République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

Université Mouloud Mammeri,Tizi-Ouzou.
Faculté de Génie Electrique et Informatique.
Département Informatique.

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option : Systèmes Informatiques

Une fonction de hachage basée sur les cartes chaotiques

Réalisé par :
M. Maloum Ahmed
M. Megraoui Mouloud

Encadré par :
M. Daoui Mehammed

Promotion : 2022/2023

Dédicace

“

On a le plaisir de dédier ce travail à ceux qui ont été une source d'inspiration et de soutien tout au long de notre parcours universitaire :

À nos précieux parents, dont aucune dédicace ne saurait exprimer nos sincères sentiments. Leur patience infinie, leurs encouragements constants et leur aide inestimable témoignent de notre profond amour et respect envers leurs sacrifices incommensurables.

À nos chers frères et sœurs, dont l'amour indéfectible et le soutien sans faille ont été notre force. C'est ici que nous exprimons notre immense gratitude envers eux. À toutes nos familles, sans exception, ainsi qu'à tous nos chers amis, pour leurs encouragements précieux et leur affection. Nous les incluons tous dans nos remerciements.

À nos enseignants, qui nous ont transmis leur savoir et nous ont guidés avec bienveillance tout au long de notre parcours académique.

À tous le personnel du département Informatique.

À toutes les personnes qui nous ont apporté de l'aide.

Merci à tous pour votre précieuse contribution à notre cheminement.

”

- Ahmed

- Mouloud

Remerciements

Nous tenons à saisir cette occasion et adresser nos sincères remerciements et nos profondes reconnaissances à Dieu le tout-puissant et miséricordieux qui nous a donné la force et la patience d'accomplir ce travail.

Nous tenons à remercier M. Daoui Mehammed notre promoteur, pour ses précieux conseils et son orientation avisés tout au long de notre recherche. Sa contribution a été d'une grande importance pour mener à bien ce projet.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions. Enfin, nous tenons également à remercier toutes nos familles, nos nos amis et toutes nos amies qui ont participé de près ou de loin à la réalisation de ce travail. Leur soutien moral et leur contribution ont été d'une grande valeur pour notre réussite.

Résumé

Une fonction de hachage est un algorithme qui transforme des données de taille variable en une empreinte digitale. Elle est utilisée pour vérifier l'intégrité des données, garantir l'authenticité des messages et faciliter la recherche de données. Une bonne fonction de hachage doit être rapide, produire des empreintes uniques et être résistante aux collisions.

Dans le cadre de nos recherches sur la sécurité, nous avons implémenté une fonction de de hachage qui est basé sur la construction de Merkle-Damgård en collaboration avec les cartes chaotiques. Nous avons commencé par diviser le message original en blocs de 16 octets (128 bits). Nous avons ensuite intégré une phase appelée prétraitement qui effectue des calcluls et qui a comme but d'éxtirper la condition initiale (CI) et de définir le nombre de rotations nécessaire. Puis nous avons implémenté la phase logistique qui aura comme sortie une séquence de clés à l'aide d'une fonction récursive qui seront utilisé par la fonction de compression. Enfin chaque bloc du message va etre compressé dans la phase de compression qui aura comme entrée le message et les clés sous forme d'un vecteur, puis des opérations seront éffectuéss pour qu'à la fin on aura le message chiffré final.

Mots clés : Fonction de hachage, Empreinte digitale, Intégrité des données, Authenticité, Collisions, Construction de Merkle-Damgård, Cartes chaotiques.

Abstract

A hash function is an algorithm that transforms variable-sized data into a fixed-size digital fingerprint. It is used to verify data integrity, ensure message authenticity, and facilitate data retrieval. A good hash function should be fast, produce unique fingerprints, and be resistant to collisions.

As part of our research on security, we have implemented a hash function based on the Merkle-Damgård construction in collaboration with chaotic maps. We started by dividing the original message into 16-byte (128-bit) blocks. We then incorporated a preprocessing phase that performs calculations to extract the initial condition (IC) and determine the required number of rotations. Next, we implemented the logistic phase, which outputs a sequence of keys using a recursive function that will be used by the compression function. Finally, each message block will be compressed in the compression phase, taking the message and keys as input in vector form, and performing operations to obtain the final encrypted message.

Keywords : Hash function, Digital fingerprint, Data integrity, Authenticity, Collisions, Merkle-Damgård construction, Chaotic maps.

Table des matières

Table des figures

Liste des sigles et acronymes

3DES	*Triple Data Encryption Standard*
A5/1	*Algorithm 5/1*
AES	*Advanced Encryption Standard*
CBC	*Cipher Block Chaining*
CFB	*Cipher Feedback*
DES	*Data Encryption Standard*
DSA	*Digital Signature Algorithm*
EC0	*Encryption 0*
ECB	*Electronic Code Book*
HMAC	*Hash-Based Message Authentication Code*
IPSec	*Internet Protocol Security*
MAC	*Message Authentication Code*
MD5	*Message Digest 5*
OFB	*Output Feedback*
PGP	*Pretty Good Privacy*
RC4	*Rivest Cipher 4*
RSA	*Rivest Shamir Adleman*
SHA-1	*Secure Hash Algorithm 1*

SHA-256 *Secure Hash Algorithm 256 bits*

SHA-3 *Secure Hash Algorithm 3*

SSL *Secure Sockets Layer*

SRTP *Secure Real-Time Transport Protocol*

SSH *Secure Shell*

TLS *Transport Layer Security*

VPN *Virtual Private Network*

ZRTP *Z Real-time Transport Protocol*

Introduction générale

Introduction générale

Dans le cadre de ce projet de fin d'études, nous explorons le domaine de la cryptographie, des systèmes dynamiques et des fonctions de hachage. La cryptographie joue un rôle essentiel dans la sécurisation des informations confidentielles, tandis que les systèmes dynamiques et les fonctions de hachage sont des concepts clés dans le domaine de la cryptographie moderne.

La première partie de ce projet se concentre sur la cryptographie. Nous examinerons les principaux services offerts par la cryptographie, ainsi que les différents types de cryptographie utilisés dans divers domaines. Nous explorerons également les domaines d'application du chiffrement et les applications des algorithmes de cryptage.

Dans cette deuxième partie, nous explorons en détail les systèmes dynamiques et les fonctions de hachage. Nous examinons les différents types de systèmes dynamiques, tels que les systèmes linéaires et non linéaires, à temps discret et à temps continu. Ensuite, nous plongeons dans les propriétés cryptographiques des fonctions de hachage, leur fonctionnement, ainsi que leurs utilisations et applications. Une attention particulière est accordée à la construction de Merkle-Damgård, un modèle largement utilisé, pour en comprendre les étapes principales, les domaines d'application et les propriétés cryptographiques. Nous effectuons également une comparaison approfondie entre cette construction et d'autres plus récentes, comme la construction à éponge. Cette partie nous permet d'approfondir notre compréhension des systèmes dynamiques, des fonctions de hachage et des avantages de la construction de Merkle-Damgård.

Nous nous pencherons sur l'étude de la carte logistique et son implication dans les problématiques actuelles liées aux fonctions de hachage dans cette troisiéme partie. Nous commencerons par définir la carte logistique et analyserons attentivement son comportement, en mettant un accent particulier sur l'exposant de Lyapunov. Ensuite, nous examinerons les limitations et les problèmes existants dans d'autres fonctions de hachage, ce qui nous permettra d'identifier la problématique à laquelle notre fonction de hachage proposée cherche à remédier. Nous présenterons les principes de fonctionnement ainsi que la construction de notre fonction de hachage innovante. Cette partie nous permettra d'approfondir notre compréhension de la carte logistique et de son rôle crucial dans le domaine des fonctions de hachage, tout en offrant une solution novatrice aux défis actuels.

Enfin la dernière partie de ce projet sera consacrée aux tests de notre fonction de hachage. Nous effectuerons des tests approfondis pour évaluer les performances, la sécurité et les propriétés cryptographiques de notre fonction de hachage. Ces tests nous permettront de démontrer l'efficacité et la fiabilité de notre fonction de hachage par rapport aux normes et aux attentes du domaine de la cryptographie.

Ce projet de fin d'études vise à approfondir notre compréhension de la cryptographie, des systèmes dynamiques et des fonctions de hachage, en explorant leurs concepts, leurs applications et leurs limitations. Nous espérons ainsi contribuer à l'amélioration de la sécurité des informations confidentielles et apporter de nouvelles perspectives dans ce domaine en constante évolution.

Chapitre 1

Introduction à la cryptographie

Introduction

Dans ce premier chapitre dédié à la cryptographie, nous explorons les principes fondamentaux de cette discipline et son importance dans la sécurisation des informations. Nous commençons par examiner les principaux services offerts par la cryptographie, tels que la confidentialité, l'intégrité, l'authentification et la non-répudiation, qui sont essentiels pour protéger les données sensibles. Ensuite, nous nous penchons sur les deux types de cryptographie les plus couramment utilisés, à savoir la cryptographie symétrique et asymétrique [1], en analysant leurs différences et en mettant en évidence leurs domaines d'application spécifiques. Nous passons ensuite en revue les différents algorithmes de cryptage les plus répandus, tels que le DES, l'AES, le RSA, et nous examinons leurs caractéristiques, leurs forces et leurs faiblesses. Enfin, nous explorons les divers domaines où le cryptage est largement utilisé, tels que les communications sécurisées, la protection des données personnelles, les transactions financières, et nous présentons des exemples concrets d'applications des algorithmes de cryptage dans ces domaines. Ce chapitre jette les bases nécessaires pour comprendre les concepts clés de la cryptographie et nous prépare à approfondir notre étude dans les chapitres suivants.

1.1 La cryptographie

C'est une technique d'écriture ou un message chiffré est écrit à l'aide de codes secrets ou des clés de chiffrement, et elle est utilisée pour protéger un message considéré comme confidentiel. Autrement dit, la cryptographie utilise des algorithmes de chiffrement pour transformer les données en texte clair (ou "plaintext") en texte codé (ou "ciphertext")[2]. Le processus de chiffrement utilise une clé pour coder les données, tandis que le processus de déchiffrement utilise une clé correspondante pour décoder les données et les restaurer en texte clair. D'une autre manière, la cryptographie est l'étude des méthodes de protection de l'information contre l'accès non autorisé. Elle concerne la création et l'utilisation de techniques de chiffrement pour sécuriser la communication entre des parties qui ne se font pas confiance et protéger la confidentialité, l'intégrité et l'authenticité des données.

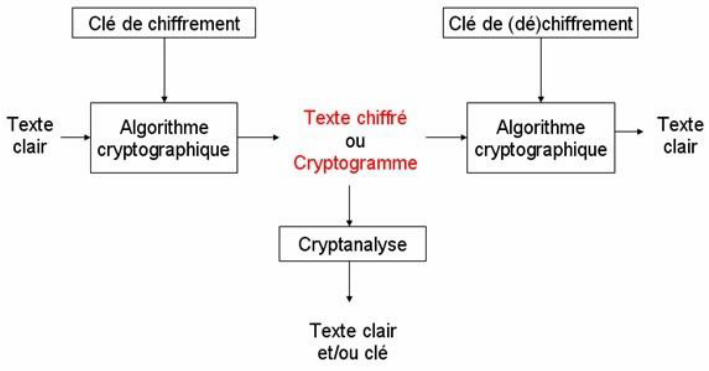

Fig. 1.1 : Protocle de chiffrement

1.1.1 Les principaux services de la cryptographie

La cryptographie est l'étude des techniques mathématique qui sont utilisés pour accomplir plusieurs objectifs pour garantir la sécurité de communication[3], ces objectifs sont :

• **La confidentialité :** la cryptographie permet de protéger les données contre l'accès non autorisé en les chiffrant à l'aide d'algorithmes de chiffrement. Seules les personnes possédant la clé de déchiffrement appropriée peuvent accéder aux données chiffrées (de telle sorte que seul le destinataire autorisé puisse les lire). D'un point de vue pratique, si A envoie un message à B, il ne faut pas qu'un attaquant voie le message pendant le trajet. Ceci nécessite une identification précise du destinataire, et une méthodologie permettant de rendre inutilisable l'information à tout autre qu'à ce destinataire. Cet objectif de la cryptographie est une base qui a toujours été traitée et exécutée dans toute l'histoire de la pratique cryptographique.

• **L'intégrité des données :** La cryptographie permet de garantir que les données n'ont pas été altérées ou corrompues pendant leur transmission ou leur stockage. On peut dire que c'est un mécanisme pour s'assurer que les données reçues n'ont pas été modifiées durant la transmission, frauduleusement ou accidentellement. Lorsque B reçoit un message de la part de A, il faut que B s'assure que A est bel est bien l'expéditeur.

• **L'authenticité :** La cryptographie permet de garantir l'authenticité des données en s'assurant que leur origine est vérifiée et que les données n'ont pas été modifiées depuis leur création. Lorsque B reçoit un message de la part de A, il faut que B s'assure qu'il n'a pas été modifié pendant le trajet.

• **La non répudiation :** La cryptographie permet de garantir que l'expéditeur ne peut pas nier avoir envoyé les données. A pour but enregistrer un acte ou un engagement d'une personne ou d'une entité de telle sorte que celle-ci ne puisse pas nier avoir accompli cet acte ou pris cet engagement.

1.1.2 Les types de cryptographie

Il existe deux types de cryptographie : la cryptographie symétrique et la cryptographie asymétrique.

1.1.2.1 Cryptographie symétrique

La cryptographie symétrique utilise une clé secrète unique pour chiffrer et déchiffrer les données. La clé de chiffrement et la clé de déchiffrement sont identiques et doivent être partagées entre les parties qui souhaitent communiquer de manière sécurisée[4]. Cette méthode est rapide et efficace pour le chiffrement de grandes quantités de données, mais la gestion de la clé secrète peut être complexe et représente un point faible de la sécurité. La sécurité de cette solution repose sur le fait que la clé est connue uniquement par l'émetteur et le récepteur du message.

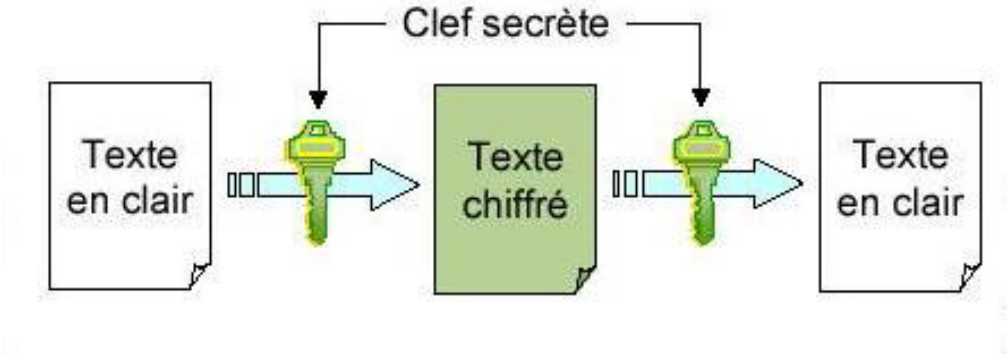

FIG. 1.2 : Principe de la cryptographie symétrique

Il existe plusieurs types d'algorithmes de cryptographie symétrique, qui diffèrent principalement par leur méthode de chiffrement. Voici les principaux types de cryptographie symétrique :

1.1.2.2 Chiffrement par substitution :

Ce type de chiffrement remplace chaque caractère ou groupe de caractères dans le texte clair par un autre caractère ou groupe de caractères. Il s'agit d'une méthode plus générale qui englobe le chiffrement par décalage[5]. En effet, à chaque lettre de l'alphabet on fait correspondre une autre, c'est-à-dire que l'on effectue une permutation de l'ensemble des lettres. Les algorithmes de chiffrement par substitution les plus courants sont le chiffrement de César, le chiffrement de Vigenère et le chiffrement de Playfair.

Chiffrement de césar :
Est une technique de chiffrement par substitution qui a été utilisée par Jules César pour protéger ses communications militaires. C'est l'un des systèmes les plus anciens et les plus simples est le codage par substitution mono alphabétique (ou alphabets désordonnés)[6]. Il consiste à remplacer chaque lettre par une lettre différente. Il existe donc grâce à ces techniques 26 façons de coder un message, ce qui fait que ce système a été longtemps utilisé par les armées pendant l'Antiquité. Ce procédé très fiable à l'époque est tout de même problématique car il nécessite que les interlocuteurs se souviennent tous deux de la clef. De plus, il est évident que la sûreté de ce codage est quasi nulle et qu'il pourrait être déchiffré par n'importe quelle personne qui y mettrait le temps nécessaire[7]. Voici les étapes pour chiffrer un message avec le chiffrement de César :

- Choisir un décalage (appelé clé) entre 1 et 25.
- Prendre le message à chiffrer et le diviser en lettres individuelles.
- Pour chaque lettre, la remplacer par la lettre qui se trouve à la position décalée dans l'alphabet. Par exemple, si la lettre est A et que le décalage est de 3, la lettre chiffrée sera D.
- Recomposer les lettres chiffrées pour former le message chiffré.

Chiffrement de Vigenère :
Le chiffrement de Vigenère est une amélioration décisive du chiffrement de César. Il a été élaboré par Blaise de Vigenère (1523-1596), diplomate français du XVIe siècle. Sa force réside dans l'utilisation non pas d'un, mais de 26 alphabets décalés pour chiffrer un message. Ce chiffrement utilise une clef qui définit le décalage pour chaque lettre du message (A : décalage de 0 cran, B : 1 cran, C : 2 crans, ..., Z : 25 crans)[8]. Voici les étapes pour chiffrer un message avec le chiffrement de Vigenère :

- Choisir une clé de chiffrement, qui est une série de lettres de la même longueur que le message à chiffrer.
- Diviser le message à chiffrer en blocs de la même longueur que la clé de chiffrement.
- Pour chaque bloc, appliquer un chiffrement de César avec un décalage déterminé par la lettre correspondante de la clé de chiffrement. Par exemple, si la première lettre de la clé de chiffrement est A, le décalage sera de zéro et aucune substitution ne sera effectuée. Si la deuxième lettre de la clé de chiffrement est B, le décalage sera de 1, si c'est C, le décalage sera de 2, et ainsi de suite.
- Recomposer les blocs chiffrés pour former le message chiffré complet.

Chiffrement de Playfair :
Le chiffrement de Playfair est une méthode de chiffrement par substitution polygraphique inventée par Charles Wheatstone en 1854, mais popularisée par Lord Playfair. Cette méthode utilise une matrice carrée de 5x5 lettres (sans les lettres J ou W) pour chiffrer les messages[8]. Voici les étapes pour chiffrer un message avec le chiffrement de Playfair :

- Choisir une clé, qui est une série de lettres qui seront placées dans la matrice carrée. Les lettres J et W sont généralement exclues, ce qui donne 25 lettres pour remplir la matrice.
- Placer les lettres de la clé dans la matrice, en commençant par la première lettre en haut à gauche, puis en remplissant la première ligne de gauche à droite, puis la deuxième ligne, etc.
- Diviser le message à chiffrer en paires de lettres (les lettres J et W sont traitées comme des I).
- Si les deux lettres de la paire sont identiques, insérer une lettre X entre elles (par exemple, "AA" deviendra "AXA").
- Pour chaque paire de lettres, trouver la position de chaque lettre dans la matrice. Les lettres sont chiffrées en utilisant les positions suivantes :

-Si les deux lettres sont sur la même ligne, remplacer chaque lettre par la lettre qui se trouve à sa droite dans la matrice (en considérant que la matrice est cylindrique, c'est-à-dire que la lettre à droite de la dernière lettre de la ligne est la première lettre de la même ligne).

-Si les deux lettres sont sur la même colonne, remplacer chaque lettre par la lettre qui se trouve en dessous dans la matrice (en considérant que la matrice est torique, c'est-à-dire que la lettre en dessous de la dernière lettre de la colonne est la première lettre de la même colonne).

-Si les deux lettres ne sont pas sur la même ligne ni sur la même colonne, remplacer chaque lettre par la lettre qui se trouve sur la même ligne mais dans la colonne de l'autre lettre.

1.1.2.3 Chiffrement par transposition :

Il utilise le concept mathématique des permutations. Un chiffrement à transposition est un chiffrement dans lequel les caractères du texte en clair demeurent inchangés mais dont les positions respectives sont modifiées. [9] Pour déchiffrer le texte chiffré, il suffit d'écrire verticalement celui-ci sur un morceau de papier quadrillé de la même largeur et de lire horizontalement le texte clair. Les algorithmes de chiffrement par transposition les plus courants sont la méthode de la colonne et la méthode du rail fence.

Transposition par colonne :
Consiste à réarranger les lettres d'un message en colonnes, en fonction d'une clé donnée. Un mot clé secret (avec uniquement des caractères différents) est utilisé pour dériver une séquence de chiffres commençant de 1 et finissant au nombre de lettres composant le mot clé. Cette séquence est obtenue en numérotant les lettres du mot clé en partant de la gauche vers la droite et en donnant l'ordre d'apparition dans l'alphabet. Une fois la séquence de transposition obtenue, on chiffre en écrivant d'abord le message par lignes dans un rectangle, puis on lit le texte par colonnes en suivant l'ordre déterminé par la séquence.

Rail fence :
consiste à écrire le message à chiffrer en zigzag sur plusieurs rails (lignes), puis à lire les lettres dans l'ordre des rails pour obtenir le message chiffré. Voici les étapes pour chiffrer un message avec l'algorithme rail fence :

- Choisir le nombre de rails, qui est généralement compris entre 2 et la moitié de la longueur du message à chiffrer.
- Écrire le message en zigzag sur les rails, en suivant l'ordre de lecture suivant :

-Écrire la première lettre du message sur la première ligne.

-Descendre d'un rail et écrire la deuxième lettre.

-Continuer en descendant d'un rail pour chaque lettre impaire du message, jusqu'à atteindre la dernière ligne.

-Remonter d'un rail et écrire la première lettre paire suivante sur cette ligne.

-Continuer en remontant d'un rail pour chaque lettre paire du message, jusqu'à atteindre la première ligne.

-Lire les lettres dans l'ordre des rails pour obtenir le message chiffré.

1.1.2.4 Chiffrement par flot :

Ce type de chiffrement utilise un générateur de nombres pseudo-aléatoires pour produire un flux de bits qui est combiné avec le texte clair pour produire le texte chiffré. Ce type de chiffrement permet de travailler sur un message de taille arbitraire, et le traiter octet par octet (voir bit par bit).[10] Autrement dit, le message est chiffré bit par bit en utilisant une clé de chiffrement et un XOR, ce type de chiffrement utilise un nombre aléatoire pour générer de nouvelles clés. Les algorithmes de chiffrement par flot les plus courants sont RC4, A5/1 et E0.

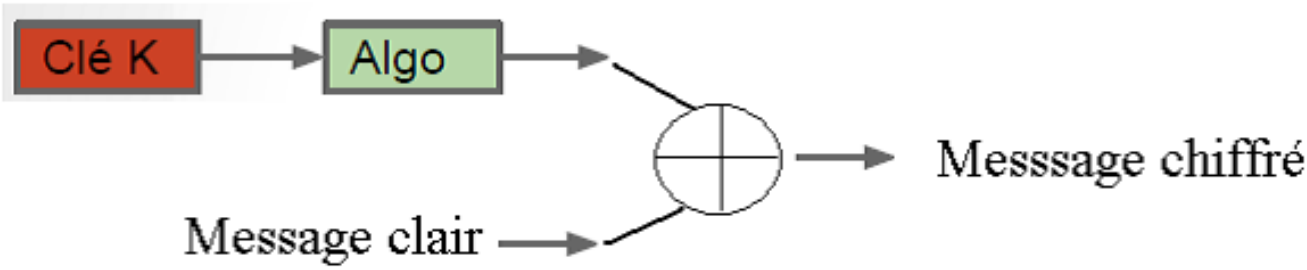

Fig. 1.3 : Principe du chiffrement par flot

L'algorithme RC4 :
C'est un algorithme de chiffrement à flux symétrique, inventé par Ron Rivest en 1987. Il est largement utilisé dans les protocoles de sécurité sur internet, tels que SSL/TLS, WEP et SSH. RC4 génère un flux de bits pseudo-aléatoires, qui est ensuite combiné avec le texte en clair pour produire le texte chiffré. La sécurité du RC4 repose sur la qualité de la génération du flux de bits pseudo-aléatoires, qui est basée sur une clé secrète.

L'algorithme A5/1 :
C'est un algorithme de chiffrement de flux utilisé dans les réseaux de téléphonie mobile pour chiffrer les communications. Il a été conçu pour être utilisé dans les réseaux GSM (Global System for Mobile Communications), qui sont les réseaux de téléphonie mobile les plus couramment utilisés dans le monde. L'algorithme A5/1 utilise une combinaison de registres à décalage à rétroaction linéaire (LFSR) pour générer une séquence pseudo-aléatoire qui est ensuite combinée avec les données à chiffrer pour produire des données chiffrées. Cette séquence pseudo-aléatoire est régénérée à chaque nouvelle transmission, ce qui garantit une sécurité renforcée des communications.

L'algorithme E0 :
Le chiffrement E0 est un algorithme de chiffrement utilisé dans les réseaux de téléphonie mobile de deuxième génération (2G) pour sécuriser les communications entre les téléphones portables et les stations de base. Il a été développé par l'European Telecommunications Standards Institute (ETSI) et est utilisé dans les réseaux GSM (Global System for Mobile Communications) et EDGE (Enhanced Data Rates for GSM Evolution). Cet algorithme génère une séquence de bits pseudo-aléatoires qui est combinée avec les données à chiffrer. La longueur de la séquence pseudo-aléatoire est déterminée par une clé

de chiffrement partagée entre le téléphone portable et la station de base. Cette séquence est ensuite combinée avec les données à chiffrer à l'aide d'une opération de OU exclusif (XOR), ce qui produit la sortie chiffrée[11].

1.1.2.5 Chiffrement par blocs :

Ce type de chiffrement divise le texte clair en blocs de taille fixe et applique un algorithme de chiffrement à chaque bloc de manière indépendante. D'une autre manière un message est découpé en blocs identiques en appliquant une clé pour chaque bloc et on obtient un message chiffré[12]. Les algorithmes de chiffrement par bloc les plus courants sont AES, DES, 3DES.

AES (Advanced enryption standard) :
AES a été adopté en 2001 par le NIST (National Institute of Standards and Technology) comme norme de chiffrement officielle du gouvernement américain. La taille du bloc est de 16 octets (128 bits), alors que la taille de la clé est de trois variantes (128, 192, 256 bits)[13]. Les opérations de chiffrement et de déchiffrement sont effectuées à l'aide d'un processus itératif qui combine des substitutions de bytes, des permutations de bits, des rotations de mots et des opérations arithmétiques[14]. Les étapes du chiffrement AES sont les suivantes :

• Clé d'expansion : La clé de chiffrement de tailles variables (128, 192 ou 256 bits) est étendue pour produire un ensemble de sous-clés qui seront utilisées dans les tours de chiffrement.

• Ajout de la clé initiale : Le bloc de données à chiffrer est combiné avec la première sous-clé en utilisant une opération XOR.

• Tours de substitution et permutation : Le bloc de données est soumis à un certain nombre de tours (10, 12 ou 14, en fonction de la taille de la clé de chiffrement) dans lesquels des opérations de substitution et de permutation sont effectuées.

-SubBytes : Les octets de données sont remplacés par d'autres octets en utilisant une table de substitution spécifique.

-ShiftRows : Les rangées de l'état sont décalées de façon circulaire d'un certain nombre de positions.

-MixColumns : Chaque colonne de l'état est mélangée en utilisant une opération matricielle.

-AddRoundKey : La sous-clé de tour correspondante est combinée avec l'état en utilisant une opération XOR.

• Dernier tour sans MixColumns : Dans le dernier tour, l'opération MixColumns n'est pas effectuée.

• Sortie du texte chiffré : Le texte chiffré final est obtenu en combinant l'état final avec la dernière sous-clé de tour en utilisant une opération XOR[15] .

Le processus de déchiffrement de l'AES est similaire au processus de chiffrement.

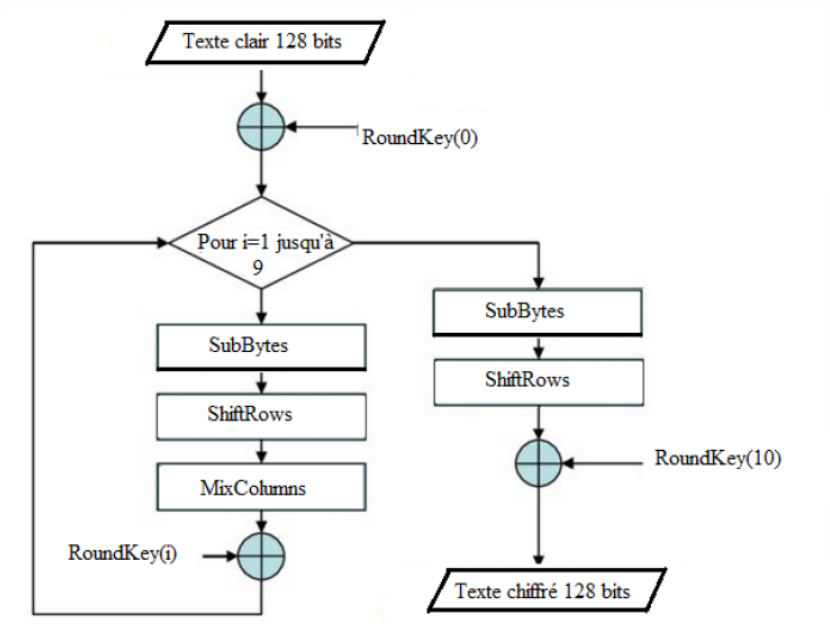

FIG. 6-2 – Principe du chiffrement AES.

FIG. 1.4 : Principe du chiffrement AES

DES (Data encryption standard) :
A été développé dans les années 1970 par IBM et adopté en 1977 comme norme de chiffrement du gouvernement américain. Le DES utilise une clé de chiffrement symétrique de 56 bits pour chiffrer les données en blocs de 64 bits. Ce système de chiffrement symétrique fait partie de la famille des chiffrements itératifs par blocs, plus particulièrement il s'agit d'un schéma de Feistel (du nom de Horst Feistel à l'origine du chiffrement Lucifer)[16]. Le processus de chiffrement DES peut être résumé en trois étapes principales :

- Initial permutation : Le bloc de données en clair est permuté selon une table fixe.
- Feistel network : Le bloc permuté est divisé en deux moitiés et les moitiés sont traitées à travers une série de 16 tours (rounds) en utilisant une combinaison de substitutions de bits et de permutations, ainsi que des opérations de XOR avec une sous-clé générée à partir de la clé de chiffrement principale.
- Final permutation : Le résultat final est permuté à nouveau selon une table fixe pour produire le bloc de données chiffrées[17].

Le processus de déchiffrement est similaire, mais utilise les opérations inverses dans l'ordre inverse.

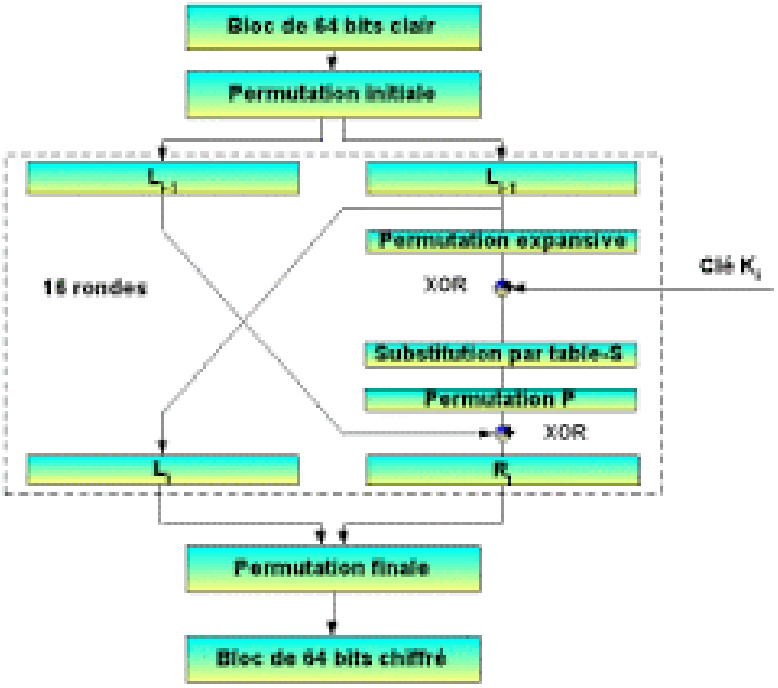

FIG. 1.5 : Principe du chiffrement DES

3DES (Triple data encryption standard) :
En cryptographie, Triple DES est également appelé algorithme de cryptage de données triple, qui est un chiffrement par blocs. Triple Data Encryptions Standard (3DES) a été publié pour la première fois en 1998 et tire son nom ainsi parce qu'il applique le chiffrement DES[18]. Le DES a été remplacé par Triple DES ou 3DES, qui est DES avec trois clés. Sa longueur de la clé est de 112 bits ou 168 bits et la taille du bloc est de 64 bits. En raison de la puissance de calcul croissante disponible ces jours-ci et de la faiblesse du chiffrement DES original, il a été soumis à des attaques par force brute et à diverses attaques cryptanalyse, Triple DES a été conçu pour fournir une méthode relativement simple pour augmenter la taille de la clé du DES pour se protéger contre de telles attaques[19]. L'idée est de chiffrer le message à trois reprises en utilisant trois clés DES différentes.

Le chiffrement 3DES peut être réalisé de deux manières différentes :

• 3DES avec deux clés : Dans ce mode de fonctionnement, le message en clair est chiffré avec la première clé DES, déchiffré avec la deuxième clé DES et chiffré à nouveau avec la troisième clé DES.

• 3DES avec trois clés : Dans ce mode de fonctionnement, le message en clair est chiffré avec la première clé DES, déchiffré avec la deuxième clé DES et chiffré à nouveau avec la troisième clé DES. Ensuite, le résultat est déchiffré avec la troisième clé DES, chiffré avec la deuxième clé DES et déchiffré à nouveau avec la première clé DES.

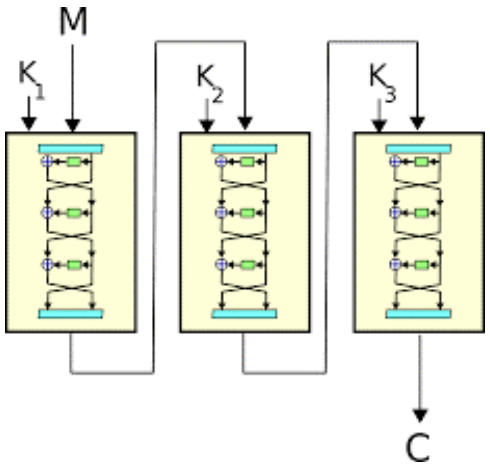

FIG. 1.6 : Principe du chiffrement 3DES

- Le chiffrement par blocs utilise des modes d'opérations, cependant les modes d'opération du chiffrement par bloc sont des méthodes pour chiffrer de grandes quantités de données en blocs de taille fixe en utilisant des algorithmes de chiffrement par blocs tels que DES, AES, etc. Les modes d'opération définissent la manière dont les données sont divisées en blocs, comment chaque bloc est chiffré et comment les blocs chiffrés sont combinés pour produire le résultat final[20].

Voici quelques-uns des modes d'opération les plus couramment utilisés : [21]

•**ECB (Electronic code book) :** Dans le mode ECB, chaque bloc de données est chiffré indépendamment des autres blocs en utilisant la même clé de chiffrement. Le processus de chiffrement est donc répétitif et indépendant pour chaque bloc de données. Les blocs de données identiques produiront les mêmes blocs chiffrés, ce qui rend le mode ECB vulnérable aux attaques.

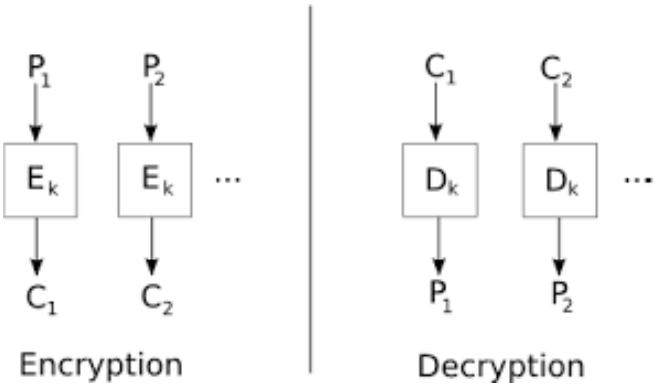

FIG. 1.7 : Le mode de chiffrement ECB

•**CBC (Cipher block chaining) :** Consiste, avant le chiffrement d'un bloc, à le masquer par le résultat du chiffrement du bloc précédent au moyen de l'opération XOR. Le premier bloc clair est lui aussi masqué, par une valeur habituellement notée IV (Initial Value) et de préférence variable (la date et l'heure peuvent faire une bonne (IV) pour que les chiffrements successifs du même message soient différents. La valeur initiale IV n'a pas besoin d'être secrète, et elle est en générale transmise en clair avant le message chiffré. Noter que si le destinataire reçoit un bloc chiffré avec des bits erronés, cela affecte

le déchiffrement de ce bloc et du suivant mais pas des autres.

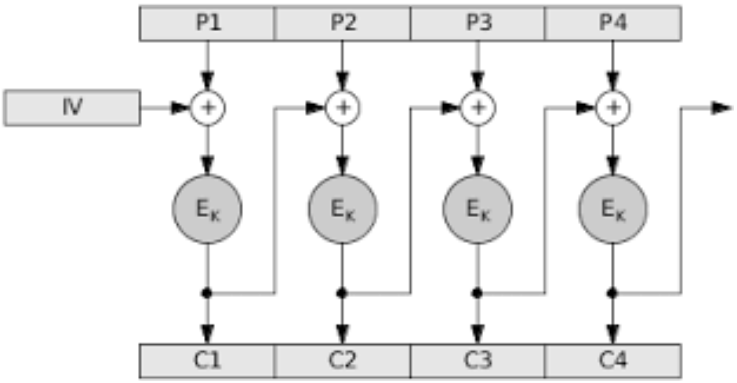

Fig. 1.8 : Le mode de chiffrement CBC

•**CFB (Cipher Feedback) :** Dans ce mode, les blocs chiffrés précédents sont utilisés pour chiffrer le bloc de données en clair actuel. Le résultat est XORé avec le bloc de données en clair pour produire le texte chiffré. Plus précisément, le mode CFB opère de la manière suivante :

-Un vecteur d'initialisation (IV) aléatoire de la même taille que le bloc de chiffrement est généré. -Le vecteur d'initialisation est chiffré avec la clé secrète pour produire un bloc de chiffrement initial. -Le bloc de texte clair suivant est combiné avec le bloc de chiffrement initial en utilisant une opération XOR pour produire un bloc de chiffrement. -Le bloc de chiffrement est stocké ou transmis, et il est également utilisé pour chiffrer le bloc de texte clair suivant en utilisant une opération XOR.

•**OFB (Output Feedback) :** est une variante de mode CFB précédemment abordé. Il est d'ailleurs parfois appelé internal feedback. Consiste à itérer la fonction de chiffrement sur une valeur initiale IV et à utiliser le flot de bits pseudo-aléatoires obtenus pour masquer les bits clairs à l'aide de l'opération XOR. Il est cette fois –ci très important que la valeur initiale IV soit différente pour chaque nouveau message. A noter que l'opération de déchiffrement est identique à celle de chiffrement, et utilise la fonction de chiffrement de bloc et non celle de déchiffrement.

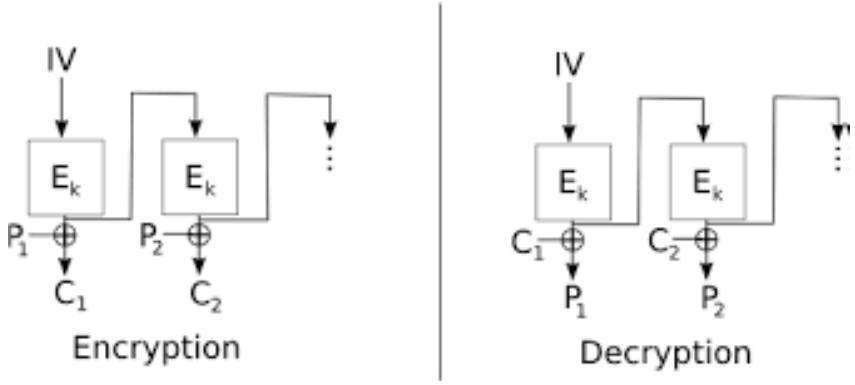

Fig. 1.9 : Le mode de chiffrement OFB

1.1.2.6 Cryptographie asymétrique

Egalement appelée cryptographie à clé publique, utilise deux clés différentes pour le chiffrement et le déchiffrement des données : une clé publique et une clé privée. La clé publique est largement diffusée et permet à quiconque de chiffrer des données à destination du propriétaire de la clé privée [22]. La clé privée est conservée secrète par le propriétaire et est utilisée pour déchiffrer les données chiffrées à l'aide de la clé publique. Cette méthode est plus sûre pour la gestion des clés, car la clé privée n'a pas besoin d'être partagée. D'une manière plus pratique, Si A veut envoyer un message à B, donc B va devoir générer 2 clés (privée et publique) et envoie sa clé publique à A qui va chiffrer le message à l'aide de la clé publique, donc si le message est intercepté par un attaquant, il ne peut pas être déchiffré car il ne connaît pas la clé privée pour ce faire[23].

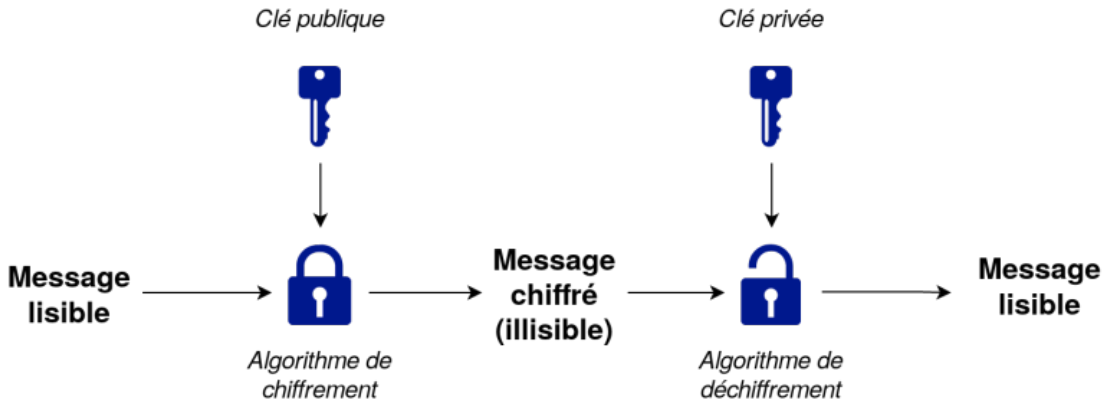

Fig. 1.10 : Principe de la cryptographie asymétrique

Il existe plusieurs types d'algorithmes de cryptographie symétrique, qui diffèrent principalement par leur méthode de chiffrement. Voici les principaux types de cryptographie asymétrique :

RSA (Rivest Shamir Adlmen) :
Il a été présenté en 1977 par Ron Rivest, Adi Shamir, et Len Adlmen. RSA utilise une paire de clés, une clé publique et une clé privée, pour chiffrer et déchiffrer des données, ou pour signer et vérifier des signatures numériques. La clé publique est utilisée pour chiffrer les données et est connue de tous, tandis que la clé privée est utilisée pour déchiffrer les données et est gardée secrète[23].

Voici les étapes principales de l'algorithme RSA :

• Génération de clés : Tout d'abord, l'utilisateur génère une paire de clés RSA en choisissant deux nombres premiers distincts p et q. Les deux nombres premiers sont multipliés pour former un nombre n = p * q, qui est utilisé comme module de chiffrement. La clé publique est alors générée en choisissant un nombre e qui est premier par rapport à (p-1) * (q-1), et qui satisfait 1 < e < (p-1) * (q-1). La clé privée est ensuite calculée à partir de la clé publique à l'aide d'un algorithme de calcul de l'inverse modulaire.

• Chiffrement : Pour chiffrer un message m, l'utilisateur calcule d'abord la valeur chiffrée c à l'aide de la clé publique en utilisant la formule c = m^e mod n.

• Déchiffrement : Pour déchiffrer le message chiffré c, l'utilisateur utilise la clé privée pour calculer la valeur déchiffrée m à l'aide de la formule m = c^d mod n. [24]

DSA (Digital Signature Algorithm) :
Développé par le National Institute of Standards and Technology (NIST) aux États-Unis. DSA est un algorithme de signature numérique basé sur le problème du logarithme discret. Il est utilisé pour garantir l'authenticité et l'intégrité des données échangées entre les parties [24].

Voici les étapes principales de l'algorithme DSA :

• Génération de clés : Tout d'abord, l'utilisateur génère une paire de clés DSA en choisissant un nombre premier p, un générateur g de l'ensemble multiplicatif Zp*, et deux nombres aléatoires x et y tels que 0 < x,y < p-1. La clé publique est alors générée à partir de p, g et y, et la clé privée est x.

• Signature : Pour signer un document m, l'utilisateur calcule un nombre de hachage h(m) à partir du document, puis choisit un nombre aléatoire k tel que 0 < k < p-1 et gcd(k, p-1) = 1. L'utilisateur calcule ensuite un nombre r à partir de k et h(m) en utilisant la formule r = (g^k mod p) mod q, où q est un nombre premier qui divise p-1. L'utilisateur calcule ensuite un nombre s à partir de x, k, h(m) et r en utilisant la formule s = (k^{-1} * (h(m) + x*r)) mod q. La signature du document est alors (r, s).

• Vérification : Pour vérifier la signature d'un document, le destinataire calcule d'abord un nombre de hachage h(m) à partir du document. Il vérifie ensuite que les valeurs r et s de la signature sont toutes deux comprises entre 1 et q-1. Le destinataire calcule ensuite un nombre w à partir de s en utilisant la formule w = s^-1 mod q, puis calcule un nombre u1 et un nombre u2 à partir de r et h(m) en utilisant les formules u1 = (h(m) * w) mod q et u2 = (r * w) mod q. Le destinataire calcule alors un nombre v à partir de la clé publique, u1, u2 et g en utilisant la formule v = ((g^{u1} * y^{u2}) mod p) mod q. La signature est considérée comme valide si v est égal à r [25].

Diffie-Hellman :
Utilisé pour l'échange de clés. Il permet à deux parties de générer une clé secrète commune sans avoir besoin de se transmettre cette clé secrète directement. Au lieu de cela, les deux parties utilisent une paire de clés publiques et privées pour générer une clé de session commune, qui peut ensuite être utilisée pour chiffrer et déchiffrer les données échangées entre les deux parties [26].

Les étapes de l'algorithme sont les suivantes :

• Les deux parties, Alice et Bob, conviennent d'un grand nombre premier p et d'un nombre entier g relativement premier à p. Ces valeurs sont publiques.

• Alice choisit un nombre secret a et calcule g^a mod p. Elle envoie cette valeur à Bob.

• Bob choisit un nombre secret b et calcule g^b mod p. Il envoie cette valeur à Alice.

• Alice calcule (g^b mod p)^a mod p, qui est égal à g^{ab} mod p.

• Bob calcule (g^a mod p)^b mod p, qui est également égal à g^{ab} mod p.

ElGamal :
Inventé par Taher Elgamal en 1985. Il est similaire à l'algorithme de Diffie-Hellman dans la mesure où il permet d'échanger des clés secrètes, mais il fournit également une fonction de chiffrement[27].

L'algorithme de chiffrement ElGamal fonctionne de la manière suivante :

• Alice génère un grand nombre premier p et un nombre entier g qui est un générateur du groupe dans Zp*. Elle choisit également un nombre entier aléatoire a comme clé privée et calcule g^a mod p. Elle publie p, g et g^a mod p comme sa clé publique.

• Pour chiffrer un message m, Bob choisit un nombre entier aléatoire k et calcule c1 = g^k mod p et c2 = m*$(g^a)^k$ mod p. Il envoie le couple (c1, c2) à Alice.

• Pour déchiffrer le message, Alice calcule $(c1^a)^{-1}$*c2 mod p, qui est égal à m.

1.2 Les domaines d'utilisation du cryptage

Les algorithmes de cryptage sont utilisés dans de nombreux domaines pour protéger les données sensibles. Voici quelques-uns des domaines d'utilisation les plus courants :

•**Sécurité des communications :** Les algorithmes de cryptage sont utilisés pour sécuriser les communications entre deux parties. Les protocoles de sécurité couramment utilisés incluent SSL/TLS pour les connexions HTTPS, SSH pour les connexions distantes sécurisées, et IPSec pour les réseaux privés virtuels.

•**Stockage de données :** Les algorithmes de cryptage sont utilisés pour sécuriser les données stockées sur des supports tels que des disques durs, des cartes mémoire ou des clés USB. Les algorithmes couramment utilisés pour le cryptage de disque incluent BitLocker pour Windows et FileVault pour Mac OS.

•**Gestion d'identité :** Les algorithmes de cryptage sont utilisés pour protéger les informations d'identification, telles que les noms d'utilisateur et les mots de passe, qui sont stockés dans les bases de données des systèmes d'authentification. Les algorithmes de hachage unidirectionnels sont utilisés pour stocker les mots de passe de manière sécurisée.

•**Protection de la propriété intellectuelle :** Les algorithmes de cryptage sont utilisés pour protéger la propriété intellectuelle, telle que les logiciels, les brevets et les marques. Les algorithmes de signature numérique sont utilisés pour garantir l'authenticité des documents numériques.

•**Services financiers :** Les algorithmes de cryptage sont utilisés pour protéger les transactions financières, telles que les transferts électroniques et les achats en ligne. Les protocoles couramment utilisés incluent le protocole de sécurisation des transactions sur Internet (Secure Electronic Transaction, SET) et le protocole 3-D Secure pour les achats en ligne.

•**Sécurité des systèmes embarqués :** Les algorithmes de cryptage sont utilisés pour sécuriser les systèmes embarqués tels que les cartes à puce et les systèmes de contrôle industriel. Les algorithmes couramment utilisés incluent DES, AES et RSA.

•**Sécurité des réseaux :** Les algorithmes de cryptage sont utilisés pour protéger

les réseaux contre les attaques telles que les intrusions, les écoutes et les usurpations. Les protocoles couramment utilisés incluent IPSec, SSL/TLS et SSH.

1.3 Les applications des algorithmes de cryptage

Les algorithmes de cryptage sont utilisés dans de nombreuses applications pour protéger les données, les communications et les transactions. Voici quelques exemples d'applications d'algorithmes de cryptage en détail :

•**Cryptage des disques durs :** Les algorithmes de cryptage, tels que AES et Twofish, sont utilisés pour chiffrer les disques durs et protéger les données stockées contre les accès non autorisés. Les données sont chiffrées avant d'être stockées sur le disque dur, et elles ne peuvent être déchiffrées que par les utilisateurs autorisés qui ont le mot de passe ou la clé de décryptage appropriés.

•**Cryptage des communications :** Les algorithmes de cryptage sont utilisés pour sécuriser les communications en ligne, tels que les connexions SSL/TLS utilisées pour les sites Web sécurisés, les réseaux privés virtuels (VPN) utilisés pour connecter des ordinateurs à distance, et les messageries sécurisées telles que Signal et WhatsApp. Les algorithmes de cryptage, tels que RSA et Diffie-Hellman, sont utilisés pour chiffrer les communications et garantir la confidentialité des informations transmises.

•**Authentification et autorisation :** Les algorithmes de cryptage sont utilisés pour garantir l'authenticité des identités et la sécurité des accès aux systèmes. Les systèmes d'authentification tels que les certificats numériques et les jetons d'authentification utilisent des algorithmes de cryptage pour garantir l'authenticité des identités. Les algorithmes de cryptage, tels que HMAC, sont également utilisés pour assurer l'intégrité des données et empêcher les attaques de type "man in the middle".

•**Transactions financières :** Les algorithmes de cryptage sont utilisés pour sécuriser les transactions financières en ligne, tels que les paiements par carte de crédit et les transferts de fonds. Les protocoles de cryptage, tels que SSL/TLS, sont utilisés pour chiffrer les informations de transaction et garantir la sécurité des paiements en ligne.

•**Cryptage des e-mails :** Les algorithmes de cryptage sont utilisés pour sécuriser les e-mails en transit et en repos. Les protocoles de cryptage, tels que PGP et S/MIME, sont utilisés pour chiffrer les e-mails et les rendre illisibles pour les tiers non autorisés.

•**Cryptage des mots de passe :** Les algorithmes de cryptage, tels que bcrypt et PBKDF2, sont utilisés pour crypter les mots de passe stockés dans les bases de données et les rendre illisibles pour les tiers non autorisés en cas de violation de sécurité.

•**Signature numérique :** Les algorithmes de cryptage, tels que RSA et DSA, sont utilisés pour créer des signatures numériques qui garantissent l'authenticité des documents et des messages. Les signatures numériques sont utilisées pour prouver l'authenticité des documents en ligne et garantir l'intégrité des données.

•**Cryptage de la voix et de la vidéo :** Les algorithmes de cryptage sont utilisés pour sécuriser les communications vocales et vidéo. Les protocoles de cryptage, tels que SRTP et ZRTP, sont utilisés pour chiffrer les communications vocales et vidéo et garantir

la confidentialité.

Conclusion

En conclusion de ce chapitre de notre nous avons mis en évidence l'exploration des fondements de la sécurité informatique et des principes essentiels de la cryptographie. Nous avons étudié en détail la cryptographie symétrique, la cryptographie asymétrique et le hachage, en examinant les mécanismes, les algorithmes et les applications de ces techniques dans différents domaines.

Nous avons conclu que la cryptographie symétrique offre une méthode efficace de chiffrement et de déchiffrement des données, mais nécessite une gestion sécurisée des clés. La cryptographie asymétrique, quant à elle, permet une communication sécurisée en utilisant une paire de clés distinctes, offrant des fonctionnalités telles que le chiffrement à clé publique et les signatures numériques. Cependant, elle est généralement plus lente et requiert une infrastructure de gestion des clés plus complexe.

Enfin la cryptographie joue un rôle crucial dans la sécurité informatique en protégeant les systèmes, les données et les communications contre les menaces et les attaques malveillantes. Toutefois, il est important de rester attentif aux avancées technologiques et aux évolutions constantes de la cryptographie pour garantir une sécurité adéquate. Les chapitres suivants approfondiront ces concepts et exploreront d'autres aspects de la cryptographie, afin de mieux appréhender les défis et les opportunités qu'elle présente dans le contexte de la sécurité informatique.

Dans le prochain chapitre, nous nous concentrerons sur les systèmes dynamiques, les fonctions de hachage et les différents algorithmes de hachage. Nous examinerons les systèmes dynamiques et leurs comportements chaotiques, en soulignant leur utilisation en cryptographie. Ensuite, nous étudierons les fonctions de hachage, leurs applications et les différents algorithmes couramment utilisés. Enfin, nous analyserons en détail la construction de Merkle-Damgard, une méthode populaire pour construire des fonctions de hachage sécurisées. Ce chapitre approfondira notre compréhension de ces concepts clés de la sécurité informatique, jetant ainsi les bases pour aborder les aspects avancés de la cryptographie et de la protection des données.

Chapitre 2

Les sytémes dynamiques et les fonctions de hachage

Introduction

Les systèmes dynamiques et non dynamiques, ainsi que les systèmes chaotiques, jouent un rôle crucial dans de nombreux domaines, allant des sciences naturelles à la modélisation mathématique et à la cryptographie. Dans le même temps, la fonction de hachage est devenue une primitive cryptographique essentielle pour garantir l'intégrité des données et la sécurité des systèmes [27]. Dans ce chapitre, nous explorerons les concepts fondamentaux des systèmes dynamiques et non dynamiques, en mettant l'accent sur les systèmes chaotiques, et nous étudierons leur relation avec la fonction de hachage.

Le premier volet de ce chapitre se concentrera sur les différents systèmes dynamiques et nous discuterons de leurs propriétés et de leurs comportements. Les systèmes dynamiques offrent une perspective intéressante sur l'évolution des systèmes au fil du temps.

Ensuite, nous aborderons le concept de systèmes chaotiques. Les systèmes chaotiques se distinguent par leur sensibilité aux conditions initiales et leur comportement imprévisible à long terme. Nous étudierons les caractéristiques des systèmes chaotiques, telles que la sensibilité aux conditions initiales, l'existence d'attracteurs étranges et les bifurcations. Nous explorerons également comment les systèmes chaotiques sont utilisés pour générer des nombres pseudo-aléatoires et pour sécuriser les fonctions de hachage. Par ailleurs, nous analyserons le rôle de la fonction de hachage dans la cryptographie, puis on passera à l'étude des principes fondamentaux des fonctions de hachage et la définition des différents algorithmes de hachage.

Enfin étudierons la construction de Merkle-Damgård, une méthode populaire de hachage en cryptographie. Nous examinons ses étapes de base, ses utilisations dans différents scénarios de sécurité, ses propriétés cryptographiques telles que la résistance aux collisions et la résistance à la préimage, ainsi que sa comparaison avec d'autres constructions de hachage comme la construction à éponge.

2.1 Les systémes dynamiques

2.1.1 Définition

Les systèmes dynamiques sont des modèles mathématiques utilisés pour étudier l'évolution des systèmes au fil du temps, qu'ils soient dans les domaines de la physique, de la biologie, de la chimie, de l'économie, de la météorologie, de la mécanique des fluides ou autres. Ils se caractérisent par leur capacité à changer et à évoluer, et sont décrits par des équations différentielles ou des différences qui décrivent comment les variables évoluent temporellement. Ces équations permettent de représenter graphiquement les solutions dans un espace de phase, mettant en évidence les interactions et les évolutions des variables. Les systèmes dynamiques peuvent être classés en linéaires ou non linéaires en fonction de la linéarité de leurs équations, et en chaotiques ou périodiques selon l'influence des conditions initiales ou des éléments aléatoires. Leur utilisation s'étend à la modélisation et à la simulation de phénomènes complexes, et trouve des applications dans de nombreux domaines scientifiques et techniques.

D'autre part, Un système dynamique est un modèle permettant de décrire l'évolution au cours du temps d'un ensemble des objets en interaction, il est défini par un triplet (X, T, f) constitue de l'espace d'état X, du domaine temporel T, et d'une application de transition d'état f : X × T →T, qui permet de définir à partir d'un vecteur de condition initiale, l'état du système à tout instant. Donc on peut définir un système dynamique comme une description d'un phénomène physique qui évolue au cours du temps (système continue), ou par rapport à une autre variable (système discret)[28].

Un système dynamique sur Rn est une application :

$$\mathrm{f} : R^{+} \times R^{n} \rightarrow R^{n}$$

2.1.2 Les systèmes dynamiques linéaires

Les systèmes dynamiques linéaires sont des systèmes dont les équations sont linéaires, ce qui signifie que les relations entre les variables sont décrites par des équations linéaires, comme des équations différentielles ou des équations de différence. Ces systèmes possèdent des propriétés mathématiques avantageuses qui les rendent plus faciles à analyser et à résoudre que les systèmes non linéaires. Par exemple, si un système dynamique linéaire est homogène, sa solution peut être obtenue en superposant les solutions correspondant à des conditions initiales particulières. De plus, la stabilité de ces systèmes peut être déterminée de manière précise et rigoureuse à partir de leurs équations. Les systèmes dynamiques linéaires sont largement utilisés pour modéliser des phénomènes dans des domaines tels que la physique, l'ingénierie et l'économie, ainsi que dans d'autres domaines. Ils jouent également un rôle crucial en contrôle automatique, où les équations linéaires sont fréquemment employées pour décrire le comportement de systèmes tels que les robots, les voitures autonomes, les avions et les satellites.

2.1.3 Les systèmes dynamiques non linéaires

Les systèmes dynamiques non linéaires se distinguent par leurs équations non linéaires qui décrivent les relations entre les variables du système. Contrairement aux systèmes linéaires, ces équations ne suivent pas des formes linéaires telles que les équations différentielles ou les équations de différence. Leur analyse et leur résolution sont souvent plus complexes, mais ils offrent une représentation plus réaliste de nombreux phénomènes naturels et sociaux. Les systèmes non linéaires se caractérisent par leur sensibilité aux conditions initiales, leur comportement chaotique, leurs bifurcations, leurs attracteurs étranges et leurs cycles limites. Leur prédiction et leur contrôle sont difficiles, mais ils engendrent des comportements complexes et fascinants. Leurs applications sont vastes, touchant des domaines tels que la physique, la biologie, la chimie, la mécanique des fluides, l'économie et la psychologie. De plus, ils sont utilisés pour modéliser et contrôler des systèmes complexes tels que les réseaux électriques, les circuits électroniques, les systèmes climatiques, les écosystèmes et les processus industriels.

2.1.4 Les systèmes dynamiques à temps discret

Les systèmes dynamiques à temps discret sont des modèles mathématiques couramment utilisés pour étudier des phénomènes qui évoluent par des sauts discrets à des moments spécifiques dans le temps. Ces systèmes sont décrits par des équations de différence et sont utilisés dans divers domaines tels que la production industrielle, les modèles de population, le contrôle des processus, les communications et les systèmes de commande numériques. Leur comportement peut être analysé à l'aide de techniques telles que l'analyse de stabilité, l'analyse de bifurcation, l'analyse de Lyapunov et la théorie du chaos. De plus, ils sont utilisés pour simuler et prédire le comportement futur des systèmes en utilisant des méthodes telles que la simulation de Monte Carlo et les algorithmes de prévision. En somme, les systèmes dynamiques à temps discret sont des outils essentiels pour comprendre et modéliser des phénomènes qui évoluent par des sauts discrets dans le temps[29].

2.1.4.1 Orbites

Les orbites sont des trajectoires périodiques qui reviennent régulièrement sur elles-mêmes. Plus précisément, une orbite d'ordre n est une trajectoire qui revient au même état après n itérations du système.

2.1.4.2 Points fixes

Les points fixes sont des états stationnaires du système qui ne changent pas au fil du temps. Plus précisément, un point fixe est un état où le système reste inchangé, quel que soit son état initial. Dans un système dynamique discret, un point fixe est caractérisé par un vecteur d'état qui reste inchangé après une itération du système. Autrement dit, si x est le vecteur d'état au temps t, et si f(x) est la fonction qui décrit l'évolution du système, alors un point fixe x* doit satisfaire l'équation f(x*) = x*.

2.1.4.3 Stabilité

La stabilité d'un système dynamique discret peut être analysée en étudiant les propriétés des points fixes du système. Plus précisément, un point fixe est considéré comme stable si toute perturbation du système à partir de cet état converge vers le point fixe au fil du temps. À l'inverse, un point fixe est instable si toute perturbation initiale entraîne une divergence du système par rapport au point fixe.

2.1.4.4 Systèmes dynamiques discrets et section de Poincaré

La technique dite des sections de Poincaré facilite l'étude des systèmes dynamiques considérés en ramenant l'analyse d'un système différentiel (temps continu) à celle d'une application (temps discret). Par le biais de cette méthode, la dimension d du problème initial sous forme de système différentiel est réduite d'une unité avec l'application en dimension d - 1.

2.1.5 Les systèmes dynamiques à temps continu

Les systèmes dynamiques à temps continu sont des modèles mathématiques largement utilisés pour étudier des phénomènes qui évoluent de manière continue dans le temps. Ces systèmes sont décrits par des équations différentielles qui décrivent comment les variables changent instantanément à chaque moment du temps. Ils sont couramment employés pour modéliser des phénomènes continus tels que les mouvements physiques, les systèmes électriques et électroniques, les processus chimiques et les systèmes biologiques. Ces systèmes peuvent être linéaires ou non linéaires et leur comportement est analysé à l'aide de méthodes telles que l'analyse de stabilité, l'analyse de bifurcation, l'analyse de Lyapunov et la théorie du chaos. De plus, ils permettent de simuler et de prédire le comportement futur d'un système en utilisant des techniques numériques de résolution d'équations différentielles. Les systèmes dynamiques à temps continu sont un outil mathématique essentiel pour la modélisation, la simulation et la compréhension des phénomènes continus dans le temps. Ils sont utilisés dans de nombreux domaines tels que la physique, l'ingénierie, la biologie, la chimie, l'économie, et bien d'autres[29].

2.1.5.1 Flot

Il s'agit de la trajectoire continue des points de l'espace des phases du système, qui décrivent l'évolution des variables du système dans le temps. Le flot est déterminé par les équations différentielles qui décrivent le système, et il peut être visualisé comme un champ de vecteurs qui indique la direction et la vitesse de l'évolution des variables du système à chaque point de l'espace des phases [30].

2.1.5.2 Orbites périodiques

Une orbite est la trajectoire suivie par un point dans l'espace des phases au cours du temps. Cette orbite peut être périodique, quasi périodique ou chaotique, selon les propriétés du système.

Une orbite périodique est une trajectoire fermée dans l'espace des phases, qui se répète régulièrement après une période de temps fixe. Cette période est appelée période orbitale. Les orbites périodiques sont des solutions stationnaires des équations du système, et elles peuvent être stables ou instables, en fonction des propriétés du système.

2.1.5.3 Stabilité

La stabilité est une notion importante dans l'étude des systèmes dynamiques, qui décrit la façon dont les solutions du système évoluent dans le temps. Un système est considéré comme stable si les solutions du système restent proches de leur état d'équilibre à long terme, même si elles sont initialement perturbées.

2.1.5.4 Bifurcation

La bifurcation se produit lorsque le système passe d'un état d'équilibre à un autre, ou lorsqu'il subit un changement qualitatif de son comportement dynamique. La bifurcation signifie un changement qualitatif de la dynamique du système, qui résulte du changement d'un des paramètres du système. Par exemple, déstabilisation d'un équilibre stable, apparition ou disparition d'un cycle ou d'un attracteur, ... La valeur pour laquelle la bifurcation se produit est nommée **le point de bifurcation** [31].

Bifurcation nœud-col
Cette bifurcation se produit lorsqu'un point d'équilibre, appelé nœud, devient instable et se sépare en deux points d'équilibre, appelé col. Cette bifurcation peut être observée dans des systèmes dynamiques qui ont une symétrie de réflexion ou une symétrie de translation. D'une autre manière, ce type de bifurcation donne naissance à deux cycles d'ordre k en même temps, l'un est attractif et l'autre est instable.

Bifurcation Neimark
Cette bifurcation aura lieu lorsque le paramètre de contrôle prend une valeur critique 0 pour laquelle la matrice jacobienne du système possède une paire de valeurs propres complexes conjuguées qui traversent l'axe imaginaire et le type de la stabilité de l'équilibre existant change avec l'apparition d'un cycle limite.

Bifurcation de Flip
Lors d'une bifurcation flip, un cycle-limite stable qui était auparavant atteint de manière régulière devient instable, et un nouveau cycle-limite stable apparaît. En d'autres termes, la période des oscillations double. Cette bifurcation peut être observée dans des systèmes dynamiques tels que les oscillateurs électriques, les systèmes de réaction-diffusion et les modèles de dynamique de population.

2.1.5.5 Les systèmes dynamiques chaotiques :

Les systèmes dynamiques chaotiques sont des systèmes non linéaires qui exhibent un comportement chaotique. Ce comportement chaotique se caractérise par une extrême sensibilité aux conditions initiales, où de petites variations initiales peuvent entraîner des différences significatives dans l'évolution future du système. Ces systèmes sont souvent associés à des attracteurs étranges, qui sont des ensembles de points dans l'espace des phases du système. Les attracteurs étranges possèdent des propriétés géométriques intéressantes, telles que la fractalité et la sensibilité aux conditions initiales. Les systèmes dynamiques chaotiques trouvent des applications dans de nombreux domaines, tels que la physique, la biologie, la chimie, la mécanique des fluides, l'économie et la psychologie. Ils sont également utilisés en cryptographie pour générer des séquences aléatoires et sont employés dans la modélisation et le contrôle de systèmes complexes tels que les réseaux électriques, les circuits électroniques, les systèmes climatiques, les écosystèmes et les processus industriels. L'analyse des systèmes dynamiques chaotiques fait appel à des méthodes telles que l'analyse de bifurcation, l'analyse de Lyapunov et l'analyse spectrale.

Les simulations numériques sont également utilisées pour étudier le comportement de ces systèmes. En somme, les systèmes dynamiques chaotiques constituent un domaine d'étude fondamental dans la théorie des systèmes dynamiques, offrant des propriétés mathématiques et des comportements fascinants qui ont des implications significatives dans de nombreux domaines scientifiques et techniques.

1. Propriétés des systèmes chaotiques

Bien qu'il n'y ait pas de définition mathématique du chaos universellement acceptée, une définition couramment utilisée stipule que pour qu'un système dynamique soit classifié en tant que chaotique, il doit comporter les propriétés suivantes : [29]

- **Non-linéarité**
- **Déterminisme**
- **Aspect aléatoire**
- **Sensibilité aux conditions initiales**

Non-linéarité

Un système chaotique est un système dynamique non linéaire. Un système linéaire ne peut pas être chaotique. Le comportement chaotique d'un système dynamique non linéaire est dû aux non linéarités. En général, pour prévoir des phénomènes générés par les systèmes dynamiques, la démarche consiste à construire un modèle mathématique qui établit une relation entre un ensemble de causes et un ensemble d'effets. Si cette relation est une opération de proportionnalité, le phénomène est linéaire. Dans le cas d'un phénomène non linéaire, l'effet n'est pas proportionnel à la cause.

Déterminisme

La notion de déterminisme signifie la capacité de prédire le futur d'un phénomène à partir d'un évènement passé ou présent. L'évolution irrégulière du comportement d'un système chaotique est dûe à la non- linéarité. Dans les phénomènes aléatoires, il est absolument impossible de prévoir la trajectoire d'une quelconque particule. À l'opposé, un système chaotique à des règles fondamentales déterministes et non probabilistes.

Aspect aléatoire

Les systèmes chaotiques se comportent, en effet d'une manière qui peut sembler aléatoire. Cet aspect aléatoire du chaos vient du fait que l'on est incapable de donner une description mathématique du mouvement, mais ce comportement est en fait décrit par des équations non linéaires parfaitement déterministes, comme par exemple les équations de Newton régissant l'évolution d'au moins trois corps en interaction.

Sensibilité aux conditions initiales

Certains phénomènes dynamiques non linéaires sont si sensibles aux conditions initiales que, même s'ils sont régis par des lois rigoureuses et parfaitement déterministes, les prédications exactes sont impossibles. Une autre propriété des phénomènes chaotiques est qu'ils sont très sensibles aux perturbations. L'un des premiers chercheurs à s'en être aperçu fut

Edward Lorenz qui s'intéressait à la météorologie et par conséquent aux mouvements turbulents d'un fluide comme l'atmosphère. Lorenz venait de découvrir que dans des systèmes non linéaires, d'infimes différences dans les conditions initiales engendraient à la longue des trajectoires totalement différentes. Il illustrait ce fait par l'effet papillon. Il est clair que la moindre erreur ou imprécision sur la condition initiale ne permet pas de décider à tout temps quelle sera la trajectoire effectivement suivie et en conséquence de faire une prédiction sur l'évolution à long terme du système, une des propriétés essentielles du chaos est donc bien cette sensibilité aux conditions initiales que l'on peut caractériser en mesurant des taux de divergence des trajectoires. En effet deux orbites chaotiques initiées avec des conditions initiales très voisines vont diverger et s'écarter l'une de l'autre très rapidement. La vitesse de divergence de deux orbites initialement voisines peut être étudiée à partir des exposants de Lyapunov afin de caractériser la nature du chaos observé [30].

Notion d'attracteur

Un attracteur est un objet géométrique vers lequel tendent toutes les trajectoires des points de l'espace des phases, c'est-à-dire une situation (ou un ensemble de situations) vers lesquelles évoluent un système, quelles que soient ses conditions initiales. Il en existe deux familles d'attracteurs : réguliers et étranges (chaotiques)[32].

•**Attracteurs réguliers** : Les attracteurs réguliers caractérisent l'évolution de systèmes non chaotiques, et peuvent être de trois sortes :

- Point fixe : est le plus simple et le plus courant d'attracteurs, dans lequel le système évolue vers un état de repos (point).

- Un cycle-limite : c'est une trajectoire fermée qui attire toutes les trajectoires proches.

- Un tore : représente les mouvements résultants de deux ou plusieurs oscillations indépendantes que l'on appelle parfois mouvements quasi périodiques[29].

•**Attracteurs étranges** Ils sont caractérisés par des propriétés telles que la sensibilité aux conditions initiales, l'existence de trajectoires périodiques et la présence d'une structure fractale. Un attracteur étrange peut être visualisé comme une trajectoire dans l'espace des phases d'un système dynamique.

Exposant de Lyapunov

L'évolution d'un flot chaotique est difficile à appréhender, parce que la divergence des trajectoires sur l'attracteur est rapide, C'est pourquoi on essaye d'estimer ou même de mesurer la vitesse de divergence ou convergence, Cette vitesse s'appelle l'exposant lyapunov. L'exposant de Lyapunov sert à mesurer le degré de stabilité d'un système et permet de quantifier la sensibilité aux conditions initiales d'un système chaotique. Le nombre d'exposants de Lyapunov est égal à la dimension de l'espace des phases et ils sont généralement indexés des plus grands au plus petit. L'apparition du chaos exige que les exposants de Lyapunov doivent remplir trois conditions :

- Au moins l'un d'eux est positif pour expliquer la divergence des trajectoires.

- Au moins l'un d'eux est négatif pour justifier le repliement des trajectoires.

- La somme de tous les exposants est négative pour expliquer qu'un système chaotique

est dissipatif, c'est-à-dire qu'il perd de l'énergie.

2. Transistion vers le chaos

Il existe plusieurs scénarios qui décrivent le passage vers le chaos. On constate dans tous les cas que l'évolution du point fixe vers le chaos n'est pas progressive, mais marquée par des changements discontinus qu'on a déjà appelés bifurcations. On peut citer trois scénarios de transition d'une dynamique régulière à une dynamique chaotique lors de la variation d'un paramètre : [33]

•**Doublement de période** Ce scénario a été observé dans les années 60 par R.May en dynamique des populations sur l'application logistique. A mesure que la contrainte augmente, la période d'un système forcé est multipliée par deux, puis par quatre, puis par huit, ... etc. ; ces doublements de période sont de plus en plus rapprochés ; lorsque la période est infinie, le système devient chaotique.

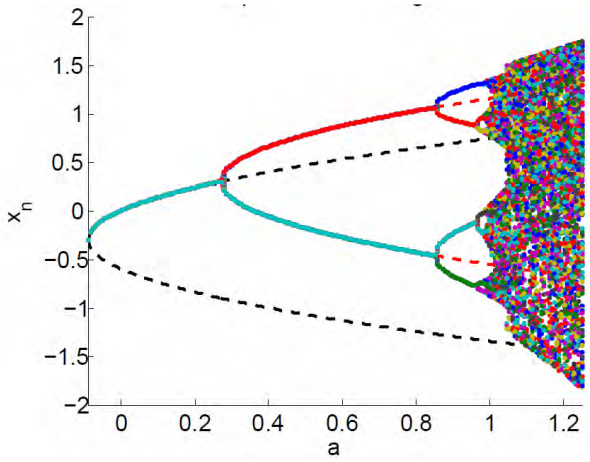

FIG. 2.1 : Cascade de doublements de période

•**Intermittence vers le chaos** Un mouvement périodique stable est entrecoupé par des bouffées chaotiques. Lorsqu'on augmente le paramètre de contrôle, les bouffées de turbulence deviennent de plus en plus fréquentes, et finalement le chaos apparaît.

•**Quasi-périodicité** Le scénario via la quasi-périodicité a été mis en évidence par les travaux théoriques de Ruelle et Takens 1971. Dans un système à comportement périodique a une seule fréquence, si nous changeons un paramètre alors il apparaît une deuxième fréquence. Si le rapport entre les deux fréquences est rationnel, le comportement est périodique. Mais, si le rapport est irrationnel, le comportement est quasi périodique. Alors, on change de nouveau paramètre et il apparaît une troisième fréquence, et ainsi de suite jusqu'au chaos.

2.1.5.6 Les différents systémes chaotiques :

Système chaotique à une dimension

Les systèmes chaotiques à une dimension sont des exemples plus simples de systèmes chaotiques qui peuvent être décrits par une seule équation, on cite comme exemple la mappe logistique.

•**Mappe logistique** La mappe logistique est un exemple classique de système chaotique à une dimension. Elle est définie par une équation récursive de la forme :

$$x \quad = r * x \ * (1 - x)$$

où x est la valeur à l'instant n, x est la valeur à l'instant n+1, et r est un paramètre qui détermine le comportement du système. Pour certaines valeurs de r, la mappe logistique peut présenter un comportement chaotique, avec des trajectoires sensibles aux conditions initiales et des bifurcations.

Système chaotique à deux dimensions
Les systèmes chaotiques à deux dimensions sont des exemples plus complexes de systèmes chaotiques qui nécessitent deux variables pour les décrire, on cite comme exemple le système de Hénon.

•**Système de Hénon** Le système de Hénon est un exemple bien connu du système chaotique à deux dimensions, qui a été proposé par Michel Hénon en 1976. Le système de Hénon présente un comportement chaotique pour certaines valeurs de ses paramètres, il peut générer des trajectoires complexes et des attracteurs chaotiques[32].

Système chaotique à trois dimensions
Les systèmes chaotiques à trois dimensions sont encore plus complexes que ceux à deux dimensions, car ils nécessitent trois variables pour les décrire, on cite comme exemple le modèle de Lorenz.

•**Modèle de Lorenz** : Edward Lorenz fut un météorologue qui, le premier, mit en évidence le caractère chaotique de la météorologie. Le couplage de l'atmosphère avec l'océan est décrit par le système d'équations aux dérivées partielles couplées de Navier-Stockes de la mécanique des fluides. En 1963, Lorenz eut l'idée de chercher un modèle d'équations pour étudier l'instabilité de Rayleigh-Bénard. Ce modèle a joué un rôle historique important puisque son évolution temporelle fait apparaître un comportement chaotique. De plus, il constitua le premier et le célèbre système différentiel dissipatif permettant d'observer un attracteur étrange pour certaines valeurs des paramètres[32].

L'attracteur de Lorenz est généré par le système d'équations suivant :

$$x = \ (y-x)$$
$$y = x(b-z)-y$$
$$z = xy-rz$$

Les paramètres , b et r sont des réels strictement positifs. La figure ci-dessous illustre l'attracteur de Lorenz en 3 dimensions x(t) ; y(t) et z(t) tel que = 10, b = 8=3 et r = 28.

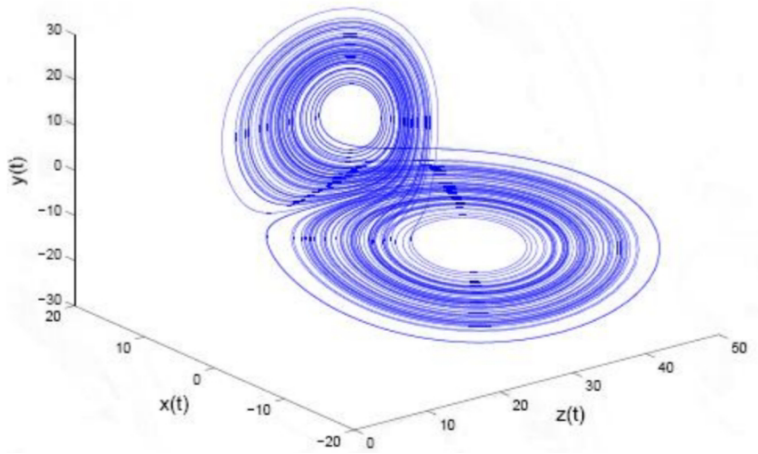

FIG. 2.2 : Modéle de Lorenz

2.2 Les fonctions de hachage

Une fonction de hachage est un algorithme utilisé en informatique pour prendre en entrée des données de tailles variables et produire une sortie de taille fixe, appelée "hache" ou "empreinte". L'objectif principal d'une fonction de hachage est de transformer les données d'entrée en un code unique, qui peut être utilisé pour identifier, comparer ou vérifier l'intégrité des données. Les fonctions de hachages sont largement utilisées dans de nombreux domaines de l'informatique, tels que la sécurité des données, la gestion des mots de passe, le stockage et la recherche de données, la vérification de l'intégrité des fichiers, la cryptographie, etc.

2.2.1 Propriétés cryptographiques des fonctions de hachage

Les fonctions de hachage doivent satisfaire certaines propriétés cryptographiques pour être considérées comme sécurisées. Voici les principales propriétés recherchées dans une fonction de hachage cryptographique :

•**Résistance à la collision :** Une fonction de hachage est résistante à la collision si elle rend extrêmement improbable la possibilité de trouver deux ensembles de données différentes qui produisent la même empreinte de hachage. Cela signifie qu'il est difficile de trouver des collisions, ce qui renforce l'intégrité des données.

•**Résistance à la seconde pré-image :** Une fonction de hachage est résistante à la seconde pré-image si, étant donné un ensemble de données d'origine, il est difficile de trouver un autre ensemble de données qui produit la même empreinte de hachage. En d'autres termes, il est difficile de trouver une deuxième entrée ayant la même empreinte que la première.

•**Résistance à la pré-image :** Une fonction de hachage est résistante à la pré-image si, étant donné une empreinte de hachage, il est difficile de trouver un ensemble de données qui produit cette empreinte spécifique. Cela signifie qu'il est difficile de retrouver les données d'origine à partir de l'empreinte de hachage.

•**Distribution uniforme :** Les empreintes de hachage doivent être distribuées de

manière uniforme, ce qui signifie que chaque empreinte de hachage possible a une probabilité égale d'être produite. Une distribution uniforme assure l'équité et l'efficacité de la fonction de hachage dans diverses applications.

•**Sensibilité aux changements :** Une petite modification apportée aux données d'entrée doit entraîner un changement significatif dans l'empreinte de hachage résultante. Cela garantit que des modifications mineures dans les données entraînent des empreintes de hachage complètement différentes, ce qui renforce l'intégrité des données.

•**Vitesse de calcul :** Les fonctions de hachage doivent être efficaces en matiére de vitesse de calcul pour pouvoir être utilisées dans des applications en temps réel. Une fonction de hachage performante peut générer rapidement des empreintes de hachage pour un grand volume de données.

2.2.2 Principes et mécanismes de la fonction de hachage

•**Compression :**La fonction de hachage utilise généralement une fonction de compression pour traiter les données d'entrée. Elle prend des blocs de données de taille fixe et génère une sortie de taille fixe, réduisant ainsi la taille des données.

•**Itération :** Elle traite souvent les données par blocs de taille fixe. Si les données d'entrée sont plus grandes que la taille du bloc, elles itèrent le processus de compression sur chaque bloc jusqu'à ce que tous les blocs aient été traités.

•**Remplissage :**Généralement conçue pour traiter des blocs de taille fixe. Lorsque les données d'entrée ne correspondent pas à un multiple de la taille du bloc, un processus de padding est utilisé pour ajuster les données à la taille requise en ajoutant des bits supplémentaires. Le padding permet de traiter les données de toutes les tailles de manière cohérente.

•**Fonctions de compression sécurisées :** Utilisées dans les fonctions de hachage, elles doivent offrir une sécurité renforcée, résistant aux collisions et à d'autres attaques cryptographiques courantes.

•**Propriétés de sécurité :** Les fonctions de hachage doivent satisfaire certaines propriétés de sécurité, comme la résistance aux collisions, la résistance à la prédiction et la distribution uniforme des empreintes de hachage. Elles garantissent l'intégrité des données et empêchent les attaquants de retrouver les données d'origine à partir de l'empreinte.

•**Utilisation dans les applications :** Les fonctions de hachage sont utilisées dans de nombreuses applications pour garantir l'intégrité des données, la confidentialité et l'authenticité. Elles sont utilisées dans les protocoles de sécurité, les signatures numériques, la vérification d'intégrité des fichiers, les systèmes de gestion d'identité, etc.

•**Réduction de la taille des données :** Elle transforme des données de tailles variables en une empreinte de hachage de taille fixe, indépendamment de la taille des données d'entrée. Cela permet de représenter de manière concise et compacte les données d'entrée.

•**Unicité de l'empreinte :** L'unicité de l'empreinte est une propriété essentielle recherchée dans une fonction de hachage. Chaque ensemble de données d'entrée devrait

idéalement produire une empreinte de hachage unique, rendant extrêmement improbable la présence de deux ensembles de données différents produisant la même empreinte. Cela garantit l'intégrité des données et évite les collisions.

2.2.3 Utilisations et applications de la fonction de hachage :

•**Vérification de l'intégrité des données :** Les fonctions de hachage sont utilisées pour vérifier si les données ont été altérées ou modifiées. En calculant le hachage d'un ensemble de données, on obtient une empreinte unique qui représente ces données. Si même un seul bit de ces données est modifié, le hachage généré sera complètement différent. Ainsi, en comparant le hachage d'origine avec le hachage recalculé, on peut détecter toute altération ou modification des données.

•**Stockage sécurisé des mots de passe :** Elles sont largement utilisées pour stocker de manière sécurisée les mots de passe dans les systèmes d'authentification. Plutôt que de stocker les mots de passe en texte brut, les applications et les services hachent les mots de passe et stockent uniquement les empreintes de hachage correspondantes. Lorsqu'un utilisateur tente de se connecter, le mot de passe fourni est haché et comparé à l'empreinte de hachage stockée. Cela permet de vérifier si le mot de passe est correct sans stocker le mot de passe réel lui-même, renforçant ainsi la sécurité en cas de violation de sécurité.

•**Authentification des données :**Elles sont utilisées pour l'authentification des données, notamment dans les protocoles de signature numérique. Dans ces protocoles, au lieu de signer directement les données, un hachage des données est signé à la place. En signant le hachage, on peut vérifier l'authenticité et l'intégrité des données sans avoir à transmettre les données complètes, ce qui peut être plus efficace en termes de performances et de sécurité.

•**Tables de hachage :**Elles sont utilisées dans la création de tables de hachage, qui sont des structures de données efficaces pour la recherche et l'indexation. Les tables de hachage permettent de stocker et de retrouver rapidement des informations en utilisant des clés qui sont hachées pour calculer les positions de stockage. Cela accélère les opérations de recherche, de récupération et de mise à jour des données, ce qui est utile dans les bases de données, les systèmes de fichiers et les caches.

•**Réduction de la complexité :** Elles permettent de réduire la complexité des problèmes en convertissant des données volumineuses en empreintes plus petites et plus gérables. Par exemple, lors de la recherche de doublons ou de la comparaison de grands ensembles de données, on peut d'abord calculer les hachages des données, puis comparer les empreintes hachées pour identifier les similitudes, ce qui peut être plus rapide et moins gourmand en ressources que de comparer directement les données complètes.

•**Cryptographie asymétrique :** Elles sont utilisées dans les algorithmes de signature numérique et de chiffrement asymétrique. Par exemple, dans le chiffrement hybride, les données sont d'abord hachées pour créer un condensé, qui est ensuite chiffré avec la clé privée du destinataire pour l'authentification et l'intégrité des données. De plus, elles sont également utilisées dans la construction de certificats numériques, où le hachage des données est signé avec une clé privée pour prouver l'authenticité des informations du certificat.

2.2.4 Les algorithmes de hachage

Les algorithmes de hachage sont des fonctions cryptographiques qui prennent une entrée, appelée message, et produisent une valeur de hachage, également appelée empreinte ou condensée. Voici quelques-uns des algorithmes de hachages les plus couramment utilisés :

2.2.4.1 MD5 (Message Digest 5)

Cet algorithme a été développé en 1992. Il produit une empreinte de 128 bits [34]. MD5 est rapide et largement utilisé pour vérifier l'intégrité des données et générer des empreintes uniques pour les fichiers. Cependant, il présente des vulnérabilités connues, notamment des collisions (deux entrées différentes produisant la même empreinte), ce qui le rend inapproprié pour les applications nécessitant une sécurité élevée.

2.2.4.2 SHA-1 (Secure Hash Algorithm 1)

Cet algorithme a été développé en 1995 et produit une empreinte de 160 bits. Il a été largement utilisé, mais des vulnérabilités ont été découvertes, notamment des collisions pratiques, ce qui signifie que deux entrées différentes peuvent produire la même empreinte[35]. Par conséquent, il est recommandé de ne plus utiliser SHA-1 pour les nouvelles applications.

2.2.4.3 SHA-256 (Secure Hash Algorithm 256 bits)

Il fait partie de la famille des algorithmes SHA-2, développée par la National Security Agency (NSA) des États-Unis[36]. SHA-256 produit une empreinte de 256 bits. Il est largement utilisé dans les applications de sécurité, tels que les certificats SSL/TLS, les systèmes de gestion d'identité et la vérification d'intégrité des données.

2.2.4.4 SHA-3 (Secure Hash Algorithm 3)

SHA-3 est le résultat d'un concours international lancé par le NIST (National Institute of Standards and Technology) en 2007. L'algorithme gagnant, Keccak, a été choisi pour devenir SHA-3[36]. Il offre une sécurité accrue et des performances similaires ou meilleures que les algorithmes SHA-2. SHA-3 est disponible en plusieurs versions, notamment SHA-3-224, SHA-3-256, SHA-3-384 et SHA-3-512.

Ces algorithmes de hachage sont principalement utilisés pour vérifier l'intégrité des données, créer des identifiants uniques pour les fichiers, stocker des mots de passe de manière sécurisée, et pour d'autres applications liées à la sécurité et à la confidentialité

des données. Il convient de noter que l'utilisation appropriée d'un algorithme de hachage dépend du contexte et des exigences de sécurité spécifiques de chaque application.

2.3 La construction de Merkle-Damgård

2.3.1 Définition

La construction de Merkle-Damgård est une méthode couramment utilisée en cryptographie pour construire des fonctions de hachage sécurisées. Elle a été proposée indépendamment par Ralph Merkle et Ivan Damgård dans les années 1980. Elle est utilisée dans des algorithmes de hachage populaires tels que SHA-1 et SHA-2. Ralph Merkle est un informaticien et cryptographe américain connu pour ses travaux dans les domaines de la cryptographie et des nanotechnologies. Ivan Damgård est un cryptographe danois reconnu pour ses contributions à la cryptographie à clé publique et aux fonctions de hachage. Ensemble, ils ont proposé la construction de Merkle-Damgård, qui repose sur l'utilisation de fonctions de compression pour construire des fonctions de hachage sécurisées. Cette construction a jeté les bases de nombreux algorithmes de hachage utilisés aujourd'hui. Les contributions de Merkle et Damgård ont eu un impact significatif sur la sécurité des communications et des données numériques, et leur travail continue d'influencer la recherche en cryptographie moderne[37]. La construction de Merkle-Damgård est basée sur l'idée de diviser un message en blocs de taille fixe et de les traiter séquentiellement à l'aide d'une fonction de compression. La fonction de compression prend en entrée le bloc de données actuel et l'état intermédiaire précédent, et produit un nouvel état intermédiaire. Ce nouvel état est ensuite utilisé comme l'entrée de la fonction de compression pour le bloc de données suivant. Ce processus est répété jusqu'à ce que tous les blocs de données soient traités, et le dernier état intermédiaire est utilisé pour produire la valeur de hachage finale.

2.3.2 Les étapes principales de la construction de Merkle-Damgård :

•**Padding (remplissage) :** le message d'entrée est divisé en blocs de taille fixe, puis un bit de valeur 1 est ajouté à la fin du dernier bloc de message pour marquer la fin du message ensuite des bits de remplissage sont ajoutés au dernier bloc de message pour le compléter jusqu'à atteindre la taille fixe, enfin l'étape d'ajout de bits de longueur du message intervient et qui consiste à ajouter des bits représentant la longueur du message d'origine après le remplissage pour indiquer la taille du message.

•**Division en blocs :** Le message d'entrée est divisé en blocs de taille fixe, généralement égale à la taille de bloc de la fonction de compression utilisée. Si le message d'entrée n'est pas un multiple de la taille du bloc, un padding est généralement ajouté pour le rendre conforme.

•**Initialiser l'état intermédiaire :** Un état intermédiaire est initialisé avec une valeur spécifique. Cette valeur est généralement définie comme un vecteur d'initialisation fixe.

•**Compression :** Une fonction de compression est appliquée à chaque bloc de données. La fonction de compression prend en entrée le bloc de données et le résultat de la compression précédente (ou une valeur d'initialisation spéciale pour le premier bloc) et produit une sortie de taille fixe. La sortie de la compression devient l'entrée pour la compression du bloc suivant. Cette fonction prend en entrée le bloc de données actuel et l'état intermédiaire précédent, et produit un nouvel état intermédiaire.

•**Mise à jour de l'état intermédiaire :** Le nouvel état intermédiaire produit par la fonction de compression est utilisé comme l'entrée de la fonction de compression pour le bloc de données suivant. Cela permet de conserver l'information des blocs de données précédentes dans l'état intermédiaire.

•**Répétition :**Les étapes 3 et 4 sont répétées pour chaque bloc de données du message. Cela permet de traiter séquentiellement tous les blocs et de mettre à jour l'état intermédiaire à chaque étape.

•**Valeur de hachage finale :** Une fois que tous les blocs de données ont été traités, le dernier état intermédiaire obtenu est utilisé pour produire la valeur de hachage finale. Cette valeur de hachage est souvent représentée par un ensemble de bits de taille fixe.

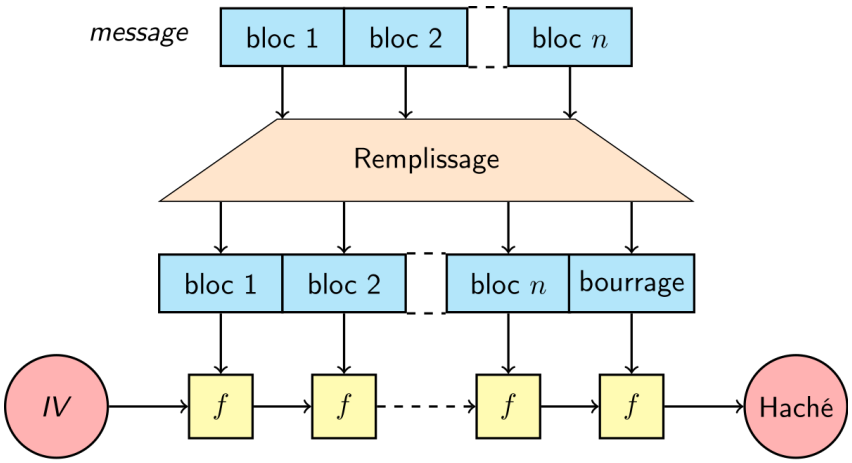

Fig. 2.3 : Principe de la construction Merkle-Damgard

2.3.3 Propriétés cryptographiques de Merkle-Damgård :

La construction de Merkle-Damgård offre des propriétés de résistance aux collisions et de résistance à la prédiction, mais il est important de noter que certaines précautions doivent être prises pour garantir leur efficacité. Voici une analyse de ses propriétés :

•**Résistance aux collisions :**

- La résistance aux collisions fait référence à la capacité de la fonction de hachage à éviter la création de deux entrées différentes qui produisent la même sortie (la même valeur de hachage).

- Dans la construction de Merkle-Damgård, les fonctions de compression utilisées doivent être conçues de manière à minimiser les collisions potentielles.

- La résistance aux collisions dépend de la qualité de la fonction de compression utilisée.

Si la fonction de compression est faible et vulnérable aux attaques cryptographiques, cela peut compromettre la résistance aux collisions de la construction.

•Résistance à la prédiction :

- La résistance à la prédiction fait référence à la difficulté de prédire la sortie (la valeur de hachage) d'un bloc de données donné sans connaître l'état intermédiaire précédent.

- La résistance à la prédiction est renforcée par l'utilisation d'une fonction de compression non inversible, qui rend difficile la reconstruction de l'état intermédiaire à partir de la valeur de hachage.

Cependant, il est important de noter que la sécurité de la construction de Merkle-Damgård dépend également de la résistance des fonctions de compression utilisées. Si une vulnérabilité est découverte dans la fonction de compression, cela peut potentiellement affecter la sécurité de la construction globale. Par conséquent, il est essentiel d'utiliser des fonctions de compression cryptographiquement sécurisées et éprouvées lors de la mise en œuvre de la construction de Merkle-Damgård.

2.3.4 Les fonctions de hachage basée sur la construction de Merkle-Damgård :

•MD5 (Message Digest 5) :

- MD5 est une fonction de hachage cryptographique basée sur la construction de Merkle-Damgård.

- Il prend en entrée un message de longueur variable et produit une empreinte de hachage de 128 bits.

- MD5 a été largement utilisé dans le passé pour diverses applications, notamment la vérification de l'intégrité des fichiers et le stockage de mots de passe.

- Cependant, MD5 présente des vulnérabilités connues, notamment des attaques de collision, qui ont réduit sa sécurité et ont conduit à son remplacement par des fonctions de hachage plus sûres.

•SHA-1 (Secure Hash Algorithm 1) :

- SHA-1 est également une fonction de hachage basée sur la construction de Merkle-Damgård.

- Il produit une empreinte de hachage de 160 bits à partir d'un message d'entrée de longueur variable.

- SHA-1 a été largement utilisé dans de nombreuses applications, y compris la sécurité des communications, les certificats numériques et les systèmes de gestion de clés.

- Cependant, des vulnérabilités ont été découvertes dans SHA-1, en particulier en ce qui concerne les attaques de collision, ce qui a conduit à son abandon progressif dans les applications critiques. Il est recommandé d'utiliser des fonctions de hachage plus sécurisées, telles que SHA-256 ou SHA-3.

2.3.5 Avantages et inconvénients de Merkle-Damgård par rapport aux constructions plus récentes

Dans le tableau ci-dessous nous allons explorer les avantages et les inconvénients de Merkle-Damgård par rapport aux constructions plus récentes de fonctions de hachage.

Inconvénients	Avantages
- Vulnérabilité aux attaques de longueur : La construction de Merkle-Damgård peut être vulnérable à des attaques de longueur, où un adversaire peut modifier le message sans modifier sa longueur, tout en obtenant le même hachage.	- Modularité : La construction de Merkle-Damgård permet de construire des fonctions de hachage en utilisant une fonction de compression simple et répétable pour traiter les blocs de message. Cela facilite la conception et l'analyse des fonctions de hachage.
- Prise en compte de la longueur du message : La construction de Merkle-Damgård nécessite une prise en compte explicite de la longueur du message à l'aide de l'étape de padding. Cela peut introduire des vulnérabilités potentielles si la gestion de la longueur est incorrecte.	- Résistance aux collisions : Si la fonction de compression utilisée dans la construction de Merkle-Damgård est considérée comme résistante aux collisions, la fonction de hachage résultante bénéficiera également d'une résistance aux collisions.
- Dépendance à la fonction de compression : La résistance aux collisions et aux autres attaques de la construction de Merkle-Damgård dépend fortement de la sécurité de la fonction de compression utilisée. Si la fonction de compression est faible, cela peut affaiblir la sécurité de la fonction de hachage globale.	- Extensibilité : La construction de Merkle-Damgård permet d'étendre la longueur de hachage en utilisant des fonctions de compression plus longues, tout en maintenant la taille des blocs de message fixe.
- Préimage faible : La construction de Merkle-Damgård peut souffrir d'une faiblesse de préimage, où il peut être relativement plus facile de trouver un message qui produit un hachage donné que de trouver une collision.	- Performances : La construction de Merkle-Damgård est généralement efficace en termes de performances, car elle permet de traiter les blocs de message de manière indépendante et parallèle.

Tab. 2.1 : Avantages et inconvénients de la construction de Merkle-Damgård

2.3.6 Comparaison de la construction de Merkle-Damgård et de la construction à éponge

La comparaison entre la construction de Merkle-Damgård et la construction à éponge met en évidence des différences significatives en termes de conception, de sécurité et de performances des fonctions de hachage, et on va mettre en évidence tous ces points dans le tableau ci-dessous.

Construction de Merkle-Damgård	**Construction à éponge**
- La construction de Merkle-Damgård est une méthode couramment utilisée pour construire des fonctions de hachage cryptographiques. - Elle divise le message d'entrée en blocs de taille fixe et utilise une fonction de compression pour traiter chaque bloc de manière itérative. - La fonction de compression prend en entrée le bloc de message actuel ainsi qu'un état intermédiaire, généralement appelé vecteur d'état, qui est mis à jour à chaque itération. - Le vecteur d'état est initialisé avec une valeur fixe avant de commencer le traitement des blocs de message. - Après le traitement de tous les blocs de message, le vecteur d'état final est renvoyé en tant que hachage du message d'entrée. - La sécurité de la fonction de hachage dépend de la sécurité de la fonction de compression utilisée, qui doit résister à des attaques telles que les collisions et les préimages.	- La construction à éponge est une approche différente pour la conception de fonctions de hachage, basée sur une permutation répétée d'un état interne. - L'état interne est initialement rempli avec une valeur fixe, appelée capacité, qui est suivie d'une zone de stockage, appelée éponge, de taille variable. - La construction à éponge utilise deux principales opérations : l'absorption et le squeeze. - L'absorption permet d'incorporer les données du message d'entrée dans l'état interne en combinant les bits de données avec l'état actuel à l'aide d'une fonction de permutation. - Lorsque les données d'entrée sont plus grandes que la capacité de l'éponge, plusieurs itérations d'absorption sont effectuées. - Le squeeze est utilisé pour extraire les bits de hachage du résultat final en les récupérant de l'état interne à chaque itération de la fonction de permutation. - La taille de sortie du hachage n'est pas limitée à une taille fixe, ce qui permet d'obtenir des hachages de différentes longueurs à partir de la même construction. - La construction à éponge offre une résistance naturelle aux attaques de longueur, car elle peut absorber et traiter des messages de longueurs variables.

Tab. 2.2 : Comparaison de la construction de Merkle-Damgård et de la construction à éponge

Conclusion

Dans ce chapitre, nous avons examiné les systèmes dynamiques et les fonctions de hachage en cryptographie. Nous avons exploré les différents types de systèmes dynamiques, tels que les systèmes linéaires et non linéaires, à temps discret et à temps continu. Ensuite, nous nous sommes penchés sur les fonctions de hachage, en étudiant leurs propriétés cryptographiques, leurs principes et mécanismes, ainsi que leurs utilisations et applications. Par la suite, nous nous sommes concentrés sur la construction de Merkle-Damgård, une méthode couramment utilisée pour construire des fonctions de hachage cryptographiques. Nous avons détaillé les étapes de cette construction, exploré ses domaines d'utilisation et mis en évidence ses propriétés cryptographiques. De plus, nous avons examiné les fonctions de hachage basées sur la construction de Merkle-Damgård et avons évalué leurs avantages et inconvénients par rapport aux constructions plus récentes. Enfin, nous avons réalisé une comparaison entre la construction de Merkle-Damgård et la construction à éponge, offrant ainsi une perspective complète sur ces deux approches de construction de fonctions de hachage. Ce chapitre nous a permis d'acquérir une compréhension approfondie des systèmes dynamiques et des fonctions de hachage, ainsi que des différentes constructions utilisées en cryptographie, jetant ainsi les bases pour la suite de notre étude. Dans la continuité de notre étude approfondie des systèmes dynamiques et des fonctions de hachage en cryptographie, nous aborderons dans le prochain chapitre une analyse approfondie du comportement chaotique de la carte logistique et son application pertinente dans la conception de notre fonction de hachage.

Chapitre 3

Analyse et Conception

Introduction

Dans ce troisième chapitre, nous nous concentrons sur l'analyse approfondie du comportement chaotique de la carte logistique et son application dans la conception d'une fonction de hachage efficace. Nous explorons les caractéristiques chaotiques de la carte logistique en utilisant l'exposant de Lyapunov pour identifier les zones chaotiques de cette fonction mathématique. Cette analyse approfondie nous aide à comprendre les effets des variations initiales sur les résultats obtenus. Sur la base de cette compréhension, nous concevons une fonction de hachage en utilisant la carte logistique comme fondement. La carte logistique est choisie pour sa sensibilité aux conditions initiales, sa simplicité d'implémentation et son faible besoin en ressources de calcul et de mémoire par rapport à d'autres cartes chaotiques. Notre approche divise les données en blocs de taille fixe, sur lesquels nous appliquons itérativement la carte logistique avec les paramètres appropriés, générant ainsi des séquences pseudo-aléatoires pour chaque bloc. Les empreintes individuelles des blocs sont ensuite combinées à l'aide d'une opération de fusion spécifique pour obtenir une empreinte finale sécurisée. L'objectif principal de cette fonction de hachage est d'assurer l'intégrité et la sécurité des données en exploitant les caractéristiques chaotiques de la carte logistique.

3.1 Étude de la carte logistique

3.1.1 Définition de la carte logistique

La carte logistique est une fonction mathématique non linéaire largement utilisée dans la cryptographie pour générer des nombres aléatoires et pour chiffrer les données. Elle a été introduite pour la première fois en 1975 par l'économiste américain Robert May comme un modèle pour décrire la dynamique des populations[38].

$$\mathbf{x_{n+1}} = \mathbf{r} \cdot \mathbf{x_n} \cdot (\mathbf{1} - \mathbf{x_n})$$

La carte logistique peut être visualisée en traçant les valeurs de x en fonction du r. Pour certaines valeurs de r, la fonction converge rapidement vers un point fixe, tandis que pour d'autres valeurs, elle présente une dynamique chaotique, oscillant entre plusieurs valeurs. Cette propriété de la carte logistique la rend particulièrement utile pour la cryptographie, car elle génère des séquences de nombres pseudo-aléatoires. Pour étudier le comportement de la carte logistique on utilise l'exposant de lyapunov.

3.1.2 L'exposant de lyapunov

L'exposant de Lyapunov est une mesure utilisée pour quantifier la sensibilité aux conditions initiales d'un système dynamique, tel que la carte logistique. Il permet d'évaluer la rapidité avec laquelle deux trajectoires initialement proches divergent au fil du temps.

L'équation de Lyapunov pour la carte logistique est donnée par :

$$\lambda = \frac{1}{N} \sum_{n}^{N} \ln \left| \int'(Xn) \right|$$

Cette formule calcule le nombre de Lyapunov en prenant la moyenne logarithmique des valeurs absolues des intégrales des trajectoires du système dynamique à chaque itération n. Cela peut être interprété comme une mesure de la sensibilité aux conditions initiales et de la divergence des trajectoires.

Il est important de noter que l'exposant de Lyapunov est un outil essentiel pour étudier le chaos et la sensibilité aux conditions initiales dans le cadre de la carte logistique et d'autres systèmes dynamiques non linéaires.

La quantification de la sensibilité aux conditions initiales se fait à l'aide du nombre de Lyapunov (λ). Le nombre de Lyapunov est une mesure de la vitesse de divergence exponentielle des trajectoires du système. Il est calculé en prenant la limite lorsque n tend vers l'infini de la moyenne logarithmique des valeurs absolues des dérivées de la trajectoire par rapport aux conditions initiales.

Un (λ) positif indique une divergence exponentielle des trajectoires, ce qui est caractéristique du comportement chaotique. Plus le (λ) est élevé, plus le système est chaotique. En revanche, un (λ) négatif ou nul indique une convergence ou une stabilité du système.

Il est important de noter que l'exposant de Lyapunov est un outil essentiel pour étudier le chaos et la sensibilité aux conditions initiales dans le cadre de la carte logistique et d'autres systèmes dynamiques non linéaires.

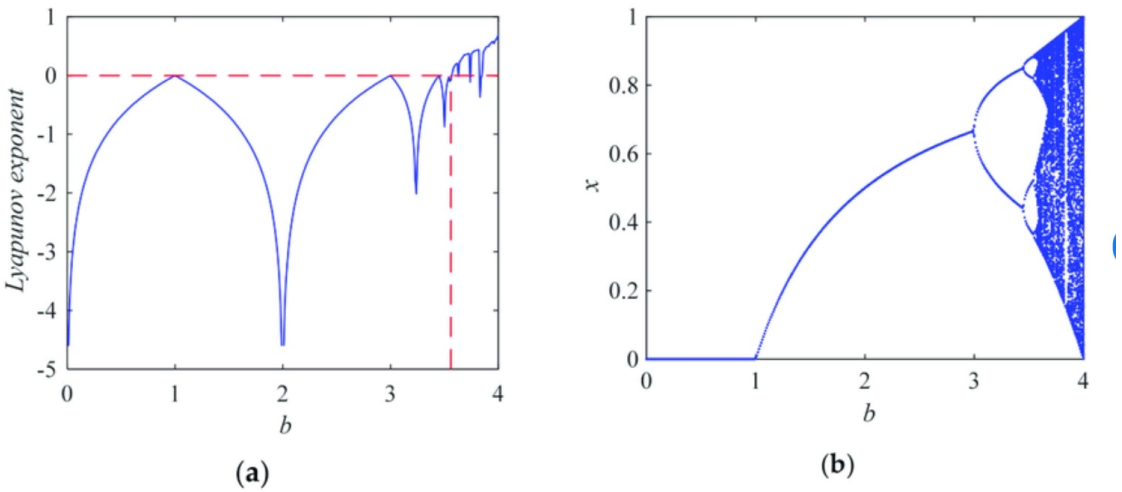

FIG. 3.1 : Figure graphique de l'éxposant de Lyapunov

3.1.3 Étude du comportement de la carte logistique

L'étude du comportement de la carte logistique, en utilisant l'exposant de Lyapunov, nous permet d'approfondir notre compréhension des transitions entre les comportements stables, périodiques et chaotiques du système. L'exposant de Lyapunov est utilisé pour quantifier la sensibilité aux conditions initiales de la carte logistique. Il mesure la vitesse de divergence exponentielle des trajectoires du système et indique si de petites variations

initiales dans la valeur de la variable conduisent à des trajectoires distinctes. Un exposant de Lyapunov positif témoigne d'une sensibilité chaotique aux conditions initiales. Lorsque nous traçons le graphe de la carte logistique en fonction du r et calculons l'exposant de Lyapunov pour chaque valeur de r, nous pouvons observer des zones où l'exposant de Lyapunov devient positif, indiquant un comportement chaotique. Ces zones chaotiques se caractérisent par une sensibilité accrue aux conditions initiales, où de petites variations initiales entraînent des trajectoires divergentes et des motifs complexes sur le graphe. En étudiant l'évolution de l'exposant de Lyapunov en fonction du r, nous pouvons identifier les valeurs de r où les transitions vers le chaos se produisent. Ces transitions se manifestent par des bifurcations dans le comportement du système, où de nouvelles trajectoires chaotiques émergent. Ainsi, l'utilisation de l'exposant de Lyapunov dans l'étude de la carte logistique nous permet d'explorer les différentes régions du système, de comprendre les bifurcations et les transitions vers le chaos, et de mettre en évidence la sensibilité aux conditions initiales. Cette approche nous permet d'appréhender la complexité et la richesse de la dynamique non linéaire de la carte logistique.

En résumé, l'étude du comportement de la carte logistique en utilisant l'exposant de Lyapunov nous offre une vision plus détaillée de la carte logistique pour analyser les transitions entre les comportements stables, périodiques et chaotiques, et pour explorer la sensibilité aux conditions initiales. Cela nous permet de mieux comprendre la dynamique non linéaire et la complexité inhérente à ce système.

d'après notre étude, on peut déduire que :

- De [**0 , 3.57**] : C'est la plage de valeurs où l'exposant de Lyapunov est négatif, est généralement appelée la "région de stabilité" ou la "région régulière".

- De [**3.57 , 3.99**] : Cette plage de valeurs correspond à un exposant de Lyapunov positif, indiquant une forte sensibilité aux conditions initiales, où de petites variations entraînent des différences significatives et un comportement chaotique du système dynamique, est généralement appelée la "région chaotique".

3.2 Les problemes existant sur les autres fonctions de hachage

Plusieurs algorithmes de hachage présentent des problèmes de collisions, qui sont la possibilité de trouver deux entrées différentes produisant la même empreinte. MD5, SHA0 et SHA1 sont vulnérables aux collisions, bien que SHA1 soit plus complexe à exploiter. Ces vulnérabilités compromettent leur sécurité dans certaines applications. Par ailleurs, SHA2, bien qu'étant considérablement plus résistant, présente une résistance théorique réduite aux attaques de collisions. SHA3, quant à lui, offre une meilleure résistance, mais ses performances de calcul sont généralement plus lentes. Les problèmes de collisions sont donc une préoccupation commune pour MD5, SHA0, SHA1 et SHA2, tandis que SHA3 présente des problèmes de performance et d'adoption limitée.

3.2.1 Problématique

Les problèmes de collisions et de vulnérabilités des algorithmes de hachage soulèvent des préoccupations majeures en matière de sécurité et d'intégrité des données. Ces vulnérabilités peuvent être exploitées pour la falsification de données, la compromission de la confidentialité ou encore la création de faux certificats. Par conséquent, il est essentiel d'utiliser des fonctions de hachage robustes et sécurisées pour garantir l'authenticité et l'intégrité des informations. C'est dans ce contexte que je présente maintenant ma fonction de hachage, qui offre des solutions aux problèmes mentionnés et constitue une mesure de sécurité fiable pour protéger les données sensibles.

3.2.2 Objectif de notre fonction de hachage

Les fonctions de hachage sont connues pour leurs sensibilitées du fait que changer un seul bit du message original conduit à une sortie totalement différente, d'un autre côté il y a les cartes chaotiques qui sont connues pour leur sensibilité envers ses conditions initiales La valeur de notre condition initiale même avec des changements d'ordre 10^{-10} , son comportement sera totalement différent. Cette qualité de sensibilité est partagée entre les fonctions de hachage et les cartes chaotiques ainsi que la légèreté de la carte logistique nous a conduit à élaborer une fonction de hachage basée sur ces deux concepts en utilisant la construction Merkel Damgard.

3.2.3 Principe de fonctionnement

Notre fonction de hachage est du type MAC qui reçoit un message de traitement en entrée et une clé sécrète de 128 bits et fournis un code unique de 128 bits. Le message est injecté à une fonction de padding et la clé est traitée pour extraire la condition initiale et le nombre de rotations, ces données acquises (les sorties) sont transmit au reste de la fonction.

Ensuite en utilisant la condition initiale (CI) obtenue à partir de l'étape de prétraitement, l'étape logistique génère une séquence de clés et un nombre de rotations.

Enfin la phase de compression reçoit la sortie de la fonction du padding, la séquence de clés générées par l'étape logistique et le nombre de rotation généré par l'étape de prétraitement. Cette phase joue un rôle essentiel dans la compression du message en effectuant plusieurs opérations.

Voici une figure qui illustre le fonctionnement général de notre fonction de hachage :

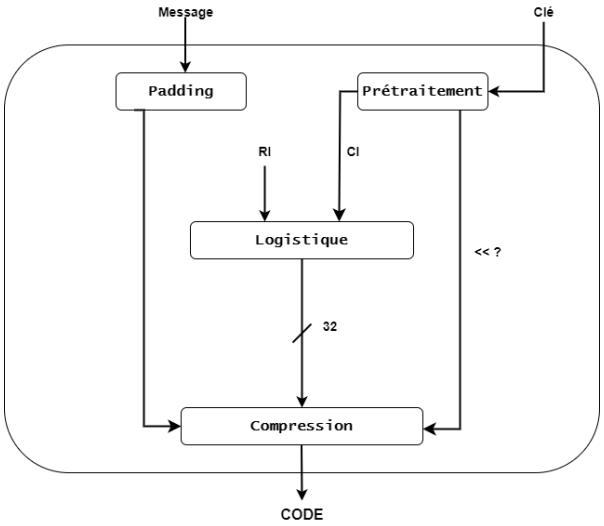

FIG. 3.2 : La construction de la fonction de hachage proposée

•**Étape de padding :** Son objectif est de diviser le message d'entrée en blocs de 16 octets (128 bits). Cette étape est essentielle pour assurer une uniformité de traitement des messages de différentes tailles.

-Tout d'abord, le message est converti en une liste d'octets représentant les valeurs ASCII du message. Par exemple, le message "Hello" serait converti en [72, 101, 108, 108, 111].

-Le padding consiste à ajouter un bit à 1 et une suite de 0 jusqu'à atteindre la moitié de la taille du dernier bloc de 16 octets et la taille du message original sera insérée dans cette partie, et si on a 8 octets vides dès le début on n'a pas à créer un nouveau bloc vide.

-Chaque bloc contient 128 bits et pour le dernier bloc on doit avoir à la fin 8 octets vide pour mentionner la taille du message.

-Après le padding, le message est divisé en blocs de 128 bits, représentés par une liste de blocs.

Voici ci-dessous une figure qui illustre le fonctionnement de l'étape du Padding :

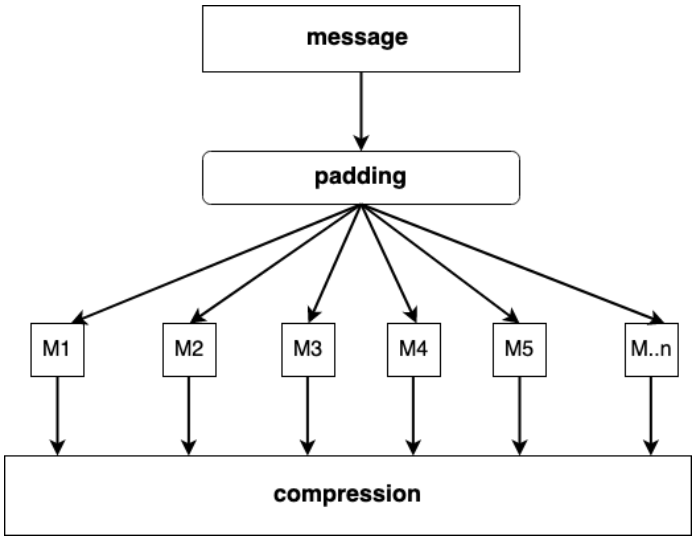

FIG. 3.3 : Etape du padding

•**Étape de prétraitement :** Elle reçoit la clé en entrée et effectue des opérations de calcul pour extraire la condition initiale (CI) et le nombre de rotations requis de la manière suivante :

-La clé est représentée par une liste d'octets, tout comme le message. Par exemple, si la clé est "SecretKey", elle serait convertie ASCII en [83, 101, 99, 114, 101, 116, 75, 101, 121].

-La clé est ensuite divisée en deux parties égales, key1 et key2, chacune ayant une taille égale à la moitié de la taille de la clé d'origine.

-La condition initiale (CI) est calculée en effectuant l'opération :

$$\mathbf{CI} = \mathbf{key1} *^{(2,5)} / \mathbf{key2} *^{(2,15)}$$

-Un nombre de rotations referentiel est calculé en prenant la somme des ASCII de toute la clé modulo 6 pour avoir le nombre de rotations strictement inférieur à 6.

•**Étape logistique :** Elle utilise la condition initiale (CI) obtenue à partir de l'étape de prétraitement et génère une séquence de clés. Dans cette étape, nous utilisons une fonction de mappage logistique qui prend une valeur (RI) de 3,9999999 et la CI pour générer les clés. Ces clés sont générées selon le nombre de blocs en utilisant une boucle itérative en respectant les itérations de la suppression de l'effet transitoire.

-Les clés est générées en utilisant la formule récursive suivante :

$$\mathbf{x_{n+1}} = \mathbf{r} \cdot \mathbf{x_n} \cdot (\mathbf{1} - \mathbf{x_n})$$

•**Étape de compression :** L'étape de compression c'est une étape importante et c'est celle qui va nous permettre la compression du message, cette étape effectue plusieurs opérations :

La fonction compression est définie pour chiffrer chaque bloc du message. Elle prend en entrée un bloc de message et les clés logistiques sous forme d'un vecteur.

-Pour chaque bloc généré par le padding il sera mis dans le processus de compression qui consiste à :

-Permutation du bloc en utilisant la fonction permute-vector qui ordonne les cellules selon les valeures generées par la carte logistique.

-Division du bloc en deux parties égales, V1 et V2.

-Application de la fonction de rotation rot-G sur V1 noté V1' (le nombre de rotations est fourni dans l'étape de prétraitement).

-Application du XOR avec une des clés logistiques générée par la fonction logistique sur V2 noté V2'.

-Application du XOR entre V1' et V2' pour obtenir V1''.

-Assemblage de V1'' et V2' dans V en les inversant.

-Application du XOR entre V et l'état précedent.

-Affichage de l'empreinte digitale.

Ce processus est répété pour chaque bloc de 16 octets, ce qui donne le bloc de message chiffré.

-Finalement, tous les blocs de message chiffrés sont concaténés pour former le message chiffré final.

Voici ci-dessous une figure illustrant le diagramme schématique de notre fonction de hachage en détail :

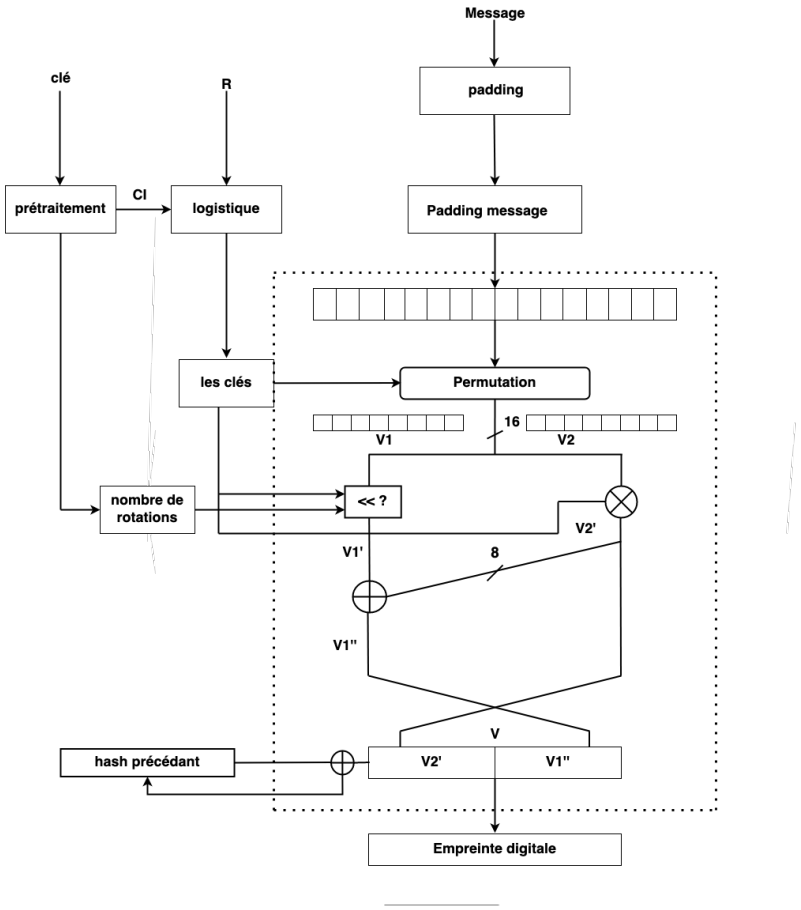

FIG. 3.4 : Diagramme schématique de la fonction de hachage proposée

En conclusion, toutes les étapes mentionnées ci-dessus sont indispensables pour le chiffrement du message. Elles assurent le traitement uniforme des messages de différentes tailles, l'extraction de la condition initiale et du nombre de rotation nécessaires, ainsi que la compression et la génération des clés de chiffrement. Ensemble, ces étapes garantissent la sécurité et la confidentialité du message chiffré final.

Conclusion

Dans ce troisième chapitre, nous nous sommes concentrés sur l'analyse approfondie du comportement chaotique de la carte logistique et son application dans la conception d'une fonction de hachage efficace. Nous avons exploré les caractéristiques chaotiques de la carte logistique en utilisant l'exposant de Lyapunov pour identifier les zones chaotiques de cette fonction mathématique. Cette analyse approfondie nous aide à comprendre les effets des variations initiales sur les résultats obtenus. Sur la base de cette compréhension, nous concevons une fonction de hachage en utilisant la carte logistique comme fondement. La carte logistique est choisie pour sa sensibilité aux conditions initiales, sa simplicité d'implémentation et son faible besoin en ressources de calcul et de mémoire par rapport à d'autres cartes chaotiques. Notre approche divise les données en blocs de taille fixe, sur lesquels nous appliquons itérativement la carte logistique avec les paramètres appropriés,

générant ainsi des séquences pseudo-aléatoires pour chaque bloc. Les empreintes individuelles des blocs sont ensuite combinées à l'aide d'une opération de fusion spécifique pour obtenir une empreinte finale sécurisée. L'objectif principal de cette fonction de hachage est d'assurer l'intégrité et la sécurité des données en exploitant les caractéristiques chaotiques de la carte logistique.

Dans le prochain chapitre, nous évaluerons notre fonction de hachage en la soumettant à des tests approfondis pour mesurer sa sécurité, sa résistance aux attaques et son efficacité. Ces tests nous permettront de déterminer si notre approche basée sur la carte logistique peut produire une empreinte finale sécurisée et fiable pour différents types de données.

Chapitre 4

Réalisation et test

Introduction

Ce chapitre décrit la partie réalisation et test de notre fonction de hachage. Premièrement, on présente le langage et les outils utilisés pour la réalisation. Dans la deuxième partie, on applique plusieurs tests sur notre fonction de hachage. Ces tests nous permettent d'évaluer notre schéma à voir, la confusion et la diffusion, la résistance aux collisions, et la distribution des hashs de sortie. Ensuite, on analyse les performances en fonction des résultas obtenues.

4.1 Réalisation

4.1.1 Environnement de développement

4.1.1.1 Langage de programmation C

Le choix du langage C pour l'implémentation de la fonction de hachage a été motivé par plusieurs facteurs. Le langage C est connu pour sa performance et son efficacité dans les calculs complexes, ce qui est essentiel pour assurer des performances optimales de l'algorithme de hachage. De plus, il permet un contrôle précis sur la mémoire, ce qui est nécessaire pour manipuler efficacement les blocs de données impliqués dans le processus de hachage. Étant largement utilisé dans la cryptographie, le langage C offre également un large éventail de ressources et de bibliothèques qui facilitent le développement et la validation de l'algorithme de hachage. En résumé, le choix du langage C offre une combinaison de performances, de contrôle de la mémoire et de disponibilité des ressources nécessaires pour créer une fonction de hachage fiable basée sur les cartes chaotiques.

4.1.1.2 IDE Visual Studio Code

Visual Studio Code (VS Code) est un environnement de développement intégré (IDE) polyvalent et léger. Il a été choisi pour ce projet en raison de sa popularité et de sa flexibilité, lui permettant de prendre en charge le langage C utilisé dans l'implémentation de la fonction de hachage. Les fonctionnalités de VS Code, telles que la coloration syntaxique, l'autocomplétion du code et la gestion de la source control, ont été utilisées pour faciliter le développement. L'IDE offre également une interface utilisateur intuitive et une intégration fluide avec les outils de compilation C. Dans l'ensemble, l'utilisation de Visual Studio Code a permis de créer un environnement de développement efficace et convivial pour la réalisation du projet.

4.1.1.3 Configuration de l'environnement de développement

La configuration de l'environnement de développement impliquait l'installation d'un compilateur C compatible et des ajustements de la configuration de Visual Studio Code. Des extensions pertinentes ont également été installées pour améliorer les fonctionnalités de l'IDE. L'objectif était de créer un environnement fluide et efficace, en garantissant une

compilation précise, des fonctionnalités d'édition avancées et une intégration harmonieuse avec les outils externes. Cette configuration a permis d'optimiser la productivité et de minimiser les problèmes de développement, contribuant ainsi à la réalisation du projet de manière efficace et sans heurts. La machine utilisé pour la réalisation de ce projet est un **MacBook Pro 13 2020** un CPU Intel Core i5 et 8 Go de RAM.

4.1.1.4 Bibliothèques et dépendances

Plusieurs bibliothèques externes ont été utilisées dans le projet pour faciliter le développement de la fonction de hachage basée sur les cartes chaotiques. La bibliothèque mathématique standard du langage C a été utilisée pour effectuer des opérations mathématiques complexes nécessaires dans les calculs chaotiques. Des bibliothèques de génération de nombres aléatoires ont également été utilisées pour obtenir des valeurs chaotiques. L'utilisation de ces bibliothèques a simplifié les calculs mathématiques, amélioré la qualité des valeurs chaotiques et accéléré le développement en utilisant des fonctionnalités prêtes à l'emploi. Cela a permis de se concentrer sur la conception et l'implémentation de l'algorithme de hachage, garantissant des fondements solides et une fiabilité du code.

4.2 Test

Une fonction de hachage sécurisé doit répondre à certains critères de sécurité parmi eux on a la confusion et la diffusion, la résistance aux collisions et la distribution des valeurs de hachages.

4.2.1 La distribution des valeurs de hachage

Pour tester une fonction de hachage on commence par visualiser ces sorties. Une bonne fonction de hachage dispose d'une distribution des valeurs sorties assez complète, autrement dit, les sorties ne se positionnent pas dans une seule zone de valeurs. Pour analyser cette qualité sur notre fonction de hachage on suit les étapes suivantes :

-Un message de 16 points d'exclamation est créé et la clé secrète est fixée à une suite de zéros.

-Un hash de 128 bits est calculé.

La figure 4.1 ci-dessous illustre le message d'entrée et la figure 4.2 illustre la sortie de la fonction sur 128 bits.

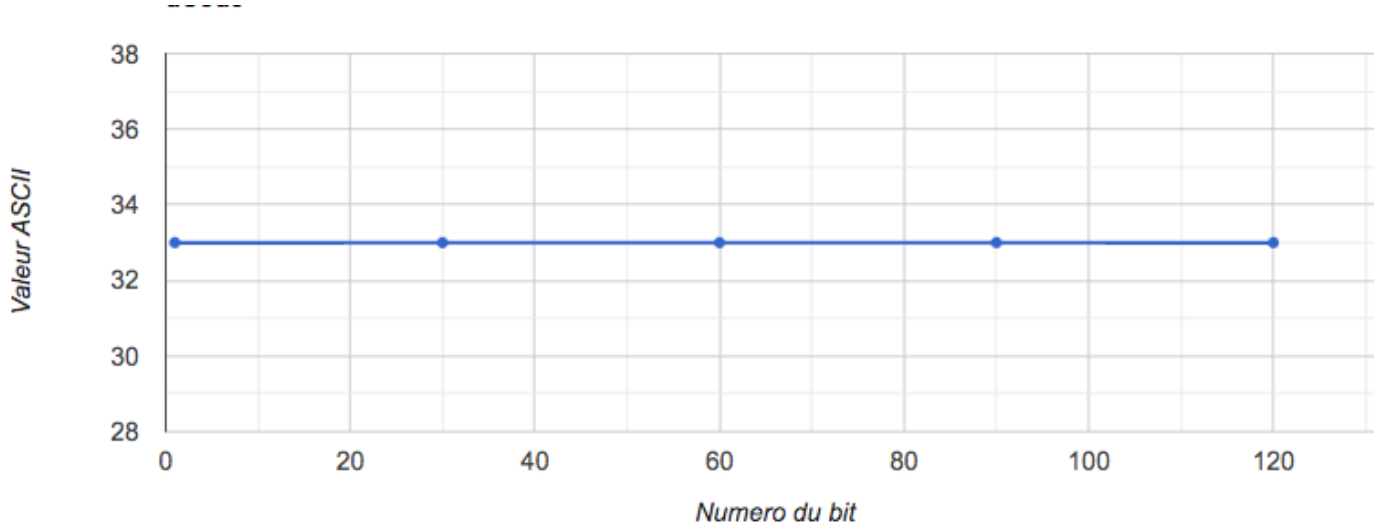

FIG. 4.1 : Message entrée de la fonction

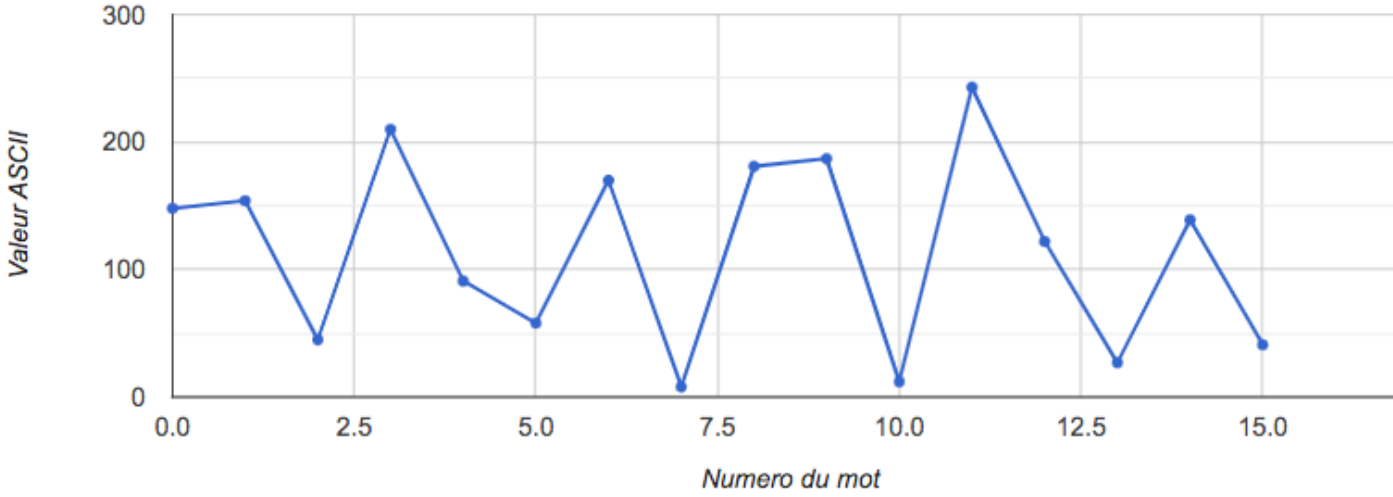

FIG. 4.2 : Sortie de la focntion sur 128 bits

D'aprés les résultats obtenus, on déduit que notre focntion de hachage dispose d'une bonne distribution.

4.2.2 La confusion et la diffusion

C'est une propriété importante qui calcule la capacité de masquer la clé secrète le message d'entrée sur les sorties. Pour évaluer notre fonction de hachage on effectue le test suivant :

-On choisit un message aléatoire et on calcule son code de hachage puis un bit du message est choisi aléatoirement et inversé et l'hash est recalculé.

-Les deux valeurs de hachage sont comparées en utilisant la distance de Hamming qu'on note B_i les valeurs statistiques suivantes sont calculées pour N = 10000.

Moyenne du nombre de changement de bits :

$$\overline{\mathbf{B}} = \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{B_i} \tag{4.1}$$

Moyenne de probabilité de changement de bits :

$$\mathbf{P} = \frac{\overline{\mathbf{B}}}{\mathbf{H}_{\text{len}}} \mathbf{100}\% \tag{4.2}$$

Minimum et maximum des changements de bits :

$$\mathbf{B}_{\min} = \min(\mathbf{B_i}) \mathbf{et} \mathbf{B}_{\max} = \max(\mathbf{B_i}), \mathbf{i = 1, 2, 3....N} \tag{4.3}$$

Variance standard du changement de bits :

$$\overline{\mathbf{B}} = \sqrt{\frac{\mathbf{1}}{\mathbf{N-1}} \sum_{\mathbf{i=1}}^{\mathbf{N}} (\mathbf{B_i} - \overline{\mathbf{B}})^{\mathbf{2}}} \tag{4.4}$$

Variance standard de la probabilité :

$$\mathbf{P} = \sqrt{\frac{\mathbf{1}}{\mathbf{N-1}} \sum_{\mathbf{i=1}}^{\mathbf{N}} \left(\frac{\mathbf{B_i}}{\mathbf{H}_{\text{len}}} - \mathbf{P} \right)^{\mathbf{2}}} \mathbf{100}\% \tag{4.5}$$

Les résultats de ces tests sont illustrés dans le tableau ci-dessous :

	$\overline{B}$	$P(\%)$	$B_{\min}$	$B_{\max}$	ΔB	$\Delta P(\%)$
Notre fonction	62.32	48.97	39	72	5.74	4.61

On déduit que notre fonction dispose d'une bonne confusion et d'une bonne diffusion qui sont proches des valeurs idéales qui sont 64.00 et 50

4.2.3 La résistance aux collisions

L'une des attaques les plus menées sur les fonctions de hachages est la recherche de collisions. Cette attaque a pour but de rechercher deux messages d'entrée différents qui produisent la même sortie. Cette vulnérabilité nous permet la falsification d'une opération d'authentification et permet les accès et les connexions non autorisées à des systèmes critiques. Pour analyser notre fonction de hachage en matière de collision on effectue le test suivant :

-On génère un message aléatoirement et on calcule son hasch, puis on choisit un bit aléatoirement et l'inverser le hasch et recalculer. -Sur les deux haches on calcule le nombre de caractère égal qui partage les mêmes positions.

Les résultas obtenus sont illustrés sur le tableau ci-dessous :

Hits	0	1	2	3	4	5	6
Fonction proposée	9314	642	39	5	0	0	0

Selon les résultats obtenus pour N=10000 on constate que notre fonction de hachage propose une bonne résistance aux collisions. Ce qui prouve la supériorité de notre fonction.

Conclusion

Le quatrième chapitre de ce mémoire a été consacré à la réalisation de la fonction de hachage basée sur les cartes chaotiques. Pour ce faire, un environnement de développement approprié a été configuré en utilisant le langage C et l'IDE Visual Studio Code. Les tests effectués ont évalué la distribution des valeurs de hachage, la confusion et la diffusion, ainsi que la résistance aux collisions. Les résultats ont confirmé l'efficacité de la fonction de hachage, avec une distribution des valeurs, une forte confusion et diffusion, ainsi qu'une résistance adéquate aux collisions. Ces résultats positifs valident la fonction de hachage basée sur les cartes chaotiques et renforcent sa fiabilité et sa sécurité. Ces conclusions fournissent une base solide pour la suite du mémoire.

Perspectives

Les perspectives pour ce projet incluent l'optimisation de la fonction de hachage, le renforcement de la sécurité, son intégration dans des applications spécifiques, ainsi que l'étude de l'impact des paramètres chaotiques et la comparaison avec d'autres fonctions de hachage existantes. Ces perspectives permettront d'améliorer les performances, la sécurité et d'explorer de nouvelles applications pour la fonction de hachage basée sur les cartes chaotiques.

Conclusion générale

Conclusion générale

Dans le cadre de ce projet de fin d'études, nous avons exploré les domaines de la cryptographie, des systèmes dynamiques et des fonctions de hachage. Nous avons étudié les principaux services de la cryptographie, les types de cryptographie utilisés et les domaines d'utilisation du chiffrement. De plus, nous avons examiné les systèmes dynamiques, en nous concentrant sur les systèmes dynamiques linéaires et non linéaires, à temps discret et à temps continu, ainsi que sur les fonctions de hachage et leurs propriétés cryptographiques.

Nous avons approfondi notre compréhension de la construction de Merkle-Damgård, une construction couramment utilisée dans les fonctions de hachage. Nous avons analysé ses étapes principales, ses domaines d'utilisation et ses propriétés cryptographiques. Nous avons également comparé les avantages et les inconvénients de cette construction par rapport aux constructions plus récentes, comme la construction à éponge.

Dans notre étude, nous avons également examiné la carte logistique et son rôle dans les problèmes existants liés aux fonctions de hachage. Nous avons étudié le comportement de la carte logistique, en mettant en évidence l'exposant de Lyapunov, et identifié les problématiques dans d'autres fonctions de hachage. À partir de ces observations, nous avons défini les objectifs de notre fonction de hachage proposée et présenté son principe de fonctionnement et sa construction.

Le quatrième chapitre de notre projet a été consacré aux tests et à l'analyse de notre fonction de hachage. Nous avons effectué des tests rigoureux pour évaluer les performances, la résistance aux attaques et l'efficacité de notre fonction de hachage. Nous avons analysé les résultats des tests et les avons comparés à des normes et des critères de qualité établis dans le domaine de la cryptographie.

Dans l'ensemble, ce projet de fin d'études nous a permis d'acquérir une compréhension approfondie de la cryptographie, des systèmes dynamiques et des fonctions de hachage. Nous avons exploré divers concepts, étudié leurs applications et leurs limites, et proposé une nouvelle fonction de hachage dans le but d'améliorer la sécurité des informations confidentielles.

Il convient de noter que la sécurité des systèmes cryptographiques est un domaine en constante évolution, et notre fonction de hachage doit être continuellement évaluée et améliorée pour résister aux attaques potentielles. Ce projet nous a permis de poser les bases d'un travail futur visant à renforcer la sécurité et à explorer de nouvelles perspectives dans le domaine de la cryptographie.

En conclusion, nous espérons que ce projet apportera une contribution significative à la sécurité des informations confidentielles et servira de point de départ pour des recherches ultérieures dans le domaine de la cryptographie. La cryptographie et les fonctions de hachage jouent un rôle crucial dans notre société numérique, et il est essentiel de poursuivre les efforts pour développer des systèmes sécurisés et résilients face aux attaques croissantes.

Bibliographie

[1] Prey Project. Types of encryption : Symmetric or asymmetric, rsa or aes.

[2] Philippe Guillot. *La cryptologie, L'art des codes secrets*. EDP Sciences, 2013.

[3] Marius Mea. Cryptographie : synthèse. En ligne.

[4] Christophe Bidan. Cryptographie et cryptanalyse. Cours.

[5] Gilles Dubertret. *Initiation à la cryptographie, Cours et exercices corrigés*. Vuibert, 4e édition edition, 2012.

[6] Damien Vergnaud. *Exercices et problèmes de cryptographie, Licence 3, master, écoles d'ingénieurs*. Dunod, 2015.

[7] El Khier Dehmeche. Étude et comparaison des principaux systèmes de cryptage, 2006.

[8] G. Zennor. *Cours de cryptographie*. 2000.

[9] Cryptographie nouvel algorithme de chiffrement evolutionnaire basé occurrences (aceo).

[10] Douglas Stinson. *Cryptographie, théorie et pratique*. Vuibert, 2003.

[11] D. Barsky and G. Dartois. *Cryptographie Paris 13*.

[12] Canteaut02a.pdf. https://www.inria.fr/.

[13] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In *Advances in Cryptology – CRYPTO '94*, pages 227–233. Springer US, Boston, MA, 1994.

[14] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In *Advances in Cryptology – CRYPTO '94*, pages 227–233. Springer US, Boston, MA, 1994.

[15] Fadia TALEB. *cours16*.

[16] Frederic Bayart. La saga du des.

[17] Daniel Genkin, Adi Shamir, and Eran Tromer. Rsa key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference*, volume 8616 of *Lecture Notes in Computer Science*, pages 444–461, Santa Barbara, CA, USA, 2014. Springer.

[18] Auteurs. A comprehensive evaluation of cryptographic algorithms : Des, 3des, aes, rsa and blowfish.

[19] H. Delfs and H. Knebl. *Introduction to Cryptography : Principles and Applications*. Springer-Verlag, Berlin Heidelberg New York, 2007.

[20] FILALI Mohamed Amine. Etude et implémentation pipeline sur fpga de l'algorithme de chiffrement aes. Master's thesis, Université de Mohamed Boudiaf, Oran, 2015.

[21] F. Arnault. Théorie des nombres & cryptographie. Cours D.E.A, Université de Limoges, mai 2003.

[22] El Khier DEHMECHE. Etude et comparaison des principaux systèmes de cryptage. Master's thesis, Université M'sila, 2006.

[23] Mémoire de fin d'étude présenté en vue d'obtention du diplôme de master en systèmes informatiques distribués. Master's thesis, Université Mouloud Mammeri de Tizi Ouzou.

[24] Documentation python - module sys. http://docs.python.org/3.3/library/sys.html.

[25] Simplilearn. Digital signature algorithm tutorial. https://www.simplilearn.com/tutorials/cryptography-tutorial/digital-signature-algorithm.

[26] J.F. Raymond and A. Stiglic. Security issues in the diffie-hellman key agreement protocol. 2000.

[27] Assia Senouci. https://thesis.univ-biskra.dz/4938/1/SenouciAssia.pdf.

[28] A Gasri. *Chaos et synchronisation (généralisé) dans les systèmes dynamiques*. PhD thesis, Université Frères Mentouri-Constantine-1, 5 2018.

[29] Systémes dynamiques non linéaires. http://dspace.centre-univ-mila.dz/jspui/bitstream/123456789/1837/1/Systeme%20dynamique%20non%20lineaire.pdf.

[30] S Barka and A Khenfri. Bifurcations et chaos pour un modèle d'oscillation. Mémoire de master, Centre Universitaire Abd Elhafid Boussouf Mila, 2017/2018.

[31] W Laouira. *Contrôle des systèmes dynamiques chaotiques*. Thèse de doctorat, Université Constantine 1, 2017/2018.

[32] M Tidjani. Synchronisation des systèmes dynamiques chaotiques à dérivées fractionnaires. Magistère en mathématiques, Université Mentouri Constantine, 2014.

[33] R Djenhi and A Makouf. Introduction à la théorie des systèmes dynamiques discrets. Mémoire de master, Centre Universitaire Abd Elhafid Boussouf Mila, 2016/2017.

[34] Md5. https://www.techtarget.com/searchsecurity/definition/MD5.

[35] Sha-1. https://www.lifewire.com/what-is-sha-1-2626011.

[36] Analog Devices. Back to basics : Secure hash algorithms. https://www.analog.com/en/technical-articles/back-to-basics-secure-hash-algorithms.html.

[37] Educative. What is merkle-damgard construction ? https://www.educative.io/answers/what-is-merkle-damgard-construction.

[38] Educative. Robert may. https://www.pnas.org/doi/10.1073/pnas.2016616117.