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 : 3079

THESE

Pour l'obtention du ***DOCTORAT***

Structure de Recherche : Laboratoire de Recherches Informatiques
Discipline : Informatique
Spécialité : Informatique

Présentée et soutenue le *15 MARS 2018*

Par

Bouchra ECHANDOURI

Contributions à la Sécurité Informatique par la Conception de Nouveaux Codes d'Authentification de Message

JURY

Fatima-Zahra BELOUADHA	*PES, Université Mohammed V, Rabat. Ecole Mohammadia d'Ingénieurs.*	*Président*
Fouzia OMARY	*PES, Université Mohammed V, Rabat. Faculté des sciences.*	*Directeur de Thèse*
Hafssa BENABOUD	*PH, Université Mohammed V, Rabat. Faculté des sciences.*	*Rapporteur*
AbdelAziz MOULOUDI	*PES, Université Ibn Tofail, Kénitra. Faculté des sciences.*	*Rapporteur*
Abderrahim TRAGHA	*PES, Université Hassan II, Casablanca. Faculté des sciences Ben M'Sik.*	*Examinateur*

Année Universitaire : 2017/2018

Remerciements

Tout d'abord je remercie Dieu de m'avoir donné l'inspiration de mener des travaux de recherche en sécurité informatique.
Ces travaux de recherche, réalisés dans le Laboratoire de Recherche en Informatique (LRI) à la Faculté des Sciences de l'Univérsité Mohammed V de Rabat et présentés dans ce mémoire de thèse, sont le fruit de la collaboration de plusieurs personnes, sans qui ce travail n'aurait pu aboutir. A la tête de ces personnes, je tiens à remercier particulièrement ma directrice de thèse et encadrante, Professeur Fouzia OMARY, de m'avoir fait confiance en me proposant cette thèse, de m'avoir guidé tout au long de mes travaux de recherches en doctorat et de m'avoir aidé à améliorer mes capacités de présentation et de rédaction. J'ai beaucoup appris de sa méthodologie de recherche, de son raisonnement scientifique et de sa précision. Merci Professeur pour vos précieux conseils, votre temps et votre soutien moral aussi.

Je tiens aussi à remercier vivement chacun des membres du jury :
Madame Fatima Zahra BELOUADHA, Professeur de l'Enseignement Supérieur à l'Ecole Mohammadia d'Ingénieurs, Université Mohammed V, Rabat. Je tiens à vous exprimer toute ma gratitude pour l'honneur que vous me faites en acceptant de présider le jury de ma soutenance. Je tiens à vous exprimer ma reconnaissance et mon respect.
Monsieur AbdelAziz MOULOUDI, Professeur de l'Enseignement Supérieur à la Faculté des Sciences, Université Ibn Tofail, Kénitra. Je vous remercier d'avoir accepté de faire partie de ce jury en tant que rapporteur et examinateur de cette thèse. Je tiens à vous exprimer ma très respectueuse considération.
Monsieur Abderrahim TRAGHA, Professeur de l'Enseignement Supérieur à la Faculté des Sciences Ben M'Sik, Université Hassan II, Casablanca. Je vous remercie, d'avoir accepté de faire partie de ce jury en tant que rapporteur et examinateur de cette thèse. Je vous remercie vivement pour l'intérêt que vous avez porté à ce travail.

Madame Hafssa BENABOUD, Professeure Habilitée à la Faculté des Sciences, Université Mohammed V, Rabat. Je vous remercie, vous aussi, d'avoir accepté de faire partie de ce jury en tant que rapporteur et examinateur de cette thèse. Je tiens à vous exprimer ma profonde reconnaissance.

Je tiens à exprimer ma gratitude profonde à mon père et à ma mère de m'avoir soutenue tout au long de mes études, je ne saurais jamais être reconnaissante pour toute leurs affections, leurs amours infini et leurs soutiens. Merci à vous. Je remercie aussi mon frère, ma sœur, ainsi que toute ma famille pour leurs encouragements.

Je voudrais exprimer ma reconnaissance envers tous mes amies et amis qui m'ont apporté leur soutien moral tout au long de mon doctorat.

Résumé

La transmission des données, sur un canal non sécurisée, est risquée. Ainsi, la nécessité de concevoir des solutions cryptographiques robustes assurant l'intégrité et l'authentification des données est d'ordre primordial. Afin d'établir une communication de données sécurisées, de nombreuses techniques de sécurité des données ont été développées pour en assurer la confidentialité, l'intégrité, l'authentification et la non-répudiation. En particulier, le Code d'Authentification de Message (MAC) est l'un des mécanismes les plus fiables garantissant l'intégrité et l'authentification. En combinant une clé secrète avec le message pour générer une valeur numérique, appelée Tag, un MAC permet au récepteur légitime du bon Tag de vérifier son authenticité, en testant si aucune altération n'a affecté le message pendant sa transmission. Pour concevoir un MAC, il existe deux principales méthodes, la première utilisant un algorithme de chiffrement symétrique, à savoir CMAC et CBC-MAC. La seconde utilisant une fonction de hachage combinée à une clé secrète.

Cette thèse consiste en la conception et la réalisation de nouveaux Codes d'Authentification de Message. En fait, trois MACs ont été proposés ; le premier nommé SEC-CMAC, ayant comme outil de base l'algorithme de chiffrement symétrique nommé Symmetrical Evolutionist Ciphering (SEC-binaire). Le second nommé Keyed-CAHASH rapide et le troisième nommé LCAHASH-MAC léger. Ces deux derniers se basent sur des fonctions de hachage utilisant les Automates Cellulaires. Des analyses et des expérimentations ont été effectuées afin de prouver la performance et résistance aux attaques auxquelles les MACs en sont sujets.

Mots-clés : Sécurité Informatique, Authentification, Code d'Authentification de Message (MAC), Chiffrement Evolutioniste Symétrique (SEC), Fonctions de Hachage, Automates Cellulaires (AC), Attaques de falsification de MAC, Attaque de récupération de clés

Abstract

Private and sensitive data transmission over arbitrary canal is considered very risky. It became one of the flaws of computer security that needs to be handled. Hence, the need for designing robust cryptographic solutions ensuring both data integrity and authentication is crucial. In order to establish a secure data communication, many data security techniques were developed to ensure privacy, integrity and authentication. Therefore, Message Authentication Code (MAC) is one of the most provably secure algorithms that ensure integrity and authentication. Using a secret key on the message to generate a checksum, also called a Tag, allows the legitimate receiver of the right Tag to verify its authenticity and to check if the message was altered during transmission. Particularly, to conceive a MAC there are many ways, one using a symmetric encryption algorithm, especially Block Cipher-based Message Authentication Code (CMAC) and Cipher Block Chaining Message Authentication Code (CBC-MAC). The second may be derived from Hash Functions using a secret key. This thesis is concerned with the design of new Message Authentication Codes. A first solution, termed (SEC-CMAC), is based on a previously developed Symmetric Ciphering algorithm named Symmetrical Evolutionist Ciphering (SEC). Two additional solutions are proposed, which are fast and lightweight and based on respectively Keyed Hash Function and Cellular Automata ; they are termed Keyed-CAHASH and LCAHASH-MAC. These proposed MACs provide high efficiency and robustness and fulfill security against MAC forgery and key recovery attacks compared to well-known MACs standards. Further, the results obtained show that our proposed MACs satisfy data integrity and authentication requirements.

Keywords : Information Security, Authentication, Message Authentication Code (MAC), Symmetrical Evolutionist Ciphering (SEC), Hash Functions, Cellular Automata (CA), MAC forgery attack, key recovery attack

Table des matières

Liste des abréviations

AES Advanced Encryption Standard 2

AES-CMAC Code d'Authentification de Message basé sur AES 2

APN Almost Perfect Nonlinear 23

AC Automates Cellulaires 38

CBC-MAC Code d'Authentification de Message basé sur un système de Chiffrement par Blocs Chaînés 20

Ch-Initial Chromosome Initial 32

Ch-Final Chromosome Final 32

CMAC Code d'Authentification de Message basé sur un système de Chiffrement 2

DES Data Encryption Standard 2

DES-CMAC Code d'Authentification de Message basé sur DES 2

IDEA International Data Encryption Algorithm 10

Liste des figures

LISTE DES TABLEAUX

Liste des algorithmes

Introduction Générale

1 Contexte Général

Avec l'avancement des technologies de l'information, la communication des données, à travers un canal arbitraire, est très dangereuse, puisque n'importe quelle entité non autorisée peut l'intercepter. La discipline de base contribuant à la sécurisation de cette communication est la cryptologie. Cette dernière offre des outils robustes assurant la confidentialité, l'intégrité, l'authentification et la non répudiation.

Parmi ces propriétés, il y en a qui sont très antiques comme la confidentialité et l'authentification. Par contre, les processus les garantissant évoluent, au fil des années, en parallèle avec l'évolution de la technologie. En effet, la confidentialité date de 3000 ans av J.-C.. Quant à l'authentification est apparue à l'époque d'«Ali Baba» et les quarante voleurs. «Ali Baba» utilisait une phrase « Sesame, ouvres Toi» comme un mot de passe pour que la porte de la caverne magique s'ouvre (authentification par mot de passe).

L'authentification joue un rôle important dans le contrôle d'accès à un système, en limitant cet accès à des entités autorisées. En contraste avec l'identification qui permet de prouver l'identité, l'authentification est le processus de vérification de l'identité réclamée d'une entité. Sa forme la plus connue, dans des cas simple, est l'authentification par mot de passe.

En général, il y existe trois manières différentes d'authentification [2] :

- Authentification par quelque chose qu'on sait : cette authentification se fait par la saisie à l'aide d'un clavier d'une chaîne de caractères que le système connait déjà et qu'on mémorise (un mot de passe, PIN, schéma ou une phrase, etc.)
- Authentification par quelque chose qu'on possède : cette authentification se fait à l'aide d'un outil qu'on possède qui interagit avec un lecteur et qu'on doit utiliser lors de l'authentification (une carte à puce, une clé USB ou un Tag, etc.)
- Authentification par ce qu'on est (biométrie) : cette authentification se fait en utilisant une partie de notre corps ou l'une de nos caractéristiques physiques (Empreintes digitales,

impression de voix, modèle iris, forme de visage ou la forme de la main, etc.).

L'appel à chaque type d'authentification dépend du contexte sollicitant le besoin. En effet, l'authentification par outils biométriques est très demandé dans des contextes de sureté nationale tels que les aéroports, les frontières,etc.

En particulier, le Code d'Authentification de Message (MAC) a été décrit comme l'une des solutions cryptographiques les plus robustes assurant l'intégrité et l'authentification des données. En faisant intervenir une clé secrète, ces MAC produisent une valeur numérique de taille fixe, appelée Tag, grâce à un algorithme de génération de MAC. Le Tag permet à son récepteur de vérifier l'intégrité et l'authentification du message reçu et de s'assurer qu'aucune altération n'est survenue lors de la transmission. La vérification se fait à travers un second algorithme, utilisant la même clé secrète et le message reçu [3]. En 1996, Bellare et al. [4] ont présenté un standard, nomé HMAC qui est une fonction de Hachage à clé. Ce standard proposé obtient presque toute sa sécurité à partir de la fonction de Hachage utilisée. La robustesse d'un code d'authentification de message repose généralement sur la robustesse de la fonction de Hachage utilisé dans son processus. Cependant, la plupart des fonctions de hachages connues sont déjà attaquées et cassées. De plus, dans [5], les auteurs ont introduit un schéma d'authentification léger basé sur les Automates Cellulaires pour plusieurs utilisateurs dans le contexte du Cloud Computing. Cependant, ce système n'assure pas l'intégrité. En 2013, dans [6], les auteurs ont décrit l'implémentation d'un autre HMAC se basant sur la fonction de Hachage légère PHOTON [7]. Néanmoins, sa mise en œuvre n'est pas pratique pour les ressources à grande vitesse avec une puissance de calcul restreinte [6].

Dans sa recommandation pour le Code d'Authentification de Message basé sur un système de Chiffrement (CMAC), le National Institute of Standards and Technology (NIST) indique que le Code d'Authentification de Message est obtenu en utilisant trois clés. De cette façon, la réduction séquentielle du texte en clair et son chiffrement sont effectués simultanément [8].

L'algorithme du NIST, CMAC utilise l'Advanced Encryption Standard (AES) comme un système de chiffrement pour le Code d'Authentification de Message basé sur AES (AES-CMAC). Bien que la sécurité de l'AES-CMAC repose sur la sécurité de l'algorithme AES, la clé secrète doit être néanmoins sécurisée. Si elle est partagée ou modifiée d'une manière inappropriée, l'authentification et l'intégrité ne sont plus garanties, comme démontré dans [9].

Le Code d'Authentification de Message basé sur DES (DES-CMAC) a une complexité similaire à celle de AES-CMAC. Il consiste à utiliser un cycle unique de Data Encryption Standard (DES) et offre une sécurité élevée contre les attaques de récupération de clé [10]. Cependant dans [11],

les auteurs ont présenté plusieurs nouvelles techniques de cryptanalyse réussies sur ces MAC.

2 Contributions

Cette thèse propose la conception et la réalisation de trois nouveaux Codes d'Authentification de Message MAC permettant d'enrichir et de remédier à la carence, en mécanismes robustes, rapides et légers 1, dans le cadre d'authentification.
Ainsi mes contributions aux Codes d'Authentification de Message ont été groupées en deux catégories : une basée sur un système de chiffrement et deux basées sur une fonction de Hachage. Comparées aux MAC existants, ces mécanismes suggérés se sont tous avérés robustes et répondent aux exigences de sécurité en termes d'intégrité et d'authentification. Ils ont été prouvés résistants aux attaques des MACs, à savoir la récupération des clés et la falsification du Tag.

Contribution aux Codes d'Authentification de Message basés sur un système de chiffrement (12) : Parmi toutes les contributions aux Codes d'Authentification de Message, basés sur un système de chiffrement, cette contribution est l'une des premières utilisant un système de chiffrement évolutionniste, à savoir Symmetrical Evolutionist Ciphering (SEC) [13]. Ce dernier est l'un des premiers systèmes de chiffrement symétriques à utiliser un algorithme évolutionniste pour chiffrer les données. Pour chaque session, SEC a une grande capacité à changer la fréquence et les positions des caractères dans le message clair et aussi de générer plusieurs clés différentes. La robustesse de SEC-CMAC réside dans l'utilisation du chiffrement évolutionniste, ce qui lui donne un caractère aléatoire et le rend résistant aux attaques de falsification du Tag. De plus la diversification des clés de chiffrement renforce la résistance aux attaques de récupération des clés.

Contribution aux Codes d'Authentification de Message basés sur une fonction de Hachage (14) : Cette contribution aux Codes d'Authentification de Message est nommé Keyed-CAHASH. Dans son processus de génération de Tag, cette contribution utilise une nouvelle fonction de Hachage à clé ainsi que PSOCA [1], un nouveau générateur de nombre pseudo aléatoire. Cette nouvelle fonction de Hachage à clé et PSOCA sont tous les deux basés sur les Automates Cellulaires non uniforme. L'utilisation des Automates Cellulaires a fait de Keyed-CAHASH un MAC rapide et robuste par rapport aux MAC basés sur une fonction de Hachage bien connus. En comparant ce Code d'Authentification de Message avec d'autres existants, l'utilisation des Automates Cellulaires, dans leurs processus, offre une simplicité, une homogénéité

[15]. Ces caractéristiques ont permis aussi une facilité d'implémentation logicielle et matérielle et une imprévisibilité, ce qui rend Keyed-CAHASH résistante aux attaques de falsification du Tag. Aussi, l'utilisation du processus de génération de nombre pseudo aléatoire PSOCA, prouvé imprévisible, fait de Keyed-CAHASH un MAC résistante aux attaques de récupération de clés.

Contribution aux Codes d'Authentification de Message basés sur une fonction de Hachage légère (16) : LCAHASH-MAC est une contribution aux Codes d'Authentification de Message légers, basée sur une fonction de Hachage légère. Lors de la génération du Tag, cette contribution utilise une nouvelle fonction de Hachage légère L-CAHASH [17] ainsi que PSOCA [1]. Cette nouvelle fonction de Hachage légère est dédiée au contexte des Radio-Frequency Identification (RFID).

Ces derniers souffrent d'un grand nombre de menaces, parmi lesquelles celles liées à l'authentification. Cependant, la plupart des mécanismes d'authentification classiques, déjà existants, sont inadaptables pour leur contexte. Ils sont connus par leur taille de mémoire restreinte et leur puissance de calcul réduite. L'utilisation des Automates Cellulaires a fait de cette contribution un MAC léger, rapide et robuste comparé aux MAC bien connus.

Tableau 1 – Comparaison des trois contributions aux MACs connus

	rapide	léger	diversité de la clé	comportement aléatoire
(SEC-CMAC)	-	-	√√√	√√√
(Keyed-CAHASH)	√	-	√√√	√√√
(LCAHASH-MAC)	√	√	√√√	√√√
KMAC(KeccakMAC-512)	√	√	-	-
AES-MAC	√	-	-	-

3 Organisation du Mémoire de Thèse

Ce mémoire se compose d'une introduction générale, de cinq chapitres et d'une conclusion et perspectives.

- **Chapitre 1 : les principaux objectifs de la sécurité de l'information :**

  Dans ce premier chapitre, je présente, en général, les objectifs basiques et essentiels pour avoir une sécurité de l'information. Je donne les définitions de chaque de ces objectifs, ainsi que les notations, tout en détaillant les concepts fondamentaux des différents outils cryptographiques les assurant. Je cite aussi un nombre de travaux déjà existants liés aux Codes d'Authentification de Message, pour ceux basés sur un système de chiffrement et ceux basés sur une fonction de Hachage.

- **Chapitre 2 : Contribution aux Codes d'Authentification de Message basés sur les Systèmes de Chiffrement**

  Dans ce chapitre je décris l'algorithme SEC-CMAC, le nouveau Code d'Authentification de Message basé sur le système de chiffrement déjà proposé. Je donne aussi une présentation générale de SEC l'un des premiers systèmes de chiffrement symétriques à utiliser un algorithme évolutionniste. Je présente en détail les algorithmes ; notamment l'algorithme de génération et de vérification de SEC-CMAC.

- **Chapitre 3 : Contribution aux Codes d'Authentification de Message basés sur les Fonctions de Hachage :**

  Dans ce chapitre je décris l'algorithme Keyed-CAHASH, le nouveau Code d'Authentification de Message rapide basé sur une fonction de Hachage utilisant les Automates Cellulaires. je présente l'algorithme de génération de clé, l'algorithme de génération de Tag ainsi que celui de vérification.

- **Chapitre 4 : Contribution aux Codes d'Authentification de Message basés sur les Fonctions de Hachage légères :**

  Dans ce chapitre je décris l'algorithme LCAHASH-MAC, le nouveau Code d'Authentification de Message léger basé lui aussi sur une fonction de Hachage utilisant les Automates Cellulaires. je présente l'algorithme de génération de clé, l'algorithme de génération de Tag et l'algorithme de vérification.

- **Chapitre 5 : Analyse de Sécurité des Codes d'Authentification de Message proposés et Étude de leur Performance :**

  Dans ce chapitre je donne une présentation de l'étude et de l'évaluation de la résistance de chaque conception proposée aux attaques de récupération des clés et de falsification. Je donne aussi les résultats des expérimentations et tests effectués. A la fin, je présente leurs performances.

- **Conclusion et perspectives :**

  Cette conclusion conclut la thèse en rappelant les différentes contributions réalisées tout au long du travail de recherche, et met en relief les perspectives des travaux futures.

Chapitre

1 Principaux Objectifs de la Sécurité de l'Information

Sommaire

1.1 Introduction

Au cours des deux dernières décennies, le monde a été marqué par l'essor très important des systèmes de communication et des échanges électroniques de l'information. Cette information qui concerne, entre autres, les fichiers, les bases de données, les données nominatives des personnes physiques telles que les fiches des salariés, les fiches médicales, les fiches industrielles, les numéros

de cartes de crédit et les processus de fabrication, etc. fait la force de l'existence unique de toute entreprise. Cependant, toute manipulation frauduleuse de cette information peut engendrer une perte colossale dans le chiffre d'affaire de l'entreprise ou même nuire au pays. Il est donc primordial de l'intérêt de l'entreprise d'assurer la sécurité et la protection de leurs informations qu'ils détiennent. Aussi, un acte de vol ou une destruction du matériel informatique, du fait d'une catastrophe naturelle ou d'un acte humain peut générer une perte de données. Certes un risque zéro n'existe pas, néanmoins il reste possible de protéger ces données en se prémunissant des solutions matérielles et logicielles adéquates. La mise en place d'onduleurs permet de prendre le relais en cas de coupure d'électricité, de protéger tout matériel informatique des surtensions et de continuer de fonctionner. Encore, pour éviter en amont les intrusions, la mise en place de pare-feu, de systèmes de détection d'intrusion, de système de prévention d'intrusion et de proxies peuvent être une solution au renforcement de la sécurité informatique. En outre, la sécurité logique fait appel à des mécanismes de sécurité. Cette dernière repose sur la mise en œuvre d'un système de contrôle d'accès s'appuyant sur un service d'identification, d'authentification et d'autorisation. De même, les stratégies de sauvegarde, une gestion efficace des mots de passe, des access lists et des procédures d'authentification aident à la continuité des services en cas de sinistres. Les antivirus et les anti-malwares installés sur les ordinateurs et régulièrement mis à jour permettent une protection maximale. A cet égard, la prise de conscience en sécurité de l'information a engagé des réflexions sur la nécessité du développement des primitives cryptographiques dans le but d'assurer cette sécurité. Nous pouvons distinguer quatre principaux objectifs de la sécurité informatique, à savoir [2] :

- la confidentialité : C'est un moyen de garder l'information inintelligible pour toute entité malveillante, non autorisée.
- L'intégrité : C'est un moyen de garder l'information protégée contre les altérations malveillantes.
- L'authentification : C'est un moyen de vérifier sa provenance revendiquée.
- La non-répudiation : C'est un moyen assurant que personne ne peut nier le fait d'avoir commis un acte se rapportant à une information.

Aussi, une information a besoin d'être toujours accessible, à ses utilisateurs autorisés, à n'importe quel moment, pour permettre ainsi à un système d'information de mener à terme un bon fonctionnement[18]. En addition, il faut que les ressources et les services soient opérationnels et non pas saturés puisque le but des attaques sur la disponibilité est de rendre le système inutilisable, hors service et que l'information soit inaccessible.

Dans ce chapitre, nous présentons des systèmes cryptographiques, notamment les crypto-systèmes à clé privée, les Fonctions de Hachage et les Codes d'Authentification de Message, garantissant les principaux objectifs de la sécurité de l'information, respectivement la confidentialité, l'intégrité des données et l'authentification.

1.2 Confidentialité

Pendant la transmission de données sensibles, à travers un canal arbitraire, la confidentialité garantit que ces données demeurent compréhensibles et intelligibles uniquement par les entités légitimes [2]. Pour ce faire, cette confidentialité est obtenue principalement par le biais de deux types de crypto-systèmes, à savoir les crypto-systèmes symétriques ou à clés secrètes et les crypto-systèmes asymétriques ou à clés publiques. Les crypto-systèmes à clé privée utilisent une clé secrète symétrique, partagée entre les utilisateurs légitimes. Par contre, les crypto-systèmes asymétriques, à clé publique, utilisent une combinaison de clés privées et publiques pour chaque utilisateur légitime [2]. Généralement, les transformations qui entrent dans le processus des crypto-systèmes symétriques utilisent la même clé secrète pour le chiffrement et le déchiffrement. En particulier, il existe deux types de schémas, les chiffrements par bloc et les chiffrements par flux [2].

1.2.1 Chiffrement symétrique :

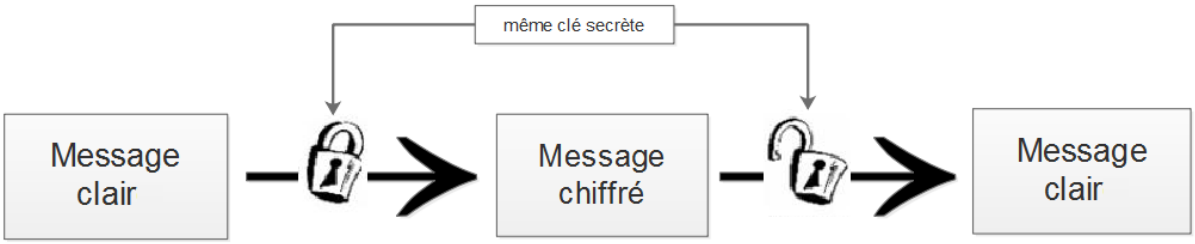

Figure 1.1 – Chiffrement symétrique.

Chiffrement par blocs : Spécifiquement, le chiffrement par bloc est un schéma de chiffrement qui prend en entrée un bloc de message en clair et une clé de longueur de bit fixe. Explicitement, c'est une fonction $E : \{0,1\}^K \times \{0,1\}^n \rightarrow \{0,1\}^n$ [19] qui prend une chaîne de k-bit (la clé secrète) et une chaîne de n-bits (un bloc de message) comme entrée et renvoie une chaîne de n-bits (un

bloc de chiffré) comme sortie (figure 1.2). Tout d'abord, dans ce mode de chiffrement le message est partitionné en blocs de chiffré de longueur fixe afin de fournir un chiffrement individuel de chaque bloc. Les algorithmes les plus connus de ce mode sont DES, AES, International Data Encryption Algorithm (IDEA), etc...

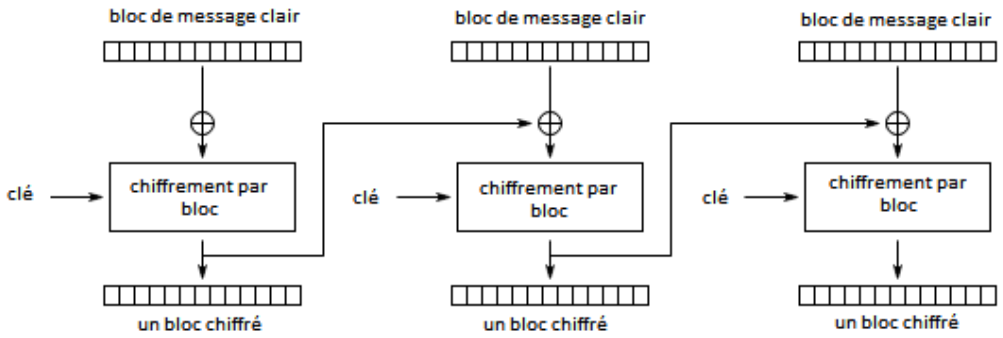

Figure 1.2 – Chiffrement par chainage de bloc.

Il existe trois types de chiffrement par bloc :

- Chiffrement par substitution : Dans le but de créer la confusion (i.e. création d'une dépendance complexe entre le clair, le chiffré et la clé, afin que l'attaquant ne puisse obtenir des informations sur la clé pour toute paire clair/chiffré qu'il possède.), une substitution sert à remplacer un groupe de symboles par un autre groupe de symboles [2].
- Chiffrement par transposition : Dans le but de créer la diffusion (i.e le changement minimal en texte clair, en entrée, doit avoir un effet important sur le chiffré, en sortie. Toute partie du chiffré dépend d'une partie du clair et d'une partie de la clé) une transposition consiste à mettre en désordre un groupe de symboles d'un message clair suivant des manières prédéfinies par la clé de chiffrement. Une suite de transpositions forme une permutation [2].
- Chiffrement par produit : Pour obtenir un chiffrement plus robuste, la combinaison du chiffrement par substitution et du chiffrement par transposition se fait et donne un chiffrement par produit.

Chiffrement par flux : L'autre mode de chiffrement symétrique, le chiffrement par flux (figure 1.3) est un chiffrement qui prend le message clair sous forme d'une suite finie de bits où chaque bit est chiffré séparément, en utilisant une clé pseudo aléatoire de taille fixe [20]. Ce type de chiffrement est connu par sa rapidité et son adaptation à la téléphonie mobile et aux réseaux.

Les bits du chiffré sont calculés comme suit : $c_i = m_i * k_i$, où m_i sont les bits finies du message

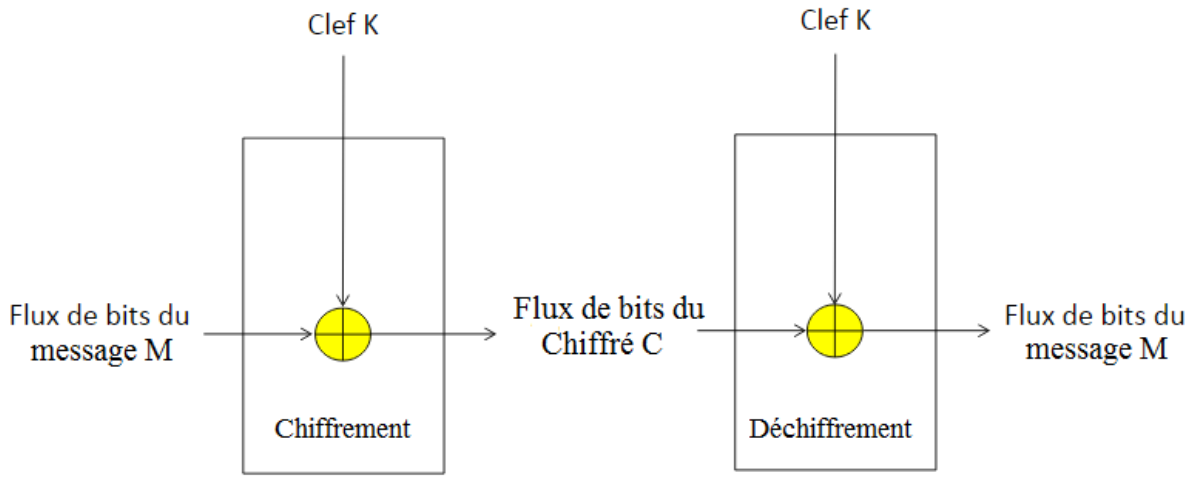

Figure 1.3 – Chiffrement par flux.

clair, k_i les bits de la clé secrète. Le déchiffrement utilise la même opération [20] : $m_i = c_i * k_i$. Les chiffrements par flux les plus connus sont entre autre le chiffrement de Vernam, l'algorithme Rivest Cipher 4 (RC4) [21], A5/1[22], E0[23] etc.

1.2.2 Chiffrement asymétrique

Introduit par Diffie et Hellman, le mode de chiffrement asymétrique(figure 1.4) [24] utilise deux clés, une clé privée et l'autre publique, en diffusant la clé publique servant au chiffrement et en gardant la clé privée servant au déchiffrement. Ce mode de chiffrement a résolu le problème de transmission des clés figurant dans le mode de chiffrement symétrique. En effet, les systèmes de chiffrement les plus connus de ce mode de chiffrement sont le Rivest, Shamir et Adleman (RSA) [25], El Gamal [26].

Figure 1.4 – Chiffrement asymétrique.

RSA : Présenté en 1977 par Rivest, Shamir et Adleman, le chiffrement RSA [25] a été le premier algorithme de chiffrement asymétrique. Basé sur la théorie des nombres premiers, la robustesse de ce dernier repose sur la difficulté à factoriser un grand nombre en entiers premiers. Or, cette factorisation nécessite une capacité très puissante de calcul. Pour la génération des clés, le RSA utilise deux très grands nombres premiers p et q. Soit n, φ respectivement les produits $p \times q$ et $(p-1) \times (q-1)$. Prenons un grand nombre premier avec φ choisi aléatoirement et e son inverse modulo φ, $d \equiv e-1$ [φ]. En particulier, la clé publique de chiffrement est (e, n) et la clé privée de déchiffrement est d. Pour chiffrer un message à l'aide de RSA, il est décomposé en une série d'entiers m de valeurs dans l'intervalle $[0, \ldots, n-1]$. Pour chaque m, nous calculons le chiffré constitué à partir de la succession d'entiers $c \equiv m^e[n]$. Pour déchiffrer, le chiffré reçu C est décomposé en séries d'entiers de valeurs dans l'intervalle $[0, \ldots, n-1]$. Pour chaque c, nous calculons $m \equiv c^d[n]$. Le message original est alors reconstitué à partir de la succession d'entiers m.

El Gamal : Présenté par Taher Elgamal en 1984, le chiffrement d'EL Gamal [26] est basé sur le problème du logarithme discret. Sa robustesse réside dans la difficulté de calculer le logarithme discret qui consiste à retrouver un entier λ tel que $h \equiv g^{\lambda}[p]$. Néanmoins, il est peu utilisé en pratique, car sa faiblesse réside dans la taille très grande du message chiffré. Pour la génération de clé, nous choisissons un nombre premier très grand p. Puis, nous choisissons la clé secrète s entre 0 et $p-2$. La clé privée est le triplet (p, q, r) tel que r est choisi entre 0 et $p-1$ et $r \equiv q^s[p]$. Pour le chiffrement d'un message $m(m < p)$, nous choisissons aléatoirement un entier k entre 0 et $p-1$, $(s^k \neq 1[p])$. Le couple (c_1, c_2) correspond au chiffré de m $\{c_1 \equiv q^k[p], c_2 \equiv mr^k[p]\}$. Pour le déchiffrement, le message d'origine $m_1 = c_2/(c_1^s)[p]$.

Courbes elliptique :

1.3 Intégrité

L'intégrité des données garantit que, durant la transmission, les données n'ont pas été modifiées malicieusement, notamment l'insertion, la suppression ou la substitution des bits. Pour ce faire, pour aider le récepteur à s'assurer que les données reçues sont identiques aux originales, il existe des mécanismes qui peuvent être établis par les crypto-systèmes à clé secrète ou à clé publique [2].

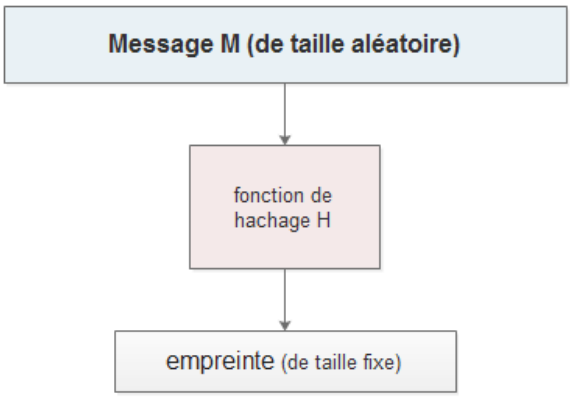

Figure 1.5 – Fonction de Hachage.

Dans ce contexte, divers algorithmes de hachage (figure 1.5) ont été développés. Ayant deux propriétés principales, à savoir la facilité de calcul et la compression (c'est-à-dire la transformation d'une entrée de longueur de bit arbitraire en une sortie de longueur de bit fixe), ces algorithmes de hachage génèrent une valeur, appelée empreinte, qui fournit une vérification de l'intégrité des données [2].

$$H() : \{0,1\}^{\star} \rightarrow \{0,1\}^{n}; M \rightarrow H(M) \tag{1.1}$$

Par définition, une fonction de Hachage ayant pour entrées m, m' et pour sorties h,h', est robuste si elle répond aux trois propriétés suivantes, notamment la résistance à la pré-image, à la seconde pré-image et à la collision [3].

- Résistance à la pré-image : il est facile de produire une valeur h pour un message m, mais il est difficile en un temps raisonnable, de calculer le message m pour cette valeur de hachage h ;
- Résistance à la seconde pré-image : il est difficile, en un temps raisonnable, de trouver un autre message d'entrée m' avec la même valeur de hachage en sortie $h = h'$;
- Résistance à la collision : il est difficile, en un temps raisonnable, de trouver n'importe quelle paire de messages m et m' ayant le même haché en sortie ($h = h'$).

En particulier, une fonction de Hachage à sens unique satisfait les propriétés de résistance à la pré-image et à la seconde pré-image [2]. Pour ce faire, il existe des techniques permettant d'évaluer la robustesse d'une fonction de Hachage.

1.3.1 Attaques génériques sur les fonctions de Hachage

Recherche de secondes pré-images et de pré-images : La recherche de secondes pré-images ou de pré-images se fait en utilisant la recherche probabiliste. Spécifiquement, étant donné une empreinte h, l'adversaire calcule plusieurs empreintes pour des messages aléatoires jusqu'à ce que l'une des empreintes soit égale à h. Chacune des empreintes calculées est égale à h avec une probabilité proche de 2^{-n} (avec n la taille de l'empreinte), donc le nombre moyen d'empreintes à calculer pour trouver une pré-image de h est de l'ordre de 2^n [3].

Recherche de collisions : Trouver une collision repose sur l'utilisation de l'attaque du paradoxe des anniversaires qui consiste à trouver au moins deux messages différents qui donnent deux empreintes égales. Le paradoxe des anniversaires est un raisonnement montrant que parmi un groupe d'au moins 23 individus pris au hasard, il y a au moins une chance sur deux pour que deux d'entre eux aient le même jour d'anniversaire. D'après cela, puisque le calcul des empreintes de messages aléatoires correspond à un tirage aléatoire sur l'espace des empreintes $\{0,1\}^n$, de taille 2^n, le nombre d'empreinte qu'il faut calculer pour trouver une collision est de l'ordre de $2^{n/2}$ [3].

Le tableau 1.1 présente, en général, que pour une recherche de pré-image, il est nécessaire que la fonction ne soit pas attaquable par une méthode nécessitant moins de 2^n calculs et pour une recherche de collisions moins de $2^{n/2}$ calculs.

Tableau 1.1 – Complexité des attaques génériques sur les fonctions de hachage

	complexité
Résistance à la seconde pré-image et de pré-image	$\bigcirc(2^n)$
Résistance à la collision	$\bigcirc(2^{n/2}$

Pour concevoir une fonction de Hachage, il existe de nombreuses façons, à savoir la construction d'éponge, la construction de Merkle-Damgard, etc...

1.3.2 Mode de construction

La construction de Merkle-Damgård : C'est une méthode itérative (figure 1.6) qui permet de créer une fonction de Hachage à partir d'une fonction de compression [3]. Les fonctions de hachage les plus connues de cette construction sont MD5, SHA-1, SHA-2,...

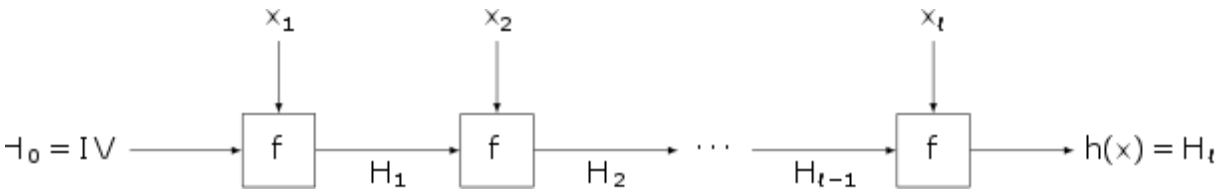

Figure 1.6 – Construction de Merkle-Damgård.

La construction éponge : C'est aussi une méthode itérative [3] (figure 1.7). Pour construire une fonction de Hachage à partir d'une fonction de permutation f sur un nombre fixe b de bits, b étant la largeur. La construction de l'éponge fonctionne sur un état de $b = r + c$ bits, où "r" est la valeur de Bitrate et "c" est la capacité. Tout d'abord, dans la phase d'initialisation tous les bits de l'état sont initialisés à zéro. Le message d'entrée est paddé et ensuite coupé en blocs de r bits. La construction de l'éponge se déroule ensuite en deux phases, la phase d'absorption suivie par celle de la compression.

- Dans la phase d'absorption, une opération «XOR» est effectuée sur les blocs de message d'entrée de r-bit, en alternance avec l'application de la fonction f. Lorsque tous les blocs de messages ont été traités, la construction éponge passe à la phase de compression.
- Dans la phase de compression, les premiers r bits de l'état sont renvoyés comme des blocs de sortie, entrelacés avec les applications de la fonction f. Le nombre de blocs de sortie est préalablement choisi par l'utilisateur.

Les derniers c bits de l'état ne sont jamais directement affectés par les blocs d'entrée et ne le sont jamais en sortie pendant la phase de compression.

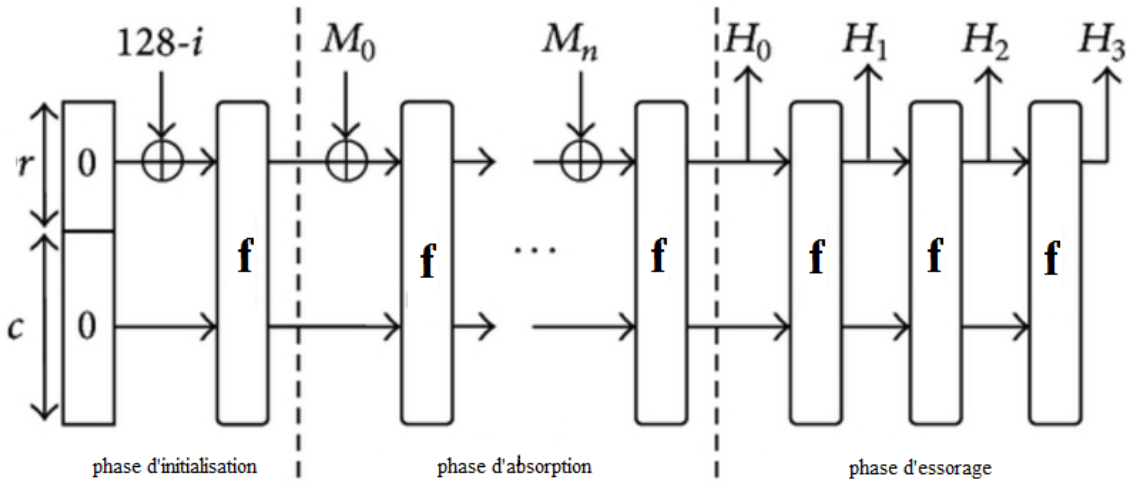

Figure 1.7 – Construction éponge.

La fonction de Hachage la plus connue de cette construction et la gagnante de la compétition SHA-3 est la fonction de Hachage Keccak [27],[3].

1.4 Non répudiation

Pendant l'échange numérique des données, la non-répudiation garantit que les entités communicantes, autorisées à partager l'information, ne peuvent nier leurs actes. Elle fournit une protection du déni de ces actes. A cette notion sont associées notamment l'imputabilité, la traçabilité et l'auditabilité. En particulier, l'imputabilité garantit la preuve d'une action par son enregistrement automatique, la traçabilité est l'ensemble des mécanismes permettant de retrouver les opérations réalisées sur les ressources de l'organisme et l'auditabilité garantit la capacité d'un système à évaluer et analyser les informations nécessaires, relatives à un événement, pour sa continuité. Pour ce faire, pour aider les entités communicantes à s'assurer que les événements ne peuvent êtres niés, prouver la validité et la fiabilité du contenu des informations ainsi que de l'identification du signataire, il existe des mécanismes qui peuvent être établis par les cryptosystèmes à clés publiques, notamment les signatures numériques [24]. La signature numérique permet de vérifier l'origine des données, de garantir l'identité de l'expéditeur et d'authentifier le signataire au même titre qu'une signature manuscrite. Cependant, cette dernière peut être facilement falsifiée, alors qu'une signature numérique est difficilement falsifiable en un temps raisonnable. Une signature numérique ne doit aucunement être confondue avec le chiffrement à clé publique, malgré que leur principal point commun soit l'utilisation d'une clé privée et d'une clé publique. En addition, elle est basée sur l'utilisation d'une fonction de Hachage et de la cryptographie asymétrique. D'après la norme de International Organization for Standardization (ISO) 7498-2, la signature numérique est définie comme étant des données jointes à un message, ou encore une transformation cryptographique d'un message. Pour signer un message, la génération de la signature numérique (figure1.8) consiste à calculer son empreinte, puis à la chiffrer en utilisant la clé privée. Après le résultat, qui est la signature (donnée numérique), est joint au message. Afin de vérifier la validité de la signature, il suffit de disposer de la clé publique. La vérification de la signature consiste à tout d'abord la déchiffrer en utilisant la clé publique du signataire puis à la comparer à celle calculée à partir du message reçu. Si les deux hachés sont identiques, alors la signature est valide.

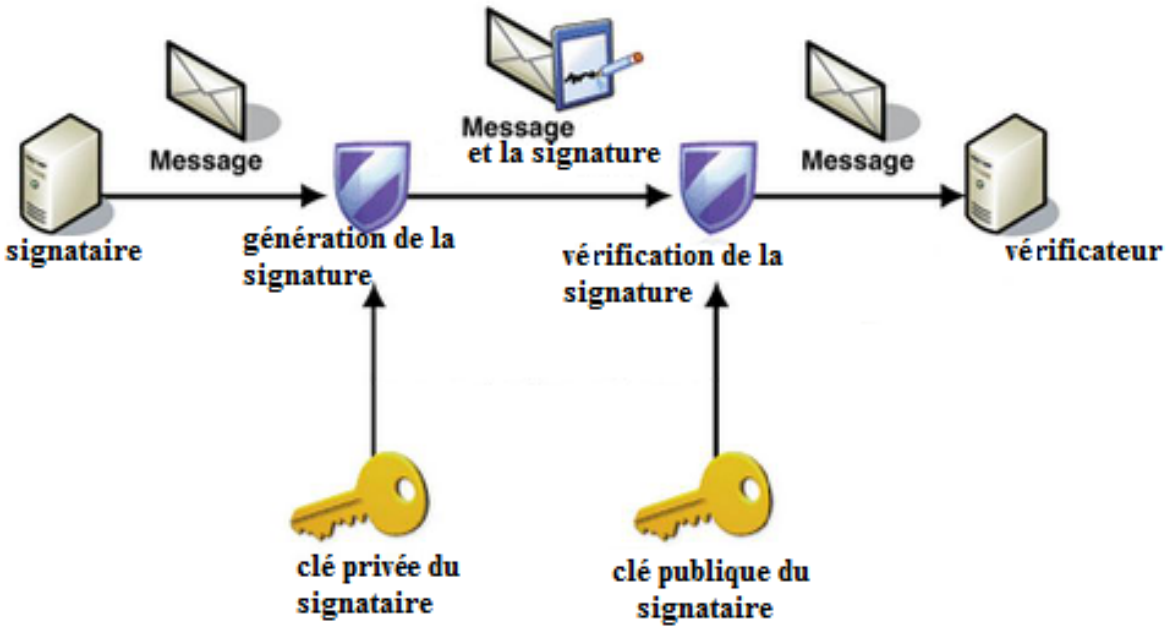

Figure 1.8 – Signature numérique.

1.4.1 Différents types de signatures numériques

Signature numérique simple : Dans le cas de cette signature, le vérificateur ne dispose que des informations essentielles à la validation de la signature, notamment le certificat du signataire et la valeur numérique de la signature. En addition, il n'existe aucune garantie relative à la validité des informations du certificat [28].

Signature numérique avancée : Toute signature numérique avancée dispose d'une reconnaissance légale. Elle doit [28] :

- Être liée de manière unique au signataire.
- Permettre d'identifier le signataire.
- Être créée par des moyens que le signataire doit garder sous contrôle exclusif.
- Être liée aux données auxquelles elle se rapporte.

Signature numérique sécurisée (qualifiée) : Il s'agit d'une signature numérique avancée à laquelle s'ajoutent des informations pouvant aider le vérificateur à valider la signature sur le long terme en joignant, par exemple, un jeton temporel d'horodatage au message signé. Par cela, Le vérificateur est assuré que le signataire a utilisé un dispositif de création de signature fiable,

puisque le certificat du signataire et sa paire de clé lui ont été remis en toute sécurité[28].

1.4.2 Différents genres de signature :

La signature invisible et indéniable : Une signature est dite invisible et indéniable [29], lorsque tout le monde ne peut pas vérifier la signature à l'aide de la clé publique, cependant la non-répudiation est assurée et le signataire ne peut nier sa signature. A cette signature sont associés 3 protocoles : la signature en elle-même, un protocole de vérification et un protocole de désaveu.

La signature en aveugle : Une signature est dite en aveugle [30], lorsque nous souhaitons bénéficier de l'avantage de la signature qui est l'authentification sans que l'autorité de vérification ne sache le contenu que le signataire a signé.

La signature du groupe : Une signature est dite du groupe [31] lorsqu'une personne peut signer numériquement un message au nom du groupe. Ce protocole permet de prouver que la signature vient réellement du groupe (cercle) mais il est impossible de savoir qui à l'intérieur du groupe est le signataire.

1.4.3 Quelques algorithmes de signatures numériques

Signature El Gamal : Le schéma de cette signature [26] est l'un des premiers systèmes de signature numérique basé sur une arithmétique modulo un nombre premier. Pour générer les clés, nous sélectionnons tout d'abord un sous-groupe cyclique G d'ordre premier q, avec un générateur g. Après, on sélectionne un entier aléatoire secret a, $1 \leq a \leq q-1$ et on calcule $y = g^a$ dans G. La clé publique est (G,g,y) et la clé privée est a. Pour générer la signature d'un message m, nous sélectionnons aléatoirement un nombre k secret tel que pgcd $(k, q) = 1$, $1 \leq k \leq q-1$. Puis, nous calculons l'élément r du groupe tel que $r = g^k$. Nous calculons après $k-1[q]$, puis l'empreinte de m et de r (h(m) et h(r)). Après, nous calculons $s = k^{-1}(h(m) - ah(r))[q]$. La signature de m générée est la paire (r,s). Pour vérifier la signature (r,s), nous prenons la clé publique du signataire (g,y) et nous calculons $h(m)$ et $h(r)$. Puis, nous calculons $s_1 = y^{h(r)}.r^s$ et $s_2 = g^{h(m)}$. Si $s_1 = s_2$, la signature est valide, sinon elle ne l'est pas.

Digital Signature Algorithm DSA : Proposé en 1991 par NIST, le DSA [32] est une signature qui a des points communs avec la signature El Gamal. La sécurité des clés repose sur la difficulté du problème du logarithme discret dans un groupe fini. Pour les générer, nous choisissons un nombre premier p et un autre nombre premier q de telle façon que $p-1 = qz$, avec z un entier. Puis nous choisissons h, avec $1 < h < p-1$ de manière à ce que $g = h^z[p] > 1$. Nous générons aléatoirement un entier s, avec $0 < s < q$ et calculons y = g^s [p]. La clé publique est (p, q, g, y). La clé privée est s. Pour générer la signature du message m, nous choisissons un nombre aléatoire x, $1 < x < q$. Puis, nous calculons $(s_1 = s^x$ [p]) [q] et $s_2 = (H(m) + s_1 * s)x^{-1}[q]$, où $H(m)$ est le résultat du hachage du message m. La signature est la paire (s_1, s_2). Pour la vérification de la signature, nous calculons $w = (s_2)^{-1}$ [q], $u_1 = H(m) * w$ [q], $u_2 = s_1 * w$ [q] et $v = (g^{u_1} * y^{u_2}[p])[q]$. La signature est valide si $v = s_1$

Signature RSA : Pour la génération des clés, nous choisissons deux très grands nombres premiers p et q [2]. Soit n et φ respectivement les produits $p \times q$ et $(p-1) \times (q-1)$. Prenons un grand nombre premier avec φ choisi aléatoirement et e son inverse modulo $\varphi(d \equiv e - 1$ $[\varphi])$. En particulier, la clé publique de chiffrement est (e, n) et la clé privée de déchiffrement est (d, n). Pour générer la signature s du message m, nous calculons l'empreinte de m, $s = h(m)^d[n]$. Pour vérifier la signature, nous calculons $m' = s^e[n]$ et calculons après l'empreinte de m'. Si les empreintes de m et de m' sont égales, alors la signature est valide, sinon elle est rejetée.

1.5 Authentification

L'authentification est un processus permettant la vérification de l'identité revendiquée d'une entité souhaitant accéder à l'information [2]. Particulièrement, ce processus peut avoir lieu en :

- utilisant ce que connait l'entité voulant s'authentifier (le mot de passe, le code PIN, etc) ;
- utilisant un objet que détient l'entité voulant s'authentifier (un badge, une carte à puce, un Tag RFID, etc) ;
- utilisant l'entité voulant s'authentifier elle-même (une empreinte).

Ainsi, l'intérêt et le besoin de mettre en place de nouveaux processus d'authentification, plus robustes, demeure existant. Le MAC est l'une des primitives cryptographiques existantes assurant l'authenticité, en permettant au récepteur de l'empreinte authentifiante (nommée Tag) de vérifier si une altération s'est produite pendant la transmission, à l'aide de l'utilisation d'une clé secrète. En d'autres termes, cela implique l'utilisation d'une clé secrète k et trois algorithmes,

un algorithme de génération de clés secrètes, un algorithme de génération de Code d'Authentification de Message $MACG_k$ et un autre algorithme de vérification $MACV_k$. Pour ce faire, l'algorithme de génération $MACG_k$ prend en entrée un message M de longueur arbitraire et génère une valeur unique, de longueur fixe. Cette valeur générée est jointe au message M pour construire un message authentifié. L'algorithme de vérification $MACV_k$ utilise la même clé secrète k et le message M pour reconstruire une valeur valide si aucune altération n'a affecté le message, sinon il est invalide et a été modifié [3]. Dans la littérature, il existe principalement deux grandes familles d'algorithmes MAC [3] : les algorithmes MAC basés sur les systèmes de chiffrement et d'autres basés sur les fonctions de hachage.

1.5.1 Différentes méthodes de construction de Codes d'Authentification de Message

Codes d'Authentification de Message basés sur les chiffrements symétriques : Généralement, ces MAC sont produits en utilisant un algorithme de chiffrement symétrique fortement sécurisé. Les Codes d'Authentification de Message basés sur un système de chiffrement les plus

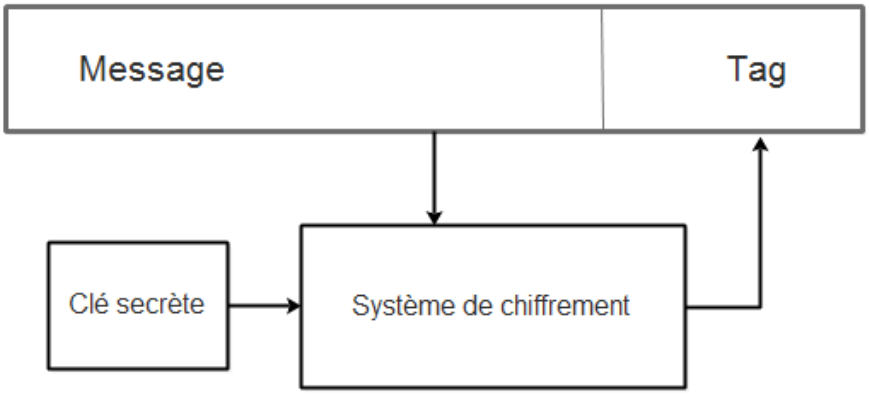

Figure 1.9 – MAC basé sur système de chiffrement

connus sont le Code d'Authentification de Message basé sur un chiffrement par bloc CMAC [8] et le Code d'Authentification de Message basé sur un chiffrement de chaînage par bloc Code d'Authentification de Message basé sur un système de Chiffrement par Blocs Chaînés (CBC-MAC) [33]. L'exemple le plus répandu de ces constructions est le AES-MAC.

En général, pendant le processus de génération MAC, le message M est chiffré. A la fin, la valeur générée Tag est envoyée avec le message, en utilisant une fonction de chiffrement E et une clé

secrète k (voir la figure 1.9).

Codes d'Authentification de Message basés sur des fonctions de hachage : Les MAC basés sur une fonction de Hachage, appelés aussi fonction de Hachage à clé, sont des familles de fonctions de hachages H assurant l'authenticité par l'utilisation d'une clé secrète k d'un ensemble de clés secrètes K[34], exprimé comme suit (voir la figure 1.10) :

$H_{(k \in K)} : \{0,1\}^* \rightarrow \{0,1\}^n ; M \rightarrow H(M)$

Ce type de Code d'Authentification de Message est largement déployé par de nombreux proto-

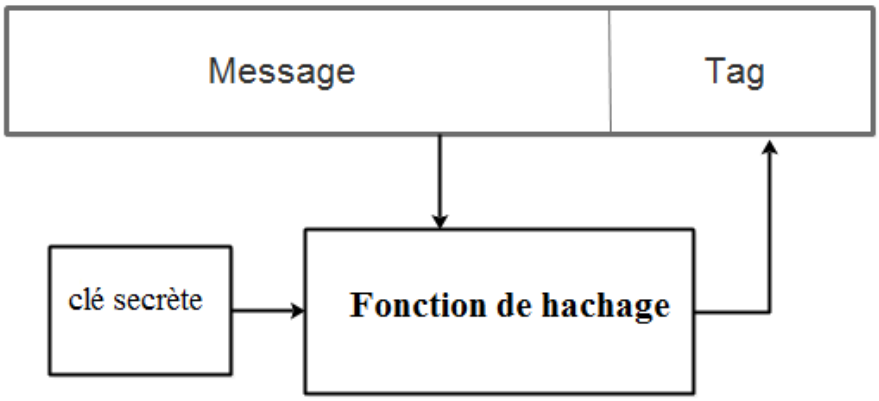

Figure 1.10 – MAC basé sur une fonction de Hachage

coles de réseau tels que Secure Shell (SSH), Internet Protocol Security (IPSec), Transport Layer Security (TLS) et a été standardisé par NIST et ISO. De plus, l'authenticité fournie, offre un niveau élevé de sécurité [35] en empêchant toute entité malveillante de forger un nouveau Tag valide. Car tout nouveau Tag, créé malicieusement, est immédiatement rejeté après la phase de vérification. De cette manière, l'acceptation ne devient possible que pour la valeur correctement construite à l'aide de la clé valide et du message valide. Ce Code d'Authentification de Message est aussi appelé fonction de Hachage à clé [4] car lors de sa construction il implique l'utilisation d'une clé secrète k à partir d'un ensemble de clés secrètes K, combinée avec la fonction de Hachage H pour générer le Tag [3]. La robustesse de cet algorithme réside dans les propriétés cryptographiques robustes de la fonction de Hachage H utilisée [4]. La plus répandu parmi ces constructions est le Code d'Authentification de Message basé sur une fonction de Hachage (HMAC) [4]. Il repose sur l'utilisation de deux constantes "ipad" et "opad" et sur une

fonction de Hachage H. Le MAC d'un message M avec une clé K est défini par :

$$HMAC(M) = H(K \oplus opad \parallel H(K \oplus ipad \parallel M)) \tag{1.2}$$

1.5.2 Travaux antérieurs en Codes d'Authentification de Message

Travaux existants en MAC basés sur un système de Chiffrement : Dans sa recommandation pour le CMAC, le NIST indique que le Code d'Authentification de Message est obtenu en utilisant trois clés. De cette façon, la réduction séquentielle du texte en clair et son chiffrement sont effectués simultanément [8].

L'algorithme de NIST, CMAC utilise l'AES comme un système de chiffrement pour le AES-CMAC. Il a besoin d'une clé secrète, le texte en clair et la longueur du texte en clair comme entrées pour générer une chaîne de bits de longueur fixe. Bien que la sécurité de l'AES-CMAC repose sur la sécurité de l'algorithme AES, la clé secrète doit être sécurisée. Si elle est partagée de manière inappropriée ou modifiée, l'authentification et l'intégrité ne sont plus garanties [9].

Le DES-CMAC a une complexité similaire à celle de AES-CMAC. Il consiste à utiliser un cycle unique de DES et offre une sécurité élevée contre les attaques de récupération de clé [10].

Dans l'article [36], les auteurs décrivent l'Alpha-MAC qui présente des avantages dans le domaine des systèmes embarqués (comme dans les applications de cartes à puce). C'est un MAC itératif généré par des injections successives de blocs de messages. Sa clé secrète est obtenue à partir de deux transformations AES, appliquées au début et à la fin du processus.

Cependant dans l'article [11], les auteurs ont présenté plusieurs nouvelles techniques de cryptanalyse réussies sur ces MAC.

Dans l'article [37], les auteurs proposent un nouveau code d'authentification de message appelé ZMAC construit à partir d'un Tweakable Block-Cipher (TBC). Ces TBC utilisent la construction de type PMAC (Parallelizable Message Authentication) [38] basée une entrée de bit t supplémentaire nommée tweak et des blocs de n bits pour agir comme une famille de chiffrement par blocs indépendants de 2^t indexés par le tweak.

Nous avons constaté que le nombre des Codes d'Authentification de Message basés sur les systèmes de chiffrement est très réduit, ce qui nous a poussé à en proposer un nouveau. En addition, notre nouveau MAC est considéré comme l'un des premiers MAC a utiliser un système de chiffrement évolutionniste lors de son processus. En particulier, l'utilisation du chiffrement évolutionniste SEC donne une diversification de clés, ce qui rend notre conception Code d'Au-

thentification de Message proposé basé sur le système de Chiffrement SEC (SEC-CMAC) robuste aux attaques de récupération de clés.

Travaux existants en MAC basés sur une Fonction de Hachage : En 1995, les XOR-MAC, des Codes d'Authentification de Message basés sur XOR, ont été proposés par Bellare et al. afin d'obtenir des Tags aléatoires. Leur construction démontre l'application d'une fonction PseudoRandom Function (PRF) utilisant l'opération XOR sur l'ensemble des blocs générés [39]. Dans l'article [40], les auteurs ont publié un standard, à savoir le HMAC. Ce Code d'Authentification de Message est dérivé d'une fonction de Hachage basé sur la méthode de Merkle-Damgard en combinaison avec une clé secrète partagée. Ce standard proposé obtient presque toute sa sécurité à partir de la fonction de Hachage utilisée.
De plus, dans l'article [5], les auteurs ont introduit un schéma d'authentification léger basé sur les automates cellulaires pour utilisateurs multiples dans le contexte du Cloud Computing.
Aussi, un autre Code d'Authentification de Message, appelé KMAC, basé sur la fonction de Hachage Keccak et vainqueur de la compétition "SHA-3" a été introduit par Bertoni et al. dans l'article [27]. Le Tag de ce MAC est la donnée résultante produite par le hachage "Keccak" de la concaténation de la clé secrète avec le message [41].
En 2013, dans l'article [6], les auteurs ont décrit l'implémentation d'un autre HMAC basé sur la fonction de Hachage légère PHOTON [7]. Néanmoins, sa mise en œuvre n'est pas pratique pour les ressources à grande vitesse avec une puissance de calcul restreinte [6].
Aussi "Auth256" [42] est un autre HMAC dérivé de la fonction hachage "Hash256" combinée avec une clé secrète qui est obtenue à l'aide de l'AES en mode compteur.
Dans le travail présenté par Wu et al., un nouveau Code d'Authentification de Message est proposé suivant la construction basée sur des fonctions de hachage universelles telles que les fonctions non linéaires presque parfaites (Almost Perfect Nonlinear (APN)). L'algorithme proposé offre une sécurité contre les attaques de substitution et de falsification linéaire [43].
Dans un autre travail de Bellare et al., appelé AMAC [44], un Code d'Authentification de Message est proposé. Celui-ci utilise une construction rapide et simple d'une fonction pseudo aléatoire (PRF). Ce MAC s'est avéré plus efficace que HMAC surtout pour les messages courts.
La robustesse d'un Code d'Authentification de Message repose généralement sur la robustesse de la fonction de Hachage utilisée dans son processus. Puisque la plupart des fonctions de hachages connues sont déjà attaquées et cassées, dans ce mémoire nous présentons deux de nos nouveaux Codes d'Authentification de Message conçus et prouvés robustes, l'un rapide et l'autre léger.

1.5.3 Attaques génériques sur les Codes d'Authentification des Messages

Un MAC(.,.) est dit sécurisé s'il satisfait ces cinq propriétés requises [45] :

- Étant donné la clé k et MAC (k, m), il est difficile, dans le temps, de retrouver le message m ;
- Étant donné la clé k, il est difficile, dans un temps raisonnable, de trouver deux messages différents m et m' tels que MAC (k, m) = MAC (k, m') ;
- Étant donné (peut-être plusieurs) paires de m et de MAC (k, m), il est difficile, dans un temps raisonnable, de trouver la clé k ;
- Sans une connaissance préalable de k, il est difficile, dans un temps raisonnable, de calculer MAC (k, m) pour tout m.

Supposons que la transmission se fait à travers un canal non sécurisé et qu'un adversaire ou une entité malicieuse l'intercepte. En générale, afin de produire une falsification de Tag, deux attaques génériques peuvent être appliquées aux Codes d'Authentification de Message, soit l'adversaire devine le Tag directement, ou il devine la clé secrète puis calcule le Tag.

La falsification du MAC [46] : était parmi l'un des premiers à proposer un modèle d'authentification en utilisant les MAC, et aussi à avoir démontré deux modèles d'attaques différents contre ceux-ci, notamment la falsification par usurpation d'identité et par substitution. La falsification signifie que l'adversaire peut forger un Tag valide, notamment une falsification sélective où l'adversaire peut forger un Tag valide pour un message aléatoire et une falsification existentielle où l'attaquant est capable de forger un couple (Message/Tag) valide en observant plusieurs paires de (Messages/Tag). Pour forger un Tag valide, tel que ce Tag est d'une longueur qui est égal à n bits et la clé secrète est d'une longueur qui est égale à k bits, il faut au maximum 2^n paires de Messages/Tag connues, et au minimum $2^{n/2}$ essais [47].

Recherche de clé exhaustive : Il s'agit d'une attaque de récupération de clé où l'adversaire essaie les valeurs possibles de la clé secrète jusqu'à trouver la clé correcte qui a été choisie. Si l'adversaire a accès à au moins une paire (message/Tag), la clé peut être devinée en examinant les éléments de l'ensemble des clés contre les paires (messages/Tag). L'adversaire attaquera l'espace de clés en testant toutes les clés possibles pour un message donné et vérifie si le Tag obtenu est valide. Si on prend un Tag de longueur qui est égal à n bits et une clé secrète de longueur qui est égale à k bits, ceci nécessite la connaissance de (k/n) paires de (Messages/tag), et environ

min $(2^{n-1}, 2^{k-1})$ essais (une recherche dans la moitié de l'espace clé) [47].

1.6 Conclusion

Si le développement des réseaux informatiques a permis de communiquer plus facilement, il n'en reste pas moins qu'il a soulevé plusieurs problématiques concernant l'authenticité des données échangées. Étant exposée à plusieurs attaques, la donnée traitée, stockée et surtout celle communiquée, fait l'objet de convoitises. Les menaces les plus connues sont les virus, phising, les mal configurations de site Web, les programmes malveillants pouvant être installés, par un utilisateur non sensibilisé, ouvrant la porte à des intrusions. les problématiques liées à l'authenticité, dans les communications en réseaux arbitraires, ont conduit les experts en cryptographie à proposer des solutions et développer un ensemble d'outils. L'une des solutions les plus efficaces proposées pour remédier à ces problèmes est le Code d'Authentification de Message. L'intérêt et le besoin de mettre en place de nouveaux systèmes d'authenticité, plus robustes, demeure existant. En générale, un Code d'Authentification de Message ne garantit que l'authenticité et non pas la confidentialité ou la non répudiation. Cette thèse propose la conception et la réalisation de trois nouveaux Codes d'Authentification de Message MAC permettant ainsi d'enrichir et de remédier à la situation de carence, en ce domaine d'authenticité, en mécanismes robustes, rapides, et légers.

Dans ce qui suit, je donne une description détaillée de nos trois conceptions proposées.

CHAPITRE

2 CONTRIBUTION AUX CODES D'AUTHENTIFICATION DE MESSAGE BASÉS SUR LES SYSTÈMES DE CHIFFREMENT

Sommaire

2.1 Introduction

Un MAC est utilisé pour permettre l'intégrité et l'authentification du message entre l'émetteur et le récepteur. Ce mécanisme implique une utilisation d'un processus de génération de Tag et un processus de vérification [48]. Ayant un espace de clé K, un espace de message M et un espace de Tag. Le processus de génération MACG_k a comme entrée le texte en clair et une clé secrète.

$$Tag = \text{MAC}G_k(M). \tag{2.1}$$

Le processus de vérification MACV_k a comme entrées le texte en clair et le Tag, pour vérifier que le Tag est valide ou invalide (Figure 2.1).

$$\text{MAC}V_k(M, Tag) = valid, invalid \tag{2.2}$$

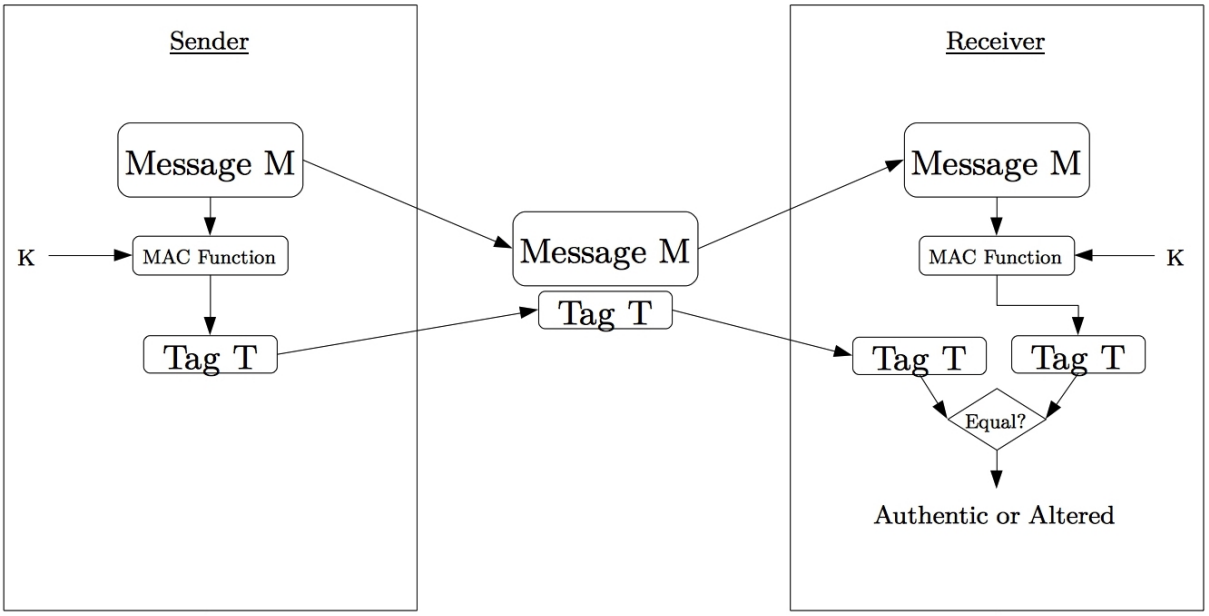

Figure 2.1 – Processus de génération et de vérification (MAC)

Dans la littérature, il existe principalement deux grandes familles d'algorithmes MAC : algorithmes MAC basés sur les fonctions de hachage et d'autres basés sur les systèmes de chiffrement. En particulier, les constructions de Codes d'Authentification de Message basés sur un système de chiffrement les plus connues sont à savoir le Code d'Authentification de Message basé sur un chiffrement par bloc CMAC [8] et le Code d'Authentification de Message basé sur un chiffrement par chaînage de bloc CBC-MAC [33].
En utilisant une fonction de chiffrement E et une clé secrète k, les blocs de longueur n sont chiffrés d'une manière itérative [3]. À la fin, de ce processus itératif, la valeur générée Tag est envoyée avec le message (voir la Figure 2.2)

Par ailleurs, dans le processus de vérification MAC, le récepteur commence par le processus de génération MAC, pour obtenir un Tag'. Ce Tag' est comparé par la suite au Tag reçu pour confirmer que le message est intègre et authentique ou révéler que le message a été modifié.
Ainsi, la carence en Code d'Authentification de Message surtout en ceux basés sur les systèmes de chiffrement, nous a poussé à concevoir et implémenter un nouveau Code d'Authentification de Message basé sur le système de chiffrement évolutionniste déjà proposé "SEC". Ce MAC est à présent l'un des premiers à utiliser un algorithme évolutionniste dans son processus. Nous commençons dans ce chapitre par une présentation des algorithmes évolutionniste et de l'algorithme de chiffrement du système de chiffrement symétrique évolutionniste "SEC". Ensuite nous

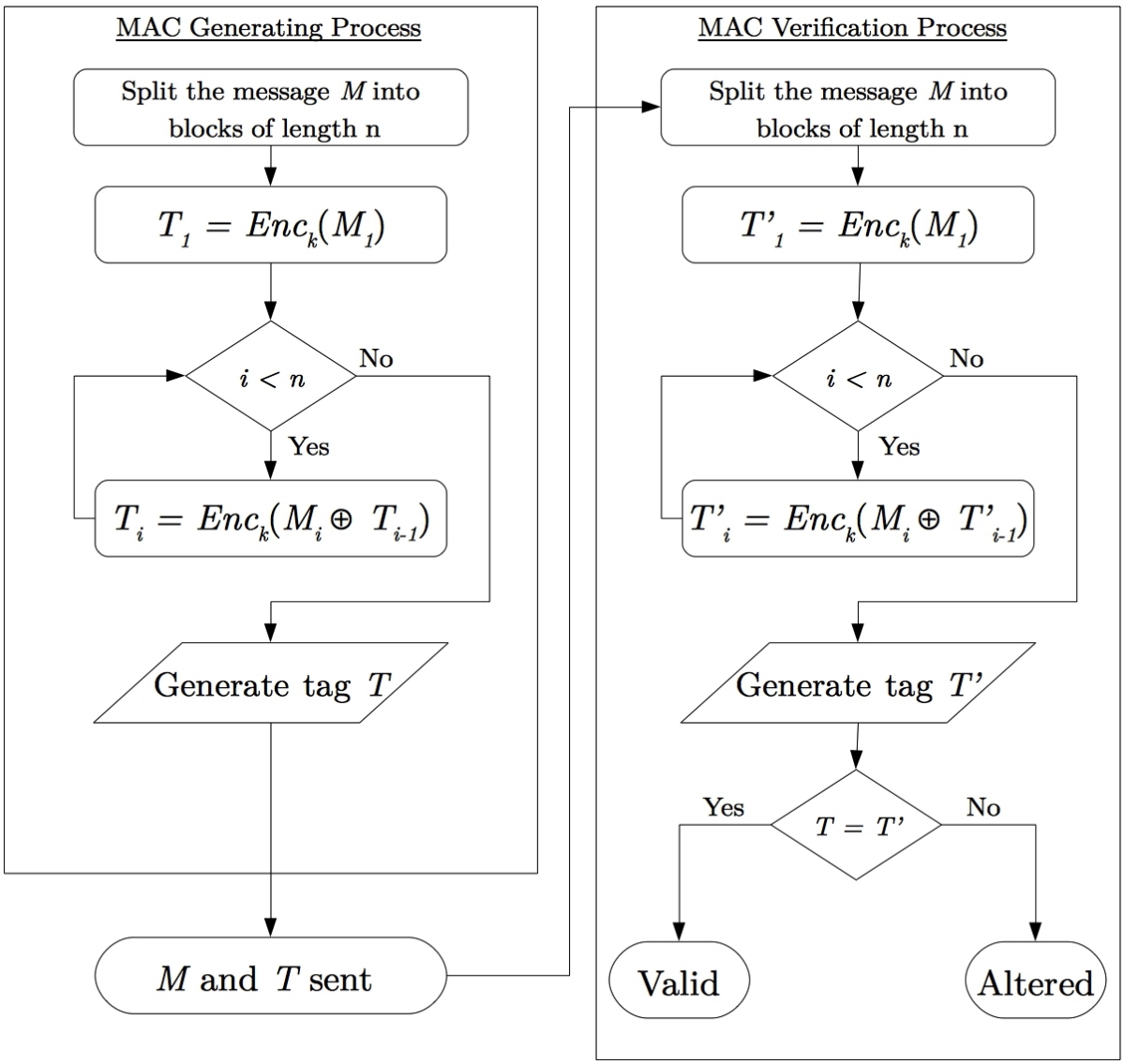

Figure 2.2 – Processus général d'un (CMAC)

présentons en détail notre nouveau Code d'Authentification de Message basé sur le système de chiffrement symétrique, précédemment proposé, utilisant un algorithme évolutionniste (SEC) [13].

2.2 Description du système de chiffrement évolutionniste (SEC)

2.2.1 Description des algorithmes évolutionnistes

Les algorithmes évolutionnistes sont omniprésents de nos jours, après avoir été appliqués avec succès à de nombreux problèmes comme l'optimisation, l'apprentissage automatique, bio-informatique et cryptographie. Contrairement à la plupart des techniques d'optimisation, ces algorithmes sont des méthodes de recherche stochastique, qui prennent une population de solutions provisoires qui est manipulée de manière compétitive en appliquant des opérateurs de variation pour trouver une solution satisfaisante, sinon une solution optimale [49].
Pour ce faire, soit une fonction F à optimiser à valeurs définie sur un espace Ω.

- Les individus sont les points de l'espace de recherche Ω ;
- Les populations sont des ensembles d'individus ;
- On parlera d'une génération pour la boucle principale de l'algorithme. Le temps à la génération i, de l'évolution est noté Π_i, pour la population de taille fixe P .
- La fonction performance (fitness) est la fonction objectif F ;

Un algorithme évolutionniste procède de la manière suivante :

- **la phase d'initialisation de la population Π_0 :**
  D'une façon aléatoire avec une probabilité uniforme sur Ω, P individus sont choisis dans Ω ;
- **La phase d'évaluation des individus de Π_0 :** Le calcul des valeurs de F pour tous les individus ;
- **La génération i construit la population Π_i à partir de la population Π_{i-1} :**
  - ☐ Sélection des individus les plus performants de Π_{i-1} au sens de F
  - ☐ Application des opérateurs de variation à la population Π_{i-1} sélectionnés, ce qui génère de nouveaux individus, la population Π_i ;
  - ☐ Évaluation la population Π_i ;
  - ☐ Remplacement de la population Π_{i-1} par une nouvelle population créée à partir des populations Π_i et/ou Π_{i-1}, en utilisant une sélection darwinienne (les survivants les mieux adaptés).

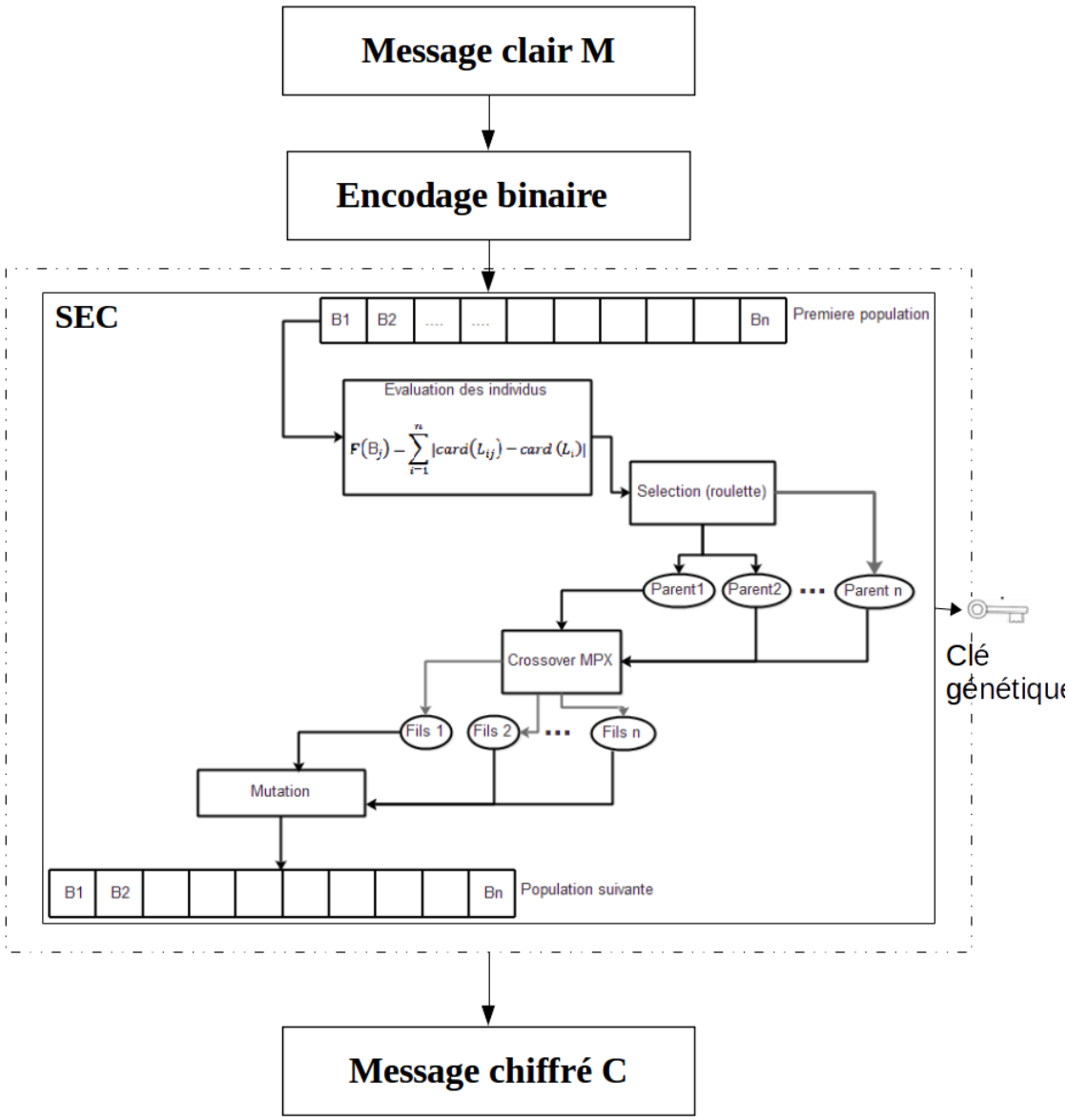

Figure 2.3 – Algorithme de chiffrement de SEC binaire pour une itération .

2.2.2 Description de ESEC

Les chiffrements évolutionnistes sont des chiffrements basés sur les algorithmes évolutionnistes. En particulier, le système de chiffrement SEC est l'un des premiers systèmes de chiffrement symétrique à utiliser un algorithme évolutionniste pour chiffrer les données [50].

Dans ce qui suit, je décrirai brièvement l'algorithme de chiffrement ESEC (figure 2.3). ESEC est l'extension binaire du Chiffrement évolutionniste SEC puisque nous n'aurons besoin d'utiliser que l'algorithme de chiffrement dans la construction de notre nouveau Code d'Authentification

de Message basé sur un système de chiffrement symétrique [50].
Le processus de chiffrement ESEC commence par un pré-chiffrement utilisant un ensemble d'opérations, notamment la substitution ou la permutation ou le chiffrement affine, etc. Ensuite, un encodage binaire du message est effectué. Puis, un nombre entier k est tiré au hasard, pour pouvoir découper le message M en des blocs binaires de même taille k, ce qui aboutit au Chromosome Initial (Ch-Initial) dénoté par $(B_1, B_2, \ldots, B_m)$. un éventuel remplissage par des "0" sera effectué au dernier bloc si sa taille est inférieur à k. nous soulignons que Ch-Initial est un vecteur de taille m dont les éléments sont les listes L_i représentant les positions de blocs B_i dans M.
Pour chiffrer, l'algorithme ESEC, nous suivons les étapes ci-dessous :

- Étape 1 : q permutations sont appliquées surCh-Initial. Après, les q individus résultant $(B_1, B_2, \ldots, B_q)$ forme la population initiale P_0 (le 0 pour désigne l'itération initiale).
- Étape 2 : Une fonction d'évaluation F est alors appliquée sur B_j individus.
  $F(B_j) = \sum_{i=1}^{m} |card(L_{ij}) - card(Li)|$
- Étape 3 : Les individus sont sélectionnés en prenant en considération les résultats de fonction d'évaluation F.
- Étape 4 : L'opérateur génétique "croisement MPX" est appliqué aux individus sélectionnés sera appliqués avec un taux variant de 70% à 100 %. tandis que l'opérateur génétique de "mutation" est appliqué aux individus issus du croisement avec un taux variant de 0.1% à 5 %. Ce qui donnera naissance à la population suivante.

Les étapes 2 à 4 sont alors répétées jusqu'à l'atteinte du critère d'arrêt prédéfini pour aboutir au Chromosome Final (Ch-Final)) qui représente le chiffré C. Par la suite, la clé secrète nommée clé génétique [50] $ESEC_k$ est construite et représente la permutation transformant Ch-Initial en Ch-Final.

2.3 Description de SEC-CMAC

2.3.1 Algorithme de SEC-CMAC de génération de Tag

Notre processus de génération proposé (voir Algorithme 2.1 figure 2.4) commence par un découpage du texte clair M en n blocs de taille 512 bits. Si la taille du dernier bloc est inférieure à 512 bit, nous ajoutons une séquence de '1000..0′ pour compléter le bloc en 512 bit. Puis chaque bloc M_i est chiffré en utilisant l'algorithme SEC, comme suit :

- Pour i = 1, $T_1 = SEC_{k_1}(M_1)$

– Pour $2 \leq i \leq n-1, T_i = SECk_i(T_{i-1} \oplus M_i)$

– Pour i = n, $T_n = SEC_{k_n}(M_n \oplus T_{n-1} \oplus k_{ranSEC}$, tel que k_{ranSEC} est choisi aléatoirement parmi les n k_i clé de chiffrement.

le Tag est égale à T_n.

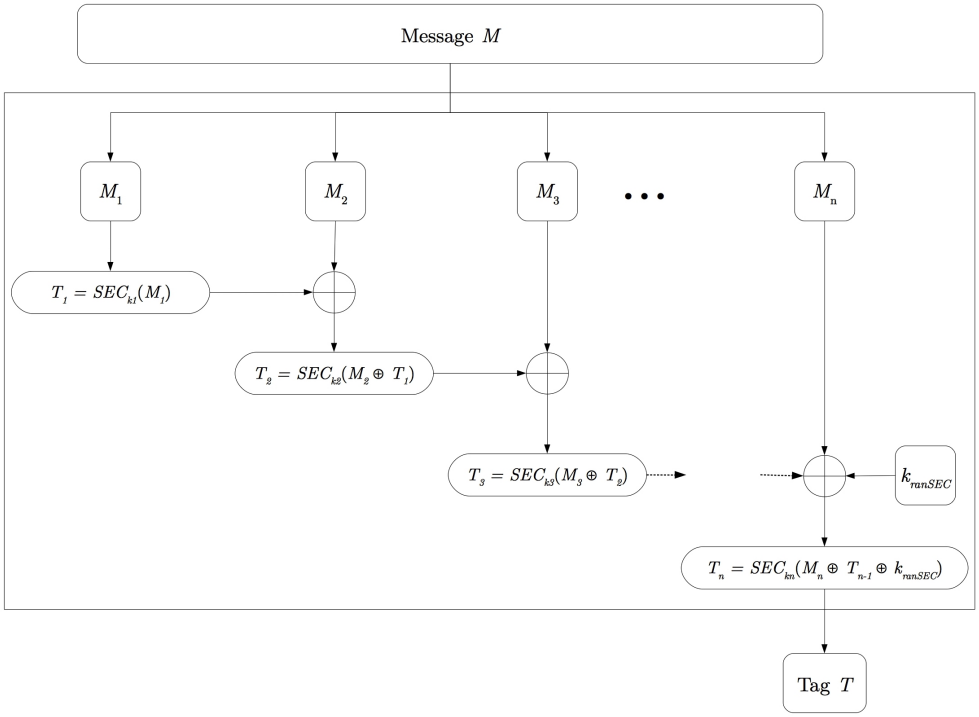

Figure 2.4 – Algorithme de génération de Tag de SEC-CMAC .

Algorithme 2.1 Algorithme de génération de Tag de SEC-CMAC

input : Message M
output : Tag
Initialisation
n $\leftarrow$ 0 /$*$ the number of M_is$*$/
Tag$\leftarrow$ " "
kranSEC $\leftarrow$ " "
SECkeys $\leftarrow \varnothing$
$SEC_{k_i} \leftarrow \varnothing$
$SEC_{k_1} \leftarrow$ Extraire la clé généré lors du chiffrement de M_1
$T_i \leftarrow \varnothing$
$T_n \leftarrow \varnothing$
commencement :
$M_{binary} \leftarrow$ Encoder M en binaire
$M_i \leftarrow$ découpage du M_{binary} en M_is en blocs de 512 bit et padder si nécessaire
$T_1 \leftarrow$ le résultat du chiffrement utilisant SEC sur M_1 avec la première clé de SEC_{k_1}
SECkeys $\leftarrow$ Inserer SEC_{k_1}
pour $2 < i < n-1$ **faire**
– $T_i \leftarrow$ le chiffrement résultant utilisant SEC sur $(T_{i-1} \oplus M_i)$ avec la clé SEC_{k_i}
– SECkeys $\leftarrow$ Inserer (SECkeys SEC_{k_i})
fin pour
$k_{ranSEC} \leftarrow$ choisir aléatoirement une clé des SECkeys et padder si nécessaire pour avoir une longueur de 512 bit
$T_n \leftarrow$ le chiffrement résultant utilisant SEC sur $(T_{n-1} \oplus M_n \oplus k_{ranSEC})$ avec la clé SEC_{k_n}
Tag$\leftarrow T_n$
End

2.3.2 Algorithme de SEC-CMAC de vérification

Cet algorithme de vérification proposé utilise les mêmes clés lors de la génération du Tag. Le receveur prend le message, régénère le Tag et le compare avec le Tag reçu puis décide de sa validité (voir Algorithme 2.2).

Algorithme 2.2 Algorithme de vérification de Tag de SEC-CMAC

input : M, SECkeys , k_{ranSEC}, Tag
output : Valid_Tag , Invalid_Tag
Tag' $\leftarrow$ ESECDechiffrement(M, SECkeys) ;
si Tag' = Tag **alors**
  Valid_Tag ;
sinon
  Invalid_Tag ;
finsi

2.4 Conclusion

Dans ce chapitre nous avons donné une description des algorithme de génération et vérification de SEC-CMAC. Dans le dernier chapitre nous présenterons l'évaluation de la sécurité et de la performance de notre SEC-CMAC.

CHAPITRE

3 CONTRIBUTION AUX CODES D'AUTHENTIFICATION DE MESSAGE BASÉS SUR LES FONCTIONS DE HACHAGE

Sommaire

3.1 Introduction

Les MAC basés sur une fonction de Hachage, appelés aussi fonctions de hachage à clé, sont des familles de fonctions de hachage assurant l'authentification par l'utilisation d'une clé secrète k, d'un ensemble de clés secrètes K [34], exprimé comme suit :

$$H_{(k\in K)} : \{0,1\}^* \rightarrow \{0,1\}^n; M \rightarrow H(M) \tag{3.1}$$

Encore, la robustesse d'un Code d'Authentification de Message basé sur une fonction de hachage réside dans la résistance de cette dernière aux attaques par collision. Puisque la plupart des fonctions de hachage connues sont déjà attaquées, dans ce chapitre notre nouveau Code d'Authentification de Message conçu, prouvé robuste et rapide.

Dans ce qui suit, nous présentons en détaille notre contribution aux Codes d'Authentification de Message basés sur une fonction de Hachage. Nommé «keyed-CAHASH», ce nouveau Code d'Authentification de Message a été prouvé rapide. Tout d'abord, nous avons commencé par une

présentation des automates cellulaires. Ces derniers sont considérés comme un modèle mathématique pour les systèmes discrets complexes avec des structures auto-reproductrices. Ayant aussi comme caractéristiques la qualité aléatoire, le fonctionnement rapide, l'homogénéité et l'imprévisibilité, nous a poussé à les utiliser. Par la suite, nous présentons chacun des algorithmes celui de génération de clé, de génération de Tag et de vérification. Pour générer des clés secrètes nous choisissons aléatoirement un grand nombre premier, un indice. Après nous faisons évoluer « PSOCA » un nouveau générateur de nombre pseudo aléatoire récemment conçu, prouvé robuste et imprévisible, pour générer les deux clés de longueur de 512 bits.

3.2 Description Des Automates Cellulaires (AC)

Introduits la première fois par Neuman, en 1940, [15], les Automates Cellulaires (AC) ont été considérés comme un modèle mathématique pour les systèmes discret complexe avec des structures auto-reproductrices. Par la suite, en 1980, les AC ont été développé par Stephen Wolfram [15], dans le but de générer un nombre pseudo-aléatoire [51]. En raison de leur qualité aléatoire, leur fonctionnement rapide, leur simple implémentation et leur mise en œuvre matérielle et en addition, de leurs plusieurs caractéristiques telles que l'homogénéité, la localité, le parallélisme, la simplicité et l'imprévisibilité, les AC sont devenu un outil efficace dans la conception de nouvelles primitives cryptographiques [5], en particulier pour concevoir des générateurs pseudo-aléatoire [51], des fonctions de hachage et des MAC robustes.
En particulier, un automate cellulaire est un système dynamique discret, formant un réseau régulier de cellules. Chaque cellule, possédant un état, communique avec ses voisins directs et évolue suivant une règle de transition. Les AC unidimensionnels [15] sont considérés comme un système dynamique qui se compose d'un ensemble de cellules, fini, linéairement connectées où chaque cellule prend un état à partir d'un ensemble de possibilités d'états. En utilisant une règle de transition locale, la valeur d'un état est mise à jour en tenant compte l'état des cellules voisines. Un AC est un système dynamique qui peut être considéré comme un ensemble de cellules, où l'état de chaque cellule interagit avec un ensemble d'états de cellules appelées voisines, selon une fonction de transition. Mathématiquement, un AC est définie comme un quadruple A = $\{S, D, N, f\}$ où :

- S est l'ensemble de base qui détermine l'ensemble des états des cellules possibles.
- D est la dimension ou la taille de AC
- N est l'état voisin suivant la structure de voisinage existante.

– f est la fonction de transition. elle est défini comme une fonction booléenne f : $S^n \rightarrow S$ qui relie un état à son état suivant. En particulier, la transition d'état au temps (t + 1), notée $s_i^{(t+1)}$.

- Tel que r est le rayon du voisinage et N_i^t sont les voisins de la $i^{ème}$ cellule au temps t.
- Il convient de noter que cette règle de transition dépend uniquement des valeurs d'états possibles et du rayon de voisinage r qui définit le nombre de voisins.
- Par ailleurs, cette fonction peut être représentée comme une table de vérité ou comme un nombre décimal nommé numéro de règle. Par exemple, dans notre conception, nous avons utilisé la fonction $\sigma()$, qui est une fonction de transition globale, définie comme étant $\sigma : \{0,1\}^* \rightarrow \{0,1\}^*$ qui a pour but renvoyer l'état de toutes les cellules à la prochaine étape à travers l'évolution du AC, suivant l'équation $x^{t+1} = \sigma(x^t)$, tel que chacune des cellules évolue selon l'une des règles "23" et "150". Ces règles sont à la fois une AC à une seule dimension (voir tableau 3.1). En particulier, les fonctions logiques de ces règles utilisées sont décrites comme suit [52] :

  ▶ "Règle 150" : $s_i^{t+1} = s_{i-1}^t \oplus s_i^t \oplus s_{i+1}^t$

  ▶ "Règle 23" : $s_i^{t+1} = (\bar{s^t}_{i-1} \odot \bar{s^t}_{i+1}) \oplus (\bar{s^t}_i \odot \bar{s^t}_{i+1}) \oplus (\bar{s^t}_{i-1} \odot \bar{s^t}_i)$

  (tel que $\oplus$ et $\odot$ désignent, respectivement, les opérations XOR, ET bit à bit.)

Tableau 3.1 – la représentation des règles "23" et "150" (3-voisinages ,r = 1)

Nom de la règle	111	110	101	100	011	010	001	000
Règle 23	0	0	0	1	0	1	1	1
Règle 150	1	0	0	1	0	1	1	0

Les conditions aux limites : En extrémités de l'espace des cellules, les cellules doivent interagir avec les cellules des voisines. Il est nécessaire de définir les conditions aux limites appropriées d'un AC. Notamment ; il existe différents types de conditions aux limites, à savoir ils peuvent être nulles (des cellules extrêmes connectées à la logique 0) ou périodiques (les cellules extrêmes sont adjacentes l'une à l'autre)[15].

Définitions :

– AC uniforme : Si toutes les cellules obéissent à la même règle, le AC est dit uniforme. Sinon, il ne l'est pas.

- AC linéaire : si la règle d'une cellule AC ne comporte qu'une logique "XOR", elle est dite linéaire. Un AC avec toutes les cellules ayant des règles linéaires s'appelle un AC linéaire.
- AC non linéaire : si la règle d'une cellule d'un AC ne comporte qu'une logique AND-OR, elle est dite règle non linéaire. Un AC avec toutes les cellules ayant des règles non linéaires est dit AC non linéaire.
- Groupe et non AC de groupe : si une transition d'état d'un AC ne comporte que des états cycliques, l'AC est dit un groupe AC ; sinon il s'agit d'un AC non-groupe de AC. La règle appliquée sur une AC de groupe uniforme est appelée une règle de groupe ; sinon il s'agit d'une règle non-groupée.

En conséquence, l'exploitation logique des automates cellulaires, leur haute vitesse et leur faible complexité matérielle nous a inspiré à concevoir de nouveau MAC. Dans ce qui suit, dans ce chapitre et le chapitre suivant, certains de nos travaux de recherches sur le Code d'Authentification de Message ont été implémentés et mis en évidence en utilisant le concept des AC.

3.3 Description de Keyed-CAHASH

Dans cette section nous présentons notre nouveau Code d'Authentification de Message sécurisé basé sur l'utilisation d'une nouvelle fonction de Hachage à clé en utilisant le nouveau algorithme de génération de clés secrètes "PSOCA" proposé dans [1] basé sur les automates cellulaires.

Notre Code d'Authentification de Message suggéré $MAC = \{SubkeyGeneration, Generate - Keyed - CAHASH, Verify - Keyed - CAHASH\}$ est un triple algorithme associé à l'espace de clé K, l'espace de message M et l'espace de Tag T.

3.3.1 Génération des sous-clés

Tout d'abord, un ensemble de clés secrètes est utilisées $\{sk_1, sk_2, sk_{index}, sk_p\}$, qui sont générées en utilisant l'algorithme 3.2), à savoir sk_1 et sk_2 sont générées en utilisant le Générateur de nombres pseudo-aléatoires "PSOCA" [1] .

3.3.1.1 Description De (PSOCA) (1)

Les Générateurs de Nombres Pseudo-Aléatoires (PRNG) sont largement utilisés dans plusieurs domaines de la cryptographie, en particulier pour la génération de clés. En outre, différentes façons sont proposées pour les concevoir, à savoir les AC. En exploitant leurs propriétés (i.e. l'homogénéité, la localité, le parallélisme, la simplicité et l'imprévisibilité) font des AC un bon candidat pour un PRNG. En outre, un PRNG est un processus permettant de générer une séquence binaire uniforme, imprédictible et indépendante [2].

Par ailleurs, afin de sélectionner une nouvelle combinaison de règles optimales pour AC, Hanin et al. [1] ont proposé un nouveau PRNG appelé "PSOCA" basé sur une AC non uniforme,a n cellules d'ordre 1, périodique combiné à une Optimisation par Esaims Particulaire Binaire Modifié (MBPSO) pour sélectionner une liste de règles optimale. Chaque règle d'automate représente une particule de l'essaim, exprimée en binaire.

Cet algorithme MBPSO commence par initialiser d'une manière aléatoire la configuration des cellules et de la population des particules de taille P_{max}. Après cette initialisation, chaque itération conduit l'essaim à évoluer dans le temps pour avoir une nouvelle combinaison de particules. Cet algorithme est alors itéré jusqu'à atteinte d'une combinaison meilleure de règles avec une fonction objective maximale. Pour évaluer la séquence pseudo aléatoire obtenue, la fonction entropie représente la fonction objectif de l'algorithme MBPSO.

Par la suite, PSOCA (3.1) utilise la solution obtenue comme entrée pour évoluer l'AC plusieurs fois en lui associant des règles sélectionnées aléatoirement.

Si les conditions des tests ne sont pas satisfaites, PSOCA fait appel, une autre fois, à l'algorithme MBPSO afin de sélectionner à nouveau une liste de règles pour l'AC.

Algorithme 3.1 PSOCA

input : un AC (A) de taille n, P_{max} : le nombre de règles
Output : ε : séquence de nombres pseudo-aléatoires
· Initialiser la configuration de cellules de l'AC aléatoirement
· Sélectionner la liste de règles optimale en utilisant "MBPSO"
· Appliquer les règles sélectionnés par MBPSO sur les cellules de (A)
· passer la séquence pseudo aléatoire par les tests requis d'un PRNG

3.3.1.2 Description De notre algorithme de génération de clés

tout d'abord, l'algorithme "PSOCA" prend en entrée un nombre binaire, la graine, autant que entrée pour générer une séquence de clés à partir de cette graine.

Algorithme 3.2 Algorithme de génération des sous clés(s, k_z,n)

input : seed s, key size k_z,number of blocks n
Output : sk_1, sk_2, sk_p , sk_{index}
· $sk_1 \leftarrow$ PSOCA(s,k_z) [1]
· $sk_2 \leftarrow$ PSOCA(sk_1, k_z) [1]
· $sk_p \leftarrow$ choose a prime number
· $sk_{index} \leftarrow$ choose an index $\in [0, 1]$

Pour obtenir les sous-clés $Sk_s = \{Sk, Sk_1, Sk_2\}$, "PSOCA" [1] prend deux paramètres, une graine (s) de taille (k_z) pour fournir la première sous-clé de longueur 512 bits Sk_1. Par la suite, "PSOCA" est évolué pour générer une deuxième clé Sk_2 de 512 bits à partir de Sk_1. En outre, Sk_p est un nombre secret qui est premier avec une longueur spécifié de bits (c'est-à-dire 13 bits dans notre cas). Sk_{index} est un index secret choisi aléatoirement de $[0, n]$ (l'algorithme 3.2). Pour chaque session, une fois que les sous-clés sk = $\{sk_1, sk_2, sk_{index}, sk_p\}$ sont générées et secrètement partagées (par exemple via un support sécurisé) entre les expéditeurs légitimes et les récepteurs. L'expéditeur peut générer le tag en utilisant notre algorithme de génération de MAC proposé.

3.3.2 Algorithme de génération de MAC Generate-Keyed-CAHASH

Cet algorithme 3.3 d'authenticité probabiliste prend en entrée un message m∈M, une clé secrète k∈K et a et produit une valeur d'authenticité Tag∈T.

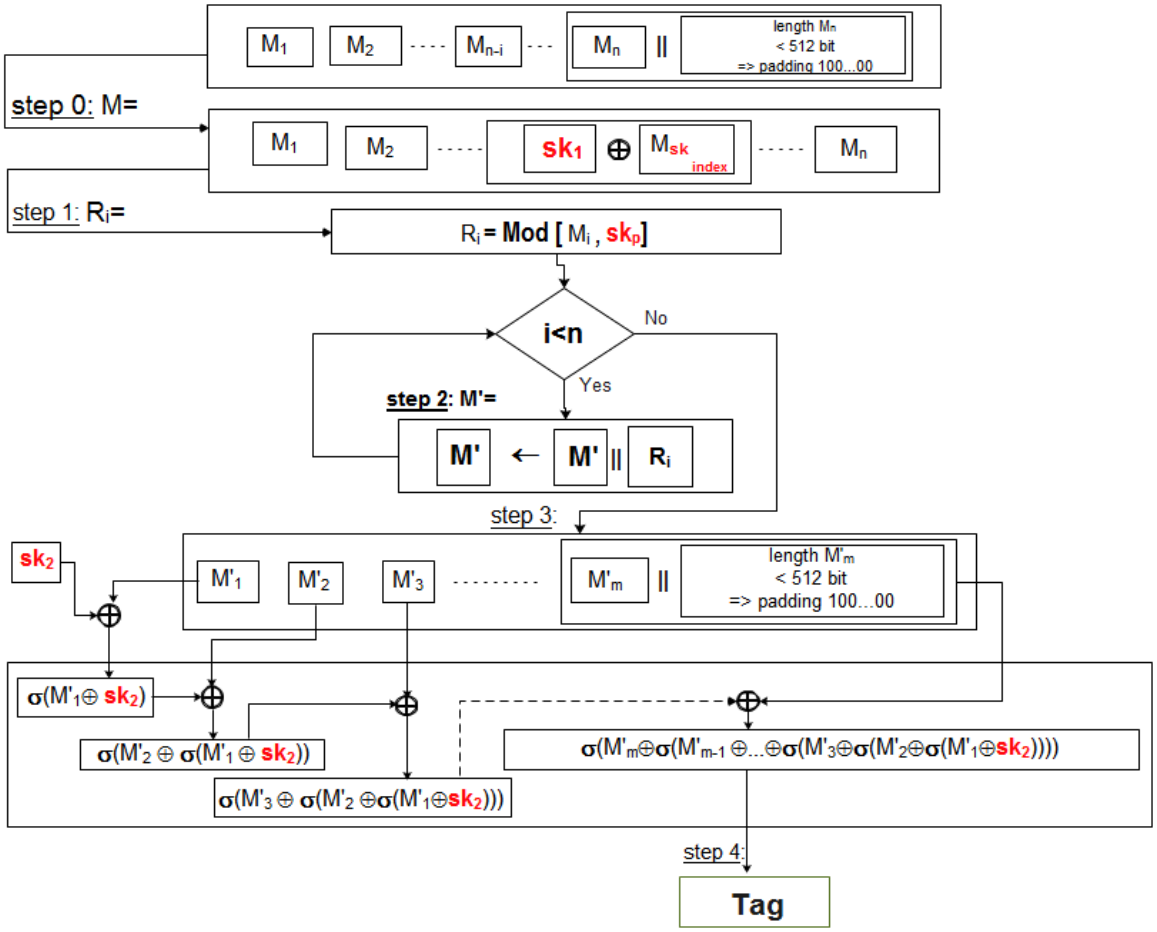

Figure 3.1 – Algorithme de génération de Tag proposé Generate-Keyed-CAHASH.

Algorithme 3.3 Algorithme de génération de Tag Generate-Keyed-CAHASH $(M, sk_1, sk_2, sk_{index}, sk_p)$

Input : Message $M, sk_1, sk_2, sk_{index}, sk_p$
Variable : n, m
Output : Tag
Split M into n blocks of 512 bit
si M≠ multiple of 512 bit **alors**
  Padd M
finsi
$M_{sk_{index}} \leftarrow M_{sk_{index}} \oplus sk_1$
pour i ←1 to n **faire**
  $R_i \leftarrow Mod[M_i, sk_p]$
fin pour
M'← $R_1 \| R_2 \| ... \| R_n$
Split M' into m blocks of 512 bit
si M' ≠ multiple of 512 bit **alors**
  Padd M'
finsi
$Tag \leftarrow \sigma(sk_2 \oplus M'_1)$
pour i ← 2 to m **faire**
  Tag ← $\sigma(M'_i \oplus Tag)$
fin pour

Notre algorithme de génération de MAC proposé (Generate-Keyed-CAHASH) (voir la figure 3.1 et l'algorithme 3.3) commence par diviser le message d'entrée M de longueur arbitraire L en n blocs de longueur fixe 512 bits, M_i, en utilisant également une opération de padding, si nécessaire. Par la suite, le message devient M = $\{M_1, ..., M_n\}$ avec M_i, i ème bloc de 512 bits. Ensuite, l'opération "XOR" est effectuée au $M_{sk_{index}}$ choisi, à partir du message M, avec la première sous-clé sk_1. En particulier, notre fonction de compression proposée consiste à calculer chaque M_i modulo sk_p, dénoté R_i. M', la concaténation de tous les résidus R_i, est par la suite divisée, lui aussi, en blocs de 512 bits, dénoté M'_i, en utilisant également une opération de padding, si nécessaire. Ensuite, le résultat de l'opération "XOR" effectué sur la deuxième sous-clé sk_2 avec le premier bloc de M'(i.e. $(M'_1 \oplus sk_2)$) est pris comme paramètre d'entrée pour la fonction d'évolution $\sigma()$ décrite ci-dessus. Ainsi de suite, le résultat final est évoluée tel que sur chaque sortie est effectuée une opération "XOR" avec tout le résultat précédent, jusqu'à la dernière itération (i.e.($M'_m \oplus ... \oplus \sigma(M'_2 \oplus \sigma(M'_1 \oplus sk_2))$) pour générer le Tag. La sortie de cet algorithme est d'une longueur de 512 bits.

$$Tag := Generate - keyed - CAHASH(M, sk_1, sk_2, sk_{index}, sk_p) \tag{3.2}$$

3.3.3 Algorithme de vérification Verify-Keyed-CAHASH

Cet algorithme de vérification déterministe prend comme entrée un message m∈M, une clé secrète k∈K et un Tag∈T (3.2) et produit un élément de l'ensemble valide, invalide.

Algorithme 3.4 Algorithme de vérification de tag Verify-Keyed-CAHASH

Input : $M, sk_1, sk_2, sk_{index}, sk_p, Tag$
Output : Valid$_{Tag}$, $Invalid_{Tag}$
Tag' ← Generate-Keyed-CAHASH($M, sk_1, sk_2, sk_{index}, sk_p$)
si $Tag' = Tag$ **alors**
  Valid$_{Tag}$
sinon
  $Invalid_{Tag}$
finsi

Une fois le récepteur reçoit le Tag, il recalcule le Tag' en utilisant l'algorithme de génération MAC. Par la suite, il utilise l'algorithme de vérification Verify-keyed-CAHASH () (voir Algorithme 3.4) en prenant comme entrées le message M, les clés secrètes $\{sk_1, sk_2, sk_{index}, sk_p\}$ et le Tag reçu 3.2. La sortie de cet algorithme est soit INVALID ou VALID, c'est à dire si les deux Tag sont égaux, l'expéditeur est authentifié et l'intégrité du message est garantie.

3.4 Conclusion

Dans ce chapitre nous avons donné une description détaillée de «keyed-CAHASH», notre nouveau Code d'Authentification de Message basé sur une fonction de Hachage. Par ailleurs, l'utilisation des automates cellulaires a fait de notre contribution un MAC rapide. En outre, notre Code d'Authentification de Message suggéré, comporte Trois algorithmes, un de génération de clés, un de génération de tag, et un de vérification ($MAC = \{SubkeyGeneration, Generate - Keyed - CAHASH, Verify - Keyed - CAHASH\}$).

Nous avons aussi décrit "PSOCA", le générateur pseudo aléatoire utilisé pour générer les sous clés de notre algorithme de génération de Tag.

La robustesse d'un Code d'Authentification de Message basé sur une fonction de Hachage repose dans la robustesse de la fonction de Hachage utilisé dans son processus. L'étude de robustesse de cette contribution ainsi que sa performance est étudiée dans le dernier chapitre.

CHAPITRE

4 CONTRIBUTION AUX CODES D'AUTHENTIFICATION DE MESSAGE LÉGERS BASÉS SUR LES FONCTIONS DE HACHAGE

Sommaire

4.1 Introduction

Au cours des dernières décennies, les nouvelles technologies de l'information ont connu le plus grand développement. Selon Cisco, d'ici 2020, il est prévu qu'il y aura 50 milliards d'appareils connectés. L'Internet des Objets (IoT) est un nouveau paradigme reliant le monde physique, à savoir la technologie informatique intelligente, le traitement intelligent, le réseau de communication, la technologie Internet et la technologie de détection du réseau de capteurs, au réseau. D'ailleurs, n'importe quel objet sera connecté à Internet. Admettant que ces objets pourraient être des dispositifs personnels comme les téléphones intelligents, les appareils photo numériques ou les objets de notre environnement comme une maison, une véhicule ou des objets avec des (RFID) [53]. En particulier, il intègre dans son processus un composant d'identification ("les objets qui identifient"), les réseaux de capteurs et les systèmes sans fil ("les objets sensibles"), les systèmes embarqués et les nanotechnologies ("les objets très petits").

Les RFID sont la composante la plus importante du processus de fonctionnement de l'IoT. Elles sont largement utilisées pour l'identification des objets physiques et sont universellement utilisées dans de nombreux domaines de notre vie quotidienne, par ex. le contrôle d'accès, la production

automatisée, la gestion du stationnement, etc. Ces RFID aident à identifier automatiquement le signal du Tag cible, sans contact, puis collectent et contrôlent les informations pertinentes, afin de transmettre les directives et instructions au contrôleur [54].

D'une part, ces RFID tiennent souffre de la taille de mémoire restreinte et de la puissance de calcul réduite. D'autre part, leurs principales menaces de sécurité sont principalement liées à l'authenticité, par ex. espionnage, rediffusion d'attaque, attaque de relais, contrefaçon, etc... [54]

- l'espionnage est une sorte d'attaque où l'entité malveillante tente de faire qu'un lecteur non autorisé interagisse avec le Tag puisque la plupart de ces Tags envoient leurs informations stockées sans aucune demande d'authentification pour accéder à des informations confidentielles.
- L'attaque par relecture est une sorte d'attaque où l'entité malveillante rejoue des informations clés qui sont sniffé depuis le canal entre le lecteur et le Tag, afin d'obtenir une authentification illégale et de tromper les lecteurs pour réussir la vérification.
- L'attaque par relais, également connue aussi sous le nom d'attaque «homme au milieu» [47], est une sorte d'attaque où l'entité malveillante pourrait interférer entre un serveur et un Tag et renvoyer la réponse d'un Tag obtenue lors des processus d'authentification avec un lecteur, de manière à faire semblant d'être le bon Tag.

Ainsi, le droit des personnes à la vie privée doit être garanti vis-à-vis ce grand nombre d'attaques. La nécessité de mettre en œuvre des primitives cryptographiques efficaces et peu coûteuses, en taille de mémoire et de calcul, augmente de manière exponentielle. La cryptographie fournit des mécanismes efficace garantissant la sécurité des données échangées. Assurer l'authenticité, dans le contexte des RFID, nécessite la conception d'algorithmes robuste qui peuvent fournir un équilibre entre assurer un haut niveau de sécurité et la capacité de calcul restreinte des RFID.

Basée sur les principes classiques de la cryptographie, la cryptographie légère a été proposée afin d'assurer la sécurité dans ces composantes à ressources limitée.

Généralement, la complexité temporelle d'un algorithme est mesurée en fonction du nombre de cycles d'horloge par octet et la latence [55].

Encore, la consommation d'énergie est une métrique universelle pour laquelle les unités utilisées sont Watts [55].

La complexité de calcul en mémoire est mesurée en fonction de la taille de RAM nécessaire pour effectuer le calcul et en fonction de la taille de ROM requis pour stocker les algorithmes. Aussi, la consommation d'énergie sur des appareils physiques (i.e. la mémoire) est mesurée en termes de portes logiques en unités de GE (Gate Equivalents) équivalentes à une porte NAND.

La complexité temporelle est mesurée en termes de débit, en bits par seconde pour une fréquence de 100Hz. La latence est également prise en compte comme pour la consommation de logiciels. Par ailleurs, Les mécanismes d'authentification classiques déjà existants sont très couteux en mémoire et énergie. Dans ce contexte, pour valider les exigences de sécurité en authenticité, nous avons proposé un nouveau mécanisme de Code d'Authentification de Message léger basé sur une fonction de hachage légère basée en utilisant les automates cellulaires pour RFID, nommé «LCAHASH-MAC». Dans ce chapitre, Nous donnons une description de notre contribution. Pour authentifier les données, ce nouveau MAC est basé sur une fonction de hachage légère qui sa conception repose sur l'utilisation des automates cellulaires. Cette contribution comporte trois algorithmes, un de génération de clés, un de génération de tag et un de vérification.

4.2 Description de notre Code d'Authentification de Message proposé : LCAHASH-MAC

La cryptographie légère [56] est un sous-domaine relativement jeune situé à l'intersection de la cryptographie et de l'ingénierie électrique.

En raison des contraintes des coûts de consommation élevée en énergie et en ressources d'une part, et d'autre part l'existence dans le monde d'entité malveillante forte en attaque, ont surgie un besoin croissant en implémentations efficaces de primitives cryptographiques et de protocoles légers. Pour ce faire, Chaque concepteur de solutions de sécurité légères doit être conscient de l'équilibre entre la sécurité, les coûts et la performance.

Habituellement, pour établir un compromis entre deux des trois objectifs de conception, à savoir la sécurité et les coûts bas, la sécurité et la performance, ou les faibles coûts et la performance, peut être facilement optimisé, alors qu'il est très difficile d'optimiser les trois objectifs de conception en même temps.

"LCAHASH-MAC" est une contribution aux Codes d'Authentification de Message légers, basé sur une fonction de hachage légère. Lors de génération de Tag, cette contribution utilise une nouvelle fonction de hachage légère "L-CAHASH" [17] ainsi que "PSOCA" [1].

"L-CAHASH" a été prouvé robuste et résistante aux collisions.

Par ailleurs, l'utilisation des automates cellulaires, dans leurs processus, offre une légèreté, une simplicité, une homogénéité et un parallélisme [15].

Aussi, l'utilisation du processus de génération de nombre pseudo aléatoire "PSOCA", prouvé imprévisible, a fait de cette contribution résistante aux attaques de récupération des clés.

4.2.1 Génération des sous-clés

l'ensemble de clés secrètes, $\{sk_1, sk_2, sk_{index}, sk_p\}$,utilisé dans pour notre mécanisme, est généré en utilisant l'algorithme 4.1), à savoir sk_1 et sk_2 sont générées en utilisant le Générateur de nombres pseudo-aléatoires "PSOCA " 3.3.1.1 [1] . L'algorithme "PSOCA" prend en entrée une graine, nombre binaire pour générer à partir de ceci une séquence de clés.

Algorithme 4.1 Algorithme de génération des sous clés Subkeys Generation(s, k_z,n)

input : seed s, key size k_z,number of blocks n
Output : sk_1, sk_2, sk_p , sk_{index}
- $sk_1 \leftarrow$ PSOCA(s,k_z) [1]
- $sk_2 \leftarrow$ PSOCA(sk_1, k_z) [1]
- $sk_p \leftarrow$ choose a prime number
- $sk_{index} \leftarrow$ choose an index $\in [0, 1]$

- Sk_p est un nombre secret qui est premier avec une longueur spécifié de bits (13 bits dans notre cas).
- Sk_{index} est un index secret choisi aléatoirement de $[0, n]$.
- Pour obtenir les sous-clés $Sk_s = Sk, Sk_1, Sk_2$, "PSOCA" prend deux paramètres, une graine (s) de taille (k_z) pour fournir la première sous-clé de longueur 512 bits Sk_1. Par la suite, "PSOCA" est évolué pour générer une deuxième clé Sk_2 de 512 bits à partir de Sk_1.

En outre, Pour chaque session, une fois que les sous-clés sk = $\{sk_1, sk_2, sk_{index}, sk_p\}$ (l'algorithme 4.1) sont générées et secrètement partagées (par exemple via un support sécurisé) entre les expéditeurs légitimes et les récepteurs. L'expéditeur peut générer le Tag en utilisant notre algorithme de génération de MAC proposé.

4.2.2 Algorithme de génération Generate-Keyed-LCAHASH

l'algorithme 4.2 d'authenticité probabiliste prend en entrée un message m∈M, une clé secrète k∈K et a et produit une valeur d'authenticité Tag∈T.

Notre algorithme de génération MAC proposé "Génération-Keyed-LCAHASH" (voir figure 4.1 et voir Algorithme 4.2) procède de la manière suivante. Prenons un message M, à authentifier, avec une longueur fini arbitraire L, notre MAC proposé commence tout d'abord par faire un découpage du message M en bloc de taille fixe de 512 bits M_i, en utilisant un padding, si

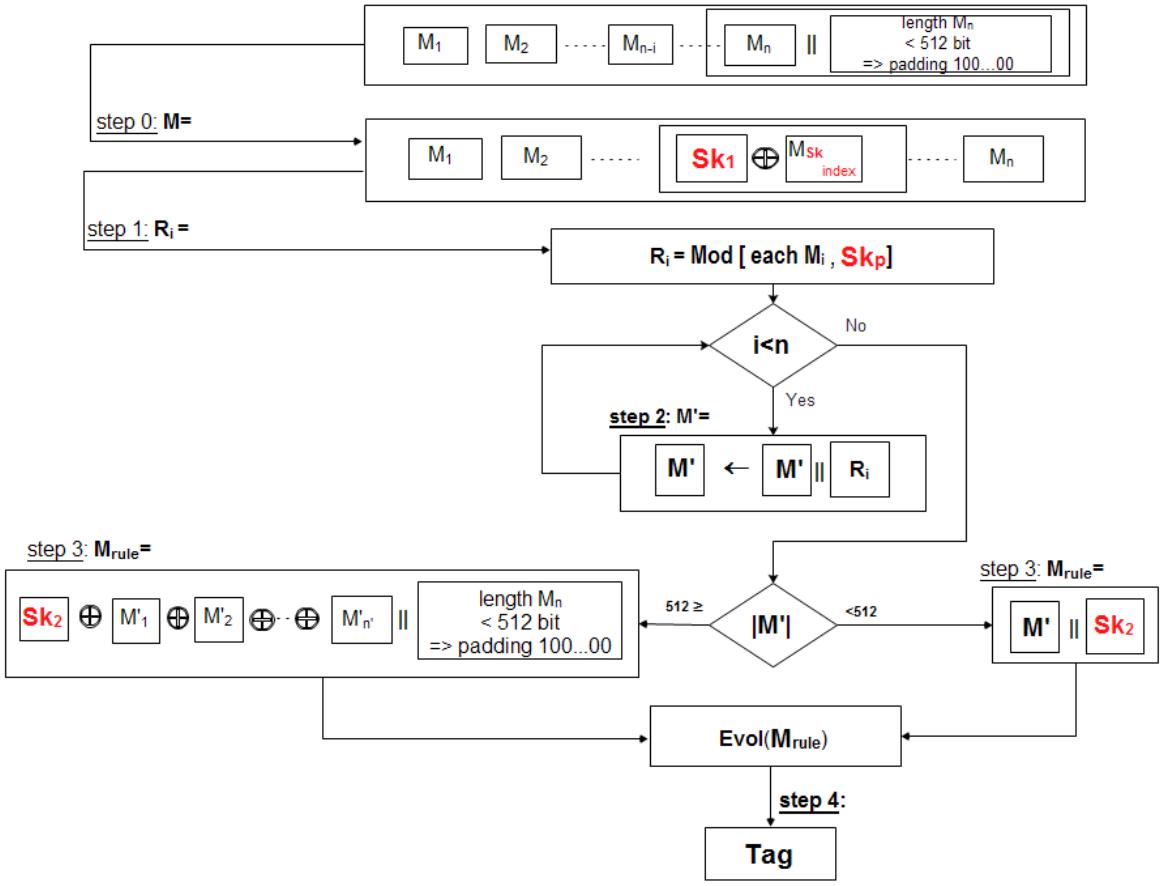

Figure 4.1 – Algorithme de génération de Tag proposé Generate-Keyed-LCAHASH.

nécessaire pour compléter le dernier bloc.

En supposant que la forme du message M après le padding est $M_{pad} = M_1, ..., M_n$, avec M_i est un bloc de 512 bits. Ensuite, l'opérateur 'XOR' est effectué sur sk_1 avec $M_{sk_{index}}$. Par la suite, chaque résidu R_i résultant de l'opération modulo de M_i avec sk_p est calculé. Le M' résultant qui est la concaténation de tous les résidus R_i, est découpé lui aussi en une longueur fixe de blocs de 512 bits, M'_i, en le complétant par un padding si nécessaire. Ensuite, une opération 'XOR' est appliquée à tous M'_i avec la deuxième sous-clé Sk_2. Sinon, si la longueur du M' global est inférieure à 512 bits, M' est concaténé avec une partie de la deuxième sous-clé $sk_2 \in \{0,1\}^{512-M'^{bit\_length}}$.

Dénoté M_rule, les bits obtenus après l'une de ces opérations seront la règle de la fonction de transition globale pour l'évolution des automates cellulaires : Evol(M_{rule}).

Le Tag est résultat final de cette évolution.

$$Tag := Generate - Keyed - LCAHASH(M, sk_1, sk_2, sk_{index}, sk_p) \tag{4.1}$$

Algorithme 4.2 Algorithme de génération de Tag Generate-keyed-LCAHASH($M, sk_1, sk_2, sk_{index}, sk_p$)

Input : Message M,$sk_1, sk_2, sk_{index}, sk_p$
Output : Tag
Split M into n blocks of 512 ;
si M $\neq$ multiple of 512 **alors**
  Pad M ;
finsi
$M_{\oplus} \longleftarrow M_{sk_{index}} \oplus sk_1$
pour each i in n **faire**
  $R_i \longleftarrow mod[M_i, sk_p]$
fin pour
$M' \leftarrow R_1 \| R_2 \| ... \| R_n$
Split M' into blocks of 512 bit
si $|M'| < 512$ **alors**
  $M_{rule} \longleftarrow M' \| sk_2$;
sinon
  pour i=1 to n **faire**
    M" $\longleftarrow \oplus M'_i$;
    $M_{rule} \longleftarrow M$" $\oplus\, sk_2$;
  fin pour
finsi
$Tag \longleftarrow \phi(M_{rule})$

4.2.3 Algorithme de vérification Verify-Keyed-LCAHASH

L'algorithme de vérification déterministe prend comme entrée un message m∈ M, une clé secrète k∈K et un Tag∈T (4.1) et produit un élément de l'ensemble valide, invalide.
Une fois que le destinataire reçoit le Tag, il calcule Tag' en utilisant l'algorithme4.2 de génération MAC. L'algorithme 4.3 de vérification "Verify-Keyed-LCAHASH" prend comme entrées le message M, les clés secrètes $\{sk_1, sk_2, sk_{index}, sk_p\}$ et le Tag reçu. La sortie de l'algorithme de vérification MAC est INVALID ou VALID. Si les deux valeur de Tag sont égaux, authenticité et l'intégrité du message est garantie.

Algorithme 4.3 Algorithme de vérification de Tag Verify-Keyed-LCAHASH

Input : $M, sk_1, sk_2, sk_{index}, sk_p, Tag$
Output : Valid$_{Tag}$, Invalid$_{Tag}$
$Tag' \leftarrow LCAHASH - MAC(M, sk_1, sk_2, sk_{index}, sk_p)$
si Tag' = Tag **alors**
  Valid$_{Tag}$
sinon
  Invalid$_{Tag}$
finsi

4.3 Conclusion

Dans ce chapitre j'ai donné un description des algorithmes de génération de clé, de génération de MAC et de vérification de "LCAHASH-MAC". En outre, l'utilisation des automates cellulaires a fait de notre contribution un MAC léger, rapide et robuste comparé aux MAC basés sur une fonction de hachage légère bien connus. Le RFID est un des éléments de base dans l'architecture de l'Internet des objets (IoT). C'est une technologie automatique d'identification, sans contact, visant à identifier le signal, afin de rassembler et envoyer les instructions de données au contrôleur. Néanmoins, d'une part, à cause de leur taille de mémoire restreinte et de leur puissance de calcul réduite, les RFID, souffrent de fréquentes menaces principalement liées à l'authenticité des données. D'autre part, les mécanismes d'authentification classiques déjà existants sont très couteux en mémoire et énergie. Pour cela, nous avons proposé un nouveau mécanisme, nommé "LCAHASH-MAC". Nous donnons une description de l'algorithme "LCAHASH-MAC". Pour authentifier les données, ce nouveau MAC est basé sur une fonction de hachage légère qui sa conception repose sur l'utilisation des automates cellulaires. La robustesse de cette contribution est étudiée dans le dernier chapitre aussi que sa performance.

CHAPITRE 5

ANALYSE DE LA SÉCURITÉ DE NOS CODES D'AUTHENTIFICATION DE MESSAGE PROPOSÉS ET ÉTUDE DE LEUR PERFORMANCE

Sommaire

5.1 Introduction

Dans ce chapitre nous présentons l'évaluation de la performance et de la résistance de chacune de nos trois contributions proposées aux attaques de récupération des clés et de falsification. Nous donnons aussi les résultats des expérimentations et tests effectués sur les mécanismes. En particulier, la sécurité d'un Code d'Authentification de Message repose sur la robustesse du système de chiffrement ou sur la fonction de hachage utilisé dans son processus. De plus, une entité malveillante, ayant pour but de fournir une valeur authentifiante valide, en essayant de forger un message, devrait avoir une complexité de temps qui est polynômiale [57]. En principe, cette entité malveillante peut prendre l'une des deux attaques, à savoir l'attaque de récupération des clés et l'attaque par falsification. En effet, l'attaque de récupération des clés consiste à trouver la clé secrète à partir d'un nombre de paires de messages/Tag. En addition, une attaque par falsification est une attaque qui consiste à forger un Tag valide, elle est dite existentiel lorsque

cette attaque est possible pour un seul message, et dite sélective s'il est possible pour un Tag de son choix [47].

5.2 Analyse de la sécurité

Dans cette partie nous évaluons la robustesse et le comportement aléatoire, des Tag générés par chacun de nos contributions proposées ainsi que ses propriétés statistiques.

5.2.1 La résistance de chacune de nos contributions à l'attaque de récupération de clé

5.2.1.1 SEC-CMAC

L'une des principales exigences de la sécurité des Codes d'Authentification de Message, surtout pour ceux basés sur les systèmes de chiffrement, est d'être fortement résistants aux attaques de recherche exhaustive de clé. Lors de ce type d'attaque, l'entité malveillante essaie toutes les clés possibles afin de récupérer la bonne. Considérant une machine à haute performance ayant besoin 10^{-10} secondes pour tester la validité d'une clé. En addition, étant donné la grande diversité des clés génétiques (SECkeys) qui sont de taille de $k \times 2^k$, où k est la taille des blocs en M_i et que le message est divisé en n blocs (le nombre totale des M_i). Aussi, la dernière clé utilisé k_{ranSEC}, est d'une longueur de 512 bits. Pour cela, une attaque de recherche de clé exhaustive nécessitera au moins environ $n \times k \times 2^k \times 4.24 \times 10^{136}$ (car $k > 3$ [50] et $n \times k \times 2^k \times 2^{512} \times 10^{-10}$ second))

Ainsi, une recherche de clé exhaustive est impossible dans un temps raisonnable par un ordinateur [47].

5.2.1.2 Keyed-CAHASH

L'un des principaux aspects de la sécurité d'un Code d'Authentification de Message, en particulier une fonction de hachage à clé, c'est d'être fortement résistant à l'attaque de récupération de clé.

La recherche exhaustive des clés :

Cette attaque est une des attaques où une entité malveillante essaie d'obtenir toutes les clés valides possibles en utilisant un outil de prédiction ou de calcul de clé. En conséquence, en tenant compte qu'une machine performante nécessite 10^{-10} secondes pour tester la validité des sous-clés. En addition, chaque sous-clés est de taille de 512 bits, notre algorithme peut utiliser $2 * 2^{512}$ clés possibles, plus l'indice n qui est choisi aléatoirement et le nombre premier p choisi aléatoirement parmi 2^{12} . Par conséquent, l'attaque de clé de recherche exhaustive nécessite environ $10^{-10} * 2 * 2^{512} + n + 2^{12}) = n * 10^{-10} + 2.681562e + 144 secondes$ $(plus de 8.49 * 10^{136} ans)$ pour obtenir la bonne clé. Ainsi, une recherche de clé exhaustive est impossible dans un temps raisonnable d'être calculé par un ordinateur [47].

la sensibilité de la clé au changement d'un bit :

Afin d'estimer le niveau de sensibilité de la clé en entrée à Keyed-CAHASH. L'expérimentation suivante est réalisée en prenant en considération la notion d'effet d'avalanche. Cette propriété prouve qu'une modification minime dans l'entrée entraîne une modification majeure dans la sortie, c'est-à-dire si un seul bit a changé dans l'entrée, la sortie est très affectée donnant une grande différence. En particulier, la moyenne de toute les distances de Hamming résultantes de toute les sorties obtenues devrait être la moitié de la taille de la sortie (i.e. $(Hamming(H(x), H(y)) = n/2)$)[58]. Le Tag généré en utilisant notre mécanisme proposé, compte tenu du changement aléatoire d'un seul bit dans la clé en entrée, en utilisant un ensemble de 1024 paires de clés, montré à la figure 5.1, indique que la moyenne des différences entre les bits de toutes les sorties est d'environ $\approx n/2$.

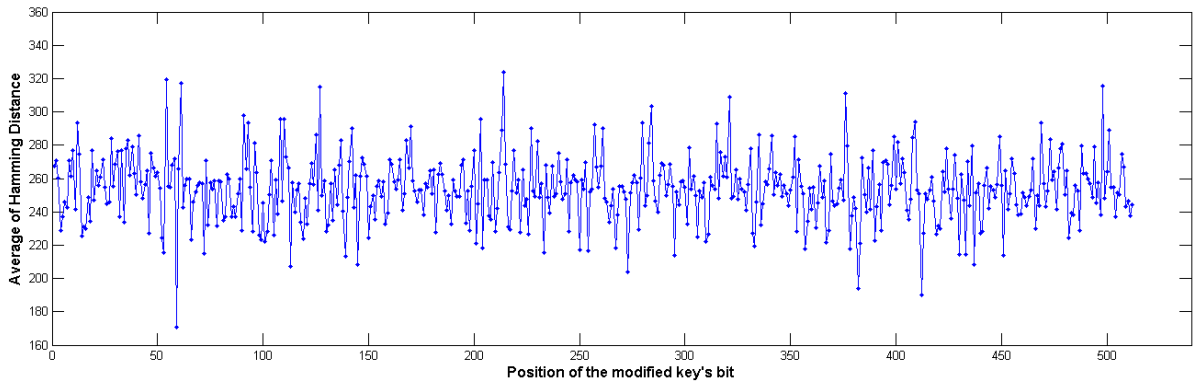

Figure 5.1 – Sensibilité du Tag en changeant la clé d'entrée d'un seul bit .

En outre, les résultats présentés dans le document [1] prouvent déjà que "PSOCA" possède des propriétés d'un générateur de nombres pseudo-aléatoires robuste. Ainsi, l'utilisation de "PSOCA" dans notre "Keyed-CAHASH" proposé sert bel et bien à générer des sous-clés imprévisibles [1].

5.2.1.3 LCAHASH-MAC

Cette attaque nécessitant environ 2^k opérations (k est la longueur de bit de la clé secrète). En utilisant une machine informatique puissante nécessitant 10^{-10} secondes pour tester la validité des sous-clés. Notre algorithme utilise deux sous-clés de longueur de 512 bits. Donc il existe $2*2^{512}$ clés possibles, n indice possibles et le nombre premier p est d'une longueur supérieure à 12 bit. En conséquence, cette attaque nécessite environ $10^{-10} * (2 * 2)^{512} + n + 2^{12}) = 2,69 * 10^{144}$ secondes (Plus de 8,49 $*10^{136}$ années) , pour obtenir la bonne clé. Ainsi, une recherche de clé exhaustive est impossible dans un temps raisonnable à l'aide d'un ordinateur [47].

la Sensibilité du Tag aux changements d'un seul bit en clé Afin d'évaluer le niveau de sensibilité du Tag en changeant aléatoirement un bit de la clé entrée à notre algorithme "Generate-LCAHASH-MAC", nous avons aussi utilisé la notion de l'effet d'avalanche.
La figure suivante 5.2 montre la moyenne de la distance de Hamming pour les changements d'un bit dans la clé entrée qui a induit un changement d'environ $\simeq 50\%$ dans les Tag de sortie.

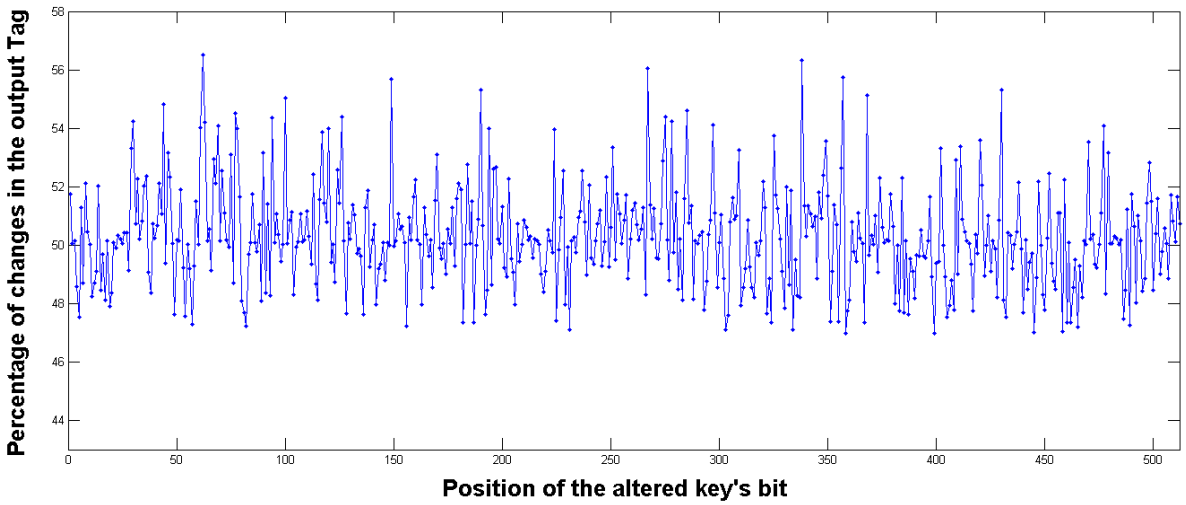

Figure 5.2 – Sensibilité du Tag en changeant la clé d'entrée d'un seul bit.

Par ailleurs, nous avons aussi ici utilisé le générateur pseudo aléatoire "PSOCA" pour sa

robustesse d'après les résultats présentés dans l'article [1].

5.2.2 La résistance de chacune de nos contributions à l'attaque par falsification

L'attaque par falsification est l'attaque la plus connue sur les MAC, où l'entité malveillante peut fournir au moins une paire de Message/Tag valide sans posséder déjà la clé secrète [47]. En outre, pour utiliser un MAC, il devrait être résistant à diverses attaques. La capacité d'un attaquant est affirmée par sa complexité temporelle qui doit être polynomiale [57].

En général, l'attaque la plus connue de l'attaque de falsification de MAC est l'attaque des anniversaires, qui est basée sur la fréquence de collisions trouvée. Cette attaque offre une limite maximale de sécurité. Elle nécessite d'utiliser $2^{n/2}$ paires de (Messages/Tag) connues (n étant la taille du Tag) [2].

5.2.2.1 SEC-CMAC

Puisque SEC-CMAC est décrit comme un MAC basé sur des évolutions itérées des blocs chiffrés, pour réussir une attaque de falsification de MAC sur notre système proposé, une entité malveillante doit connaître 2^{256} (Messages/Tag) [2].

La sensibilité du Tag en changeant le message d'un seul bit :

Afin d'évaluer le comportement aléatoire des Tags générés, par notre algorithme SEC-CMAC de génération de Tag, en entrant des messages différents d'un seul bit, nous avons utilisé la propriété d'effet Avalanche [58]. Pour cela, nous avons calculé la distance de Hamming moyenne entre la sortie d'une entrée donnée de longueur n et la sortie de cette entrée où certains des bits sont changés [58]. La distance moyenne devrait être changée d'environ 50%.

Pour évaluer la sensibilité du Tag en utilisant SEC-CMAC, nous avons pris en entrée un ensemble de 512 paires de messages différents d'un seul bit (différence aléatoire de bit) et leurs Tags sont générés, le résultat des expérimentations a montré que la distance de Hamming moyenne des bits de sortie changés est presque $\approx$50% (figure 5.3).

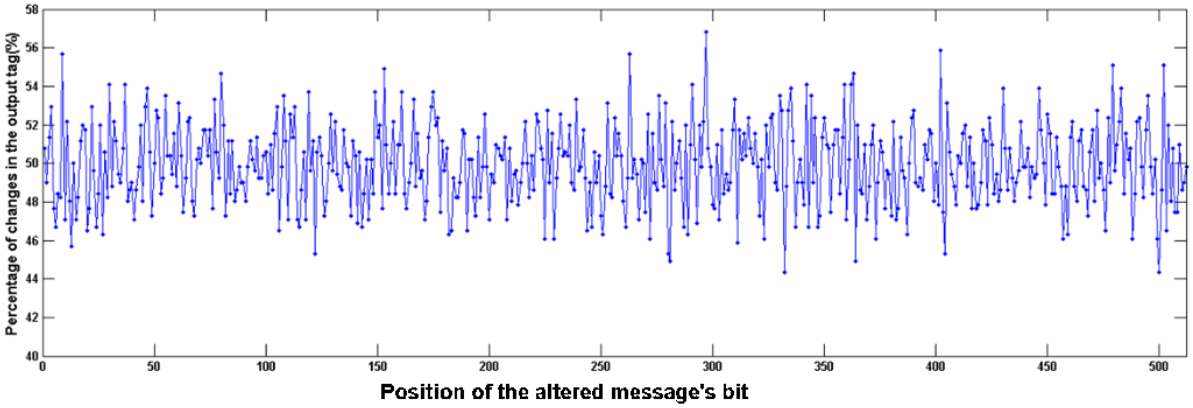

Figure 5.3 – Sensibilité du Tag en changeant le message d'entrée d'un seul bit .

tests statistiques :

Dans le but d'évaluer le comportement imprévisibles des Tags générés, du point de vue statistique, nous avons utilisé la suite statistique NIST STS pour les générateurs de nombres aléatoires et pseudo-aléatoires pour les applications cryptographiques [59].

Tableau 5.1 – Résultats de NIST STS Test Suite sur les Tags générés par SEC-CMAC

Le nom du Test	P-value	Interprétation
The Frequency (Monobit) Test	0.51	PASSE
Frequency Test within a Block	0.51	PASSE
The Runs Test	0.5	PASSE
Tests for the Longest-Run-of-Ones in a Block	0.49	PASSE
The Binary Matrix Rank Test		non applicable
The Discrete Fourier Transform (Spectral) Test	0.49	PASSE
The Non-overlapping Template Matching Test	0.61	PASSE
The Overlapping Template Matching Test		non applicable
Maurer's "Universal Statistical" Test		non applicable
The Linear Complexity Test	0.74	PASSE
The Serial Test	0.51	PASSE
The Approximate Entropy Test	0.46	PASSE
The Cumulative Sums (Cusums) Test	0.52	PASSE
The Random Excursions Test		non applicable
The Random Excursions Variant Test		non applicable

Cette suite est composée de 15 tests. Au cours de ces tests, nous avons utilisé la configuration par défaut : le niveau de confiance α A été maintenu à 1% et nous avons testé 100 séquences (de taille 1 000 000 bits chacune). Parmi ces tests, certains n'étaient pas applicables à notre système en raison de la taille des blocs.

Les résultats de ces tests sont présentés dans le tableau 5.1.

5.2.2.2 Keyed-CAHASH

En particulier, la robustesse d'un hachage à clé repose sur les propriétés cryptographiques robustes de la fonction de hachage, notamment sa résistance à la collision et sa propriété de compression [40]. Afin de concevoir un mécanisme de Code d'Authentification de Message basé sur une fonction de hachage, nous avons proposé l'utilisation d'une combinaison de règles d'AC appartenant à des groupes linéaires et non linéaires. Particulièrement, une règle d'AC appartenant à un groupe linéaire fournit la résistance à la collision et celle appartenant à un groupe non-linéaire fournit la propriété du sens unique [60]. Explicitement, la règle "150", basée sur une condition limite périodique, a la plus grande dépendance du voisinage par rapport à presque toutes les règles linéaires. D'autre part, la règle "23" fournit à la fois une non-linéarité élevée et une forme de transition spéciale [60]. La combinaison de ces deux règles ("150" et "23") garantit à la fois la linéarité et la non-linéarité, qui sont les propriétés les plus importantes de la cryptographie. Par conséquent, une faible linéarité et une non-linéarité maximale garantissent respectivement une résistance à la cryptanalyse différentielle et linéaire [61] ; [62].

Pour réussir une attaque de falsification de MAC sur notre système proposé, une entité malveillante doit connaître un nombre de 2^{256} de (Message/Tag) [2] .

la sensibilité du Tag au changement d'un bit du message initial entré :
En utilisant la notion d'effet avalanche [58]. Un message M est pris comme entrée et son Tag est généré à l'aide de "Keyed-CAHASH". Les résultats des expérimentations montrent que la moyenne de la distance de Hamming des bits de sortie a changé, en considérant le changement d'un seul bit aléatoire dans l'entrée, en utilisant un ensemble de 1024 paires de messages (voir la figure 5.4).

Le tableau 5.2 présente les valeurs d'effet d'avalanche (minmum, maximum, moyenne, pourcentage) de la variation d'un bit de l'entrée. Presque, $\approx 50\%$ dans les Tags de sortie sont affectés par la modification par rapport aux HMAC bien connus.

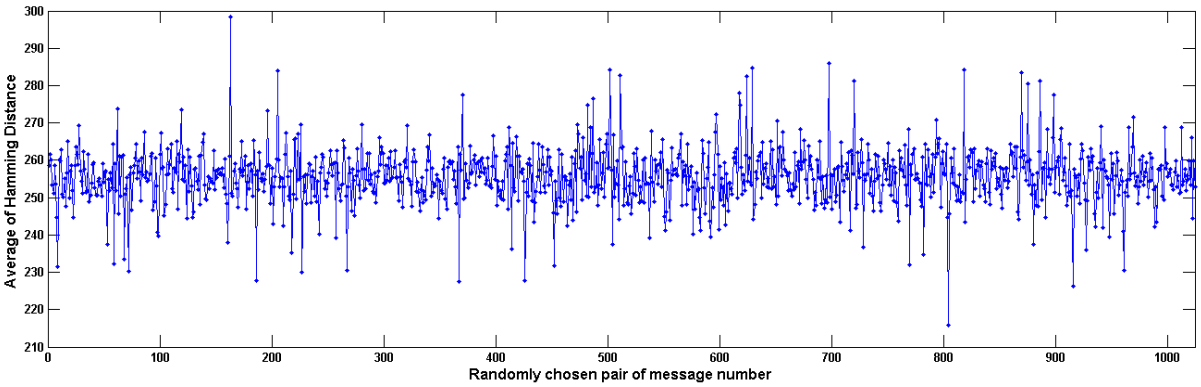

Figure 5.4 – Sensibilité du Tag en changeant le message entré d'un seul bit .

Tableau 5.2 – Sensibilité du Tag généré par Keyed-CAHASH par rapport à celle des HMAC connus

	A_{min}	A_{max}	A_{mean}	A(%)
HMAC(Sha2-512)	323.60	350.00	337.74	65.9027
HMAC(MD5)	76.90	91.50	84.92	66.3489
Keyed-CAHASH-512	215.90	298.50	255.71	49.94

les tests statistiques :

En principe, le Tag transféré devrait être imprévisible. Dans ce qui suit, nous présentons l'étude de qualité de l'aléatoire de notre mécanisme proposé en vérifiant s'il est résistant aux attaques statistiques standard.

Pour cette raison, il existe des piles de tests, à savoir le test "DIEHARD" [63], ayant pour objectif évaluer le caractère aléatoire d'un tel algorithme. Le test "DIEHARD" contient un ensemble de tests statistiques variés, chacun de ces tests a ses propres outils d'évaluation. Ces tests sont considérés comme forts pour vérifier l'aléatoire jusqu'à un niveau extrême. Pour obtenir de bons résultats. Il vérifie la valeur p, des nombres aléatoires, comprise entre $0.025 < p < 0.975$. Par conséquent, pour nos expériences, on génère de nombreuses synthèses afin d'obtenir un flux de données de 10Mb. Ce flux est créé à l'aide d'un schéma de mode compteur employant notre algorithme de génération de MAC proposé pour calculer les valeurs de sortie successives concaténées pour créer le flux de données. Par la suite, ce flux produit est statistiquement analysé à

Tableau 5.3 – Résultat du Test 'Diehard' sur Keyed-CAHASH

le nom du Test	le nombre P-value	Interprétation
diehard birthdays	0.90315496	PASSE
diehard operm5	0.87063499	PASSE
diehard rank 32x32	0.98995661	PASSE
diehard rank 6x8	0.29111246	PASSE
diehard bitstream	0.91261990	PASSE
diehard opso	0.43442292	PASSE
diehard oqso	0.53436364	PASSE
diehard dna	0.50201120	PASSE
diehard count 1s str	0.67388288	PASSE
diehard count 1s byt	0.37199848	PASSE
diehard parking lot	0.91507220	PASSE
diehard 2dsphere	0.86872396	PASSE
diehard squeeze	0.37402560	PASSE
diehard sums	0.03667907	PASSE
diehard runs	0.93794343	PASSE
diehard craps	0.38196019	PASSE
sts runs	0.56014598	PASSE

l'aide du test "Diehard" [63].

Ensuite, ces résultats finaux sont en moyenne, puis rapportés dans le tableau 5.3.

le Tag obtenu en utilisant notre conception proposée a passé avec succès presque tous les tests "DIEHARD" essentiels. Par ailleurs, notre mécanisme proposé vérifie un bon comportement aléatoire et peut être statistiquement indiscernable des séquences aléatoires.

5.2.2.3 LCAHASH-MAC

L'attaque de la falsification est l'attaque la plus connue sur la fonction MAC. Une contrefaçon existentielle de MAC est indiquée lorsque cette attaque est possible pour un seul message. En outre, on dit une falsification sélective de MAC si elle est possible pour une Étiquette de son choix [47].

Généralement, l'attaque de falsification la plus connue sur les MAC basée sur une fonction de hachage est l'attaque d'anniversaire basée sur la fréquence des collisions [47]. Elle nécessite la connaissance d'environ $O(2^{n/2})$ paire de Message/Tag (n la longueur du Tag) [2].

Par conséquent, pour réussir une attaque de falsification de MAC sur notre LCHAHASH-MAC proposée, une entité malveillante doit connaître 2^{256} paires de Messages/Tag [47], ce qui est impossible dans un temps raisonnable à l'aide d'une machine de calcul.

la sensibilité du Tag en changeant le message entré d'un bit :

En appliquant la notion d'effet d'avalanche [58] sur notre algorithme "Generate-LCAHASH-MAC" proposé pour évaluer la sensibilité du Tag en changeant aléatoirement un bit du message M entré. Les résultats présentés dans la figure 5.5 montrent que la moyenne de la distance de Hamming des bits des Tags affectés par la modification, parmi un ensemble de 1024 paires de messages, est d'environ $\simeq 50\%$.

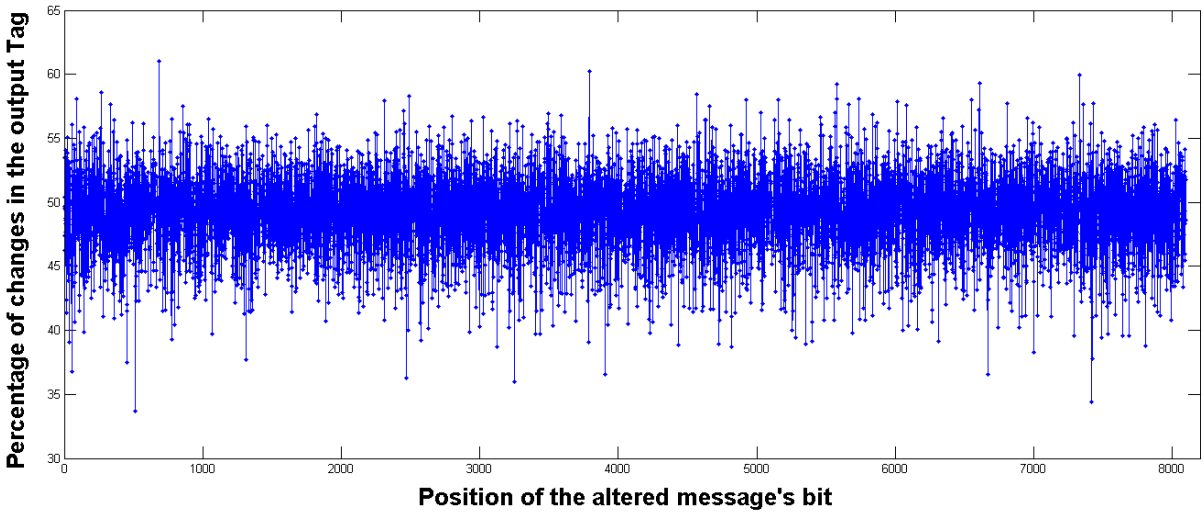

Figure 5.5 – Sensibilité du Tag en changeant le message d'entrée d'un seul bit.

les tests statistiques :

Pratiquement, la sortie d'une fonction MAC sécurisée doit être statistiquement indiscernable d'une sortie d'une fonction aléatoire et avoir un comportement aléatoire. En outre, la qualité aléatoire de notre "LCAHASH-MAC" proposé a été étudiée en testant si elle passe un certain nombre d'attaques statistiques standard. Pour ce faire, il existe un certain nombre de tests qui aident à évaluer le caractère aléatoire d'un tel algorithme, à savoir le test "DIEHARD" [63] puisqu'il comporte un ensemble de tests statistiques différents qui sont considérés comme forts pour vérifier l'aléatoire jusqu'à un niveau extrême. Nous prenons un flux de Tag de 10Mb généré à l'aide de notre algorithme (Generate-LCAHASH-MAC). Par la suite, ce flux est analysé statistiquement en utilisant le test "Diehard" [63]. Les résultats sont présentés sur le tableau 5.4.
Nous remarquons que notre algorithme a réussi à passer presque tous les tests "DIEHARD" essen-

Tableau 5.4 – Résultat du Test Diehard sur LCAHASH-MAC

le nom du test	P-value	Interprétation
diehard birthdays	0.27594168	PASSE
diehard operm5	0.63396067	PASSE
diehard rank 32x32	0.28117120	PASSE
diehard rank 6x8	0.33766941	PASSE
diehard bitstream	0.71348304	PASSE
diehard opso	0.11380714	PASSE
diehard oqso	0.15826437	PASSE
diehard dna	0.53281279	PASSE
diehard count 1s str	0.19919494	PASSE
diehard count 1s byt	0.93198603	PASSE
diehard parking lot	0.83952972	PASSE
diehard 2dsphere	0.45822316	PASSE
diehard 3dsphere	0.72299571	PASSE
diehard squeeze	0.22374292	PASSE
diehard sums	0.15384030	PASSE
diehard runs	0.76711679	PASSE
diehard craps	0.86268184	PASSE
marsaglia tsang gcd	0.32677634	PASSE
sts monobit	0.45515029	PASSE
sts runs	0.75671173	PASSE

tiels. Donc, nous pouvons conclure que LCAHSAH-MAC a effectivement un bon comportement aléatoire et pourrait être statistiquement indiscernable des séquences aléatoires.

5.3 La performance de nos contributions

5.3.1 SEC-CMAC

L'objectif de notre travail est de proposer un nouveau code efficace d'authenticité de message qui garantit l'intégrité et l'authenticité des données. Le tableau 5.5 montre que notre algorithme proposé SEC-CMAC a une grande diversité de clés par rapport aux autres Codes d'Authentification de Message basés sur des chiffres connus tout en ayant des performances satisfaisantes en implémentation logicielle simple. Pour un message M de longueur 2048 et m_i de taille 512, le SEC-CMAC a 4 sous clés de longueur 544.

Tableau 5.5 – Diversité de clés de SEC-CMAC par rapport aux autres CMACs connus

	taille de la clé (en bits)	le nombre de clés possibles
(AES-CMAC)	128, 192, 256	2^{128} , 2^{192} , 2^{256}
(TDES-CMAC)	256 (2 keys of 128 bit)	2^{256}
(SEC-CMAC)	544 (4 keys)	2^{544}

5.3.2 Keyed-CAHASH

Le but principal de notre travail est de proposer une nouvelle fonction de hachage à clé avec une vitesse de calcul élevée qui garantit l'intégrité et l'authenticité des données.
Notre nouveau MAC basée sur les automates cellulaires "Keyed-CAHASH" a été tourné sur un

Tableau 5.6 – Comparaison de la rapidité de Keyed-CAHASH à celle des autres HMACs connus

Message size (Kilo byte)	HMAC(Sha2-512) (s)	HMAC(MD5) (s)	KMAC(KeccakMAC-512) (s)	(Keyed-CAHASH) (s)
1	0.0136	0.0132	0.01257	0.0110
10	0.1981	0.1905	0.03318	0.02262
20	0.3850	0.2812	0.04041	0.04871

processeur Intel Core i5-34227, OS 64 bits et 1.8 GHz avec 4 Go de RAM. Le tableau 5 montre que notre algorithme proposé est rapide en comparaison avec les autres Codes d'Authentification de Message basé sur une fonction de hachage bien connus, tout en obtenant des performances satisfaisantes à l'aide de simples implémentations de logiciels.

5.3.3 LCAHASH-MAC

L'objectif principal de cette sous section est d'évaluer la performance de notre MAC léger avec une vitesse de calcul élevée garantissant au même temps l'authenticité des données. Par ailleurs, "LCAHASH-MAC" est basé sur une fonction de hachage rapide et facile à mettre en œuvre, avec une complexité déjà étudiée qui est égale à $O(n)$.
En outre, notre Code d'Authentification de Message léger proposé basé sur des automates cellulaires (Generate-LCAHASH-MAC) a été tourné sur un processeur Intel Core i5-34227, OS 64 bits, 1.8Ghz avec 4 Go de RAM. Le tableau 5.7 montre que notre mécanisme est rapide en comparaison avec les autres Codes d'Authentification de Message basés sur une fonction de hachage bien connus. Il fait preuve de bonnes performances en utilisant des implémentations logicielles faciles.

Tableau 5.7 – Comparaison de la rapidité de notre algorithme LCAHASH-MAC avec celle des autres HMACs

Message size (Kilo byte)	HMAC(Sha2-512) (s)	KMAC(KeccakMAC-512) (s)	LCAHASH-MAC(512) (s)
1	0.0136	0.0126	0.0103
10	0.1981	0.0332	0.0222
20	0.3850	0.0404	0.0364
Cycle per byte(cpb)	23906	22148	18105

5.4 Conclusion

Dans ce chapitre nous avons présenté une évaluation de sécurité et performance de nos MAC proposés 5.8. Offrant une efficacité élevée, nos MAC proposés se sont révélés robustes, satisfont à l'exigence d'intégrité et d'authentification des données, et résistent aux attaques de falsification et de récupération de clés par rapport à des MACs standards, bien connus.

Tableau 5.8 – Synthèse sur nos trois contributions par rapport aux autres MACs connus

Notre contribution	rapide	léger	diversité de la clé	comportement aléatoire
(SEC-CMAC)	-	-	√√√	√√√
(Keyed-CAHASH)	√	-	√√√	√√√
(LCAHASH-MAC)	√	√	√√√	√√√
KMAC(KeccakMAC-512)	√	√	-	-
AES-MAC	√	-	-	-

CONCLUSION

Synthèse

Les travaux de recherches de cette thèse s'inscrivent dans le cadre des Codes d'Authentification de Message.

A travers, la conception et la réalisation de trois nouveaux Codes d'Authentification de Message MAC, cette thèse vient enrichir le domaine de l'authentification marqué par une carence Codes d'Authentfication de Messages robustes, rapides et légers.

Nos contributions sont réparties en deux catégories : une se basant sur un système de chiffrement (SEC-CMAC) et deux autres s'appuyant sur une fonction de hachage (Keyed-CAHASH et LCAHASH-MAC). Dans un premier temps, nous avons donné une description des algorithmes de génération et de vérification de SEC-CMAC. Ce dernier est basé sur ESEC, une extension binaire du SEC, un des premiers systèmes de chiffrement à utiliser un algorithme évolutionniste.

Par la suite, nous avons donné la description détaillée de Keyed-CAHASH, un nouveau Code d'Authentification de Message construit à l'aide des automates cellulaires. L'utilisation de ces derniers fait de Keyed-CAHASH un MAC rapide. Ce Code d'Authentification de Message comporte trois algorithmes : un de génération de clés utilisant "PSOCA", un générateur pseudo aléatoire utilisé pour générer les sous clés et renforçant la résistance aux attaques de récupération de clé, un de génération de tag et un de vérification.

Enfin, nous avons présenté LCAHASH-MAC, la contribution concernant la cryptographie légère, dans le contexte des RFID, se basant sur une fonction de hachage et faisant appel aux automates cellulaires. Nous en avons présenté les algorithmes de génération de clé, de génération

de MAC et de vérification.

Pour évaluer nos trois MAC nous avons fait une étude de leur sécurité et de leur performance et ce par des outils théoriques et statistiques (NIST, Dihard).

Perspectives

Nos travaux ont aboutit à quelques perspectives à savoir :

- Les Codes d'Authentification de Message proposés ne garantissant pas la non-répudiation, nous proposons de combiner nos systèmes avec une clé publique et de concevoir de nouvelles signatures numériques.
- Bien que le mécanisme SEC-CMAC soit robuste, nous proposons d'améliorer sa rapidité en utilisant des techniques avancées telles que le parallélisme.
- Adapter nos contributions proposées dans d'autres contextes tels que : Big Data, Cloud Computing, etc.

LISTE DES PUBLICATIONS

Revues internationales

B. Echandouri, C.Hanin, F. Omary and S. Elbernoussi « Keyed-CAHASH : a New Fast Keyed Hash Function based on Cellular Automata for Authentication », ***International Journal of Computer Science and Applications (IJCSA)***, VOLUME 14, ISSUE 2, 2017

B. Echandouri, F. Omary, F.Z. Ziani, A. Sadak, « SEC-CMAC : A New Message Authentication Code Based on the Symmetrical Evolutionist Ciphering Algorithm », ***International Journal of Information Security and Privacy(IJISP)***, VOLUME 12, ISSUE 3, 2018.

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)***, soumis.

Conférences internationales

C.Hanin, **B. Echandouri**, F. Omary, and S. Bernoussi. « L-CAHASH : A Novel Lightweight Hash Function Based on Cellular Automata for RFID. », ***In International Sym-***

posium on Ubiquitous Networking, pp. 287-298. Springer, Cham, 2017.

B. Echandouri, C.Hanin, F. Omary and S. Elbernoussi « LCAHASH-MAC : A New lightweight Message Authentication code Using Cellular Automata for RFID. », ***Wincom'17***, pp.1-6,IEEE, 2017.

B. Echandouri, Y. Gahi, F. Omary, M. Guennoun « A Secure Processor Using Homomorphic Encryption », ***Networked Systems : 4th International Conference, NETYS 2016***,LNCS 9944,pp 374, Springer International Publishing, Marrakech, Morocco, May 18-20, 2016 .

Bibliographie

[1] Charifa Hanin, Fouzia Omary, Souad Elbernoussi, and Bouchra Boulahiat. 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.

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

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

[4] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Crypto*, volume 96, pages 1–15. Springer, 1996.

[5] Sang-Ho Shin, Dong-Hyun Kim, and Kee-Young Yoo. A lightweight multi-user authentication scheme based on cellular automata in cloud environment. In *Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on*, pages 176–178. IEEE, 2012.

[6] Susana Eiroa and Iluminada Baturone. Fpga implementation and dpa resistance analysis of a lightweight hmac construction based on photon hash family. In *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on*, pages 1–4. IEEE, 2013.

[7] Jian Guo, Thomas Peyrin, and Axel Poschmann. The photon family of lightweight hash functions. *Advances in Cryptology–CRYPTO 2011*, pages 222–239, 2011.

[8] Morris J Dworkin. Recommendation for block cipher modes of operation : The cmac mode for authentication. *Special Publication (NIST SP)-800-38B*, 2016.

[9] JH Song, R Poovendran, J Lee, and T Iwata. Rfc 4493 : The aes-cmac algorithm. Technical report, Technical report, Corporation for National Research Initiatives, Internet Engineering Task Force, Network Working Group, 2006.

[10] Lars R Knudsen and Bart Preneel. Macdes : Mac algorithm based on des. *Electronics Letters*, 34(9) :871–873, 1998.

[11] Zheng Yuan, Wei Wang, Keting Jia, Guangwu Xu, and Xiaoyun Wang. New birthday attacks on some macs based on block ciphers. In *CRYPTO*, volume 5677, pages 209–230. Springer, 2009.

[12] Bouchra Echandouri, Fouzia Omary, Fatima Ezzahra Ziani, and Anas Sadak. Seccmac a new message authentication code based on the symmetrical evolutionist ciphering algorithm, 2018.

[13] Fouzia Omary. *Application des algorithmes évolutionnistes à la cryptographie*. PhD thesis, Université Mohammed V-Agdal, Faculté des Sciences, Rabat, 2006.

[14] Bouchra Echandouri, Charifa Hanin, Fouzia Omary, and Souad Elbernoussi. Keyedcahash : a new fast keyed hash function based on cellular automata for authentication. *International Journal of Computer Science & Applications*, 12(2) :164–180, 2017.

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

[16] Bouchra Echandouri, Charifa Hanin, Fouzia Omary, and Souad Elbernoussi. Lcahashmac : A new lightweight message authentication code using cellular automata for rfid. In *Wireless Networks and Mobile Communications (WINCOM), 2017 International Conference on*, pages 85–91. IEEE, 2017.

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

[18] Sunny Behal and Krishan Kumar. Detection of ddos attacks and flash events using novel information theory metrics. *Computer Networks*, 116 :96–110, 2017.

[19] Nigel P Smart. Block ciphers and modes of operation. In *Cryptography Made Simple*, pages 241–269. Springer, 2016.

[20] Nigel P Smart. Historical stream ciphers. In *Cryptography Made Simple*, pages 179–194. Springer, 2016.

[21] William Stallings. The rc4 stream encryption algorithm. *Cryptography and network security*, 2005.

[22] Ross Anderson and Mike Roe. A5-the gsm encryption algorithm. *sci. crypt*, 1994.

[23] Scott R Fluhrer and Stefan Lucks. Analysis of the e˜ 0 encryption system. In *Selected Areas in Cryptography*, volume 2259, pages 38–48. Springer, 2001.

[24] Nigel P Smart. Public key encryption and signature algorithms. In *Cryptography Made Simple*, pages 313–347. Springer, 2016.

[25] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[26] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4) :469–472, 1985.

[27] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 313–314. Springer, 2013.

[28] Sénat. Loi n°2000-230 française du 13 mars 2000 portant adaptation du droit de la preuve aux technologies de l'information et relative à la signature électronique, 2000. (page consultée le 05/01/2011).

[29] David Chaum and Hans Van Antwerpen. Undeniable signatures. In *Conference on the Theory and Application of Cryptology*, pages 212–216. Springer, 1989.

[30] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[31] David Chaum and Eugène Van Heyst. Group signatures. In *Advances in Cryptology—EUROCRYPT'91*, pages 257–265. Springer, 1991.

[32] Patrick Gallagher. Digital signature standard (dss). *Federal Information Processing Standards Publications, volume FIPS*, pages 186–3, 2013.

[33] Sheila Frankel and Howard Herbert. The aes-xcbc-mac-96 algorithm and its use with ipsec. Technical report, 2003.

[34] Hans Delfs and Helmut Knebl. *Introduction to Cryptography : Principles and Applications*. Springer, 2015.

[35] Mihir Bellare. New proofs for nmac and hmac : Security without collision resistance. *Journal of Cryptology*, 28(4) :844–878, 2015.

[36] Alex Biryukov, Andrey Bogdanov, Dmitry Khovratovich, and Timo Kasper. *Collision attacks on AES-based MAC : Alpha-MAC*. Springer, 2007.

[37] Tetsu Iwata, Kazuhiko Minematsu, Thomas Peyrin, and Yannick Seurin. Zmac : a fast tweakable block cipher mode for highly secure message authentication. In *Annual International Cryptology Conference*, pages 34–65. Springer, 2017.

[38] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *Advances in Cryptology—EUROCRYPT 2002*, pages 384–397. Springer, 2002.

[39] Mihir Bellare, Roch Guérin, and Phillip Rogaway. Xor macs : New methods for message authentication using finite pseudorandom functions. In *Annual International Cryptology Conference*, pages 15–28. Springer, 1995.

[40] Hugo Krawczyk, Ran Canetti, and Mihir Bellare. Hmac : Keyed-hashing for message authentication. 1997.

[41] Mostafa Taha and Patrick Schaumont. Differential power analysis of mac-keccak at any key-length. In *International Workshop on Security*, pages 68–82. Springer, 2013.

[42] Daniel J Bernstein and Tung Chou. Faster binary-field multiplication and faster binary-field macs. In *International Workshop on Selected Areas in Cryptography*, pages 92–111. Springer, 2014.

[43] Teng Wu and Guang Gong. Two new message authentication codes based on apn functions and stream ciphers. *Security and Communication Networks*, 9(12) :1864–1871, 2016.

[44] Mihir Bellare, Daniel J Bernstein, and Stefano Tessaro. Hash-function based prfs : Amac and its multi-user security. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 566–595. Springer, 2016.

[45] Colin Boyd and Anish Mathuria. *Protocols for authentication and key establishment*. Springer Science & Business Media, 2013.

[46] Gustavus Simmons. Authentication theory/coding theory. In *Advances in Cryptology*, pages 411–431. Springer, 1985.

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

[48] William E Burr, Donna F Dodson, Elaine M Newton, Ray A Perlner, W Timothy Polk, Sarbari Gupta, and Emad A Nabbus. Sp 800-63-1. electronic authentication guideline. 2011.

[49] Enrique Alba and Marco Tomassini. Parallelism and evolutionary algorithms. *IEEE transactions on evolutionary computation*, 6(5) :443–462, 2002.

[50] Abdelaaziz Mouloudi, Fouzia Omary, Abderrahim Tragha, and Abdelghani Bellaachia. An extension of evolutionary ciphering system. In *Hybrid Information Technology, 2006. ICHIT'06. International Conference on*, volume 1, pages 314–321. IEEE, 2006.

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

[52] Stephen Wolfram. *Theory and applications of cellular automata*, volume 1. Advanced Series on Complex Systems, Singapore : World Scientific Publication, 1986, 1986.

[53] Louis Coetzee and Johan Eksteen. The internet of things-promise for the future ? an introduction. In *IST-Africa Conference Proceedings, 2011*, pages 1–9. IEEE, 2011.

[54] Ahmed Khattab, Zahra Jeddi, Esmaeil Amini, and Magdy Bayoumi. *RFID Security : A Lightweight Paradigm*. Springer, 2016.

[55] Wanican Julian Okello, Qingling Liu, Faizan Ali Siddiqui, and Chaozhu Zhang. A survey of the current state of lightweight cryptography for the internet of things. In *Computer, Information and Telecommunication Systems (CITS), 2017 International Conference on*, pages 292–296. IEEE, 2017.

[56] Axel York Poschmann. *Lightweight cryptography : cryptographic engineering for a pervasive world*. PhD thesis, Ruhr University Bochum, Germany, 2009.

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

[58] 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.

[59] Andrew Rukhin, Juan Soto, James Nechvatal, Elaine Barker, Stefan Leigh, Mark Levenson, David Banks, Alan Heckert, James Dray, and San Vo. Statistical test suite for random and pseudorandom number generators for cryptographic applications, nist special publication. 2010.

[60] Jun-Cheol Jeon. Analysis of hash functions and cellular automata based schemes. *International Journal of Security and Its Applications*, 7(3) :303–316, 2013.

[61] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In *Advances in Cryptology-CRYPTO*, volume 90, pages 2–21. Springer, 1991.

[62] Mitsuru Matsui. Linear cryptanalysis method for des cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 386–397. Springer, 1993.

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