ROYAUME DU MAROC
Université Mohammed V - RABAT -
جامعة محمد الخامس – الرباط –
Faculté des sciences
كلية العلوم
CENTRE D'ETUDES DOCTORALES - SCIENCES ET TECHNOLOGIES

N° d'ordre 3561

THESE

En vue de l'obtention du : ***DOCTORAT***

Structure de Recherche *: Intelligent Processing Systems & Security (IPSS)*
Discipline *: Informatique*
Spécialité *: Cryptographie*

Présentée et soutenue le 10/12/2021 par :

Khadija ACHKOUN

Contribution des Automates Cellulaires dans la Conception des Systèmes Cryptographiques à Clé Secrète

JURY

Fatima-Zahra BELOUADHA	PES,	Ecole Mohammadia d'Ingénieurs, Université Mohammed V, Rabat	Présidente
Abderrahim TRAGHA	PES,	Faculté des sciences Ben M'Sik, Université Hassan II, Casablanca	Rapporteur/Examinateur
Zine El Abidine GUENNOUN	PES,	Faculté des Sciences, Université Mohammed V, Rabat	Rapporteur/Examinateur
Soumia ZITI	PH,	Faculté des Sciences, Université Mohammed V, Rabat	Rapportrice/Examinatrice
Fouzia OMARY	PES,	Faculté des Sciences, Université Mohammed V, Rabat	Directrice de Thèse

Année Universitaire : 2021/2022

بسم الله الرحمن الرحيم

الحمد لله الذي بنعمته تتم الصالحات، وبفضله تتنزل الخيرات والبركات، وبتوفيقه تتحقق المقاصد والغايات، وأزكى صلواتِ الله وتسليماتِه على المبعوث رحمة للعالمين، نبيِّ الرحمة وإمامِ الهدى سيدنا محمدٌ وآله وصحبه أجمعين

DÉDICACES

Je dédie ce travail à :

- ***A mes chers parents***

  Dont La vertu, le dévouement, l'affection et les valeurs humaines m'ont permis de vivre ce jour.

  Puisse Dieu leur accorder miséricorde, bonne santé, bonheur et prospérité.

- ***A mes chers frères et sœurs***
  - Mon frère Mohammed et mes soeurs : Amina et Aïcha.

    Pour leur amour inconditionnel et leur soutien permanant et incontestable.
  - Mon frère Ammar, ma soeur Hafsa, mes neveux : Rayyan et Rhassan et mes nièces : Yacout et Alia.

    Pour leur amour et leur vivacité.

  Je leur souhaite la réussite dans leur vie, avec tout le bonheur qu'il faut pour les combler.

- ***A mes chers amis***

- ***A tous ceux qui me sont chers et proches.***

Remerciements

Les travaux réalisés dans ce mémoire ont été effectués en cotutelle entre la structure de recherche Intelligent Processing Systems & Security (IPSS) et la Faculté des Sciences de l'Université Mohammed V de Rabat. Avant tout développement, je souhaite d'abord exprimer ma profonde gratitude à tous ceux qui ont, de près ou de loin, contribuer à l'aboutissement de ce travail de thèse.

J'adresse ma reconnaissance la plus absolue à mon encadrante, Professeur **Fouzia OMARY**, pour son encadrement de qualité, sa disponibilité mais aussi ses directives qui ont encadré ce travail et l'ont conduit vers la bonne voie. Son soutien sans faille me fut d'une grande aide pour surmonter toutes les difficultés rencontrées durant cette période et m'a apporté la motivation nécessaire pour toujours aller de l'avant.

Mes sincères remerciements s'adressent également aux membres de jury. Je saisis l'occasion pour remercier **Fatima-Zahra BELOUADHA**, professeur à l'Ecole Mohammedia d'Ingénieurs attachée à l'université Mohammed V de Rabat, d'avoir accepté de présider ce jury. Merci à **Abderrahim TRAGHA** professeur à la Faculté des sciences Ben M'sik de l'université Hassan II de Casablanca, à **Zine El Abidine GUENNOUN**, professeur à la Faculté des Sciences de l'université Mohammed V de Rabat ainsi que **Soumia ZITI**, professeur à la Faculté des sciences de l'université Mohammed V de Rabat d'avoir accepté de juger ce travail de thèse et d'en être les rapporteurs .

Je ne pourrais rater l'occasion d'exprimer ma reconnaissance envers mes chers amis et plus particulièrement Charifa et Fatima Zahra pour leur patience, écoute et conseils.

Enfin, j'adresse toute la gratitude et mes remerciements à ma famille à commencer par mes chers parents pour leur soutien continu, leurs prières, leurs encouragements et leur patience qui m'ont accompagné tout au long de mon parcours académique. Je tiens à remercier particulièrement ma mère qui a eu la patience de m'accompagner, me guider et subir mes humeurs tout au long de ce travail.
Merci également à mes frères et sœurs, Mohammed, Amina, Aicha pour leurs encouragements continus et de m'avoir donné main forte dans les moments les plus difficiles.

Résumé

Pouvant être interceptée par des entités malveillantes, la transmission de données sensibles via un canal non sécurisé est très risquée. Pour remédier à cette faiblesse, plusieurs solutions basées sur la cryptographie ont été développées en vue d'atteindre un bon niveau de sécurité. Les cryptosystèmes interviennent dans ce sens comme une excellente alternative pour assurer la confidentialité des données. Dans la présente thèse, nous nous sommes d'abord intéressés de près à la confidentialité puis à l'authenticité et ce, en premier lieu, à travers BC-CaACO qui représente un nouveau chiffrement par blocs évoluant des ACs unidimensionnels réversibles et irréversibles avec une S-boîte basée sur l'optimisation par colonie de fourmis (ACO). Ce chiffrement est doté de bonnes propriétés de confusion et de diffusion. En deuxième lieu, nous avons présenté un nouveau chiffrement par blocs SPF-CA combinant les deux schémas : SPN et Feistel basé sur les ACs. Ce dernier utilise le générateur PSOCA lors de la génération des sous-clés et fait appel à des S-boîtes dynamiques robustes dépendantes de la clé. Comme résultat, notre chiffrement fait preuve d'une grande efficacité et résistance dépassant les systèmes de chiffrement bien connus. Dans un troisième lieu, SPF-CA 1.2, une variante de SPF-CA, vient explorer le caractère aléatoire des ACs, leurs dégré algébrique élevé ainsi que leur non-linéarité maximale. Durant le processus de génération des sous-clés et au niveau de la fonction du tour de Feistel, SPF-CA 1.2 évolue un ensemble de règle linéaires et non linéaires bien choisi. L'efficacité et la robustesse de ce système ont été vérifiées à travers une analyse de sécurité et ce en réalisant moins de tours. En quatrième et dernier lieu, nous avons introduit CMAC-SPF-CA 1.2, une extension intégrée dans les technologies IoT se basant sur SPF-CA 1.2, un code d'authentification de message démontré résistant aux attaques de falsification et de récupération de clés.

Mots-clés : Sécurité de l'information, Confidentialité, Cryptographie, chiffrement par bloc, Automates cellulaires, S-boîte dynamique, Réseau de substitution-permutation, Schéma de Feistel, Symétrique, Code d'authentification de message, Intégrité, Authentification, IoT.

Abstract

Being able to be intercepted by malicious entities, the transmission of sensitive data via an insecure channel is highly risky. To remedy this weakness, several solutions based on cryptography have been developed in order to achieve a good level of security. Cryptosystems intervene in this sense as an excellent alternative to ensure data confidentiality. In this thesis, we were closely interested in confidentiality and authenticity and this, first of all, through BC-CaACO which represents a new block cipher using one-dimensional reversible and irreversible CA with an S-box based on ant colony optimization (ACO). This encryption has good confusion and diffusion properties. Secondly, we presented a new SPF-CA block cipher combining the two schemes : SPN and Feistel based on CAs. The latter uses the PSOCA generator during the generation of the sub-keys and uses robust dynamic S-boxes dependent on the key. As a result, our encryption high efficiency and resilience goes beyond well-known encryption systems. In a third place, SPF-CA 1.2, a variant of SPF-CA, explores the randomness of CAs, their high algebraic degree as well as their maximum non-linearity. During the process of generating the sub-keys and at the level of the Feistel turn function, SPF-CA 1.2 evolves a well-chosen set of linear and non-linear rules. The efficiency and robustness of this system have been verified through a safety analysis and this by performing fewer rounds. Fourthly and at las, we introduced CMAC-SPF-CA 1.2, an extension built into IoT technologies based on SPF-CA 1.2, a message authentication code proven to be resistant to forgery and key recovery attacks.

Keywords : Information Security, Confidentiality, Cryptography, block cipher, Cellular automata, Dynamic S-Box, substitution-permutation network, Feistel scheme, Symmetric, Message authentication code, Integrity, Authentication. IoT.

Table des matières

Liste des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Liste des tableaux

Liste des algorithmes

Liste des notations et abréviations

AC Automate Cellulaire 20

ACO optimisation par colonies de fourmis 30

ACs Automates Cellulaires 19

BC-CaACO A new block cipher system using cellular automata and ant colony optimization 30

CMAC Cipher Message Authentication Code 67

MAC Code d'Authentification de Message 13

MACs Codes d'Authentification de Message 14

MBPSO Modified binary particle swarm optimization 43

P-boîte boîte de permutation 11

PSOCA Design of new pseudo-random number generator based on non-uniform cellular automata 42

S-boîte boîte de substitution 11

S-boîtes boîtes de substitution 11

SD-boîte boîte de substitution dynamique 42

SPF-CA A new cellular automata based block cipher using key-dependent S-boxes 41

SPF-CA 1.2 An enhanced version of cellular automata-based block cipher system 53

Introduction Générale

Sommaire

1 Motivation

À l'instar des technologies de communication qui sont en perpétuelles évolution, la transmission de données sensibles via un canal non sécurisé est hautement risquée pouvant être interceptée par des entités malveillantes. Par conséquent, la sécurité informatique apparaît dès lors comme une nécessité fondamentale pour protéger ces communications privées. La cryptographie permet de développer de nombreuses solutions pouvant contribuer à une sécurité plus élevée tout en assurant la confidentialité, l'authentification, l'intégrité, la disponibilité et la non-répudiation des données [47].

La cryptographie est une branche de la cryptologie qui comprend également la cryptanalyse. Cette dernière convient à étudier des techniques mathématiques pour découvrir des éventuelles faiblesses dans les cryptosystèmes développés [47]. Elle comporte la cryptographie symétrique, asymétrique et hybride. La cryptographie symétrique utilise une clé secrète dans son processus et comprend fondamentalement trois familles d'algorithmes : les algorithmes de chiffrement par blocs [46], les algorithmes de chiffrement par flot [50] et les codes d'authentification de message (MACs) [64]. Les algorithmes de chiffrement par blocs agissent sur un bloc de taille fixe et sont élaborés à partir d'une fonction itérative, appelée fonction de tour, et d'une clé secrète K. Cette fonction de tour doit effectuer des opérations simples pour assurer la vitesse d'exécution de l'algorithme, telles que : DES [22], IDEA [44], AES [53], . . . etc., tandis que la cryptographie asymétrique [63] utilise deux clés différentes privée et publique [47] comme : RSA [69], ElGamal [68], Diffie-Hellman [49], . . . etc. Par ailleurs, la cryptographie hybride comprend la cryptographie symétrique et asymétrique ; elle vise à combiner les avantages des deux systèmes. Dans ce cadre, de nombreux travaux ont été menés pour répondre aux exigences de la sécurité informatique. Toutefois, ces systèmes cryptographiques deviennent parfois

vulnérables en raison des attaques existantes. De plus, la plupart des algorithmes proposés sont complexes et requièrent une multitude de calculs. Par conséquent, la sécurité est devenue un défi qui doit être surmonté pour fournir un niveau de sécurité raisonnable afin de protéger les données. Cela nous a incité à développer des systèmes cryptographiques offrant une sécurité élevée et une implémentation facile dédiée à des ressources limitées.

Les automates cellulaires (ACs) sont caractérisés par : La simplicité, l'homogénéité, le parallélisme et l'imprévisibilité [78] qui aident à faciliter l'implémentation matérielle et logicielle. Un automate cellulaire est un système dynamique discret qui représente un réseau régulier de cellules toutes identiques. Chaque cellule a un état spécifique qui communique avec ses voisines directes et évolue selon une règle de transition. Le générateur pseudo-aléatoire conçu par S.Wolfram [76], qui est la première application des ACs en cryptographie, fournit une très bonne séquence de nombres pseudo-aléatoires.

En cryptographie, générer des données aléatoires est une tâche difficile car selon le deuxième principe de Kerckhoffs, la sécurité d'un système cryptographique ne doit pas reposer uniquement sur la confidentialité des clés. Par conséquent, la qualité et l'imprévisibilité des clés cryptographiques sont essentielles pour assurer la sécurité des communications grâce aux techniques de la cryptographie moderne. Dans ce contexte, les chercheurs ont commencé à comparer les registres à décalage à rétroactions linéaires (LFSRs) avec des automates cellulaires unidimensionnels à la fin des années 1980. Ils ont montré que les ACs pouvaient atteindre une période maximale par rapport aux générateurs de LFSRs [27, 31]. De plus, pour concevoir un bon générateur pseudo-aléatoire avec une longue période, il est nécessaire de choisir congrûment les règles des ACs. Une variété d'applications cryptographiques a utilisé les ACs telles que : le chiffrement, le hachage, les codes correcteurs d'erreur, . . . etc.

Les travaux de recherche présentés dans cette thèse proposent principalement de nouveaux systèmes de chiffrement par blocs basés sur les ACs visant à fournir un bon niveau de sécurité en moins de temps et de calculs tout en possédant de bonnes propriétés de confusion et de diffusion comme décrites par C. Shannon [62].

2 Contributions

L'objectif principal de cette thèse est de concevoir de nouveaux systèmes de chiffrements par blocs basés sur les automates cellulaires. Nos contributions permettent essentiellement de protéger la confidentialité et l'authenticité, soit l'authentification et l'intégrité, des données :

- **BC-CaACO : Système de chiffrement par blocs utilisant les automates cellulaires et l'optimisation par colonies de fourmis** [30] : nous présentons un nouveau chiffrement par blocs en évoluant des ACs unidimensionnels réversibles et irréversibles avec une boite de substitution basée sur l'optimisation par colonie de fourmis (ACO) [3] en but d'instaurer plus de confusion. Si l'utilisation des S-boîtes et des clés secrètes est conforme aux normes de sécurité, le chiffrement par blocs est considéré comme sécurisé. La S-boîte optimisée basée sur l'ACO [3] est une S-boîte efficace qui fournit en plus de ses propriétés caractéristiques une non-linéarité élevée. En

sus, par rapport aux systèmes de chiffrement par blocs existants, les propriétés spéciales des ACs apportent une meilleure confusion et diffusion.

- **SPF-CA : Système de chiffrement par blocs basé sur des automates cellulaires utilisant des S-boîtes dépendantes des sous-clés** [2] : Nous proposons un nouveau chiffrement par blocs impliquant explicitement les deux schémas : Le réseau de permutation-substitution (SPN) et le schéma de Feistel, basé sur des ACs. La couche de substitution dédiée au schéma SPN utilise une boîte de substitution dynamique (SD-boîte) dépendante de la clé inspirée de [39] tout en évoluant un AC pour augmenter l'effet de confusion et fournir une maximale non-linéarité au sein de notre algorithme proposé. La couche de permutation, à son tour, propose une P-boîte efficace pour augmenter la diffusion. Ces propriétés cryptographiques sont atteintes en utilisant un nombre minimum de tours. De plus, la fonction de Feistel est une évolution d'un AC utilisant la règle 30 introduite par Wolfram [77] et est largement utilisée en cryptographie. En outre, les sous-clés sont générées par PSOCA [29] dans le but d'apporter plus d'aléatoire et de robustesse contre les attaques par clé. En conséquence, notre conception proposée dépassse l'algorithme de chiffrement par blocs bien connu, à savoir AES-192, en terme d'efficacité et de résistance contre les attaques existantes telles que la cryptanalyse différentielle, linéaire mais aussi les attaques statistiques.
- **SPF-CA 1.2 : Variante du système de chiffrement par blocs SPF-CA** [1] : Afin d'exploiter au mieux le degré algébrique élevé, la non-linéarité maximale et le caractère aléatoire que proposent les ACs, nous présentons une variante de SPF-CA, nommée SPF-CA 1.2 utilisant un nouvel ensemble de règles linéaires et non linéaires dans la fonction du tour de Feistel et pendant le processus de génération des sous-clés pour consolider son efficacité et sa robustesse qui ont été appuyées à travers une analyse de sécurité. De plus, le nombre de tours a été réduit tout en conservant les propriétés de confusion et de diffusion élevées de sa primitive, SPF-CA.
- **CMAC basé sur SPF-CA 1.2** : Vu le considérable nombre des appareils interconnectés, ces derniers présentent une haute vulnérabilité aux diverses failles de sécurité dont on cite la falsification des données . Cette fragilité explique le continuel besoin de développer des primitives cryptographiques légères adaptées à la puissance de calcul et à la capacité de stockage de ces dispositifs. A cet effet, nous nous sommes intéressés à la conception d'un MAC qui utilise l'algorithme de chiffrement par blocs SPF-CA 1.2 pour assurer l'intégrité et l'authentification des données tout en vérifiant sa résistance aux attaques notamment les attaques de falsification et de récupération de clés. Ces caractéristiques alignent CMAC basé sur SPF CA 1.2 avec les codes d'authentification de messages existants.

3 Organisation du manuscrit

Ce document comprend cinq chapitres :

- **Chapitre 1 :** Introduit des généralités sur la cryptographie et les ACs en citant, en conclusion, leurs applications dans ce champs.
- **Chapitre 2 :** Détaille notre algorithme de chiffrement par blocs BC-CaACO.
- **Chapitre 3 :** Présente notre système de chiffrement par blocs SPF-CA.
- **Chapitre 4 :** Expose notre système de chiffrement SPF-CA 1.2, une variante de SPF-CA.
- **Chapitre 5 :** Propose un code d'authentification de message qui utilise le système de chiffrement par blocs SPF-CA 1.2.

Enfin, ce manuscrit se termine par une conclusion générale, faisant le point sur nos contributions en décrivant les perspectives de recherches futures.

Chapitre

I

Notions générales sur la cryptographie

Sommaire

Au fil des siècles, un ensemble conçu de protocoles et de mécanismes a été créé pour traiter les problèmes de sécurité de l'information lorsqu'elle est véhiculée par des documents physiques. Souvent, les objectifs de la sécurité de l'information ne sont pas être atteints uniquement par le biais d'algorithmes et de protocoles mathématiques mais nécessitent des techniques procédurales ainsi que le respect des lois pour obtenir le résultat souhaité. [47]

De nos jours, l'information est principalement stockée et transmise sous forme électronique. En vue d'assurer et d'atteindre sa sécurité indépendamment du support physique qui l'enregistre ou la transporte, un vaste éventail de compétences techniques et juridiques est requis. Il n'y a cependant aucune garantie que tous les objectifs de sécurité de l'information jugés nécessaires puissent être atteints de manière adéquate. Les moyens techniques sont fournis par la cryptographie qui comporte un ensemble de techniques pour assurer la sécurité de l'information [47].

La cryptographie est une discipline de la cryptologie qui comprend également la cryptanalyse. C'est la science de la protection des données secrètes échangées via des canaux considérés comme non sécurisés. Quant à la cryptanalyse, il s'agit de l'étude des techniques mathématiques visant à déjouer les techniques cryptographiques, et plus largement, les services de sécurité de l'information.

La cryptographie est scindée en deux classes : La cryptographie classique et moderne. La cryptographie classique décrit la période avant l'avènement des ordinateurs. Elle développe des systèmes basés sur les lettres et les caractères d'une langue naturelle. Les principaux outils utilisés remplacent les caractères par d'autres et les transposent dans un ordre différent. Cela suppose que les procédures de chiffrement et de déchiffrement soient tenues secrètes sinon le système est inefficace, on cite par exemple : Le code de César, le chiffrement de Hill, le chiffrement de Vigenère, les systèmes à clés jetables, etc... Quant à la cryptographie moderne, Les méthodes utilisées de nos jours sont plus complexes, cependant le concept reste le même. La différence fondamentale est que les méthodes modernes manipulent des bits contrairement aux anciennes méthodes. Les règles de chiffrement et de déchiffrement sont connues de tous ce qui offre une garantie sur la sécurité du système. La sécurité de ces méthodes repose uniquement sur la clé. Cette classe contient deux principaux types d'algorithmes de chiffrement : les algorithmes à clé secrète (ou les algorithmes symétriques) et les algorithmes à clé publique (ou les algorithmes asymétriques). Chacun de ces algorithmes détient ses propres avantages et inconvénients. En outre, la cryptographie moderne s'intéresse aux problèmes de sécurité des communications et vise à assurer les exigences suivantes : [47]

- La confidentialité veille à ce que seules les personnes autorisées puissent accéder aux informations. [47]
- L'authentification consiste à vérifier l'identité d'une entité en vue de lui autoriser l'accès à des ressources, tels : systèmes, réseaux, applications... [47]
- L'intégrité veille à ce que les informations ne soient pas modifiées ou détruites, intentionnellement ou accidentellement, et qu'elles conservent leur format d'origine. [47]
- La non-répudiation permet d'empêcher une personne de nier avoir exécuté une opération. [47]
- La disponibilité offre aux utilisateurs autorisés du système un accès rapide et ininterrompu aux informations contenues dans ce système. [47]

Dans ce chapitre, nous introduisons dans la section 1 quelques généralités sur la cryptographie symétrique, notamment les chiffrements par blocs et les MACs comprenant également les classiques attaques cryptographiques existantes. Tandis que, la cryptographie asymétrique est détaillée au niveau de 2. Enfin, nous introduisons les automates cellulaires (ACs) ainsi que leurs différentes propriétés tout en mettant en évidence leur application en cryptographie au niveau de 3 et 4.

1 Cryptographie à clé secrète

La cryptographie à clé secrète met en oeuvre une seule clé pour échanger entre deux interlocuteurs. Cette clé est utilisée pour le chiffrement du texte clair et pendant le déchiffrement du texte chiffré. On en distingue deux catégories : Les systèmes de chiffrement par blocs et les systèmes de chiffrement par flot.

1.1 Chiffrement par blocs

Le chiffrement par blocs prend la clé de chiffrement K et des blocs de texte clair (P) de n bits en entrée et génère un bloc de texte chiffré de n bits (C), où n est la taille d'un bloc [46]. En divisant le message M en blocs de n bits, un message en clair (M) de n'importe quelle longueur peut être chiffré. En utilisant une fonction définie, chaque bloc est chiffré avec une clé K de k-bits. En général, le dernier bloc implique une méthode de remplissage par zéro, si nécessaire, pour que M soit un multiple de n-bits (par exemple, concaténer la valeur $00\ldots.0$, qui sont au nombre de $n - |\mathrm{M}|$, au message M, où $|\mathrm{M}|$ représente le nombre de bits de M) et tous les blocs sont concaténés pour obtenir le texte chiffré. Le processus de déchiffrement fait usage du processus inverse de la fonction de chiffrement qui prend le texte chiffré C et la clé K en entrée.

Définition 1.1. Un chiffrement par blocs n-bit est une fonction $E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$, telle que pour chaque clé $K \in \mathcal{K}$, $E(P,K)$ est une fonction bijective de $\{0,1\}^n$ à $\{0,1\}^n$, notée $E_K(P)$. La fonction inverse est la fonction de déchiffrement, notée $D_K(C)$.

$C = E_K(P)$ indique que le texte chiffré C résulte du chiffrement du texte clair P avec la clé K.

Le chiffrement par blocs doit être exécuté en utilisant l'un des différents modes de chiffrement. Le NIST a proposé les premiers modèles standardisés pour l'algorithme Data Encryption System (DES) [22] et pour l'algorithme Advanced Encryption System (AES) dans [53]. Les modes principaux de base comprennent Electronic CodeBook (ECB), Cipher-Block Chaining (CBC), Cipher FeedBack (CFB) ou Output FeedBack (OFB) et Counter (CTR) [46]. Ces modes sont souvent utilisés pour concevoir d'autres systèmes cryptographiques, tels que des chiffrements à flots, les fonctions de hachage ou les codes d'authentification des messages.

Définition 1.2. Un chiffrement par blocs itéré implique la répétition séquentielle d'une fonction interne appelée fonction du tour. Les paramètres incluent le nombre de tours r, la taille du bloc n-bits et k-bits la taille de la clé d'entrée K à partir de laquelle les r sous-clés K_i sont dérivées. En ce qui concerne l'inversibilité, permettant un déchiffrement unique, la fonction du tour est une bijection sur l'entrée du tour pour chaque valeur K_i. [46]

Dans ce qui suit, nous présentons les principales structures de base des chiffrements par blocs : Le schéma de Feistel et les réseaux de Substitution-Permutation.

1.1.1 Schéma de Feistel

Définition 1.3. Un chiffrement de Feistel est un chiffrement itératif par blocs opérant sur des blocs de $2n$ bits. La fonction itérée F est définie par

$$\begin{array}{rl} \{0,1\}^n \times \{0,1\}^n \longrightarrow & \{0,1\}^n \times \{0,1\}^n \\ (L_{i-1}, R_{i-1}) \longmapsto & (L_i, R_i) \end{array}$$

avec $L_i = R_{i-1}$ et $R_i = L_{i-1} + f\left(R_{i-1}, K_i\right)$

où chaque sous-clé K_i est dérivée de la clé de chiffrement K [46].

La structure du schéma de Feistel est symétrique et est généralement appelée réseau de Feistel. Cela comprend la division de chaque bloc en deux parties égales (R_i et L_i) ; où R_i représente la partie droite, L_i représente la partie gauche, pour $0 \leq i \leq r$ où r est le nombre de tours. La figure 1.1 illustre le processus d'un seul tour du réseau Feistel.

En vue de chiffrer un message, une fonction non linéaire f qui dépend de la clé de tour K_i, qui est dérivée de l'algorithme d'extraction de clés (key scheduling algorithm), est appliquée à la partie R_i. Puis le résultat de cette opération est XORé à l'autre moitié des données ; L_i pour après inverser les deux moitiés dans le tour suivant. Ainsi, le chiffrement correspondant à un seul tour d'une construction Feistel qui s'écrit comme suit :

$$\begin{cases} L_{i+1} & = R_i \\ R_{i+1} & = L_i \oplus f(R_i, K_i) \end{cases} \tag{1.1}$$

Ce processus est itéré pour l'intégralité des tours r. Enfin, les deux parties sont concaténées ($\mathrm{L_r} \| \mathrm{R_r}$) représentant le texte chiffré final.

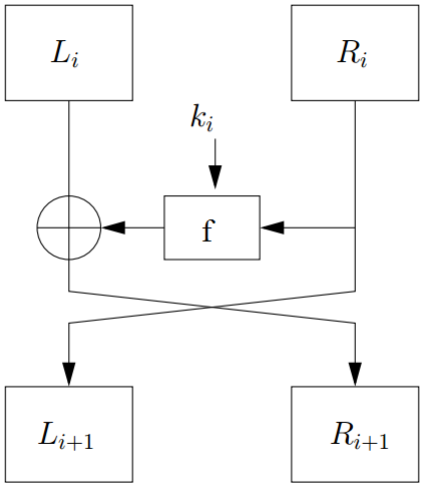


FIGURE 1.1 – Un tour d'un chiffrement de Feistel

Notons que, si la fonction de Feistel f utilisée n'est pas inversible le schéma, quant à lui, l'est. Le déchiffrement est obtenu en utilisant le même processus à r tours en inversant l'ordre des clefs K_i et l'échange du fonctionnement de L_i et R_i. Cette architecture contribue à une implémentation à faible coût dans les logiciels et les matériels. Le premier modèle de chiffrement moderne qui a suivi ce schéma était le Data Encryption System (DES) [70].

En résumé, les principales propriétés des chiffrements par blocs basés sur le réseau de Feistel s'énoncent

comme suit :

- Une fonction de tour simple : la fonction appliquée à l'une des parties gauche ou droite facilite l'étude du code.
- La similarité : Etant similaire au chiffrement, il permet un déchiffrement plus facile. En conséquence, la fonction de tour ne doit pas être bijective. Cependant, il est recommandé que la fonction soit proche de la surjectivité. Sinon, l'attaque proposée par V.Rijmen et al. [54] devient possible.
- Un petit nombre de tours : est utilisé s'il implique une fonction de tour et un algorithme d'extraction de clés robustes.

1.1.2 Réseau de substitutions-permutations (SPN)

Définition 1.4. Un réseau de substitution-permutation (SPN) est un chiffré composé d'un certain nombre d'étapes impliquant chacune des substitutions et des permutations [46].

Un réseau de substitutions-permutations est un système de chiffrement itératif. A chaque étage, une fonction non linéaire est appliquée au bloc d'entrée appelée substitution suivi d'une fonction linéaire appelée permutation. Ces réseaux utilisent deux concepts définis par Shannon [62]. La substitution est généralement représentée par une boîte de substitution (S-boîte) garantissant une bonne confusion. La permutation, à son tour, assure une bonne diffusion de l'information. La figure 1.2 illustre le processus d'un seul tour de SPN. La fonction de tour du chiffrement SPN comprend les opérations suivantes :

1. Ajout d'une sous-clé K_i, généralement par un XOR bit à bit.
2. Une couche de substitution S remplace l'état d'une manière non linéaire par substitution des groupes de bits en parallèle selon certaines tables de substitution. On les appelle généralement boîtes de substitution (S-boîtes). Les S-boîtes dans le chiffrement SPN sont essentiellement bijectives.
3. Une couche de permutation linéaire P ou boîte de permutation (P-boîte) permute les positions des bits.

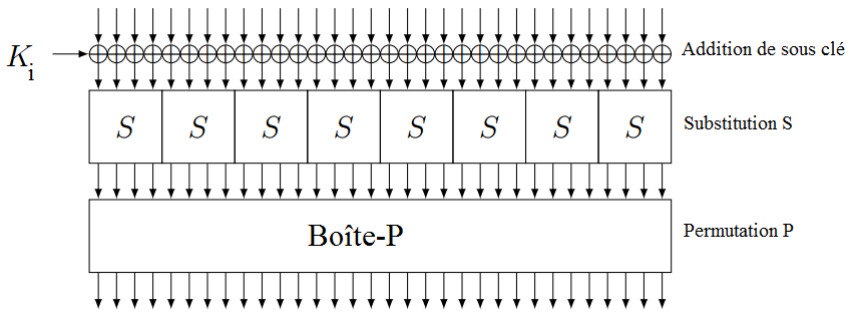


FIGURE 1.2 – Un tour d'un chiffrement de SPN

En plus de ces trois opérations, la fonction de tour comporte parfois une opération supplémentaire qui consiste à ajouter un tour constant pour distinguer entre les tours individuels les uns des autres ce qui limite certains types d'attaques telles que les attaques de diapositives (slide attacks) [6]. La variante standard de la fonction d'un tour peut être décrite comme suit : $X_{i+1} = P\left(S\left(X_i \oplus K_i\right)\right)$ où X_i est le bloc à chiffrer.

Les chiffrements SPN exigent que chaque fonction de tour soit inversible pour assurer le déchiffrement.

1.1.3 Confusion et diffusion

En théorie de l'information, Claude Shannon a introduit deux propriétés importantes [62] que tout bon cryptosystème doit posséder : la confusion et la diffusion.

La propriété de confusion crée une forte dépendance entre le clair, le chiffré et la clé de sorte qu'un attaquant ne réussit pas à obtenir des informations sur la clé pour chaque paire (clair,chiffré). Afin d'atteindre la confusion, deux méthodes sont utilisées : Premièrement, on peut utiliser des opérateurs non-linéaires tels que le ET logique en matériel ou sa version bit-à-bit en logiciel ou encore utiliser l'addition modulaire [72]. Deuxièmement, on peut utiliser des S-boîtes, qui sont choisies pour maximiser la confusion qu'elles provoquent dans un système de chiffrement.

La propriété de diffusion est une notion quantitative qui indique l'interdépendance entre chaque partie du chiffré avec chaque partie du clair et de la clé. Autrement, de petits changements dans l'entrée auront inévitablement un grand impact sur la sortie. Le but de la diffusion est de mélanger la sortie de plusieurs S-boîte opérant sur différentes parties du chiffrement. Plus précisément, la diffusion est assurée par des boîtes de permutation (P-boîte) linéaires, qui appellent des fonctions avec de grands domaines d'entrée et de sortie. Pour des raisons d'efficacité, l'opération de diffusion est préférée d'être simple.

1.2 Chiffrement par flot

Le chiffrement par flot, appelé également chiffrement par flux, est composé d'algorithmes qui chiffrent à la volée en générant une suite chiffrante à partir d'une clé. Cette suite doit se comporter aussi près que possible d'une suite aléatoire afin de masquer au mieux le message [26].

la sécurité du système dépend de la qualité de la suite chiffrante générée. La figure 1.3 illustre la structure générale d'un chiffrement par flot. L'algorithme de chiffrement Rivest Cipher 4 (RC4) [65], développé par Ron Rivest en 1987, est un exemple d'algorithme de chiffrement par flot qui fonctionne avec des clés de tailles variables.

Les chiffrements par flot sont généralement subdivisés en deux types de base : synchrone et asynchrone. Dans le chiffrement par flot synchrone, la suite chiffrante utilise uniquement la génération de la clé secrète du chiffrement. Dans le chiffrement par flot asynchrone, la suite chiffrante est générée en utilisant la clé secrète et un nombre de bits ou de mots [50].

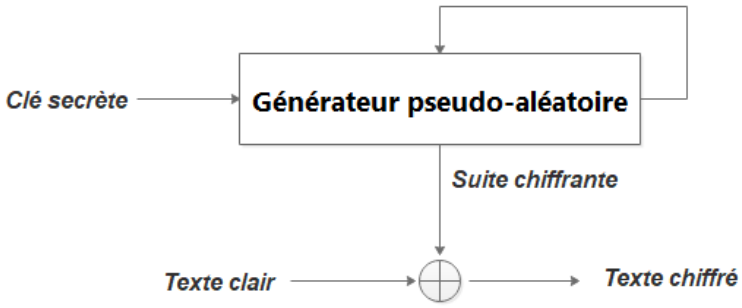


FIGURE 1.3 – Structure générale d'un chiffrement par flot

Un exemple de chiffrement par flot, proposé par Vernam en 1917, est le masque jetable (one-time pad). Dans ce schéma, le texte chiffré C_k est le résultat d'une combinaison du texte clair $\mathrm{m_k}$ avec la clé $\mathrm{K_k}$, de même taille, par un XOR ou $\mathrm{C_k} = \mathrm{M_k} \oplus \mathrm{K_k}$. A partir du texte chiffré, le texte clair peut être retrouvé par l'opération inverse : $\mathrm{M_k} = \mathrm{C_k} \oplus \mathrm{K_k}$. Le flux des clés $\mathrm{K_k}$ est utilisé une et une seule fois pour chiffrer les messages en clairs $\mathrm{M_k}$, d'où le nom de masque jetable. Ce flux des clés $\mathrm{K_k}$ est généré de façon aléatoire et indépendamment des flux passés.

1.3 Code d'authentification de message

Un Code d'Authentification de Message (MAC) est un algorithme à clé secrète qui assure l'authentification et l'intégrité du message et non pas la confidentialité. Il comprend deux algorithmes : l'algorithme de génération (MACG_k) et l'algorithme de vérification (MACV_k). L'algorithme MACG_k prend un message M de taille arbitraire et une clé secrète k et génère une empreinte, appelée Tag, jointe au message M. De plus, l'algorithme MACV_k est utilisé pour vérifier que la paire reçue (M, MACG_k(M)) est bien égale à celle régénérée par le récepteur [64] et qu'elle n'a pas été modifiée. En particulier, la clé k ne permet à ses possesseurs de s'authentifier qu'en générant le Tag. La figure 1.4 illustre cette configuration.

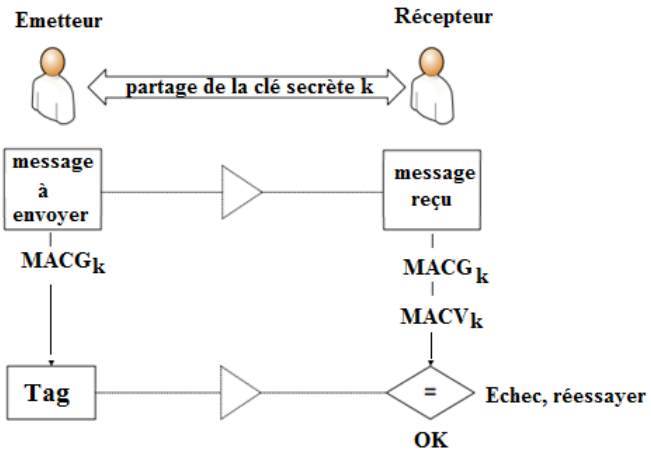


FIGURE 1.4 – Schéma d'un code d'authentification de message

D'une manière générale, il existe deux façons de concevoir les Codes d'Authentification de Message (MACs) : La première est basée sur un algorithme de chiffrement symétrique [64] en mode CBC ou CFB, dans ce cas le Tag est le dernier bloc chiffré obtenu. En outre, la deuxième conception est basée sur la fonction de hachage à clé qui intègre une clé K dans le processus de calcul du Tag. En effet, la sécurité d'un MAC repose sur sa résistance aux attaques cryptographiques dédiées aux fonctions de hachage et aux algorithmes de chiffrement symétrique (voir les sections 1.1 et 1.3.3).

Etant donné une fonction MAC h_k paramétrée par une clé secrète k et une entrée x de longueur arbitraire; l'algorithme MAC demande de vérifier les propriétés suivantes :

- Facile à calculer : $h_k(x)$ doit être facile à calculer.
- Compression : la fonction h_k prend x en entrée et produit une sortie $h_k(x)$ de longueur fixe.
- Résistance au calcul : Etant donné un ensemble de paires $(x', h_k(x'))$, il est difficile de calculer une paire $(x, h_k(x))$ pour chaque $x \neq x'$.

Si la $3^{ème}$ propriété n'est pas vérifiée, l'algorithme MAC sera soumis à une attaque de falsification (MAC forgery attack).

1.3.1 *Attaque de falsification contre MAC*

L'attaque de falsification contre les MACs est l'attaque la plus connue contre l'algorithme MAC, dans laquelle une entité malveillante peut délivrer au moins une paire valide de (message,Tag) quelle que soit la clé secrète. Considérons un message noté x tel que $\text{MACG}_k(x) = y$. Il existe deux modèles d'attaques de falsification [48] :

- L'attaque de falsification existentielle : l'attaquant peut générer une nouvelle paire valide (message, Tag) pour un message x_i (que la valeur de ce texte soit vérifiée ou non). Particulièrement,

l'attaquant choisit aléatoirement une clé k puis il calcule le Tag $\text{MACG}_k(x_i)$ pour chaque x_i dans un temps q. L'attaque est réussie si et seulement si $x \notin \{x_1, \ldots, x_q\}$ et $\text{MACG}_k(x) = y$.

- L'attaque de falsification sélective : L'attaquant peut générer une nouvelle paire (message,Tag) pour un message x_i de son choix (ou peut être en partie sous son contrôle). En particulier, l'attaquant choisit une clé k et un message x_i aléatoirement puis il calcule $\text{MACG}_k(x_i)$ pour $x_i \neq x$. Si $\text{MACG}_k(x_i) = y$, l'attaque est reussie.

Un MAC doit également résister à l'attaque de récupération de clé (key recovery attack) décrite dans la section 1.4. En général, une attaque de récupération de clé réussie sur l'algorithme MAC nécessite un nombre quelconque de paires forgées (message, Tag) afin de construire la clé de manière triviale.

1.3.2 MAC basé sur un chiffrement par bloc en mode CBC

Le CBC-MAC repose sur l'utilisation d'une seule clé secrète k à chaque itération du chiffrement noté E_K. Dans la figure 1.5, le premier bloc M_1 est chiffré avec la clé k et le résultat du chiffrement est considéré comme une entrée de la fonction E_K après avoir appliqué l'opérateur XOR à la paire $(E_k(M_1), M_2)$. Ainsi, l'opération est répétée jusqu'à ce que le dernier chiffré $(E_k(M_{n-1}), M_n)$ représentant le Tag soit obtenu.
La fonction de compression pour CBC-MAC peut être exprimée comme suit :

$$Tag_{i+1} = E_k(M_i \oplus E_k(M_{i-1})) \tag{1.2}$$

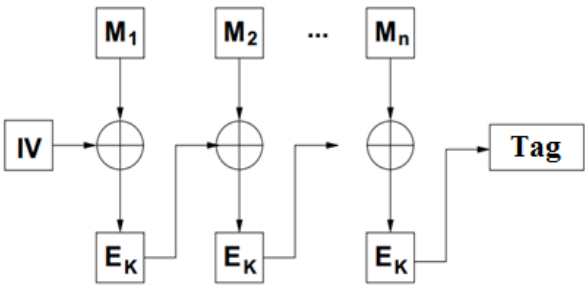


FIGURE 1.5 – MAC basé sur un chiffrement par bloc en mode CBC (CBC-MAC)

1.3.3 MAC basé sur les fonctions de hachage

Une fonction de hachage cryptographique H est une fonction qui prend en entrée une séquence de bits de n'importe quelle longueur et génère en sortie une séquence de bits de longueur fixe (appelée "empreinte" ou digest en anglais ou simplement "haché") [64]. La fonction de hachage est principalement utilisée pour assurer l'intégrité des données : si l'empreinte du texte clair change, alors le texte clair a également changé. Les fonctions de hachage les plus anciennes incluent Secure Hash Algorithm 1 (SHA-1), qui fournit un haché de 160 bits, et Message Digest 5 (MD-5), qui fournit un haché de 128

bits. Des faiblesses ont été trouvées dans MD5 et SHA-1 : il est recommandé d'utiliser des méthodes alternatives plus récentes, telles que les fonctions SHA-2 et SHA-3.

Les fonctions de hachage cryptographiques sont utilisées dans la conception de protocoles de sécurité de l'information notamment dans le cas des MACs. Elles sont également utilisées dans les signatures électroniques et d'autres formes d'authentification.

Une fonction de hachage $h : \{0,1\}^* \rightarrow \{0,1\}^n$ est dite sécurisée si les trois propriétés de sécurité de base suivantes sont remplies [64] :

- Résistance en pré-image : cet attribut explique la nature unidirectionnelle de la fonction. Formellement, étant donné le haché y, il est difficile de trouver un message x tel que $h(x) = y$
- Résistance en seconde pré-image : Étant donné un message x, il est difficile de trouver un autre message x′ différent de x tel que $h(x') = h(x)$.
- Résistance en collision : il est difficile de trouver deux messages différents x et x′ tels que $h(x) = h(x')$.

Dans ces points, la difficulté réside dans le fait qu'il n'est possible de résoudre ces problèmes qu'avec des méthodes génériques. Etant donné une empreinte y de longueur n, pour trouver une pré-image de y, la méthode générique consiste à tirer un message x aléatoirement uniformément jusqu'à l'obtention de $h(x) = y$. En suivant le même raisonnement, on tire x′ jusqu'à obtenir $h(x') = h(x)$ pour la recherche de la seconde pré-image [64]. La complexité de cette attaque est d'environ 2^n opérations pour trouver une pré-image ou une seconde pré-image. Finalement, pour la recherche de collision, on construit une table contenant toutes les images $\mathrm{h(x)}$ pour tout message x tiré aléatoirement uniformément. On s'arrête lorsqu'on a trouvé un x tel que $\mathrm{h(x)}$ se trouve déjà dans la table. En utilisant l'attaque de paradoxe des anniversaires (voir la section 1.4), pour trouver une collision dans un ensemble de taille 2^n est estimée à $2^{n/2}$. En résumé, une fonction de hachage est considérée bonne s'il n'y a pas de méthode efficace autre que ces méthodes génériques : soit 2^n pour la pré-image et la seconde pré-image, et $2^{n/2}$ pour les collisions [64].

Le principe de conception d'un MAC sécurisé basé sur une fonction de hachage H est d'intégrer la clé K dans le calcul du Tag. Spécialement, en utilisant cette méthode nous pouvons définir les fonctions de hachages à clé comme une famille de fonctions.

$$\mathrm{HMAC}_k(M) = H[(K \oplus \mathrm{opad})||H[(K \oplus \mathrm{ipad})||M]] \tag{1.3}$$

avec :

- M : le message d'entrée de l'algorithme.

- K : la clé permettant le calcul du MAC.
- ipad : ox36 (répété b/8 fois) ou b est la taille du bloc
- opad : ox5C (répété b/8 fois)

Ce HMAC est largement utilisé dans de nombreux protocoles de réseau, tels que SSH, IPSec, TLS . . . etc. La robustesse de HMAC dépend de celle de la fonction de hachage ainsi que de la taille et de la qualité de la clé.

1.4 Attaques cryptographiques

Le chiffrement a pour but d'assurer la confidentialité des données. Ces données peuvent être interceptées par un attaquant s'il réussit à récupérer le texte en clair qui correspond au texte chiffré. Un chiffrement peut être cassé si un adversaire est capable de trouver une clé ou une partie du texte clair à partir du texte chiffré. Les exigences de sécurité des clairs sont fournies par trois grandes classes [59] :

La résistance de récupération du texte clair (Plaintext recovery resistance) : Elle consiste à résister à l'attaque qui récupère avec succès la valeur du texte clair correspondante à partir de K et C ; telle que $E_K(P) = C$, où K est une clé arbitraire et C un texte chiffré arbitraire.

La résistance de récupération de la clé (Key recovery resistance) : Elle consiste à résister à l'attaque qui récupère avec succès la valeur de K.

Indiscernabilité (Indistinguishability) : La transformation E_K doit être indiscernable d'une permutation aléatoire pour K, c'est-à-dire qu'un adversaire ne pourrait pas calculer la bonne paire (clair,chiffré) sans connaitre la clé K.

Dans un système de chiffrement, si la résistance de récupération de la clé est cassée, les deux notions seront automatiquement cassées. Par conséquent, cette notion de sécurité est la plus forte parmi les trois ci-dessus. En outre, les mesures de la sécurité d'un chiffrement doivent résister aux modèles d'attaque que l'attaquant peut utiliser qui sont essentiellement classées en fonction de l'information que les attaquants peuvent atteindre, citant les modèles suivants :

Attaque sur texte chiffré seul (Ciphertext only attack) : L'attaquant tente d'obtenir le texte clair à partir d'un échantillon de texte chiffré.

Attaque à texte clair connu (Known plaintext attack) : L'attaquant peut détecter des combinaisons de texte clair et le texte chiffré correspondant. Le but de cette attaque est de simuler un attaquant qui est à l'écoute de la communication entre deux interlocuteurs. Il n'a aucun contrôle sur le texte clair à chiffrer. La cryptanalyse linéaire entre dans cette catégorie.

Attaque à texte clair choisi (Chosen plaintext attack) : L'attaquant peut obtenir le texte chiffré correspondant au texte clair choisi arbitrairement. Cette attaque est plus puissante que l'attaque à texte clair connu. L'attaquant vise à récupérer toute la clé ou une partie. La cryptanalyse différentielle en est un exemple.

Attaque adaptative à texte clair choisi (Adaptive chosen plaintext attack) : l'attaquant commence par une attaque à texte clair choisi dans le premier tour puis il adapte d'autres tours de déchiffrement en reposant sur le premier tour.

Attaque à texte chiffré choisi (Chosen ciphertext attack) : Il s'agit d'un modèle similaire à l'Attaque à texte clair choisi. L'attaquant peut obtenir le texte clair correspondant pour n'importe quel texte chiffré.

Attaque adaptative à texte chiffré choisi (Adaptive chosen ciphertext attack) : Il s'agit d'un modèle similaire à l'attaque adaptative à texte clair choisi. Une fois que l'attaquant a choisi les chiffrés que le système va déchiffrer, il est possible de demander les textes clairs de ces chiffrés.

Dans de nombreux cas, les paramètres d'un chiffrement doivent répondre aux exigences de sécurité. Par conséquent, on doit garantir l'indiscernabilité contre l'attaque du texte clair et l'attaque du texte chiffré [59]. Il existe d'autres modèles d'attaques cryptographiques, notamment :

Attaque de recherche exhaustive de la clé ou attaque en force brute : Elle dépend de la génération d'espaces de clés possibles. Le texte clair sera récupéré s'il reste suffisamment de temps. Il s'agit d'une attaque efficace contre tous les systèmes symétriques basés sur l'utilisation de clés hormis le masque jetable (one-time pad), car pour les algorithmes à clé publique où la difficulté n'est pas de trouver la bonne clé par attaque en force brute, mais plutôt de dériver la clé privée à partir de la clé publique [59].

Cryptanalyse différentielle : C'est une sorte d'attaque adaptative à texte clair choisi qui peut être appliquée aux algorithmes de chiffrement itératif par blocs et aux fonctions de hachage. Elle cherche à trouver la différence entre les clairs et leurs chiffrés correspondants à chaque itération de l'algorithme de chiffrement. L'attaquant utilise alors une analyse statistique [63] pour déterminer une partie de la clé ou de réduire suffisamment le nombre de clés possibles pour pouvoir mener une attaque en force brute rapide.

Cryptanalyse linéaire : C'est une sorte d'attaque à texte clair connu dans laquelle l'attaquant trouve des paires (texte clair, texte chiffré) créés avec la même clé. Les paires sont étudiées pour obtenir l'intégralité la clé utilisée pour les avoir [63].

Paradoxe des anniversaires : Cette attaque est basée sur le fait que dans un groupe d'au moins 23 personnes choisies aléatoirement, la probabilité que deux personnes du groupe aient le même anniversaire est supérieure à 50% [67]. Le but de cette attaque est de modifier les messages transmis entre deux ou plusieurs entités.

Regroupement de clés ou Key Clustering : L'objectif principal d'un système de chiffrement cryptographique est qu'une seule clé puisse dériver le texte clair à partir du texte chiffré. Regroupement de clés permet d'utiliser deux clés différentes appliquées au même texte clair pour produire le même chiffré [46].

2 Cryptographie à clé publique

Les cryptosystèmes asymétriques ou à clé publique diffèrent entre autres dans la gestion de la clé. Ils utilisent deux clés différentes : l'une pour chiffrer appelée la clé publique et l'autre pour déchiffrer appelée la clé privée. La sécurité est donc basée sur le fait que même s'il connait la clé du chiffrement, l'attaquant ne peut pas trouver la clé de déchiffrement.

La mise en oeuvre d'un cryptosystème à clé publique se fait suivant ces 4 étapes indispensables [67] :

1. Chaque entité produit la paire clé publique-clé privée.
2. Chaque entité divulgue la clé publique dans un registre ou fichier publique et détient la clé privée.
3. Pour qu'une entité A envoie un message secret à une autre entité B, elle le chiffre en utilisant la clé publique puis transmet le message obtenu à B.
4. Quand B reçoit le message provenant de A, elle le déchiffre en utilisant la clé privée.

La cryptographie à clé publique est utilisée dans différentes applications telles que : le chiffrement, la signature numérique et le partage de clés secrètes. Voici quelques exemples : Les algorithmes ECC (Elliptic Curve Cryptosystems) [36], RSA [69], Diffie-Hellman [49] et ElGamal [68].

3 Automates cellulaires

En vue de simuler des mécanismes d'auto-réplication, le mathématicien J. von Neumann [74] a introduit les Automates Cellulaires (ACs) au début des années 1950. Ces derniers sont utilisés dans différentes applications, telles que la biologie et la physique, où ils sont considérés comme un modèle dynamique discret de systèmes complexes. En fait, un automate cellulaire spécifie un réseau cellulaire régulier où les cellules sont toutes identiques. Chaque cellule possède un état spécifique qui communique avec ses voisines directes et évolue en fonction d'une règle de mise à jour définie localement.

Récemment, les ACs sont devenus un outil efficace pour le développement de systèmes cryptographiques grâce à leurs caractéristiques importantes dont on cite : l'homogénéité, la localité, le parallélisme, la simplicité et l'imprévisibilité [78] ce qui les rend faciles à implémenter, qu'il s'agisse de matériel ou de logiciel.

Par la suite, les concepts, les catégories et les principales propriétés des ACs seront discutés :

Définition 3.1. Un automate cellulaire $A = \{d, \mathbb{S}, \nu, f, \Phi\}$ est la donnée de :

- Sa dimension $d \in \mathbb{N}^*$, qui est la dimension de son réseau $\mathbb{Z}^d$.
- Un ensemble fini d'états $\mathbb{S}$, appelé alphabet.
- Son voisinage $\nu = (\nu_1, \nu_2, \ldots, \nu_n) \in \left(\mathbb{Z}^d\right)^n$.
- Sa fonction de transition locale f appelée la règle locale de l'Automate Cellulaire (AC) $f : \mathbb{S}^{|\nu|} \longrightarrow \mathbb{S}$, avec $|\nu|$ est le cardinal de ν.
- Sa fonction de transition globale Φ appelée la règle globale de l'AC $\Phi : \{0,1\}^* \to \{0,1\}^*$

Un automate cellulaire est donc caractérisé par sa dimension, son ensemble d'état, son voisinage et ses fonctions de transition.

Définition 3.2. Soit un automate cellulaire A, la configuration de A est son état global à un instant t.

Définition 3.3. On appelle automate cellulaire élémentaire tout automate $\mathcal{A} = (d, \mathbb{S}, \nu, f)$ tel que :

- $\mathrm{d} = 1$: Les cellules de l'automate correspondent à des entiers
- $\mathbb{S} = \{0, 1\}$: les cellules n'ont que deux états possibles
- $\nu = \{-1, 0, 1\}$: trois voisinages au total pour chaque cellule.
- f fonction de transition dont le fonctionnement est le suivant : l'état de la cellule i au temps $\mathrm{t} + 1$ (noté $x_i^{t+1}\big)$ dépend des états des cellules $\mathrm{i} - 1$, i et $\mathrm{i} + 1$ au temps t (le voisinage de la cellule i de rayon ı) :

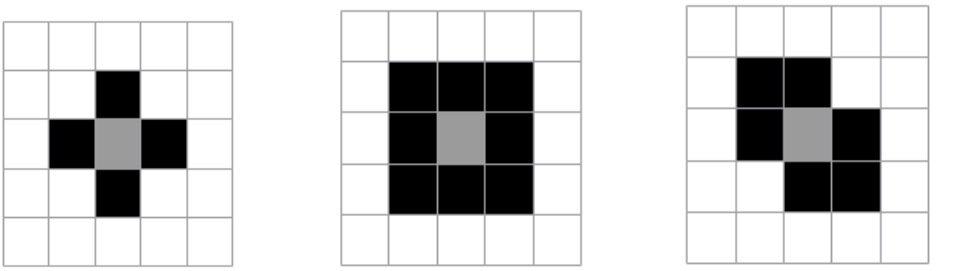
$$x_i^{t+1} = f\left(x_{i-1}^t, x_i^t, x_{i+1}^t\right)$$

3.1 Définitions et généralités

3.1.1 Voisinages

Le voisinage est l'élément de base des automates cellulaires. Il peut déterminer certains paramètres comme l'espace des fonctions de transition qui sont utilisées dans l'évolution d'un AC. Nous avons mentionné dans la section précédente que nous ne nous sommes intéressés qu'aux ACs unidimensionnels. Dans ce cas, le voisinage le plus usuel pour ce type d'AC qui fait référence au voisinage le plus proche et le plus couramment utilisé est $\{-1, 0, 1\}$ qui désigne la cellule de droite et gauche y compris la cellule centrale. Comme extension de cette notation, le voisinage d'un AC de rayon r peut être représenté par $\nu = \{-r, -r+1, \ldots, 0, \ldots, r-1, r\}$. La taille standard de ce voisinage $|\nu| = 2r+1$ où le rayon est identique dans toutes les directions. Il existe trois types de voisinage classique pour un

AC bidimensionnel : Von Neumann, Moore et Margulous. La figure 1.6 illustre l'espace des cellules pris pour chaque type de voisinage.



(a) Voisinage de Von Neumann (b) Voisinage de Moore (c) Voisinage de Margulous

FIGURE 1.6 – Les différents types de voisinage d'un AC.

Définition 3.4. Si un AC utilise un rayon $r = 1$, on dit que l'AC est d'ordre 1 sinon $(r \geq 2)$ l'AC est d'ordre 2.

3.1.2 Fonction de transition

La fonction de transition d'un AC définit le comportement typique de l'automate qui ne dépend que de l'alphabet S et du voisinage ν choisi. On utilise s_i^t pour représenter l'état d'une cellule i au temps t. La fonction de transition locale $f : \mathbb{S}^{|\nu|} \longrightarrow \mathbb{S}$ transforme la cellule i à son état suivant s_i^{t+1} en utilisant l'expression suivante :

$$s_i^{t+1} = f\left(s_{i-r}^t s_{i-r+1}^t, \ldots, s_i^t, \ldots, s_{i+r-1}^t, s_{i+r}^t\right) \tag{1.4}$$

En effet, la fonction f peut être représentée sous la forme d'une table de vérité qui contient le voisinage d'une cellule et son état prochain correspondant (voir le tableau 1.1). Elle est également représentée par une valeur binaire, qu'on appelle le numéro de règle.

Prenons l'exemple de la règle 30 la plus connue décrite par Wolfram [76] qui est inspirée du jeu de la vie. On écrit la forme binaire correspondant à cette règle $30 = (00011110)_2$, qui correspond à l'expression logique suivante 1.5 :

$$x_i^{t+1} = x_{i-1}^t \oplus \left(x_i^t \vee x_{i+1}^t\right) \tag{1.5}$$

où $\oplus$: OU Exclusif $\vee$: ET logique

Considérons le voisinage $\nu = \{i-1, i, i+1\}$ (i.e. les voisins d'une cellule i sont $i-1$ et $i+1$) et $f : \mathbb{S}^{|\nu|} \longrightarrow \mathbb{S}$, comme défini par le tableau 1.1.

Cela signifie que si à un temps t donné, une cellule est à l'état ' o ' et son voisinage est $(0, 0, 1)$, elle

sera à l'état 'ı' au temps $t + 1$. Si l'on part d'une configuration initiale dans laquelle toutes les cellules sont à l'état 'o', à l'exception d'une, on aboutira à la séquence de configuration décrite à la figure ı.7, également appelée diagramme espace-temps, dans laquelle l'ensemble des états est représenté par des cellules blanches pour les 'o' et des cellules en noir par les 'ı'.

ν_i^t	ıı	ıo	oı	oo	oı	oo	oı	oo
$f(\nu_i^t)$	o	o	o	ı	ı	ı	ı	o

Tableau ı.ı – Représentation de la règle 30

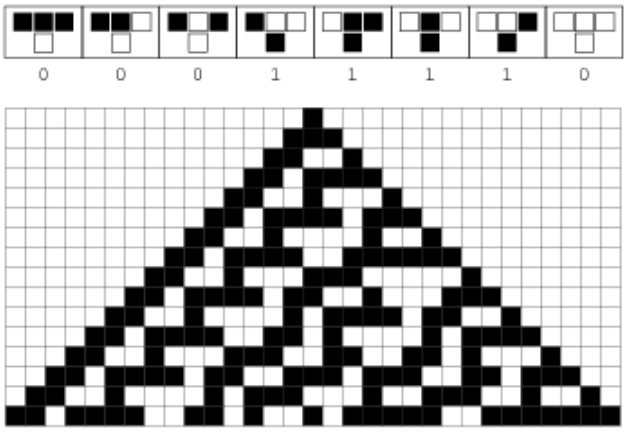


Figure ı.7 – Diagramme espace-temps de la règle 30

Après avoir appliqué la fonction de transition locale f à toutes les cellules, la fonction de transition globale Φ évolue en changeant toute la configuration vers une nouvelle [35]. Elle est définie par $C^{t+1} = \Phi\left(C^t\right)$

À certaines fins cryptographiques, la fonction f doit être injective ; c'est à dire que chaque entrée de la fonction a une sortie spécifique. Cette propriété d'injectivité a été l'objet principal dans la recherche des ACs. En fait, on dit qu'un AC est injectif si et seulement si sa fonction de transition globale Φ l'est. Cependant, un AC est bijectif si et seulement si Φ est injective et surjective, on parle donc de la réversibilité, ce qui signifie que chaque configuration a exactement un successeur et un prédécesseur. Par conséquent, les fonctions de transition réversibles peuvent être utilisées pour revenir à un état antérieur à tout moment.

D'après Kari [34], pour les automates unidimensionnels, le problème de réversibilité est bien résolu alors qu'il est indécidable pour les AC au delà de 2 dimensions.

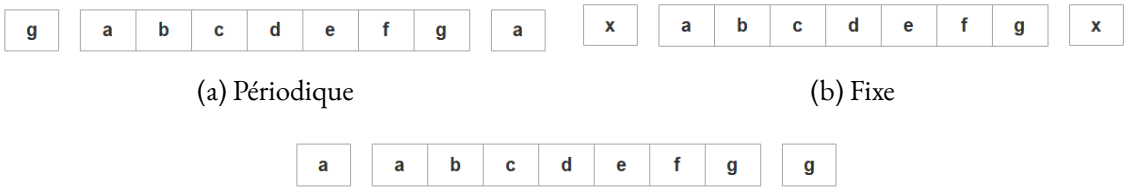
3.1.3 Conditions aux limites

En ce qui concerne les extrémités du réseau, situées à l'extrême gauche et droite dans le cas des ACs unidimensionnels, elles auront besoin de cellules voisines pour pouvoir interagir. Il faut donc définir

des conditions aux limites. Il convient de noter qu'il en découle deux principaux types, comme c'est illustré dans 1.9, à savoir les conditions aux limites fixes et les conditions aux limites périodiques [78]. Dans le cas d'une limite fixe, les cellules extrêmes reçoivent certaines valeurs d'états constantes. Parmi les conditions aux limites fixes, la limite la plus courante est la limite nulle où les cellules extrêmes sont connectées par des états logiques 0 [13]. Dans les ACs de limites périodiques, les cellules extrêmes sont reliées l'une à l'autre et forment un cycle.

La figure 1.8 illustre les conditions aux limites périodiques telles que pour un rayon $r = 1$, le voisinage de la première cellule est (cellule n, cellule 0, cellule 1) et le voisinage de la $n^{ème}$ cellule est (cellule n-1, cellule n, cellule 0). En outre, il y a d'autres variantes de conditions aux limites : la limite adiabatique, la limite réflexive [73] et la limite intermédiaire [13].

FIGURE 1.8 – Conditions aux limites périodiques



(a) Périodique

(b) Fixe

(c) Réflexive

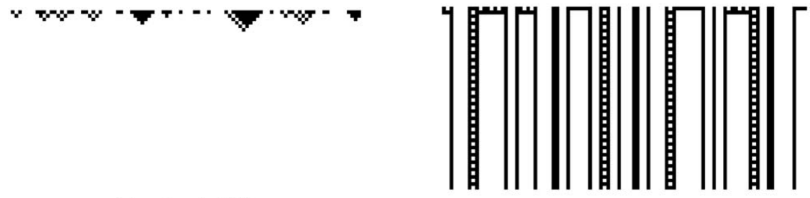
FIGURE 1.9 – Types des conditions aux limites

3.1.4 *Systèmes de classification et propriétés d'un AC*

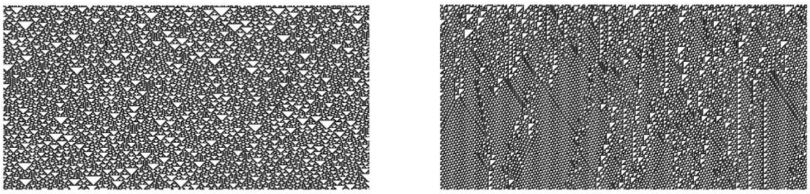
Wolfram a proposé quatre principales classes qui divisent les règles des ACs selon leurs évolutions [78]. Il s'est appuyé sur le résultat d'observation d'extraits de diagrammes espace-temps de l'AC à partir d'un ensemble d'états initiaux tirés au hasard. (voir figure 1.10)

- Classe I : l'évolution du système tend vers une configuration homogène c'est-à-dire que toutes les cellules sont dans le même état et n'évoluent plus après un nombre fini d'états. Par exemple, on peut citer les règles de numéros : 40, 172, 234.
- Classe II : L'évolution de l'automate tend vers un ensemble de structures périodiques stables. Les règles de numéros 24, 108, 218 sont des exemples de cette classe.
- Classe III : L'évolution conduit à des configurations chaotiques (c'est à dire que toutes les conditions initiales conduisent à des motifs qui ne se répètent jamais). Par exemple : 30, 101, 150.

- Classe IV : Cette classe peut être considérée comme un mélange entre la classe 2 et la classe 3. On peut retrouver à l'intérieur d'un AC des motifs répétitifs et oscillants dont l'évolution conduit à des structures complexes localisées dont le temps de vie peut être long. Par exemple : 20, 110.



(a) Classe I : Règle 160 (b) Classe II : Règle 108



(c) Classe III : Règle 126 (d) Classe IV : Règle 110

FIGURE 1.10 – Exemples des diagrammes espace-temps de la classification de S. Wolfram.

3.2 Types d'automates cellulaires

Différents types d'AC peuvent être identifiés en modifiant les propriétés, la dimension, les conditions aux limites et la règle de transition des états de la cellule. Nandi et al. [51] les ont explorés afin de faciliter la conception et la modélisation des systèmes complexes. Ces types d'AC ont été introduits dans divers domaines d'application, particulièrement la cryptographie.

3.2.1 AC linéaire et non linéaire

Si toutes les règles associées aux cellules d'un AC sont linéaires alors l'AC est dit linéaire. Si l'expression logique de la règle ne fait intervenir que l'opérateur logique XOR, elle est alors appelée règle linéaire. Sinon, il s'agit d'une règle non linéaire. Cependant, un AC non linéaire utilise tous les opérateurs logiques disponibles dans la théorie des ACs.

3.2.2 AC complémentaire

Si toutes les règles associées aux cellules d'un AC sont complémentaires alors l'AC est dit complémentaire. Si la règle de l'AC ne comprend que l'opérateur logique XNOR alors elle est dite règle complémentaire.

3.2.3 AC additif

Un AC ayant des règles basées sur une combinaison des opérateurs XOR et XNOR est appelé AC additif. Il existe sept règles linéaires / additifs pour les ACs élémentaires dont les règles de numéros

sont : 60, 90, 102, 150, 170, 204 et 240 (à l'exclusion de la règle 0 qui est évidement linéaire / additif).

3.2.4 AC uniforme et non uniforme (ou hybride)

Un AC est dit uniforme si sa règle de transition est appliquée à toutes les cellules pendant son évolution. Sinon, il est non uniforme ou hybride : Dans ce cas, les cellules ne sont pas homogènes de par leur soumission à des règles de transition différentes de l'espace de l'automate cellulaire. Ce dernier type d'AC est considéré comme une variante linéaire/additive dans laquelle l'ensemble de règles peut être analysé par l'algèbre matricielle [14].

3.2.5 AC réversible

Dans un AC réversible, chaque configuration n'a qu'une seule configuration précédente de sorte que son processus d'évolution peut être retracé de manière unique. Autrement dit, si la règle globale Φ de l'AC est bijective, il est réversible.

3.2.6 AC programmable

L'AC programmable est une structure d'AC améliorée dans laquelle la logique combinatoire de chaque cellule n'est pas définie mais contrôlée par plusieurs signaux de contrôle. Un AC programmable implémente dynamiquement diverses fonctions selon différentes règles.

3.3 Caractérisation d'un automate cellulaire

L'étude du comportement d'un AC est déterminante pour caractériser et deviner son prochain comportement. Elle peut être réalisée en exécutant l'AC, ou par le biais d'une analyse mathématique [78]. Ainsi, certains outils de caractérisation ont été proposés pour analyser le comportement de certains ACs et découvrir leurs propriétés spécifiques. Par exemple le diagramme espace-temps, le diagramme de transition, le graphique de Bruijn et l'algèbre matricielle.

Dans cette section, nous introduisons l'outil de caractérisation le plus couramment utilisé dans la cryptographie pour les ACs élémentaires qui est l'algèbre matricielle.
L'algèbre matricielle est un outil qui caractérise les ACs linéaires ou additifs binaires $(\mathrm{S} = 0, 1)$ [13]. Les règles d'un AC de taille n sont alors caractérisées par une matrice caractéristique (eq. 1.6) de taille $n \times n$ sur $\{0, 1\}$, où la $i^{ème}$ ligne de la matrice exprime la dépendance de la $i^{ème}$ cellule avec ses voisines.

$$T[i,j] = \begin{cases} 1 & \text{si l'état suivant de la } i^{ème} \text{ cellule dépend de l'état actuel du } j^{ème} \text{ cellule} \\ 0 & \text{sinon} \end{cases} \tag{1.6}$$

Sachant que les ACs non uniformes utilisent des règles différentes pour différentes cellules, nous avons besoin d'un vecteur de règle pour spécifier la règle correspondante pour chaque cellule.

Prenons un AC élémentaire non uniforme de 4 cellules. Le vecteur de règle $\mathrm{R} =< 150, 150, 90, 150 >$ est représenté par les équations 1.8 et 1.9 où les conditions aux limites sont nulles. La matrice caractéristique 1.7 de l'AC est formée comme suit :

$$T = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{1.7}$$

$$90: \quad S_i^{t+1} = \quad S_{i-1}^t \oplus S_{i+1}^t \tag{1.8}$$

$$150: \quad S_i^{t+1} = \quad S_{i-1}^t \oplus S_i^t \oplus S_{i+1}^t \tag{1.9}$$

Ici, la règle de la première cellule de la première ligne est 150. Donc cette cellule ne dépend que des voisins droit et gauche et de la cellule actuelle (eq. 1.9). Puisque les conditions aux limites sont nulles, l'état du voisin gauche de la première cellule est 0. La configuration suivante de l'AC est obtenue en multipliant la matrice T par le vecteur de configuration courant, c'est-à-dire $S(t+1) = T \cdot S(t)$, où $S(t)$ est la configuration de l'AC à l'instant t.

Il est également à noter que la matrice T peut être utilisée pour caractériser efficacement, entre autres, le comportement de transition d'état, les propriétés de la réversibilité d'un AC linéaire. Voir [13] pour plus de détails.

4 Application des ACs à la cryptographie

Plusieurs recherches basées sur des automates cellulaires ont été proposées pour concevoir des algorithmes de chiffrement à clé symétrique. Essentiellement, Wolfram [76] a été le premier à appliquer l'AC en cryptographie en 1985 et ce en proposant un générateur de nombres pseudo-aléatoires où il a conçu un générateur de flux de clés utilisant un AC unidimensionnel uniforme de rayon 1 suivant la règle 30. Cependant, en 1991, Meier et Staffelbach [45] ont exploité les faiblesses de cette règle pour ensuite l'attaquer via une attaque de texte clair choisi, ce qui est venu contredire ce que Wolfram garantissait : La récupération de la graine de son générateur est un problème NP-complet. Depuis, de nombreuses primitives cryptographiques basées sur des automates cellulaires ont été proposées. De surcroît, Das et al. [15] ont présenté en 2010 un nouveau chiffrement par blocs basé sur des automates cellulaires programmables réversibles en exploitant leurs propriétés, à savoir : le parallélisme et la simplicité de l'implémentation. Quelques années plus tard, en 2014, Roy et al., dans leur article [55] ont proposé un chiffrement par blocs symétrique basé sur des automates cellulaires programmables. D'autre part, en 2012 [4], Bhaumik et al. ont présenté une technique pour construire des couches de diffusion de n'importe quelle longueur entre 16 bits et 128 bits, implémentées à l'aide d'un AC pour

développer un chiffrement par blocs SPN basé sur des codes séparables à distance maximale. Ils ont démontré la diffusion plus élevée produite par l'AC. Par suite, des automates cellulaires bi-dimensionnels avec une nouvelle structure de voisinage MARGOLUS ont été utilisés pour sélectionner un ensemble de règles réversibles dans [7]. Dans un autre travail, en 2014, Faraoun et al. ont présenté dans [21] un chiffrement par blocs non uniforme basé sur des automates cellulaires du second ordre en utilisant une stratégie d'algorithme génétique pour construire un espace optimal de règles élémentaires. En outre, dans l'article [43] publié en 2016, Mehta et al. ont présenté un nouvel algorithme de chiffrement par blocs utilisant des automates cellulaires réversibles non uniformes. Dans la même année, Bouchkaren .S et al. ont proposé dans [8] un algorithme de chiffrement par blocs rapide basé sur des automates cellulaires non uniformes réversibles et irréversibles. Quant à Li et al. [38], ces derniers ont introduit en 2017 un algorithme de chiffrement d'image basé sur un AC réversible de second ordre non uniforme équilibrant ainsi les règles de l'AC pour offrir un caractère aléatoire élevé. Ils ont également montré que leur algorithme de chiffrement peut résoudre le problème du mode de chiffrement ECB. En outre, dans [52] les auteurs ont proposé un nouveau système de chiffrement d'image basé sur des automates cellulaires non uniformes pour créer une image clé. Ils ont également utilisé des fonctions hyper-chaotiques pour sélectionner des nombres aléatoires à partir de l'image clé. Dans [11] les auteurs ont présenté un nouveau système de chiffrement d'image utilisant un système hyper-chaotique memristif (memristive hyperchaotic), les automates cellulaires ACs et les opérations de séquence d'ADN. La fonction de hachage SHA-256 est utilisée pour générer la clé secrète et calculer les valeurs initiales du système. Cette approche peut être appliquée aux images et aux champs de communication de vidéos sécurisées. Un autre modèle de chiffrement d'image est proposé une année plus tard en 2018 dans [75] basé sur un automate cellulaire partitionné à deux dimensions. Ce système introduit des opérations de substitution et de permutations répondant au critère global strict d'avalanche en au plus $M + N + 7$ tours de chiffrement pour une image $M \times N$. De plus, une méthode d'intégration d'algorithmes de chiffrement par blocs dans un nouvel algorithme est présentée en 2016 dans [61] afin de créer des variations et d'améliorer la sécurité de l'algorithme d'origine, ce qui a été expliqué concrètement sur AES-128. Cette intégration permet de créer une famille de chiffrements ayant un niveau de sécurité amélioré par rapport à un algorithme d'origine choisi et possédant la même longueur de clé avec la possibilité de l'étendre. Récemment en 2020 dans [56], Roy .S et al. ont proposé un chiffrement par blocs basé sur des automates cellulaires programmables compatibles avec l'IoT appelé IECA. Les résultats des tests NIST et Diehard ont montré que l'IECA génère un degré élevé d'aléatoire dans l'image chiffrée produite. En plus de cela, la corrélation, l'entropie et l'analyse différentielle du schéma proposé ont appuyé sa robustesse à différents types d'attaques tandis que les résultats expérimentaux ont montré l'efficacité de l'IECA par rapport aux chiffrements par blocs existants. En plus, Dennunzio A. et al. [16] ont fourni en 2021 de nombreux résultats de décidabilité et de caractérisation du comportement dynamique de l'AC additif sur des groupes abéliens finis et ont décrit comment ces résultats peuvent être exploités dans certaines applications emblématiques des cryptosystèmes tout en proposant des modifications importantes aux méthodes existantes afin de renforcer leur niveau de sécurité et de les

rendre plus résistants aux attaques difficiles.

En raison de leur propriétés et dynamisme, les automates cellulaires constituent un bon outil pour créer des S-boîtes cryptographiques puissantes. Plusieurs schémas des S-boîtes sont proposés utilisant des ACs. Nous citons dans ce sens la contribution de M. Szaban et al. [71] réalisée en 2010 dans laquelle ils ont décrit un générateur de tables S-boîte dynamique et flexible basé sur un AC élémentaire. Ils ont montré que les S-boîtes proposées présentent des propriétés cryptographiques de haute qualité. En outre, en 2016, dans [42] une méthode de synthèse de schémas efficaces des S-boîtes de haute qualité basées sur des ACs bi-dimensionnels répondant aux principaux critères cryptographiques ont été proposés. De plus, dans [24], une autre approche d'une S-boîte basée sur des ACs réversibles unidimensionnels programmables de deuxième ordre est utilisée dans l'algorithme AES afin de réduire la consommation d'énergie de réseaux sans fils.

Les ACs sont également utilisés pour améliorer les systèmes cryptographiques existants afin de résister aux attaques. En 2020 et dans [81], les auteurs ont combiné les ACs avec leur système de chiffrement de partition PCS précédemment développé pour éviter les attaques fréquentielles et statistiques. Ce qui s'ajoute aux propositions effectuées en 2017, dans [19], où les auteurs ont proposé une fonction de hachage à clé efficace et robuste basée sur un AC résistant aux attaques HMAC. Dans la même année et cette fois-ci dans le contexte de la cryptographie légère, une fonction de hachage légère (LCAHASH) basée sur un AC est proposée [28]. Les auteurs ont exploité la simplicité de l'AC pour implémenter leurs travaux dans la technologie RFID. En 2019 , dans [58], une extension de LCAHASH a été présentée et généralisée pour être utilisée dans l'IoT.

Les ACs ne sont pas exclusivement introduits dans les applications mentionnées ci-dessus, mais ils figurent également dans les systèmes d'authentification. Un exemple est le travail de Faraoun, K. et al. réalisé en 2014 [20] dans lequel ils ont utilisé les ACs réversibles pour construire un système de chiffrement d'authentification fort. Ils ont montré que le système proposé est beaucoup plus rapide que les autres chiffrements d'authentification standards car chaque bloc n'est exécuté qu'une seule fois pour assurer le chiffrement et l'intégrité de l'information. En outre, le travail [37] présente une nouvelle fonction de hachage inspirée de la construction éponge appelée CASH en utilisant un AC linéaire. D'autre part, [79] ont utilisé les ACs appliqués au cloud computing en 2017 afin d'assurer la confidentialité et l'intégrité des données stockées dans le cloud. Ce système proposé intègre le hachage homomorphe ce qui facilite la vérification des données sur les serveurs. Récemment en 2020, en vue d'exploiter les propriétés pseudo-aléatoires des automates cellulaires programmables (PCA) 90-150, les auteurs les ont utilisés dans [33] pour rendre ACORN v3, un schéma de chiffrement authentifié finaliste du concours CAESAR, robuste contre les attaques par faute différentielles.

Chapitre 2

BC-CaACO : Système de chiffrement par blocs utilisant les automates cellulaires et l'optimisation par colonies de fourmis

Sommaire

1 Introduction

Compte tenu de sa vulnérabilité par rapport aux entités non autorisées qui risquent de l'intercepter, la transmission des données privées via des canaux non sécurisés s'avère critique et de grande importance. Afin de remédier à cette menace d'envergure, nous avons porté un intérêt particulier au concept de chiffrement par blocs qui représente l'une des principales solutions cryptographiques pour assurer la confidentialité des données. Il s'agit d'un mécanisme dans lequel seules les entités autorisées peuvent comprendre les données transmises [47]. Ainsi, de nombreuses solutions de chiffrement ont été développées en utilisant différents modèles. Cependant, ces alternatives demeurent fragiles face aux attaques. Par conséquent, les chercheurs redirigent leur attention sur la sécurité de la conception des systèmes de résistance aux diverses attaques existantes.

Dans ce chapitre, nous proposons un nouveau système de chiffrement par blocs : A new block cipher system using cellular automata and ant colony optimization (BC-CaACO) [30] en évoluant des ACs unidimensionnels réversibles et irréversibles avec une boîte de substitution basée sur l'optimisation par colonies de fourmis (ACO) [3] pour renforcer la confusion. Si l'utilisation des S-boîtes et des clés secrètes s'aligne aux normes de sécurité, la sûreté du chiffrement par blocs est de fait vérifiée. La S-boîte basée sur l'ACO [3] est une S-boîte efficace qui fournit une non-linéarité élevée et satisfait d'autres caractéristiques cryptographiques spécifiques à ce genre de boites. De plus, par rapport aux systèmes de chiffrement par blocs existants, les propriétés dotées des ACs ont apporté une meilleure confusion et diffusion.

Dans la section 2, nous décrivons d'abord l'algorithme BC-CaACO pour ensuite le tester dans la section 3 à travers une analyse de sécurité effectuée par le biais d'un ensemble de tests et enfin présenter dans la section 4 les résultats issus du test de Diehard et proposer une comparaison de notre système avec d'autres algorithmes de chiffrement par blocs connus.

2 Description du système BC-CaACO

Le chiffrement BC-CaACO proposé, s'apparente au schéma de Feistel, dans lequel un AC non uniforme unidimensionnel est utilisé pour concevoir un algorithme de chiffrement par blocs associé à une S-boîte proposé dans [3]. Les S-boîtes sont cruciales dans la conception des algorithmes de chiffrement par blocs robustes. Mathématiquement, une nxm S-boîte est une fonction booléenne : $S : \{0,1\}^n \rightarrow \{0,1\}^m$, qui convertit une entrée de n bits en une sortie de m bits.
Afin d'apporter une bonne confusion à notre système, nous avons introduit une S-boîte basée sur l'ACO [3]. Cette dernière est une approche métaheuristique inspirée du comportement naturel des fourmis qui permet de concevoir une S-boîte efficace.

La S-boîte optimisée utlisée est illustrée dans 2.1 :

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

TABLEAU 2.1 – Boîte de substitution optimisée basée sur l'ACO intégrée dans BC-CaACO

[3] montre les performances de cette S-boîte ce qui nous a incité à l'intégrer dans notre nouveau modèle de chiffrement par blocs BC-CaACO et précisément l'algorithme de chiffrement et de déchiffrement qui utilise un ensemble de quatre règles réversibles et introduit une règle irréversible pour générer des sous-clés. Le processus de génération de clés, les systèmes de chiffrement et de déchiffrement seront décrits en détail ci-dessous.

2.1 Algorithme de génération des sous clés

Afin de générer un nombre aléatoire de sous clés de 128 bits en tant que graine, un AC uniforme avec la règle 110 et des conditions aux limites périodiques est utilisé. Le processus est décrit dans l'algorithme itératif 2.1. Cette règle a été choisie en raison de son comportement complexe et chaotique [60] et s'exprime en particulier par l'équation suivante :

$$A = \{\mathbb{S} = \{0, 1\}, D = 1, \nu = \{i - 1, i, i + 1\}\}$$

$$\text{AC-110} : (x_{-1}, x_0, x_1) \mapsto \bar{x}_{-1}x_0 + x_0\bar{x}_1 + \bar{x}_0x_1 \tag{2.1}$$

Où x_{-1} est la cellule voisine de gauche, x_1 la cellule voisine de droite et x_0 la cellule elle-même.
Dans l'algorithme 2.1, la fonction AC-110$(x^t) = x^{t+1}$ est la règle globale de l'automate, consistant à évoluer la sous-clé k_i en appliquant la règle 110 à toutes les cellules (voir eq. 2.1). L'algorithme fait d'abord évoluer la clé mère k_0 pour générer la première sous-clé k_1. Ensuite, il utilise la règle globale AC-110 pour obtenir chaque sous-clé k_i à partir de la sous-clé précédente k_{i-1}. Puis le processus est itéré jusqu'à la dernière sous-clé correspondant au dernier tour de l'algorithme de chiffrement. Enfin, on obtient l'ensemble de sous-clés $K = \{k_1, k_2, ..., k_n\}$ où n correspond au nombre de tours de BC-CaACO.

Algorithme 2.1 Génération des sous-clés de BC-CaACO

Entrée : k_0, n
Sortie : k
1: $k \leftarrow k_0$
2: **Pour** $i = 1$ à n **Faire**
3: $\quad k_i \leftarrow$ AC-110(k_{i-1})
4: **Fin Pour**

2.2 Processus de chiffrement

Le processus de chiffrement BC-CaACO comme décrit dans 2.2, commence par diviser le texte clair (M_p) en blocs de 256 bits, et si nécessaire, utilise une opération de remplissage. En vue d'obtenir un texte chiffré (M_c), une S-boîte [3] notée ACO-S-boîte, basée sur l'ACO, est appliquée à chaque bloc

(M_i). Subséquemment, chaque M_i est divisé en deux parties : la partie gauche (L_i) et la partie droite (R_i) de même taille : 128 bits. La partie droite R_i est XORée avec la clé k_i du tour correspondant. A posteriori, le résultat est évolué par quatre règles linéaires et réversibles, à savoir {60,90,150,102} (voir équation 2.2) avec des conditions aux limites nulles. En définitive, ces règles sont représentées par la matrice caractéristique T dans cet ordre.

$$T = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

La matrice caractéristique T est générée à l'aide de l'équation suivante [25] :

$$T[i,j] = \begin{cases} 1 & \text{si l'état suivant de la } i^{\text{ème}} \text{ cellule dépend de l'état actuel du } j^{\text{ème}} \text{ cellule} \\ 0 & \text{sinon} \end{cases}$$

A cette étape, la valeur de $(R_i \oplus k_i)$ est multipliée par la matrice caractéristique T.

$$\begin{array}{ll} \text{Règle'60'} : & x_{i-1} \oplus x_i \\ \text{Règle'90'} : & x_{i-1} \oplus x_{i+1} \\ \text{Règle'150'} : & x_{i-1} \oplus x_i \oplus x_{i+1} \\ \text{Règle'102'} : & x_i \oplus x_{i+1} \end{array}$$

Représentation des quatre règles utilisées en BC-CaACO

En outre, la partie gauche L_{i+1} reçoit à l'itération i+1 le résultat $T.(R_i \oplus k_i)$ et la valeur de L_{i+1} obtenue est XORée avec la valeur de (L_i). Le résultat sera la nouvelle valeur de la partie droite R_{i+1}. Ce processus est répété jusqu'au $n^{\text{ème}}$ tour. Finalement, les parties L_n et R_n sont concaténées pour former le texte chiffré (M_c). La figure 2.1 montre le schéma explicatif d'un seul tour du chiffrement BC-CaACO.

Algorithme 2.2 Algorithme de chiffrement de BC-CaACO

Entrée : M_p, k_0, n, T

Sortie : M_c

1: M_p en m blocs de 256 bits
2: **Si** $|M_i| \neq$ multiple de 256 bits **Alors**
3: Remplissage de M_i
4: **Fin Si**
5: K $\leftarrow$ Génération_sous_clés(k_0, n);
6: $M_i \leftarrow ACO\text{-}S\text{-}boîte(M_i)$;
7: **Pour** $i = 1$ à n **Faire**
8: $L_i \leftarrow Partie\_Gauche(M_i)$;
9: $R_i \leftarrow Partie\_Droite(M_i)$;
10: $L_{i+1} \leftarrow T.(R_i \oplus K_i)$;
11: $R_{i+1} \leftarrow L_{i+1} \oplus L_i$;
12: **Fin Pour**
13: $M_c \leftarrow L_i||R_i$;

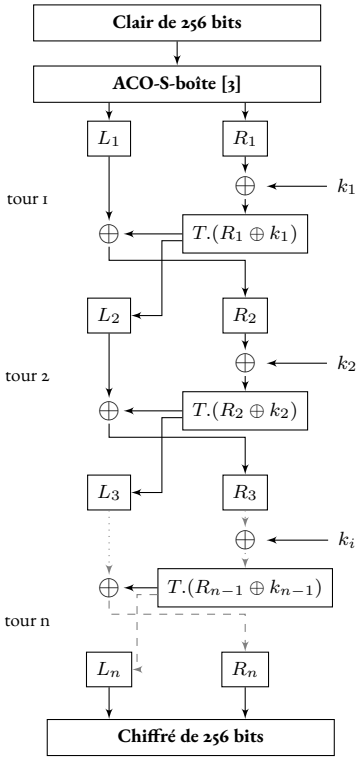


FIGURE 2.1 – BC-CaACO : Schéma de chiffrement d'un seul bloc

Complexité de l'algorithme

- Diviser le message M en n blocs et appliquer un remplissage nécessite au plus $s - \frac{L}{s}$ étapes, où s est la taille du bloc et L est la longueur du message.
- AC-110 nécessite $64 \times \frac{s}{2} \times r$, où r est le nombre de tours.
- Appliquer la ACO-S-boîte nécessite s étapes.
- Dans chaque tour i,
  - Diviser le message M en parties droite et gauche nécessite $2 \times \frac{s}{2}$ étapes.
  - XORer la partie droite R_i et $k[i]$ nécessite des ajouts $\frac{s}{2}$ modulo 2.
  - Multiplier la matrice caractéristique T par $R_i \oplus k[i]$ nécessite $32 \times 4^2 \times 64$ opérations.
  - XORer les deux parties gauche L_i et L_{i+1} nécessite des ajouts $\frac{s}{2}$ modulo 2.
- Enfin, Concaténer les deux parties, droite et gauche, nécessite s étapes.

Globalement, la complexité de notre algorithme est de $O\left(\mathrm{s}^2\right)$.

2.3 Processus de déchiffrement

Par définition, le processus de déchiffrement constitue l'algorithme de chiffrement exécuté dans l'ordre inverse (voir algorithme 2.3). Dans le processus de déchiffrement BC-CaACO que nous proposons, le texte chiffré représente l'entrée de l'algorithme. Ce procédé utilise l'ordre inverse des sous-clés, l'inverse de la S-boîte basée sur ACO [3], noté $ACO\text{-}S\text{-}boîte^{-1}$, de même que l'inverse de la matrice caractéristique T, notée T^{-1}. La figure 2.2, illustre le processus de déchiffrement concernant un seul bloc.

Algorithme 2.3 Algorithme de déchiffrement de BC-CaACO

Entrée : M_c, k_0, n, T^{-1}

Sortie : M_p

1: K ← Génération_sous_clés(k_0, n)
2: **Pour** $i = n$ à 0 **Faire**
3: $L_i \leftarrow Partie\_Gauche(M_i)$
4: $R_i \leftarrow Partie\_Droite(M_i)$
5: $L_i \leftarrow R_{i+1} \oplus L_{i+1}$
6: $R_i \leftarrow (T^{-1}.L_{i+1}) \oplus K_i$
7: **Fin Pour**
8: $M_c \leftarrow L_i || R_i$
9: $M_p \leftarrow ACO\text{-}S\text{-}boîte^{-1}(M_p)$

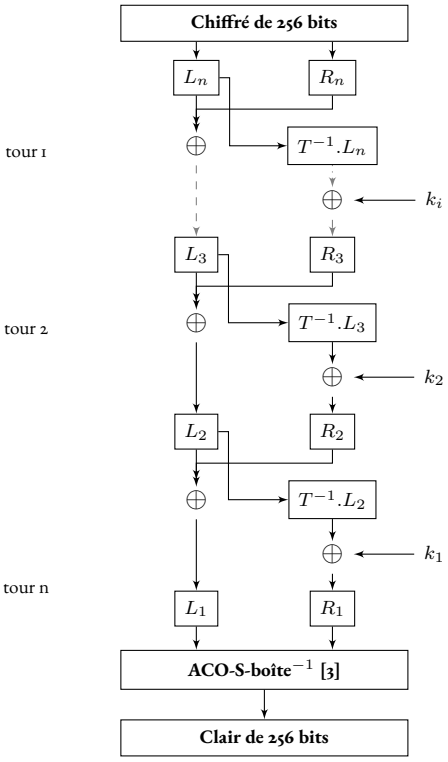


FIGURE 2.2 – BC-CaACO : Schéma de déchiffrement d'un seul bloc

3 Analyse de sécurité

Selon Claude Shannon, un algorithme de chiffrement par blocs irréprochable est celui qui satisfait deux propriétés essentielles : la diffusion et la confusion [62] (voir section 1.1). Pour vérifier ces propriétés, une expression mathématique décrite dans l'équation 2.2 est utilisée permettant ainsi le calcul du taux de changement d'un ou de plusieurs bits dans le message en clair ou dans la clé. Cette équation s'inspire du critère d'avalanche, qui est une propriété recherchée dans tout algorithme de chiffrement ce qui signifie qu'un petit changement dans le texte clair ou la clé produirait à son tour des modifications significatives dans le texte chiffré. Cet attribut mesure l'effet de la modification d'un ou de plusieurs bits de l'entrée sur la sortie [10].

Cet effet d'avalanche est mesuré par l'équation suivante :

$$Bits_{Diff} = (1/256) * \omega(C \oplus C')) * 100 \tag{2.2}$$

Où ω constitue le poids de Hamming, C étant le chiffré résultant du message d'origine et C' le chiffré provenant du message modifié.

3.1 Test de diffusion

Le test de sensibilité du changement de bits dans le texte clair d'un algorithme vérifie la propriété de diffusion qui produit un effet d'avalanche [10] entre le message en clair et le message chiffré. En te-

nant compte des paires de clairs et de clés secrètes générées, nous utilisons l'algorithme BC-CaACO en raison de produire le texte chiffré correspondant à chaque paire (texte clair, clé secrète). On reprend ce même procédé sur les textes claires en modifiant un ou plusieurs bits tout en gardant la clé inchangée. Suite à cela, nous avons recours à l'équation 2.2 en vue de calculer la valeur moyenne du pourcentage de la différence de bits.

Comme en témoigne la figure 2.3, plus de 50% des bits du texte chiffré ont changé. En particulier, la moyenne du pourcentage de différence de bits de notre système de chiffrement (BC-CaACO) est comprise entre 48,75% et 50,75% devançant ainsi l'AES-128 dont la moyenne du pourcentage de différence de bits demeure restreinte entre 47,1 % et 50,80 %. Ces pourcentages démontrent la bonne diffusion qu'apporte notre algorithme BC-CaACO par rapport à l'AES-128.

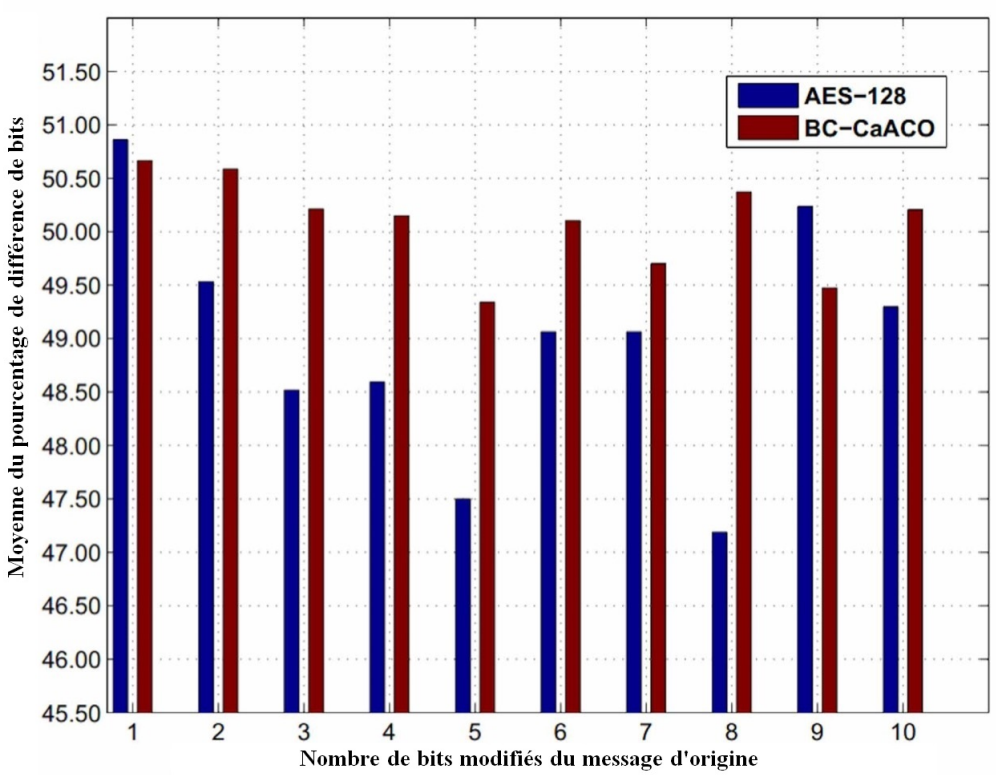


FIGURE 2.3 – BC-CaACO : Sensibilité du texte clair aux changements de bits

3.2 Test de confusion

La propriété de confusion établit une relation entre la clé, le texte clair et le texte chiffré qui elle-même est garantie par le test de sensibilité de la clé. Dans ce sens, nous considérons un ensemble de paires de textes clairs et de clés secrètes où chaque paire est chiffrée à l'aide de l'algorithme BC-CaACO proposé. Nous modifions par la suite un ou plusieurs bits dans les différentes clés générées aléatoirement, cette fois-ci en maintenant le text clair inchangé. Finalement, nous calculons la moyenne du pourcentage de différence de bits par le biais de l'équation 2.2.

La figure 2.4 indique que plus de 50% des bits du texte chiffré a changé. En détails, la moyenne du pourcentage de différence de bits de notre système de chiffrement BC-CaACO est comprise entre 48,75% et 52,23% et en ce qui concerne l'AES-128 elle s'étend entre 48,25% et 50,73%. Ces résultats obtenus prouvent que notre algorithme BC-CaACO répond pleinement au critère d'avalanche [10]. De plus, par rapport à l'AES-128, l'algorithme BC-CaACO offre une bonne confusion.

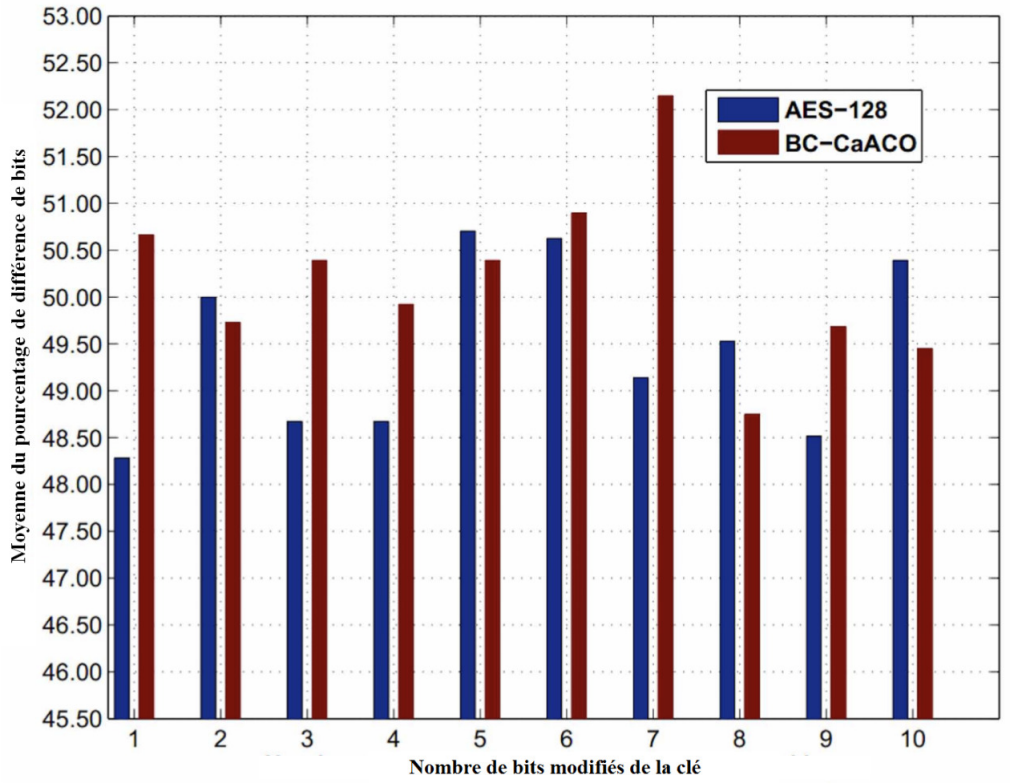


FIGURE 2.4 – BC-CaACO : Sensibilité de la clé aux changements de bits

Tout bien considéré, les résultats présentés témoignent de la grande sensibilité de la sortie BC-CaACO face aux simples changements de bits de n'importe quelle entrée ce qui confirme la résistance du système BC-CaACO proposé aux attaques différentielles et linéaires.

3.3 Attaque par force brute

L'attaque par force brute est un moyen permettant de trouver tous les arrangements de clés potentielles à l'aide d'un outil de prédiction rapide. Considérons une machine de haute qualité qui nécessite 10^{-10} secondes pour tester la validité de la clé. Notre algorithme BC-CaACO comprend 2^{128} clés possibles. Lors d'une tentative d'attaque par force brute, il faudrait environ $10^{-10} * 2^{128} = 3,40 * 10^{28}$ secondes (soit plus de $1,8 * 10^{34}$ ans) pour obtenir la bonne clé. Par conséquent, il est impossible d'effectuer une attaque par force brute avec recherche de clé exhaustive en un temps raisonnable.

4 Tests statistiques et performance

Dans cette section, nous présenterons la batterie de tests statistiques Diehard et les performances en vue d'évaluer notre chiffrement par blocs BC-CaACO. Pour cela, nous réalisons en premier une étude paramétrique du nombre de tours.

4.1 Paramétrage du nombre de tours

Le nombre de tours est primordial pour assurer la sécurité d'un système de chiffrement. Convoitant à atteindre la limite idéale tout en assurant une bonne confusion et diffusion, nous avons mené une étude paramétrique en utilisant le critère d'avalanche strict (voir l'équation 2.2) qui est une propriété nécessaire dans tout système cryptographique : En effet, un seul bit modifié sur l'entrée (texte clair ou clé) devrait produire une probabilité de 50% de changements dans la sortie (texte chiffré) [10].

Pour mener à bien ce test, nous avons testé l'effet du nombre de tours sur la moyenne d'avalanche à travers un ensemble hétéroclite de clés et de messages. S'ensuit plusieurs exécutions de notre algorithme dans le but ultime de décrocher le nombre optimal de tours en fonction de la moyenne d'avalanche. A partir de la figure 2.5, nous constatons que la valeur de tours convoitée est 10, un chiffre qui réalise à la sortie un bon effet d'avalanche.

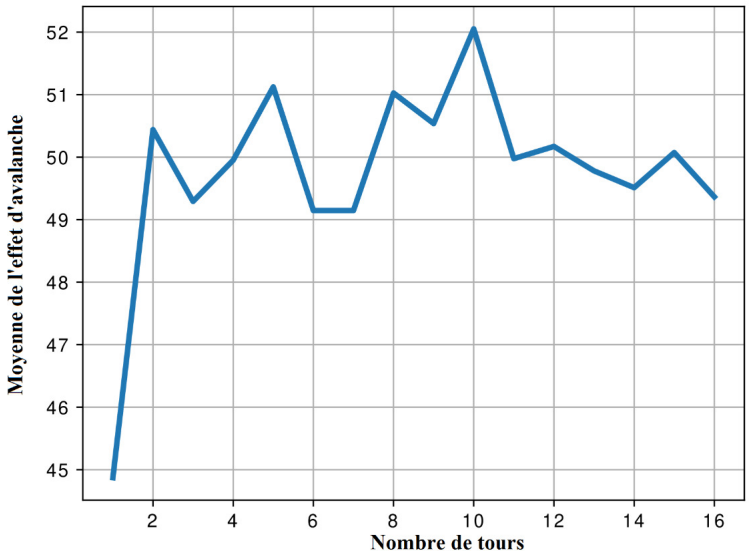


FIGURE 2.5 – BC-CaACO : Analyse du nombre de tours en fonction de l'effet d'avalanche

4.2 Test de Diehard

Dans cette section, nous utilisons Diehard [41] pour tester et analyser la qualité de l'aléatoire du chiffrement par blocs proposé. L'objectif principal de ce test est de contourner les attaques statistiques contre notre algorithme BC-CaACO. En particulier, une sortie du chiffrement par blocs sécurisé de-

vrait être statistiquement indiscernable d'une sortie aléatoire via la fonction de chiffrement. Pour effectuer ce test, nous convertissons d'abord une séquence de chiffrés générée de manière aléatoire en un format binaire en but de générer un flux de bits supérieur à 10 Mo. De là et en utilisant la batterie de tests Diehard [41] nous menons une analyse statistique sur ce flux. La valeur p (p-value) des chiffrés aléatoires obtenue est bien vérifiée par les tests de Diehard et apparaît dans l'intervalle [0.025, 0.975]. La moyenne des résultats est résumée dans le tableau 2.2. Sur cette base, le flux binaire généré à l'aide de notre proposition vérifie l'ensemble des tests DIEHARD. Ainsi, notre chiffrement par blocs BC-CaACO offre un bon comportement aléatoire au point d'être statistiquement impossible à distinguer d'une fonction aléatoire.

Nom de test	valeur de p	Interprétation
diehard_birthdays	0.92351383	PASSED
diehard_operm5	0.67202970	PASSED
diehard_rank_32x32	0.02714417	PASSED
diehard_rank_6x8	0.88412479	PASSED
diehard_bitstream	0.74214657	PASSED
diehard_opso	0.85347761	PASSED
diehard_oqso	0.41263403	PASSED
diehard_dna	0.40319163	PASSED
diehard_count_1s_str	0.38192448	PASSED
diehard_count_1s_byt	0.79099013	PASSED
diehard_parking_lot	0.68281092	PASSED
diehard_2dsphere	0.96183572	PASSED
diehard_3dsphere	0.79423482	PASSED
diehard_squeeze	0.54179745	PASSED
diehard_sums	0.24566165	PASSED
diehard_runs	0.37524660	PASSED
diehard_craps	0.25955910	PASSED
marsaglia_tsang_cd	0.92455244	PASSED
sts_monobit	0.82312061	PASSED
sts_runs	0.90123823	PASSED
sts_serial	0.61312798	PASSED
rgb_bitdist	0.52796448	PASSED
rgb_minimum_distance	0.72448457	PASSED
rgb_permutations	0.76952369	PASSED
rgb_lagged_sum	0.67102237	PASSED
dab_byte_distrib	0.03934358	PASSED
dab_dct	0.55245199	PASSED
dab_filltree	0.42950182	PASSED
dab_filltree2	0.80567738	PASSED
dab_monobit2	0.16948857	PASSED

Tableau 2.2 – Les tests de Diehard pour BC-CaACO

4.3 Test de performance et comparaison

Le tableau 2.3 exprime le test de performance du temps CPU en comparant notre algorithme BC-CaACO avec certains systèmes de chiffrement par blocs existants (spécialement, triple DES et AES [66]). L'algorithme BC-CaACO a été exécuté sur un processeur Intel Core i5-34227, OS 64 bits, 1,8 GHz avec 4 Go de RAM. Comparé à d'autres systèmes de chiffrement standards, l'algorithme BC-CaACO ainsi conçu présente de fortes chances d'atteindre une très bonne vitesse. En outre, si nous utilisons une implémentation matérielle il peut atteindre une meilleure vitesse.

Taille de message (Kilo byte)	AES(ms)	3DES(ms)	BC-CaACO (ms)
1	82.69	86.49	48.64
10	951.2	614.9	555.6
20	1972	1096	1029

TABLEAU 2.3 – Comparaison du temps de chiffrement entre notre système BC-CaACO avec d'autres systèmes de chiffrement par blocs connus

5 Conclusion

Ce chapitre présente un nouveau chiffrement par blocs utilisant des ACs unidimensionnels réversibles et irréversibles et une S-boîte basée sur l'ACO en vue d'apporter plus de confusion. Divers tests ont été menés dans ce sens pour attester la sécurité de notre algorithme BC-CaACO dont on peut citer l'évalutaion de l'effet d'avalanche, les tests statistiques et les performances. L'ensemble de ces tests a confirmé une robustesse élevée de notre nouvelle conception par rapport aux schémas de chiffrement par blocs bien connus.

Bien que BC-CaACO offre un niveau de sécurité plus élevé après 10 tours, ce système de chiffrement nécessite des blocs de 256 bits ce qui va à l'encontre de la cryptographie légère. Par conséquent, il devient nécessaire de concevoir un système de chiffrement par blocs de taille plus réduite tout en exploitant les avantages des schémas de chiffrement classiques (particulièrement, le schéma de Feistel et SPN) ainsi que ceux des automates cellulaires.

Chapitre 3

SPF-CA : Système de chiffrement par blocs basé sur des automates cellulaires utilisant des S-boîtes dépendantes des sous-clés

Sommaire

1 Introduction

Tout au centre de notre intérêt se trouve le dévleppemment de cryptosystèmes robustes, assurant la confidentialité des données, concurrençant les systèmes les plus connus et intégrant les technologies les plus récentes reposant sur la cryptographie légère. Ainsi intervient notre motivation de concevoir un système de chiffrement par blocs de taille plus réduite intégrant les schémas de chiffrement classiques (particulièrement, le schéma de Feistel et SPN) et les automates cellulaires.

Dans ce chapitre, nous proposons un nouveau chiffrement par blocs hybride qui implique explicitement les deux schémas : Réseau de permutation-substitution (SPN) et le schéma de Feistel, basé sur des automates cellulaires, nommé A new cellular automata based block cipher using key-dependent

S-boxes (SPF-CA) [2]. Le chiffrement SPF-CA consiste à chiffrer chaque bloc du texte clair suivant un tour de SPN comprenant ses trois couches combinées à un tour du schéma de Feistel. Nous présentons par la suite une boîte de substitution dynamique (SD-boîte) dépendante de la clé inspirée de [39].Cette SD-boîte utilise un automate cellulaire avec une règle de la classe IV (voir 3.1.4) qui est évoluée pour augmenter l'effet de confusion et fournir une maximale non-linéarité dans l'algorithme SPF-CA proposé. Dans l'optique d'augmenter l'effet de diffusion, nous proposons une P-boîte efficace. De plus, la fonction de Feistel est une évolution d'automate cellulaire utilisant la règle 30 introduite par Wolfram et est largement utilisée en cryptographie [77]. En outre, de nombreuses sous-clés sont générées à partir d'une clé secrète en utilisant Design of new pseudo-random number generator based on non-uniform cellular automata (PSOCA), un algorithme proposé par Hanin et al. [29], dans le but d'apporter plus d'aléatoire et de robustesse contre les attaques par clé. En conséquence, notre conception proposée dépassse l'algorithme de chiffrement par blocs bien connu, à savoir AES-192, en terme d'efficacité et de résistance contre les attaques en texte clair connus/choisis et les attaques de récupération de clé.

Le système SPF-CA atteint également les propriétés de confusion et de diffusion élevées en utilisant un minimum nombre de tours. De plus, l'analyse de sécurité et les résultats expérimentaux témoignent de la haute résistance de notre chiffrement par blocs SPF-CA proposé et ce face aux attaques existantes telles que la cryptanalyse différentielle, linéaire mais aussi les attaques statistiques.

Nous présentons dans la section 2 le système de chiffrement SPF-CA pour plus tard l'évaluer dans la section 3 par le biais d'une analyse de sécurité assurée par un ensemble de tests et finalement présenter les résultats des tests de Diehard et proposer une comparaison de notre système avec d'autres algorithmes de chiffrement par blocs connus dans la section 4.

2 Description du système SPF-CA

En vue, de bénéficier au mieux des schémas SPN et Feistel, nous proposons un nouvel algorithme hybride SPN-Feistel basé sur des automates cellulaires uniformes, nommé SPF-CA. Dans ce qui suit, nous décrivons notre système SPF-CA proposé en utilisant des S-boîtes dynamiques dépendantes des clés.

SPF-CA chiffre chaque bloc du texte clair suivant SPN. Communément, un schéma SPN nécessite une fonction du tour inversible pour assurer le déchiffrement. Cela comprend trois couches : une couche d'addition de clé ainsi que des couches de substitution et de permutation. Plus précisément, ces deux dernières couches assurent respectivement les propriétés de confusion et de diffusion ce qui impose une conception soignée de la S-boîte qui est élaborée afin de développer des systèmes de chiffrement sécurisés. Plusieurs cryptosystèmes bien connus utilisent une S-boîte statique ce qui risque de la rendre parfois vulnérable. Par conséquent, une couche de substitution dynamique est introduite dans le but d'augmenter l'efficacité de SPF-CA. En sus, il est question d'une structure itérative comprenant la division du texte clair en deux parties égales et utilisant la fonction du tour pour mettre à jour l'une de ces deux parties. Dans notre exemple, nous avons utilisé la règle 30 de l'AC appliquée à

la partie droite de Feistel. En outre, l'algorithme SPF-CA prend en entrée un texte clair de 128 bits et une clé de 192 bits produite par le générateur de nombres pseudo-aléatoires PSOCA [29].

2.1 Algorithme de génération des clés

Afin de générer les sous-clés $k = \{k_1, k_2, \ldots, k_r\}$, nous utilisons le générateur de nombres pseudo-aléatoires : PSOCA proposé par hanin et al. dans [29] dans lequel ils ont proposé une nouvelle approche comprenant la combinaison de l'algorithme d'optimisation par essaim particulaire binaire modifié (Modified binary particle swarm optimization (MBPSO)) avec les ACs pour produire des séquences de bits aléatoires.

L'algorithme PSOCA prend trois paramètres en entrée : une graine s, une longueur de clé l et le nombre de sous-clés r dans le but de générer un ensemble de clés aléatoires. Par conséquent, l'algorithme PSOCA prend une longueur arbitraire de bits comme graine et fournit des sous-clés de 192 bits correspondant à la clé du tour lors le processus de chiffrement [29]. À chaque tour de chiffrement, nous prenons les 128 premiers bits de $k_i (i \in [1, r])$ associés à l'étape SPN et les 64 derniers bits associés à l'étape Feistel.

2.2 Processus de chiffrement

Le processus de chiffrement divise d'abord le texte clair M en blocs de 128 bits et effectue par la suite un remplissage si nécessaire. Nous prenons le texte clair $M = \{m_1, m_2, \ldots, m_n\}$ et le texte chiffré $C = \{c_1, c_2, \ldots, c_n\}$ où n est le nombre des blocs du message. Le chiffrement SPF-CA proposé utilise deux clés différentes : Une pour le schéma SPN, écrite sous la forme $KS = \{ks_1, ks_2, \ldots, ks_r\}$ de taille 128 bits chacune ainsi qu'une autre exprimée comme suit : $KF = \{kf_1, kf_2, \ldots, kf_r\}$ correspondant au schéma de Feistel de longueur 64 bits chacune, où r est le nombre de tours de notre système.

Afin d'obtenir le texte chiffré C, l'algorithme SPF-CA est principalement composé des opérations : SD-boîte, P-boîte et AC-30 présentées ci-après.

2.2.1 Transformations de substitution

La sécurité d'un algorithme de chiffrement dépend notamment de l'efficacité de la S-boîte. Dans ce sens, nous proposons SD-boîte qui est une S-boîte dynamique inspirée de [39] et adaptée à notre système SPF-CA.

La SD-boîte est une transformation affine liée à l'opération inverse modulaire d'une valeur d'un octet qui peut être calculée par l'algorithme d'Euclide étendu. En addition à cela, on applique un modulo 256 pour garantir que le résultat soit toujours un octet. Cette SD-boîte est calculée selon la for-

mule 3.1 suivante :

$$DS\text{-}box(m_i) = \begin{cases} (a(m_i^{-1} \mod 257) + b) \mod 256, & \text{si } m_i \neq 0 \\ 0, \qquad sinon \end{cases} \tag{3.1}$$

Où :

- m_i : Un octet du message en clair M.
- a : 7 bits extraits du deuxième octet de la clé du tour correspondant, concaténés avec 1 sur le côté droit afin de trouver l'inverse modulo 256.
- b : 8 bits extraits du premier octet de la clé du tour correspondant.

Le couple (a, b) ainsi obtenu est évolué par la règle chaotique 110 suivant un automate cellulaire unidimensionnel uniforme avec condition aux limites nulles. Ergo, la SD-boîte proposée vise à augmenter autant que possible la confusion dans notre système et également à apporter une sécurité efficace.

Considérons invSD-boîte() comme étant la transformation inverse de SD-boîte().

2.2.2 Transformations de permutation

P-boîte vise à assurer la propriété de diffusion. Il s'agit de déplacer chaque bit vers un autre octet qui ne l'inclut pas et ceci en suivant le schéma illustré dans le tableau 3.1.

invP-boîte étant la transformation inverse de la table P-boîte.

77	6	63	120	49	106	35	92	21	78	7	64	121	50	107	36
93	22	79	8	65	122	51	108	37	94	23	80	9	66	123	52
109	38	95	24	81	10	67	124	53	110	39	96	25	82	11	68
125	54	111	40	97	26	83	12	69	126	55	112	41	98	27	84
13	70	127	56	113	42	99	28	85	14	71	0	57	114	43	100
29	86	15	72	1	58	115	44	101	30	87	16	73	2	59	116
45	102	31	88	17	74	3	60	117	46	103	32	89	18	75	4
61	118	47	104	33	90	19	76	5	62	119	48	105	34	91	20

TABLEAU 3.1 – P-boîte utilisée dans SPF-CA

2.2.3 Fonction du tour de Feistel : AC-30

La structure du schéma de Feistel assure l'inverse même si la fonction utilisée dans chaque tour n'est pas inversible. De ce fait, on peut faire évoluer un automate cellulaire avec une règle qui n'est pas forcément réversible. Pour cela, notre fonction de Feistel AC-30 est un automate cellulaire utilisant la règle 30 avec condition aux limites périodique. En particulier, la partie droite obtenue est XORée avec une valeur de sous-clé et le résultat est considéré ainsi comme l'entrée de la fonction AC-30.

L'algorithme 3.1 décrit la processus de chiffrement de SPF-CA.

Algorithme 3.1 Processus de chiffrement de SPF-CA

Entrée : M : texte clair, K : clé

Sortie : C : texte chiffré

1: Diviser M en n blocs de 128 bits; $M = \{m_1, m_2, ..., m_n\}$
2: **Si** $|m_n| < 128$ **Alors**
3: Remplissage de m_n
4: **Fin Si**
5: $K \leftarrow PSOCA(s, 192, r)$ ▷ Génération de r sous-clés
6: $KS \leftarrow$ Extraire les 128 premiers bits de K
7: $KF \leftarrow$ Extraire les 64 derniers bits de K
8: **Pour** $i = 1$ à $r - 1$ **Faire**
9: $C \leftarrow M \oplus ks[i]$
10: $C \leftarrow SD\text{-}boîte(C)$
11: $C \leftarrow P\text{-}boîte(C)$
12: $R(i+1) \leftarrow L(i) \oplus$ AC-30$(R(i) \oplus kf[i])$ ▷ R : la partie droite et$L$: la partie gauche
13: $L(i+1) \leftarrow R(i)$
14: $C \leftarrow L||R$
15: **Fin Pour**
16: $C \leftarrow SD\text{-}boîte(C)$
17: $C \leftarrow C \oplus ks[r]$

La figure 3.1 montre le schéma explicatif d'un seul tour du chiffrement SPF-CA.

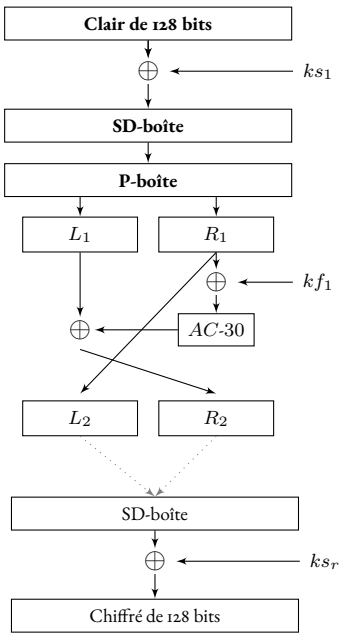


FIGURE 3.1 – SPF-CA : Schéma de chiffrement d'un seul bloc

Complexité de l'algorithme

- Diviser le message M en n blocs et appliquer un remplissage nécessite au plus $s - \frac{L}{s}$ étapes, où s est la taille d'un bloc et L est la longueur du message.
- PSOCA nécessite $96 \times 192 \times r$, où r est le nombre de tours.
- Extraction des s premiers bits de K nécessite s et l'extraction des $\frac{s}{2}$ derniers bits de K nécessite $\frac{s}{2}$ étapes.
- Dans chaque tour,
  - XORer M et $\mathrm{ks[i]}$ nécessite des ajouts s modulo 2.
  - Appliquer lla SD-boîte nécessite s modulo 2 divisions.
  - Appliquer la P-boîte nécessite s étapes.
  - Diviser le message M en parties droite et gauche nécessite $2 \times \frac{s}{2}$ étapes.
  - Appliquer l'AC-30 nécessite $\frac{s}{2} \times 32$ étapes.
  - XORer l'AC-30 et $\mathrm{kf[i]}$ nécessite des ajouts $\frac{s}{2}$ modulo 2.
  - XORer $\mathrm{L_i}$ et l'AC-30 $\oplus$ $\mathrm{kf[i]}$ nécessite des ajouts $\frac{s}{2}$ modulo 2.
  - Concaténer les deux parties, droite et gauche, nécessite s étapes.
- Appliquer la SD-boîte nécessite s modulo 2 divisions.
- Enfin, XORer C et $\mathrm{ks[r]}$ nécessite des ajouts s modulo 2.

Globalement, la complexité de notre algorithme est de $O\ (\mathrm{s}^2)$.

2.3 Processus de déchiffrement

Pour le processus de déchiffrement, comme décrit dans l'algorithme 3.2, le texte chiffré est considéré comme entrée de l'algorithme de chiffrement en utilisant l'ordre inverse des sous-clés et en appliquant les transformations invP-boîte() et invSD-boîte().

Algorithme 3.2 Processus de déchiffrement de SPF-CA

Entrée : C : texte chiffré, K : clé

Sortie : M : texte clair

1: $K \leftarrow PSOCA(s, 192, r)$ ▷ Génération de r sous-clés
2: $KS \leftarrow$ Extraire les 128 premiers bits de K
3: $KF \leftarrow$ Extraire les 64 derniers bits de K
4: $M \leftarrow C \oplus ks[r]$
5: $M \leftarrow invSD\text{-}boîte(M)$
6: **Pour** $i = r - 1$ to 1 **Faire**
7: $R(i) \leftarrow L(i+1)$ ▷ R : la partie droite et$L$: la partie gauche
8: $L(i) \leftarrow R(i+1) \oplus$ AC-30$(L(i+1) \oplus kf[i])$
9: $M \leftarrow L \| R$
10: $M \leftarrow invP\text{-}boîte(M)$
11: $M \leftarrow invSD\text{-}boîte(M)$
12: $M \leftarrow M \oplus ks[i]$
13: **Fin Pour**

3 Analyse de sécurité

Le fait qu'une petite modification de l'entrée (texte clair ou clé) devrait générer des changements significatifs dans la sortie (texte chiffré) rend l'effet d'avalanche une propriété déterminante dans tout système cryptographique [10]. En conséquence, cette propriété est mesurée par l'équation :

$$\forall x, y \in \{0,1\}^m H(x,y) = 1 \Rightarrow \text{ Average } (H(F(x), F(y))) = n/2 \tag{3.2}$$

Où H est la distance de Hamming.

Une multitude de tests a été menée pour évaluer l'efficacité de notre algorithme SPF-CA proposé. Dans ce qui suit, les résultats obtenus sont présentés.

3.1 Test de diffusion

La propriété de diffusion vise à produire un effet d'avalanche entre le texte clair et le texte chiffré. Le test de sensibilité du changement de bit du texte clair est employé pour vérifier la propriété de diffusion d'un algorithme donné. Pour cela, nous générons un ensemble de paires de clés secrètes et de textes clairs, nous utilisons ultérieurement notre algorithme proposé SPF-CA pour chiffrer chaque paire. Suite à cela, un seul bit est modifié dans les différents textes clairs générés aléatoirement tout en gardant la clé inchangée. En dernier, nous calculons la moyenne de distance de Hamming en appliquant l'équation 3.2.

Comme en témoigne la figure 3.2, les valeurs presque obtenues sont proches de la moitié de la

taille du bloc en tant que valeur idéale d'effet d'avalanche. De ce fait, la différence moyenne de bits est comprise entre $45,46\%$ et $54,21\%$ pour notre algorithme SPF-CA et entre $46,79\%$ et $54,14\%$ en ce qui concerne AES-192 chose qui atteste la bonne diffusion qu'apporte notre système de chiffrement par blocs suggéré.

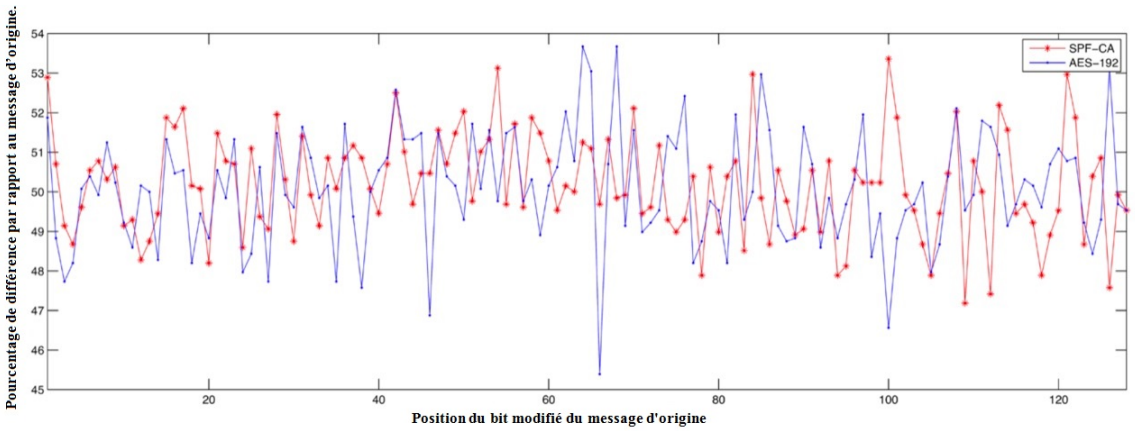


FIGURE 3.2 – SPF-CA : Sensibilité du message au changement d'un bit

De la même manière, un certain nombre de bits est modifiés sur chaque entrée du message en gardant la clé inchangée afin de mettre en évidence les modifications de sortie issue de chacun des chiffrements SPF-CA et AES-192. La figure 3.3 représente le résultat de ce test. In fine, notre algorithme atteint un bon pourcentage de différence de sortie comparé à AES-192, offrant ainsi une bonne diffusion.

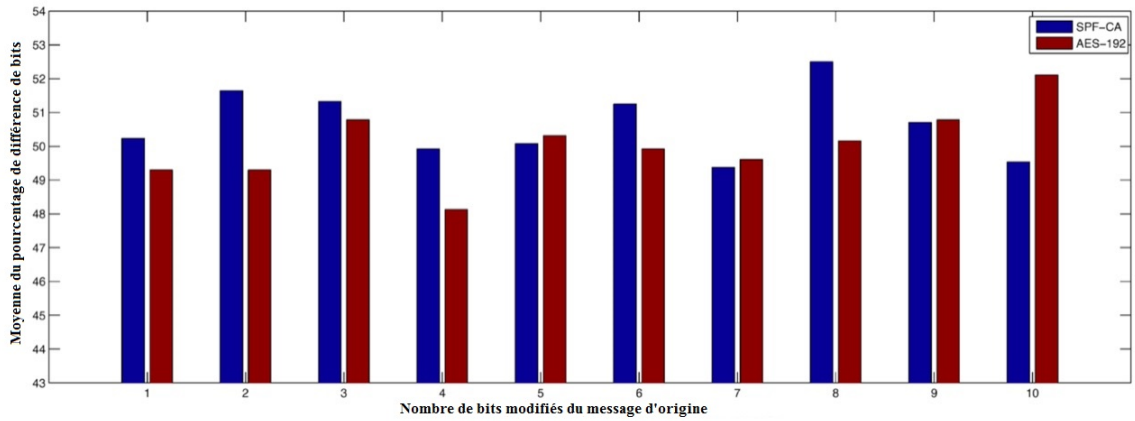


FIGURE 3.3 – SPF-CA : Sensibilité par rapport au message d'origine

3.2 Test de confusion

La propriété de confusion crée une relation entre la clé, le texte clair et le texte chiffré. Le test de sensibilité de clé de son côté évalue la propriété de confusion d'un algorithme de chiffrement. A cette fin, nous considérons un ensemble de paires de clés secrètes et de textes clair, nous utilisons par la suite

notre algorithme proposé SPF-CA pour chiffrer chaque paire. A posteriori, un seul bit est modifié dans les différentes clés générées aléatoirement tout en conservant le texte clair inchangé. Au final, nous calculons la moyenne du pourcentage de différence de bits en appliquant l'équation 3.2.

La figure 3.4 montre que plus de 50% des bits du texte chiffré a été modifié. Effectivement, le pourcentage moyen de distance de Hamming se situe entre $47,18\%$ et $53,35\%$ pour notre système de chiffrement proposé SPF-CA et demeure entre $45,39\%$ et $53,67\%$ pour l'AES-192. Partant de ces résultats, on note la haute sensibilité de notre suggestion face aux changements de clés d'un bit.

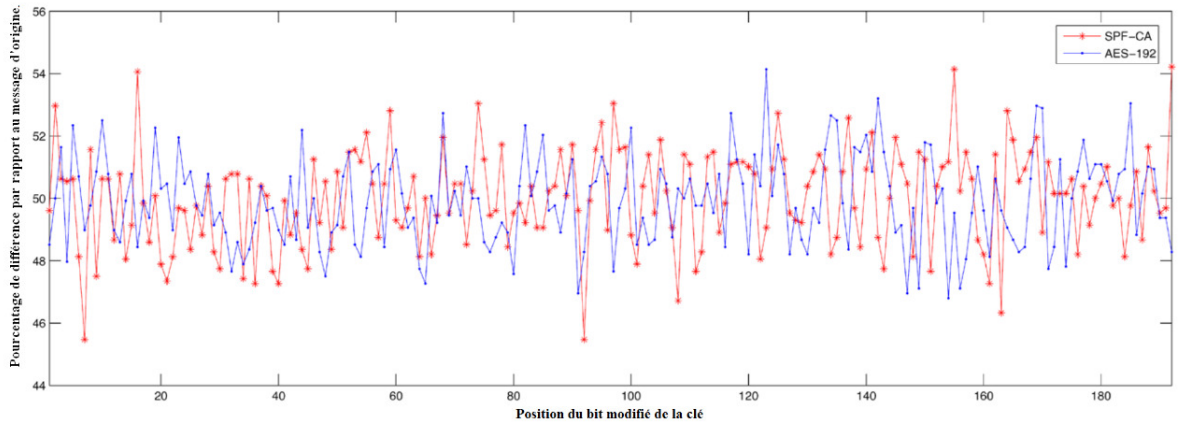


FIGURE 3.4 – SPF-CA : Sensibilité de la clé au changement d'un bit

Au delà, un certain nombre de bits est modifié sur chaque entrée de la clé de chiffrement tout en gardant le message inchangé et ce dans l'optique de montrer le changement de sortie issue de chacun des chiffrements SPF-CA et AES-192. La figure 3.5 illustre le résultat de ce test. En toute évidence, notre algorithme est en concurrence avec AES-192. On en conclut que notre chiffrement SPF-CA est très sensible aux variations de bits et qu'il apporte une bonne confusion.

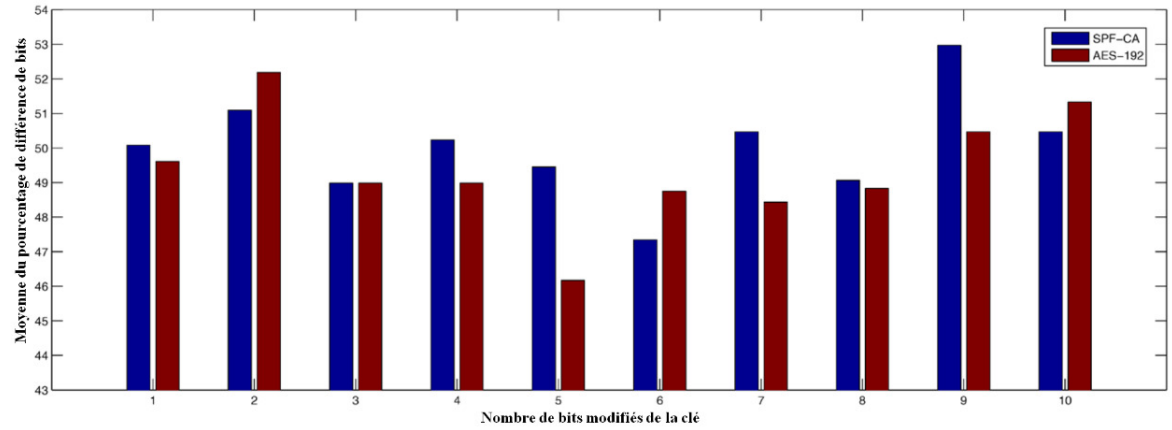


FIGURE 3.5 – SPF-CA : Sensibilité par rapport à la clé d'origine

3.3 Analyse de la résistance aux attaques différentielles et linéaires

Une essentielle caractéristique de sécurité des systèmes de chiffrement par blocs est leur résistance à la cryptanalyse linéaire et différentielle. Si le texte chiffré est très sensible aux petites modifications de la clé secrète et s'il utilise un nombre exhaustif de S-boîte actives, ces attaques ne peuvent être satisfaites.

Pour le cas de notre algorithme SPF-CA, ce dernier présente une sensibilité face à un ou plusieurs bits modifiés par rapport à la clé d'origine (voir 3.2) ou au texte clair d'origine (voir 3.1). De plus, notre algorithme SPF-CA utilise des S-boîtes dynamiques qui renforcent sa sécurité et fournissent une infinité de S-boîte actives : Un résultat bien vérifié à travers un calcul successif du nombre de S-boîte actives. Nous pouvons ainsi conclure que notre SPF-CA résiste aux attaques différentielles et linéaires.

3.4 Attaque par force brute

L'attaque par force brute est une tentative de trouver tous les arrangements de clés potentielles à l'aide d'un outil de prédiction rapide. Si la précision de l'ordinateur est de 10^{-13} secondes pour tester la validité de la clé. Notre algorithme SPF-CA nécessite 2^{192} d'espace de clés possibles. Ensuite, la machine sélectionnée met $10^{-13} * 2^{192} = 6.27 * 10^{44}$ secondes (soit plus de $1.9 * 10^{37}$ ans) pour trouver la clé secrète valide. En fin de compte, l'espace clé est suffisamment grand pour résister à l'attaque par force brute.

4 Tests statistiques et performance

4.1 Paramétrage du nombre de tours

Le nombre de tours est capital dans la sécurité d'un système de chiffrement. Afin d'atteindre la limite idéale qui assure la confusion et la diffusion, le nombre de tours est calculé à l'aide d'une étude paramétrique relative au critère d'avalanche (voir équation 3.2).

Nous avons utilisé plusieurs exécutions de l'algorithme SPF-CA afin d'obtenir le meilleur nombre de tours en fonction de la valeur moyenne de l'effet d'avalanche. Le tableau 3.2 illustre que la valeur optimale est atteinte en 9 tours où la valeur moyenne de l'effet d'avalanche est 64.

Nombre de tours	Moyenne de l'effet d'avalanche
2	63.9664
3	64.1242
5	64.0734
9	64.0102
10	64.0805
11	63.9555
14	63.7656

TABLEAU 3.2 – Nombre de tours selon la valeur de l'effet d'avalanche

4.2 Tests de Diehard

Dans cette section, nous utilisons la batterie de tests Diehard [41] pour analyser le caractère aléatoire du chiffrement par blocs proposé. L'objectif principal de ces tests est d'évaluer la robustesse de notre système proposé pour éviter les attaques statistiques. Une sortie de chiffrement par blocs sécurisé doit être statistiquement indiscernable d'une sortie aléatoire. Pour effectuer ces tests, une séquence de textes chiffrés générés aléatoirement est convertie dans la forme binaire correspondante, représentant un flux de bits supérieur à 30 MB. Ensuite, ce flux est analysé statistiquement en utilisant les tests de Diehard [41] qui vérifie la valeur p (p-value) des nombres aléatoires obtenus : La valeur p doit être comprise dans l'intervalle [0.025, 0.975]. Comme il apparaît dans le tableau 3.3 le flux binaire généré à l'aide de notre chiffrement par blocs vérifie tous les tests de Diehard. Somme toute, le comportement de notre chiffrement par blocs SPF-CA est indiscernable d'une fonction aléatoire.

Nom de test	valeur de p	Interprétation
diehard_birthdays	0.65962310	PASSED
diehard_operm5	0.46712366	PASSED
diehard_rank_32x32	0.62100955	PASSED
diehard_rank_6x8	0.40950307	PASSED
diehard_bitstream	0.26518472	PASSED
diehard_opso	0.36103301	PASSED
diehard_oqso	0.62610282	PASSED
diehard_dna	0.12835196	PASSED
diehard_count_1s_str	0.92104869	PASSED
diehard_count_1s_byt	0.22211509	PASSED
diehard_parking_lot	0.10511838	PASSED
diehard_2dsphere	0.74652201	PASSED
diehard_3dsphere	0.93330916	PASSED
diehard_squeeze	0.14621419	PASSED
diehard_runs	0.62591594	PASSED
diehard_craps	0.08837297	PASSED
marsaglia_tsang_gcd	0.34797406	PASSED
sts_monobit	0.50881096	PASSED
sts_runs	0.94200465	PASSED
sts_serial	0.62368357	PASSED
rgb_bitdist	0.60768440	PASSED
rgb_minimum_distance	0.57154599	PASSED
rgb_permutations	0.46398261	PASSED
dab_monobit2	0.63215899	PASSED

Tableau 3.3 – Tests de la batterie de Diehard

4.3 Test de performance et comparaison

Le tableau 3.4 représente le test de performance de notre algorithme SPF-CA, comparé à d'autres chiffrements par blocs bien connus, à savoir triple DES et AES [66]. Les tests ont été implémentés en C++ sur un Intel Core i534227,64 -bit OS, 1.8GHz avec 4 GB de RAM .Comparé à d'autres standards, notre algorithme est capable d'atteindre une vitesse très élevée.

Taille de message (Kilo byte)	AES-192 (ms)	3DES (ms)	SPF-CA (ms)
1	82.69	86.49	34.09
10	951.2	614.9	231.7
20	1972	1096	451.2

TABLEAU 3.4 – Comparaison de la vitesse de notre algorithme SPF-CA avec d'autres chiffrements par blocs bien connus

5 Conclusion

Nous avons présenté un nouveau chiffrement par blocs hybride SPN-Feistel basé sur des automates cellulaires unidimensionnels utilisant des S-boîtes dépendantes de la clé : SPF-CA. La taille du bloc est de 128 bits et celle de la clé est de 192 bits. Notre algorithme vise à utiliser le moins de tours tout en assurant un niveau de sécurité plus élevé basé sur des S-boîtes dynamiques. La robustesse et l'efficacité de notre système proposé ont été analysées par le biais de plusieurs tests. Au final, les résultats obtenus attestent la forte résistance de notre chiffrement par blocs SPF-CA face à la menace d'une variété d'attaques : Statistiques, linéaires et différentielles.

Une séquence générée par une règle uniforme non linéaire présente un bon caractère d'aléatoire et garantit un meilleur niveau de sécurité. Toutefois, en la générant avec une séléction de règles hybrides on assure un degré algébrique plus élevé et une non-linéarité maximale tout en préservant ce caractère d'aléatoire. D'où la nécessité de proposer une variante de SPF-CA en choisissant une bonne sélection de règles hybrides cryptographiquement robustes tout en mélangeant les règles linéaires et non linéaires censées renforcer la robustesse du système de chiffrement.

Chapitre

4 SPF-CA 1.2 : Variante du système de chiffrement par blocs SPF-CA

Sommaire

1 Introduction

Les automates cellulaires sont connus par leur capacité de générer de bonnes séquences pseudo-aléatoires, une bonne caractéristique apportée par les règles linéaires. Les règles non linéaires ne possèdent pas cette caractéristique mais elles présentent un degré algébrique élevé dont résulte leur résistance à la cryptanalyse linéaire [12]. Pour ces raisons, nous avons conçu une nouvelle variante de SPF-CA : An enhanced version of cellular automata-based block cipher system (SPF-CA 1.2) [1] qui utilise un nouvel ensemble de règles hybrides bien adaptées à cette conception en combinant les règles linéaires et non linéaires de l'AC dans la fonction du tour de Feistel et durant la génération des sous-clés

pour consolider son efficacité et sa robustesse chose qui a été démontrée par le biais d'une analyse de sécurité. De plus et tout en conservant les propriétés de confusion et de diffusion élevées de la version précédente, le nombre de tours a été réduit.

Ce chapitre s'articule comme suit : La section 2 décrit SPF-CA 1.2. Quant à l'analyse de sécurité, elle est établie dans la section 3 et en ce qui concerne la section 4, elle présente les résultats statistiques et les performances de SPF-CA 1.2.

2 Description du chiffrement SPF-CA 1.2

SPF-CA [2] satisfait bien les propriétés cryptographiques le rendant ainsi résistant à différents types d'attaques en utilisant la règle 30 comme fonction du tour du schéma de Feistel et en introduisant PSOCA [29] comme algorithme de génération des clés. En vue d'utiliser plus de règles et d'explorer leur robustesse cryptographique, nous avons conçu SPF-CA 1.2 [1] qui est un chiffrement par blocs basé sur des automates cellulaires élémentaires introduisant un nouvel ensemble de règles dans le schéma de Feistel et lors de la génération des sous-clés. Cet ensemble a été sélectionné conformément aux directives de [12] qui fait appel aux règles linéaires et non linéaires également trouvées dans [40] vérifiant les propriétés cryptographiques, à savoir : Le degré algébrique, la non-linéarité et l'uniformité différentielle. Cette nouvelle variante utilise les mêmes transformations efficaces SD-boîte et P-boîte (voir 2.2.1) dans le schéma SPN . En conséquence, SPF-CA 1.2 offre un bon effet d'avalanche avec moins de tours.

2.1 Phase de chiffrement

L'objectif principal de SPF-CA 1.2 est d'exploiter les avantages des ACs en combinant les règles linéaires et non linéaires.

Afin de chiffrer un message M, SPF-CA 1.2 commence par diviser M en blocs de 128 bits pour effectuer par la suite l'opération de remplissage si nécessaire.

Les notations suivantes sont utilisées :

- $M = \{m_0, m_1, \ldots, m_{n-1}\}$: texte clair.
- $C = \{c_0, c_1, \ldots, c_{n-1}\}$: texte chiffré..
- $KS = \{ks_0, ks_1, \ldots, ks_{r-1}\}$: clé SPN de 128-bit.
- $KF = \{kf_0, kf_1, \ldots, kf_{r-1}\}$: clé Feistel de 64-bit.

où n indique le nombre de blocs et r correspond au nombre de tours.

Après avoir divisé M en blocs de 128 bits, chaque bloc est XORé avec la clé SPN correspondante et le résultat est soumis aux opérations SD-boîte et P-boîte.

La fonction du tour de Feistel utilisée dans la conception précédente SPF-CA a été attaquée par Meier et Staffelbach [45] risquant ainsi de rendre l'algorithme vulnérable. Dès lors, un nouvel ensemble de règles est proposé pour esquiver les attaques sur la fonction du tour de Feistel. Cet ensemble est une AC hybride à 3-voisins avec des conditions aux limites périodiques. Il se compose de règles linéaires et

non linéaires qui sont : $R_F = \{30, 90, 120, 150, 180, 45\}$, dictant l'évolution de bloc correspondant $64/2 = 32$ fois (voir Algorithme 4.1). L'objectif principal de l'utilisation de ces règles est d'acquérir une plus grande période de AC.

Algorithme 4.1 Evolution_AC utilisée dans SPF-CA 1.2

Fonction $Evolution\_AC(S, nbr\_iter)$ ▷ S : Un ensemble de bits
$n \leftarrow$ Longueur de S
$S_{-1} \leftarrow S_n$
$S_{n+1} \leftarrow S_1$
Pour $k = 1$ à nbr_iter **Faire**
    Pour $i = 1$ à $len(S)$ **Faire**
        Si $i \bmod 6 = 0$ **Alors**
            $R_i \leftarrow S_{i-1} \oplus (S_i \vee S_{i+1})$ ▷ Règle 30
        Fin Si
        Si $i \bmod 6 = 1$ **Alors**
            $R_i \leftarrow S_{i-1} \oplus S_{i+1}$ ▷ Règle 90
        Fin Si
        Si $i \bmod 6 = 2$ **Alors**
            $R_i \leftarrow S_{i-1} \oplus (S_i \wedge S_{i+1})$ ▷ Règle 120
        Fin Si
        Si $i \bmod 6 = 3$ **Alors**
            $R_i \leftarrow S_{i-1} \oplus S_i \oplus S_{i+1}$ ▷ Règle 150
        Fin Si
        Si $i \bmod 6 = 4$ **Alors**
            $R_i \leftarrow S_{i-1} \oplus S_i \oplus (S_i \wedge S_{i+1})$ ▷ Règle 180
        Fin Si
        Si $i \bmod 6 = 5$ **Alors**
            $R_i \leftarrow S_{i-1} \oplus (S_i \vee \neg S_{i+1})$ ▷ Règle 45
        Fin Si
    Fin Pour
Fin Pour
Renvoyer R

Dans chaque tour, le schéma de Feistel consiste à diviser les blocs en deux parties : Droite (R_i) et Gauche (L_i). La partie droite est XORée avec la sous-clé de Feistel correspondante (kf_i) et le résultat obtenu est évolué en utilisant la fonction d'évolution : Evolution_AC 4.1. Enfin, une phase de substitution est appliquée pour éviter les attaques linéaires et différentielles.

L'algorithme 4.2 représente les étapes de chiffrement pour un bloc de 128 bits.

Algorithme 4.2 Algorithme de chiffrement de SPF-CA 1.2

Entrée : M : texte clair, K : clé

Sortie : C : texte chiffré

1: $KS \leftarrow$ Extraire les 128 premiers bits de K
2: $KF \leftarrow$ Extraire les 64 derniers bits de K
3: $C \leftarrow$ ""
4: **Pour** $i = 1$ à $r - 1$ **Faire**
5: $C \leftarrow M \oplus ks[i]$
6: $C \leftarrow SD\text{-}boîte(C)$
7: $C \leftarrow P\text{-}boîte(C)$
8: $R_{i+1} \leftarrow L_i \oplus Evolution\_AC(R_i \oplus kf[i], 32)$ ▷ R_i : la partie droite et L_i : la partie gauche
9: $L_{i+1} \leftarrow R_i$
10: $C \leftarrow L_{i+1}||R_{i+1}$
11: **Fin Pour**
12: $C \leftarrow SD\text{-}boîte(C)$
13: $C \leftarrow C \oplus ks[r]$

2.2 Génération des clés

Concevoir un algorithme pour générer des clés de chiffrement est une tâche difficile d'autant plus qu'un bon chiffrement dépend d'un bon algorithme de génération des clés. Un ensemble de règles est utilisé pour produire à la fois la clé SPN de 128 bits et la clé Feistel de 64 bits. L'algorithme 4.3 prend une graine en entrée et utilise l'ensemble de règles $R_F = \{30, 90, 120, 150, 180, 45\}$ dictant ainsi une évolution de $96(= 192/2)$ fois puis générant sept sous-clés correspondant au nombre de tours de ce nouveau système de chiffrement. A posteriori, les 128 premiers bits et 64 bits restants de chaque sous-clé sont affectés respectivement aux clés SPN et Feistel (voir les algorithmes 4.2 et 4.4).

Algorithme 4.3 Génération des sous-clés de SPF-CA 1.2

Entrée : graine

Sortie : K : clé

1: **Pour** $i = 1$ à r **Faire**
2: $K \leftarrow Evolution\_AC(seed, 96)$ ▷ Génération de r sous-clés
3: **Fin Pour**

Complexité de l'algorithme

- Diviser le message M en n blocs et appliquer un remplissage nécessite au plus $s - \frac{L}{s}$ étapes, où s est la taille d'un bloc et L est la longueur du message.
- La génération des sous-clés nécessite $96 \times 192 \times r$, où r est le nombre de tours.

- Extraction des s premiers bits de K nécessite s et l'extraction des $\frac{s}{2}$ derniers bits de K nécessite $\frac{s}{2}$ étapes.
- Dans chaque tour i,
  - XORer M et ks[i] nécessite des ajouts s modulo 2.
  - Appliquer lla SD-boîte nécessite s modulo 2 divisions.
  - Appliquer la P-boîte nécessite s étapes.
  - Diviser le message M en parties droite et gauche nécessite $2 \times \frac{s}{2}$ étapes.
  - Appliquer l'Evolution_AC nécessite $\frac{s}{2} \times 32$ étapes.
  - XORer l'Evolution_AC et kf[i] nécessite des ajouts $\frac{s}{2}$ modulo 2.
  - XORer L[i] et l'$Evolution\_AC \oplus$ kf[i] nécessite des ajouts $\frac{s}{2}$ modulo 2.
  - Concaténer les deux parties, droite et gauche, nécessite s étapes.
- Appliquer la SD-boîte nécessite s modulo 2 divisions.
- Enfin, XORer C et ks[r] nécessite des ajouts s modulo 2.

Globalement, la complexité de notre algorithme est de $O\left(s^2\right)$.

2.3 Phase de déchiffrement

Au cours de la phase de déchiffrement, le texte chiffré représente une entrée pour le chiffrement en utilisant les sous-clés dans l'ordre inverse (voir l'algorithme 4.4). Les transformations invSD-boîte() et invP-boîte(); l'inverse de SD-boîte() et P-boîte() sont respectivement utilisés dans l'algorithme.

Algorithme 4.4 Algorithme de déchiffrement de SPF-CA 1.2

Entrée : C : texte chiffré, K : clé

Sortie : M : texte clair

1: $KS \leftarrow$ Extraire les 128 premiers bits de K
2: $KF \leftarrow$ Extraire les 64 derniers bits de K
3: $M \leftarrow C \oplus ks[r]$
4: $M \leftarrow invSD\text{-}boîte(M)$
5: **Pour** $i = r - 1$ à 1 **Faire**
6: $\quad R_i \leftarrow L_{i+1}$ ▷ R : la partie droite et L : la partie gauche
7: $\quad L_i \leftarrow R_{i+1} \oplus Evolution\_AC(L_{i+1} \oplus kf[i], 32)$
8: $\quad M \leftarrow L_i \| R_i$
9: $\quad M \leftarrow invP\text{-}boîte(M)$
10: $\quad M \leftarrow invSD\text{-}boîte(M)$
11: $\quad M \leftarrow M \oplus ks[i]$
12: **Fin Pour**

3 Analyse de sécurité

Comme Shannon l'a décrit dans [62], un chiffrement par blocs sécurisé devrait satisfaire les bonnes propriétés de confusion et de diffusion. Ces propriétés peuvent être mesurées par la formule 4.1 de l'effet d'avalanche. Plusieurs évaluations de sécurité ont été menées sur le SPF-CA 1,2. Cette section détaille les résultats obtenus :

$$\forall x, y \in \{0,1\}^m H(x,y) = 1 \Rightarrow \frac{\text{Average}(H(F(x), F(y)))}{m} \times 100 = 50\% \qquad (4.1)$$

Où H est la distance de Hamming et m est la longueur du bloc.

3.1 Test de diffusion

L'objectif de la propriété de diffusion est de générer une indépendance complexe entre la sortie (texte chiffré) et l'entrée (texte clair). Cette propriété est évaluée par le test de sensibilité du message pour un chiffrement par blocs. En particulier, un texte clair et une clé sont générés de manière aléatoire et un seul bit du message est modifié à chaque fois. Afin de calculer les valeurs moyennes d'avalanche, ce test vise à garder la clé inchangée puis à chiffrer le message d'origine et le message modifié par la même clé. Ici, la moyenne des valeurs d'avalanche pour chaque bit modifié du message a été appliquée à 10 paires (texte clair, clé) par l'usage de la formule 4.1.

La figure 4.1 montre que la moyenne des valeurs d'avalanche est concentrée autour de 50 % pour SPF-CA 1.2. Ainsi, le taux moyen d'effet avalanche est compris entre 46,48 % et 54,53 % pour cette nouvelle approche tandis que le taux moyen d'effet avalanche de SPF-CA se situe entre 46,56 % et 54,22 %. En conséquence, le système SPF-CA 1.2 est également sensible au changement d'un bit du message et présente une bonne propriété de diffusion.

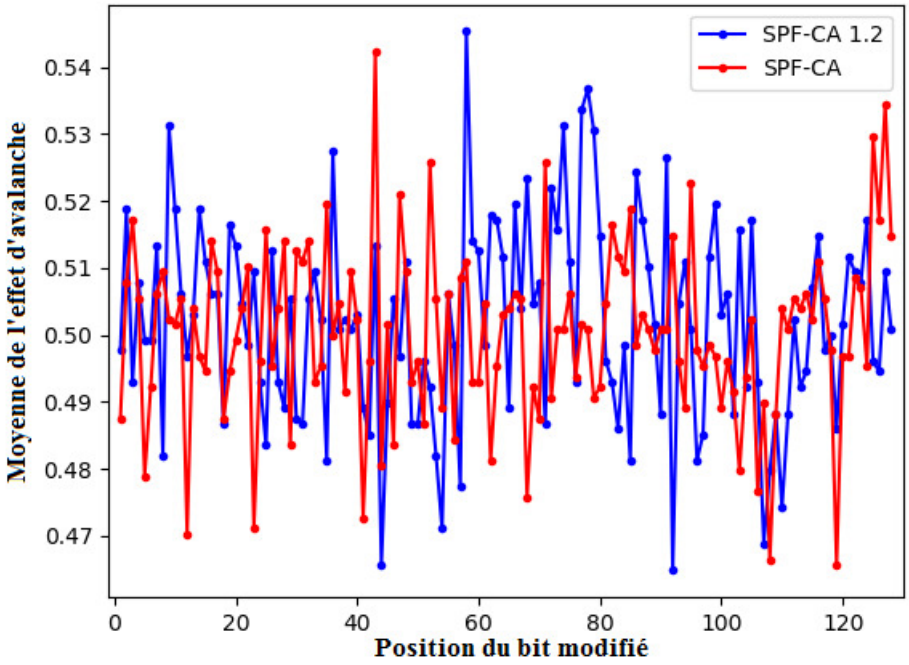


FIGURE 4.1 – SPF-CA 1.2 : Sensibilité du message au changement d'un bit

3.2 Test de confusion

La propriété de confusion consiste à établir une relation entre chaque bit du texte chiffré et plusieurs parties de la clé. Plus précisément, il fait référence à la capacité de rendre difficile la récupération de la clé à partir du texte chiffré. Pour un chiffrement par blocs, cette propriété est mesurée par le test de sensibilité de la clé. Un texte clair et une clé sont générés aléatoirement et un seul bit de la clé est modifié à chaque fois. L'objectif de ce test est de conserver le texte clair inchangé puis de le chiffrer avec la clé d'origine et aussi avec la clé modifiée tout en évaluant la moyenne d'avalanche. Ici, la moyenne des valeurs d'avalanche pour chaque bit modifié de la clé a été appliquée à 10 paires (texte clair, clé) par l'usage de la formule 4.1.

Les résultats obtenus sont présentés sur la figure 4.2. Pour SPF-CA 1.2, il apparaît que plus de 50 % des bits du texte chiffré sont modifiés. Ainsi, le taux moyen d'effet avalanche de cette nouvelle conception est compris entre 46,87 % et 55,15 % tandis que le taux moyen d'effet avalanche de SPF-CA se situe entre 45,62 % et 54,14 %. Par conséquent, le système SPF-CA 1.2 est également sensible au changement d'un bit de la clé et et présente une bonne propriété de confusion.

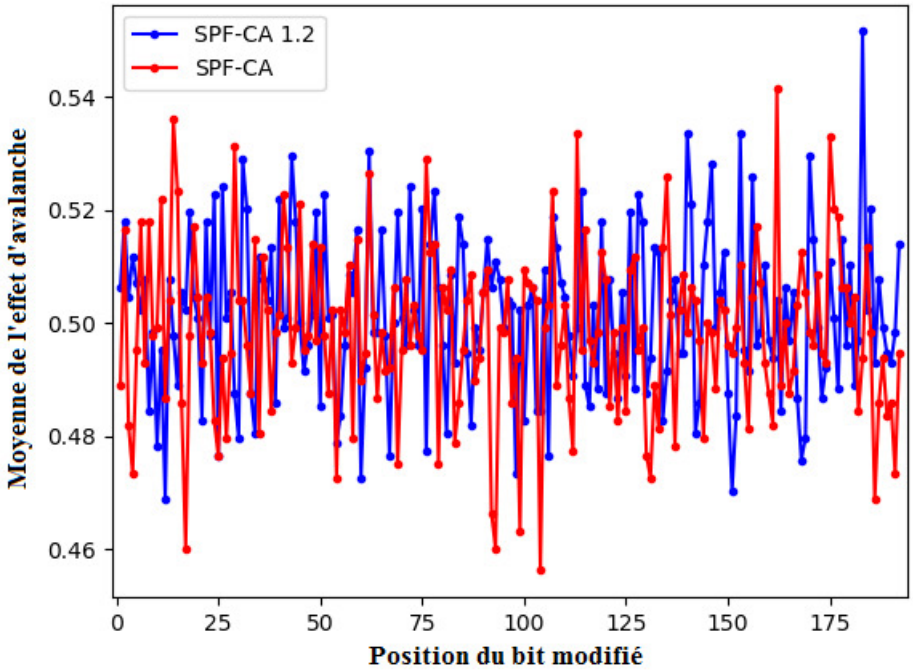


FIGURE 4.2 – SPF-CA 1.2 : Sensibilité de la clé au changement d'un bit

3.3 Analyse de la résistance aux attaques différentielles et linéaires

La cryptanalyse linéaire et différentielle représentent les majeures menaces pour les chiffrements par blocs. Ils ont d'abord été appliqués au Data Encryption Standard (DES) proposé par Biham et Shamir dans [5]. La cryptanalyse différentielle (ou Attaque à texte clair choisi) vise à attribuer des probabilités pour localiser la clé la plus probable en mesurant l'effet des différences entre les paires de texte clair et les paires de texte chiffré résultants [5]. Cette attaque consiste à trouver un algorithme qui déchiffre n'importe quel texte chiffré sans connaître la clé secrète. La cryptanalyse linéaire (ou Attaque à texte clair connu) vise à construire une approximation linéaire entre le texte clair et le texte chiffré pour deviner la clé secrète. En pratique, ces attaques se concentrent sur les propriétés de la S-boîte. Par conséquent, l'avantage de notre conception est d'utiliser une S-boîte dynamique qui fournit 256 S-boîtes, où chaque S-boîte est caractérisée par sa résistance ou sa non-résistance aux attaques différentielles et linéaires. Dans chaque bloc du texte clair, on obtient 16 S-boîtes parmi 256 possibilités qui diffèrent d'un bloc à l'autre de par leur dépendance de la clé du tour correspondant. De plus, la sensibilité élevée de la clé et du message (voir figures 4.2 et 4.1) suite à une légère modification de l'entrée rendent ces attaques plus difficiles à se produire.

3.4 Attaque par force brute

L'attaque par force brute ou recherche de clé exhaustive consiste à trouver toutes les clés valides possibles à l'aide d'un outil de prédiction. Soit une machine qui a besoin de 10^{-13} secondes pour tester la validité de la clé. SPF-CA 1.2 nécessite 2^{192} d'espace de clés possibles. Ainsi, la machine choisie exige $10^{-13} * 2^{192} = 6.27 * 10^{44}$ secondes (soit plus de $1.9 * 10^{37}$ ans) pour obtenir la bonne clé. En conséquence, l'attaque par force brute est infaisable d'un point de vue informatique.

4 Tests statistiques et performance

4.1 Paramétrage du nombre de tours

Le nombre de tours est un paramètre de sécurité important dans les primitives cryptographiques telles que les chiffrements par blocs. Pour obtenir une bonne propriété de confusion et de diffusion, une étude paramétrique du nombre de tours est réalisée. L'effet d'avalanche est une propriété nécessaire dans la cryptographie. Cela signifie que de petits changements dans l'entrée conduiraient à des changements significatifs dans la sortie [10]. Autrement dit, cette propriété mesure l'influence d'un léger changement du texte clair (ou de la clé) sur le texte chiffré. L'équation 4.1 est utilisée pour évaluer le système proposé. Ce dernier satisfait un bon effet d'avalanche si sa valeur atteint 50 %.

Par conséquent, en fonction de l'effet d'avalanche, SPF-CA 1.2 est exécuté plusieurs fois pour trouver le nombre de tours optimal. L'équation 4.1 est utilisée à la fois pour la clé et le message. Pour la première fois, un bit est modifié dans la clé d'origine et la moyenne de l'effet d'avalanche est calculée. Le tableau 4.1 illustre les résultats. De la même manière, un bit est modifié dans le message d'origine et les résultats sont indiqués dans le tableau 4.2. D'après ces résultats, l'effet d'avalanche atteint exactement 50 $\%$ pour la clé et le message lorsque le nombre de tours de SPF-CA 1.2 est de 7 alors qu'il était de 10 pour SPF-CA. En conclusion, comparé au SPF-CA, SPF-CA 1.2 obtient un bon effet d'avalanche avec moins de tours.

Nombre de tours	2	3	5	7	9	10
Effet d'avalanche (SPF-CA 1.2)(%)	49.98	49.94	50.01	50.16	50.01	50.00
Effet d'avalanche (SPF-CA)(%)	49.79	49.97	49.99	49.86	49.93	49.85

Tableau 4.1 – Nombre de tours par rapport à la valeur de l'effet d'avalanche pour la clé

Nombre de tours	2	3	5	7	9	10
Effet d'avalanche (SPF-CA 1.2)(%)	50.15	49.87	50.01	50.21	50.07	49.8
Effet d'avalanche (SPF-CA)(%)	50.18	49.97	50.05	50.08	49.82	50.03

Tableau 4.2 – Nombre de tours par rapport à la valeur de l'effet d'avalanche pour le message

4.2 Tests de Diehard

La batterie de tests Diehard est une suite de tests statistiques utilisée pour mesurer la qualité de l'aléatoire des différentes primitives cryptographiques telles que les chiffrements par blocs [41]. Ces derniers sont considérés comme sûrs si leur sortie est statistiquement indiscernable d'une sortie aléatoire. Pour réussir ces tests, on génère à partir de texte clair et de clé aléatoires un fichier binaire de taille supérieure à 10 MB composé de textes chiffrés. Ce flux est utilisé comme entrée dans la batterie de tests Diehard. Ce dernier vérifie la valeur de p après avoir analysé le fichier d'entrée. Les valeurs de p doivent être situées dans la plage $[0.025, 0.975]$.

Le tableau 4.3 montre que le SPF-CA 1.2 réussit tous les tests de Diehard. Ainsi, un comportement indiscernable d'une fonction aléatoire est présenté par ce système de chiffrement proposé.

Nom du test	P-value(SPF-CA)	P-value(SPF-CA 1.2)	Interprétation
diehard_birthdays	0.6596	0.7398	PASSED
diehard_operm5	0.4671	0.8524	PASSED
diehard_rank_32x32	0.6210	0.8480	PASSED
diehard_rank_6x8	0.4095	0.7386	PASSED
diehard_bitstream	0.2651	0.8257	PASSED
diehard_opso	0.3610	0.0822	PASSED
diehard_oqso	0.6261	0.7392	PASSED
diehard_dna	0.1283	0.7348	PASSED
diehard_count_1s_str	0.9210	0.8630	PASSED
diehard_count_1s_byt	0.2221	0.7222	PASSED
diehard_parking_lot	0.1051	0.7936	PASSED
diehard_2dsphere	0.7465	0.4989	PASSED
diehard_3dsphere	0.9333	0.7743	PASSED
diehard_squeeze	0.1462	0.9649	PASSED
diehard_runs	0.6259	0.7121	PASSED
diehard_craps	0.0883	0.8349	PASSED
marsaglia_tsang_gcd	0.3479	0.3549	PASSED
sts_monobit	0.5088	0.6159	PASSED
sts_runs	0.9420	0.5025	PASSED
sts_serial	0.6236	0.6256	PASSED
rgb_bitdist	0.6076	0.7561	PASSED
rgb_minimum_distance	0.5715	0.4574	PASSED
rgb_permutations	0.4639	0.6026	PASSED
dab_monobit2	0.6321	0.9661	PASSED

Tableau 4.3 – Batterie de test Diehard

4.3 Test de NIST

NIST Statistical Test Suite (NIST STS) est un autre standard de tests statistiques développé par le National Institute of Standards and Technology (NIST) [57]. Il consiste à tester la propriété d'aléatoire des primitives cryptographiques telles que les chiffrements par blocs. Il est utilisé , en sus de la batterie de tests Diehard, pour réaffirmer la haute qualité aléatoire du SPF-CA 1.2. Pour réaliser ces tests, un fichier binaire de taille supérieure à 10 MB a été produit à partir de textes clair et de clés aléatoires en les chiffrant par SPF-CA 1.2. Ce flux a été ensuite utilisé comme entrée dans la suite de test NIST. Un test est réussi si et seulement si la valeur p dépasse le seuil de signification $\alpha = 0.001$.

Le tableau 4.4 présente les résultats du test. Ainsi, SPF-CA 1.2 réussit tous les tests de la suite de test NIST.

Tests	P-Value (SPF-CA)	P-Value (SPF-CA 1.2)
Approximate Entropy	0.4420	0.2947
Blocks Frequency	0.7159	0.4989
Frequency	0.1767	0.2403
Cumulative Sums	0.1575	0.2530
FFT	0.2134	0.3400
Linear Complexity	0.2626	0.5762
Longest Run	0.2206	0.3648
NonOverlapping Template	0.6673	0.5773
Overlapping Template	0.3936	0.5310
Random Excursions	0.6138	0.6624
Random Excursions Variant	0.4842	0.8691
Rank	0.1914	0.2149
Runs	0.2012	0.6177
Serial	0.4031	0.3209
Universal	0.1721	0.0807

TABLEAU 4.4 – Suite de test NIST

4.4 Test de performance et compraison

Une analyse de vitesse est réalisée pour attester que la performance de l'implémentation logicielle de SPF-CA 1.2 dépasse celle de SPF-CA. L'utilisation du nouveau schéma de génération des clés proposé et le nombre minimum de tours accélèrent l'exécution de SPF-CA 1.2. Le tableau 4.5 illustre le test de performance comparé à des chiffrements par blocs bien connus utilisant Intel Core $i534227, 64$ -bit OS, 1.8GHz avec 4 GB de RAM.

Comme présenté dans le tableau 4.5, comparé à triple DES et AES [66], l'algorithme proposé SPF-CA 1.2 atteint une vitesse supérieure à celle du système SPF-CA.

Mesage size (Kilo byte)	AES-192 (ms)	3DES (ms)	SPF-CA (ms)	SPF-CA 1.2 (ms)
1	82.69	86.49	34.09	10.8
10	951.2	614.9	231.7	149.5
20	1972	1096	451.2	205.6

TABLEAU 4.5 – Comparaison du temps de chiffrement entre SPF-CA et SPF-CA 1.2 avec d'autres systèmes de chiffrement par blocs connus

5 Conclusion

Dans ce chapitre, nous avons proposé un système de chiffrement SPF-CA 1.2, une variante de SPF-CA, qui utilise un ensemble de règles linéaires et non linéaires lors de la génération des sous-clés et au niveau de la fonction du tour de Feistel. Cet ensemble de règles renforce la sécurité de l'algorithme et assure une bonne confusion et diffusion avec moins de tours. 7 tours suffisent pour obtenir de meilleures propriétés cryptographiques par rapport aux 10 tours que requiert SPF-CA. Ceci est réalisé entre autres grâce à l'algorithme de génération de clés caractérisé par sa rapidité et sa sécurité mais aussi sa capacité d'évoluer l'AC avec le même ensemble de règles.

Afin d'exploiter l'efficacité et la robustesse de notre variante proposée, SPF-CA 1.2, nous développerons une alternative aux codes d'authentification de message existants pour garantir l'authentification et l'intégrité des données et répondre aux exigences de sécurité des appareils de l'Internet des Objets (IoT).

Chapitre

5 CMAC basé sur SPF-CA 1.2

Sommaire

1 Introduction

Au cours des dernières années, nos sociétés ont connu un développement technologique rapide grâce à l'avènement et au développement d'Internet ainsi qu'aux avancées technologiques en matière d'électronique et logiciels. Actuellement, le nombre d'appareils IoT interconnectés est amplement important. Cependant, la plupart de ces appareils connectés souffrent d'un manque de sécurité qui peut être sujet à des failles de sécurité, notamment, la falsification de données. Ceci explique le besoin croissant de développer des primitives cryptographiques légères adaptées à la puissance de calcul et à la capacité de stockage de ces dispositifs. Pour cette raison, nous nous intéressons à l'authenticité de ces données entre l'émetteur et le récepteur en permettant la vérification de l'identité exigée d'une entité souhaitant accéder à l'information [64] à travers la conception d'un MAC (voir section 1.3). Ces derniers ont été décrits comme l'une des solutions cryptographiques les plus robustes qui assurent l'intégrité et l'authentification des données.

Un MAC inclut deux algorithmes : l'algorithme de génération du Tag (MACG_k) et l'algorithme de vérification du Tag (MACV_k). L'algorithme de génération du Tag (MACG_k) est basé sur l'utilisation d'une clé secrète k et un message M pour générer le Tag.

$$\text{Tag} = \text{MACG}_k(M) \tag{5.1}$$

Le deuxième algorithme permet au récepteur du Tag de vérifier son authenticité en prouvant son identité et en examinant s'il a subi une altération au cours de sa transmission (voir figure 5.1).

$$\text{MACV}_k(M,\ \text{Tag}\ ) = \{\text{valid, invalid}\} \tag{5.2}$$

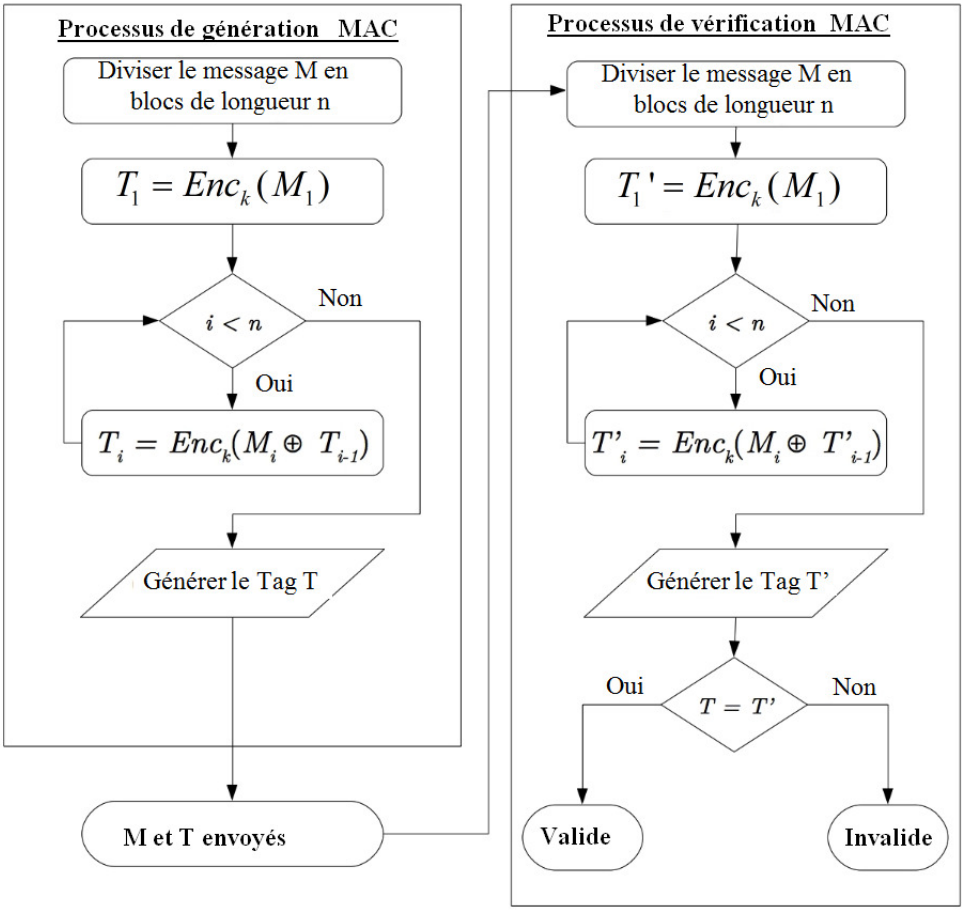


FIGURE 5.1 – Processus de génération et de vérification d'un MAC

Dans ce chapitre, nous développons dans la section 2 un code d'authentification de message basé sur SPF-CA 1.2. pour finalement examiner sa sécurité et ses performances dans les sections 3 et 4.

2 Description de CMAC-SPF-CA 1.2

Il existe deux principales familles d'algorithmes MACs : Les algorithmes MACs basés sur les fonctions de hachage et ceux basés sur les systèmes de chiffrement. Spécialement, les constructions de MACs les plus connues basées sur un système de chiffrement sont : Cipher Message Authentication Code (CMAC) un Code d'Authentification de Message basé sur un chiffrement par bloc [17, 18] et CBC-MAC un Code d'Authentification de Message basé sur un chiffrement par chaînage de blocs [23]. Or ce dernier présente des lacunes en matière de sécurité : Son remplissage cause une surcharge en particulier lorsque la longueur du message en bits est un multiple entier de la longueur du bloc. En 2005, NIST a remedié à cette faille à travers la standarisation d'un algorithme CMAC [17, 18]. Cette contribution a permis une meilleure assurance d'intégrité des données par rapport à une somme de contrôle ou un code de détection d'erreur qui sont conçus pour détecter uniquement les modifications accidentelles des données contrairement à CMAC conçu pour détecter en plus des modifications accidentelles, les altérations intentionnelles et non autorisées.

En utilisant une fonction de chiffrement E et une clé secrète k, chaque bloc de longueur n est chiffré d'une manière itérative [64]. À la fin de ce processus, le Tag généré est envoyé avec le message (voir figure 5.1).

En outre, dans la phase de vérification MAC, le récepteur lance le processus de génération MAC pour obtenir un Tag' qu'il compare au Tag reçu puis décide si le message est intègre et authentique ou s'il a été altéré. Par conséquent, l'insuffisance en Codes d'Authentification de Message basés sur les systèmes de chiffrement nous a incité à concevoir et implémenter une version répondant aux exigences d'intégrité et d'authentification des données.

2.1 Processus de génération de CMAC-SPF-CA 1.2

CMAC [17, 18] est un algorithme MAC basé sur des chiffrements par blocs, c'est une variante de CBC-MAC. Chaque bloc n'a besoin que d'une seule clé de chiffrement et est hautement optimisé en termes de nombre de chiffrements.

Notre processus de génération proposé divise d'abord le texte clair M en n blocs de 128 bits moyennant une opération de remplissage, si nécessaire, où il ajoute une séquence de la forme '100..0' pour avoir les 128 bits. A partir de ceci, nous utilisons la clé de chiffrement par bloc k pour en dériver deux clés secrètes supplémentaires k_1 et k_2. Si le dernier bloc est complet on lui ajoute k_1, sinon, on lui ajoute k_2. Et ceci comme illustré dans le schéma 5.2.

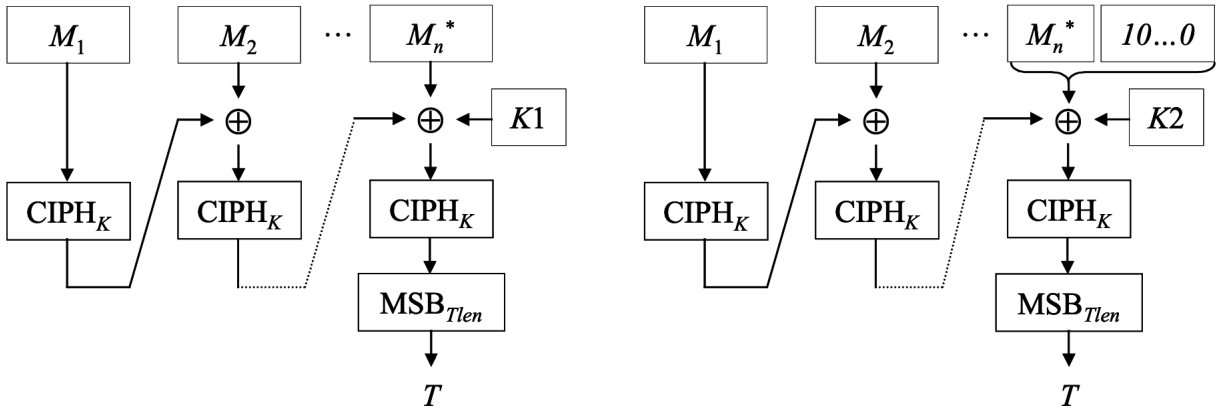


FIGURE 5.2 – Processus général d'un CMAC

À gauche le cas où la longueur du message est un multiple de la taille du bloc; à droite est le cas où la longueur du message n'est pas un multiple de la longueur du bloc.

2.1.1 Génération des sous-clés k_1 et k_2

Le chiffrement par blocs SPF-CA 1.2 est appliqué au bloc qui se compose entièrement de 128 bits de 0. Puis, la première sous-clé k_1 est dérivée de la chaîne résultante par un décalage vers la gauche d'un bit et, conditionnellement par XOR d'une constante qui dépend de la taille du bloc, R_{128}. Ensuite, la deuxième sous-clé k_2 est dérivée de la même manière à partir de la première sous-clé k_1. Notons que la sous-clé SPF-CA 1.2_k (0^{128}) doit être secrète. L'algorithme de génération de ces deux clés est décrit dans 5.1 :

Algorithme 5.1 Algorithme de génération de k_1 et k_2

Entrée : Clé k

Sortie : k_1 et k_2

1: $L = \text{SPF-CA1.2}_k\,(0^{128})$
2: **Si** $\text{MSB}_1(L) = 0$ **Alors**
3: $\quad k_1 = L << 1$
4: **Sinon**
5: $\quad k_1 = (L << 1) \oplus R_{128}$
6: **Fin Si**
7: **Si** $\text{MSB}_1(k_1) = 0$ **Alors**
8: $\quad k_2 = k_1 << 1$
9: **Sinon**
10: $\quad k_2 = (k_1 << 1) \oplus R_{128}$
11: **Fin Si**

où :

- 0^s désigne la chaîne composée de s bits de 0.
- $||$ représente l'opération de concaténation sur les chaînes de bits.
- R_{128} est une représentation d'un polynôme binaire irréductible de degré 128, à savoir, le premier lexicographiquement parmi tous ces polynômes avec le nombre minimum possible de termes non nuls. Si ce polynôme est exprimé par $u^{128} + c_{127}u^{127} + \ldots + c_2u^2 + c_1u + c_0$, où les coefficients $c_{127}, c_{126}, \ldots, c_2, c_1, c_0$ valent 0 ou 1, alors R_{128} est le bit chaîne $c_{127}c_{126} \ldots c_2c_1c_0$. En particulier, $R_{128} = 0^{120}10000111$ est l'expression approuvée pour les chiffrements par blocs de taille 128.
- Étant donné une chaîne de bits X, les fonctions $\mathrm{LSB}_s(X)$ et $\mathrm{MSB}_s(X)$ renvoient les s les moins significatifs (c'est-à-dire les plus à droite) bits et les s bits les plus significatifs (c'est-à-dire les plus à gauche), respectivement, de X. Par exemple, $\mathrm{LSB}_3(111011010) = 010$ et $\mathrm{MSB}_4(111011010) = 1110$.
- Étant donné une chaîne de bits X composée de bits Xlen, la fonction de décalage vers la gauche, notée $X << 1$, est $\mathrm{LSB}_{X\,\mathrm{len}}\,(X||0)$, avec $Xlen$ la longueur de X. Par exemple, $1101110 << 1 = 1011100$.

2.1.2 Génération de CMAC-SPF-CA 1.2

Après avoir généré les sous-clés k_1 et k_2, le message d'entrée est formaté en une séquence de blocs complets dans lesquels le bloc final a été masqué par une sous-clé. Notons deux cas :

- Si la longueur du message est un multiple de la taille du bloc, le message est alors décomposé en blocs complets. Le bloc final est XORé avec la première sous-clé k_1. La séquence de blocs résultante est le message formaté.
- Si la longueur du message n'est pas un multiple de la taille du bloc, alors le message est divisé en blocs complets, dans la mesure du possible, suivis d'une chaîne de bits finale dont la longueur est inférieure à la taille du bloc. Une chaîne de remplissage est ajoutée à cette chaîne de bits finale, en particulier, un bit de " 1 " suivi du nombre minimum de bits de " 0 " qui sont nécessaires pour compléter le dernier bloc. Ce bloc final complété est XORé avec la deuxième sous-clé k_2. La séquence de blocs résultante est le message formaté.

Puis, la technique de chaînage de blocs de chiffrement (CBC) avec le bloc zéro comme vecteur d'initialisation est appliquée au message formaté. Finalement, le bloc de sortie CBC final est tronqué en fonction du paramètre de la longueur du MAC associé à la clé et le résultat est ensuite renvoyé en tant que MAC.

L'algorithme 5.2 illustre les étapes de génération de CMAC-SPF-CA 1.2 :

Algorithme 5.2 Algorithme de génération de CMAC-SPF-CA 1.2

Entrée : Message M, Clé k

Sortie : Tag

1: Appliquer le processus de génération de $K1$ et $K2$ à la clé K (voir 2.1.1)
2: **Si** $lenM = 0$ **Alors** ▷ $LenM$: Longueur de M
3: $n = 1$ ▷ n : Nombre de blocs
4: **Sinon**
5: $n = \lceil Mlen/128 \rceil$ ▷ Le plus petit entier qui n'est pas inférieur à $Mlen/128$.
6: **Fin Si**
7: Diviser le message M en blocs de taille 128 : $M_1, M_2, \ldots, M_{n-1}, M_n^*$ la séquence unique de chaînes de bits telle que $M = M_1||M_2|| \ldots ||M_{n-1}||M_n{}^*$, où $M_1, M_2, \ldots, M_{n-1}$ sont des blocs complets de taille 128
8: **Si** $M_n{}^*$ **Alors** est un bloc complet
9: $M_n = k_1 \oplus M_n^*$
10: **Sinon**
11: $M_n = k_2 \oplus (M_n^*||10^j)$, où $j = 128n - Mlen - 1$
12: **Fin Si**
13: $C_0 = 0^{128}$
14: **Pour** $i = 1$ à n **Faire**
15: $C_i = \text{SPF-CA 1.}\, 2_k\, (C_{i-1} \oplus M_i)$.
16: **Fin Pour**
17: $T = \text{MSB}_{\text{Tlen}}\ (C_n)$ ▷ $lenT$: Longueur du Tag

2.2 Processus de vérification de CMAC-SPF-CA 1.2

Initialement, le processus de génération MAC vu dans les sections 2.1.1 et 2.1.2 est appliqué au message M. Finalement, le MAC résultant est comparé au MAC reçu pour déterminer sa validité.

Algorithme 5.3 Algorithme de vérification de CMAC-SPF-CA 1.2

Entrée : Message M de longueur $lenM$, MAC reçu Tag'

Sortie : VALIDE ou INVALIDE

1: Appliquer le processus de génération MAC à M pour produire Tag (voir 2.1.2)
2: **Si** $Tag = Tag'$ **Alors**
3: **Renvoyer** VALIDE
4: **Sinon**
5: **Renvoyer** INVALIDE
6: **Fin Si**

3 Analyse de sécurité

Dans cette section nous évaluons la performance et la résistance de CMAC-SPF-CA 1.2 face aux attaques de récupération de clé et de falsification. Nous exposons également les résultats expérimentaux et tests effectués sur les mécanismes. En particulier, la sécurité du MAC se base sur la robustesse du système de chiffrement utilisé dans son processus. En outre, les entités malveillantes visant à fournir une valeur d'authentification valide devrait avoir une complexité en temps polynomial lorsqu'elle tente de forger des messages [32]. En principe, l'entité malveillante peut prendre l'une des deux attaques : les attaques de récupération de clé qui consiste à trouver la clé secrète à partir de nombreuses paires (message/Tag) et l'attaque par falsification consistant à forger un Tag valide. Lorsqu'une telle attaque peut être menée sur un seul message, on parle d'attaque existentielle. Si c'est possible pour un Tag choisi par une personne, elle est appelée attaque sélective. [80]. (voir section 1.3.1)

3.1 La résistance de CMAC-SPF-CA 1.2 à l'attaque de récupération de clé

3.1.1 Sensibilité de la clé au changement d'un bit

En vue d'évaluer la sensibilité de la clé dans le système d'authentification CMAC-SPF-CA 1.2, nous avons utilisé la notion d'effet d'avalanche comme mesure de la non-linéarité du système proposé : De très petites différences dans l'entrée devraient provoquer d'importants changements dans la sortie. Mathématiquement, une fonction $F : \{0,1\}^m \rightarrow \{0,1\}^n$ a l'effet d'avalanche si le résultat suivant est vérifié :

$$\forall x, y \in \{0,1\}^m H(x,y) = 1 \Rightarrow \text{moyenne}(H(F(x), F(y))) = n/2 \tag{5.3}$$

Où H représente la distance de Hamming entre deux blocs de n bits.

Selon l'équation 5.3, un minimum changement aléatoire sur l'entrée (un bit) devrait entraîner en moyenne un maximum changement sur la sortie (la moitié des bits). Ce qui reflète le concept général d'indépendance entre l'entrée et la sortie.

Ainsi, ce test de sensibilité au changement de la clé consiste donc à mesurer la différence entre le message et le Tag généré à partir de la même clé. Pour effectuer ce test nous avons généré une clé fixe K de chiffrement et un ensemble de messages en clair aléatoires $M = \{M_1, M_2, \ldots, M_L\}$. Pour chaque paire (M_i, K) on génère le Tag en utilisant la clé K puis on modifie un seul bit dans la clé pour recalculer le Tag correspondant tout en laissant le message inchangé. Ensuite nous calculons la distance de Hamming entre chaque pair (M_i, K_j) et (M_i, K), ceci est répété pour j $= 1 \ldots$ L et i $= 1 \ldots 192$. Suite à cela, nous appliquons l'équation 5.3 sur les distances de Hamming obtenues. Comme nous pouvons constater dans la figure 5.3, la modification d'un seul bit de la clé d'entrée produit une variation de 50% dans le Tag résultant en fonction de la moyenne des distances de Hamming. D'où, CMAC-SPF-CA 1.2 est très sensible aux variations de chaque bit de la clé.

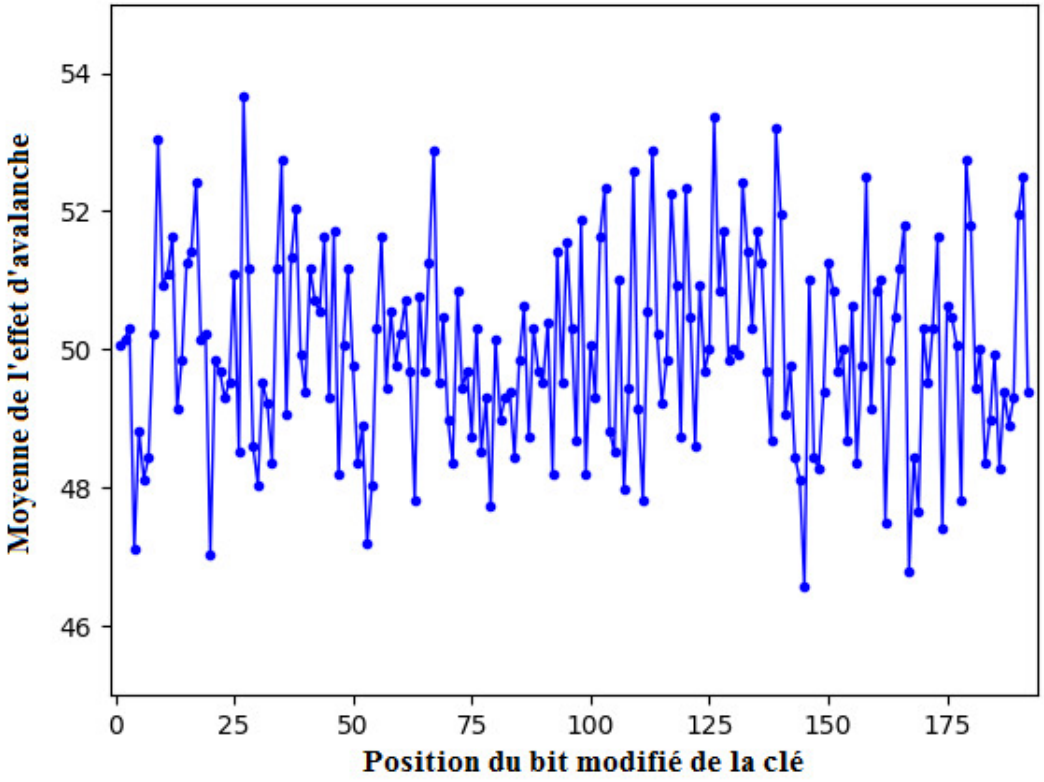


FIGURE 5.3 – CMAC-SPF-CA 1.2 : Sensibilité de la clé au changement d'un bit

3.1.2 Recherche exhaustive de la clé

Il s'agit d'une attaque au cours de laquelle une entité malveillante essaie d'obtenir toutes les clés possibles à l'aide d'un outil de prédiction. Soit une machine qui a besoin de 10^{-10} secondes pour tester la validité de la clé. CMAC-SPF-CA 1.2 nécessite 2^{192} d'espace de clés possibles. Ainsi, la machine choisie exige $10^{-10} \times 2^{192} = 6.28 \times 10^{47}$ secondes (soit plus de 1.99×10^{40} ans) pour obtenir la bonne clé. Par conséquent, l'attaque de recherche exhaustive de la clé est impossible dans un temps raisonnable.

3.2 La résistance de CMAC-SPF-CA 1.2 à l'attaque de falsification

3.2.1 Attaque de falsification contre MAC

L'attaque de falsification est l'attaque la plus connue pour les MACs dans laquelle une entité malveillante peut délivrer au moins une paire valide de Message/Tag sans avoir la clé secrète. En général, l'attaque de falsification contre un MAC la plus connue est celle du paradoxe des anniversaires basée sur la fréquence des collisions trouvées. Cette attaque nécessite environ O $\left(2^{n/2}\right)$ de (message /Tag) paires connues, où n est la longueur du Tag. Pour réussir une attaque de falsification contre CMAC-SPF-CA 1.2, une entité malveillante a besoin de connaitre environ O (2^{64}) de paires (message / Tag).

3.2.2 Sensibilité du message d'origine au changement d'un bit

Le test de sensibilité du message d'origine face au changement d'un bit dans le système d'authentification CMAC-SPF-CA 1.2 vise à vérifier la relation entre le message et le Tag en utilisant la notion d'effet d'avalanche. Ainsi, pour effectuer ce test nous avons généré une clé fixe K de chiffrement et un ensemble de messages en clair aléatoires $M = \{M_1, M_2, \ldots, M_L\}$. Pour chaque paire (M_i, K), on reprend le même schéma pour mesurer l'impact du changement d'un seul bit de chaque message sur le Tag par 5.3 en utilisant la clé K, puis on modifie un seul bit du message pour recalculer le Tag correspondant tout en laissant la clé inchangée.

Les résultats illustrés sur la figure 5.4 montrent que la modification d'un seul bit du message d'origine génère une variation de 50% dans le Tag résultant en fonction de la moyenne des distances de Hamming. D'où, CMAC-SPF-CA 1.2 est très sensible aux variations de chaque bit du message d'origine.

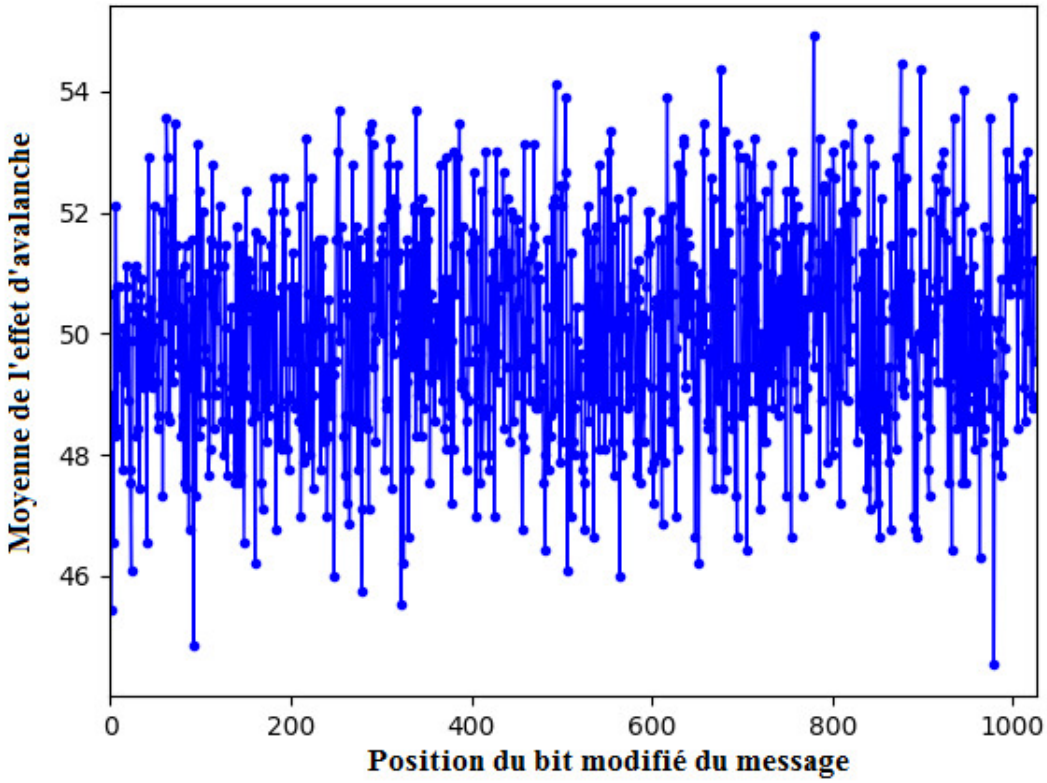


FIGURE 5.4 – CMAC-SPF-CA 1.2 : Sensibilité du message d'origine au changement d'un bit

Le tableau 5.1 présente les valeurs d'effet d'avalanche de la variation d'un bit de l'entrée. Notons A_{min}, A_{max}, A_{mean} et $A(\%)$ désignant respectivement l'avalanche minimal, l'avalanche maximal, l'avalanche moyenne et le taux d'avalanche. Presque 50% des bits de sortie est affecté.

	A_{min}	A_{max}	A_{mean}	$A(\%)$
CMAC-SPF-CA 1.2	$59,6$	$68,7$	$63,99$	$49,99$

TABLEAU 5.1 – Effet d'avalanche de notre proposition

4 Analyse statistique et performance

4.1 Tests statistiques de Dieharder

La batterie de tests Dieharder a été conçue par Robert G. Brown pour vérifier le comportement des PRNG et des primitives cryptographiques telles que les systèmes de chiffrement, les fonctions de hachage et les MACs. Il s'agit de tests de Diehard, de certains tests du NIST et d'autres tests développés par Brown et Bauer [9]. Principalement, le Tag transféré devrait être indiscernable d'un autre Tag. Pour effectuer ce test, nous avons généré un flux de données de 1Mo par CMAC-SPF-CA 1.2. Ce flux a été créé en combinant tous les Tags générés par notre algorithme de génération MAC. pour ensuite être analysé statistiquement par Dieharder [9].

La séquence est dite aléatoire si les valeurs p sont comprises entre $0,005$ et $0,995$. Dans ce cas la séquence de bits est indiscernable d'une séquence binaire aléatoire.

Nom du test	P-value	Interprétation
diehard_birthdays	0.52287411	PASSED
diehard_operm5	0.69133458	PASSED
diehard_rank_32x32	0.85843512	PASSED
diehard_rank_6x8	0.97606718	PASSED
diehard_bitstream	0.00500638	PASSED
diehard_opso	0.64592841	PASSED
diehard_oqso	0.92287395	PASSED
diehard_dna	0.48570767	PASSED
diehard_count_1s_str	0.92194245	PASSED
diehard_count_1s_byt	0.30631849	PASSED
diehard_parking_lot	0.77403241	PASSED
diehard_2dsphere	0.90741185	PASSED
diehard_3dsphere	0.27396097	PASSED
diehard_squeeze	0.58111880	PASSED
diehard_sums	0.69362144	PASSED
diehard_runs	0.90097556	PASSED
diehard_craps	0.38296983	PASSED
marsaglia_tsang_gcd	0.83952424	PASSED
sts_monobit	0.37766827	PASSED
sts_runs	0.76698249	PASSED
sts_serial	0.77234997	PASSED
rgb_bitdist	0.21762991	PASSED
rgb_minimum_distance	0.16257667	PASSED
rgb_permutations	0.67212346	PASSED
rgb_lagged_sum	0.96040147	PASSED
rgb_kstest_test	0.76969467	PASSED
dab_bytedistrib	0.11023175	PASSED
dab_dct	0.14806169	PASSED
dab_filltree	0.77837817	PASSED
dab_filltree2	0.14861727	PASSED
dab_monobit2	0.90232987	PASSED

Tableau 5.2 – Batterie de test Dieharder

Le tableau 5.2 montre que le flux binaire obtenu en utilisant notre conception proposée a réussi tous les tests de DIEHARDER nécessaires. En outre, CMAC-SPF-CA 1.2 permet de produire un bon comportement aléatoire et peut être statistiquement indiscernable des séquences aléatoires.

4.2 Performance

L'objectif de notre travail est de présenter un nouveau code d'authentification de message efficace garantissant l'authenticité des données : CMAC-SPF-CA 1.2. Ce dernier atteint des performances satisfaisantes en utilisant des implémentations logicielles simples. Le tableau 5.3 montre que CMAC-SPF-CA 1.2 offre une diversité de clés aussi bonne que d'autres codes d'authentification de message basés sur des chiffrements bien connus. En plus CMAC-SPF-CA 1.2 se démarque par sa robustesse provenant de celle de SPF-CA 1.2, dont il est lié.

	taille de la clé (en bits)	le nombre de clés possibles
CMAC-AES	$128, 192, 256$	$2^{128}, 2^{192}, 2^{256}$
CMAC-3DES	256 (2 clés de 128 bits)	2^{256}
CMAC-SPF-CA 1.2	192	2^{192}

Tableau 5.3 – Diversité de clés de CMAC-SPF-CA 1.2 par rapport aux autres CMACs connus

5 Conclusion

Dans ce chapitre, nous avons présenté une extension de SPF-CA 1.2 en se basant sur les recommandations de NIST qui rentre dans les nouvelles technologies IoT et ce dans le but d'assurer l'authenticité des données échangées. Le système proposé est robuste en partie grâce à sa liaison avec SPF-CA 1.2. Il satisfait également l'exigence d'intégrité et d'authentification des données et résiste aux attaques de falsification et de récupération de clés. De plus, il offre une diversité de clés similaire aux codes d'authentification de message basés sur des chiffrements bien connus. Au final, il est à noter que notre MAC a réussi les tests de Dieharder.

CONCLUSION

1 Apports de la thèse

Les travaux de recherches de cette thèse s'inscrivent dans le cadre de conception et réalisation des solutions cryptographiques à clé secrète robustes basées sur les ACs. Initialement, la première contribution concerne BC-CaACO, un chiffrement par blocs basé sur des ACs réversibles et irréversibles en introduisant des boites de substitutions efficaces basées sur l'optimisation par colonie de fourmis offrant une meilleure confusion et diffusion par rapport aux systèmes de chiffrement par blocs existants. Par suite, la deuxième contribution s'axe sur le chiffrement par blocs SPF combinant les deux schémas classiques SPN et Feistel et utilisant des S-boîtes dynamiques dépendantes de la clé et une P-boîte robustes et efficaces. Ce système fait appel au générateur PSOCA pour générer les sous-clés. Notre conception proposée surpassse l'algorithme de chiffrement par blocs bien connu en terme d'efficacité et de résistance aux attaques existantes telles que la cryptanalyse différentielle, linéaire et les attaques statistiques. En ce qui concerne la troisième contribution, elle présente une variante de SPF, nommée SPF-CA 1.2, qui explore trois majeurs avantages des ACs, à savoir, un degré algébrique élevé, une non-linéarité maximale et un caractère d'aléatoire important, en utilsant un ensemble de règles linéaires et non linéaires bien choisi dans la fonction du tour de Feistel et lors la génération des sous-clés. Cette variante est efficace et robuste ce qui est démontrée à travers une analyse de sécurité. De plus, elle conserve une bonne confusion et diffusion tout en réduisant le nombre de tours par rapport à sa version antérieure. Finalement, le dernier axe de recherche concerne une extension de SPF-CA 1.2 dans l'authenticité, nommée CMAC-SPF-CA 1.2, qu'on peut intégrer dans IoT satisfaisante aux exigences de sécurité et résistante aux attaques de falsification de MAC et de récupération de clés.

2 Perspectives d'approfondissement

Ce travail propose des solutions cryptographiques à clé secrète basées sur les ACs pouvant être implémentées dans des ressources limitées. Cependant, elles peuvent être améliorées suivant les exigences de sécurité et les contraintes des nouvelles technologies :

- Proposer la version lightweight de SPF-CA 1.2.
- Proposer une amélioration de CMAC-SPF-CA 1.2.
- Approfondir l'étude des ACs pour ensuite les intégrer dans de nouvelles solutions cryptographiques.

Liste des publications et communications

1. **Achkoun, K.**, Hanin, C., Sadak, A., Ziani, F., & Omary, F. (2021). SPF-CA-1.2 : An enhanced version of cellular automata based block cipher system. **International Journal of Computer Mathematics : Computer Systems Theory**, (just-accepted), 1-17.

2. Abouchouar, A., Omary, F., & **Achkoun, K.** (2020). New concept for cryptographic construction design based on noniterative behavior. **IAES International Journal of Artificial Intelligence**, 9(2), 229.

3. **Achkoun, K.**, Hanin, C., & Omary, F. (2020). SPF-CA : A new cellular automata based block cipher using key-dependent S-boxes. **Journal of Discrete Mathematical Sciences and Cryptography**, 23(8), 1529-1544.

4. C.Hanin, F. Omary, S. Elbernoussi, **K. Achkoun**, B. Echandouri « A New Block Cipher System Using Cellular Automata and Ant Colony Optimization (BC-CaACO) », ***International Journal of Information Security and Privacy (IJISP)***.

Bibliographie

[1] K Achkoun, C Hanin, A Sadak, F Ziani, and F Omary. Spf-ca-1.2 : An enhanced version of cellular automata based block cipher system. *International Journal of Computer Mathematics : Computer Systems Theory*, 6(3) :194–208, 2021.

[2] Khadija Achkoun, Charifa Hanin, and Fouzia Omary. Spf-ca : A new cellular automata based block cipher using key-dependent s-boxes. *Journal of Discrete Mathematical Sciences and Cryptography*, 23(8) :1529–1544, 2020.

[3] Musheer Ahmad, Deepanshu Bhatia, and Yusuf Hassan. A novel ant colony optimization based scheme for substitution box design. *Procedia Computer Science*, 57 :572–580, 2015.

[4] Jaydeb Bhaumik and Dipanwita Roy Chowdhury. Design and implementation of cellular automata based diffusion layer for spn-type block cipher. In *Informatics, Electronics & Vision (ICIEV), 2012 International Conference on*, pages 828–831. IEEE, 2012.

[5] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1) :3–72, 1991.

[6] Alex Biryukov and David Wagner. Slide attacks. In *International Workshop on Fast Software Encryption*, pages 245–259. Springer, 1999.

[7] Said Bouchkaren and Saiida Lazaar. A fast cryptosystem using reversible cellular automata. *Int. J. Adv. Comput. Sci. Appl*, 5(5) :207–210, 2014.

[8] Said Bouchkaren and Saiida Lazaar. A new iterative secret key cryptosystem based on reversible and irreversible cellular automata. *IJ Network Security*, 18(2) :345–353, 2016.

[9] Robert G Brown, Dirk Eddelbuettel, and David Bauer. dieharder : A random number test suite, 2007. *URL http ://www. phy. duke. edu/~ rgb/General/dieharder. php. C program archive dieharder, version*, 2(3), 2020.

[10] Julio Cesar Hernandez Castro, José María Sierra, Andre Seznec, Antonio Izquierdo, and Arturo Ribagorda. The strict avalanche criterion randomness test. *Mathematics and Computers in Simulation*, 68(1) :1–7, 2005.

[11] Xiuli Chai, Zhihua Gan, Kang Yang, Yiran Chen, and Xianxing Liu. An image encryption algorithm based on the memristive hyperchaotic system, cellular automata and dna sequence operations. *Signal Processing : Image Communication*, 52 :6–19, 2017.

[12] Kaushik Chakraborty and Dipanwita Roy Chowdhury. Cshr : selection of cryptographically suitable hybrid cellular automata rule. In *International Conference on Cellular Automata*, pages 591–600. Springer, 2012.

[13] Parimal Pal Chaudhuri. *Additive cellular automata : theory and applications*, volume 1. John Wiley & Sons, 1997.

[14] AK Das, A Ganguly, A Dasgupta, S Bhawmik, and P Pal Chaudhuri. Efficient characterisation of cellular automata. *IEE Proceedings E-Computers and Digital Techniques*, 137(1) :81–87, 1990.

[15] Debasis Das and Abhishek Ray. A parallel encryption algorithm for block ciphers based on reversible programmable cellular automata. *arXiv preprint arXiv :1006.2822*, 2010.

[16] Alberto Dennunzio, Enrico Formenti, Darij Grinberg, and Luciano Margara. Decidable characterizations of dynamical properties for additive cellular automata over a finite abelian group with applications to data encryption. *Information Sciences*, 563 :183–195, 2021.

[17] Morris Dworkin. Recommendation for block cipher modes of operation : The cmac mode for authentication. *National Institute of Standards and Technology (NIST) Special Publication*, 2005.

[18] Morris J Dworkin et al. Recommendation for block cipher modes of operation : The cmac mode for authentication. *National Institute of Standards and Technology (NIST) Special Publication*, 2016.

[19] Bouchra Echandouri, Charifa Hanin, Fouzia Omary, and Souad Elbernoussi. Keyed-cahash : a new fast keyed hash function based on cellular automata for authentication. *Int. J. Comput. Sci. Appl*, 14(2) :64–180, 2017.

[20] Kamel Mohamed Faraoun. Design of fast one-pass authenticated and randomized encryption schema using reversible cellular automata. *Communications in Nonlinear Science and Numerical Simulation*, 19(9) :3136–3148, 2014.

[21] Kamel Mohamed Faraoun. A genetic strategy to design cellular automata based block ciphers. *Expert Systems with Applications*, 41(17) :7958–7967, 2014.

[22] PUB FIPS. 81, des modes of operation. *Issued December*, 2 :63, 1980.

[23] Sheila Frankel and Howard Herbert. Rfc3566 : The aes-xcbc-mac-96 algorithm and its use with ipsec, 2003.

[24] Bhoopal Rao Gangadari and Shaik Rafi Ahamed. Low power s-box architecture for aes algorithm using programmable second order reversible cellular automata : An application to wban. *Journal of medical systems*, 40(12) :1–13, 2016.

[25] Niloy Ganguly, Biplab K Sikdar, and P Pal Chaudhuri. Theory of additive cellular automata. *Fundamenta Informaticae*, 21 :1001–1021, 2001.

[26] Benoît Gérard. *Cryptanalyses statistiques des algorithmes de chiffrement à clef secrète.* PhD thesis, Université Pierre et Marie Curie-Paris VI, 2010.

[27] Clay S Gloster and Franc Brglez. Boundary scan with cellular-based built-in self-test. In *Test Conference, 1988. Proceedings. New Frontiers in Testing, International*, pages 138–145. IEEE, 1988.

[28] Charifa Hanin, Bouchra Echandouri, Fouzia Omary, and Souad El Bernoussi. L-cahash : A novel lightweight hash function based on cellular automata for rfid. In *International Symposium on Ubiquitous Networking*, pages 287–298. Springer, 2017.

[29] Charifa Hanin, Fouzia Omary, Bouchra Boulahiat, and Souad Elbernoussi. Design of new pseudo-random number generator based on non-uniform cellular automata. *International Journal of Security and Its Applications*, 10(11) :109–118, 2016.

[30] Charifa Hanin, Fouzia Omary, Souad Elbernoussi, Khadija Achkoun, and Bouchra Echandouri. A new block cipher system using cellular automata and ant colony optimization (bc-caaco). *International Journal of Information Security and Privacy(IJISP)*, 2017.

[31] Peter D. Hortensius, Robert D. McLeod, and Howard C. Card. Parallel random number generation for vlsi systems using cellular automata. *IEEE Transactions on Computers*, 38(10) :1466–1473, 1989.

[32] Shaoquan Jiang. On the size of source space in a secure mac. *IEEE Transactions on Information Forensics and Security*, 10(9) :2007–2015, 2015.

[33] Jossy Joseph, Joseph Jacob, MK Abinshad, KN Ambili, and Jimmy Jose. Strengthening acorn authenticated cipher with cellular automata. In *International Conference on Cellular Automata for Research and Industry*, pages 8–17. Springer, 2020.

[34] Jarkko Kari. Reversibility of 2d cellular automata is undecidable. *Physica D : Nonlinear Phenomena*, 45(1-3) :379–385, 1990.

[35] Jarkko Kari. Cryptosystems based on reversible cellular automata. *Manuscript, August*, 1992.

[36] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177) :203–209, 1987.

[37] Sukhendu Kuila, Dhiman Saha, Madhumangal Pal, and Dipanwita Roy Chowdhury. Cash : cellular automata based parameterized hash. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 59–75. Springer, 2014.

[38] Kai Li, Mingyu Sun, Lvzhou Li, and Juhua Chen. Image encryption algorithms based on non-uniform second-order reversible cellular automata with balanced rules. In *International Conference on Intelligent Computing*, pages 445–455. Springer, 2017.

[39] Yang Liu, Xiaojun Tong, and Jing Ma. Image encryption algorithm based on hyper-chaotic system and dynamic s-box. *Multimedia Tools and Applications*, 75(13) :7739–7759, 2016.

[40] Gitika Maity and Jaydeb Bhaumik. New hybrid ca rule sets for cryptographic design. In *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*, pages 1–5. IEEE, 2015.

[41] Georges Marsaglia. Diehard test suite. *Online : http ://www. stat. fsu. edu/pub/diehard/. Laste visited*, 8(01) :2014, 1998.

[42] Michael I Mazurkov and Artem V Sokolov. Algorithm for synthesis of efficient s-boxes based on cellular automata. *Radioelectronics and Communications Systems*, 59(5) :212–220, 2016.

[43] Rajat Kumar Mehta and Rajneesh Rani. Pattern generation and symmetric key block ciphering using cellular automata. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2692–2695. IEEE, 2016.

[44] Willi Meier. On the security of the idea block cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 371–385. Springer, 1993.

[45] Willi Meier and Othmar Staffelbach. Analysis of pseudo random sequences generated by cellular automata. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 186–199. Springer, 1991.

[46] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. Block ciphers. In *Handbook of applied cryptography*, pages 223–282. CRC press, 1996.

[47] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[48] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. Hash functions and data integrity. In *Handbook of applied cryptography*, pages 326–327. CRC press, 1996.

[49] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. Number-theoretic reference problems. In *Handbook of applied cryptography*, page 113. CRC press, 1996.

[50] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. Stream ciphers. In *Handbook of applied cryptography*, pages 191–216. CRC press, 1996.

[51] S Nandi, BK Kar, and P Pal Chaudhuri. Theory and applications of cellular automata in cryptography. *IEEE Transactions on computers*, 43(12) :1346–1357, 1994.

[52] Abolfazl Yaghouti Niyat, Mohammad Hossein Moattar, and Masood Niazi Torshiz. Color image encryption based on hybrid hyper-chaotic system and cellular automata. *Optics and Lasers in Engineering*, 90 :225–237, 2017.

[53] Vincent Rijmen and Joan Daemen. Advanced encryption standard. *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pages 19–22, 2001.

[54] Vincent Rijmen, Bart Preneel, and Erik De Win. On weaknesses of non–surjective round functions. *Designs, Codes and Cryptography*, 12(3) :253–266, 1997.

[55] Satyabrata Roy, Subrata Nandi, Jayanti Dansana, and Prasant Kumar Pattnaik. Application of cellular automata in symmetric key cryptography. In *Communications and Signal Processing (ICCSP), 2014 International Conference on*, pages 572–576. IEEE, 2014.

[56] Satyabrata Roy, Umashankar Rawat, Harsh Ajay Sareen, and Sanjeet Kumar Nayak. Ieca : an efficient iot friendly image encryption technique using programmable cellular automata. *Journal of Ambient Intelligence and Humanized Computing*, 11(11) :5083–5102, 2020.

[57] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, and Elaine Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Booz-allen and hamilton inc mclean va, 2001.

[58] Anas Sadak, Bouchra Echandouri, Fatima Ezzahra Ziani, Charifa Hanin, and Fouzia Omary. Lcahash-1. 1 : a new design of the lcahash system for iot. *International Journal of Advanced Computer Science and Applications*, 10(11), 2019.

[59] Kazuo Sakiyama, Yu Sasaki, and Yang Li. *Security of block ciphers : from algorithm design to hardware implementation*. John Wiley & Sons, 2016.

[60] Joel L Schiff. *Cellular automata : a discrete view of the world*, volume 45. John Wiley & Sons, 2011.

[61] Shravani Shahapure, Virendra Sule, and RD Daruwala. Variation and security enhancement of block ciphers by embedding. *Journal of Discrete Mathematical Sciences and Cryptography*, 22(2) :151–160, 2019.

[62] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4) :656–715, 1949.

[63] Nigel P Smart. *Cryptography made simple*. Springer, 2016.

[64] Nigel P Smart. Hash functions, message authentication codes and key derivation functions. In *Cryptography made simple*, pages 271–294. Springer, 2016.

[65] Nigel P Smart. Modern stream cipher. In *Cryptography made simple*, page 238. Springer, 2016.

[66] Nigel P Smart. Modern stream cipher. In *Cryptography made simple*, pages 241–269. Springer, 2016.

[67] Nigel P Smart. Modular arithmetic, groups, finite fields and probability. In *Cryptography Made Simple*, pages 3–25. Springer, 2016.

[68] Nigel P Smart. Public key encryption and signature algorithms. In *Cryptography made simple*, pages 315–324. Springer, 2016.

[69] Nigel P Smart. The "naive" rsa algorithm. In *Cryptography made simple*, pages 295–310. Springer, 2016.

[70] Data Encryption Standard et al. Federal information processing standards publication 46. *National Bureau of Standards, US Department of Commerce*, 23, 1977.

[71] Miroslaw Szaban and Franciszek Seredynski. Ca-based generator of s-boxes for cryptography use. In *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, pages 1–8. IEEE, 2010.

[72] Gael Thomas. *Design et Analyse de sécurité pour les constructions en cryptographie symétrique*. PhD thesis, Limoges, 2015.

[73] Selman Uguz, Uğur Sahin, Hasan Akin, and Irfan Siap. Self-replicating patterns in 2d linear cellular automata. *International Journal of Bifurcation and Chaos*, 24(01) :1430002, 2014.

[74] John Von Neumann and Arthur Walter Burks. *Theory of self-reproducing automata*. University of Illinois Press Urbana, 1996.

[75] Yong Wang, Yi Zhao, Qing Zhou, and Zehui Lin. Image encryption using partitioned cellular automata. *Neurocomputing*, 275 :1318–1332, 2018.

[76] Stephen Wolfram. Cryptography with cellular automata. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 429–432. Springer, 1985.

[77] Stephen Wolfram. Random sequence generation by cellular automata. *Advances in applied mathematics*, 7(2) :123–169, 1986.

[78] Stephen Wolfram. *A new kind of science*, volume 5. Wolfram media Champaign, IL, 2002.

[79] Yousheng Zhou, Feng Wang, Fei Tang, and Xiaojun Wang. Cellular automata based secure distributed storage scheme with integrity proof. *Computers & Electrical Engineering*, 59 :291–304, 2017.

[80] Bo Zhu. *Analysis and Design of Authentication and Encryption Algorithms for Secure Cloud Systems*. PhD thesis, University of Waterloo, 2015.

[81] Fatima Ezzahra Ziani, Anas Sadak, Charifa Hanin, Bouchra Echandouri, and Fouzia Omary. Ca-pcs : A cellular automata based partition ciphering system. *International Journal of Advanced Computer Science and Applications*, 2020.

Résumé

Pouvant être interceptée par des entités malveillantes, la transmission de données sensibles via un canal non sécurisé est très risquée. Pour remédier à cette faiblesse, plusieurs solutions basées sur la cryptographie ont été développées en vue d'atteindre un bon niveau de sécurité. Les cryptosystèmes interviennent dans ce sens comme une excellente alternative pour assurer la confidentialité des données. Dans la présente thèse, nous nous sommes d'abord intéressés de près à la confidentialité puis à l'authenticité et ce, en premier lieu, à travers BC-CaACO qui représente un nouveau chiffrement par blocs évoluant des ACs unidimensionnels réversibles et irréversibles avec une S-boîte basée sur l'optimisation par colonie de fourmis (ACO). Ce chiffrement est doté de bonnes propriétés de confusion et de diffusion. En deuxième lieu, nous avons présenté un nouveau chiffrement par blocs SPF-CA combinant les deux schémas : SPN et Feistel basé sur les ACs. Ce dernier utilise le générateur PSOCA lors de la génération des sous-clés et fait appel à des S-boîtes dynamiques robustes dépendantes de la clé. Comme résultat, notre chiffrement fait preuve d'une grande efficacité et résistance dépassant les systèmes de chiffrement bien connus. Dans un troisième lieu, SPF-CA 1.2, une variante de SPF-CA, vient explorer le caractère aléatoire des ACs, leurs dégré algébrique élevé ainsi que leur non-linéarité maximale. Durant le processus de génération des sous-clés et au niveau de la fonction du tour de Feistel, SPF-CA 1.2 évolue un ensemble de règle linéaires et non linéaires bien choisi. L'efficacité et la robustesse de ce système ont été vérifiées à travers une analyse de sécurité et ce en réalisant moins de tours. En quatrième et dernier lieu, nous avons introduit CMAC-SPF-CA 1.2, une extension intégrée dans les technologies IoT se basant sur SPF-CA 1.2, un code d'authentification de message démontré résistant aux attaques de falsification et de récupération de clés.

Mots-clefs (5) : Sécurité de l'Information, Cryptographie Symétrique, Automates Cellulaires, Confidentialité, Authenticité.

Abstract

Being able to be intercepted by malicious entities, the transmission of sensitive data via an insecure channel is highly risky. To remedy this weakness, several solutions based on cryptography have been developed in order to achieve a good level of security. Cryptosystems intervene in this sense as an excellent alternative to ensure data confidentiality. In this thesis, we were closely interested in confidentiality and authenticity and this, first of all, through BC-CaACO which represents a new block cipher using one-dimensional reversible and irreversible CA with an S-box based on ant colony optimization (ACO). This encryption has good confusion and diffusion properties. Secondly, we presented a new SPF-CA block cipher combining the two schemes: SPN and Feistel based on CAs. The latter uses the PSOCA generator during the generation of the sub-keys and uses robust dynamic S-boxes dependent on the key. As a result, our encryption high efficiency and resilience goes beyond well-known encryption systems. In a third place, SPF-CA 1.2, a variant of SPF-CA, explores the randomness of CAs, their high algebraic degree as well as their maximum non-linearity. During the process of generating the sub-keys and at the level of the Feistel turn function, SPF-CA 1.2 evolves a well-chosen set of linear and non-linear rules. The efficiency and robustness of this system have been verified through a safety analysis and this by performing fewer rounds. Fourthly and at las, we introduced CMAC-SPF-CA 1.2, an extension built into IoT technologies based on SPF-CA 1.2, a message authentication code proven to be resistant to forgery and key recovery attacks.

Key Words (5) : Information Security, Symmetric Cryptography, Cellular Automata, Confidentiality, Authenticity.