1

| SPN-1 | Temps moyen (m*s*) d'une itération | G(m*s*) | A(m*s*) | S(ms) | P(ms) | Bin2dec (m*s*) | Dec2bin (m*s*) |
|---|---|---|---|---|---|---|---|
| **SPN-1-C : Chiffrement** | 3.2120 | 1.3525 | 0.0156 | 0.4492 | 1.212 | 0.9859 | 0.1341 |
| **SPN-1-C : Chiffrement en %** | | **32.5959** | **0.3760** | **10.8259** | **29.2097** | **23.7606** | **3.2319** |
| **SPN-1-T : Chiffrement** | 3.5327 | 1.3416 | 0.0249 | 0.4368 | 1.3712 | 0.8361 | 0.1232 |
| **SPN-1-T : Chiffrement en %** | | **32.4544** | **0.6024** | **10.5665** | **33.1704** | **20.2259** | **2.9803** |
| **SPN-1-C : Déchiffrement** | 3.41237 | 1.3556 | 0.0265 | 0.9500 | 1.1793 | 0.8455 | 0.0998 |
| **SPN-1-C : Déchiffrement en %** | | **30.4171** | **0.5946** | **21.3162** | **26.4613** | **18.9714** | **2.2393** |
| **SPN-1-T : Déchiffrement** | 4.0856 | 1.3696 | 0.03744 | 1.0857 | 1.5693 | 0.8580 | 0.1107 |
| **SPN-1-T : Déchiffrement en %** | | **27.2246** | **0.7442** | **21.5813** | **31.1942** | **17.0551** | **2.2005** |

Nous remarquons d'après les résultats obtenus que, quelle que soit la version utilisée du crypto-système, les processus les plus coûteux en temps de chiffrement/déchiffrement par ordre décroissant sont : la génération de la clé dynamique globale, les permutations, la conversion binaire vers décimal, la substitution, la conversion décimale binaire, et addition des clés d'itérations.

Globalement le crypto-système SPN-1-C est plus rapide (10%) que celui SPN-1-T. Cette différence vient du processus de permutation utilisé. En effet, la carte Skew tent utilise une opération de division, ce n'est pas le cas de la carte Cat.
Le déchiffrement est plus coûteux que le chiffrement, cette différence découle essentiellement du processus de substitution inverse.

Dans le tableau 3.14, nous donnons les performances en temps (m*s*) du SPN-1-T, dans le cas de $Tb = 128$ bits, permettant de comparer les résultats avec l'AES.

Tableau 3.14 : Performances en temps *(ms)* du premier crypto-système SPN-1 pour $Tb$=128

| SPN-1 | Temps moyen (m*s*) d'une itération | G(m*s*) | A(m*s*) | S(m*s*) | P(m*s*) | Bin2dec (m*s*) | Dec2bin (m*s*) |
|---|---|---|---|---|---|---|---|
| **SPN-1-T : Chiffrement** | 2.25421 | 1.0639 | 0.0156 | 0.3993 | 0.7581 | 0.5038 | 0.0764 |
| **SPN-1-T : Chiffrement en %** | | **37.7658** | **0.5538** | **14.1742** | **26.9107** | **17.8836** | **17.8836** |
| **SPN-1-T : Déchiffrement** | 3.2604 | 1.1466 | 0.05304 | 0.7687 | 1.1764 | 0.4836 | 0.09672 |
| **SPN-1-T : Déchiffrement en %** | | **30.780** | **1.4239** | **20.6359** | **31.5807** | **12.9823** | **12.9823** |

Dans le tableau 3.15, nous donnons les performances en temps (en milli seconde) du deuxième crypto-système SPN-2, dans ces deux variantes selon la couche de diffusion utilisée : statique SPN2-B, dynamique à base de la carte cat SPN2-C, et pour $Tb = 256$, correspondant au cas d'une matrice de diffusion $D$ de taille 32 x 32. Le processus optionnel de la permutation est fait sur les octets.

Tableau 3.15 : Performances en temps *(ms)* du deuxième crypto-système SPN-2, pour $Tb$=256

| SPN-2 | Temps moyen (m*s*) d'une itération | G(m*s*) | A(ms) | S(m*s*) | D(m*s*) | P(m*s*) |
|---|---|---|---|---|---|---|
| **SPN-2-C : Chiffrement** | 2.1372 | 1.27920 | 0.0156 | 0.2808 | 0.4056 | 0.1404 |
| **SPN-2-C : Chiffrement en %** | | 60.2934 | 0.73529 | 13.2352 | 19.1176 | 6.6176 |
| **SPN-2-B : Chiffrement** | 1.8564 | 1.37280 | 0.0468 | 0.2028 | 0.1092 | 0.1248 |
| **SPN-2-B : Chiffrement en %** | | 73.9495 | 2.5210 | 10.9241 | 5.8823 | 6.7226 |
| **SPN-2-C : Déchiffrement** | 2.4648 | 1.29480 | 0.0156 | 0.5304 | 0.4524029 | 0.156 |
| **SPN-2-C : Déchiffrement en %** | | 52.8662 | 0.63694 | 21.6560 | 18.47133 | 6.3694 |
| **SPN-2-B : Déchiffrement** | 2,2116 | 1.18560 | 0.0312 | 0.5268 | 0.2808 | 0.1872 |
| **SPN-2-B : Déchiffrement en %** | | 47.2049 | 1.24223 | 32.9192 | 11.1801 | 7.4534 |

Le tableau 3.16, contient les mêmes types de résultats que le tableau précédent, mis à part que la taille du bloc est fixée à $Tb = 128$, qui correspond au cas d'une matrice de diffusion de taille 16 x 16.

Tableau 3.16 : Performances en temps *(ms)* du deuxième crypto-système SPN-2, pour *Tb*=128

| SPN-2 | Temps moyen (m*s*) d'une itération | G(m*s*) | A(ms) | S(m*s*) | D(m*s*) | P(m*s*) |
|---|---|---|---|---|---|---|
| **SPN-2-C : Chiffrement** | 1.659 | 1.1013 | 0.0296 | 0.2527 | 0.170 | 0.0982 |
| **SPN-2-C : Chiffrement en %** | | 66.66 | 1.794 | 15.297 | 10.29 | 5.949 |
| **SPN-2-B : Chiffrement** | 1.4898 | 1.0701 | 0.0156 | 0.2355 | 0.0889 | 0.067 |
| **SPN-2-B : Chiffrement en %** | | 72.4392 | 1.055 | 15.945 | 6.019 | 4.5406 |
| **SPN-2-C : Déchiffrement** | 2.934 | 1.496 | 0.05304 | 0.833 | 0.318 | 0.209 |
| **SPN-2-C : Déchiffrement en %** | | 51.4209 | 1.823 | 28.632 | 10.938 | 7.1849 |
| **SPN-2-B : Déchiffrement** | 2,34 | 1.3322 | 0.00468 | 0.639 | 0.115 | 0.2262 |
| **SPN-2-B : Déchiffrement en %** | | 57.4697 | 0.2018 | 27.59 | 4.979 | 9.7577 |

Nous remarquons que le crypto-système SPN-2-B utilisant une matrice de diffusion binaire est plus rapide que SPN-2-C qui utilise une matrice de diffusion dynamique à base d'une carte cat multidimensionnelle.

Le crypto-système SPN-2 est plus rapide que le premier crypto-système SPN-1, quelque soit la variante considérée. Par exemple, dans le cas où $Tb = 128$ bits, SPN-2-B est pratiquement 2 fois plus rapide que SPN-1-T.

A titre comparatif, nous donnons dans le tableau 3.17, les performances en temps de l'algorithme AES, avec $Tb = 128$ bits. Ces résultats montrent que l'AES est pratiquement 2 fois plus lent que le SPN-1-T et 3 fois plus lent que le SPN-2-B.

Tableau 3.17 : Performances en temps *(ms)* de l'AES pour *Tb*=128

| AES | Temps moyen (m*s*) d'une itération | E(m*s*) | A(m*s*) | S(m*s*) | D(m*s*) | P(m*s*) |
|---|---|---|---|---|---|---|
| **Chiffrement** | 3.977 | 0.25329 | 0.0144958 | 0.03509 | 3.65791 | 0.01716 |
| **Chiffrement en %** | | 6.367% | 0.364 | 0.882 | 91.9544 | 0.431 |
| **Déchiffrement** | 6.87561 | 0.04348 | 0.0164032 | 0.02708 | 6.76956 | 0.01907 |
| **Déchiffrement en %** | | 0.63249 | 0.2385702 | 0.39391 | 98.4576 | 0.27740 |

## 3.8 Conclusion

Dans ce chapitre, nous avons présenté en détail les deux crypto-systèmes basés chaos que nous avons conçus et réalisés. Ces crypto-systèmes utilisent la structure SPN adaptée au calcul en parallèle.

Les deux crypto-systèmes s'appuient sur les mêmes couches d'addition de clé, et de substitution. Ils se différencient par la couche de permutation qui est réalisée sur les bits pour le premier crypto-système SPN-1 et sur les octets pour le deuxième crypto-systèmes SPN-2. Ce dernier contient en plus une couche de diffusion, mais ne nécessite pas des opérations de conversion décimal vers binaire et binaire vers décimal. De ce fait, il est une fois et demi plus rapide que le SPN-1 et 3 fois plus rapide que l'AES et d'autres crypto-systèmes basés chaos de la littérature.

Afin de quantifier les performances de ces crypto-systèmes, nous avons d'abord, présenté la quasi-totalité des outils utilisés pour mesurer les performances des couches et des crypto-systèmes, ensuite, nous avons quantifié les performances couche par couche et sur les crypto-systèmes proposés.

Les résultats obtenus, montrent que les crypto-systèmes proposés résistent à toutes les attaques cryptographiques connues de la littérature. En plus ils sont assez rapides comparés à l'algorithme standard AES et à d'autres crypto-systèmes basés chaos de la littérature. Par ailleurs, leurs structures sont adéquates pour une implémentation matérielle sur cartes FPGA par exemple.

[Adams et al, 1993] C. M. Adam, S.E. Tavares, "Designing S-boxes for ciphers resistant to differential cryptanalysis", Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography, Rome, Italy, pp. 181–190, 1993.

[Amigo et al, 2007] J. M. Amigó, L. Kocarev , J. Szczepanski, "Theory and practice of chaotic cryptography", Physics Letters A, Vol. 366, pp. 211–216 , 2007.

[Amin et al, 2010] M. Amin , O. S. Faragallah , A. A. Abd El-Latif, "A chaotic block cipher algorithm for image cryptosystems", Commun Nonlinear Sci Numer Simulat, Vol. 15 , pp. 3484–3497, 2010.

[Aoki et al, 2001] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita, "Camellia: A 128-bit Block Cipher Suitable for Multiple Platforms – Design and Analysis", Selected Areas in Cryptography — 7th Annual International Workshop, SAC2000, Springer LNCS, pp. 39–56, 2001.

[Biham et Shamir, 1991] E. Biham, A. Shamir. "Differential cryptanalysis of DES-like cryptosystems". Journal of Cryptology, Vol. 4, no. 1, 1991.

[Brookes, 2005] M. Brookes, "The matrix reference manual", 2005, http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html.

[Canteaut, 2001] A. Canteaut, "Cryptographic functions and design criteria for block ciphers", In Proceedings of Indocrypt, Springer, 2001.

[Chen et al, 2004] G. Chen, Y. Mao, Charles K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps ", Chaos, Solitons Fractals, Vol. 21, pp. 749-761, July 2004.

[Dawson et al, 1991] M. H. Dawson, S. E. Tavares, "An expanded set of S-box design criteria based on information theory and its relation to differential-like attacks", Proceedings of EUROCRYPT» 91 : Advances in Cryptology, pp. 352–367, 1991.

[FIP PUB 197, 2001] National Institute of Standards and Technology: "FIPS-197: Advanced Encryption Standard (AES) ", November 2001.

[Fridrich, 1998] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps", International Journal of Bifurcation Chaos, Vol. 8, pp. 1259–1284, 1998.

[Ibrahim et al, 2005] S. Ibrahim, M. A.Maarof, N .B. Idris , "Avalanche Analysis of Extended Feistel Network", Proceedings of the Postgraduate Annual Research Seminar 2005, pp. 265-269, 2005.

[Jakimoski et al, 2001] G. Jakimoski, L. Kocarev, "Chaos and cryptography: block encryption ciphers based on chaotic maps", IEEE Transactions on Circuits and Systems I Fundamental Theory and Applications, Vol. 48 (2), pp. 163-169, 2001.

[Knudsen et al, 1998] Lars R. Knudsen, "Block Ciphers —A Survey", Lecture Notes in Computer Science, 1998, Vol. 1528, pp. 18-48, 1998.

[Koo et al, 2003] B. W. Koo, H. S. Jang, J. H. Song, "Constructing and Cryptanalysis of a 16 × 16 Binary Matrix as a Diffusion Layer", WISA2003, Springer Verlag LNCS, Vol. 2908, pp. 489-503, 2003.

[Koo et al, 2006] B. Koo, H. Jang, J. Song, "On constructing of a 32 × 32 binary matrix as a diffusion layer for a 256-bit block cipher", Proc. ICISC 2006, Springer Verlag LNCS, Vol. 4296, pp. 51–64, 2006.

[Kwon et al, 2003] D. Kwon, J. Kim, S. Park, et al. , " New block cipher: ARIA", Proc. ICISC 2003, pp. 432–445, 2004 .

[Kwon et al, 2005] D. Kwon, S. H., Sung, J. H. Song, S. Park, "Design Of Block Ciphers and coding theory", Trends in Mathematics Information Center for Mathematical Sciences Vol. 8, pp. 13-20, 2005.

[Liu et al, 2008] Y. Liu; W.K.S. Tang, H.S. Kwok, "Formulation and analysis of high-dimensional chaotic maps", ISCAS 2008. IEEE International Symposium on Circuits and Systems, pp. 772-775, May 2008.

[Masuda et al, 2002] N. Masuda, K. Aihara, "Cryptosystems with discretized chaotic maps", IEEE Transactions on Circuits and Systems I, Vol. 49, pp. 28-40, 2002.

[Masuda et al, 2006] N. Masuda, G. Jakimoski, K. Aihara, L. Kocarev , "Chaotic block ciphers: from theory to practical algorithms", IEEE Transactions on Circuits and Systems I Regular Papers, Vol. I 53, pp. 1341–1352, 2006.

[Matsui, 1993] M. Matsui, "Linear cryptanalysis method for DES cipher", Proc. Eurocrypt 93, Vol. 765 of LNCS, pp. 386-397, Springer, 1993.

[Mister et al, 1996] S. Mister, C. Adams, "Practical S-Box Design", Workshop on Selected Areas in Cryptography (SAC-96), pp. 61–76, 1996.

[Noura et al, 2011] H. Noura, S. el Assad, C. Vladeanu, D. Caragata, "An efficient and secure SPN cryptosystem based on chaotic control parameters", 6th International Conference on Internet Technology and Secured Transactions, ICITST-2011, Abu Dhabi, pp. 226-231, December 2011.

[Rivest et al,1998 ] R.L. Rivest, M.J.B. Robshaw, R.Sidney, Y.L. Yin, "The RC6 Block Cipher. v1.1», August 1998.

[Shannon, 1949] C. E. Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol. 28, pp. 656-715, October 1949.

[Socek et al, 2005 ] D. Socek, S. Li, S. S. Magliveras, B. Furht, " Enhanced 1-D Chaotic Key Based Algorithm for Image Encryption", IEEE Security and Privacy for Emerging Areas in Communications Networks, 2005.

[Tang et al, 2005] G.P. Tang, X.F. Liao, "A method for designing dynamical S-boxes based on discretized chaotic map", Chaos, Solitons & Fractals, Vol. 23, pp. 1901–1909, 2005.

[Tang et al, 2010] Y. Tang, Z. Wang, J. Fang, "Image encryption using chaotic coupled map lattices with time-varying delays", Communications in Nonlinear Science and Numerical Simulation, Vol. 15, no. 9, pp. 2456-2468, 2010 .

[Webster et al, 1986] A.F. Webster, S.E. Tavares, "On the design of S-boxes. In Lecture notes in computer sciences on Advances in cryptology", CRYPTO 85, Springer-Verlag New York, Inc. New York, NY, Vol. 218, pp. 523–534, USA, 1986.

[Xiao et al, 2008] D. Xiao, X. Liao, S. Deng, "Parallel keyed hash function construction based on chaotic maps", Physics Letters A, Vol. 372, no. 26, pp. 4682-4688, 2008.

# *Chapitre 4 : Conception et réalisation d'une fonction de hachage basée chaos sans ou avec clé secrète performante*

## 4.1 Introduction :

Dans ce chapitre, nous abordons la conception et la réalisation des fonctions de hachage basée chaos, sans ou avec clé secrète.

Une fonction de hachage est une composante fondamentale dans un système de sécurité de l'information et joue un rôle important dans des applications cryptographiques pour la vérification de l'intégrité des données, l'authentification symétrique de source de messages, et la signature numérique.

Une fonction de hachage est à sens unique, elle compresse un bloc d'entrée de taille finie $m$ (partie d'un message de longueur arbitraire) en un bloc de taille fixe $n$, nommé empreinte digitale ou condensat, plus petite ou égale à la taille $m$ du bloc d'entrée. Les fonctions de hachage ont été inventées par Knuth dans les années 1950 [Knuth ,1998].

Depuis l'apparition d'algorithmes de chiffrement avec clé publique et des schémas de signature numériques, la construction de fonctions de hachage est devenue importante.

Avant sa transmission, un message haché puis signé permet d'assurer en réception, l'intégrité du message, l'authentification de sa source et la non répudiation de l'expéditeur. Autrement, sans authentification ou signature, un intrus peut intercepter le message transmis, le modifier et le renvoyer avec la valeur de hachage recalculée pour le message modifié.

La structure d'une fonction de hachage, est pratiquement identique à celle d'un algorithme de chiffrement symétrique en mode CBC, dont la sortie ne comprend que le dernier bloc chiffré. Ce dernier est alors l'empreinte digitale. Pour cela, une fonction de hachage est à sens unique, car il est pratiquement impossible qu'à partir de l'empreinte digitale de retrouver le message d'origine. Cette structure permet l'authentification de la source du message (MAC), mais elle est très lente comparée à la structure de Merkle-Damgard [Merkle, 1989], [Damgard, 1989].

Les fonctions de hachage sont groupées en deux classes :

- Fonctions de hachage sans clé sécrète : originalement, sont utilisées pour assurer le service de l'intégrité des données. En outre, leurs sorties sont chiffrées avec un algorithme de chiffrement asymétrique pour réaliser la signature numérique.
- Fonctions de hachage avec clés sécrète : elles sont utilisées pour assurer l'authentification des messages [Menezes et al, 1996]. Les fonctions de hachage avec clé, prennent deux entrées : un message et une clé secrète, pour produire une empreinte numérique dépendante de la clé et du message. Elles sont utilisées dans les protocoles de sécurité de la couche de transport comme le SSL (Secure Socket Layer)

[Freier et Karlton, 2011], et l'IPSEC (Internet Protocol Security) [Kent et Atkinson, 1998], etc.

Les fonctions de hachage les plus répandues sont basées sur la structure itérative de Merkle-Damgard présentée plus loin.

L'organisation de ce chapitre se déroule comme suit : Dans la section 2, nous introduisons des généralités sur les fonctions de hachage, leurs propriétés et le modèle itératif de Merkle-Damgard. Dans la section 3, nous présentons la fonction de hachage chaotique proposée utilisant deux variantes pour la couche de diffusion (binaire et cat multidimensionnelle). Dans la section 4, nous donnons les résultats de mesure des nombres d'itérations nécessaires $rl$ et $rt$ des cartes chaotiques Logistique et Skew tent. Dans la section 5, nous analysons, les performances et la sécurité de la fonction de hachage proposée, avant de conclure.

## 4.2 Fonctions de hachage: Généralités et propriétés

La famille standard des fonctions de hachage la plus connue est la SHA (Secure Hash Standard) qui est développée par la NSA et certifiée par le NIST. Les fonctions SHA-256 et SHA-512 sont à ce jour, les plus sûres et les plus utilisées [FIPS 180-1, 1995]. Par ailleurs, NIST a ouvert un concours public pour développer un nouvel algorithme de hachage cryptographique pour octobre 2012, appelé "SHA-3", et à ce jour cinq candidats sont retenus.

A l'état actuel, toutes les fonctions de hachage standard ou basées chaos, utilisent le modèle itératif de Merkle-Damgard. Elles se différentient entre elles, par la manière de réaliser la fonction de compression $h$. Cette dernière est basée sur l'architecture SPN des algorithmes de chiffrement [Xiao et al, 2008], [Xiao et al, 2009], [Deng et al, 2010], [Wang et al, 2012].

Parmi les différentes fonctions de compression basées chaos, utilisées dans la littérature, celles de [Yang, 2010], et [Xiao et al, 2008] nous semblent intéressantes. Cependant, les deux structures en question présentent des faiblesses manifestes. En effet, la structure de Yang, utilise quatre couches de diffusion locale (pour un bloc de taille $Tb = 128$), et un nombre d'itérations aussi égal à quatre pour assurer l'effet d'avalanche. La valeur de hachage finale est obtenue en extrayant 32 bits de poids fort de chacune des 4 dernières sorties.

La structure utilisée par Xiao, a l'avantage par rapport à la structure de Yang de n'utiliser qu'une seule itération pour chaque bloc. Cependant, la couche d'addition de clés (OUex) est réalisée après la fonction d'itération, ce qui a permis de cryptanalyser la fonction en question [Guo et al, 2009]. Suite à cela, [Xiao et al, 2009-2] ont proposé une amélioration de leur fonction de hachage.

### 3.2.1 Propriétés des fonctions de hachage

Une fonction de hachage est à sens unique et peut être sans clé ou avec clé, dans ce cas, en plus du message d'entrée, il y a la clé secrète. La fonction de hachage doit être rapide.

L'empreinte numérique $H_{nb}$ est de taille fixée à $n$ bits (voir figure 4.1)

$$H_{nb} = h(H_{nb-1}, M_{nb})$$

Les fonctions de hachage possèdent plusieurs propriétés utiles en cryptographie [Bakhtiari, 1995]. Les principales sont la résistance aux attaques recherchant des collisions, des préimages ou des secondes préimages.

- collision : trouver deux messages distincts $M1$ et $M2$, tels que $h(M1) = h(M2)$.
- préimage : étant donné un haché $H1$ choisi aléatoirement, trouver un message $M1$ tel que $H(M1) = H1$.
- seconde préimage : trouver une deuxième entrée, qui a la même sortie qu'une entrée indiquée. c'est à dire étant donné un message $M1$ choisi aléatoirement, trouver un message distinct $M2$ tel que : $H2 = H1$.

Il doit être impossible pour un attaquant de trouver une collision, une préimage ou une seconde préimage. Cependant, puisque la taille d'entrée de la fonction de hachage est arbitrairement grande, des collisions existent nécessairement.
La fonction de hachage avec clé $H_K(M)$ doit en plus avoir assez d'espace de clés sécrètes, pour contrecarrer la recherche exhaustive, et aussi être sensible à la clé secrète.

### 3.2.2 Algorithme de Merkle-Damgård

En 1989, Ralph Merkle et Ivan Damgård proposèrent indépendamment un algorithme de hachage dont la structure est présentée dans la figure 4.1. Le message $M$ à hacher est tout d'abord adapté en y ajoutant un rembourrage pour ramener la taille $M$ du message à hacher à un multiple de $m$, la fonction de compression prenant en entrée des blocs de message de taille fixe $m$. Le rembourrage consiste à rajouter tout d'abord, au dernier bloc de $M$ un bit à 1 puis $u$ bits à 0, où $u$ est le plus petit nombre positif ou nul telle que la longueur finale du message soit égale à $m - v \ (mod\ m)$ (typiquement $v = 64$ bits). Ensuite, un deuxième rembourrage intervient en ajoutant un bloc de $v$ bits indiquant la représentation en base binaire de la taille de message $M$. La taille maximale du message pouvant être hachée est donc limitée à $2^{v}$ bits, et en pratique ce nombre est assez grand. Le message rembourré est ensuite divisé en $nb$ blocs de message $M_i$ de m bits chacun, qui serviront à mettre à jour la variable de chaînage $H_{i-1}$ pour donner $H_i$ à l'aide de la fonction de compression $h$ :

$$H_i = h(H_{i-1}, M_i) \qquad (4.1)$$

La variable de chaînage initiale $H_0$ est la clé secrète $K_0$ de la fonction de hachage, et la dernière variable de chaînage $H_{nb}$ est le haché final ou empreinte digitale. En général, la taille $m$ de l'état interne de la fonction de hachage est identique à la taille $n$ du haché final. Si la taille de l'état interne est plus grande que celle du haché, on doit alors, effectuer une troncature pour obtenir la bonne taille en sortie. Il est aussi possible d'appliquer une fonction de sortie sur $H_{nb}$. Le processus entier est décrit en figure 4.1. Le mode opératoire utilisé, comme dans toutes les fonctions de hachage, est le mode CBC (enchaînement des blocs) afin d'atteindre une haute sensibilité au changement de texte en clair et de la clé sécrète.

Dans la suite, sauf mention contraire, nous considérons que la taille de l'état interne de la fonction de hachage est égale à celle du haché.

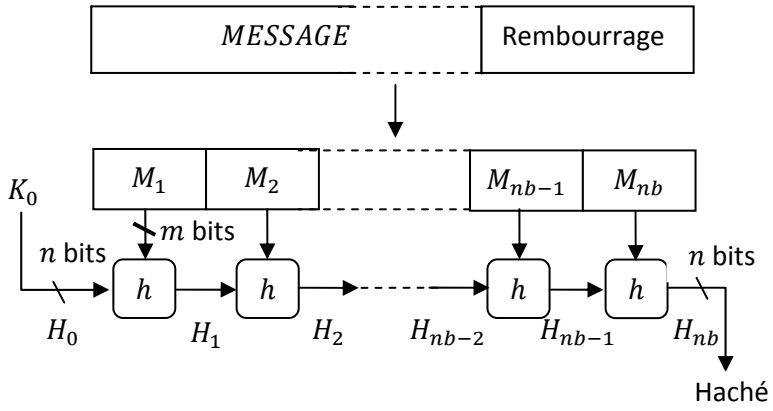



Fig. 4. 1 : Algorithme de Merkle-Damgård

Cette structure permet l'authentification d'image fragile, c'est à dire, à la fois l'authentification d'image de la source et la vérification de l'intégrité des données. Dans la figure 4.2, nous donnons les schémas de réalisation permettant d'assurer l'intégrité des données (fonction de hachage sans clé) et l'authentification de la source de l'image.

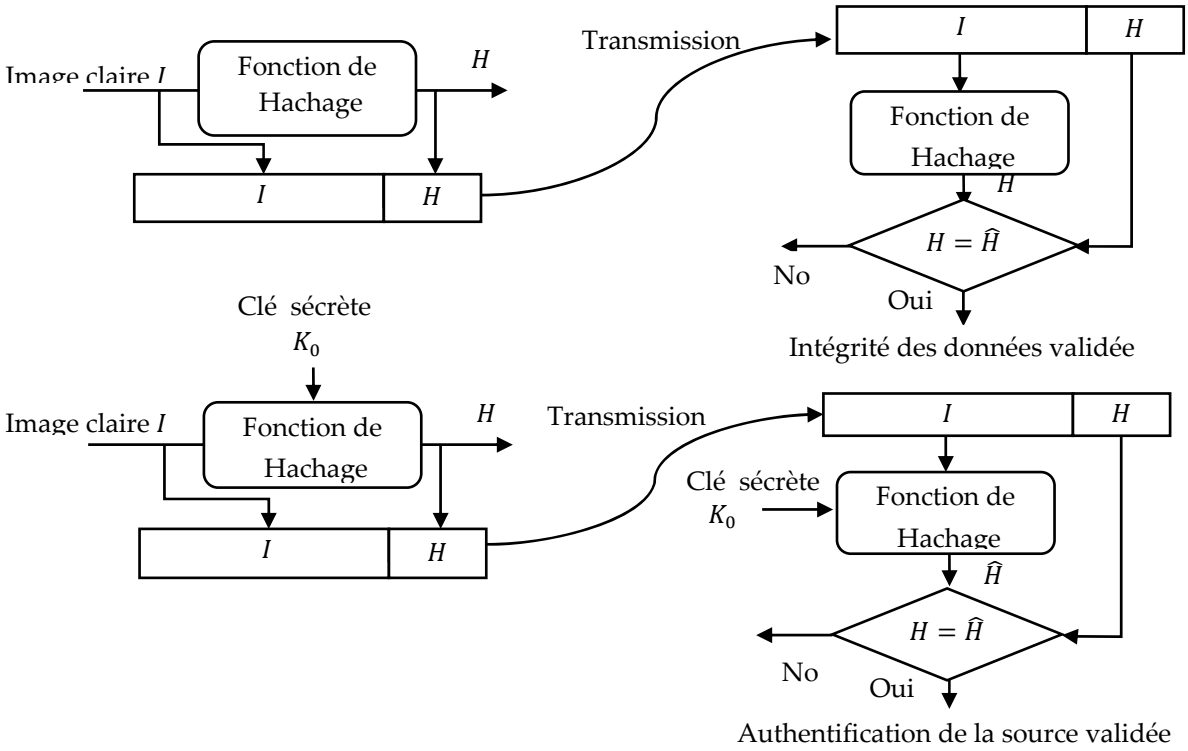



Fig. 4. 2 : Schémas de réalisation pour l'intégrité des données et l'authentification de la source

## 4.3 Fonction de hachage basée chaos avec clé sécrète proposée:

Partant de la structure générale de l'algorithme de Merkle-Damgård, nous proposons, dans cette section, une fonction de hachage basée chaos avec clé secrète $K_0$, robuste et flexible (voir figure 4.3). La fonction de hachage en question peut être facilement adaptée pour produire une empreinte digitale de taille : 128, 196, 256, 384, 512, 1024 ou 2048 bits, selon l'application désirée.

La fonction de hachage proposée est plus performante que celles de [Yang, 2010], et [Xiao et al, 2009] et [Xiao et al, 2008]. Elle résiste contre les différents types d'attaques dédiées, elle est assez rapide et tient compte dans sa conception, des faiblesses des autres structures.

Dans ce schéma, les deux couches de confusion utilisées sont les cartes chaotiques logistiques et Skew tent discrétisées sur $N = 32$ bits (voir chapitre 2). Chaque bloc de message $M_i$ de taille $Tb$ bits est divisé en $nsb = Tb/N$ sous blocs, chacun de taille $N$.

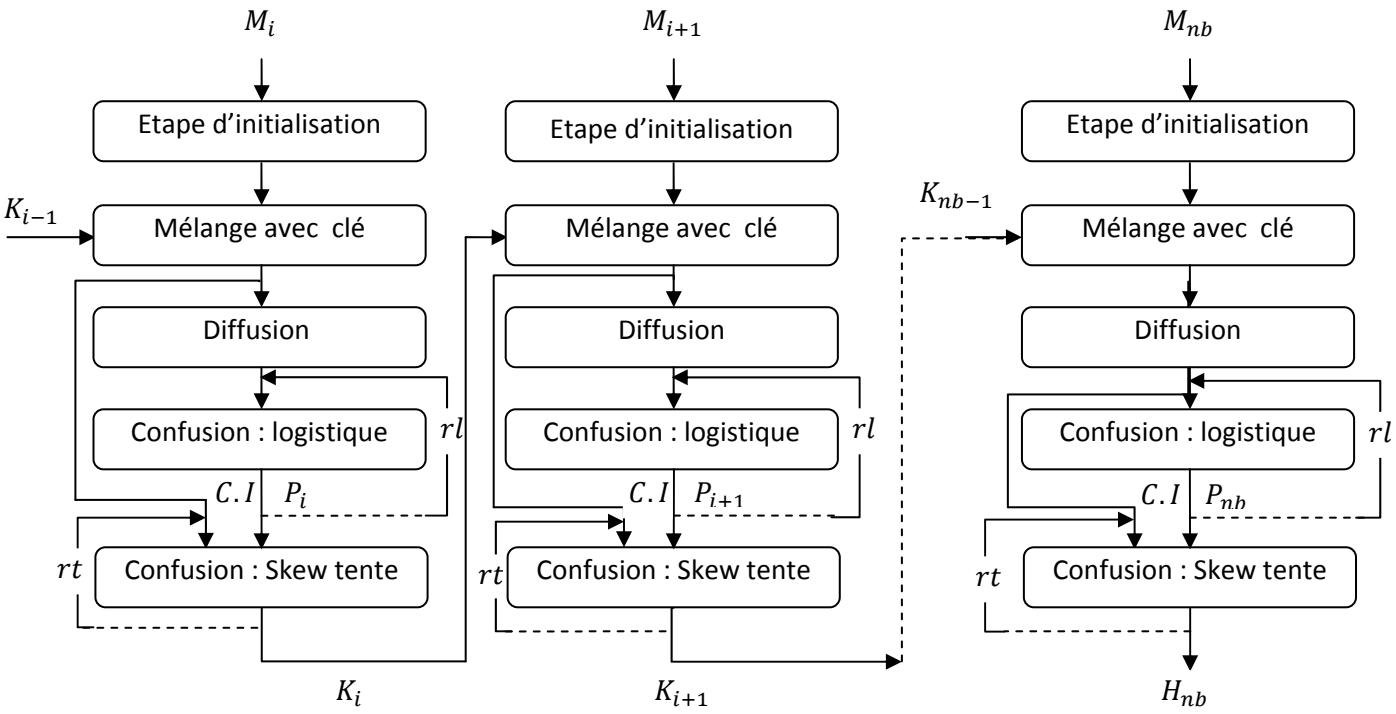



Fig. 4. 3: Architecture de la fonction de hachage proposée avec *n=m*

La combinaison de ces deux cartes, assure la propriété de la confusion (confirmée expérimentalement), tout en étant simple, puisque le paramètre de contrôle de la carte logistique discrète est déjà fixé à la valeur 4. Par ailleurs, étant donné la sensibilité forte de la carte Skew tent au paramètre du contrôle, nous produisons ce dernier à partir des résultats d'un enchainement d'opérations de mélange, de diffusion et de confusion mettant en œuvre le message et la clé secrète.

Pour chaque condition initiale de la carte Skew tent, celle-ci est itérée $rt \times nsb$ fois en utilisant l'ensemble des paramètres de contrôle $P_i = \{P_i^1,\ P_i^2, \dots, P_i^{nsb}\}$. L'expérience montre que la valeur $rt = 1$ est suffisante pour achever l'effet d'avalanche.

Deux variantes sont possibles pour la couche de diffusion : la diffusion à base d'une matrice binaire, ou la diffusion à base de la carte cat multidimensionnelle (voir chapitre 3)

Sans clé secrète, donc en omettant l'opération de mélange, la fonction de hachage est utilisée pour réaliser le service d'intégrité.

La clé secrète $K_0$ est divisée en $nsb$ blocs : $K_0 = \{K_0^1, K_0^2, \cdots, K_0^{nsb}\}$.

### 3.3.1 Description détaillée des différentes couches de la fonction de hachage chaotique proposée

Pour authentifier une image en clair $I$ de taille $TL = L \times C \times P$ en échelle de gris, la première étape consiste à transformer l'image en un tableau $IL$, de taille $(L \times C \times P)$ octets ; en langage Matlab :

$$IL = reshape(I, 1, L \times C \times P)$$

où : $L, C, et\ P$ sont, le nombre de lignes, colonnes, et plans de l'image,

puis, d'appliquer le processus de remplissage sur le dernier bloc comme suit :

$$uo = mod(To - vo - TL, To)$$

$$u = [128\ \ zeros(1, uo - 1)]$$

$$v = dec2bin\ \{reshape(bin2dec(TL * 8, vb), vo, 8)\}$$

$$IL = [IL\ \ u\ \ v]$$

Ensuite, de diviser $IL$ en $nb$ blocs, de taille $Tb$ chacun.

$$IL = \{M_1, M_2, \cdots, M_{nb}\}$$

Avec :

| $Tb$ | $To$ | $vb$ | $vo$ |
|---|---|---|---|
| 128 | 16 | 16 | 2 |
| 256 | 32 | 32 | 4 |
| 512 | 64 | 64 | 8 |
| 1024 | 128 | 128 | 16 |

Chaque bloc de message $M_i$ de taille $m = n = Tb$ bits est divisé en $nsb = Tb/N$ sous blocs, chacun de taille $Tsb = N$=32 bits.

Dans la figure 4.4, nous illustrons le processus itératif de la fonction de hachage proposée pour $Tb = 128$, et $nsb = 4$, puisque $N = 32$ bits.

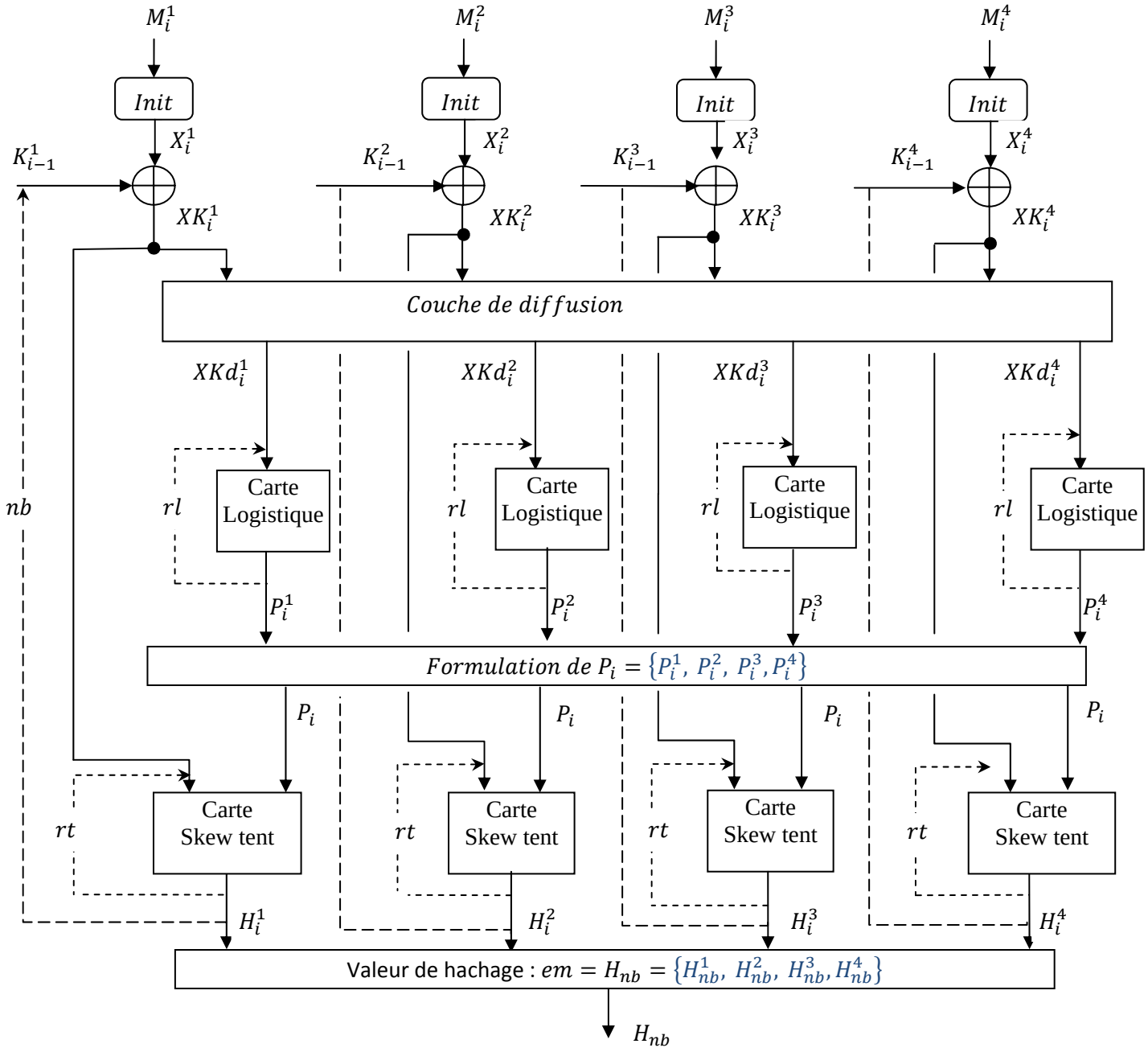



Fig. 4. 4: Processus itératif de la fonction de hachage proposée

La valeur de hachage (empreinte digitale) est produite comme suit :

**Étape *Init*** : Cette étape permet de protéger la fonction de hachage contre l'attaque à texte en clair connu ou choisi. Dans cette étape, voir équation (4.2) ci-dessous, chaque sous bloc du message $M_i^w$, $w = 1, \cdots, 4$ est mélangé avec $temp1$, pour l'indice $w$ pair ou avec $temp2$ pour l'indice $w$ impair. Le type de mélange est, soit ou exclusif pour $i$ impair, soit addition modulo (arithmétique) pour $i$ pair.

$$\begin{cases} X_i^w = M_i^w \oplus temp1 & si\ \ mod(w,2) = 0\ et\ si\ \ mod(i,2) \neq 0 \\ X_i^w = M_i^w \oplus temp2 & si\ \ mod(w,2) \neq 0\ et\ si\ \ mod(i,2) \neq 0 \\ X_i^w = mod(M_i^w + temp1, 2^N) & si\ \ mod(w,2) = 0\ et\ si\ \ mod(i,2) = 0 \\ X_i^w = mod(M_i^w + temp2, 2^N) & si\ \ mod(w,2) \neq 0\ et\ si\ \ mod(i,2) = 0 \\ w = 1, \cdots, 4\ et\ i = 1,2, \ldots, nb \end{cases} \quad (4.2)$$

Les constantes $temp1$ et $temp2$ sont calculées à partir de deux octets suivantes $92, 54$. Le choix de ces deux octets, découle de leur valeurs particulières. En effet, sur les 8 bits de chaque octet, nous avons 4 bits à 1.

Les valeurs de $temp1$ et $temp2$ sur $N = 32$ bits, sont données en Matlab par:

$$temp1 = typecast(\ uint8([54\ \ 92\ \ 54\ \ 92]), 'uint32') = 1547066422$$

$$temp2 = typecast(\ uint8([\ 92\ 54\ 92\ 54\ ]), 'uint32') = \ 912012892$$

**Étape 2 :** La sortie de cette étape, sert comme entrée de la couche de diffusion et aussi comme valeur initiale de la carte Skew tent :

$$XK_i^w = X_i^w \ \oplus K_{i-1}^w\ , w = 1,2, \cdots, 4, et\ i = 1,2, \ldots, nb \quad (4.3)$$

Notons que : pour $i = 1$, $K_0^W$, $w = 1,2, \cdots, 4$ est la clé secrète ($K_0 = \{K_0^1, K_0^2, \cdots, K_0^4\}$), et pour $i > 1$, $K_{i-1}^w = H_i^w$ est la valeur de hachage intermédiaire à l'itération $i$.

Si la fonction de hachage est sans clé secrète, alors, pour le premier bloc ($i = 1$) seulement, nous avons : $XK_1^w = X_1^w$ , w $= 1, 2, \cdots, 4$.

**Étape 3** : Production des paramètres de contrôle de la carte Skew tente.

Cette étape est composée d'un processus de diffusion, dont la sortie $XKd_i^w, w = 1, 2, \cdots, 4$ est utilisée comme condition initiale d'un processus de confusion, réalisé par la carte Logistique discrète. La sortie de cette dernière, fournit les paramètres de contrôle de la carte Skew tent.

**Couche de diffusion**

Les deux couches de diffusion que nous utilisons ici, sont celles que nous avons détaillé dans le chapitre 3, à savoir la matrice de diffusion binaire et la matrice de diffusion à base de la carte cat multidimensionnelle (deuxième approche).

**Cas de la matrice de diffusion binaire.**

Dans ce cas, rappelons le, la transformation linéaire est donnée par :

$$\begin{bmatrix} XKd_i^1 \\ XKd_i^2 \\ \vdots \\ \vdots \\ XKd_i^{nsb} \end{bmatrix} = D \odot \begin{bmatrix} XK_i^1 \\ XK_i^2 \\ \vdots \\ \vdots \\ XK_i^{nsb} \end{bmatrix} \quad (4.4)$$

$$i = 1, 2, \cdots, nb$$

Où $D$ est la matrice de diffusion binaire carrée de taille $n \times n = nsb \times nsb$, définit pour chaque $nsb$ donné et qui dépend de la taille de $Tb$ utilisée. Nous avons testé les valeurs suivantes de $nsb = 4, 6, 8, 12, 16, 32$, correspondants aux tailles $Tb$ suivantes des blocs : 128, 192, 256, 384, 512, 1024 bits respectivement. La taille de l'empreinte numérique est $Tb$.

Par exemple, dans le cas $= 4$ , $D$ est donnée par :

$$\begin{bmatrix} XKd_i^1 \\ XKd_i^2 \\ XKd_i^3 \\ XKd_i^4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \odot \begin{bmatrix} XK_i^1 \\ XK_i^2 \\ XK_i^3 \\ XK_i^4 \end{bmatrix}$$

Pour $nsb > 4$, les valeurs des différentes matrices de diffusion $D$ utilisées, sont celles données par le chapitre 3.

**Cas de la matrice de cat multidimensionnelle.**

La transformation linéaire est donnée par (voir chapitre 3)

$$\begin{bmatrix} XKd_i^1 \\ XKd_i^2 \\ \vdots \\ \vdots \\ XKd_i^{nsb} \end{bmatrix} = Mod\left( D \times \begin{bmatrix} XK_i^1 \\ XK_i^2 \\ \vdots \\ \vdots \\ XK_i^{nsb} \end{bmatrix}, 2^N \right) \tag{4.5}$$

Pour chaque valeur testée de $nsb = 4, 8, 16, 32$, nous avons cherché la matrice de diffusion optimale, la moins coûteuse en termes de temps et de ressources. Cela veut dire que (voir paragraphe 3.4.5.2 du chapitre 3): le nombre de zéros dans la matrice est relativement faible, les matrices $M_u$ et $M_v$ sont binaires, la matrice de diffusion est formée par $D = M_0 \times M_1$. De cette manière les éléments de la matrice $D$ sont majoritairement égaux à 1 et 2.

Nous donnons ci-dessous les différentes matrices de diffusion trouvées :

Pour $nsb = 4$, la transformation linéaire est :

$$\begin{bmatrix} XKd_i^1 \\ XKd_i^2 \\ XKd_i^3 \\ XKd_i^4 \end{bmatrix} = Mod\left( \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 2 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} XK_i^1 \\ XK_i^2 \\ XK_i^3 \\ XK_i^4 \end{bmatrix}, 2^N \right)$$

Pour $nsb = 8$, la transformation linéaire est :

$$\begin{bmatrix} XKd_i^1 \\ XKd_i^2 \\ XKd_i^3 \\ XKd_i^4 \\ XKd_i^5 \\ XKd_i^6 \\ XKd_i^7 \\ XKd_i^8 \end{bmatrix} = Mod\left( \begin{bmatrix} 1 & 1 & 2 & 2 & 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 & 1 & 2 & 1 \\ 1 & 1 & 3 & 2 & 1 & 1 & 2 & 1 \\ 1 & 1 & 2 & 3 & 1 & 1 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 1 & 2 & 1 \\ 1 & 1 & 2 & 2 & 1 & 2 & 2 & 1 \\ 1 & 1 & 2 & 2 & 1 & 1 & 3 & 1 \\ 1 & 1 & 2 & 2 & 1 & 1 & 2 & 2 \end{bmatrix} \times \begin{bmatrix} XK_i^1 \\ XK_i^2 \\ XK_i^3 \\ XK_i^4 \\ XK_i^5 \\ XK_i^6 \\ XK_i^7 \\ XK_i^8 \end{bmatrix}, 2^N \right)$$

Pour $nsb = 16$, la matrice de transformation linéaire $D$ trouvée est :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 |
| 1 | 3 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 |
| 2 | 4 | 5 | 2 | 4 | 2 | 2 | 2 | 4 | 4 | 4 | 4 | 4 | 2 | 4 | 4 |
| 2 | 4 | 4 | 3 | 4 | 2 | 2 | 2 | 4 | 4 | 4 | 4 | 4 | 2 | 4 | 4 |
| 2 | 4 | 4 | 2 | 5 | 2 | 2 | 2 | 4 | 4 | 4 | 4 | 4 | 2 | 4 | 4 |
| 1 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 |
| 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 |
| 2 | 4 | 4 | 2 | 4 | 2 | 2 | 3 | 4 | 4 | 4 | 4 | 4 | 2 | 4 | 4 |
| 2 | 4 | 4 | 2 | 4 | 2 | 2 | 2 | 5 | 4 | 4 | 4 | 4 | 2 | 4 | 4 |
| 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 3 | 2 | 2 | 2 | 1 | 2 | 2 |
| 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 3 | 2 | 2 | 1 | 2 | 2 |
| 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 2 | 1 | 2 | 2 |
| 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 1 | 2 | 2 |
| 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 3 | 2 |
| 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 3 |

Pour $nsb = 32$, la matrice de transformation linéaire est :

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 3 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 3 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 3 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 3 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 3 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 3 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 3 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 3 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 3 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 3 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 3 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 3 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 3 | 1 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 2 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 3 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 3 | 1 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 2 | 1 |
| 1 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 2 | 2 | 1 | 2 |

Effectivement, nous remarquons que la majorité des éléments des trois matrices de diffusion pour $nsb = 8, 16$ et $32$, prennent les valeurs 1, 2. Ci-dessous, nous donnons la fréquence de chacune des valeurs.

Tableau 4. 1: Fréquence de valeur des trois matrices de diffusion proposée

| Fréquence \valeur | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Cat proposée(8× 8) | 0 | 36 | 25 | 3 | 0 | 0 |
| Cat proposée(16× 16) | 0 | 63 | 134 | 9 | 47 | 3 |
| Cat proposée (32 × 32) | 44 | 486 | 479 | 15 | 0 | 0 |

**Couche de confusion intermédiaire**

La couche de confusion intermédiaire est réalisée par la carte Logistique, dont l'équation peut s'écrire aussi comme suit :

$P_i^w = L^{rl}(XKd_i^w),\ w = 1,2,\dots,nsb,$ et $1 \le i \le nb.$

où $rl$ est le nombre d'itérations utilisé de la carte Logistique, déterminé à partir de l'étude statistique de la sensibilité au texte en clair et à la clé sécrète.

Suite à cette étape, nous formons les paramètres de contrôle $P_i = \{P_i^1,\ P_i^2, \dots, P_i^{nsb}\}$ de la carte Skew tent.

**Étape 4** : Couche de confusion finale

La couche de confusion finale est réalisée par la carte Skew tent utilisant l'ensemble des paramètres de contrôle $P_i = \{P_i^1,\ P_i^2, \dots, P_i^{nsb}\}$, et dont l'équation peut s'écrire comme suit :

$$H_i^w = ST^{rt}(XK_i^w,\ P_i)\,, \qquad w = 1,2,\dots,nsb, \text{et } 1 \le i \le nb. \tag{4.6}$$

où $rt$ est le nombre d'itérations utilisé de la carte skew tent, déterminé à partir de l'étude statistique de la sensibilité au texte en clair et à la clé sécrète.

**Étape 5** : Test

Tant que : $i \neq nb$ alors : $K_{i-1}^w = H_i^w$ et on itère tous les processus.

Pour $i = nb$, le dernier bloc du message est traité et alors, nous obtenons l'empreinte digitale $em = H_{nb} = \{H_{nb}^1,\ H_{nb}^2, \dots, H_{nb}^{nsb}\}$

de taille $Tb$, qui servira pour authentifier le message, si le processus de hachage utilise une clé sécrète, ou pour vérifier l'intégrité du message, si le processus de hachage n'utilise pas une clé sécrète.
La fonction de hachage proposée avec une empreinte digitale de taille $Tb = 512$ bits, est largement suffisante pour toutes les applications de hachage utilisées actuellement. Pour des applications ultra sécurisées, on préconise d'utiliser notre fonction de hachage avec une empreinte digitale de taille $Tb = 1024$ bits, qui a l'avantage aussi d'être plus rapide que celle de 512 bits.

#### *2.3.1.1 Mesure des nombres d'itérations nécessaires $rl$ et $rt$*

Une fonction de hachage est dite robuste, si le changement d'un seul bit d'entrée de la fonction de hachage (bit de la clé secrète ou du texte en clair), et quelques soit sa position, provoque le changement de la moitié des bits de l'empreinte digitale comparée à celle sans modification. Ceci démontre que la valeur de chaque bit final de l'empreinte digitale est lié à tous les bits d'entrée (message et clés). Nous présentons dans la figure 4.5, la procédure de test de la sensibilité à la clé secrète (la procédure de la sensibilité au texte en clair est similaire), et nous donnons dans les différents tableaux ci-dessous, les résultats obtenus pour $Tb = 128$. Pour plus de détail sur la procédure de mesure, voir le paragraphe 3.7.2 du chapitre 3.

$rll = 8, rtt = 8, nk = 1000$
Pour $itl = 1 : rll$
  Pour $itt = 1 : rtt$
    Pour $itk = 1 : nk$
//création de la clé sécrète $K_j$
// Empreinte digitale de la fonction de hachage proposée
$em = Hachage(itl, itt, K_j, message)$
Changement d'un bit de façon aléatoire (bit en position $i$) de la clé sécrète pour obtenir $Ki_j$
$i = ceil\ (rand \times Tb)$
$emi = Hachage(itl, itt, Ki_j, message)$
// calcule du pourcentage de distance de Hamming
$PDH(itl, itl, itk) = PDH(em, emi)$
$Fin\ pour\ itk$
$Fin\ Pour\ itt$
$Fin\ pour\ itl$
$Moyenne\_PDH = mean(PDH')$
$Ecart\_PDH = std(PDH')$

Fig. 4. 5: Procédure de mesure de la sensibilité à la clé secrète

Dans les tableaux 4.1 et 4.2, nous présentons les résultats obtenus du test de la sensibilité à la clé secrète, à savoir, respectivement: la moyenne, sur les 1000 clés testés, du pourcentage $M\_PDH$ et de l'écart type $E\_PDH$ correspondant.

Tableau 4. 2 : Moyenne du pourcentage *M_PDH* en fonction de *rt* et *rl*

| $rt \backslash rl$ | 1 | 2 | 3 | **4** | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 49,96881 | 49,98156 | 49,99224 | **50,01098** | 49,99774 | 50,05804 | 49,98309 | 49,98266 |
| 2 | 49,99932 | 50,04559 | 49,95513 | 49,98168 | 50,02484 | 50,02679 | 49,99444 | 50,01843 |
| **3** | 49,94824 | 49,92547 | 49,97473 | 50,03479 | 49,97442 | 49,99768 | 49,95288 | 50,00854 |
| 4 | 49,97259 | 49,96514 | 50,03894 | 49,99548 | 49,94427 | 49,94195 | 50,00891 | 49,96875 |
| 5 | 50,01708 | 50,02026 | 50,03753 | 50,00018 | 50,02178 | 49,95971 | 50,03613 | 49,96368 |
| 6 | 49,96160 | 49,97503 | 50,05517 | 50,04412 | 49,91540 | 49,99658 | 49,99365 | 50,05572 |
| 7 | 50,01959 | 49,98266 | 49,96252 | 50,11914 | 50,02691 | 49,97479 | 50,01196 | 50,01519 |
| 8 | 49,96850 | 49,94610 | 50,01324 | 49,94580 | 50,01812 | 49,98413 | 49,97332 | 50,01245 |

Tableau 4. 3: Moyenne de l'écart type *E_PDH* en fonction de *rt* et *rl*

| $rt \backslash rl$ | 1 | 2 | 3 | **4** | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0,326515 | 0,338126 | 0,316553 | **0,279597** | 0,329582 | 0,303699 | 0,316534 | 0,302522 |
| 2 | 0,287983 | 0,305482 | 0,305655 | 0,316465 | 0,330110 | 0,306095 | 0,281263 | 0,286665 |
| **3** | 0,282552 | 0,304228 | 0,316601 | 0,294944 | 0,311534 | 0,316231 | 0,299729 | 0,302158 |
| 4 | 0,332646 | 0,321320 | 0,317834 | 0,313080 | 0,339186 | 0,331382 | 0,290306 | 0,320784 |
| 5 | 0,319918 | 0,341952 | 0,323141 | 0,317493 | 0,328182 | 0,336230 | 0,318566 | 0,317877 |
| 6 | 0,325738 | 0,323705 | 0,323435 | 0,350275 | 0,269166 | 0,322736 | 0,327636 | 0,306833 |
| 7 | 0,309142 | 0,304798 | 0,315500 | 0,284078 | 0,310286 | 0,339655 | 0,331746 | 0,318446 |
| 8 | 0,305093 | 0,323084 | 0,299070 | 0,302210 | 0,339660 | 0,321553 | 0,329644 | 0,302037 |

Nous remarquons que, même le couple $\{rl = 1, rt = 1\}$ assure l'effet d'avalanche souhaitée. Le meilleur résultat est obtenu pour le couple $\{rl = 1, rt = 4\}$.

Dans les tableaux 4.3 et 4.4, nous présentons les résultats obtenus du test de la sensibilité au texte en clair, à savoir, respectivement: la moyenne, sur l'ensemble du message (l'image Lena 512 x512 x 3), du pourcentage *M_PDH* et de l'écart type *E_PDH* correspondant.

Tableau 4. 4: Moyenne du pourcentage *M_PDH* en fonction de *rt* et *rl*

| $rt \backslash rl$ | 1 | 2 | 3 | **4** | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 48,64461 | 48,75725 | 48,99916 | **49,15181** | 49,31549 | 49,41460 | 49,46168 | 49,60096 |
| 2 | 49,18900 | 49,21430 | 49,33826 | 49,41330 | 49,51634 | 49,60108 | 49,63886 | 49,63287 |
| 3 | 49,38986 | 49,45181 | 49,54179 | 49,61075 | 49,62235 | 49,66858 | 49,73241 | 49,75699 |
| 4 | 49,54011 | 49,56252 | 49,67449 | 49,68187 | 49,76221 | 49,74912 | 49,75786 | 49,77936 |
| 5 | 49,69508 | 49,70564 | 49,70720 | 49,75361 | 49,77912 | 49,80781 | 49,79443 | 49,84865 |
| 6 | 49,77467 | 49,75223 | 49,76299 | 49,80078 | 49,79879 | 49,82648 | 49,83744 | 49,87828 |
| 7 | 49,82675 | 49,84816 | 49,85516 | 49,82895 | 49,84846 | 49,89502 | 49,86472 | 49,90841 |
| 8 | 49,83861 | 49,84308 | 49,88597 | 49,86565 | 49,86400 | 49,92007 | 49,91404 | 49,93168 |

Tableau 4. 5: Moyenne de l'écart type *E_PDH* en fonction de *rt* et *rl*

| $rt \backslash rl$ | 1 | 2 | 3 | **4** | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0,628446 | 0,553214 | 0,527195 | **0,428644** | 0,406071 | 0,346209 | 0,349215 | 0,351770 |
| 2 | 0,492434 | 0,474022 | 0,444580 | 0,419702 | 0,441403 | 0,367062 | 0,383844 | 0,354614 |
| 3 | 0,470143 | 0,428940 | 0,441429 | 0,409270 | 0,414600 | 0,409482 | 0,358877 | 0,355117 |
| 4 | 0,491663 | 0,519895 | 0,419081 | 0,394367 | 0,432201 | 0,389659 | 0,395411 | 0,426087 |
| 5 | 0,497932 | 0,512064 | 0,464905 | 0,447319 | 0,443571 | 0,412222 | 0,460080 | 0,410018 |
| 6 | 0,504276 | 0,477535 | 0,512610 | 0,487308 | 0,447641 | 0,494716 | 0,496643 | 0,478757 |
| 7 | 0,559399 | 0,509248 | 0,548091 | 0,586460 | 0,517973 | 0,499939 | 0,506755 | 0,462696 |
| 8 | 0,585501 | 0,555361 | 0,541158 | 0,533273 | 0,515644 | 0,531936 | 0,437398 | 0,453642 |

Nous remarquons que, l'effet d'avalanche est mieux assuré pour les couples tels que la valeur $M\_PDH \geq 49.5$.

La sensibilité à la clé secrète est plus prononcée que celle du texte en clair, du fait de la structure même de l'algorithme Merkle-Damgård.

## 4.4 Analyse des performances et de la sécurité de la fonction de hachage proposée

Dans cette section, nous effectuons quelques testes nécessaires pour quantifier et analyser la sécurité de la fonction de hachage proposée.

### 3.4.1 Sensibilité de l'empreinte digitale au message

Pour ce teste, nous partons du message proposé par [Xiao, 2005 ] suivant, dont le nombre des caractéres ASCII est 568.

*"As a ubiquitous phenomenon in nature, chaos is a kind of deterministic random-like process generated by nonlinear dynamic systems. The properties of chaotic cryptography includes: sensitivity to tiny changes in initial conditions and parameters, random-like behavior, unstable periodic orbits with long periods and desired diffusion and confusion properties, etc. Furthermore, benefiting from the deterministic property, the chaotic system is easy to be simulated on the computer. Unique merits of chaos bring much promise of application in the information security field."*

La valeur de la clé sécrète utilisée est :

| Clé sécrète | $K_0^1$ | $K_0^2$ | $K_0^3$ | $K_0^4$ |
|---|---|---|---|---|
| Hexadécimale : | A4FC8A60 | 52EAC9EC | D9328FF6 | 667F400D |
| Décimale : | 2768013920 | 1391118828 | 3643969526 | 1719615501 |

Pour évaluer la sensibilité de l'empreinte digitale au message, nous réalisons les mêmes expériences, au nombre de 6, que [Xiao, 2005 ] :

C1 : le message original est inchangé;
C2 : changement du premier caractère (A) du message original par le caractère (B);
C3 : changement du mot (unstable) dans le message original par le mot (anstable);
C4 : changement du point à la fin du message original par une virgule;
C5 : rajout d'un espace blanc à la fin du message original;
C6 : découpe de la partie suivante du message (nistic random-like process generated by nonlinear dynamic system), et la remettre au début du message.

Les empreintes digitales (valeurs de hachage) correspondantes aux différentes expériences, en hexadécimal, obtenues par la fonction de hachage proposée, dans le cas de $Tb = 128, rl = 4, rt = 1$, sont :

$$C1 : 02fbf9d8d111b87e434595a2e80237b8$$

$$C2 : a8137e8d3a9f6feda4af9562d08cd329$$

$$C3 : 6c7cb15396f934f5522de32470fe7a41$$

$$C4 : 7d9a394bf46e8f395a0ea0027a29f61f$$

$$C5 : d9407c0165ad69942bba1b50c5c7b856$$

$$C6 : 203dea537eb37789df4ee9b400e94eed$$

En représentation signal binaire, les différentes empreintes digitales obtenues sont représentées par la figure suivante.

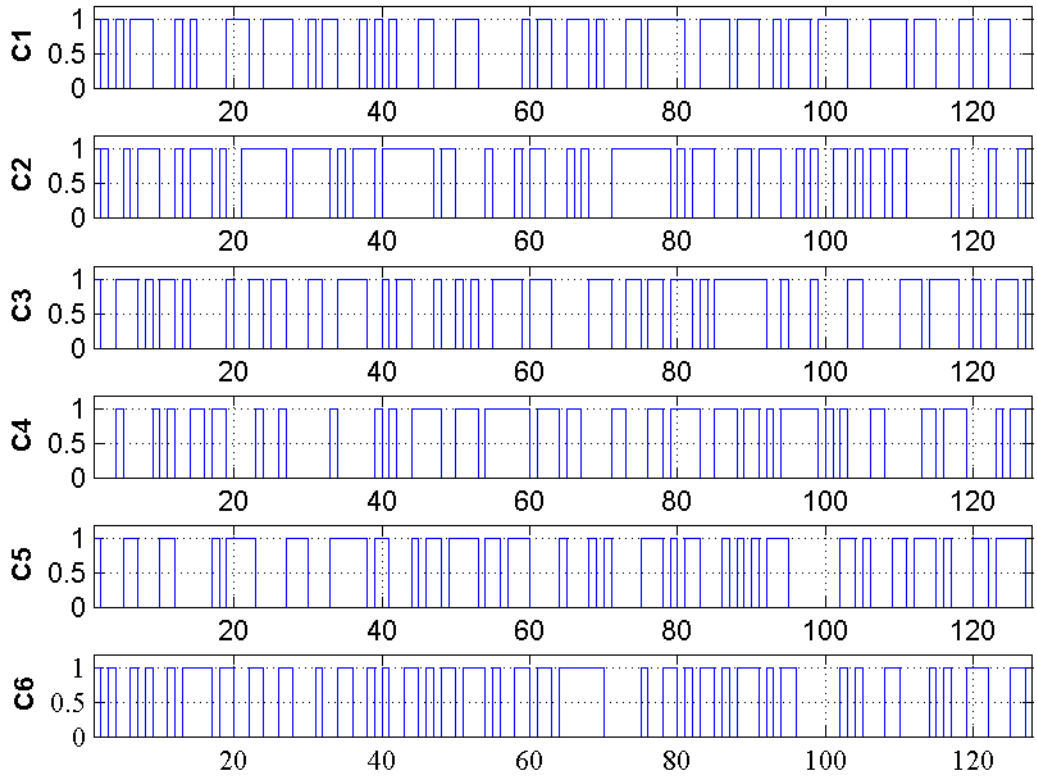



Fig. 4. 6 : Sensibilité de l'empreinte digitale au message

Visuellement, nous pouvons déjà observer que les différentes empreintes digitales sont distinctes. Pour quantifier la différence entre ces empreintes digitales, nous avons calculé le pourcentage (en bits) de la distance de hamming entre ces fonctions prises deux à deux.Les résultats obtenus, et montrés dans le tableau 4.6 suivant, confirment la sensibilité de l'empreinte digitale au message et prouvent que l'effet d'avalanche est achevé.

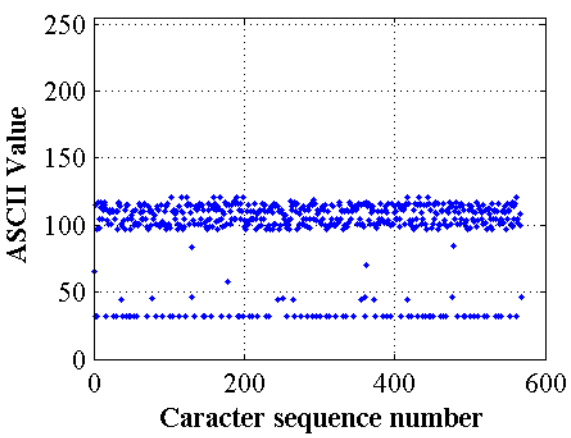
Tableau 4.6: Pourcentage en bits de la distance de hamming en bits pour les différentes empreintes digitales prises deux à deux.

| % | C1 | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|
| C1 | - | 49,2187 | 57,0312 | 46,0937 | 57,8125 | 55,4687 |
| C2 | 49,2187 | - | 59,375 | 56,25 | 63,2812 | 53,125 |
| C3 | 57,0312 | 59,375 | - | 53,125 | 42,9687 | 48,4375 |
| C4 | 46,0937 | 56,25 | 53,125 | - | 47,6562 | 50 |
| C5 | 57,8125 | 63,2812 | 42,9687 | 47,6562 | - | 42,9687 |
| C6 | 55,4687 | 53,125 | 48,4375 | 50 | 42,9687 | - |

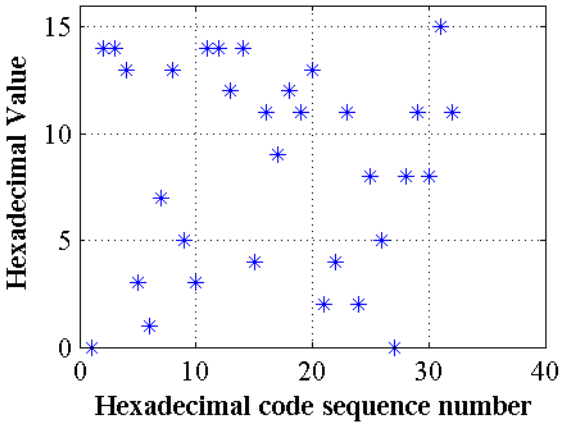
### 3.4.2 Distribution de l'empreinte digitale

La distribution de l'empreinte digitale est directement liée à la sécurité de la fonction de hachage. Aussi, afin de montrer visuellement la différence entre le message précédent (formé

de 568 caractère ASCII) de et l'empreinte digitale (formé de 32 caractère hexadécimal), nous avons tracé dans la figure 4. 7 a) et b), les distributions du message et de l'empreinte digitale, chacune en fonction de sa séquence de caractère (respectivement ASCII, et Hexadécimal).




a)




b)

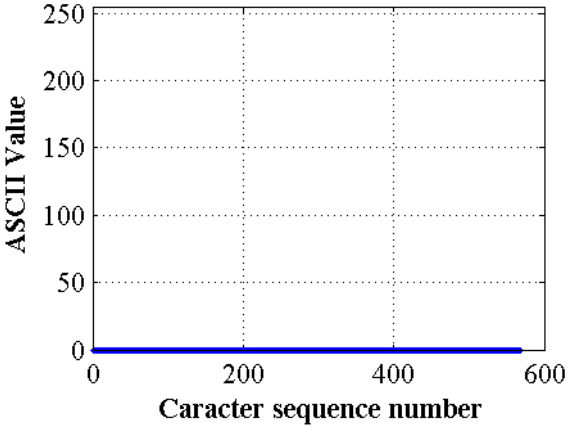
Fig. 4. 7 :a) Distribution du message en fonction de sa séquence de caractères ASCII b) Distribution de l'empreinte digitale en fonction de sa séquence de caractères hexadécimal.

Nous observons clairement dans la figure 4.7 a), que la distribution du message en fonction de sa séquence de caractères ASCII est globalement localisée dans une zone assez faible, tandis que la figure 4.7 b) montre que la distribution de l'empreinte digitale en fonction de sa séquence de caractères hexadécimal est répartie de façon aléatoire dans l'ensemble de la région concernée.

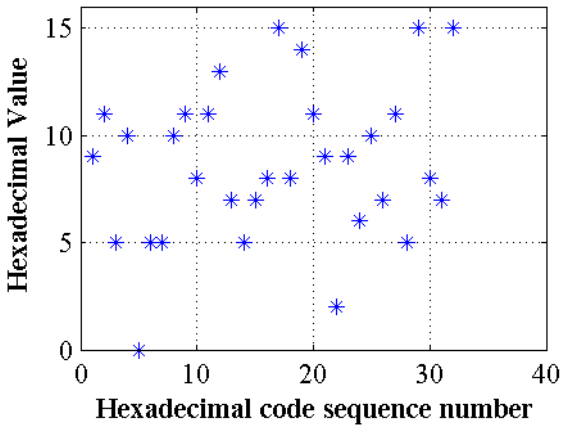
Par ailleurs, dans le cas d'un message nul, nous avons calculé son empreinte digitale

$$em = 02f58a01b08524cca1756c8a03f3cbe3$$

et nous avons répété l'expérience précédente. Les résultats obtenus sont montrés dans la figure 4.8 a) et b). Nous observons clairement, d'après la figure 4.8 b), que malgré l'absence de message à l'entrée de la fonction de hachage, l'empreinte digitale est non nulle et sa distribution est aléatoire dans toute la région concernée. Ceci, montre encore une fois, la robustesse de la fonction de hachage proposée contre l'attaque à texte en clair choisi et démontre que les processus de diffusion-confusion sont d'une efficacité telle qu'aucune information statistique ou autre ne peut pas être extraite à partir de l'empreinte digitale produite.




a)




b)

Fig. 4. 8: a) Distribution du message nul b) Distribution de son empreinte digitale

### 3.4.3 Résistance contre la collision type seconde préimage :

Seconde préimage : Etant donné un message $M$ choisit aléatoirement, trouvé un message $Mi$, tel que :

$$emi = H(Mi) = H(M) = em$$

Pour montrer la résistance de la fonction de hachage proposée contre la collision de type seconde préimage, nous avons réalisé l'expérience suivante :

D'abord, nous générons de façon aléatoire et uniforme 100 clés. Puis, nous générons un message $M$ de taille $To = 1000$ octets (caractères) selon une loi gaussienne de moyenne égale $\cong 100$ à et d'écart type égale à 26 : $M = round\left\{50 + \left(50 + 26 \times randn(1, To)\right)\right\}$

Ensuite, pour chaque clé $K_j$, $j = 1 : nk = 100$, nous réalisons les étapes suivantes :

Calcul de l'empreinte digitale $em$ du message $M$
Pour $i = 1{:}\,1000$

- création du message $Mi(i)$, à partir du message $M$, en changeant le bit de poids faible de l'octet $i$ du message d'origine
- calcul de l'empreinte digitale $emi$ du message $Mi(i)$
- Calcul des collisions des octets identiques se trouvant aux mêmes indices
- Calcul de la moyenne de la différence des intensités des niveaux de gris UACI entre les deux empreintes d'origine et modifiée

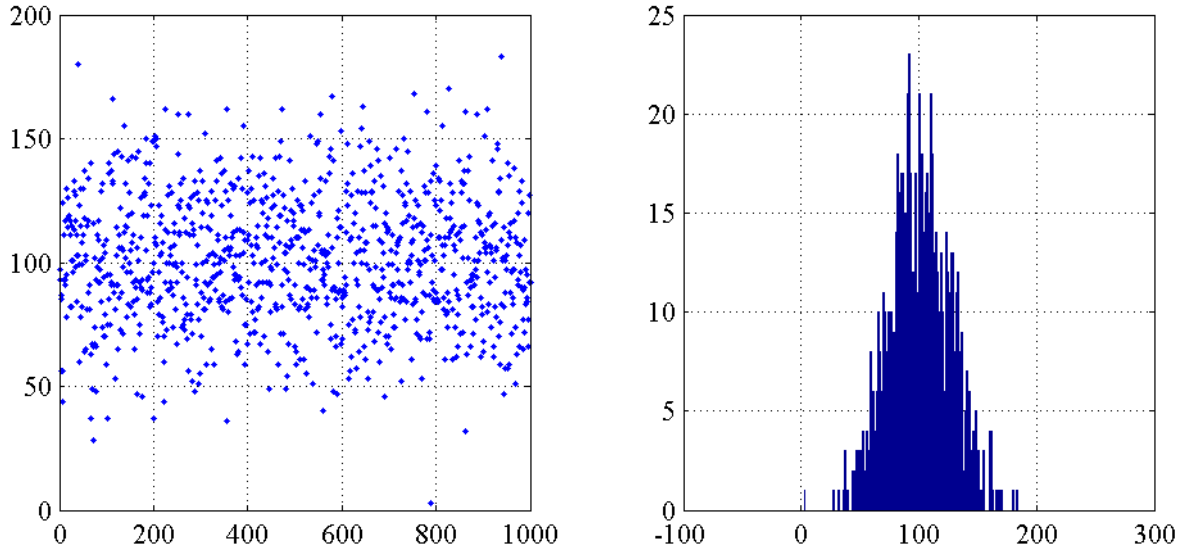
Fin de la boucle $i$
Fin de la boucle $j$

Nous donnons ci-dessous, le pseudo-code Matlab, réalisant les calculs précédents, et aussi des calculs sur les bits : distance de Hamming $DH$ et sa proportion $PDH$.

**Entrée** : message $M$, $nk$,$Tb$
**Sortie** : $hits, d, To$
**Initialisation** : $rl = 4, rt = 1$
$Pour\ j = 1 : nk$
$//Génération\ de\ clé\ sécrète\ \ K_j$
$K_j = floor(rand(1, Tb/N) \times 2^N)$
// Empreinte digitale de la fonction de hachage proposée
$em = Hachage(rl, rt\ , K_j, M)$
$em_b = reshape(dec2bin(\ em, 8), 1, To \times 8)$
// Création des différentes messages $Mi$ à partir de $M$, en changeant le bit de poids faible de chaque caractère du message $M$
Pour $i = 1 : length(M)$
$Mi = M$
$Mi(i) = Mi(i) \oplus 1$
$emi\ = Hachage(rl, rt, K_j, Mi)$
$emi_b = reshape(dec2bin(\ emi, 8), 1, To \times 8)$
$DH(j, i) = sum\{\ emi_b \oplus em_b\}$
$PDH(j, i) = DH(j, i)/(To \times 8) \times 100\%$
// calcul du nombre de caractères (octets) identiques, la somme de la différence des intensités des niveaux de gris, et UACI.
$collision\big((j-1) \times nk + i\big) = length\ (find(em == emi))$
$d(j, i) = sum(abs(em - emi)))$
$UACI(j, i) = \ \{d(j, i)/(255 \times To)\} \times 100\%$
$Fin\ pour\ i$
$Fin\ pour\ j$

Fig. 4. 9: Pseudo code Matlab pour le teste de la résistance à la collision

Dans la figure 4.10, nous montrons en a) la distribution du message généré $M$ en fonction de sa séquence d'octets (caractères) et en b) son histogramme.




a) b)

Fig. 4. 10 : a) Distribution du message $M$ généré b) Histogramme du message $M$

Dans le tableau 4.6, nous donnons la fréquence des collusions,

$$fo = hist\ (collision, length\ (unique\ (collision)\ )\ )$$

pour différentes tailles $Tb$, des empreintes digitales $em$ et $emi$, créés à partir de $M$ et $Mi$ pour l'ensemble des 100 clés aléatoires.

Tableau 4. 7: Fréquence des collisions entre les empreintes *em* et *emi*

| $\frac{fo}{Tb}$ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 128 | 93947 | 5860 | 189 | 4 | 0 | 0 |
| 256 | 88507 | 10828 | 641 | 22 | 2 | 0 |
| 512 | 78138 | 19321 | 2352 | 180 | 8 | 1 |

Nous remarquons par exemple dans le cas $Tb = 128$, que sur les 100.000 comparaisons entres empreintes digitale $em$ et $emi$ , nous avons :

93947 fois sans aucun octet en commun (absence de collusions), 5860 fois, un seul octet en collision, 189 fois, deux octets en collision, et 4 fois, trois octets en collusion.

Par ailleurs, plus $Tb$ est grand, plus la fréquence des collusions est grande.

Cependant, globalement, quelque soit la taille $Tb$, et dans le pire des cas, la fréquence des collusions de plus d'un octet est très faible # 2.3%.

Dans la figure 4.11 a), b) et c), nous présentons respectivement, pour trois tailles de $Tb = 128, 256, et\ 512$, l'évolution de la moyenne sur les 100 clés, du paramètre $UACI$ en fonction de l'emplacement de l'indice $i$ de l'octet modifié du message $M$ : $M_{UACI} = Mean(UACI')$

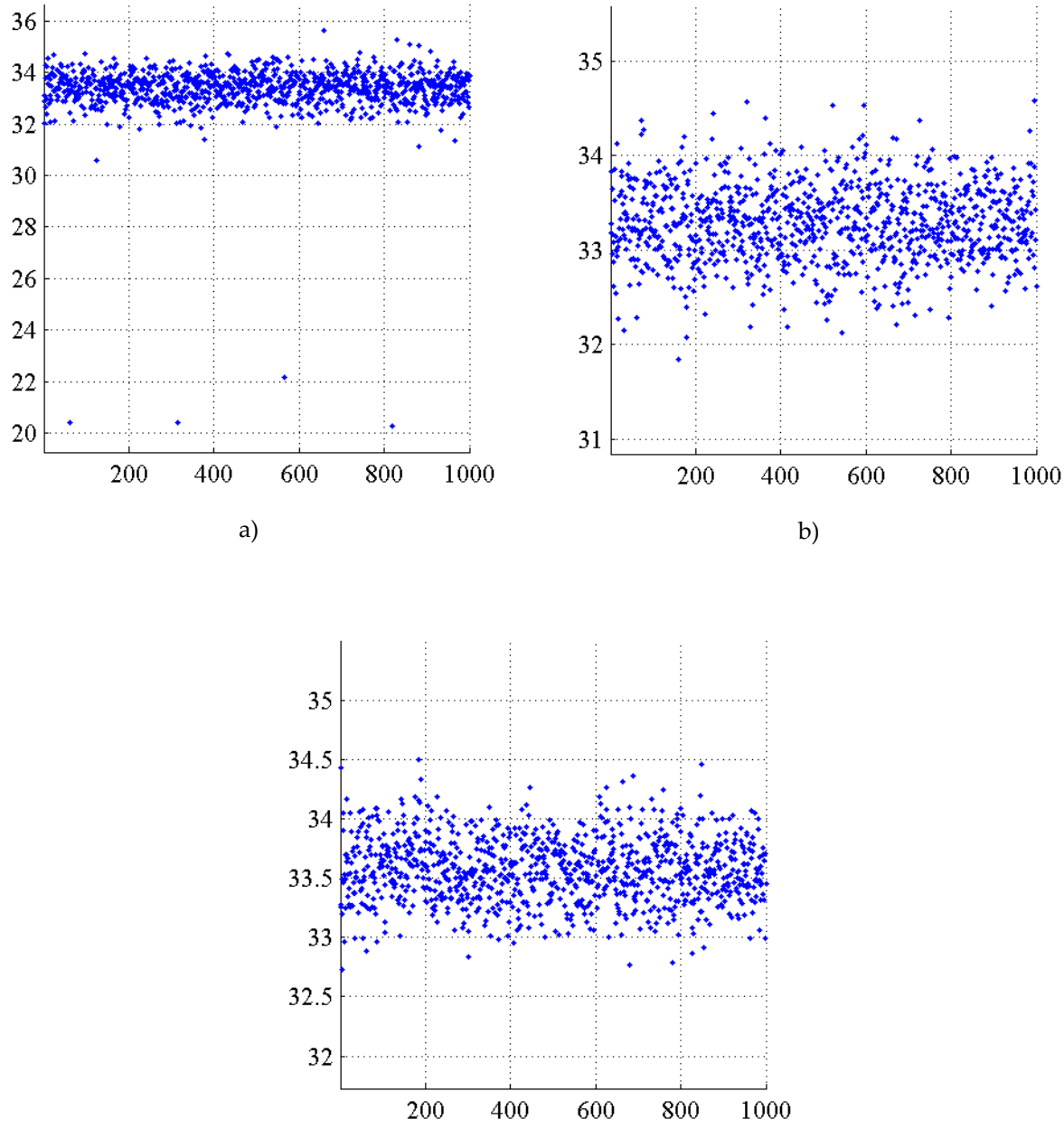



Fig. 4. 11: Evolution du $M$ en fonction de l'emplacement de l'indice $i$ de l'octet modifié du message $M$, et ceci pour a) $Tb = 128$, b) $Tb = 256$, et $Tb = 512$.

Par ailleurs, pour les trois tailles de $Tb$ considérées, nous présentons : dans le tableau 4.7, les minimum, moyenne, maximum et écart-type de $M_{UACI}$ ; dans le tableau 4.8, les minimum, moyenne et maximum de la distance de Hamming $M_{DH}$ ; et dans le tableau 4.9, les minimum, moyenne, maximum et écart-type du pourcentage de la distance de Hamming : $M_{PDH}$.

Tableau 4. 8 : Min ($M_{UACI}$), Moy ($M_{UACI}$), Max ($M_{UACI}$) et Ecart-Type ($M_{UACI}$)

| $Tb$ | Min ($M_{UACI}$)% | Moy ($M_{UACI}$)% | Max ($M_{UACI}$)% | Ecart-Type ($M_{UACI}$)% |
|---|---|---|---|---|
| 128 | 20.2505 | 33,3613873 | 35.6578 | 0,4631 |
| 256 | 31.8407 | 33.2777 | 34.5793 | 0.4035 |
| 512 | 32.7267 | 33.5717 | 34.4998 | 0.2724 |

Tableau 4. 9: Valeurs des : Min ($M_{DH}$), Moy ($M_{DH}$) et Max ($M_{DH}$)

| $Tb$ | Min ($M_{DH}$) | Moy ($M_{DH}$) | Max ($M_{DH}$) |
|---|---|---|---|
| 128 | 34 | 63.982 | 87 |
| 256 | 93 | 127.97 | 165 |
| 512 | 204 | 256.035 | 305 |

Tableau 4. 10: Valeurs des : Min ($M_{PDH}$), Moy ($M_{PDH}$), Max ($M_{PDH}$) et Ecart-Type ($M_{PDH}$)

| $Tb$ | Min ($M_{PDH}$)% | Moy ($M_{PDH}$)% | Max ($M_{PDH}$)% | Ecart-Type ($M_{PDH}$) |
|---|---|---|---|---|
| *128* | *26.5625* | *49.9859* | *67.9688* | *0.4377* |
| *256* | *36.3281* | *49.9883* | *64.4531* | *0.5727* |
| *512* | *39.8438* | *50.0068* | *59.5703* | *0.8615* |

Tous ces résultats se concordent et montrent clairement que, quelque soit la valeur de $Tb$, les empreintes digitales $em$ et $emi$ possèdent pratiquement une distribution uniforme. En effet, deux séquences parfaitement aléatoires ont une valeur $UACI_{opt} \cong 33.46\%$, et une valeur $PDH \cong 50\%$, qui correspond à une modification de la moitié des bits entre $em$ et $emi$.

### 3.4.4 Flexibilité

La fonction de hachage proposée est flexible. En effet, elle est facilement adaptable pour n'importe quelle taille $Tb = N \times nsb$, de l'empreinte digitale, avec $N = 32$ bits. En effet, dans la figure 4.4, nous avons illustré la structure de la fonction de hachage proposée pour une taille d'empreinte $Tb = 128$, soit un nombre de sous blocs $nsb = 4$. Maintenant, si nous souhaitons avoir une taille d'empreinte $Tb = 512$, soit un nombre de sous blocs $nsb = 16$, il suffit de rajouter à la structure précédente de la fonction de hachage, 12 cartes Logistique, et 12 cartes Skew tent, puis de remplacer la matrice de diffusion $D$ pour une autre de taille $nsb = 16$. Par ailleurs, le traitement dans les différents sous blocs $nsb$ est fait en parallèle.

## 4.5 Conclusion

Dans ce chapitre, nous avons tout d'abord, introduit les généralités, propriétés et le modèle de Merkle-Damgard couramment utilisé pour la réalisation des fonctions de hachage standard et chaotique. Ensuite, après avoir révélé les faiblesses des fonctions de hachages chaotiques de [Yang, 2010], et [Xiao et al, 2009], et partant du modèle de Merkle-Damgard, nous avons proposé une fonction de hachage chaotique plus performante que celles déjà étudiées dans la littérature. En effet, les différents résultats obtenus sur la sensibilité de l'empreinte digitale au message, la sensibilité au texte en clair, et à la clé secrète puis, la résistance contre la collision, montrent l'efficacité cryptographique de la fonction de hachage proposée. Par ailleurs, l'architecture de la fonction de hachage proposée, admet un certain parallélisme en implémentions matérielle et logiciel, lui permettant d'être rapide, et possède aussi une flexibilité par rapport à la taille $Tb$ des blocs sous traitement.

[Bakhtiari, 1995] S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk, "Keyed hash functions", Proceedings of the Cryptography: Policy and Algorithms Conference, Lecture Notes in Computer Science, Vol. 1029, Springer-Verlag, Berlin, pp. 201–214, 1995.

[Damgard, 1989] I. B. Damgard, "A design principle for hash functions", in: Proceedings Crypto' 89, pp. 416–427, 1989.

[Deng et al, 2010] S. Deng, Y. Li, D. Xiao, "Analysis and improvement of a chaos-based Hash function construction", Communications in Nonlinear Science and Numerical Simulation, Vol. 15, no. 5, pp. 1338-1347, May 2010.

[FIPS 180-3, 2008] FIPS 180-1, "Secure Hash Standard", Federal Information Processing Standard (FIPS) ", Publication 180-3, National Institute of Standards and Technology, US Department of Commerce, Washington DC, 2008.

[Freier et Karlton, 2011] A. Freier, P. Karlton, P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", August 2011.

[Guo et al, 2009] W.Guo, X.Wang, D.He, Y.Cao, "Cryptanalysis on a parallel keyed hash function based on chaotic maps", Physics Letters A, Vol. 373, no. 36, pp. 3201-3206, August 2009.

[Kent et Atkinson, 1998] S. Kent, R. Atkinson, "IP Encapsulating Security Payload (ESP) ", IETF. RFC 2406, November 1998.

[Knuth, 1998] D. Knuth, "The Art of Computer Programming, Sorting and Searching ", Vol. 3 Addison-Wesley, 1998.

[Menezes et al, 1996] A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.

[Merkle, 1989] R.C. Merkle, "One way hash functions and DES", in Proceedings Crypto' 89, pp. 428–446, August, 1989.

[Wang et al, 2012] S. Wang, G. Hu, "Coupled map lattice based hash function with collision resistance in single-iteration computation", Information Sciences, Vol. 195, pp. 266-276, July 2012.

[Xia et al, 2008] D. Xiao, X.F. Liao, S.J. Deng, "Parallel keyed hash function construction based on chaotic maps", Phys Lett A, Vol. 372, pp. 4682–4688, 2008.

[Xiao et al, 2009-1] Xiao D, Liao XF, Wang Y. "Parallel keyed Hash function construction based on chaotic neural network", Neurocomputing 2009, pp. 2288–2296, 2009.

[Xiao et al, 2009-2] D. Xiao, X. Liao, Y. Wang, "Improving the security of a parallel keyed hash function based on chaotic maps", Physics Letters A, Vol. 373, no. 47, pp. 4346-4353, November 2009.

[Xiao et al, 2010] D. Xiao, W. Peng, X.F. Liao, T. Xiang, "Collision analysis of one kind of chaos-based hash function", Phys Lett A, Vol. 374, pp. 1228–1231, 2010.

[Yang et al, 2010] H. Yang, K. Wong, X. Liao, W .Zhang, P. Wei, "A fast image encryption and authentication scheme based on chaotic maps", Communications in Nonlinear Science and Numerical Simulation, Vol. 15, no. 11, pp. 3507-17, ISSN 1007-5704, November 2010.

# *Conclusion et perspectives*

Dans ces travaux de thèse, nous avons étudié, conçu et réalisé, des générateurs de séquences chaotiques, des crypto-systèmes et fonctions de hachage basés chaos, et nous avons étudié et analysé leurs performances.

Dans le chapitre 1, nous avons d'abord, introduit les définitions et généralités sur la sécurité de l'information, nécessaires à la compréhension de la suite des travaux. Puis, nous avons dégagé les hypothèses et caractéristiques principales de la sécurité et ses services, et montré l'intérêt d'utiliser des systèmes basés chaos.

Dans le deuxième chapitre, nous avons étudié la question de la génération des séquences pseudo-chaotiques performantes, qui sont utilisées comme clés dynamiques dans les crypto-systèmes basés chaos ou pour la production des clés secrètes. A ce propos, nous avons d'abord étudié les performances en précision finie de quelques cartes chaotiques connues (Logistique PWLCM, Skew tent, Cat). Puis, nous avons montré l'effet de la précision finie sur les performances obtenues des cartes, et nous avons décrit la procédure de mesure des orbites chaotiques. Ensuite, nous avons présenté la technique de perturbation permettant de pallier l'inconvénient de la précision finie et nous avons décrit la structure des trois générateurs proposés ainsi que leurs performances, selon une panoplie de tests signal et NIST.

Chacun des trois générateurs proposés, inclut la technique de perturbation exposée. Le premier générateur utilise un couplage de fonctions booléennes non linéaires. Le deuxième s'appuie sur une couche de diffusion globale, flexible et efficace, basée sur une carte cat multidimensionnel. Le troisième générateur utilise le couplage en parallèle de deux filtres récursifs, comprenant chacun une carte chaotique en tant que fonction non linéaire.

Dans le troisième chapitre, nous avons conçu et réalisé deux crypto-systèmes basés chaos robustes et rapides. A ce sujet, nous avons tout d'abord, présenté et analysé la structure des crypto-systèmes et spécialement celle de type SPN. De cette analyse, il s'avère que les éléments les plus sensibles de la sécurité des crypto-systèmes sont les couches de confusion et de diffusion. Pour cela, nous avons développé différentes couches (addition de clé, substitution, permutation, diffusion) basés chaos et les couches inverses, très robustes, puisqu'elles sont dynamiques, donc variables au cours des itérations. En nous basant sur ces couches, nous avons décrit ensuite les deux crypto-systèmes proposés, qui sont à la fois robustes vis-à-vis des attaques cryptographiques et adéquats pour des applications de confidentialités en temps réel. Dans notre conception, nous avons aussi gardé à l'esprit la nécessité de pouvoir implémenter les crypto-systèmes développés sous forme logicielle et matérielle (cartes FPGA, cartes à micro-processeurs). Nous avons quantifié les performances des crypto-systèmes proposés, couche par couche et aussi globalement, en nous appuyant sur la panoplie des tests existants dans la littérature (bijectivité, non linéarité, corrélation, indépendance des bits produits en sortie, critère d'avalanche, sensibilité à la clé, temps de calcul, etc).

Enfin, l'analyse des résultats obtenus, démontre l'intérêt de l'approche crypto système basé chaos proposé, à la fois pour la confidentialité des données et la rapidité de l'exécution.

Dans le quatrième chapitre, nous avons proposé une fonction de hachage basée chaos très performante, avec ou sans clé, basée d'une part, sur la structure de Merkle-Damgard, et d'autre part, sur les couches de diffusion et confusion développées dans le deuxième et troisième chapitres. L'analyse des performances obtenues par la fonction de hachage en question, a montré clairement son efficacité pour achever respectivement le service de l'authentification de la source des données et le service de l'intégrité des données.

En perspective, nous allons nous intéresser maintenant aux axes suivants :

- La conception et la réalisation de crypto-systèmes basés sur la structure Feistel.
- L'adaptation des crypto-systèmes proposés en crypto-compression, pour le traitement des différents format d'images compressés JPEG, JPEG 2000, ainsi que les vidéo MJP2 et H.264, afin de fournir plus de rapidité et de sécurité.
- L'extension de la méthode de construction des générateurs chaotiques, en nous appuyant sur les primitives cryptographiques utilisées actuellement, comme celle du HMAC-DRBG.

[Adams et al, 1993] C. M. Adam, S.E. Tavares, "Designing S-boxes for ciphers resistant to differential cryptanalysis", Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography, Rome, Italy, pp. 181–190, 1993.

[Addabbo et al, 2004] T. Addabbo, M. Alioto, S. Bernardi, A. Fort, S. Rocchi, V. Vignoli, "The digital tent map: performance analysis and optimized design as a source of pseudo-random bits", IEEE Transactions on Instrumentation and Measurement, Vol. 2, pp. 1301-1304, May 2004.

[Amigo et al, 2007] J. M. Amigó, L. Kocarev , J. Szczepanski, "Theory and practice of chaotic cryptography", Physics Letters A, Vol. 366, pp. 211–216 , 2007.

[Amin et al, 2010] M. Amin , O. S. Faragallah , A. A. Abd El-Latif, "A chaotic block cipher algorithm for image cryptosystems", Commun Nonlinear Sci Numer Simulat, Vol. 15 , pp. 3484–3497, 2010.

[Aoki et al, 2001] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita, "Camellia: A 128-bit Block Cipher Suitable for Multiple Platforms – Design and Analysis", Selected Areas in Cryptography — 7th Annual International Workshop, SAC2000, Springer LNCS, pp. 39–56, 2001.

[Arnold et Avez, 1967] V. I. Arnold, A. Avez, "Ergodic Problems in Classical Mechanics" , New York, Benjamin, 1967.

[Bakhtiari, 1995] S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk, "Keyed hash functions", Proceedings of the Cryptography: Policy and Algorithms Conference, Lecture Notes in Computer Science, Vol. 1029, Springer-Verlag, Berlin, pp. 201–214, 1995.

[Bellare et al, 1996] M. Bellare, R. Canetti, H. Krawczyk, "Message authentication using hash functions: The HMAC construction", CryptoBytes, Spring 1996.

[Biham et Shamir, 1991] E. Biham, A. Shamir. "Differential cryptanalysis of DES-like cryptosystems". Journal of Cryptology, Vol. 4, no. 1, 1991.

[Billings et Bollt, 2001] L. Billings, E.M. Bollt, "Probability density functions of some skew tent maps", Chaos Solitons Fractals, Vol. 12, pp. 365–376, 2001.

[Blum et al, 1986] L. Blum, M. Blum, M. Shub, "A simple Unpredictable Pseudo-Random Number Generator", SIAM Journal on Computing, Vol. 15, no. 2, pp. 364-383, 1986.

[Brookes, 2005] M. Brookes, "The matrix reference manual", 2005, http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html.

[Canteaut, 2001] A. Canteaut, "Cryptographic functions and design criteria for block ciphers", In Proceedings of Indocrypt, Springer, 2001.

[Chen et al, 2004] G. Chen, Y. Mao, Charles K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps ", Chaos, Solitons Fractals, Vol. 21, pp. 749-761, July 2004.

[Cherrier et al, 2010] E. Cherrier, M. M'Saad, M. Farza, "High-gain observer synchronization for a class of time-delay chaotic systems, Application to secure communications", Journal of Nonlinear Systems and Applications, pp. 102-112, 2010.

[Dachselt et Schwarz, 2001] F. Dachselt, W. Schwarz, "Chaos and cryptography", IEEE Trans. Circuits and Systems–I, Vol. 48, pp. 1498–1509, 2001.

[Daemen et Rijmen, 1999] J. Daemen, V. Rijmen , "AES proposal: the Rijndael block cipher", 1999.

[Daemen et Rijmen, 2002] J. Daemen, V. Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard", Springer, ISBN 3-540-42580-2, 2002.

[Damgard, 1989] I. B. Damgard, "A design principle for hash functions", in: Proceedings Crypto' 89, pp. 416–427, 1989.

[Dawson et al, 1991] M. H. Dawson, S. E. Tavares, "An expanded set of S-box design criteria based on information theory and its relation to differential-like attacks", Proceedings of EUROCRYPT» 91 : Advances in Cryptology, pp. 352–367, 1991.

[Deng et al, 2010] S. Deng, Y. Li, D. Xiao, "Analysis and improvement of a chaos-based Hash function construction", Communications in Nonlinear Science and Numerical Simulation, Vol. 15, no. 5, pp. 1338-1347, May 2010.

[Di et al, 2007] X. Di, L. Xiaofeng, W. Pengcheng, "Analysis and improvement of a chaos-based image encryption algorithm", Chaos Soliton Fract, December 2007.

[Dworkin, 2005] M. Dworkin, National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special", Publication 800-38B, May 2005.

[Ecuyer, 1990] P. L'Ecuyer, "Random numbers for Simulation", Communications of the ACM, 1990, Vol. 33, no. 10, pp. 85-97.

[El Assad et al, 2008] S. El Assad, H. Noura, I. Taralova, "Design and analyses of efficient chaotic generators for crypto-systems", Advances in Electrical and Electronics Engineering- IAENG Special Edition of the World Congress on Engineering and Computer Science, Vol. I, pp. 3-12, 2008.

[El Assad et Noura, 2010-2] S. El Assad, H. Noura, " Brevet France : FR2958100 Procédé de mesure d'orbite de séquences chaotiques et programme d'ordinateur correspondant", Novembre 2010. Extension PCT.

[El Assad et Noura, 2010-1] S. El Assad, H. Noura, "Brevet France : FR2958057 Générateur de séquences chaotiques", Mars 2010. Extension PCT.

[ENT] ENT program, "A Pseudorandom Number Sequence Test Program", Fourmilab, [Online]. Available: http://www.fourmilab.ch/random/

[FIP PUB 197, 2001] National Institute of Standards and Technology: "FIPS-197: Advanced Encryption Standard (AES) ", November 2001.

[FIPS 180-3, 2008] FIPS 180-1, "Secure Hash Standard", Federal Information Processing Standard (FIPS) ", Publication 180-3, National Institute of Standards and Technology, US Department of Commerce, Washington DC, 2008.

[Fournier et al, 2006] D. Fournier, R. Lopez-Ruiz, A. K. Taha,"Route to chaos in three-dimensional maps of logistic type", Inst. für Math. Univ. Graz Grazer Math. Ber. Vol. 350, pp. 82-95, 2006.

[Fournier et al, 2011] D. Fournier, P. Chargé, L. Gardini, "Border Collision Bifurcations and Chaotic Sets in a Two-Dimensional Piecewise Linear Map", Communications in Nonlinear Science and Numerical Simulation Vol. 16, no. 2, pp. 916-927, February 2011.

[Freier et Karlton, 2011] A. Freier, P. Karlton, P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", August 2011.

[Fridrich, 1998] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps", International Journal Bifurcation Chaos, Vol. 8, no. 6, pp. 1259–84, 1998.

[Gotz et al, 1997] M. Gotz, K. Kelber, W. Schwarz, "Discrete-time chaotic encryption systems. I. Statistical design approach", IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications, Vol. 44, no. 10, pp. 963-970, 1997.

[Grapinet et al, 2008] M. Grapinet, V.S. Udaltsov, L. Larger, J. M. Dudley, "Synchronisation and communication with regularly clocked optoelectronic discrete time chaos", Electronics Letters, Vol. 44, no. 12, pp. 764-766, June 2008.

[Grebogi et al, 1998] C. Grebogi, E. Ott, G. A. Yorke, "Roundoff-induced periodicity and the correlation dimension of chaotic attractors", Physical Review A, Vol. 38, no. 7, pp. 3688-3692, 1988.

[Guanrong et al, 2004] C. Guanrong, M. Yaobin, C. Charles, "A symmetric image encryption scheme based on 3D chaotic cat maps", Chaos Soliton Fract, Vol 21, pp. 749–61, 2004.

[Guo et al, 2009] W.Guo, X.Wang, D.He, Y.Cao, "Cryptanalysis on a parallel keyed hash function based on chaotic maps", Physics Letters A, Vol. 373, no. 36, pp. 3201-3206, August 2009.

[Ibrahim et al, 2005] S. Ibrahim, M. A.Maarof, N .B. Idris , "Avalanche Analysis of Extended Feistel Network", Proceedings of the Postgraduate Annual Research Seminar 2005, pp. 265-269, 2005.

[Jakimoski et al ,2001] G. Jakimoski, L. Kocarev, "Chaos and cryptography: block encryption ciphers based on chaotic maps", IEEE Transaction on Circuits and Systems I: Fundamental Theory and Applications, Vol. 48, no. 2, pp. 163-169, 2001.

[Jessa et Walentynowicz, 2001] M. Jessa, M. Walentynowicz, "Statistical properties of number sequences generated by 1D chaotic maps considered as a potential source of pseudorandom number sequences", The 8th IEEE Int. Conference on Electronics, Circuits and Systems, Vol. 1, pp. 449-455, 2001.

[Jessa, 2006] M. Jessa, "On some properties of chaotic maps implemented in finite-state machines", Proceedings International Symposium on Nonlinear Theory and its Applications, pp. 694-694, 2006.

[Kai et al, 2005] W. Kai, P. Wenjiang, Z. Liuhua, S. Aiguo, H. Zhenya, "On the security of 3D cat map based symmetric image encryption scheme", Phys Lett A , pp. 432–439, 2005.

[Katz et Lindell, 2007] J. Katz, Y. Lindell, "Introduction to Modern Cryptography", CRC Press, 2007.

[Kent et Atkinson, 1998] S. Kent, R. Atkinson, "IP Encapsulating Security Payload (ESP) ", IETF. RFC 2406, November 1998.

[Khodor et al, 2010] N. Khodor, J.P Cances, V. Meghdadi, R. Quéré, "Performances of Chaos Coded Modulation Schemes Based on Mod-MAP Mapping and High Dimensional LDPC Based Mod-MAP Mapping with Belief Propagation. IJCNS, Vol. 3, pp. 495-506, 2010.

[Knudsen et al, 1998] Lars R. Knudsen, "Block Ciphers —A Survey", Lecture Notes in Computer Science, 1998, Vol. 1528, pp. 18-48, 1998.

[Knuth, 1998] D. Knuth, "The Art of Computer Programming, Sorting and Searching", second ed, Vol. 3 Addison-Wesley, 1998.

[Kocarev et Jakimoski, 2003] L. Kocarev, G. Jakimoski, "Pseudorandom bits generated by chaotic maps", IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications", Vol. 50, no. 1, pp. 123-126, 2003.

[Koo et al, 2003] B. W. Koo, H. S. Jang, J. H. Song, "Constructing and Cryptanalysis of a 16 × 16 Binary Matrix as a Diffusion Layer", WISA2003, Springer Verlag LNCS, Vol. 2908, pp. 489-503, 2003.

[Koo et al, 2006] B. Koo, H. Jang, J. Song, "On constructing of a 32 × 32 binary matrix as a diffusion layer for a 256-bit block cipher", Proc. ICISC 2006, Springer Verlag LNCS, Vol. 4296, pp. 51–64, 2006.

[Kumar et Ghose, 2010] A . kumar, M. k. Ghose, "Extended substitution-diffusion based image cipher using chaotic standard map", Commun Nonlinear Sci Numer Simulat, Vol. 16, no. 1, pp. 372–382, January 2011.

[Kwok et Tang, 2007] H.S. Kwok, W. K. Tang, "A fast image encryption system based on chaotic maps with finite precision representation", Chaos Solitons Fractals, Vol. 32, no. 4, pp. 1518-1529, May 2007.

[Kwon et al, 2003] D. Kwon, J. Kim, S. Park, et al. , " New block cipher: ARIA", Proc. ICISC 2003, pp. 432–445, 2004 .

[Kwon et al, 2005] D. Kwon, S. H., Sung, J. H. Song, S. Park, "Design Of Block Ciphers and coding theory", Trends in Mathematics Information Center for Mathematical Sciences Vol. 8, pp. 13-20, 2005.

[Lanford, 1998] O.E. Lanford III, "Informal remarks on the orbit structure of discrete approximations to chaotic maps", Experimental Mathematics, 1998, Vol. 7, no. 4, pp. 317-324, 1998.

[Li et al, 2001-1] S. Li, X. Mou, Y. Cai, "Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography", Progress in Cryptology, INDOCRYPT 2001, LNCS, Vol. 2247, pp. 316-329, 2001.

[Li et al, 2001-2] S. Li, Q. Li, W. Li, X. Mou, Y. Cai, "Statistical properties of digital piecewise linear chaotic maps and their roles in cryptography and pseudorandom coding", Proceedings of the IMA International Conference on Cryptography and Coding, Vol. 2260, pp. 205-221, 2001.

[Li et al, 2003] S. Li, X. Mou, , Y. Cai, , Z. Ji, J. Zhang, "On the security of a chaotic encryption scheme: Problems with computerized chaos in finite computing precision", Computer Physics Communications, Vol. 153, pp. 52–58, 2003.

[Lian et al, 2005] S. Lian, J. Sun, Z. Wang, "A block cipher based on a suitable use of the chaotic standard map", Chaos Soliton Fract, Vol 26, pp. 117–29, 2005.

[Lian et al, 2007] S. Lian, J. Sun, J. Wang, Z. Wang, "A chaotic stream cipher and the usage in video protection", Chaos, Solitons & Fractals, 2007, Vol. 34, no. 3, pp. 851-859, 2007.

[Liu et al, 2008] Y. Liu; W.K.S. Tang, H.S. Kwok, "Formulation and analysis of high-dimensional chaotic maps", ISCAS 2008. IEEE International Symposium on Circuits and Systems, pp. 772-775, May 2008.

[Lozi et Cherrier, 2011] R. Lozi, E. Cherrier," Noise-resisting ciphering based on a chaotic multi-stream pseudo-random number generator", IEEE, 6th International Conference for Internet Technology and Secured Transactions, ICITST-2011, pp. 91-96, Abu Dhabi, December 2011.

[Lozi et Fiol, 2009] R. Lozi, C. Fiol, "Global Orbit Patterns for One Dynamical Systems", Iteration Theory ECIT'08, 2009, pp. 1-33, 2009.

[Lozi, 2006] R. Lozi, "Giga-periodic orbits for weakly coupled tent and logistic discretized maps", Anamaya Publishers, New Delhi, India, pp. 80-110, 2006.

[Lozi, 2008] R. Lozi, "New enhanced chaotic number generators", Indian journal of industrial and applied mathematics, Vol. 1, pp. 1–23, 2008.

[Marsaglia, 1996] G. Marsaglia, "Diehard: a battery of tests of randomness," http://stat.fsu.edu/ geo/diehard.html, 1996.

[Masuda et al, 2002] N. Masuda, K. Aihara, "Cryptosystems with discretized chaotic maps", IEEE Transactions on Circuits and Systems I, Vol. 49, pp. 28-40, 2002.

[Masuda et al, 2006] N. Masuda, G. Jakimoski, K. Aihara, L. Kocarev , "Chaotic block ciphers: from theory to practical algorithms", IEEE Transactions on Circuits and Systems I Regular Papers, Vol. I 53, pp. 1341–1352, 2006.

[Matsui, 1993] M. Matsui, "Linear cryptanalysis method for DES cipher", Proc. Eurocrypt 93, Vol. 765 of LNCS, pp. 386-397, Springer, 1993.

[Menezes et al, 1996] A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.

[Menezes et al, 1997] A.J. Menezes, P.C. van Oorschot, S. A. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1997.

[Merkle, 1989] R.C. Merkle, "One way hash functions and DES", in Proceedings Crypto' 89, pp. 428–446, August, 1989.

[Millerioux et Guillot, 2010] G. Millerioux, P. Guillot, "Self-synchronizing stream ciphers and dynamical systems: state of the art and open issues", International Journal of Bifurcation and Chaos Vol. 20, no. 9, pp. 2979-2991, 2010.

[Mister et al, 1996] S. Mister, C. Adams, "Practical S-Box Design", Workshop on Selected Areas in Cryptography (SAC-96), pp. 61–76, 1996.
[Mooney, 2009] A. Mooney, "Chaos Based Digital Watermarking", Intelligent Computing Based on Chaos, Studies in Computational Intelligence, 2009, Vol. 184, pp. 315-332, 2009.

[NIST FIPS 180-3, 2008] NIST FIPS 180-1, "Secure Hash Standard", Federal Information Processing Standard, Publication 180-3, National Institute of Standards and Technology, US Department of Commerce, Washington DC, 2008.

[NIST FIPS 186-3, 2009] NIST FIPS 186-3, "Digital Signature Standard (DSS)", June 2009.

[NIST FIPS 197, 2001] NIST FIPS 197, "Advanced Encryption Standard _ Federal Information Processing Standards Publication 197", 2001.

[NIST SP 800-22, 2008] NIST Special Publication 800-22 rev. 1. "A statistical test suite for random and pseudorandom number generators for cryptographic applications", August 2008.

[NIST SP 800-90A, 2012] NIST Special Publication 800-90A, "Recommendation for Random Number generation Using Deterministic Random Bit Generators", January 2012.

[NIST SP800-90, 2007] NIST Special Publication 800-90, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", Revised Mars 2007.

[Noura et al, 2011] H. Noura, S. el Assad, C. Vladeanu, D. Caragata, "An efficient and secure SPN cryptosystem based on chaotic control parameters", 6th International Conference on Internet Technology and Secured Transactions, ICITST-2011, Abu Dhabi, pp. 226-231, December 2011.

[Peng et al, 2007] J. Peng, M. You, Z. Yang, S. Jin, "Research on a Block Encryption Cipher Based on Chaotic Dynamical System", In Proceedings of the Third International Conference on Natural Computation , IEEE Computer Society, Vol. 5, pp. 744-748, 2007.

[Phatak et al, 1995] S. C. Phatak, S. Suresh Rao, "Logistic map: a possible random number generator", Physics Review E, Vol. 51, no. 4, pp. 3670-3678, 1995.

[RFC 1321, 1992] R. Rivest, "The MD5 message-digest algorithm", IETF Network Working Group, RFC 1321, 1992.

[RFC 3174, 2001] RFC 3174, US, "Secure Hash Algorithm 1", http://www.faqs.org/rfcs/rfc3174.html.

[Rivest et al, 1978] R. Rivest, A. Shamir; L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Vol. 21, no. 2, pp. 120-126, 1978.

[Rivest et al,1998 ] R.L. Rivest, M.J.B. Robshaw, R.Sidney, Y.L. Yin, "The RC6 Block Cipher. v1.1», August 1998.

[Schneider, 1996] B. Schneider, "Applied Cryptography", John Wiley & Sons, ISBN 0-471-12845-7, 1996

[Shannon, 1949] C. E. Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol. 28, pp. 656-715, October 1949.

[Shannon, 1949] C. E. Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol. 28, no. 4, pp. 656-715, 1949.

[Socek et al, 2005 ] D. Socek, S. Li, S. S. Magliveras, B. Furht, " Enhanced 1-D Chaotic Key Based Algorithm for Image Encryption", IEEE Security and Privacy for Emerging Areas in Communications Networks, 2005.

[Stinson, 2006] D. R. Stinson, "Cryptography, Theory and Practice", Third edition. Chapman & Hall/CRC, 2006.

[Tang et al, 2005] G.P. Tang, X.F. Liao, "A method for designing dynamical S-boxes based on discretized chaotic map", Chaos, Solitons & Fractals, Vol. 23, pp. 1901–1909, 2005.

[Tang et al, 2010] Y. Tang, Z. Wang, J. Fang, "Image encryption using chaotic coupled map lattices with time-varying delays", Communications in Nonlinear Science and Numerical Simulation, Vol. 15, no. 9, pp. 2456-2468, 2010 .

[Tao et al, 1998 ] S. Tao, W. Ruli, Y. Yixun, "Perturbance based algorithm to expand cycle length of chaotic key stream," Electronics Letters, Vol. 34, no. 9, 1998, pp. 873-874, 1998.

[Taralova et Fournier, 2002] I. Taralova, D. Fournier, "Dynamical study of a second order DPCM transmission system modeled by a piece-wise linear function", IEEE Transactions of Circuits and Systems I , IEEE publisher Vol. 49, November 2002.

[Ulam et Neumann, 1947] S. Ulam, J. V. Neumann, "Random ergodic theorems", Bull. Amer. Math. Soc. Vol. 51, p. 660, 1945.

[Verhulst, 1845] P. F. Verhulst, "Recherches mathématiques sur la loi d'accroissement de la population", Nouv. mém. de l'Academie Royale des Sci. et Belles-Lettres de Bruxelles, Vol. 18, pp. 1-41, 1845.

[Wang et al, 2004] S. Wang, W. Liu, H. Lu, J. Kuang, G. Hu, "Periodicity of chaotic trajectories in realizations of finite computer precisions and its implication in chaos communications", International Journal of Modern Physics B. 2004, Vol. 18, pp. 2617-2626, 2004.

[Wang et al, 2009] Y. Wang, K. W. Wong, X. Liao, T. Xiang, G. Chen, "A chaos-based image encryption algorithm with variable control parameters", Chaos, Solitons & Fractals, Vol. 41, no. 4, pp. 1773-1783, August 2009.

[Wang et al, 2012] S. Wang, G. Hu, "Coupled map lattice based hash function with collision resistance in single-iteration computation", Information Sciences, Vol. 195, pp. 266-276, July 2012.

[Webster et al, 1986] A.F. Webster, S.E. Tavares, "On the design of S-boxes. In Lecture notes in computer sciences on Advances in cryptology", CRYPTO 85, Springer-Verlag New York, Inc. New York, NY, Vol. 218, pp. 523–534, USA, 1986.

[Wong et al, 2008] K. Wong, S. K. Bernie, W. Law, "A fast image encryption scheme based on chaotic standard map", Phys Lett A, Vol. 372, pp. 2645–2652, 2008.

[Wu et Guan, 2007] X. Wu and Z. Guan, "A novel digital watermark algorithm based on chaotic maps", Physical Letters A, Vol. 365, pp. 403–406, 2007.

[Xiao et al, 2008] D. Xiao, X. Liao, S. Deng, "Parallel keyed hash function construction based on chaotic maps", Physics Letters A, Vol. 372, no. 26, pp. 4682-4688, 2008.

[Xiao et al, 2009-1] Xiao D, Liao XF, Wang Y. "Parallel keyed Hash function construction based on chaotic neural network", Neurocomputing 2009, pp. 2288–2296, 2009.

[Xiao et al, 2009-2] D. Xiao, X. Liao, Y. Wang, "Improving the security of a parallel keyed hash function based on chaotic maps", Physics Letters A, Vol. 373, no. 47, pp. 4346-4353, November 2009.

[Xiao et al, 2010] D. Xiao, W. Peng, X.F. Liao, T. Xiang, "Collision analysis of one kind of chaos-based hash function", Phys Lett A, Vol. 374, pp. 1228–1231, 2010.

[Yang et al, 1998] T. Yang, L.O. Chua, "Application of chaotic digital code-division multiple access (CDMA) to cable communication systems", International Journal of Bifurcation and Chaos, Vol. 8, no. 8, pp. 1657-1669, 1998.

[Yang et al, 2010] H. Yang, K. Wong, X. Liao, W .Zhang, P. Wei, "A fast image encryption and authentication scheme based on chaotic maps", Communications in Nonlinear Science and Numerical Simulation, Vol. 15, no. 11, pp. 3507-17, ISSN 1007-5704, November 2010.

[Zhang et al, 2000] H. Zhang, J. Guo, H. Wang, R. Ding, W. Chen, "Oversampled chaotic map binary sequences: definition, performance and realization", The 2000 IEEE Asia-Pacific Conf. on Circuits and Systems, pp. 618-621, 2000.

[Zheng et al, 2009] G. Zheng, D. Boutat, T. Floquet ,J-P Barbot, "Secure Communication Based on Multi-input Multi-output Chaotic System with Large Message Amplitude'', Chaos Solitons & Fractal, Vol. 41, no. 3, pp. 1510-1517, August 2009.

l'université nantes angers le mans
PÔLE DE RECHERCHE ET D'ENSEIGNEMENT SUPÉRIEUR

UNIVERSITÉ DE NANTES

# Thèse de Doctorat

Hassan NOURA

## Conception et simulation des générateurs, crypto-systèmes et fonctions de hachage basés chaos performants

Design and simulation of efficient chaos based generators, crypto-systems and hash functions

**Résumé**

Dans cette thèse, nous étudions la problématique de la sécurité de l'information basée sur les séquences chaotiques, et ses services à savoir : la confidentialité, l'intégrité des données et l'authentification de la source. D'abord, le problème de la génération des séquences chaotiques performantes est traité. A ce sujet, nous proposons trois générateurs performants, incluant chacun une technique de perturbation afin de palier aux inconvénients de la précision finie. Le premier générateur utilise un couplage de fonctions booléennes non linéaires. Le deuxième s'appuie sur une couche de diffusion globale, flexible et efficace. Le troisième générateur utilise le couplage en parallèle de deux filtres récursifs.

Ensuite la question, de la conception et de la réalisation des crypto-systèmes basés chaos robustes et rapides, est étudiée et analysée. A ce propos, nous proposons deux crypto-systèmes basés chaos dynamiques, de structure SPN, très performants comparés à l'algorithme AES. Le deuxième crypto-système, contient par rapport au premier, une couche de diffusion efficace, mais ne nécessite pas d'opérations de conversion décimal vers binaire et binaire vers décimal. La quantification des performances est réalisée, aux niveaux de chaque couche et globalement, grâce au développement et à l'application d'une panoplie d'outils adéquats. Les deux crypto-systèmes sont réalisables en logiciel et en matériel.

Enfin, une nouvelle fonction de hachage basée chaos performante, avec ou sans clé est proposée. Elle permet d'achever efficacement le service d'authentification de la source et l'intégrité des données.

**Mots clés**

Générateurs chaotiques, crypto-systèmes basés chaos, fonctions de hachage basées chaos, sécurité, intégrité, authentification, critères des performances.

**Abstract**

In this thesis, we study the problem of chaos based information security and its services namely: the confidentiality, the data integrity and the source authentication. Firstly, the generation of efficient chaotic sequences problem is treated. Thus, we propose three efficient generators, each one including a perturbation technique to overcome the inconveniences of the finite precision. The first generator uses a coupling of non linear Boolean functions. The second generator is based on a flexible and effective layer of global diffusion. The third generator uses the parallel coupling of two non linear recursive filters.

Then, the question of the design and the implementation of a strong and fast chaos based crypto-systems is studied and analyzed. Concerning this issue, we propose, two dynamic chaos based crypto-systems of structure SPN, very efficient compared with the AES algorithm. The second crypto-system contains with regard to the first one, a layer of effective diffusion, but does not require operations of decimal towards binary and binary towards decimal conversions. The quantification of the performances is achieved, at every layer and globally, using adequate tools. Both crypto-systems are suitable for software and hardware implementations.

Finally, a new efficient chaos based hash function, with or without key is proposed. It allows effectively the services of the data integrity and the source authentication.

**Key Words**

Chaotic generators, chaos based cryptosystems, chaos based hash functions, security, integrity, authentication, performances criteria.