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

N° d'ordre : 3592

THESE

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

Structure de Recherche *: Intelligent processing and security of systems (IPSS)*
Discipline *: Informatique*
Spécialité *: Sécurité Informatique*

Présentée et soutenue le 04-02-2022 par :

Khalid BEKKAOUI

Développement et optimisation d'un système cryptographique basé sur les principes de décomposition et de NP-complétude de la théorie des graphes

Fouzia OMARY	*PES*	*Faculté des Sciences de Rabat, Université Mohammed V.*	*Présidente*
Zine-El-Abidine GUENNOUN	*PES*	*Faculté des Sciences de Rabat, Université Mohammed V.*	*Rapporteur/Examinateur*
Chouaib MOUJAHDI	*PH*	*Institut Scientifique de Rabat, Université Mohamed V.*	*Rapporteur/Examinateur*
Younes CHIHAB	*PH*	*Ecole Supérieure de Technologie de Kénitra, Université Ibn Tofail.*	*Rapporteur/Examinateur*
Nassim KHARMOUM	*PA*	*Centre National pour la Recherche Scientifique et Technique (CNRST).*	*Invité*
Soumia ZITI	*PES*	*Faculté des Sciences de Rabat, Université Mohammed V.*	*Directrice de thèse*

Année Universitaire : 2020/2021

✉ Faculté des Sciences, avenue Ibn Battouta, BP. 1014 RP, Rabat –Maroc
☎ 00212(0) 37 77 18 76 🖷 00212(0) 37 77 42 61 ; http://www.fsr.ac.ma

À ma mère et mon père bien aimés
Je dédie ce travail de thèse aux 2 personnes
qui me sont le plus chers au monde
pour avoir fait de moi ce que je suis.

À mes chers frères : Hamza et Toufik
À my second half : Keltoum
À mon cousin : Oussama
À mes collègues de Linedata Maroc

À toute ma famille

Remerciements

Les travaux présentés dans ce mémoire ont été effectués au sein de l'équipe Intelligent Processing and Security of Systems (IPSS) de la Faculté des Sciences de Rabat (FSR)-Université Mohammed V au Maroc sous la direction de **Mme. Soumia ZITI**, Professeur d'Enseignement Supérieur à la Faculté des Sciences de Rabat.

En premier lieu, je tiens à remercier ma directrice de thèse **Mme. Soumia ZITI**, Professeur d'Enseignement Supérieur à la Faculté des Sciences de Rabat avec qui j'ai travaillé depuis mon stage de Master. Dans une ambiance toujours décontractée, mais néanmoins studieuse, elle a bien guidé mes premiers pas dans le monde de la Recherche. Et je réitère mes remerciements pour sa grande disponibilité, son aide précieuse, ainsi que pour ses efforts qu'elle a prodigués pour l'accomplissement de ce travail de thèse.

Mes sincères remerciements vont également à **Mme. Fouzia OMARY**, qu'elle me soit permis de saluer sa bienveillance à mon égard. C'est avec honneur que j'ai pu bénéficier de ses connaissances et de ses remarques et critiques aiguisées. Je la remercie, particulièrement, pour les nombreuses discussions et échanges qui ont grandement contribué à la réussite de cette thèse.

Je tiens à remercier **Mme. Fouzia OMARY**, Professeur d'Enseignement Supérieur à la Faculté des Sciences de Rabat, d'avoir accepté de présider le jury de ma thèse.

Je tiens aussi à remercier **M. Zine-El-Abidine GUENNOUN**, Professeur d'Enseignement Supérieur à la Faculté des Sciences de Rabat, d'avoir accepté de juger la qualité de mon travail en tant que rapporteur/examinateur.

Mers remerciements vont également à **M. Chouaib MOUJAHDI**, Professeur Habilité à l'Institut Scientifique, Université Mohammed V, d'avoir accepté de rapporter et d'examiner ce mémoire de thèse.

Je tiens aussi à remercier **M. Younes CHIHAB**, Professeur Habilité à la Faculté des Sciences de Rabat, d'avoir accepté de juger la qualité de mon travail en tant que rapporteur/examinateur.

Je remercie aussi **M. Nassim KHARMOUM**, Professeur assistant au CNRST, d'avoir accepté d'évaluer ce travail de recherche.

Je remercie mes amis qui m'ont apporté leur soutien et leurs encouragements tout au long de cette aventure. Dans le désordre, un merci spécial à Ayoub LAHMIDI, Marouane NAZIH et Mehdi ZOUITINI pour être de vrais amis, merci pour vos beaux cœurs et votre soutien continu, merci pour les discussions inspirantes et tous les moments que nous avons passés ensemble.

Je garde le meilleur pour la fin, ma famille qui a supporté toutes les difficultés morales et matérielles pour me soutenir au terme de mes études. J'adresse ma profonde gratitude et mon immense reconnaissance à mes raisons d'être, ma mère et mon père, qui m'ont éduqué et orienté. Merci de m'avoir encouragé et soutenu dans mes choix. Nul mot et nulles expressions refléteront le grand amour et la profonde gratitude que je porte pour vous. À mes chers frères et mon cousin Oussama pour leurs soutiens et attentions. Ils étaient pour moi, une vraie source d'inspiration et ont été toujours à mes côtés. Que Dieu le Tout Puissant vous garde et vous procure santé et bonheur.

Table des matières

Liste des abréviations

GPAs *Générateurs de nombres Pseudo-aléatoires*

GPA *Générateur de nombres Pseudo-aléatoire*

MAC *Code d'authentication de Message*

MACS *Codes d'authentication de Messages*

boite-S *boîte de substitution*

PSOCA *Design of New Pseudo-Random Number Generator Based on Non-Uniform Cellular Automata*

BC-GT *A new block cipher system using graph theory*

EBC-GT *An enhanced block cipher system using graph theory*

Liste des figures

Liste des tableaux

Liste des algorithmes

Résumé

Dans le cadre du travail réalisé, nous nous sommes intéressés à la conception et au développement des nouvelles approches de sécurité informatique en s'appuyant sur les propriétés de la théorie des graphes. Dans ce contexte, différentes techniques développées dans la littérature souffrent de certaines lacunes selon la nature des données traitées, la précision, la robustesse et le temps de calcul. Dans ce travail, l'objectif principal porte sur l'intégration et l'exploitation des concepts liés à la théorie des graphes dans la conception des systèmes cryptographiques. Dans notre première contribution, nous avons proposé un nouveau système de chiffrement par blocs (BC-GT) qui met en jeu les concepts fondamentaux de la théorie des graphes afin de faciliter les manipulations des données brutes. Le principe s'articule sur la génération des graphes pondérés à partir d'un nouvel usage des circuits Hamiltoniens. Concernant la génération des sous-clés, nous avons fait appel à un générateur de sous-clés très particulier qui a été soigneusement conçu pour produire les clés de chiffrement conformément aux spécifications du système. Les résultats expérimentaux obtenus démontrent que notre système de chiffrement est robuste contre les attaques statistiques. Quant à la deuxième contribution, nous avons proposé une nouvelle variante améliorée du premier système. Notre proposition est en fait un nouveau système de chiffrement par blocs qui procède par la représentation des messages en clair à l'aide des circuits hamiltoniens disjoints, puis les traite sous forme d'une matrice d'adjacence dans une phase de pré-chiffrement. Les résultats expérimentaux obtenus démontrent que notre nouveau système de chiffrement est aussi robuste contre les attaques statistiques, à travers le test de DIEHARD, et présente à la fois une bonne confusion et diffusion.

Mots clés : Cryptosystème, Théorie des graphes, Circuits hamiltoniens, Matrice d'adjacence, Chiffrement, Chiffrement par blocs.

Abstract

In the context of this work, we are interested in the design and development of new approaches for computer security based on the properties of graph theory. The various techniques developed in the literature suffer from certain shortcomings in terms of the nature of the processed data, accuracy, robustness and computation time. In this work, the main objective is to integrate and exploit the concepts related to graph theory to achieve a relatively high level of security. First, we presented a new block cipher system (BC-GT) that uses the fundamental concepts of graph theory to facilitate the manipulation of raw data. The principle is based on the generation of weighted graphs from a new use of Hamiltonian circuits. Concerning the generation of sub-keys, we have used a very particular sub-key generator that has been carefully designed to produce the encryption keys according to the system specifications. The experimental results obtained demonstrate that our encryption system is robust against statistical attacks. Subsequently, we presented a new and improved variant of the first system. Our proposal is in fact a new block cipher system that proceeds by representing plaintext messages using disjoint Hamiltonian circuits and then treating them as an adjacency matrix in a pre-encryption phase. The experimental results obtained show that our new encryption system is also robust against statistical attacks, through the DIEHARD test, and has both good confusion and diffusion.

Keywords :Cryptosystem, Graph theory, Hamiltonian circuits, Adjacency matrix, Encryption, Block cipher.

Introduction générale

Contexte et problématique

La cryptologie désigne de manière générale tout ce qui concerne la sécurité et la confidentialité des communications et des informations. Son domaine ne se limite pas seulement aux domaines du chiffrement ou de la cryptanalyse, mais aussi à la signature électronique, à l'authentification, au vote électronique, ou à tout protocole nécessitant de sécuriser des données. La cryptographie (et son opposée la cryptanalyse) est une branche importante de la cryptologie. La cryptographie concerne tout ce qui a trait au chiffrement des données et des communications, c'est-à-dire le fait de rendre inintelligibles ces données aux personnes ne possédant pas de clés de déchiffrement. La cryptanalyse, au contraire, désigne toute méthode cherchant à extraire de l'information de données chiffrées, sans en posséder la clé. La cryptographie est un élément essentiel de la communication numérique. Les banques, les entreprises, les gouvernements et autres ont tous intérêt à communiquer sur Internet. Peut-être plus important encore, ils ont tous intérêt à communiquer en toute sécurité : avec l'assurance que le message transmis n'avait pas été modifié, n'avait pas été lu par une autre partie et provenait véritablement de celui dont l'expéditeur prétend. En conséquence, des systèmes cryptographiques ont été développés pour atteindre ces objectifs. Le but de la cryptographie est donc la construction de schémas, qui peuvent contribuer à la maintenance de la sécurité souhaitée même après des tentatives malveillantes, tout en assurant la confidentialité, l'authenticité, la disponibilité et la non-répudiation des données [Menezes *et al.*, 2018].

La cryptographie est une discipline de la cryptologie qui comporte aussi la cryptanalyse. Cette dernière correspond à l'étude de techniques mathématiques visant à trouver des faiblesses dans les cryptosystèmes [Menezes *et al.*, 2018]. La cryptographie peut être classée en trois catégories : la cryptographie symétrique, asymétrique et hybride. La cryptographie symétrique consiste à utiliser une clé secrète lors de son processus, tels que les algorithmes DES [FIPS, 1980], IDEA [Meier, 1993], AES [Rijmen et Daemen, 2001], etc. Tandis que la cryptographie asymétrique repose sur l'utilisation de deux clés différentes, une clé privée et une clé publique (RSA [Smart, 2016g], ElGamal [Smart, 2016h], Diffie-Hellman [Menezes *et al.*, 1996b], etc). En outre, la cryptographie hybride comporte à la

fois la cryptographie symétrique et asymétrique, elle a pour but de combiner les avantages des deux systèmes sans souffrir de leurs inconvénients respectifs.

Historiquement, de simples chiffrements ont tenté de cacher un message à d'autres parties, le chiffrement César est un exemple de base. D'autres ont été développés pour résister aux attaques au fur et à mesure que ces attaques se développaient. Et avec l'avènement de l'ordinateur, des modifications majeures ont été nécessaires pour vaincre les capacités de calcul croissantes. Malheureusement, la plupart des algorithmes de chiffrement proposés s'avèrent complexes et nécessitent un temps de traitement très important. De ce fait, nos motivations ont été inscrites dans ce contexte.

Aujourd'hui, la cryptologie moderne a pu employer un ensemble d'outils mathématiques, ce qui a permis des gains en matière de performance et d'efficacité. La théorie des graphes est un domaine qui a été considéré très prometteur en ce sens, du fait qu'il fournit des concepts capables de résoudre des problèmes dans tous les domaines liés à la notion de réseau.

La théorie des graphes est une discipline informatique et mathématique qui étudie les graphes. De manière générale, un graphe permet de représenter la structure, les connexions d'un ensemble complexe en exprimant les relations entre ses éléments : réseau de communication, réseaux routiers, circuits électriques, etc. Les graphes constituent donc une méthode de pensée qui permet de modéliser une grande variété de problèmes en les ramenant à l'étude de sommets et d'arcs(ou arrêtes). Le problème des sept ponts de Königsberg (en 1736) est un problème mathématique connu pour être à l'origine de la théorie des graphes. Depuis le début du 20^e siècle, elle constitue une branche à part entière des mathématiques, grâce aux travaux de König, Menger, Cayley puis de Berge et d'Erdös.

En effet, il s'agit essentiellement de modéliser des problèmes. Nous exprimons le problème en termes de graphes de sorte qu'il relève d'un problème de la théorie des graphes que nous savons le plus souvent résoudre. Les solutions de problèmes de graphes peuvent être faciles et efficaces (le temps nécessaire pour les traiter informatiquement étant raisonnable car ils dépendent polynomialement du nombre de sommets du graphe) ou difficiles (le temps de traitement étant alors exponentiel) auquel cas nous utilisons une heuristique, c'est-à-dire un processus de recherche d'une solution optimale.

La théorie des graphes est devenue un élément clé dans de nombreuses applications dans le domaine informatique. C'est un concept récent qui a été intégré avec succès pour fournir des algorithmes cryptographiques plus puissants, et difficiles à casser par n'importe quel logiciel moderne. Cette discipline apporte une grande contribution en tant qu'un nouvel outil pour les agences gouvernementales et les organisations intéressées par la sécurité, pour l'employer dans le développement de diverses techniques de chiffrement ou dans les communications des données sophistiquées. Cela a conduit à une large application des concepts de la théorie des graphes en cryptographie, car de nombreux problèmes NP-difficiles proviennent de cette théorie.

De ce fait, on a été motivé par la conception d'un nouveau système cryptographique

basé sur la théorie des graphes afin de fournir à la fois une sécurité élevée et surtout des traitements simples des ressources. Les travaux de recherche présentés dans cette thèse s'inscrivent dans le cadre du développement de nouveaux systèmes de chiffrement basés sur la théorie des graphes afin de fournir un niveau de sécurité relativement élevé avec un calcul rapide.

Présentation des Contributions

L'objectif principal de cette thèse est la conception de nouveaux systèmes de sécurité basés sur la théorie des graphes. Nos contributions peuvent être présentées comme suit :

- **BC-GT, un nouveau système de chiffrement par blocs en utilisant les propriétés de la théorie des graphes [Bekkaoui *et al.*, 2020a] :**
  Nous proposons un nouveau système de chiffrement basé essentiellement sur les propriétés de la théorie des graphes notamment les circuits hamiltoniens disjoints qui sont très prometteurs pour la représentation du texte clair, et du principe "Diviser pour régner", pour simplifier les traitements et faciliter le chiffrement.
  Notre approche propose une nouvelle utilisation des circuits hamiltoniens et de matrice d'adjacence en utilisant un générateur des sous-clés particulier qui a été soigneusement conçu pour produire les clés de chiffrement conformément aux spécifications du système. Les résultats expérimentaux obtenus démontrent que notre système de chiffrement proposé est robuste contre les attaques statistiques, en particulier le test DIEHARD.

- **EBC-GT, un nouveau système de chiffrement par blocs en utilisant les circuits hamiltoniens disjoints [Bekkaoui *et al.*, 2021] :**
  Notre proposition est en fait un nouveau système de chiffrement par blocs qui procède en représentant les messages en clair à l'aide de circuits hamiltoniens disjoints, puis en les traitant comme une matrice d'adjacence dans une phase de pré-chiffrement. Le système proposé repose sur un générateur de sous-clé particulier ayant été prudemment mis au point pour permettre la génération de clés de chiffrement répondant aux exigences du système. Les résultats expérimentaux démontrent que le système de chiffrement suggéré est bien protégé contre les attaques statistiques et présente à la fois une bonne confusion et une bonne diffusion.

Organisation du rapport

Ce document comprend quatre chapitres :

- **Chapitre 1 :** Présente des généralités sur la sécurité informatique et les principaux concepts de la théorie des graphes. Nous terminons ce chapitre en citant les différentes modes de représentations d'un graphe.

- **Chapitre 2 :** Cite les différentes applications de la théorie des graphes en cryptographie en présentant quelques algorithmes de chiffrement basés sur la théorie des graphes.
- **Chapitre 3 :** Décrit notre première contribution à la confidentialité. Nous détaillons notre algorithme de chiffrement par blocs BC-GT.
- **Chapitre 4 :** Décrit notre deuxième contribution à la confidentialité. Nous détaillons notre algorithme de chiffrement par blocs EBC-GT.

Pour clôturer ce mémoire, une conclusion générale présente un bilan sur l'ensemble des travaux réalisés. Ensuite, nous allons dresser quelques perspectives et améliorations qui peuvent être une suite logique des travaux réalisés et présentés dans ce mémoire.

Chapitre

1 Généralités sur la cryptographie et la théorie des graphes

Sommaire

Introduction

Dans ce chapitre, nous introduisons les généralités sur la cryptographie symétrique et asymétrique en mettant l'accent sur les chiffrements par blocs, les générateurs de nombres aléatoires, les Codes d'authentication de Messages (MACs) et les fonctions de hachages. Ensuite, nous abordons les attaques contre les mécanismes des systèmes cryptographiques. Le chapitre 1 est organisé de la manière suivante : Dans la section 1.1, nous décrivons brièvement, les algorithmes de cryptographie à clé secrète. Dans la section 1.2, nous nous intéressons aux concepts fondamentaux des graphes, leurs différentes définitions et leurs propriétés en citant les différents modes de représentations d'un graphe. Enfin, le bilan du chapitre est présenté dans la section conclusion.

1.1 Cryptographie

Depuis l'antiquité, la sécurité est un élément fondamental pas seulement pour protéger les messages militaires et diplomatiques, mais aussi pour protéger les communications privées, visant à assurer la confidentialité des données. Après l'évolution de l'informatique et l'apparition de l'internet, l'ère de la communication de ce jour a accru l'importance de l'échange de données financières, du traitement de l'image, de la biométrie et des transactions du commerce électronique qui, à son tour, a rendu la sécurité des données une question importante. Par conséquent, d'autres exigences sont apparues : l'authentification, l'intégrité, la disponibilité et la non-répudiation. La sécurité informatique est donc l'ensemble des règles et techniques qui assurent que les ressources du système informatique (matérielles ou logicielles) d'une organisation sont utilisées uniquement dans le cadre où il est prévu qu'elles le soient.

En effet, le développement des systèmes de sécurité conçus pour protéger les communications et les données électroniques est devenu de plus en plus important. Cette protection requise englobe.

Un ensemble d'exigences et de solutions basées sur la cryptographie, un élément primaire dans le domaine de sécurité informatique. La cryptographie est une discipline de la cryptologie qui comporte aussi la cryptanalyse. Elle est la science de protection des données secrètes échangées à travers un canal considéré non sécurisé, alors que la cryptanalyse correspond à l'étude de techniques mathématiques visant à trouver des faiblesses dans les systèmes cryptographiques. La cryptographie classique se distingue de la cryptographie moderne. Les méthodes de la cryptographie classique reposent sur l'utilisation de deux opérations essentielles : la substitution et la transposition. La substitution consiste à remplacer des lettres par d'autres ou par des symboles. Quant à la transposition signifie la permutation des lettres du message pour le rendre inintelligible. À titre d'exemple : Le code de César, le chiffrement de Hill, le chiffrement de Vigenère, les systèmes à clés jetables, etc. Le but principal de la cryptographie classique est de transmettre les données de manière confidentielle. De plus, la cryptographie moderne s'intéresse aux problèmes de sécurité

des communications. Elle vise à assurer les exigences suivantes [Menezes *et al.*, 2018] :

- **La confidentialité** : est un mécanisme où les données transmises ne doivent être compréhensibles que par les entités autorisées [Menezes *et al.*, 2018].
- **L'intégrité** : assure que les données échangées n'ont pas subi d'altérations par des entités non autorisées [Menezes *et al.*, 2018].
- **L'authentification** est un processus lié à l'identification d'une entité. Les entités entrant dans une communication devraient s'identifier chacune à l'autre. Par exemple, une entité A peut se reconnaître en prouvant l'identité d'une entité B qui connait un secret S [Menezes *et al.*, 2018].
- **La non-répudiation** est un mécanisme qui empêche une entité de nier ses actes passés. Lorsque des conflits surviennent en raison d'une entité refusant que certaines actions ont été prises, une procédure impliquant un tiers de confiance est nécessaire pour résoudre la situation.
- **La disponibilité** permet le bon fonctionnement du système dans les plages d'utilisation prévues et garantit l'accès aux données pour les entités autorisées avec un temps de réponse attendu [Menezes *et al.*, 2018].

La cryptographie comporte les algorithmes qui servent à transformer un message compréhensible à un message incompréhensible par les entités non autorisées. Il existe trois types principaux de la cryptographie moderne symétrique, asymétrique et hybride. La cryptographie symétrique utilise une seule clé secrète pour chiffrer et déchiffrer. La cryptographie asymétrique repose sur l'usage de deux clés, une publique pour le chiffrement et une privée pour le déchiffrement. Finalement, la cryptographie hybride qui comporte à la fois la cryptographie symétrique et asymétrique. Par conséquent, nous utilisons une littérature standard sur la cryptographie telle que les livres : "Cryptography made simple" de "N. P. Smart" [Nigel P, 2016] et "Handbook of applied cryptography" de "A. J. Menezes et al" [Menezes *et al.*, 2018].

1.1.1 Cryptographique à clé secrète

La cryptographie à clé secrète utilise pour les échanges entre deux correspondants, une seule clé. Cette clé est utilisée à la fois pour le chiffrement du texte clair et pour le déchiffrement du texte chiffré. Pour cette raison un tel système est aussi appelé un cryptosystème symétrique (illustré par la FIGURE 1.1). Les systèmes de cryptographie à clé secrète sont divisés en trois catégories principales : chiffrement par blocs, chiffrement à flot et code d'authentification de messages (Code d'authentication de Message (MAC)). Ces types sont utilisés pour assurer respectivement la confidentialité et l'authenticité des données.

Dans cette section, nous décrivons, en moins de détails, ces algorithmes de cryptographie à clé secrète.

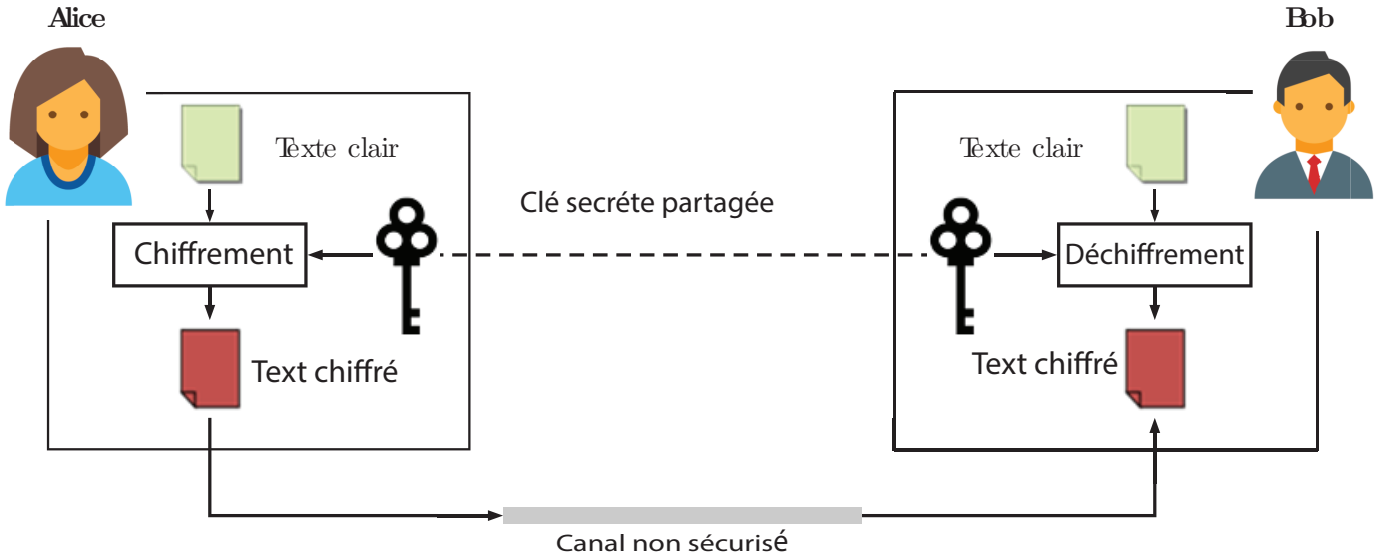


FIGURE 1.1 – Cryptographie symétrique

1.1.1.1 *Chiffrement par blocs*

Le chiffrement par bloc est une transformation inversible pour avoir un déchiffrement unique, qui prend en entrée une clé de chiffrement K et des blocs de texte clair (P) de n-bits et donne des blocs de texte chiffré (C) de n bits ; n est la taille du bloc [Alfred J Menezes et Vanstone, 1996a]. Il permet de chiffrer un message en clair (M) de longueur arbitraire, en divisant le message M en blocs de n bits. En utilisant une fonction définie, chaque bloc est chiffré avec une clé K de k-bits. Généralement, le dernier bloc implique une méthode de zero-padding (par exemple concaténant la valeur $10\ldots0|\mathrm{M}|$ au message M, où |M| indique la longueur en bits de M) si nécessaire (pour que M soit un multiple de k-bits) et tous les blocs sont concaténés pour obtenir le texte chiffré.

Le processus de déchiffrement utilise l'inverse de la fonction de chiffrement, où la fonction prend en entrée le texte chiffré et la clé secrète K.

Définition 1.1.

Mathématiquement, un chiffrement par bloc est une transformation $\mathrm{E} : (0,1)^n * (0,1)^k \longrightarrow (0,1)^n$, tel que pour chaque clé $K \in (0,1)^k$ et un texte en clair P, $E(P,K)$ est une bijection inversible de $(0,1)^n$ à $(0,1)^n$ noté $E_K(P)$.

Le chiffrement par bloc doit être effectué avec l'un des différents modes de chiffrement. Les premiers modes standardisés ont été proposés par NIST pour l'algorithme Data Encryption System (DES) [FIPS, 1980] et pour Advanced Encryption System (AES) dans

[Rijmen et Daemen, 2001]. Les modes principaux de base comprennent Electronic CodeBook (ECB), Cipher Block Chaining (CBC), Cipher FeedBack (CFB) ou Output FeedBack (OFB) et Counter (CTR) [Alfred J Menezes et Vanstone, 1996a]. Ces modes sont typiquement utilisés pour concevoir d'autres systèmes cryptographiques, tels que les chiffrements à flots, les fonctions de hachages ou les codes d'authentification de message.

Electronic CodeBook (ECB)

Il représente le mode de fonctionnement le plus naïf possible. Étant donné un message clair, la fonction de chiffrement est appliquée directement et indépendamment sur chaque bloc de texte en clair. La séquence résultante des blocs de sortie représente le texte chiffré. Réciproquement, l'opération du déchiffrement est appliquée directement et indépendamment à chaque bloc de texte chiffré. La séquence résultante de blocs de sortie est le texte en clair.

Pour chiffrer un texte en mode ECB, on effectue donc les opérations suivantes :

$$C_i = E_k(P_i), \text{pour i} >= 0 \tag{1.1}$$

Pour déchiffrer un texte en mode ECB, on effectue les opérations suivantes :

$$P_i = D_k(C_i), \text{pour i} >= 0 \tag{1.2}$$

Le mode de chiffrement ECB est illustré par la FIGURE 1.2.

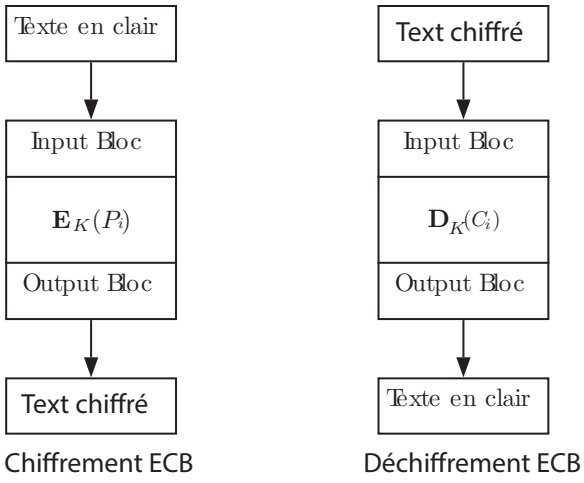


FIGURE 1.2 – Mode de chiffrement ECB

Le processus de chiffrement ici est déterministe, en utilisant la même clé le résultat de chiffrement d'un bloc de texte en clair est toujours le même bloc de texte chiffré, donc le mode de fonctionnement ECB n'est pas sécurisé contre les attaques CPA. Si cette propriété est indésirable dans des applications particulières qui nécessitent un niveau élevé de sécurité où le mode ECB ne devra pas être utilisé.

Cipher Block Chaining(CBC)

Le mode de chiffrement avec chaînage de blocs, en anglais "Cipher Block Chaining" (CBC), est un des modes les plus populaires.

Il offre une solution à la plupart des problèmes du mode ECB. Le chaînage utilise une méthode de rétroaction comme le résultat du chiffrement du bloc précédent est réutilisé pour le chiffrement du bloc courant. Plus précisément, l'opérateur binaire XOR est appliqué entre le bloc actuel de texte en clair et le bloc précédent de texte chiffré et on applique ensuite l'algorithme de chiffrement au résultat de cette opération. Pour le tout premier bloc, un bloc ayant un contenu aléatoire, appelé vecteur d'initialisation (Initialization Vector), est généré et utilisé pour l'application de l'opération XOR.

Ainsi, chaque bloc chiffré ne dépend non seulement du bloc de texte en clair correspondant, mais également de tous les blocs chiffrés qui le précèdent. Pour chiffrer un texte en mode CBC, on effectue par conséquent les opérations suivantes :

$$\begin{cases} C_i = E_k(P_i \oplus C_{i-1}), & \text{pour } i >= 1 \\ C_0 = E_k(P_0 \oplus VI) \end{cases} \tag{1.3}$$

Pour déchiffrer un bloc de texte chiffré en mode CBC, on applique l'algorithme de déchiffrement au bloc chiffré et puis on le combine par ou exclusif avec le bloc chiffré précédent, respectivement avec le vecteur d'initialisation, dans le cas du premier bloc. On effectue donc les opérations suivantes :

$$\begin{cases} P_i = C_{i-1} \oplus D_k(C_i), & \text{pour i} >= 1 \\ P_0 = VI \oplus D_k(C_0) \end{cases} \tag{1.4}$$

Le mode de chiffrement CBC est illustré par la FIGURE 1.3.

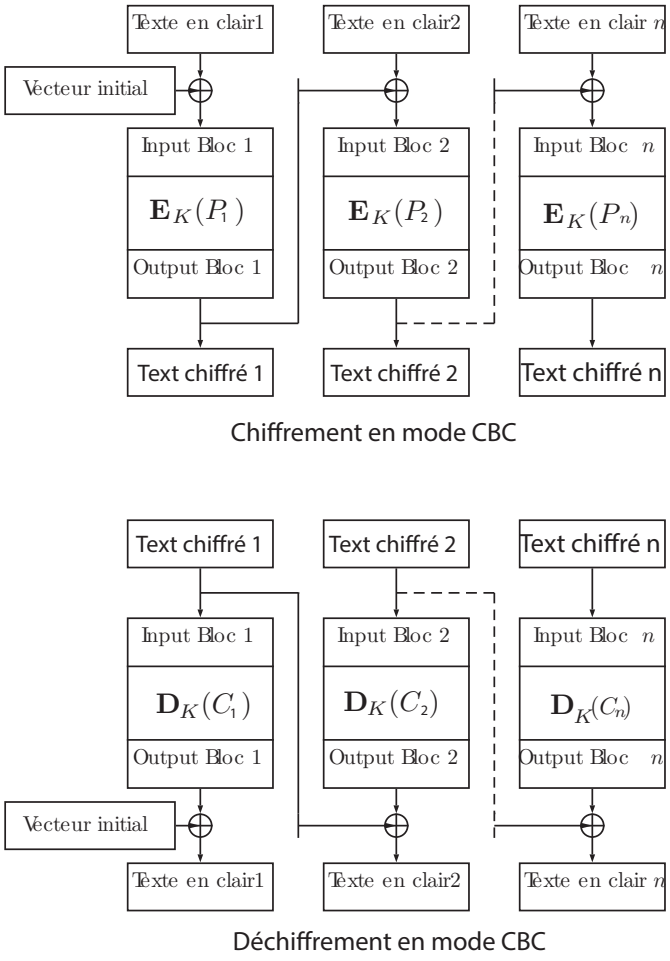


FIGURE 1.3 – Mode de chiffrement CBC

Le rôle du vecteur d'initialisation est d'empêcher que si deux textes en clair débutent de la même façon, les textes chiffrés correspondants le font également.

Il n'est pas obligatoire de le choisir aléatoirement ; il peut aussi être un numéro d'ordre qui est augmenté après chaque message. La seule chose importante est qu'il doit être différent pour chaque message chiffré avec la même clé.

Grâce au vecteur d'initialisation, même un bloc de texte en clair identique donnera un message chiffré différent. Il n'est plus possible pour un espion d'apprendre la moindre information concernant le texte en clair à partir du texte chiffré. Il n'est pas nécessaire non plus de tenir secret le vecteur initialisation. Généralement on le transmet en clair avec le texte chiffré.

Cipher FeedBack (CFB)

Avec les modes EBC ou CBC, le chiffrement ne peut pas commencer avant qu'un bloc complet de données n'ait été reçu, ce qui peut poser un problème dans certaines applications réseaux, où les données à traiter se présentent sous forme de paquets de la taille de l'octet.

Le mode de chiffrement à rétroaction, en anglais "Cipher Feedback" (CFB)(illustré par la FIGURE 1.4), permet de chiffrer les données par unités plus petites que la taille de bloc. On appelle mode de chiffrement à rétroaction à k bits le mode CFB qui chiffre les données par unités de s bits. Souvent on chiffre un caractère ASCII à la fois, c'est-à-dire s = 8, mais s peut en principe prendre n'importe quelle valeur, pourvu qu'elle ne dépasse pas la taille d'un bloc.

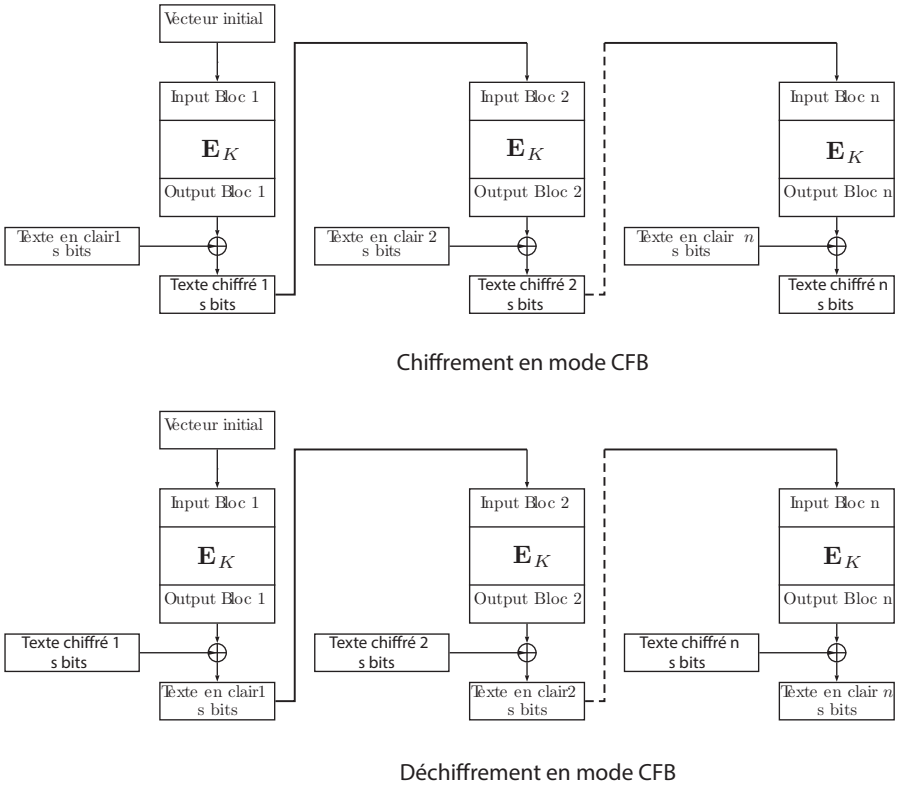


FIGURE 1.4 – Le mode de chiffrement à rétroaction (CFB)

En mode CFB, on manipule un registre de décalage de la taille d'un bloc de texte en clair. Pour commencer, le registre de décalage est rempli par un vecteur d'initialisation. On chiffre ensuite le registre de décalage avec l'algorithme de chiffrement utilisé et on combine les k bits les plus significatifs, c'est-à-dire les s bits les plus à gauche du résultat par un ou exclusif avec les s premiers bits du texte en clair pour obtenir ainsi les k premiers bits du texte chiffré qu'on peut alors transmettre. On place ce bloc de texte chiffré dans les s bits les plus à droite du registre de décalage et on décale tous les autres bits du registre

de décalage de k positions vers la gauche en coupant ses s premiers bits. On continue alors de chiffrer le reste du texte en clair de la même manière.

En mode CFB, le déchiffrement se fait en appliquant de nouveau la fonction de chiffrement de l'algorithme au même registre de décalage que pour le chiffrement et en ajoutant les s premiers bits de ce résultat au bloc précédent du texte chiffré.

Dans ce qui suit, nous discutons les constructions de base des chiffrements par blocs à savoir les schémas de Feistel et les réseaux de Substitution-Permutation.

a. Schéma de Fiestel

La construction de Feistel est une structure symétrique, il est communément appelé réseau de Feistel. Elle consiste à diviser chaque bloc en deux parties égales (R_i et L_i), où R_i représente la partie droite et L_i représente la partie gauche, pour $0 \leq i \leq r$ ou r est le nombre de tours. La FIGURE 1.5 illustre un seul tour pour le réseau de Feistel.

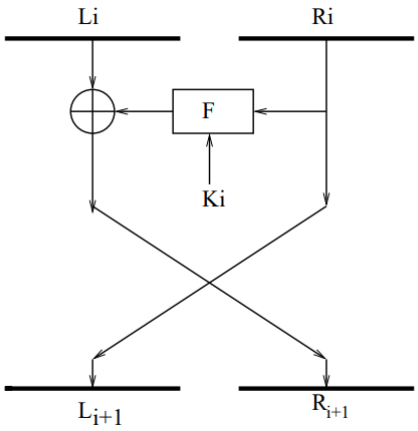


FIGURE 1.5 – Un Tour de chiffrement Fiestel

Afin de chiffrer un message, une fonction non linéaire f qui dépend d'une clé de tour K, est appliquée à l'une des parties (R_i et L_i) impliquant l'opérateur XOR. Ainsi, le chiffrement correspondant à un seul tour d'une construction Feistel peut être écrit comme suit :

$$\begin{cases} L_{i+1} = L_i \\ R_{i+1} = L_i \oplus f(L_i, R_i) \end{cases} \tag{1.5}$$

Où K, est la sous-clé du ième tour correspondante, qui est dérivé de "key scheduling algorithm" (l'algorithme d'extraction de clés) spécifié. Ce processus est itéré tant qu'il est défini par le nombre de tours r. Enfin, les deux parties sont concaténées ($Lr \| Rr$) représentant le texte chiffré final.

Notons que la fonction f ne doit pas nécessairement être bijective. Le déchiffrement est obtenu de manière similaire au chiffrement, ne nécessitant qu'un renversement d'ordre des clés et l'échange de fonctionnement de L_i et R_i. L'architecture similaire de chiffrement et de déchiffrement est l'avantage principal d'un schéma de Feistel, elle contribue évidemment à une implémentation dans les logiciels et les matériels à faible coût (càd ressources limitées). Le premier modèle de chiffrement moderne construit est le Data Encryption System (DES).

En résumé, les propriétés principales des chiffrements par blocs basé sur les réseaux Fristel peuvent être reprises comme suit :

- Une fonction de tour simple la fonction appliquée sur une des parties gauche ou droite rend le code facile à étudier.
- La similarité : il fournit un déchiffrement plus facile, car le déchiffrement est similaire au chiffrement. Par conséquent, la fonction de tour ne doit pas être bijective. Il est toutefois recommandé que la fonction soit proche de la surjective. Sinon, l'attaque proposée par V.Rijmen et al. Dans [Rijmen *et al.*, 1997] devient possible.
- Un petit nombre de tours : est utilisé si la fonction de tour et key scheduling algorithm (un algorithme d'extraction de clés) sont robustes.

La FIGURE 1.6 illustre le réseau de Feistel à n tours.

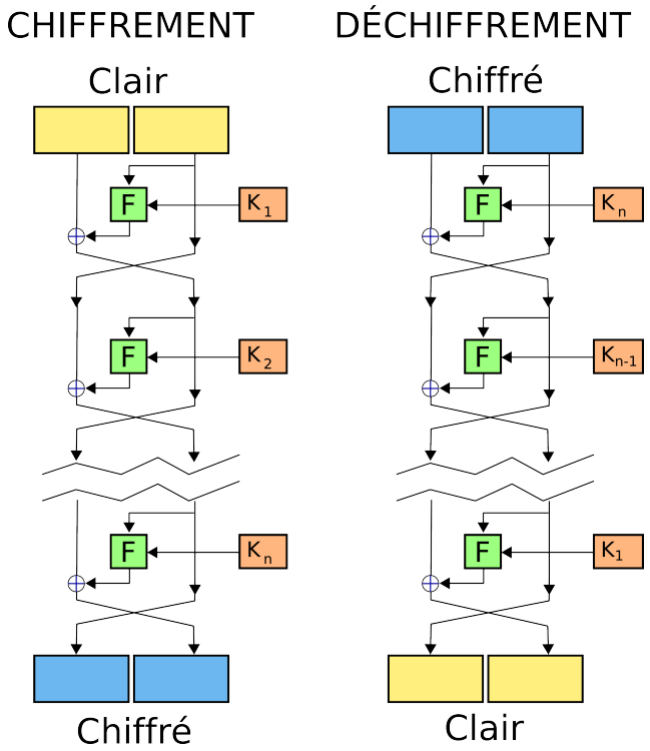


FIGURE 1.6 – Réseau de Feistel à n tours

b. Réseaux de substitutions-permutations (SPN)

Une autre catégorie pour concevoir un chiffrement par bloc est le réseau de substitution permutation (SPN : Substitution permutation network en anglais), voir la FIGURE 1.7. Le chiffrement SPN implique trois couches fondamentales, une couche d'addition de clé, une couche de substitution et une couche de permutation. Alors, la fonction de tour du chiffrement SPN comporte les opérations suivantes :

1. une couche de substitution S, qui remplace l'état d'une manière non linéaire par substitution en parallèle de groupes de bits selon certaines tables de substitution, généralement appelées boîtes de substitution ou boîtes-S (S-box en anglais), ainsi appelée car on substitue une autre séquence de bits à une séquence de bits. Les boîtes-S dans le chiffrement SPN sont essentiellement bijectives.

2. une couche de permutation linéaire P appelée boite de permutation (P-box en anglais) qui permute les positions des bits.

3. ajout d'une sous-clé K_i appliquant souvent un XOR bit à bit.

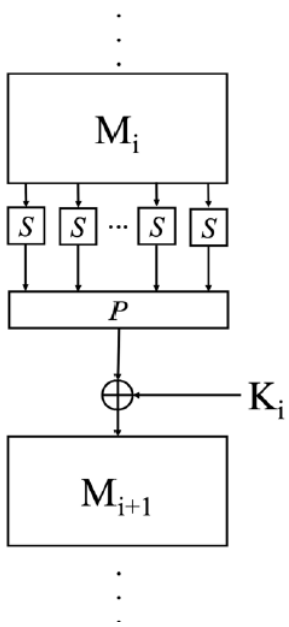


FIGURE 1.7 – Un Tour d'un réseau de substitution-permutation (SPN)

En plus de ces trois opérations, la fonction de tour comprend parfois une opération d'addition d'un tour constant, pour distinguer entre les tours individuels les uns des autres, ce qui restreint certains types d'attaques telles que les attaques de diapositives 'slide attacks' [Biryukov et Wagner, 1999]. La variante standard de la fonction d'un tour peut être décrite comme suit : $M_{i+1} = P(S(M_i)) + K_i$ tel que M est le bloc à chiffrer.

Généralement, la couche de substitution et la couche de permutation garantissent respectivement les propriétés de confusion et de diffusion, ces dernières vont être présentées à la section suivante. Les chiffrements SPN exigent que chaque fonction de tour soit inversible pour permettre le déchiffrement.

c. Confusion et Diffusion

Claude Shannon a introduit deux propriétés importantes [Shannon, 1949] que faudrait vérifier un bon algorithme de chiffrement. Il s'agit de la confusion d'une part et la diffusion d'une autre part. La propriété de confusion crée une dépendance solide entre le clair, le chiffré et la clé, ceci pour qu'un attaquant ne puisse pas obtenir des informations sur la clé pour chaque paire (clair,chiffré). Afin d'atteindre la confusion, deux méthodes sont utilisées. D'un côté, on peut utiliser des opérateurs non-linéaires comme le ET logique ou encore utiliser l'addition modulaire (qui n'est pas linéaire par rapport au XOR) [Thomas, 2015]. D'un autre côté utiliser des boîtes de substitutions (boîtes-S), qui sont choisies de sorte à maximiser la confusion d'un système de chiffrement. La propriété de diffusion est une notion quantitative qui se réfère à la dépendance de chaque partie du clair et de la clé. Autrement, de petits changements en entrée doivent avoir un effet important en sortie. Le but de la diffusion est de bien mélanger les sorties de plusieurs S-box opérant sur des parties distinctes du chiffrement. Typiquement, la diffusion est assurée par des boites de permutations (boites-P) linéaires qui font appel à des fonctions avec un grand domaine d'entrée et de sortie. Pour des raisons d'efficacité, les opérations de diffusion sont préférées d'être simples.

1.1.1.2 Chiffrement à flot

Contrairement aux chiffrements par blocs, qui chiffrent de gros blocs de texte clair en utilisant une transformation fixe, le chiffrement à flot appelé parfois chiffrement par flux' chiffre un texte en clair en le combinant avec une suite chiffrante sur $F_2\{0,1\}$ de la même taille que le message à chiffrer à une loi du groupe F_2, par exemple le Xor, appliqué bit à bit [Alfred J Menezes et Vanstone, 1996b]. Cette suite est habituellement obtenue par un générateur de nombres pseudo-aléatoires (voir la SECTION 1.1.1.3). Par conséquent, la sécurité du système repose sur la qualité de la suite chiffrante générée. La FIGURE 1.8 montre la structure générale d'un chiffrement à flot. L'algorithme de chiffrement RC4 [Smart, 2016c] qui est développé en 1987 par Ron Rivest, est un exemple d'algorithme de chiffrement à flot, fonctionne avec des clés de tailles variables.

Les chiffrements à flot sont généralement subdivisés en deux types fondamentaux : synchrone et asynchrone. Dans le chiffrement à flot synchrone, la suite chiffrante est générée uniquement en utilisant la clé secrète du chiffrement. Dans le chiffrement à flot asynchrone, la suite chiffrante est générée en utilisant la clé secrète et un nombre de bits des chiffres précédents [Alfred J Menezes et Vanstone, 1996b].

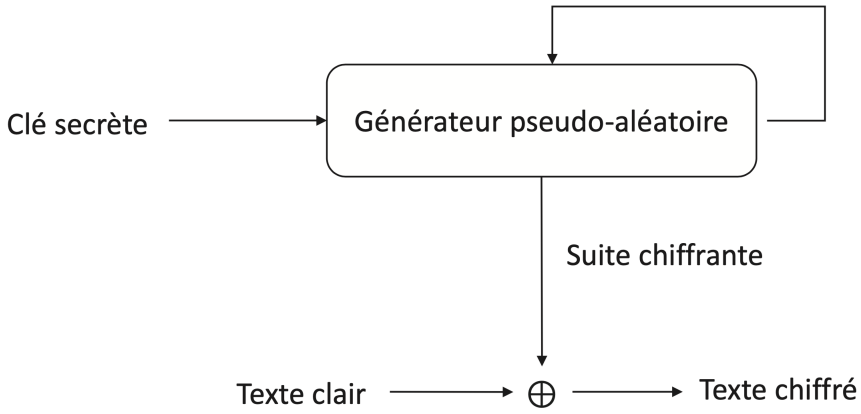


FIGURE 1.8 – Structure générale d'un chiffrement à flot

1.1.1.3 Générateur de nombre aléatoire

L'aléatoire a aujourd'hui une importance capitale, en physique pour la simulation de phénomènes trop complexes pour être décrits, dans les Jeux, en statistique pour l'étude des échantillons. En informatique en phase de tests des programmes, dans les télécommunications ou tout simplement en cryptographie où il joue un rôle vital pour produire des clés de chiffrement, des vecteurs d'initialisation, des bits de padding, des challenges dans les protocoles d'authentification. La première utilisation des générateurs en cryptographie, c'est le cas du chiffrement à flot, il est donc devenu nécessaire de pouvoir générer des nombres aléatoires. Ainsi un générateur de nombres aléatoires (random number generator (RNG)) est un algorithme permettant de générer une séquence binaire uniforme, imprédictible et indépendante [Menezes *et al.*, 1996a].

Trois grandes familles de générateurs aléatoires existent à savoir ceux qui sont extraits à partir de phénomènes physiques, on les appelle des générateurs vraiment aléatoires (true random number generator-TRNG), ceux qui sont créés grâce à des algorithmes utilisant une graine d'initialisation appelés : générateurs pseudo-aléatoires (pseudorandom number generator-PRNG) et les générateurs hybrides qui fait injecter de l'aléatoire provient d'un générateur vraiment aléatoire. Dans cette partie, on s'intéressa uniquement à la manière de générer des nombre presque aléatoire : on parle donc de générateurs de nombres pseudo-aléatoires.

Un générateur pseudo-aléatoire est une fonction qui prend en entrée une graine aléatoire de petite taille et renvoie une suite de bits de grande taille. Cette suite est appelée suite pseudo-aléatoire et doit être difficilement distinguable d'une suite de bits aléatoires, c'est-à-dire qu'elle ressemble à une suite vraiment aléatoire. Dans les GPAs, il faut s'assurer d'une part que le générateur est sûr, c'est-à-dire il constitue une période longue, d'une autre part il est imprédictible (=calculatoirement sur), c'est-à-dire qu'il doit être calculatoirement difficile de prédire, l'élément suivant ou précédent à partir d'une sous-suite de la suite pseudo-aléatoire. De plus, la sécurité d'un GPA dépend du secret de la graine

d'initialisation et de propriétés statistiques, c'est-à-dire que le générateur est capable de générer des suites de bits satisfaisant les propriétés statistiques de suites vraiment aléatoires. D'ailleurs, une autre caractéristique est désirée pour parler de générateur pseudo aléatoire cryptographiquement sur : Il est difficile de savoir les éléments précédents à partir d'un élément de l'état interne du générateur, pourtant on est capable de calculer les éléments suivants du GPA.

En effet, plusieurs constructions de générateurs pseudo-aléatoires cryptographiquement sûrs ont été effectués, citons Blum et Micali [Blum et Micali, 1984], Blum, Blum et Shub [Blum *et al.*, 1986] ou Micali et Schnorr [Micali et Schnorr, 1991] à titre d'exemple. Cependant, ces générateurs sont toutefois très lents et ne sont pas utilisés en pratique.

Dans ce qui suit nous citons brièvement quelques méthodes d'un générateur pseudo aléatoire. Ensuite, un détail sur les exigences des GPAs est explicité.

a. Méthodes des générateurs de nombres pseudo-aléatoires

a.1 GPAs basés sur les méthodes congruentielles linéaires

Ces générateurs ont été introduits en 1948 par D.H Lehmer, et sont aujourd'hui encore très populaires. En effet, il s'agit de générateurs nécessitant peu d'opérations et la production des nombres est rapide. Une séquence de nombres aléatoires X_n est créée de la manière suivante :

$$X_{n+1} = (a \cdot X_n) \text{ mod(m)} \tag{1.6}$$

Où m est le module, a le multiplicateur et b l'incrément.

Une fois les paramètres choisis, la suite dépend entièrement de la graine X_0. Un choix judicieux des paramètres a, b et m permet de maximiser la qualité du générateur.

En pratique, on essaie de prendre m, le plus grand possible. Pour avoir une suite longue aléatoire. Souvent, on prend $m = 2b$, où b est le nombre de bits utilisés pour représenter un entier positif dans la machine. Cependant, la sortie du générateur congruentiel linéaire est prédictible et, par conséquent, ce générateur est complètement inapproprié pour les applications cryptographiques [Boyar, 1989, Krawczyk, 1992].

a.2 Méthode de Fibonacci

Cette méthode est basée sur la suite de Fibonacci dont la séquence aléatoire est générée à partir de l'expression suivante :

$$X_n = X_{n-1} + X_{n-2} \text{ mod(m)} \tag{1.7}$$

Avec X_0 et X_1 valeurs en entrées appelées les graines et M est le module.

La qualité du générateur dépend des graines. De plus, ce générateur est très simple à implémenter et ne consomme que peu de ressources.

a.3 GPAs basés sur les registres à décalage à rétroaction linéaire

Un registre à décalage à rétroaction linéaire (Linear Feedback Shift Register-LFSR), est un automate à états finis permettant de produire une suite infinie satisfaisant une relation de récurrence linéaire. Mathématiquement, un LFSR binaire est un triplé T=(F_2, L, (c_1,c_2,...,c_L) où F_2 est le corps fini à deux éléments (0 et 1). L est le nombre de cellules et les coefficients (c_1,c_2,...,c_L) sont des éléments de F_2. Les L bits contenus dans le registre forment l'état interne du LFSR. Ces L cellules sont initialisées par L bits (s_0,s_1,...,s_{L-1}).

Ce registre à décalage est contrôlé par une horloge externe. Au cours de chaque unité de temps, chaque bit est décalé d'une cellule vers la droite. Le contenu de la cellule la plus à droite, S_t, sort du registre, alors que la cellule la plus à gauche reçoit le bit de rétroaction, S_{t+L}. La valeur de ce dernier est obtenue par une combinaison linéaire des valeurs des autres cellules, dont les coefficients sont des éléments fixés qui valent 0 ou 1 et appelés coefficients de rétroaction du LFSR (eq 1.8)

$$s_{t+L} = \sum_{i=1}^{L} c_i \cdot s_{t+L-i} \tag{1.8}$$

Où la somme est une somme modulo 2.

Les LFSR sont des dispositifs extrêmement rapides et d'implémentation peu coûteuse pour engendrer des suites ayant de bonnes qualités statistiques, notamment une période élevée. Néanmoins, ce générateur seul n'est pas trop sécurisé, il est typiquement utilisé comme un bloc de base pour des générateurs plus compliqués. Par conséquent, des modèles avancés de générateurs peuvent être basés sur une combinaison non linéaire d'un ou plusieurs LFSR. Une description détaillée des LFSR peut être trouvée dans [Alfred J Menezes et Vanstone, 1996b].

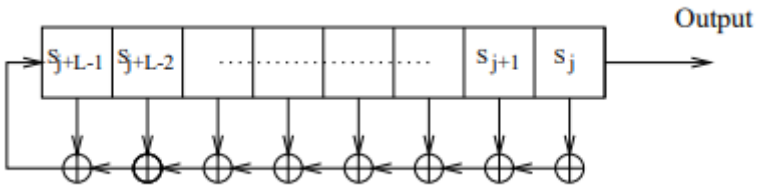


FIGURE 1.9 – Registre à décalage à rétronction linéaire (LFSR)

b. Exigences des GPAS

Cette section fournit une description des exigences de générateurs pseudo-aléatoires. On peut distinguer entre les exigences qualitatives et quantitatives pour les données aléatoires. La première englobe l'imprédictibilité et les propriétés statistiques des données

aléatoires générées. Tandis que la deuxième exigence traite la mesure de l'aléatoire et couvre également les performances des générateurs cryptographiques utilisées.

b.1. Exigences Qualitatives

Une analyse qualitative rigoureuse du caractère aléatoire et de l'imprédictibilité est probablement le problème le plus difficile pour de nombreux chercheurs dans ce domaine. Les données aléatoires générées directement à partir d'une source de hasard contiennent souvent des défauts statistiques ou des corrélations entrainant la prévisibilité des parties de données aléatoires. La première étape vers l'imprédictibilité consiste à assurer (au moins jusqu'à un certain niveau) des bonnes propriétés statistiques des données aléatoires générées. Cela peut être partiellement résolu en utilisant des tests statistiques permettent (manuellement) de détecter certains défauts de conception d'un générateur ou (automatiquement) éviter de briser ou influencer le générateur pendant son cycle de vie. Il existe plusieurs ensembles de tests statistiques ou des batteries fréquemment utilisés (par exemple DIEHARD et NIST) pour la vérification manuelle de la qualité des données aléatoires en détectant les écarts par rapport au caractère aléatoire réel (pour plus de détails voir les manuels [Marsaglia, 1998, Rukhin *et al.*, 2001]. Les résultats doivent être toujours interprétés avec soin et prudence pour éviter de mauvaises conclusions à propos d'une source ou d'un générateur de hasard spécifique.

b.2 Exigences Quantitatives

Une comparaison précise de plusieurs sources d'aléatoire nécessite également une certaine mesure de l'aléatoire La mesure de base est dans la théorie de l'information souvent appelée incertitude ou entropie de Shannon [Smart, 2016d]. La formule 1.9 de Shannon calcule l'entropie selon toutes les probabilités observées de valeurs dans la distribution de probabilité.

$$E_h = -\sum_{j=0}^{K^h} P_{hj} \log_2(P_{hj}) \tag{1.9}$$

L'entropie atteint sa valeur maximale $E_h = h$ lorsque les probabilités de toutes les sous-séquences possibles de longueur h sont égales à $1/k^h$.

1.1.1.4 Code d'authentification de message(MAC)

Un code d'authentification de message (MAC) est un algorithme à clé secrète permettant à la fois d'assurer l'authentification et l'intégrité (l'authenticité) du message et non pas la confidentialité. Il englobe deux algorithmes : un algorithme de génération ($MACG_k$) et un algorithme de vérification ($MACV_k$). L'algorithme $MACG_k$ prend un message M de taille arbitraire et une clé secrète k et génère une empreinte (appelée également Tag), cette dernière est jointe au message M (i.e. (M, $MACG_k$(M))). En outre,

l'algorithme ($MACV_k$) sert à vérifier que la paire (M. $MACG_k$(M)) reçue est bien égale à celle régénérée par le récepteur [35] et n'a pas été modifié. Particulièrement, la clé k fournit à ses possesseurs seuls de s'authentifier en générant le tag. La FIGURE 1.10 illustre cette configuration.

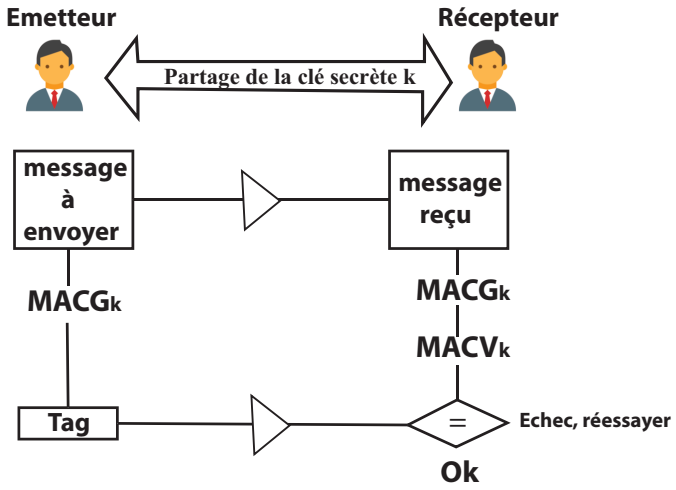


FIGURE 1.10 – Schéma d'un code d'authentification de message

En général, il existe deux façons pour concevoir des MACs, l'une est basé sur un algorithme de chiffrement symétrique [Smart, 2016b] en mode CBC ou CFB, le tag est alors le dernier bloc chiffré obtenu. En outre, la seconde construction est celle basé sur une fonction de hachage à clé, intégrant une clé K lors de calcul du tag. En effet, la sécurité de MAC repose respectivement sur la résistance aux attaques cryptographiques dédiées aux fonctions de hachages et aux algorithmes de chiffrement symétrique (voir section 1.1.1.1 et 1.1.2).

Étant donné une fonction MAC h_k paramétrée par une clé secrète k et une entrée x de longueur arbitraire, un algorithme MAC demande la vérification des propriétés suivantes :

- Facilité du calcul : h(z) doit être facile à calculer.
- Compression : la fonction h_k prend en entrée x et produit une sortie h(a) de longueur fixe.
- Résistance au calcul : Étant donné un ensemble de paires $(x_i, h_k(x_i))$, il est difficile de calculer une paire $(x, h_k(x))$ pour chaque $x \neq x^{'}$.

Si la 3^e propriété n'est pas validée, un algorithme MAC est soumis à une attaque de falsification appelée MAC forgery attack en anglais

a. MAC Forgery Attack

L'attaque de falsification contre les MACs est l'attaque la plus connue sur un algorithme MAC, où une entité malveillante peut fournir au moins une paire de (message. Tag) valide sans tenir compte de clé secrète. Étant donné un message noté x tel que $MACG_k(x) = y$. Il existe deux modèles d'attaques de falsification [Menezes *et al.*, 1996c] :

- L'attaque de falsification existentielle où un adversaire est capable de produire une nouvelle paire (message, Tag) valide pour un message x_i, (sans tenir compte au contrôle sur la valeur de ce texte). En particulier, l'attaquant choisit une clé k aléatoirement puis, il calcule le tag ($MACG_k(x_i)$) pour chaque x_i dans un temps q. L'attaque est réussie si et seulement si x $\notin \{x_1, \cdots, x_q\}$ et $MACG_k(x) = y$.

- L'attaque de falsification sélective où un adversaire est capable de produire une nouvelle paire (message,Tag) pour un message de son choix(ou peut-être partiellement sous son contrôle). En particulier, l'attaquant choisit une clé k et aussi un message a, aléatoirement puis il calcule $MACG_k(x_i)$ pour $x \neq x_i$. L'attaque est réussie si $MACG_k(x_i) = y$.

Un MAC doit résister aussi à l'attaque de key recovery attack qui est décrite dans la section 1.1.1.5. Généralement, pour réussir une attaque de récupération de clé 'key recovery attack' sur un algorithme Mac. Il faut un nombre arbitraire de paires (message, Tag) forgées afin de construire la clé de manière triviale.

a.1 MAC basé sur un chiffrement par bloc en mode CBC (CBC-MAC)

Le CBC-MAC consiste à utiliser une seule clé secrète k dans chaque itération du chiffrement noté E_k. Dans la FIGURE 1.11, le premier bloc M_1 est chiffré en utilisant la clé k et le résultat du chiffrement est une entrée à la fonction E_k après avoir appliqué un opérateur XOR à la paire $(E_k(M_1), M_2)$. Ainsi de suite, l'opération est itérée jusqu'à obtenir le dernier chiffré $(E_k(M_{n-1}), M_n)$ représentant le tag.

La fonction de compression pour le CBC-MAC peut s'exprimer comme suit :

$$Tag_{i+1} = E_k(M_i \oplus E_k(M_{i-1})) \tag{1.10}$$

a.2 MAC basé sur les fonctions de hachage

Une fonction de hachage cryptographique H est une fonction qui prend des chaines de bits de longueur arbitraire en entrée et produit en sortie une chaîne de bits de longueur fixe [Smart, 2016b]. Le principe de concevoir un MAC sécurisé à partir d'une fonction de hachage est alors d'intégrer une clé K dans de calcul du tag. En particulier, en utilisant cette approche, nous pouvons définir les fonctions de hachages à clé comme une famille de fonctions. Il existe plusieurs versions des MACs basés sur les fonctions de hachages nous citons par exemple, l'algorithme le plus connu est le HMAC (keyed-hash MAC) qui

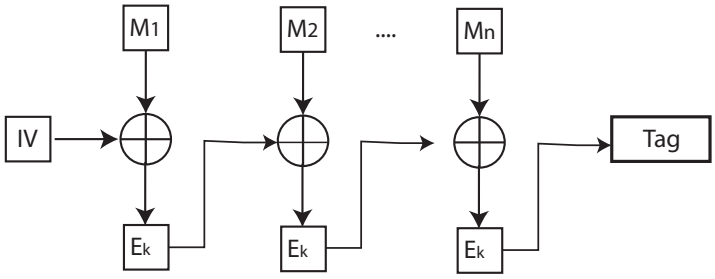


FIGURE 1.11 – MAC basé sur un chiffrement par bloc en mode CBC (CBC-MAC

consiste à intégrer n'importe quelle fonction de hachage noté H dans une fonction HMAC. Cette dernière est définie par l'expression suivante :

$$HMAC_k(M) = H[(K \oplus opad) \| H[(K \oplus ipad) \| M]] \tag{1.11}$$

avec :

- M : le message d'entrée de l'algorithme.
- K : la clé permettant le calcul du MAC.
- ipad 0x36 (répété b/8 fois) ou b est la taille du bloc
- opad 0x5C (répété b/8 fois)

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

1.1.1.5 Attaques Cryptographiques

L'objectif d'un chiffrement est d'assurer la confidentialité des données. Le but d'un attaquant est de récupérer le texte en clair qui correspond au texte chiffré. Un chiffrement peut être cassé si un adversaire est capable de trouver une clé ou une partie de texte en clair à partir du texte chiffré. Les exigences de sécurité des textes clairs sont fournies par trois grandes classes qui sont [Sakiyama *et al.*, 2016] :

- **'Plaintext recovery resistance'** : consiste à résister à l'attaque qui récupère efficacement la valeur de texte en clair correspondante à partir de K et C ; telle que $E_K(P) = C$, où K est une clé arbitraire et C un texte chiffré arbitraire.
- **'Key recovery resistance'** : consiste à résister à l'attaque qui récupère efficacement la valeur de K.
- **'Indistinguishability'** : la transformation E_k doit être indiscernable d'une permutation aléatoire pour K, c'est-à-dire qu'un adversaire ne pouvait pas calculer une paire valide(clair,chiffré) sans connaitre la clé K.

Dans un système de chiffrement, si 'Key recovery resistance' (la résistance de récupération de clé) est cassée, les deux notions sont cassées automatiquement. Par conséquent, 'Key recovery resistance' (la résistance de récupération de clé) est la notion de sécurité la plus forte parmi les trois ci-dessus. En outre, les mesures de la sécurité d'un chiffrement doit résister aux modèles d'attaques pouvant être utilisés par un attaquant. Ces modèles d'attaque sont essentiellement catégorisés par l'information que les attaquants peuvent atteindre, on citant les modèles suivants :

— **Attaque sur texte chiffré seul 'Ciphertext only attack'** : L'attaquant tente d'obtenir le texte en clair à partir d'un ensemble d'exemplaires de textes chiffrés (voir la FIGURE 1.12).

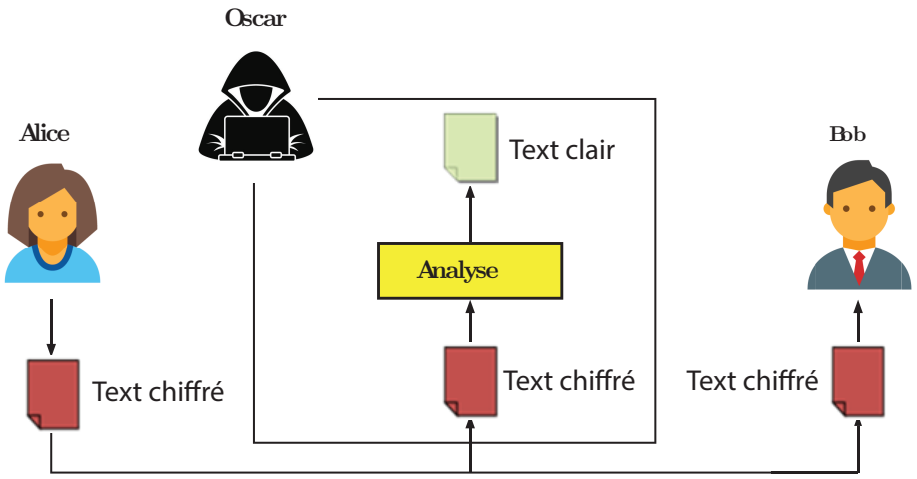


FIGURE 1.12 – Attaque texte chiffré seul (COA)

— **Attaque à texte clair connu 'Known plaintext attack'** : l'attaquant peut détecter des combinaisons de texte clair et le texte chiffré correspondant. Cette attaque vise à simuler l'attaquant qui écoute la communication entre deux interlocuteurs. L'attaquant n'a aucun contrôle sur le texte en clair à chiffrer. La cryptanalyse linéaire fait partie de cette catégorie (voir la FIGURE 1.13).

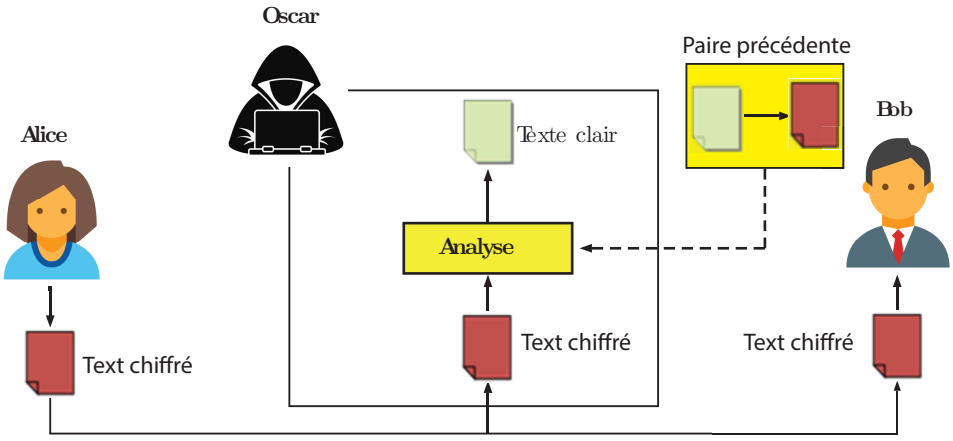


FIGURE 1.13 – Attaque à texte clair connu (KPA)

— **Attaque à texte clair choisi 'Chosen plaintext attack'** : l'attaquant peut obtenir le texte chiffré correspondant pour tout choix de texte en clair. Cette attaque est plus puissante que 'known plaintext attack'. L'attaquant vise à récupérer tout ou partie de la clé. La cryptanalyse différentielle est un exemple d'attaque à texte clair choisi (voir la FIGURE 1.14).

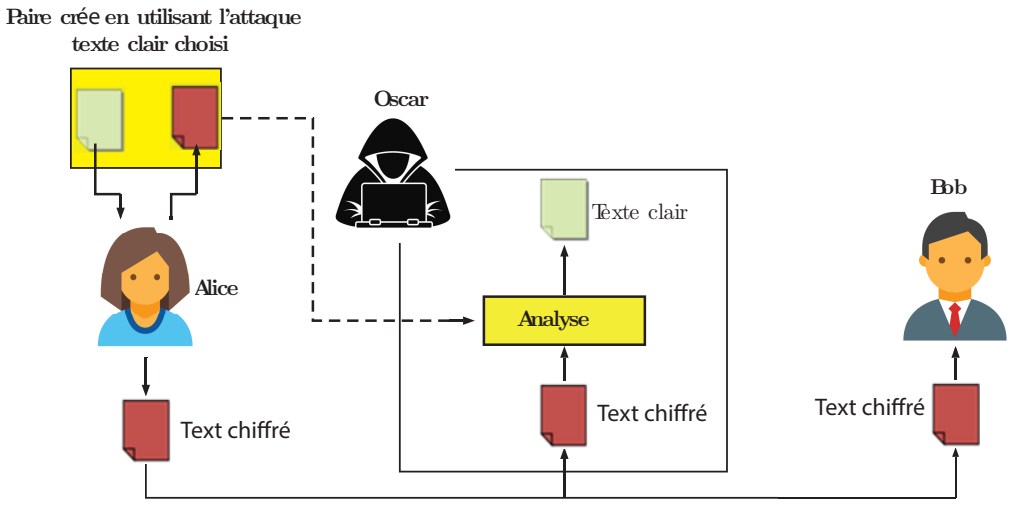


FIGURE 1.14 – Attaque à texte clair choisi (CPA)

— **Adaptive chosen plaintext attack** : l'attaquant commence par une attaque de chosen plaintext attack' dans le premier tour puis il « adapte » d'autres tours de déchiffrement en se basant sur le premier.

— **Attaque à texte chiffré choisi 'Chosen ciphertext attack'** : est un modèle similaire à celui de chosen plaintext attack. L'attaquant peut obtenir le texte en clair correspondant pour tout choix de texte chiffré (voir la FIGURE 1.15).

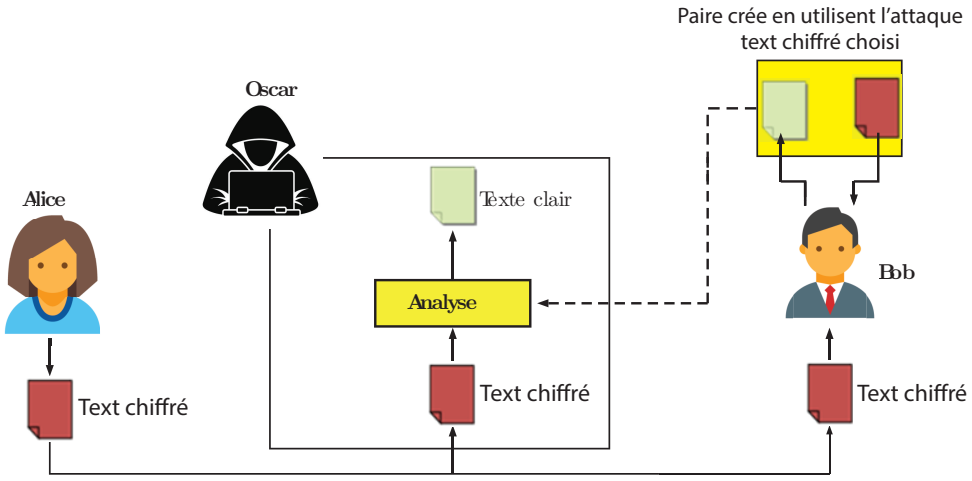


FIGURE 1.15 – Attaque à texte chiffré choisi (CCA)

— **Adaptive chosen ciphertext attack** : est un modèle similaire à celui de 'Adaptive chosen plaintext attack'. L'attaquant a la possibilité de demander les textes en clair des textes chiffrés sélectionnés après le choix des chiffrés à déchiffrer par le système.

Les paramètres d'un chiffrement doivent souvent satisfaire les exigences de sécurité. Par conséquent, l'indistinguabilité doit être assurée contre l'attaque en texte clair et l'attaque de texte chiffré [Sakiyama *et al.*, 2016].

Il existe d'autres versions d'attaques cryptographiques nous citons :

- **Attaque de recherche exhaustive de la clé (ou attaque en force brute)** : L'attaque en force brute repose sur la génération d'un espace de clés possibles. Compte tenu du temps suffisant, le texte en clair sera récupéré. C'est une attaque efficace contre tous les systèmes de chiffrement basé sur l'utilisation des clés, à l'exception du 'one-time pad', particulière ment symétriques, car pour les algorithmes à clé publique. La difficulté n'est pas de trouver la bonne clé par attaque en force brute, mais de dériver la clé privée à partir de la clé publique [Sakiyama *et al.*, 2016].

- **Cryptanalyse différentielle** : La cryptanalyse différentielle est une sorte d'attaque à texte clair choisi 'Adaptively chosen-plaintext attack' qui peut être appliquée aux algorithmes de chiffrement itératif par blocs, mais également aux fonctions de hachage. Elle cherche à trouver la «différence» entre les clairs et leurs chiffrés correspondants à chaque itération de l'algorithme de chiffrement. L'attaquant utilise alors une analyse statistique [Smart, 2016a] pour déterminer une partie de la clé ou bien de réduire suffisamment le nombre de clés possibles pour pouvoir mener une attaque en force brute rapide.

- **Cryptanalyse linéaire** : La cryptanalyse linéaire est une sorte d'attaque de texte

clair connu où l'attaquant trouve des paires (texte en clair, texte chiffré) créés avec la même clé. Les paires sont étudiées pour obtenir totalement la clé utilisée pour les créer [Smart, 2016a].

- **Paradoxe des anniversaires** : L'attaque de paradoxe des anniversaires est basé sur le fait que dans un groupe de 23 personnes ou plus choisies aléatoirement, les chances que deux personnes de ce groupe partagent le même anniversaire sont supérieures à 50% [Smart, 2016e]. Le but de cette attaque est de modifier les messages transmis entre deux entités ou plus.

- **Key Clustering** : L'objectif principal d'un système de chiffrement cryptographique est qu'une seule clé peut dériver le texte en clair à partir du texte chiffré. Key Clustering permet de produire le même chiffré en utilisant deux clés différentes s'appliquent au même clair [Alfred J Menezes et Vanstone, 1996a].

1.1.2 Fonctions de hachage

Une fonction de hachage cryptographique est appelée "fonction de hachage à sens unique" car il n'y a aucun moyen d'inverser le chiffrement. Une fonction de hachage cryptographique H est une fonction qui prend des chaînes de bits de longueur arbitraire en entrée et produit en sortie une chaîne de bits de longueur fixe (appelée "empreinte" ou digest en anglais ou simplement "haché") [Smart, 2016b]. Les fonctions de hachages sont principalement utilisées pour garantir l'intégrité des données si l'empreinte d'un clair change, alors le clair a changé aussi. Les fonctions de hachages les plus anciennes y compris Secure Hash Algorithm 1(SHA-1), qui donne une empreinte de taille 160 bit, et Message Digest 5 (MD5), qui donne une empreinte de taille 128 bit. Les faiblesses ont été trouvées à la fois dans MD5 et SHA-1 ; Des alternatives plus récentes telles que les fonctions SHA-2 et SHA-3 sont recommandées.

Les fonctions de hachage cryptographiques sont utilisées dans les conceptions de protocole en sécurité de l'informatique, et en particulier dans le cas des MACs. Ils sont également utilisés dans les signatures électroniques, et d'autres formes d'authentification.

Les fonctions de hachage cryptographiques sont utilisées dans les conceptions de protocole en sécurité de l'informatique, et en particulier dans le cas des MACs. Ils sont également utilisés dans les signatures électroniques, et d'autres formes d'authentification.

Une fonction de hachage h : $(0,1)^* \longrightarrow (0,1)^n$ est dite sécurisée si elle satisfait les trois propriétés fondamentales de sécurité [Smart, 2016b] suivantes :

- **Résistance en pré-image** : Cette propriété explique le caractère à sens unique de la fonction. Formellement, étant donné un haché y, il est difficile de trouver un message x tel que $h(x) = y$

- **Résistance en seconde pré-image** : Étant donné un message x, il est difficile de trouver un autre message x' différent de x tel que $h(x') = h(x)$.

- **Résistance en collision** : il est difficile de trouver deux messages différents x et x' tels que $h(x) = h(x')$.

Dans ces points, le mot "difficile" explique qu'il n'est possible de résoudre ces problèmes que par des méthodes génériques. Étant donné une empreinte y de longueur n, pour trouver une pré-image de y, une méthode générique consiste à tirer de manière uniformément aléatoire un jusqu'à obtenir $h(x) = y$. En suivant le même raisonnement, on tire x' jusqu'à obtenir $h(x') = h(x)$ pour la recherche d'une seconde pré-image [Smart, 2016b]. La complexité de cette attaque est d'environ 2^n opérations pour trouver une pré-image ou une seconde pré-image. Finalement, pour la recherche de collision, on construit une table contenant toutes les images h(x) des messages x tirés d'une manière uniformément aléatoire. On s'arrête lorsqu'on a trouvé un x tel que h(x) se trouve déjà dans la table. En utilisant l'attaque de paradoxe des anniversaires (voir la section 1.1.1.5), pour trouver une collision dans un ensemble de taille 2^n est estimée à $2^{n/2}$. En résumé, une fonction de hachage est dite bonne s'il n'existe aucune autre méthode efficace que ces méthodes génériques ; soit 2^n pour la pré-image et la seconde pré-image, et $2^{n/2}$ pour les collisions [Smart, 2016b].

Dans la suite, on donne les deux constructions itératives d'une fonction de hachage.

1.1.2.1 Construction de Merkle Damgard

La construction de Merkle-Damgard est une construction itérative qui comprend une fonction de compression [Smart, 2016b]. Cette dernière traite une entrée M découpée en blocs de taille fixe $m_i, \ldots, m_k$ (où k est la taille du bloc) les uns après les autres (voir la FIGURE 1.16).

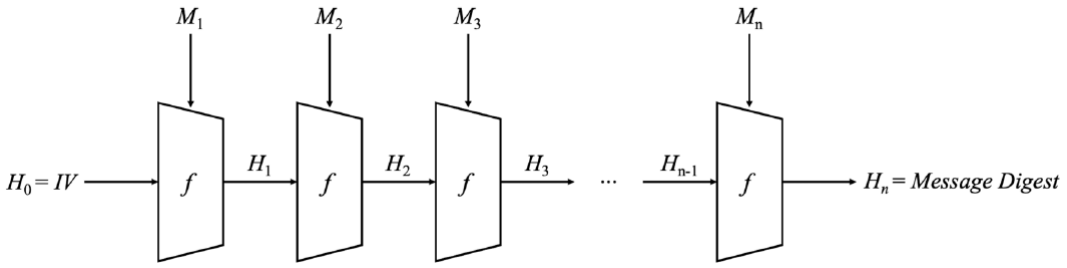


FIGURE 1.16 – La construction de Merkle Damgard

Dans une première étape. On applique la méthode de zero-padding au message m pour que sa longueur soit un multiple de la taille du bloc. Cela permet, par exemple, d'éviter des attaques en collision avec des messages de tailles différentes, des attaques avec des messages très longs, ou encore la recherche de points fixes dans la fonction de compression h. Ensuite, on initialise le premier haché par une valeur fixée de F_2^n appelée

vecteur d'initialisation (Initial Value en anglais). Puis, pour i allant de 1 jusqu'à k, on définit successivement $H_i = h(H_{i-1}, m_i)$. En fin, on retourne le haché $h(M) = H_k$.

1.1.2.2 Modèle éponge

Une autre construction introduite et proposée par Bertoni et al [Bertoni *et al.*, 2008], il s'agit d'une nouvelle méthode itérative utilisant une permutation f capable de produire des sorties de taille arbitraire. Afin de hacher un message x après avoir appliqué une méthode de zéro padding en le découpant en blocs $m_i, \ldots, m_k$ de taille fixe r, appelée le taux de l'éponge, la permutation P se déroule en deux étapes successives :

- **Absorption** : Dans cette étape, les blocs du message paddés de r bits sont combinées à l'aide d'un OU exclusif (XOR) avec l'état interne de l'éponge les uns après les autres y compris une application de la transformation f sur l'état interne.
- **Essorage** : pendant cette étape, les r bits de l'état interne sont extraits en sortie y compris une application de la transformation f sur l'état interne, jusqu'à obtenir n bits de sortie. Les derniers c bits de l'état ne sont jamais immédiatement transformés par les blocs d'entrée et ne sont jamais renvoyés pendant l'essorage.

La FIGURE 1.17 résume ces deux procédures de la construction éponge. Le niveau de sécurité d'une construction éponge repose sur les paramètres utilisés. Elle est notamment la base de plusieurs fonctions de hachage récemment introduites. Parmi elles, on trouve le nouveau standard Keccak [Bertoni *et al.*, 2013], gagnant la compétition SHA-3. On cite également d'autres fonctions de hachage à bas coût comme Quark [Aumasson *et al.*, 2013], Photon [Guo *et al.*, 2011] et Spongent [Bogdanov *et al.*, 2011].

Il existe également d'autres méthodes pour construire des fonctions de hachages. Nous citons la construction de Davies-Meyer dont chaque bloc de message m, représente la clé du chiffrement, tandis que chaque valeur de chainage h_{i-1} devient une nouvelle entrée à l'algorithme de chiffrement. Cette valeur est xoré au résultat du chiffrement :

$$h_i = E_{m_i}(h_{i-1}) \oplus h_{i-1} \tag{1.12}$$

Cette méthode est utilisée pour construire plusieurs fonctions de hachage connues, comme les fonctions MD4, MD5, SHA-0, SHA-1 ainsi que la famille SHA-2 [Menezes *et al.*, 1996c] Comme nous avons mentionné dans la section 1.1.1.4 que les fonctions de hachages sont utilisées dans la construction des Macs, elles peuvent également intervenir dans un chiffrement à clé publique en particulier la signature électronique.

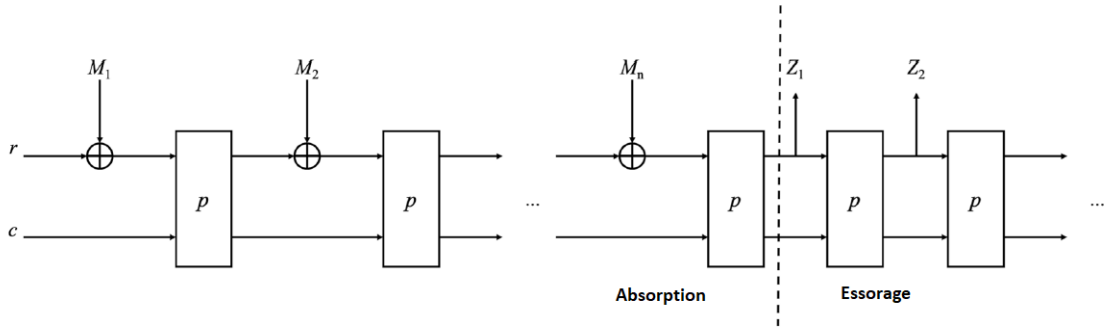


FIGURE 1.17 – La construction éponge

1.1.3 Cryptographie à clé publique

Les cryptosystèmes asymétrique (illustré par la FIGURE 1.1) ou à clé publique différente entre autres dans la gestion de la clé. Ils utilisent deux clés différentes, l'une pour chiffrer appelée la clé publique et l'autre pour déchiffrer appelée la clé privée. Ainsi la sécurité repose sur le fait que, même connaissant la clé de chiffrement, un attaquant n'est pas capable de trouver la clé de déchiffrement. L'exécution d'un cryptosystème à clé publique se fait en 4 étapes indispensables [Smart, 2016f] :

1. Chaque entité génère la paire clé publique-clé privée.
2. Chaque entité divulgue la clé publique dans un registre ou fichier publique et garde la clé privée.
3. Une entité A souhaitant envoyer un message secret à une autre entité B. L'entité A chiffre ce message en utilisant la clé publique de B et puis transmis le message obtenu à B.
4. Quand B reçoit le message provenant de A. elle le déchiffre en utilisant sa clé privée.

La cryptographie à clé publique est utilisée dans différentes applications comme le chiffre ment, la signature numérique et le partage de clés secrètes. Les algorithmes ECC (Elliptic Curve Cryptosystems) [Koblitz, 1987], RSA (issu du nom des inventeurs Rivest, Shamir et Adleman) [Smart, 2016g]. Diffie Hellman [Menezes *et al.*, 1996b] et ElGamal [Smart, 2016h] sont des exemples des cryptographie asymétrique.

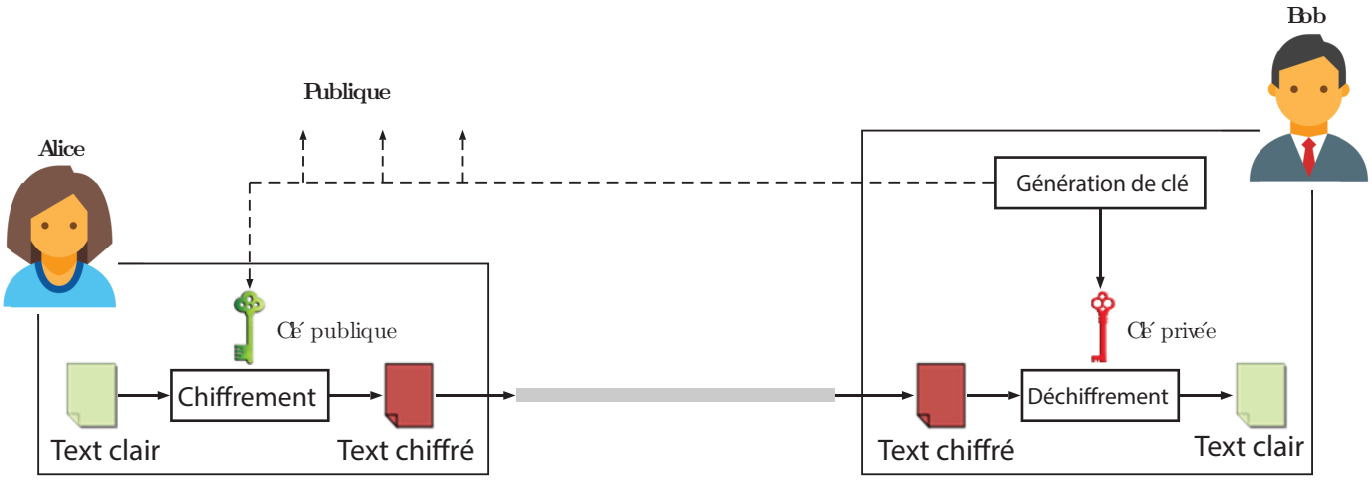


FIGURE 1.18 – Chiffrement asymétrique

1.2 Théorie des graphes

1.2.1 Historique

Le problème des sept ponts de Königsberg est présenté par Leonhard Euler à l'Académie de Saint-Pétersbourg en 1735 puis publié en 1741.

Le problème consistait à débuter une promenade à partir d'un point donné qui fasse revenir à ce point en passant une fois et une seule par chacun des sept ponts de la ville de Königsberg (voir la FIGURE 1.19). Le graphe représentant ce problème est illustré par la FIGURE 1.20.

Un problème similaire consiste à passer par chaque sommet exactement une fois, et fut d'abord résolu avec le cas particulier d'un cavalier devant visiter chaque case d'un échiquier par le théoricien d'échec arabe Al-Adli dans son ouvrage Kitab ash-shatranj paru vers 840 et perdu depuis.

On accorde donc à Euler l'origine de la théorie des graphes parce qu'il fut le premier à proposer un traitement mathématique de la question.

Les informations ci-dessous illustre l'évolution de la théorie des graphes :

— 1847 Kirchhoff : Théorie des arbres (analyse de circuits électronique).

— 1860 Cayley : Enumération des isomères saturés des hydrocarbures C_nH_{2n+2}.

— À la même époque, énoncé de problèmes importants : Conjecture des quatre couleurs (1879)
(Mobius, De Morgan, Cayley, solution trouvée en 1976).

— Existence de chemins Hamiltoniens (1859).

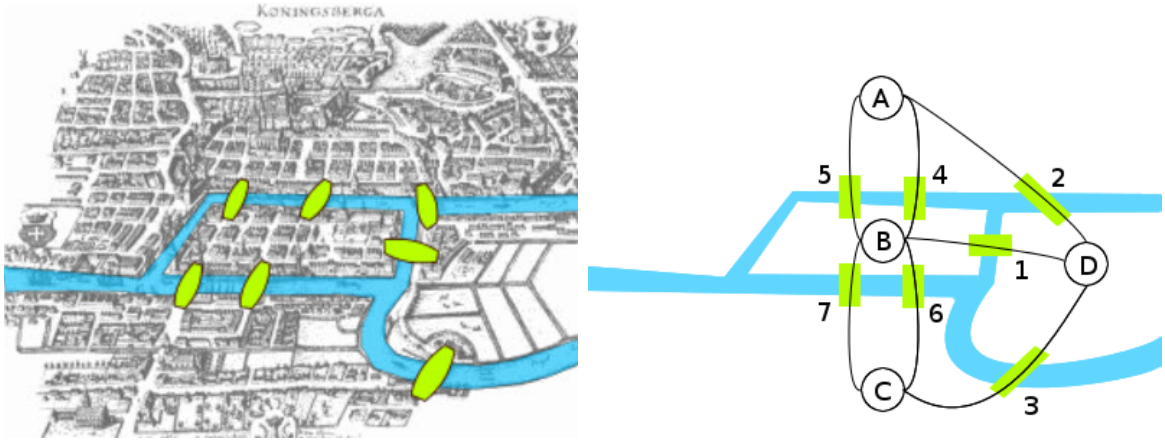


FIGURE 1.19 – Le problème des sept ponts de Königsberg

— 1936 Konig Premier ouvrage sur les graphes.

— 1946 Kuhn, Ford, Fulkerson, Roy et Berge.

— 1947 Les premiers travaux de Harold Wiener.

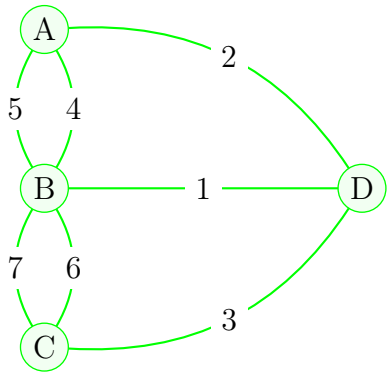


FIGURE 1.20 – Graphe représentant le problème des sept ponts de Königsberg

1.2.2 Terminologie et définitions

— **Graphe :** G=(V,E) est défini par l'ensemble fini $V = \{v_1, v_2..., v_n\}$, dont les éléments sont appelés sommets(en anglais : vertex ou vertices pour le pluriel), et un ensemble fini, $E = \{e_1, e_2..., e_n\}$ dont les éléments sont appelés arêtes ou arcs (en anglais : edge. Si V est fini, on parlera de graphe fini [West *et al.*, 2001, Berge et Berge, 1967, Labelle, 1981, Rigo, 2009, Diudea *et al.*, 2001, KBalakrishnan, 1995]
Exemple : le graphe (FIGURE 1.21) : $V = \{A, B, C, D, E\}$ et $E = \{1, 2, 3, 4, 5, 6, 7\}$.

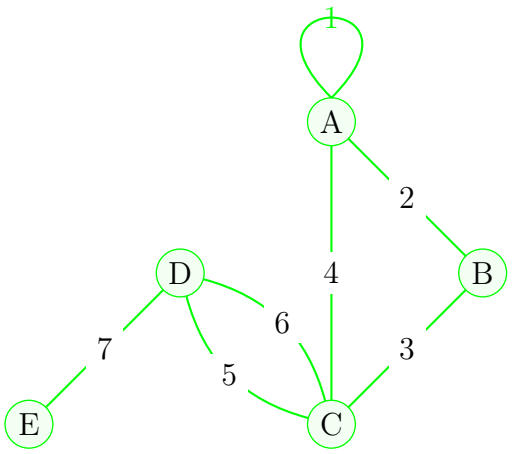


FIGURE 1.21 – Expemple d'un graphe G

— **Arête :** Une arête est le trait qui relie deux sommets dans un graphe. Nous appelons ces deux sommets les extrémités de l'arête.
— **Face :** Une face d'un graphe G notée F_G est une région du plan limitée par des arêtes et qui ne contient ni sommets ni arêtes dans son intérieur. Exemple :
Le graphe G (FIGURE 1.21) a quatre faces $|F_G| = 4$:
- La région du plan limitée par les arêtes (2, 3, 4).
- La région limitée par les arêtes (5, 6).
- La région limitée par la boucle (1).
- Et toute la région restante entourant le graphe G.
— **Sommets Adjacents :** Deux sommets sont adjacents s'ils sont reliés par une arête. On qualifie souvent de voisins deux sommets adjacents.
Exemple : les sommets A et B du graphe (FIGURE 1.21).
— **Un sommet incident :** Un sommet est incident à une arête s'il est situé à une des deux extrémités de cette arête.
Exemple : le sommet B est incident à l'arête 3 (FIGURE 1.21).
— **Arête incidente :** Une arête est incidente à un sommet si elle touche ce sommet.
Exemple : l'arête 4 est incidente au sommet C (FIGURE 1.21).
— **Boucle :** Une boucle est une arête qui relie un sommet à lui-même.
Exemple : l'arête 1 (FIGURE 1.21).
— **Voisinage de v :** Le voisinage de v est l'ensemble de tous ses sommets adjacents.
Exemple : le voisinage du sommet C et l'ensemble $\{A, B, D\}$ (FIGURE 1.21).
— **Ordre de G :** L'ordre de G est le nombre de sommets $n = |V(G)|$ de ce graphe.
Exemple : $n = 5$ (FIGURE 1.21).
— **Taille de G :** La taille de G est le nombre de ses arêtes $m = |E(G)|$.
Exemple : $m = 7$ (FIGURE 1.21).
— **Degré d'un sommet v :** Le degré de v est le nombre d'arêtes incidentes à ce sommet, on le note deg(v).

Exemple : $deg(A) = 3$ (FIGURE 1.21).

— **Feuille (pendant) :** Une feuille est le sommet de degré 1.
Exemple : le sommet D est une feuille, $deg(D) = 1$ (FIGURE 1.21).

— **Degré d'un graphe G :** Soit G un graphe le degré de G est le degré maximum de tous ses sommets.
Exemple : le degré du graphe (FIGURE 1.21) est le degré du sommet C $deg(C) = 4$.

— **Distance entre deux sommets :** La distance entre deux sommets est le nombre minimal d'arêtes entre eux, on la note d(u, v).
Exemple : $d(A, C) = 1$ et $d(A, E) = 3$ (FIGURE 1.21).

— **lemme :poignées de mains** La somme des degrés des sommets d'un graphe est égale à deux fois le nombre d'arêtes.

$$\sum_{i=1}^{n} deg(v_i) = 2|E|$$

— **Diamètre de G :** Le diamètre de G est la plus longue des distances entre deux sommets de ce graphe D(G) = $max_{u,v \in V(G)}(d(u, v))$, on le note Diam(G) ou D(G).
Exemple : $D(G) = d(A, E) = 3$ (FIGURE 1.21).

— **Chaîne :** Une chaine est la suite de sommets reliés par des arêtes. sa longueur est le nombre d'arêtes utilisées (le nombre de sommets utilisés - 1).
Exemple : D, C, B, A, C, D, E est une chaine (FIGURE 1.21).

— **Chaîne élémentaire :** Une chaine élémentaire est la chaîne dans laquelle on ne peut pas visiter le même sommet deux fois.
Exemple : A, B, C, D, E est une chaine élémentaire (FIGURE 1.21)

— **Chaîne simple :** Une chaine simple est la chaîne dans laquelle on ne peut pas visiter la même arête deux fois.

— **Cycle :** Un cycle est la chaîne simple dont ces extrémités coïncident. On ne rencontre pas deux fois le même sommet, sauf celui choisi comme sommet de départ et d'arrivée.
Exemple : A, B, C, A est un cycle (FIGURE 1.21)

— **Corde :** Une corde est l'arête reliant deux sommets non adjacents d'un cycle.

— **chaîne/cycle Hamiltonien :** une chaîne ou un cycle Hamiltonien est la chaîne/cycle dans laquelle chacun des sommets y apparaît exactement une fois.

— **chaîne/cycle eulérien** est la chaîne/cycle dans laquelle chacune de ses arêtes y apparaît exactement une fois.

— **Graphe orienté/non orienté :** Un graphe est dît orienté quand ses arêtes vont d'un sommet à un autre dans un sens précis.Sinon, il est dit non orienté.
Exemple : le graphe G* (FIGURE 1.22) est orientés. le graphe G (FIGURE 1.21) est non orienté.

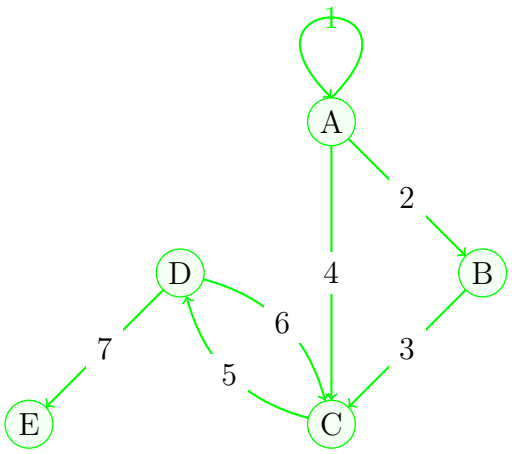


FIGURE 1.22 – Le graphe G orienté :G*

— **Graphe simple :** Un graphe simple est un graphe sans cycle, de plus entre deux sommets distincts il y à au plus une seule arête.
Exemple :

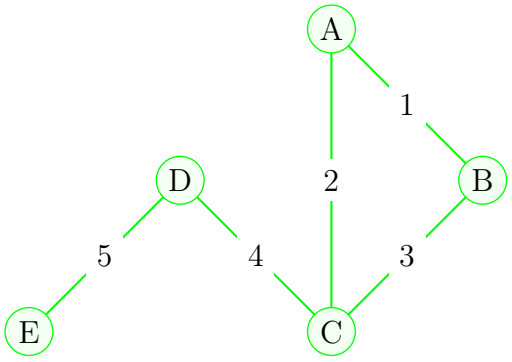


FIGURE 1.23 – Expemple d'un graphe simple : S

— **Graphe planaire :** Un graphe planaire est un graphe que l'on peut dessiner sur une surface plate sans que ses arêtes se croisent [West *et al.*, 2001, Berge et Berge, 1967, Labelle, 1981, Rigo, 2009, Diudea *et al.*, 2001, KBalakrishnan, 1995].
Exemple : les graphes G (FIGURE 1.21), G* (FIGURE 1.22), S (FIGURE 1.23) sont tous des graphes planaires.

— **Graphe connexe :** Un graphe $G = (V, E)$ est dit connexe si quels que soient les sommets u et v de G, il existe une chaîne de u vers v. C'est-à-dire, s'il existe une suite d'arêtes permettant d'atteindre v à partir de u.
Exemple : le graphe G (FIGURE 1.21) est connexe. Le graphe suivant (FIGURE 1.24) est un graphe non connexe.

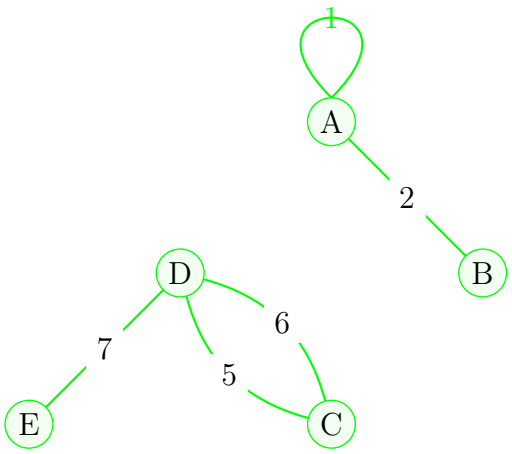


FIGURE 1.24 – Expemple d'un graphe non connexe

— **Graphe complet :** est le graphe dans lequel toutes les paires de sommets sont adjacentes. On note un graphe complet de n sommets K_n.
Exemple : le graphe suivant (FIGURE 1.25) est complet.

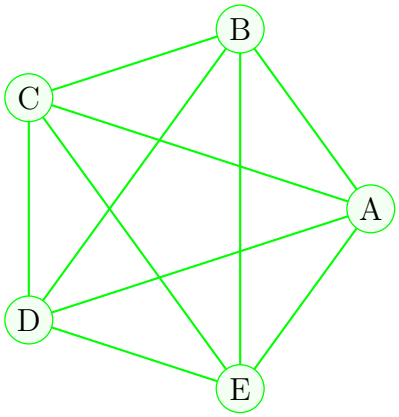


FIGURE 1.25 – Expemple d'un graphe complet : K_5

— **Graphe acyclique :** est un graphe ne contenant aucun cycle.
— **Arbre :** est un graphe non orienté, acyclique et connexe.
Exemple : le graphe suivant (FIGURE 1.26) est un arbre.

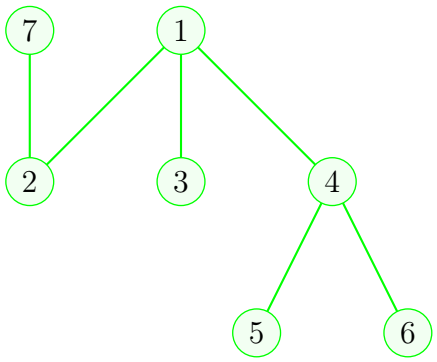


FIGURE 1.26 – Expemple d'un arbre

— **Path : (arbre chainé)** une séquence d'arêtes connectées à une séquence de sommets (FIGURE 1.27). noté P_n, avec $|E| = n - 1$ et $|V| = n$
Exemple :

FIGURE 1.27 – Exemple de path : P_4

— **Sommet complet :** sommet connecté à tous les autres sommets du graphe auquel il appartient.
— **Star : (arbre étoilé)** noté S_n est l'arbre à n sommets de diamètre maximal 2 (FIGURE 1.28), S_n a un sommet complet, les autres sont des feuilles(le seul sommet à qui ils sont connectés est le sommet complet) .
Exemple :

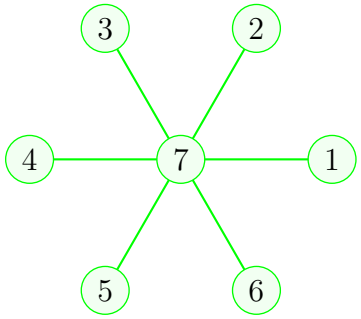


FIGURE 1.28 – Exemple de star : S_7

— **La formule d'Euler :** Soit G un graphe planaire connexe. Alors

$$|V_G| + |F_G| - |E_G| = 2 \tag{1.13}$$

Remarque : Durant notre étude, on ne s'intéressera qu'aux graphes non orientés, simples et planaires.

1.2.3 Domaines d'application

— Quelle est la méthode la plus efficace qui nous permet de relier des câbles afin que tout téléphone soit joignable à partir de tout autre téléphone, et ce avec un cout minimal ?

— Quelle route est la plus courte pour joindre toute ville du pays à partir de la capitale ?

— Comment n personnes peuvent t-elles accomplir n taches avec une utilité totale maximale ?

— Dans une puce, quel est le nombre de couches nécessaire pour que les fils de chaque couche ne se croisent pas ?

— Comment peut-on organiser une saison de ligue sportive afin qu'elle se déroule dans un nombre de semaines minimal ?

— Dans quel ordre un voyageur devrait visiter des villes pour que son voyage s'effectue en un temps minimal ?

— Peut-on colorier toutes les régions d'une carte d'une telle manière a ce que toutes régions voisines aient des couleurs différentes ?

Tous ces problèmes pratiques et beaucoup d'autres peuvent être résolu par la théorie des graphes. Ainsi, les domaines d'application de la théorie des graphes sont aussi vastes que diversifiés. En effet, la théorie des graphes peut résoudre de simples problèmes du quotidien, faisant autant pour des problèmes de chimie moléculaire ou de réseaux-télécommunication.

1.2.4 Modes de représentation d'un graphe

L'essor de la théorie des graphes est essentiellement dû à l'avènement de puissants calculateurs. Il est donc légitime de s'intéresser à la manière de représenter des graphes au sein d'un ordinateur. Plusieurs modes de représentation peuvent être envisagés selon la nature des traitements qu'on souhaite appliquer au graphe considéré. Dans cette section nous allons traiter quelqu'uns d'entre eux et pour en savoir plus, voici quelques références [Xu, 2003, Rigo, 2009, Diestel, 2000, KBalakrishnan, 1995].

1.2.4.1 Matrice d'adjacence

Définition 1.2.

Considérons un graphe G = (V,E) d'ordre n. Sa matrice d'adjacence notée $A = (a_{ij})$ est une matrice de dimension $n \times n$, tel que

$$\begin{cases} a_{ij} = 1 & \text{si (i,j)} \in E \\ a_{ij} = 0 & \text{sinon} \end{cases} \tag{1.14}$$

Remarque : Cette matrice a plusieurs caractéristiques :

- Elle est carrée.
- Il n'y a que des zéros sur la diagonale. Un 1 sur la diagonale indiquerait une boucle.
- Elle est symétrique : $a_{ij} = aji$. On peut dire que la diagonale est son axe de symétrie.
- Une fois qu'on fixe le nombre de sommets, il existe une matrice d'adjacences unique pour chaque graphe. Celle-ci n'est la matrice d'adjacence d'aucun autre graphe.
- Un graphe orienté a une matrice d'adjacence quelconque, alors qu'un graphe non orienté possède une matrice d'adjacence toujours symétrique.
- Ce mode de représentation engendre des matrices creuses. Cependant la recherche de chemin ou de chaîne s'effectue aisément avec une telle représentation.
- Le graphe complet non orienté sans boucle d'ordre n a une matrice d'adjacence A particulière :

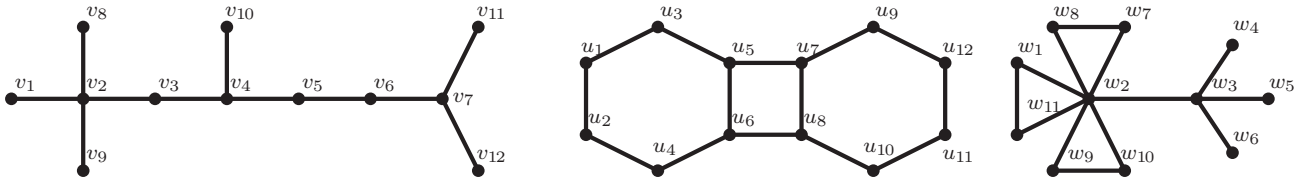
$$\begin{cases} a_{ij} = 1 & \text{pour i} \neq j \\ a_{ij} = 0 & i = j \end{cases} \tag{1.15}$$

Exemple : La matrice d'adjacence du graphe G_3 de la FIGURE 1.31 est :

Théorème 1.1.

Soient un graphe orienté $G = (V, E)$ où $V = \{ v_1, v_2, \ldots, v_n \}$ et $A = A(G)$ sa matrice d'adjacence. Si $A^k = \text{A} \times \text{A} \times \ldots \times \text{A} = a_{ij}^k$, k > 1, alors a_{ij}^k est le nombre de chemins de longueurs k du sommet v_i au sommet v_j dans G [Diestel, 2000].

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



G_1 G_2 G_3

FIGURE 1.29 – Exemple de graphes non orientés, connexes, planaires et finis

Démonstration : Par récurrence. Pour k = 1, $a_{ij}^1 = a_{ij}$ est bien le nombre de chemins de longueur 1 de v_i à v_j . Supposons l'énoncé vrai pour k-1. On a $A^k = A^{k-1} \times A$, on a alors

$$A_{ij}^k = \sum_{l=1}^{n} a_{il}^{k-1} \cdot a_{lj} \tag{1.16}$$

et a_{il}^{k-1} ni que le nombre de chemins de longueur k-1 de v_i à v_l et

$$\begin{cases} a_{ij} = 1 & \text{si (i,j)} \in E \\ a_{ij} = 0 & \text{sinon} \end{cases} \tag{1.17}$$

On a donc $A_{ij}^k = \sum_{l=1}^{n}$(nombre de chemins de longueur k de v_i à v_j et se terminant par (v_l, v_j)) = (nombre de chemins de longueur k de v_i à v_j).

Remarque : Lorsque $G = (V, E)$ est un graphe simple non orienté où $V = \{ v_1, v_2, \ldots, v_n \}$, la matrice d'adjacence de G est la matrice d'adjacence du graphe orienté G' associé à G. Cette matrice A est, bien sûr, symétrique. De plus les entrées de A^k comptent le nombre de chaînes de longueur k dans G.

1.2.4.2 Matrice de distances

Définition 1.3.

Considérons un graphe G = (V,E) d'ordre n. Sa matrice de distance notée D = (d_{ij}) est une matrice de dimension $n \times n$, tel que

$$d_{ij} = d(v_i, v_j) \tag{1.18}$$

Où d(v_i,v_j) est la distance entre les deux sommets v_i et v_j.

Exemple : La matrice de distance du graphe G_3 de la FIGURE 1.31 est :

$$D = \begin{bmatrix} 0 & 1 & 2 & 3 & 3 & 3 & 2 & 2 & 2 & 2 & 1 \\ 1 & 0 & 1 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 \\ 3 & 2 & 1 & 0 & 2 & 2 & 3 & 3 & 3 & 3 & 3 \\ 3 & 2 & 1 & 2 & 0 & 2 & 3 & 3 & 3 & 3 & 3 \\ 3 & 2 & 1 & 2 & 2 & 0 & 3 & 3 & 3 & 3 & 3 \\ 2 & 1 & 2 & 3 & 3 & 3 & 0 & 1 & 2 & 2 & 2 \\ 2 & 1 & 2 & 3 & 3 & 3 & 1 & 0 & 2 & 2 & 2 \\ 2 & 1 & 2 & 3 & 3 & 3 & 2 & 2 & 0 & 1 & 2 \\ 2 & 1 & 2 & 3 & 3 & 3 & 2 & 2 & 1 & 0 & 2 \\ 1 & 1 & 2 & 3 & 3 & 3 & 2 & 2 & 2 & 2 & 0 \end{bmatrix}$$

Remarque : La matrice de distance D d'un graphe G dépend toujours a celle de l'adjacence A du même graphe.

— La matrice de distance D d'un graphe G peut être construite en calculant successivement les puissances booléennes A^k ; avec k ∈ [1,D(G)], où $A = I + A$, I soit la matrice unitaire correspond au graphe G et D(G) son diamètre.

— Dans le cas des graphes orientés, cet algorithme s'appelle l'algorithme de Moore.

1.2.4.3 Matrice d'incidence

Définition 1.4.

Considérons un graphe orienté sans boucle G = (V,E) d'ordre n et de taille m. Sa matrice d'incidence notée $I = (i_{ij})$ est une matrice de dimension $n \times m$, tel que :

$$\begin{cases} i_{ij} = 1 & \text{si } v_i \text{ est l'extrémité initiale de } e_j \\ i_{ij} = -1 & \text{si } v_i \text{ est l'extrémité terminale de } e_j \\ i_{ij} = 0 & \text{si } v_i \text{ n'est pas une extrémité de } e_j \end{cases} \tag{1.19}$$

Où v_i sont les sommets de G et e_j sont ses arêtes.

Remarque : Pour un graphe non orienté sans boucle, la matrice d'incidence est définie par :

$$\begin{cases} i_{ij} = 1 & \text{si } v_i \text{ est une extrémité de } e_j \\ i_{ij} = 0 & \text{sinon} \end{cases} \tag{1.20}$$

Exemple : La matrice de distance du graphe G_2 de la FIGURE 1.30 est :

$$\begin{pmatrix} 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & 1 & 0 \\ -1 & -1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

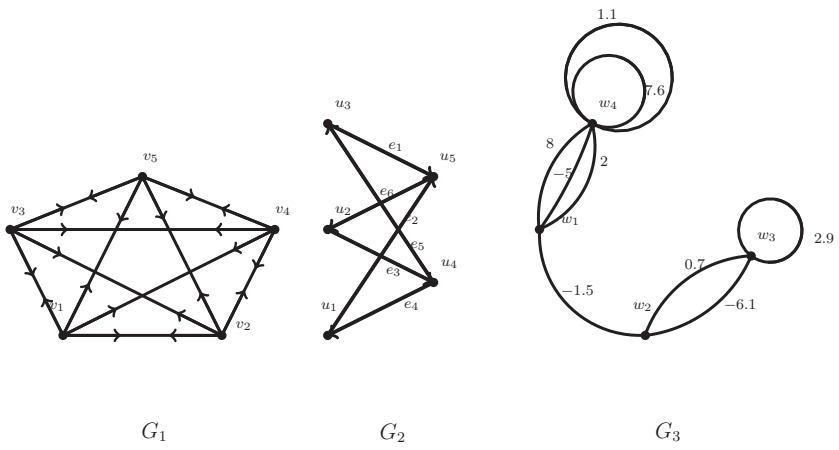


FIGURE 1.30 – Exemple de graphes orientés et non orientés finis

1.2.4.4 Listes d'adjacences

TABLE 1.1 – Les listes d'adjacences du graphe G_2 de la figure 1.30

Le sommet	Ses successeurs	Ses prédécesseurs
u_1	u_5	u_4
u_2	u_4	u_5
u_3	u_5	u_4
u_4	u_1, u_3	u_2
u_5	u_2	u_1, u_3

Un graphe orienté peut être représenté à l'aide d'un dictionnaire ; il s'agit d'une table à simple entrée où chaque ligne correspond à un sommet et comporte la liste des successeurs ou des prédécesseurs de ce sommet. Dans la mesure où, pour une table donnée, le nombre de successeurs ou de prédécesseurs n'est pas le même pour chaque sommet, il est préférable de représenter le dictionnaire sous forme de deux tableaux : le premier comprenant autant d'éléments que de sommets, ces éléments pointant, dans un second tableau, les débuts de listes de successeurs (ou de prédécesseurs).

1.2.4.5 Matrice Laplacienne

Définition 1.5.

Considérons La matrice Laplacienne notée L(G) correspondant au graphe G est la matrice définie comme suit :

$$L(G) = D(G) - A(G) \tag{1.21}$$

Où D(G) est la matrice diagonale comportant les degrés des sommets du graphe, et A(G) sa matrice d'adjacence.

Exemple : La matrice Laplacienne du graphe de la FIGURE 1.31 est :

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

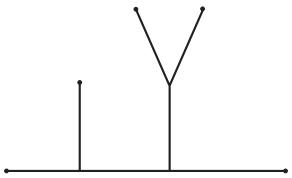


FIGURE 1.31 – Exemple d'un arbre

Remarque : Par la suite ne nous concéderons que les graphes simples, non orientés, planaires, connexes et finis.

Conclusion

Le premier chapitre introduit les principales notions en cryptographie, et en particulier les algorithmes de cryptographie à clé secrète. Après une introduction générale sur les générateurs de nombres aléatoires, les Codes d'authentication de Messages (MACs) et les fonctions de hachages, nous avons listé les attaques contre les mécanismes des systèmes cryptographiques. Par la suite, nous avons introduit les généralités sur les concepts fondamentaux des graphes et leurs propriétés en citant les différents modes de représentations d'un graphe. Nous présentons dans le chapitre suivant les différents algorithmes de chiffrement qui se basent sur la théorie des graphes en détaillant quelques-uns à travers des exemples d'illustrations du processus de chiffrement et de déchiffrement.

Chapitre

2

Chiffrement basé sur la théorie des graphes

Sommaire

Introduction

Ce chapitre vise avant tout à donner un aperçu global sur quelques travaux de la littérature qui bénéficient des propriétés de la théorie des graphes pour concevoir de nouveaux systèmes cryptographiques, et par la suite, à présenter les différentes techniques et méthodes utilisées dans ces systèmes.

2.1 Chiffrement basé sur la théorie des graphes

Aujourd'hui, la cryptographie moderne croise plusieurs disciplines, à savoir, les mathématiques, l'informatique, le génie électrique, etc. En informatique, la cryptographie désigne des techniques d'information et de communication sécurisées dérivées de concepts mathématiques et d'un ensemble de calculs basés sur des règles (Algorithmes), pour transformer les messages de manière difficile à déchiffrer. Ces algorithmes déterministes sont utilisés pour la génération de clés cryptographiques, la signature numérique, la vérification pour protéger la confidentialité des données, la navigation sur Internet et les communications confidentielles telles que les transactions par carte de crédit et le courrier électronique.

La théorie des graphes n'est pas une branche indépendante des mathématiques, elle se rattache à la programmation linéaire, la programmation convexe, la topologie, le calcul des probabilités, etc. En effet, Le concept récent de la théorie des graphes a été parfaitement intégré dans la cryptographie vu qu'il est capable de fournir des algorithmes de chiffrement plus robustes et qui sont très résistants face aux attaques contre les mécanismes de cryptographie.

De nos jours, la théorie des graphes a une grande contribution dans le développement de diverses techniques de chiffrement. Cependant, plusieurs méthodes ont été présentées dans ce contexte par la communauté de recherche. Citons à titre d'exemple quelques travaux :

Le système proposé dans [Yamuna *et al.*, 2012] se base sur un mécanisme de chiffrement utilisant la propriété du chemin hamiltonien. En fait, les données sont représentées par un chemin hamiltonien, et à partir duquel le graphe complet est construit en attribuant des faux poids aux arêtes restantes pour augmenter le niveau de sécurité.

Le travail [Al Etaiwi, 2014] propose un nouvel algorithme de chiffrement basé sur la théorie des graphes. L'auteur dans cette proposition présente un mécanisme de chiffrement utilisant le graphe complet et son arbre couvrant du poids minimal (MST) pour augmenter le degré de sécurité.

L'étude menée dans [Yamuna *et al.*, 2013a] montre que les circuits Hamiltoniens peuvent être utilisés pour représenter plusieurs messages grâce à un graphe, et que le chiffré peut être changé en fonctions du temps.

En 2015, les auteurs de [Agarwal et Uniyal, 2015] ont proposé un schéma de chiffrement efficace par la transformation des valeurs ASCII en nombre premiers ayant utilisé

une clé de chiffrement de taille similaire au message clair, par la suite ils ont généré un graphe pondéré (prime weighted graphe) d'une manière aléatoire en considérant les nombres premiers attribués en tant que poids des arrêtes.

En 2013, les auteurs de [Yamuna *et al.*, 2013b] ont utilisés la notation musicale pour représenter la clé secrète (note musicale) et des propriétés de la théorie des graphes pour générer des pseudo-clés. Cette approche est basée sur la méthode de " Propagating Cipher Block Chaining (PCBC)" pour le chiffrement des messages binaires. En 2014, les mêmes auteurs ont proposé une méthode de chiffrement de code PIN sous forme de digraphe [Yamuna *et al.*, 2014].

Le système proposé dans [AMOUNAS, 2016] applique certaines manipulations sur les données originales en utilisant la théorie des graphes et certaines de ses propriétés. L'idée principale de sa contribution dépend de la génération du graphe pondéré complet. Plus précisément, cette approche présente une nouvelle façon d'étiqueter les arrêtes d'un graphe. Ensuite, il applique l'approche matricielle basée sur les opérations ECC pour générer un texte chiffré fort.

Plusieurs propositions ont été suggérées dans la même thématique [Amudha *et al.*, 2018, Akl, 2019, Sensarma et Sarma, 2019, Rangaswamy et Gurusamy, 2018, Hashem, 2019, Yousif et Kashmar, 2019, Khaleel et Al-Shumam, 2020, Bekkaoui *et al.*, 2020b, Beaula et Venugopal, 2020, Perera et Wijesiri, 2021, Akl, 2020], nous détaillons dans les parties qui suivent quelques travaux de l'état de l'art.

2.1.1 Chiffrement basé sur un graphe eulérien

Dans cette partie nous introduisons le travail [Yamuna *et al.*, 2012].

2.1.1.1 Introduction

L'algorithme proposé représente une technique d'encodage et de décodage des données en utilisant les concepts de base de la théorie des graphes et les propriétés des matrices.

L'objectif de ce travail est de proposer un mécanisme de chiffrement où les sommets sont organisés sous forme de chemin eulérien. Cette méthode vise à utiliser la matrice d'adjacence comme un paramètre supplémentaire pour chiffrer et transmettre les données

2.1.1.2 Description du chiffrement/déchiffrement

Dans cette approche, les données sont d'abord encodées à partir du tableau d'encodages puis représentés comme poids des arêtes sur un chemin eulérien. La matrice d'adjacence est construite à partir de ce chemin eulérien où chaque entrée représente le poids de l'arête correspondante. Ensuite, chaque sommet du chemin eulérien est joint à une arête pour former un graphe complet et le faux message est transmis sur les arêtes nouvellement créées. Par la suite la matrice d'adjacence du graphe complet est construite et multipliée par la matrice du graphe eulérien. La matrice résultante est multipliée par

la matrice clé pour obtenir la matrice finale des données chiffrées. Le déchiffrement des données est effectué en utilisant les propriétés de la matrice.

Algorithme de chiffrement

Le processus de chiffrement est représenté comme suit :

— Saisir les données qui doivent être chiffrées. Puis les encoder en utilisant le tableau d'encodage.

— Ensuite, les données encodées sont représentées sur un graphe. S'il y a n éléments, nous avons besoin de $(n+1)$ sommets.

— Considérons un chemin avec n sommets. Marquer le sommet de départ avec la valeur 1.

— Attribuer à chaque arête son poids, qui correspond aux données encodées dans l'ordre.

— Calculer la matrice d'adjacence du chemin eulérien.

— Dessiner ensuite une arête entre chacun des sommets et transformer le graphe en un graphe complet.

— Attribuer des valeurs fausses aux arêtes nouvellement créées.

— Déterminer la matrice d'adjacence du graphe complet.

— Multiplier ensuite cette matrice par une matrice présente à la fois chez l'expéditeur et chez le destinataire, qui servira en tant que matrice clé permettant d'éviter que les données ne soient entre de mauvaises mains.

— La matrice finale correspond aux données chiffrées.

— Maintenant, envoyez la matrice finale et la matrice d'adjacence au récepteur sous un format linéaire (c'est-à-dire par colonne ou par ligne) avec un espace entre chaque élément. La forme du texte chiffré est définie comme suit : a n <données de la matrice finale> <matrice d'adjacence du graphe complet>.
Où.
a = valeur de l'indice du premier caractère
n = taille de la matrice

Algorithme de déchiffrement

Le processus de déchiffrement est représenté comme suit :

— Lire les données chiffrées et former les matrices correspondantes.

— Multiplier la matrice des données chiffrées par l'inverse de la matrice des clés. La matrice résultante nommée A.

— Calculer ensuite l'inverse de la matrice du graphe complet.

— Multiplier cette matrice par la matrice A.

— Dessiner le graphe à partir de la matrice obtenue après multiplication.

— Déterminer le poids des arêtes du graphe à partir du point de départ (valeur de l'indice du premier caractère).

— Enfin, décoder le graphe en utilisant le tableau d'encodage afin de récupérer le message original.

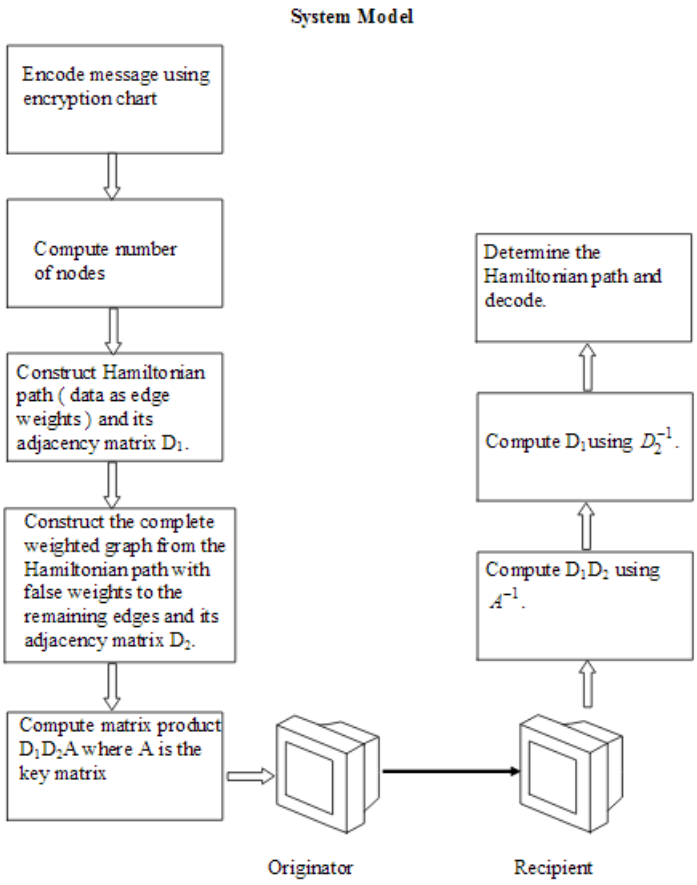


FIGURE 2.1 – Processus de chiffrement/Déchiffrement

2.1.1.3 Exemple

Supposons que nous voulons envoyer le message GOOD. Remplacer ce message par les valeurs 7, 15, 15, 4.

Étiqueter les arrêtes du chemin eulérien former par 5 sommets avec ces valeurs comme indiqué dans le graphe suivant.

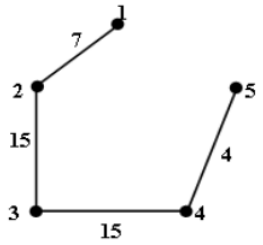


Fig. 1

Soit D_1 la matrice d'adjacence correspondant au graphe de la figure 1.

$$D_1 = \begin{bmatrix} 0 & 7 & 0 & 0 & 0 \\ 7 & 0 & 15 & 0 & 0 \\ 0 & 15 & 0 & 15 & 0 \\ 0 & 0 & 15 & 0 & 4 \\ 0 & 0 & 0 & 4 & 0 \end{bmatrix}$$

À partir du chemin eulérien à cinq sommets, nous construisons le graphe complet K_5. Ensuite, nous attribuons de faux poids aux arêtes restantes du graphe K_5, comme le montre la figure 2.

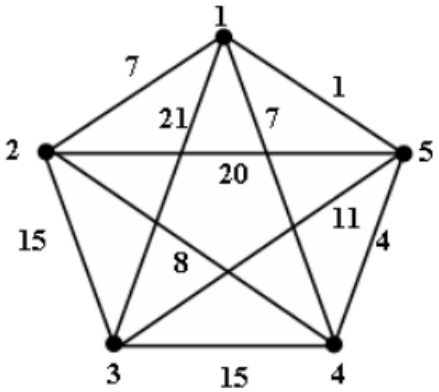


Fig. 2

Soit D_2 la matrice d'adjacence correspondant au graphe de la figure 2.

$$D_2 = \begin{bmatrix} 0 & 7 & 21 & 7 & 1 \\ 7 & 0 & 15 & 8 & 20 \\ 21 & 15 & 0 & 15 & 11 \\ 7 & 8 & 15 & 0 & 4 \\ 1 & 20 & 11 & 4 & 0 \end{bmatrix}$$

Soit $B{=}D_1 \cdot D_2$:

$$B = D_1 D_2 = \begin{bmatrix} 49 & 0 & 105 & 56 & 140 \\ 315 & 274 & 147 & 274 & 172 \\ 210 & 120 & 450 & 120 & 360 \\ 319 & 305 & 44 & 241 & 165 \\ 28 & 32 & 60 & 0 & 16 \end{bmatrix}$$

Dans cet exemple, nous choisissons la matrice A comme clé :

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

Ce qui donne :

$$BA = \begin{bmatrix} 350 & 301 & 301 & 196 & 140 \\ 1182 & 867 & 593 & 446 & 172 \\ 1260 & 1050 & 930 & 480 & 360 \\ 1074 & 755 & 450 & 406 & 165 \\ 136 & 108 & 76 & 16 & 16 \end{bmatrix}$$

Les données chiffrées à envoyer sont :

1 5 732 588 588 336 240 1168 1227 642 390 243 1368 1095 1071 621 453 641 567 284 163 33 319 242 154 121 121 0 12 21 7 1 12 0 21 8 20 21 21 0 3 11 78 3 0 11 1 20 11 11 0.

Le premier chiffre "1" est la valeur d'indice du mot et le deuxième "5" représente la taille de la matrice le reste sont les lignes de la matrice BA séparés par des espaces. Supposons que les données reçues sont les suivantes :

1 5 732 588 588 336 240 1168 1227 642 390 243 1368 1095 1071 621 453 641 567 284 163 33 319 242 154 121 121 0 12 21 7 1 12 0 21 8 20 21 21 0 3 11 78 3 0 11 120 11 11 0.

A partir des données que nous avons reçues, nous obtenons les matrices suivantes : BA et D_2.

$$BA=\begin{bmatrix} 732 & 1668 & 1368 & 641 & 319 \\ 588 & 1227 & 1095 & 567 & 242 \\ 588 & 642 & 1071 & 284 & 154 \\ 336 & 390 & 621 & 163 & 121 \\ 240 & 243 & 453 & 33 & 121 \end{bmatrix} \text{ et } D_2=\begin{bmatrix} 0 & 12 & 21 & 7 & 1 \\ 12 & 0 & 21 & 8 & 20 \\ 21 & 21 & 0 & 3 & 11 \\ 7 & 8 & 3 & 0 & 11 \\ 1 & 20 & 11 & 11 & 0 \end{bmatrix}.$$

En utilisant l'inverse de la matrice clé A, nous obtenons :

$$B=BAA^{-1}=\begin{bmatrix} 144 & 0 & 252 & 96 & 240 \\ 441 & 585 & 252 & 147 & 243 \\ 273 & 24 & 450 & 168 & 453 \\ 74 & 283 & 121 & 130 & 33 \\ 77 & 88 & 33 & 0 & 121 \end{bmatrix}$$

Également

$$D_2^{-1}=\begin{bmatrix} 0.023 & 0.019 & 0.077 & -0.113 & -0.049 \\ 0.019 & -0.036 & -0.002 & 0.066 & 0.015 \\ 0.077 & -0.002 & 0.007 & -0.010 & -0.049 \\ -0.113 & 0.066 & -0.010 & -0.099 & 0.118 \\ -0.049 & 0.015 & -0.049 & 0.118 & 0.034 \end{bmatrix}$$

Par conséquent :

$$D_1=BD_2^{-1}=D_1D_2D_2^{-1}=\begin{bmatrix} 0 & 12 & 0 & 0 & 0 \\ 12 & 0 & 21 & 0 & 0 \\ 0 & 21 & 0 & 3 & 0 \\ 0 & 0 & 3 & 0 & 11 \\ 0 & 0 & 0 & 11 & 0 \end{bmatrix}$$

L'arbre couvrant résultant correspondant à D_1 est illustré dans la figure 3.

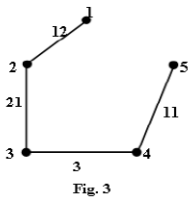


Fig. 3

Donc le message déchiffré est 12 21 3 11 à partir du tableau d'encodage, le message original est LUCK.

2.1.2 Chiffrement basé sur un graphe cycle

Dans cette partie nous introduisons le travail [Al Etaiwi, 2014].

2.1.2.1 Introduction

L'algorithme proposé représente un nouvel algorithme de chiffrement permettant de chiffrer et de déchiffrer des données en toute sécurité grâce aux avantages des propriétés de la théorie des graphes. Le nouvel algorithme de chiffrement symétrique utilise les concepts de graphe cycle, de graphe complet et d'arbre couvrant de poids minimal pour générer un texte chiffré complexe en utilisant une clé partagée.

2.1.2.2 Description du chiffrement/déchiffrement

La première étape de cet algorithme consiste à représenter les données comme des sommets dans un graphe, chaque caractère étant représenté par un sommet tandis que tous les caractères adjacents dans la donnée (texte clair) seront représentés comme des sommets adjacents dans le graphe, nous continuons à ajouter des sommets jusqu'à former un graphe cycle. Chaque arête du graphe a son propre poids qui représente la distance entre ces deux caractères dans la table d'encodage utilisée pour encoder tous les caractères de l'alphabet.

Ensuite, chaque sommet du graphe est relié à des arêtes pour former un graphe complet. Tandis que chaque nouvelle arête ajoutée à un poids de séquence qui commence à partir du dernier indice dans la table d'encodage. La matrice d'adjacence est construite pour le graphe complet. Ensuite, l'arbre couvrant de poids minimal (ACM) est calculé à partir du graphe complet et représenté sous forme de matrice d'adjacence qui conserve l'ordre des caractères de la donnée dans sa diagonale. La matrice d'adjacence du graphe complet est multipliée par la matrice d'adjacence de l'ACM. La matrice résultante est multipliée par la matrice clé. La matrice finale correspond aux données chiffrées à envoyer au destinataire.

Algorithme de chiffrement

Le processus de chiffrement est représenté comme suit :

— Ajouter un caractère spécial pour indiquer le caractère de départ (L et A).

— Ajoutez au graphe des sommets pour chaque caractère du texte clair.

— Reliez les sommets entre eux en ajoutant une arête entre chaque caractère séquentiel du texte clair jusqu'à ce que nous formions un graphe cycle.

— Pondérer chaque arête en utilisant la table d'encodage. Le poids de chaque arête représente la distance entre les deux sommets adjacents à partir de la table de codage.

— Ajouter d'autres arêtes pour former un graphe complet M_1, chaque nouvelle arête ajoutée à un poids séquentiel à partir du poids maximum dans la table d'encodage.

— Ensuite, trouver l'arbre couvrant de poids minimal (ACM) M_2.

— Puis on stocke l'ordre des sommets dans la matrice M2 au niveau de la diagonale.

— Ensuite, nous multiplions les matrices M_1 par M_2 pour obtenir M_3.

— Après cela, nous multiplions M_3 par une clé partagée prédéfinie K pour former C.

— Ensuite, nous concaténons les lignes de la matrice C et M_1 pour former un seul vecteur qui représente le texte chiffré.

Algorithme de déchiffrement

Le processus de déchiffrement est représenté comme suit :

— Le récepteur calcule M_3 en utilisant la forme inverse de la clé partagée K^{-1}.

— Ensuite, il calcule M_2 en utilisant la matrice inverse M_1^{-1}.

— finalement, il calcule le texte original en déchiffrant M_1 à l'aide de la table d'encodage.

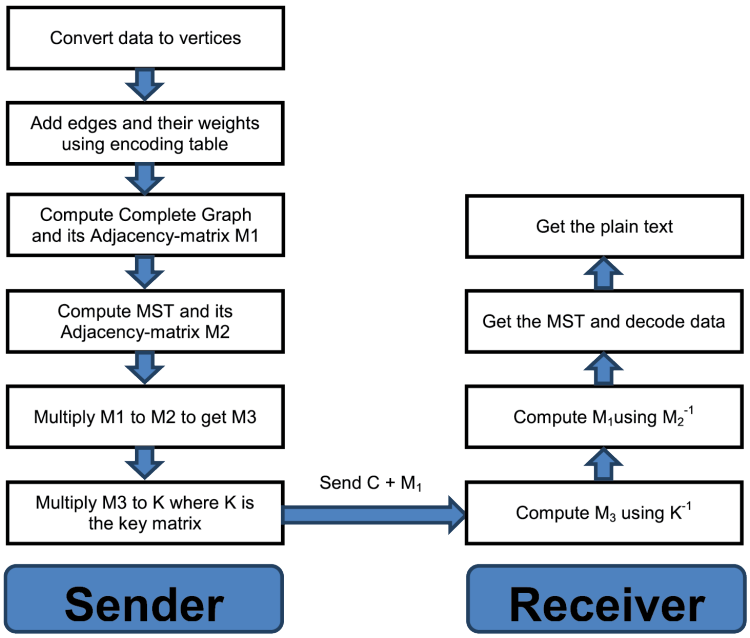


FIGURE 2.2 – Processus de chiffrement / déchiffrement

2.1.2.3 Exemple

Supposons que nous voulons chiffrer le message WAEL pour l'envoyer au récepteur. La première étape consiste à convertir le message en un graphe, en convertissant chaque caractère en un sommet, comme le montre la figure 2 :

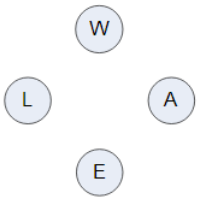


Fig. 2. Convert each character to vertex

Ensuite, reliez chaque deux caractères séquentiels ensemble pour former un graphe cycle. Puis, pondérer chaque arête en utilisant la table d'encodage ci-dessous :

Table 1. Encoding table

A	B	C	D	E	...	L	...	W	X	Y	Z
1	2	3	4	5	...	12	...	23	24	25	26

De même, le graphe sera comme indiqué dans la Figure 3 ci-dessous :

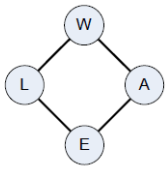


Fig. 3. Graph contains plain text characters.

Le poids de chaque arête représente la distance entre les deux sommets connectés est calculée à partir de la table d'encodage. Ainsi, l'arête reliant le sommet W au sommet A à un poids égal à la distance entre les deux lettres figurant dans la table, comme indiqué ci-dessous :

$$\begin{cases} \text{Distance (A,W)} = \text{Code(A)} - \text{Code(W)} \\ \quad = 1 - 23 \\ \quad = -22 \end{cases} \tag{2.1}$$

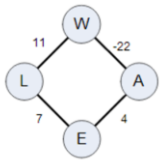


Fig. 4. Weighted graph contains plain text characters

Par la suite, nous continuons à ajouter des arêtes pour former un graphe complet, chaque nouvelle arête ajoutée a un poids séquentiel à partir du poids maximum dans la table d'encodage (26 +1 = 27). Voir la figure 5.

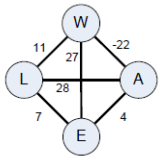


Fig. 5. Complete plain graph

À ce stade, nous ajoutons un caractère spécial avant le premier caractère pour le pointer vers le premier caractère, Soit A comme le montre la figure 6.

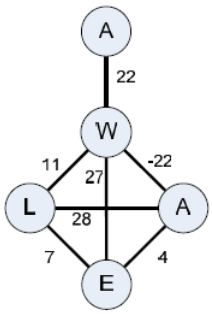


Fig. 6. Complete plain graph with a special character

Le graphe complet de la Figure 5 est représenté sous forme d'une matrice M_1 :

$$M_1 = \begin{bmatrix} 0 & 22 & 0 & 0 & 0 \\ 22 & 0 & -22 & 27 & 11 \\ 0 & -22 & 0 & 4 & 28 \\ 0 & 27 & 4 & 0 & 7 \\ 0 & 11 & 28 & 7 & 0 \end{bmatrix}$$

Ensuite, nous déterminons l'arbre couvrant de poids minimal (ACM) comme le montre la figure 7.

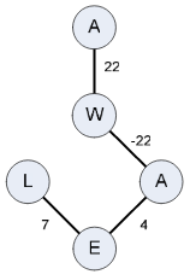


Fig. 7. Minimum spanning tree graph

Soit M_2 la matrice d'adjacence pour le graphe de la figure 7

$$M_2 = \begin{bmatrix} 0 & 22 & 0 & 0 & 0 \\ 22 & 0 & -22 & 0 & 0 \\ 0 & -22 & 0 & 4 & 0 \\ 0 & 0 & 4 & 0 & 7 \\ 0 & 0 & 0 & 7 & 0 \end{bmatrix}$$

Character	A	W	A	E	L
order	0	1	2	3	4

Maintenant, nous stockons l'ordre des caractères dans la diagonale au lieu des 0

$$M_2 = \begin{bmatrix} 0 & 22 & 0 & 0 & 0 \\ 22 & 1 & -22 & 0 & 0 \\ 0 & -22 & 2 & 4 & 0 \\ 0 & 0 & 4 & 3 & 7 \\ 0 & 0 & 0 & 7 & 4 \end{bmatrix}$$

Ensuite, nous multiplions la matrice M_1 par M_2 pour former la matrice M_3

$$M_3 = M_1 M_2 = \begin{bmatrix} 484 & 22 & -484 & 0 & 0 \\ 0 & 968 & 64 & 70 & 233 \\ -484 & -22 & 500 & 208 & 140 \\ 594 & -61 & -586 & 65 & 28 \\ 242 & -605 & -158 & 133 & 49 \end{bmatrix}$$

Finalement, nous utilisons la clé partagée K pour chiffrer M_3

$$K = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 836 & 302 & -664 & 476 & 450 \\ 352 & 280 & -180 & 476 & 450 \\ 352 & -688 & -244 & 406 & 217 \\ 836 & -666 & -744 & 198 & 77 \\ 242 & -605 & -158 & 133 & 49 \end{bmatrix}$$

Les données à envoyer sont la concaténation des lignes de la matrice C et M_1

836 302 -664 476 450 352 280 -180 476 450 352 -688 -244 406 217 836 -666 -744 198 77 242 -605 -158 133 49 0 22 0 0 0 22 0 -22 27 11 0 -22 0 4 28 0 27 4 0 7 0 11 28 7 0

Du côté du récepteur, on obtient M_3 en multipliant le texte chiffré reçu par la forme inverse de la clé partagée K^{-1}.

$$M_3 = CK^{-1} = \begin{bmatrix} 836 & 302 & -664 & 476 & 450 \\ 352 & 280 & -180 & 476 & 450 \\ 352 & -688 & -244 & 406 & 217 \\ 836 & -666 & -744 & 198 & 77 \\ 242 & -605 & -158 & 133 & 49 \end{bmatrix} * \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 484 & 22 & -484 & 0 & 0 \\ 0 & 968 & 64 & 70 & 233 \\ -484 & -22 & 500 & 208 & 140 \\ 594 & -61 & -586 & 65 & 28 \\ 242 & -605 & -158 & 133 & 49 \end{bmatrix}$$

Ensuite, on calcule M_2 en multipliant M_3 par M_1^{-1} :

$$M_2 = M_3M_1^{-1} = \begin{bmatrix} 0 & 22 & 0 & 0 & 0 \\ 22 & 1 & -22 & 0 & 0 \\ 0 & -22 & 2 & 4 & 0 \\ 0 & 0 & 4 & 3 & 7 \\ 0 & 0 & 0 & 7 & 4 \end{bmatrix}$$

Ainsi, M_2 représente le graphe final ci-dessous (Fig. 8) (sans prendre en considération la diagonale) que nous utilisons pour retrouver le message original :

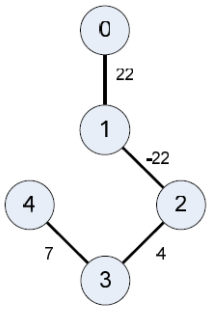


Fig. 8. Final graph

Nous supposons que le sommet 0 est A, donc en utilisant la table d'encodage, le sommet 1 = code(A) + 22 = 23 qui représente le caractère W, et le sommet 2 = code(W) - 22 = 1 qui représente le caractère A, et ainsi de suite jusqu'à ce que nous obtenions le texte original ($WAEL$).

2.1.3 Chiffrement basé sur un graphe hamiltonien

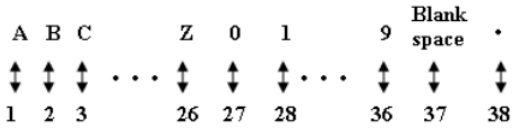
Dans cette partie nous introduisons le travail [Yamuna *et al.*, 2013a].

2.1.3.1 Introduction

La théorie des graphes est largement utilisée comme outil de chiffrement, en raison de ses diverses propriétés et de sa représentation sous forme matricielle. Les fonctions qui dépendent du temps ne sont pas très utilisées dans le chiffrement. Ce travail [Yamuna *et al.*, 2013a] propose un algorithme de chiffrement de messages multiples utilisant les circuits Hamiltoniens et les fonctions dépendantes du temps.

Tableau d'encodage

Le tableau d'encodage peut être conçu selon le besoin et l'intérêt de l'encodeur et le récepteur. Ici nous fournissons un tableau de cryptage très basique.



2.1.3.2 Description du chiffrement/déchiffrement

La première étape de cet algorithme est de décider d'abord le nombre de messages à chiffrer. Sachant que dans un graphe complet avec n sommets, il existe $(n-2)/2$ circuits hamiltoniens disjoints, si n est un nombre impair > 3.

Chaque message sera chiffré sur ces circuits hamiltoniens. Supposons que nous avons besoin de transmettre K messages, nous choisissons un graphe complet avec au moins K circuits hamiltoniens disjoints.

Chaque message est converti en une chaîne numérique utilisant le tableau d'encodage. Les arrêtes des circuits hamiltoniens sont utilisés pour chiffrer les messages. Les différents messages seront chiffrés sur différents circuits. Ensuite, nous construisons la matrice d'adjacence du graphe complet. Cette matrice est le chiffré envoyer au récepteur.

Algorithme de chiffrement

Le processus de chiffrement est représenté comme suit :

1. Soit K le nombre de messages distincts à chiffrer
2. Convertir le message en données numériques en utilisant le tableau d'encodage.
3. Choisir un graphe complet à n sommets de telle sorte que $(n-1)\ /\ 2 \geq K$.

4. Chaque message est chiffré le long des arêtes du circuit hamiltonien. Les arêtes restantes, s'il y en a, sont assignées à des valeurs numériques fictives, de telle sorte que nous générons un graphe complet à n sommets.
5. Construire sa matrice d'adjacence M_0.
6. Maintenant, on applique une fonction $f(t)$ sur $M_0[i,j]$, et avec des changements de temps les valeurs de la matrice sont changés. Supposons que nous choisissions M_1[i,j] $= 2 \times M_0$[i,j] $+ 1$.
7. Maintenant pour le temps t = r, le cas sera M_r[i,j] $= 2 \times M_{r-1}$[i,j] $\times$ r $+ 1$.
8. Envoyer $M_0[i,j]$ au récepteur.

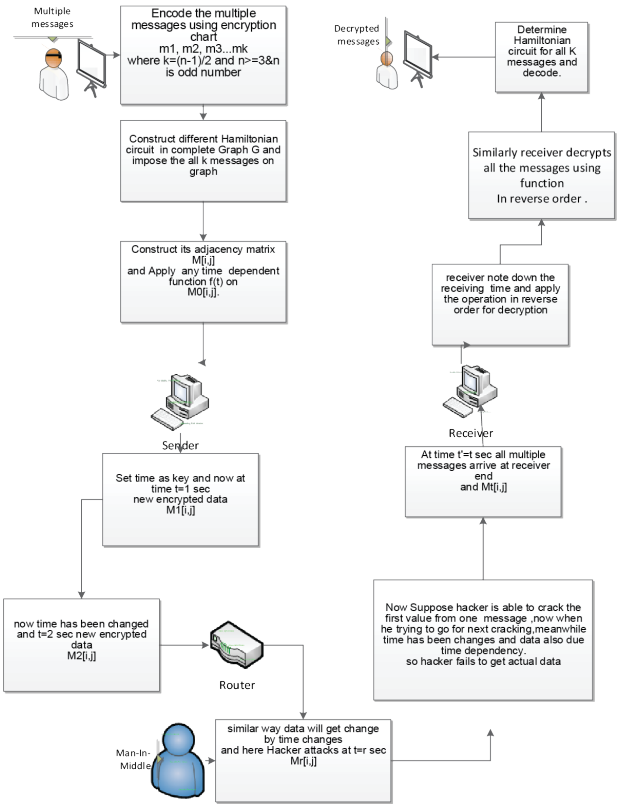


FIGURE 2.3 – Processus de chiffrement/Déchiffrement

Algorithme de déchiffrement

Le processus de déchiffrement est représenté comme suit :

1. Recevoir la matrice et noter le moment où il reçoit le message.

2. Appliquer l'opération dans l'ordre inverse pour récupérer les données en utilisant.

$$\begin{cases} M_{t-1}[i,j] = (M_t[i,j] - 1)/(2 \times t) \\ M_{t-2}[i,j] = (M_t[i,j] - 1)/(2 \times (t-1)) \\ \vdots \\ M_0[i,j] = (M_1[i,j] - 1)/(2) \end{cases} \tag{2.2}$$

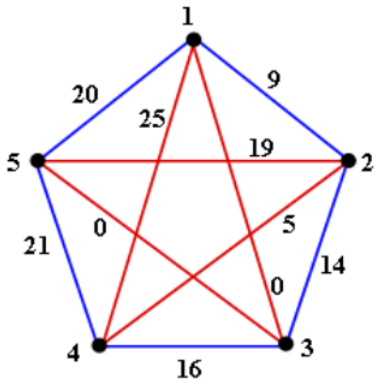
3. Extraire les différents messages des différents circuits hamiltoniens.

4. Déchiffrer le message original à partir de la table d'encodage.

2.1.3.3 Exemple

Soit INPUT et YES les deux messages à envoyer. À partir du tableau d'encodage, les caractères sont convertis en valeurs numériques à 9 14 16 21 20 et 25 5 19.

Comme le message le plus grand est de taille cinq, nous choisissons le graphe complet avec cinq sommets comme illustré dans la figure ci-dessous.



Le premier circuit hamiltonien est représenté en couleur bleue, le circuit est le suivant : 1, 2, 3, 4, 5, 1 , le second circuit est 1, 4, 2, 5, 3, 1 (montré en rouge).

Le message est chiffré dans les arrêtes du graphe cle montre la figure, Les arrêtes restantes (5, 3) et (3, 1) reçoivent l'étiquette 0.

La matrice d'adjacence du graphe ci-dessus et qui sera envoyée au récepteur est définie comme suit :

$$A = \begin{bmatrix} 0 & 9 & 0 & 25 & 20 \\ 9 & 0 & 14 & 5 & 19 \\ 0 & 14 & 0 & 16 & 0 \\ 25 & 5 & 16 & 0 & 21 \\ 20 & 19 & 0 & 21 & 0 \end{bmatrix}$$

Lorsque le récepteur reçoit la matrice, il recevra la matrice qui est modifiée en fonction de la fonction dépendante du temps que nous choisissons.

2.1.4 Chiffrement basé sur un graphe pondéré premier

Dans cette partie nous introduisons le travail [Agarwal et Uniyal, 2015].

2.1.4.1 Introduction

Dans ce travail, les auteurs définissent le graphe pondéré premier et proposent un schéma de chiffrement efficace utilisant le graphe pondéré primaire dans un système cryptographique pour une communication sécurisée.

2.1.4.2 Description du chiffrement/déchiffrement

Dans le cadre de l'algorithme proposé, le message sera chiffré sous forme d'un graphe pondéré premier. Comme le graphe est généré de manière aléatoire, il est difficile de déterminer le chemin eulérien contenant le message chiffré sans la possession de la clé de déchiffrement. Le schéma proposé utilise un graphe simple, car l'existence des arêtes parallèles et des boucles crée une incertitude lors de la sélection des arêtes. L'algorithme de chiffrement est représenté comme suit :

- Attribuer à chaque caractère du message la valeur ASCII correspondante.
- Ajouter la longueur de la chaine de caractères(texte clair) à la valeur ASCII de chaque caractère.
- Affecter un nombre premier pour chaque valeur numérique obtenue après l'addition. Si la valeur est un nombre premier, considérer le nombre tel qu'il est et si le nombre est un nombre composé, ajouter la valeur pour en faire un nombre premier qui vient juste après, de telle sorte que tous les caractères reçoivent un nombre premier.
- Dessiner un graphe pondéré premier de façon aléatoire en attribuant les nombres premiers comme poids des arêtes et en choisissant au hasard le nombre des sommets.
- Choisir 7 sommets et attribuer aléatoirement 13 nombres premiers de la chaîne aux 13 arêtes et dessiner un graphe pondéré premier aléatoire.
- Envoyez ce graphe comme message chiffré.

2.1.4.3 Exemple

Soit : « IL EST SECRET » le message à chiffrer

- affecter la valeur ASCII à chaque caractère du message en utilisant tableau 1. Le tableau 2 montre l'affectation de la valeur ASCII aux caractères.

Table 2: Assignment of ASCII Value to the Characters

Original Message	I	T		I	S		S	E	C	R	E	T	.
ASCII Value Corresponding to each character	73	84	32	73	83	32	83	69	67	82	69	84	46

— Ajoutez la longueur de la chaîne à la valeur ASCII correspondant à chaque caractère. Ici, la longueur de la chaîne est 13. L'ajout de la longueur de la chaîne à la valeur ASCII est indiqué dans le tableau 3.

Table 3: Addition of the Length of the String in the ASCII Value

Original Message	I	T		I	S		S	E	C	R	E	T	.
Number assigned to each character	73	84	32	73	83	32	83	69	67	82	69	84	46
Values after adding length of the string to ASCII value	86	97	45	86	96	45	96	82	80	95	82	97	59

— Affecter un nombre premier pour chaque valeur numérique obtenue après l'addition. Si la valeur est un nombre premier, considérez le nombre tel qu'il est et si le nombre est un nombre composé, ajouter une certaine valeur pour en faire un nombre premier qui vient juste après, pour que tous les caractères reçoivent un nombre premier comme indiqué dans le tableau 4.

Table 4: Assignment of a Prime Number to Each Character

Original Message	I	T		I	S		S	E	C	R	E	T	.
Number assigned to each character	73	84	32	73	83	32	83	69	67	82	69	84	46
Values after adding length of the string to ASCII value	86	97	45	86	96	45	96	82	80	95	82	97	59
Value to be added to make each number of the string prime	3	0	2	3	1	2	1	1	3	2	1	0	0
Prime number corresponding to each character	89	97	47	89	97	47	97	83	83	97	83	97	59

La chaîne chiffrée de nombres premiers est présentée dans le tableau 5.

Table 5: Encrypted String of Prime Numbers

Prime numbers string	89	97	47	89	97	47	97	83	83	97	83	97	59

— Dessiner un graphe pondéré primaire de manière aléatoire avec les nombres premiers attribués en tant que poids des arrêtes et le nombre de sommets doit être sélectionné aléatoirement.

— Choisissons 7 sommets et affectons au hasard les 13 nombres premiers de la chaîne aux 13 arêtes et dessiner un graphe pondéré primaire aléatoire tel qu'illustre à la figure 6.

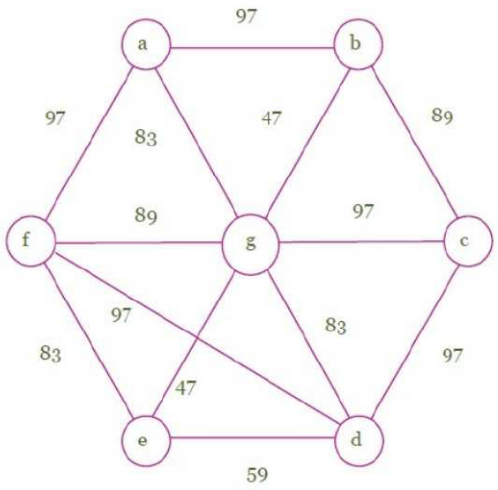


Figure 6: Prime Weighted Graph(Encrypted Message)

— Envoyez ce graphe comme message chiffré.

L'algorithme de déchiffrement est représenté comme suit :

— Le déchiffrement du graphe pour obtenir le message original nécessite d'abord de connaître la séquence d'arêtes à considérer, c'est-à-dire que la première clé de déchiffrement est la séquence d'arêtes indiquée dans le tableau 6.

Table 6: Sequence of Edges (First Decryption Key)

Prime numbers string	89	97	47	89	97	47	97	83	83	97	83	97	59

— Soustraire les éléments de la deuxième clé de déchiffrement du nombre premier correspondant. La deuxième clé de déchiffrement est présentée dans le tableau 8.

Table 8: Second Decryption key

Prime numbers string	89	97	47	89	97	47	97	83	83	97	83	97	59
Second decryption key to be subtracted	3	0	2	3	1	2	1	1	3	2	1	0	0
Resultant string of numbers	86	97	45	86	96	45	96	82	80	95	82	97	59

— La chaîne des nombres après soustraction de la deuxième clé de déchiffrement est calculée dans le tableau 9.

Table 9: String of Numbers after Subtracting Second Decryption Key

Resultant string of numbers	86	97	45	86	96	45	96	82	80	95	82	97	59
Length of the string	13	13	13	13	13	13	13	13	13	13	13	13	13
Original string of numbers after subtraction	73	84	32	73	83	32	83	69	67	82	69	84	46

— Maintenant, à partir de cette chaîne de nombres résultante, soustraire la longueur de la chaîne de caractères (texte clair) de chaque nombre de la chaîne résultante. Après la soustraction, la chaîne originale de nombres est indiquée dans le tableau 10.

Table 10: After Subtraction the Original String of Numbers

String of numbers	73	84	32	73	83	32	83	69	67	82	69	84	46
Corresponding character	I	T		I	S		S	E	C	R	E	T	.

— Déterminer le caractère correspondant à chaque nombre (valeur ASCII) de la chaîne comme indiqué dans le tableau 10 afin d'obtenir le message original.

2.1.5 Chiffrement basé sur un graphe orienté

Dans cette partie nous introduisons le travail [Yamuna *et al.*, 2014].

2.1.5.1 Introduction

Dans le cadre travail, l'algorithme proposé représente une méthode de chiffrement des codes PIN à quatre chiffres sous forme de digraphe.

2.1.5.2 Description du chiffrement/déchiffrement

Tous les graphes étudiés dans cette approche sont des graphes étiquetés. Nous considérons des chaînes numériques S de quatre chiffres. La construction des digraphe se fait comme suit :

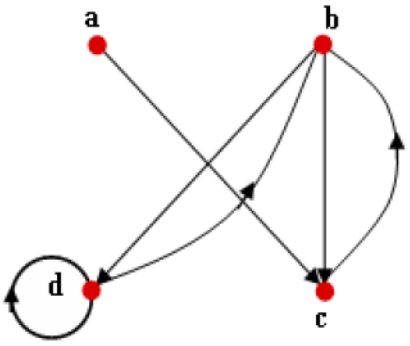
Construction du graphe

1. Convertir chaque chiffre du code pin S en une chaîne binaire de 4 bits.
2. Construire une matrice binaire M d'ordre 4, où chaque ligne de la matrice représente la conversion binaire du numéro de pin, c'est-à-dire que la première ligne représente la chaîne binaire correspondant au premier chiffre du numéro du code pin ... la quatrième ligne représente la chaîne binaire correspondant au quatrième chiffre du code pin.

2. En considérant M comme une matrice d'adjacence, on peut construire le digraphe G correspondant à M. Par exemple, si les numéros du code pin sont 2345. La conversion binaire est 0010, 0011, 0100, 0101. La matrice binaire M correspondante est :

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

Le graphe orienté G correspondant à M est :

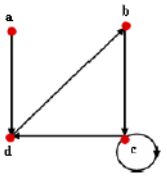
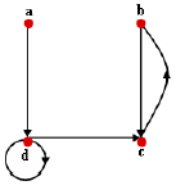
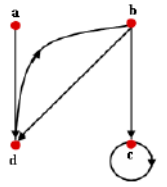
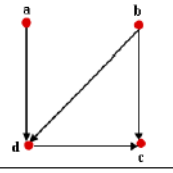
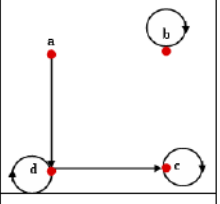
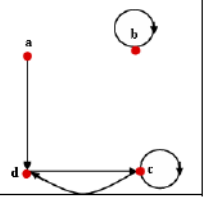


Considérant 4 chiffres, le nombre total de permutations possibles est de 24. Nous vérifions maintenant que les 24 graphes ne sont pas isomorphes et qu'ils peuvent être utilisés pour le chiffrement. Étant donné quatre chiffres 1, 2, 3, 4, il y a 24 codes pin possible en utilisant ces quatre chiffres.

$$\begin{cases} 1234, 1243, 1324, 1342, 1423, 1432, \\ 2134, 2143, 2314, 2341, 2413, 2431, \\ 3124, 3142, 3214, 3241, 3412, 3421, \\ 4123, 4132, 4213, 4231, 4312, 4321 \end{cases}$$

Dans les six premières combinaisons, 1 est le premier chiffre $\cdots$ dans les six dernières combinaisons, 4 est le premier chiffre. Le tableau 2 représente tous les graphes possibles pour les six premières combinaisons.On constate que les graphes correspondants ne sont pas isomorphes. De même les deuxième, troisième et quatrième ensemble de combinaisons engendreront des graphes non isomorphes entre eux.

Table 2

S. No	Pin Number	Adjacency Matrix	Graph
1	1234	$\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&0&1&1\\0&1&0&0\end{bmatrix}$	
2	1243	$\begin{bmatrix}0&0&0&1\\0&0&1&0\\0&1&0&0\\0&0&1&1\end{bmatrix}$	
3	1324	$\begin{bmatrix}0&0&0&1\\0&0&1&1\\0&0&1&0\\0&1&0&0\end{bmatrix}$	
4	1342	$\begin{bmatrix}0&0&0&1\\0&0&1&1\\0&1&0&0\\0&0&1&0\end{bmatrix}$	
5	1423	$\begin{bmatrix}0&0&0&1\\0&1&0&0\\0&0&1&0\\0&0&1&1\end{bmatrix}$	
6	1432	$\begin{bmatrix}0&0&0&1\\0&1&0&0\\0&0&1&1\\0&0&1&0\end{bmatrix}$	

Algorithme de chiffrement

Le processus de chiffrement est représenté comme suit :

1. Construire la matrice M comme expliqué.
2. Générer le digraphe étiqueté G correspondant à M.
3. Envoyez G au récepteur.

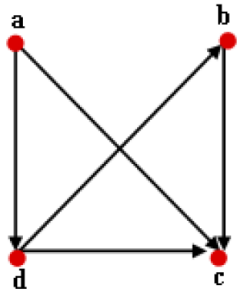
Pour le déchiffrement, nous inversons le processus.

2.1.5.3 Exemple

Supposons que nous souhaitions chiffrer le code pin de 7206. La matrice d'adjacence correspondante est :

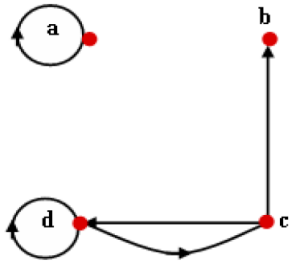
$$M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

et le graphe correspondant à la matrice M est :



qui sera envoyé au récepteur.

Si le graphe reçu est :



La matrice d'adjacence correspondante est :

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

Par conséquent, le code pin déchiffré est le 8053.

2.1.6 Génération de la Clé de chiffrement basé sur un graphe pondéré

Dans cette partie nous introduisons le travail [Yamuna *et al.*, 2013b].

2.1.6.1 Introduction

L'algorithme proposé représente un nouveau chiffrement par bloc en utilisant le mode d'opération CBC (cipher bloc chaining). Cette méthode se base sur l'utilisation d'une partition de musique pour construire un graphe. La séquence de degrés du graphe est utilisée comme clé.

2.1.6.2 Description du chiffrement

Méthode 1 (méthode CBC améliorée)

L'algorithme proposé représente une amélioration de la méthode CBC. Dans cette méthode, la chaîne binaire à chiffrer est censée être un multiple de 8. Pour ce faire, nous préfixons les valeurs manquantes par des 0. Nous joignons également une chaîne supplémentaire de longueur 8 pour indiquer le nombre de 0 préfixés. Les quatre derniers bits seront la conversion binaire du nombre de 0 préfixés, c'est-à-dire que si les 8 dernières entrées de la chaîne binaire à chiffrer sont 00000000, cela signifie que la chaîne originale à chiffrer est un multiple de 8. Ainsi, chaque chaîne binaire à chiffrer est préfixée par un minimum de 00000000.

Nous introduisons une nouvelle façon de définir le vecteur d'initialisation et la clé E_k.

Pour convertir la partition de musique en forme binaire, nous utilisons la substitution suivante :

TABLE I
Binary Conversion

Music note	Binary
C	000
D	001
E	010
F	011
G	100
A	101
B	110
-	111

Vecteur d'initialisation (C_0)

Nous choisissons une note de musique de notre choix. En utilisant la table de conversion binaire, nous obtenons la conversion binaire des quatre premiers caractères de la chaîne. Cela donne une chaîne binaire S de longueur 12. Les huit premiers bits de S sont considérés comme vecteur d'initialisation (IV).

Décalage (Shift)
Les quatre derniers bits de la chaîne S constituent le décalage.

Key (E_k)
Nous construisons d'abord le graphe G comme suit.

Les sommets de G sont les huit notes de musique C, D, E, F, G, A, B, Rest. Les arêtes sont construites à partir de la note de musique. Nous dessinons une arête de chaque note de musique à la note de musique suivante dans l'ordre dans laquelle elles apparaissent dans la chaîne musicale choisie. Si la partition de musique est EDDA, alors il y a une arête de E à D, D à D ... A à D. Nous attribuons également des poids aux arêtes du graphe. Ces poids représentent le nombre d'arêtes entre les sommets correspondants. Ainsi, le graphe correspondant à EDDDAD est illustré par la figure 1.

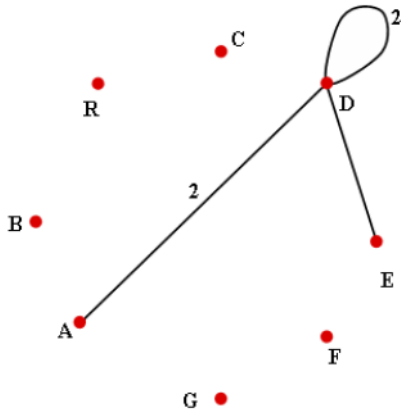


Fig. 1. Graph Corresponding to EDDDAD

Déterminer le degré de chaque sommet du graphe, pour construire une séquence d'entiers de longueur 8, $S = a_1, a_2, \ldots, a_8$, qui représente le degré de chaque sommet (Note de musique). Ensuite générer une chaîne binaire S_1 de longueur 8 comme suit.

$$S_1 = \begin{cases} 1 \text{ si } a_i \text{ est impair} \\ 1 \text{ si } a_i \text{ est pair, r} = \text{impair} \\ 0 \text{ si } a_i \text{ est pair, r} = \text{pair} \end{cases} \tag{2.3}$$

où r est le quotient si a_i est divisé par 2. Pour le graphe de la figure 1, la séquence de degrés est 00100100, et la chaîne résultante S_1 est 00100100. Cette chaîne binaire résultante S_1 est notre E_k.

Utilisation de Décalage
La valeur de E_k calculée est utilisée pour le chiffrement du premier bloc. La valeur du décalage est convertie en sa forme numérique, comprise entre 0 et 7. Si la valeur convertie du décalage est p, alors chaque position de E_k est décalée de p places. Par exemple, si E_k

est 10110101 et que la valeur du décalage est 2, la nouvelle valeur de E_k pour le deuxième bloc est 01101101. Pour chaque bloc, différentes valeurs de E_k sont calculées

Algorithme de chiffrement

Le processus de chiffrement est représenté comme suit :

1. Déterminer le vecteur d'initialisation IV (C_0), le décalage et la clé E_k comme expliqué.
2. Définir P_1 comme les huit premiers bits de la chaîne binaire à chiffrer.
3. Le premier bloc est chiffré en utilisant $C_i = E_k\ (P_i \oplus (P_{i-1} \oplus C_{i-1})\ ),\ P_0 \oplus C_0 = IV$.
4. Pour le deuxième bloc, la nouvelle valeur de E_k est calculée en utilisant le décalage.
5. Continuer l'itération jusqu'à ce que tous les blocs soient chiffrés.
6. Envoyer le chiffré au destinataire

Méthode 2 (CBC amélioré utilisant la matrice d'adjacence)

Cette méthode est similaire à la méthode 1, sauf que nous modifions la façon dont E_k est calculé. Nous construisons le graphe comme décrit dans la méthode 1, puis nous créons la matrice d'adjacence de G d'ordre 8. Chaque ligne de la matrice d'adjacence sera utilisée comme E_k. Si le nombre d'itérations est supérieur à 8, nous recommençons le traitement des lignes à partir de la ligne 1 jusqu'à ce que toutes les itérations soient terminées.

Algorithme de chiffrement

Le processus de chiffrement est représenté comme suit :

1. Déterminer le vecteur d'initialisation IV (C_0), le décalage et la clé E_k comme expliqué.
2. Définir P_1 comme les huit premiers bits de la chaîne binaire à chiffrer.
3. Le premier bloc est chiffré en utilisant $C_i = E_k\ (P_i \oplus (P_{i-1} \oplus C_{i-1})\ ),\ P_0 \oplus C_0 = IV$.
4. Pour la deuxième bloc, la nouvelle valeur de E_k est la deuxième ligne de la matrice d'adjacence A.
5. Continuer l'itération jusqu'à ce que tous les blocs soit chiffrés.
6. Envoyer le chiffré au destinataire

2.1.6.3 Exemple

Dans cette section, nous illustrons le fonctionnement des méthodes proposées.

Soit S : 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 0 0 la chaîne binaire à chiffrer.

Méthode 1

Padding

Diviser S en segments de k = 8 bits, 10110100 10110101 01001101 11110101 01110100. Comme la longueur de la chaîne est un multiple de 8, la chaîne est complétée par 00000000. Le premier bloc P_1 est donc 10110100.

Vecteur d'initialisation (IV)

Supposons que M est la partition de « Jingle bells ».
|EEE-|EEE-|EGCD|E—|FFFF|FEEE|EDDD|ED–|EEE-|EEE-|EGCD|E—|
Les quatre premières notes de jingle bell sont EEE-. La conversion binaire équivalente à partir du tableau de conversion binaire 1 est 010010010111, de sorte que IV = 01001001 et Shift = 0111.

E_k

La partition de Jingle bell est :
EEE-EEE-EGCDE—FFFFFEEEEDDED—EEE-EEE-EGCDE—FFFFFEEEGGFDC—
Le graphe équivalent construit comme expliqué dans la méthode est illustré à la figure 2. À partir du graphe de la figure 2, la séquence de degrés des sommets de C à R est 6, 12, 47, 22, 8, 0, 0, 32. En convertissant en chaîne binaire comme expliqué, on obtient E_k = 10110000.

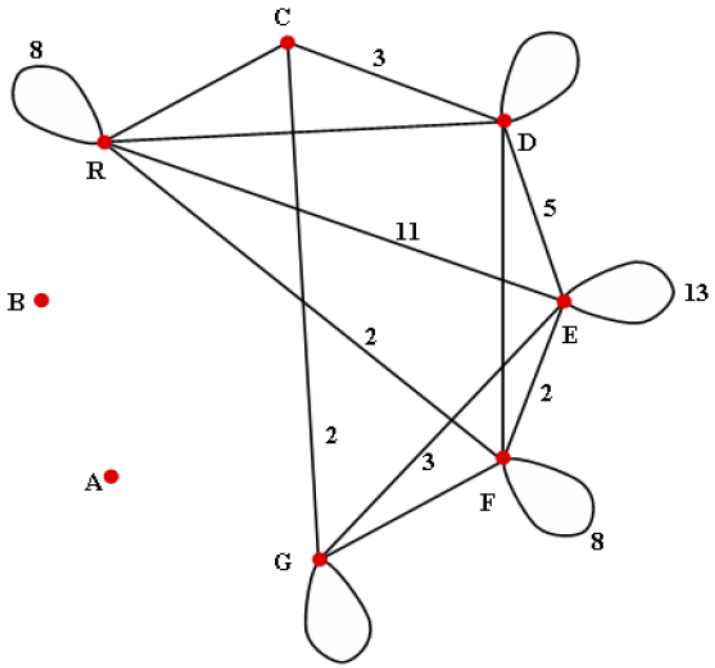


Fig. 2. Graph Equivalent to Jingle Bells

— On a $10110100 \oplus 01001001 = 11111101$

— $E_k(P_1 \oplus (P_0 \oplus C_0))$: $11111101 \oplus 10110000 = 01001101$ (c'est le texte chiffré C_1).

— $P_1 \oplus C_1 = 01001101 \oplus 10110100 = 11111001$.

— $P_2 \oplus (P_1 \oplus C_1) = 10110101 \oplus 11111001 = 01001100$.

— Il y aura un décalage vers la droite de 7 (décalage = 0111) positions dans la chaîne de la fonction E_k. Ainsi, la nouvelle clé dans E_k devient 01100001.

— $E_k(P_2 \oplus (P_1 \oplus C_1))$: $01001100 \oplus 01100001 = 00101101$ (le texte chiffré C_2)

— $C_2 \oplus P_2 = 00101101 \oplus 10110101 = 10011000$ (la IV de la prochaine itération)

— $01001101 \oplus 10011000 = 11010101$

— $11010101 \oplus 11000010 = 00010111$ (C_3)

— $01001101 \oplus 00010111 = 01000111$ ($P_3 \oplus C_3$)

— $11110101 \oplus 01000111 = 10110010$

— $10110010 \oplus 10000101 = 00110111$ (C_4)

— $11110101 \oplus 00110111 = 11000010$

— $01110100 \oplus 11000010 = 10110110$

— $10110110 \oplus 00001011 = 10111101$ (C_5)

— $10111101 \oplus 01110100 = 11001001$

— $00000000 \oplus 11001001 = 11001001$

— $11001001 \oplus 00010110 = 11011111$ (C_{pad})

— Le message envoyé après la concaténation est : 0100110100101101000101110011011110111011101111

Méthode 2

Le padding et la valeur IV sont identiques à ceux de la méthode 1. Le graphe dirigé correspondant est donné dans la figure 3. Dans le digraphe G de la figure 3, une arête représente qu'il y a au moins une arête entre les sommets correspondants. Une arête bidirectionnelle entre x et y signifie qu'il y a au moins une arête dirigée de x vers y et de y vers x.

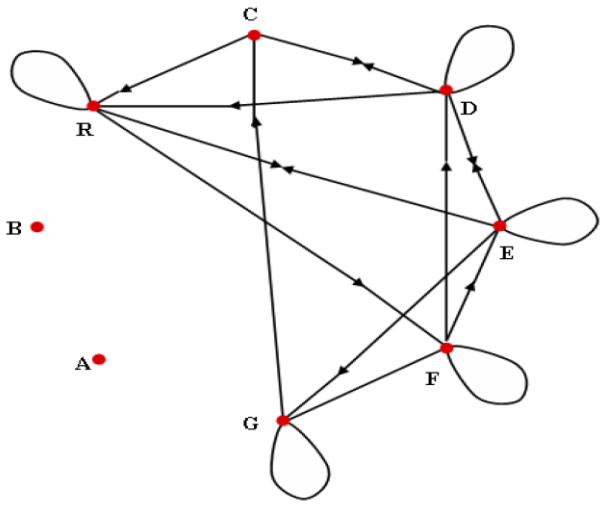


Fig. 3. Directed Graph for Jingle Bells

La matrice d'adjacence associée au graphe de la figure 3 est définie comme suit :

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

Chaque ligne représente notre fonction E_k qui sera utilisée dans l'algorithme

— $10110100 \oplus 01001001 = 11111101$ $(P_1 \oplus IV)$

— $11111101 \oplus 01000001 = 01111100$ $(P_1 \oplus IV \oplus$ (la ligne 1 de A) $= C_1)$

— $01111100 \oplus 10110100 = 11001000$ $(P_1 \oplus C_1)$

— $10110100 \oplus 11001000 = 01111100$ $(P_2 \oplus C_1 \oplus P_2)$

— $01111100 \oplus 11100001 = 10011101$ $(P_1 \oplus C_1 \oplus P_2 \oplus$ (la ligne 2 de A) $= C_2)$

— $10011101 \oplus 10110100 = 00101001$ $(P_2 \oplus C_2)$

— $01001101 \oplus 00101001 = 01100100$ $(P_3 \oplus P_2 \oplus C_2)$

— $01100100 \oplus 01101001 = 00001101$ $(P_3 \oplus P_2 \oplus C_2 \oplus$ (la ligne 3 de A) $= C_3)$

— 01001101 $\oplus$ 00001101 = 01000000 ($P_3 \oplus C_3$)

— 00000000 $\oplus$ 01000000 = 01000000 (pad $\oplus$ $P_3 \oplus C_3$)

— 01000000 01110000 $\oplus$ = 01110000 (pad $\oplus$ $P_3 \oplus C_3 \oplus$ (la ligne 4 de A) = C_4)

Le message envoyé après la concaténation est : 011111001001110100001101011110000.

Conclusion

Nous avons présenté dans ce chapitre quelques travaux de l'état de l'art qui tirent profit des principes de la théorie des graphes pour développer de nouveaux systèmes cryptographiques résistants aux différents types d'attaques pouvant être menées. Par la suite, nous avons détaillé les différentes techniques et méthodes utilisées par les algorithmes de chiffrement qui se basent sur la théorie des graphes à travers des exemples d'illustrations du processus de chiffrement et déchiffrement.

Chapitre

3 Chiffrement BC-GT : A new block cipher system using graph theory

Sommaire

Introduction

La communication des données privées dans un canal non sécurisé est très critique, car les entités non autorisées peuvent les intercepter. Dans notre travail, nous nous sommes intéressés au concept du chiffrement par bloc qui est l'une des solutions cryptographiques principales permettant d'assurer la confidentialité des données. Cette dernière est un mécanisme où les données transmises ne doivent être compréhensibles que par entités autorisées [Menezes *et al.*, 2018]. Dans ce contexte, un certain nombre de solutions cryptographiques ont été développées en utilisant différents modèles qui rendent ces algorithmes parfois vulnérables. Ainsi, les chercheurs se concentrent sur la sécurité de la conception des systèmes résistants contre les différentes attaques existantes. Dans ce chapitre, nous présentons un nouveau système de chiffrement par blocs [Bekkaoui *et al.*, 2020a], appelé A new block cipher system using graph theory (**BC-GT**), qui répond parfaitement aux exigences de sécurité. Le schéma est basé essentiellement sur les propriétés de la théorie des graphes notamment les circuits hamiltoniens disjoints qui sont très prometteurs pour la représentation du texte clair, et du principe "Diviser pour régner", pour simplifier les traitements et faciliter le chiffrement.

Notre approche propose une nouvelle utilisation des circuits hamiltoniens et de matrice d'adjacence en utilisant un générateur des sous-clés particulier qui a été soigneusement conçu pour produire les clés de chiffrement conformément aux spécifications du système. Les résultats expérimentaux obtenus démontrent que notre système de chiffrement proposé est robuste contre les attaques statistiques, en particulier le test DIEHARD.

Le chapitre 3 est organisé de la manière suivante : Dans la section 3.1, nous décrivons l'algorithme de chiffrement par bloc proposé. Dans la section 3.2, nous présentons les analyses de sécurité et de performances pour évaluer notre algorithme de chiffrement par bloc BC-GT. Enfin, le bilan du chapitre est présenté dans la section conclusion.

3.1 Description du chiffrement par bloc proposé

Le système proposé met en jeu les concepts fondamentaux de la théorie des graphes afin de faciliter les manipulations sur les données brutes. L'idée de base s'articule sur la génération des graphes pondérés à partir d'un nouvel usage des circuits Hamiltoniens.

Notre contribution vis-à-vis des travaux étudiés est très prometteuse surtout en termes de complexité et de rapidité lors des traitements du message clair. Le régime proposé se base essentiellement sur la notion "Diviser pour régner", qui consiste à diviser un problème initial en sous-problèmes de petite taille, puis à traiter chaque sous-problème à part.

Notre approche est conçue de manière à répondre principalement à la complexité, sachant que la majorité des travaux de littérature a représenté le message clair par un graphe de taille similaire, et le graphe à son tour devenait une matrice d'adjacence avec laquelle le traitement et la manipulation des données sont effectués. Donc, lorsque la matrice correspondant au message sera consistante, les opérations linéaires adjointes deviendront

automatiquement très complexes.

Toutefois, l'application du mécanisme "diviser pour régner" nous a permis de minimiser le problème de complexité, et au lieu de traiter le message tout entier, on le divise en blocs de petite taille. Chaque bloc est représenté par des circuits Hamiltoniens disjoints en utilisant le théorème 3.2, servant à diminuer la taille du graphe associé au bloc.

Théorème 3.2.

Dans un graphe complet avec n sommets, il existe $(n-1)/2$ circuits Hamiltoniens disjoints, si n est un nombre impair supérieur strictement à 3 [Deo, 2017].

La décomposition du message en plusieurs blocs se fait à l'aide de la formule suivante :

$$n = 25k + r \tag{3.1}$$

Tel que n est la taille du message, r ($r \in [0, 24]$) est le reste de la division du message par 25, et k est le quotient. Si la division est exacte, on obtient $k' = k$ blocs de taille 25, sinon on aura $k' = k + 1$ blocs de taille 25.

Sachant qu'un graphe d'ordre 13 contient 6 circuits Hamiltoniens disjoints, il y aura donc plusieurs possibilités pour représenter les sous blocs.

Par ailleurs, chaque bloc de taille 25 (Taille minimale d'un bloc) est partitionné en deux sous blocs de taille respectivement 13 et 12, le sous bloc d'une taille impaire est convertie en cycle eulérien en utilisant les valeurs ASCII des caractères qui le composent, représentant ainsi les poids des arrêtes. Concernant le deuxième sous bloc, il est représenté par un circuit Hamiltonien parmi les cinq autres en complétant les valeurs manquantes d'une façon aléatoire. Le graphe qui résulte est converti en matrice d'adjacence (bloc). Cette représentation est très avantageuse, non seulement au niveau de la complexité du traitement, mais aussi par rapport à la représentation classique du message qui s'effectuait au sein d'un seul cycle eulérien.

Le chiffrement s'effectue par bloc à l'aide du mode CBC (Cipher Block Chaining) en utilisant notre propre générateur des pseudo-clés qui se base sur le générateur pseudo-aléatoire BBS. Dans ce qui suit, nous détaillons le processus de générations des clés, les systèmes de chiffrement et de déchiffrement.

3.1.1 Algorithme de génération des sous clés

La génération des pseudo-clés se produit en deux étapes, la première concerne la génération d'une clé Key de taille m $= 13k'$ à partir d'une clé mère KEK (Key Encryption Key) de taille m=256. Ensuite, à partir de Key on procède à la génération de k' pseudo-clés sous forme de matrices carrées d'ordre 13.

Pour chaque bloc $Block_i$, on utilise la valeur ASCII du premier caractère d'une part pour préciser un nombre de la clé mère KEK (à l'aide de la position qui est censé être dans

$[0, 255]$). Et d'autre part pour construire le vecteur FCB nécessaire pour le déchiffrement. Ce chiffre sera le germe pour générer un vecteur S_i de taille 13. Le vecteur S_i est utilisé pour générer la pseudo-clé K_i sous forme de matrices carrées d'ordre 13.

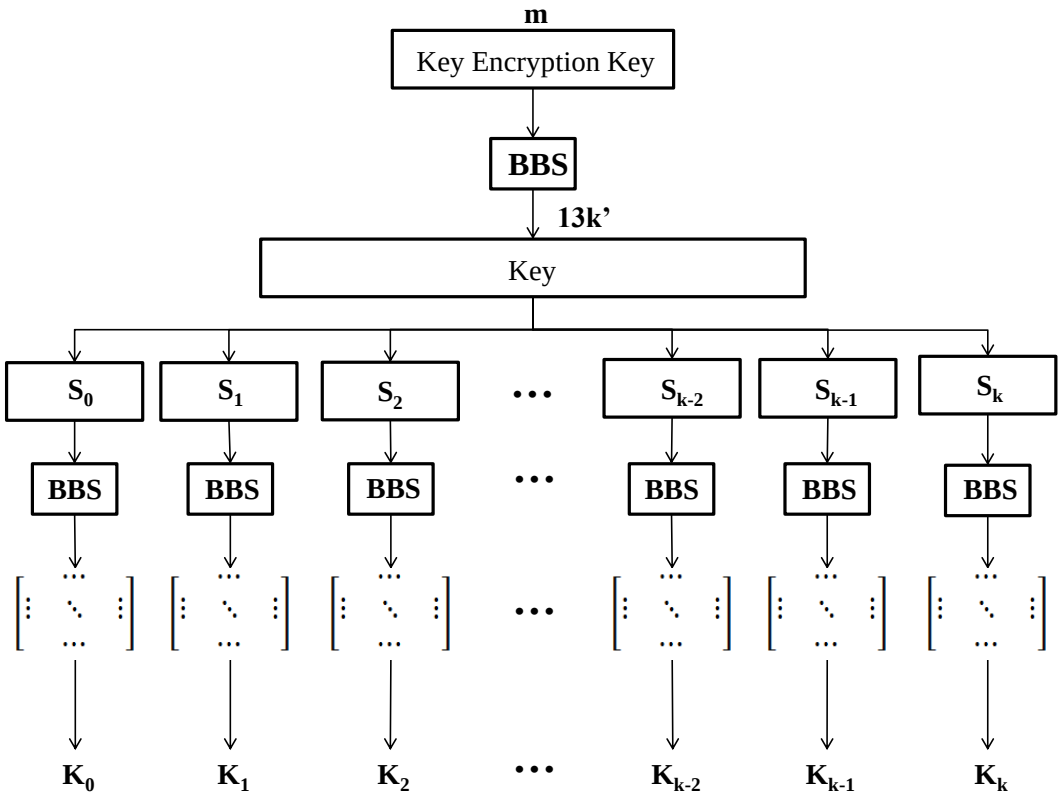


FIGURE 3.1 – Génération des sous clés

3.1.2 Processus de chiffrement

Nous chiffrons chaque bloc en utilisant le mode CBC (Cipher Block Chaining). Dans ce mode, on applique sur chaque bloc un "OU exclusif" (XOR) avec le chiffré du bloc précédent avant qu'il soit lui-même chiffré de la même manière. L'opérateur binaire XOR concerne dans notre cas la matrice M_i et la matrice C_{i-1} comme suit :

$$M'_i = C_{i-1} \oplus M_i \tag{3.2}$$

Le résultat est par la suite utilisé dans une autre opération XOR avec la pseudo-clé K_i produite depuis le générateur pseudo-aléatoire BBS pour avoir le chiffré C_i du bloc courant.

$$C_i = M'_i \oplus K_i \tag{3.3}$$

Le chiffrement du premier bloc M_0 s'effectue après l'application du "OU exclusif" avec une matrice d'initialisation MI générée d'une façon aléatoire.

$$\begin{cases} M'_0 = MI \oplus M_0 \\ C_0 = M'_0 \oplus K_0 \end{cases} \tag{3.4}$$

Chaque matrice résultante C_i (où i $\in [0, k'-1]$) est transformée en vecteur en concaténant les lignes des matrices C_i pour former un seul vecteur de taille $13^2k'$ qui représentera le message chiffré EM renvoyé, en plus du vecteur FCB contenant les positions utilisées pour générer les pseudo-clés. La FIGURE 3.2 illustre clairement le processus de chiffrement.

Algorithme 3.1 Chiffrement : BC-GT

input : Clear message of n characters CMC_n, KEK master key, Random square Matrix IM of size 13

output: Encrypted message EM(Vector V of size $169k'$), the vector FCB

begin
- CMA_n=***ASCII_ Transfomation***(CMC_n,n) ;
- **if** $n\%25 == 0$ **then**
  - $k'= n \div 25$;
- **else**
  - $k'= (n \div 25) + 1$;
- **end**
- $BlockSet_{k'}$=***Decomposition_ Block***(CMA_n,k') ;
- $K_{k'}$=***Generate_ Key***(KEK,k',FCB) ;
- **for** *element* $Block_i$ *of the set* $BlockSet_{k'}$ **do**
  - G_i =**Block_Graph**$(Block_i,13)$;
  - M_i = **Graph_Matrix**(G_i) ;
  - **if** $i==0$ **then**
    - $M'_0 = IM \oplus M_0$;
  - **else**
    - $M'_i = C_{i-1} \oplus M_i$;
  - **end**
  - $C_i = M'_i \oplus K_i$;
  - V_i=**Concatenate_Lines**(C_i) ;
- **end**
- EM=**Concatenate_Vec**$(V_{k'})$;

end

3.1.3 Processus de déchiffrement

Généralement, le processus de déchiffrement est l'algorithme de chiffrement exécuté dans l'ordre inverse. Afin de déchiffrer un message, l'algorithme prend en entrée le texte

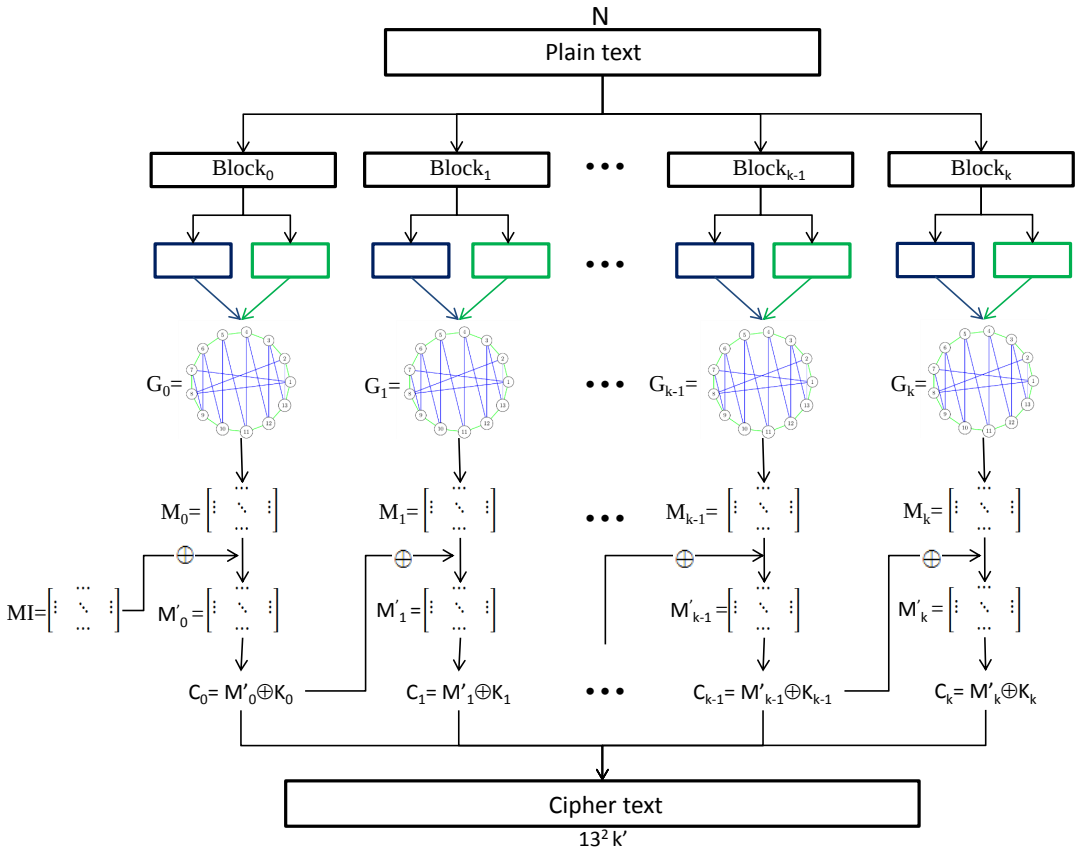


FIGURE 3.2 – Processus de chiffrement

chiffré EM et le vecteur des positions FCB et la clé mère. Le nombre de blocs k' est calculé de la manière suivante :

$$k' = m \div 13^2 \tag{3.5}$$

Chaque bloc $C_i (i = 0 \ldots k' - 1)$ est déchiffré par son propre sous-clés K_i en utilisant la formule :

$$M_i = C_{i-1} \oplus M'_i \tag{3.6}$$

avec

$$M'_i = C_i \oplus K_i \tag{3.7}$$

et

$$M_0 = IM \oplus M'_0 \tag{3.8}$$

Dans cette phase les blocs déchiffrés M_i sont transformés en graphe puis en bloc $Block_i$. Enfin le message clair est formé par la concaténation des blocs $Block_i$ (i $\in [0, k'-1]$) comme montré dans la FIGURE 3.3.

Algorithme 3.2 Déchiffrement : BC-GT

input : Encrypted message EM(Vector V of size $169k'$), KEK master key, the vector FCB

output: Clear message of n characters CMC_n, KEK master key, Random square Matrix IM of size 13

begin

$k' = m \div 13^2$;

$EMatrixSet_{k'}$=***Decomposition_ Vector***(EM,k') ;

$K_{k'}$=***Generate_ Key***(KEK,k',FCB) ;

for *element* C_i *of the set* $EMatrixSet_{k'}$ **do**

$M'_i = C_i \oplus K_i$;

if i==*0* **then**

$M_0 = IM \oplus M'_0$;

else

$M_i = C_{i-1} \oplus M'_i$;

end

G_i=**Matrix_Graph**(M_i) ;

$Block_i$= **Graph_Block**(G_i) ;

end

$BlockSet_{k'}$=**Concatenate_Block**$(Block_{k'},k')$;

CMC_n=**Concatenate_Block**$(BlockSet_{k'},k')$;

end

3.2 Analyse de sécurité et résultats expérimentaux

Dans cette section, nous présentons les analyses de sécurité et de performances pour évaluer notre algorithme de chiffrement par bloc **BC-GT**. Nous commençons par une étude des tests statistiques.

3.2.1 Tests statistiques

Dans cette section, nous analysons la qualité de la production aléatoire du chiffrement par blocs proposés en utilisant le test de DIEHARD [Marsaglia, 1998]. Le but principal de ce test est de prouver que notre algorithme résiste face aux attaques statistiques. Autrement, une sortie de chiffrement par blocs sécurisée devrait être statistiquement indiscernable à partir d'une sortie aléatoire via la fonction de chiffrement.

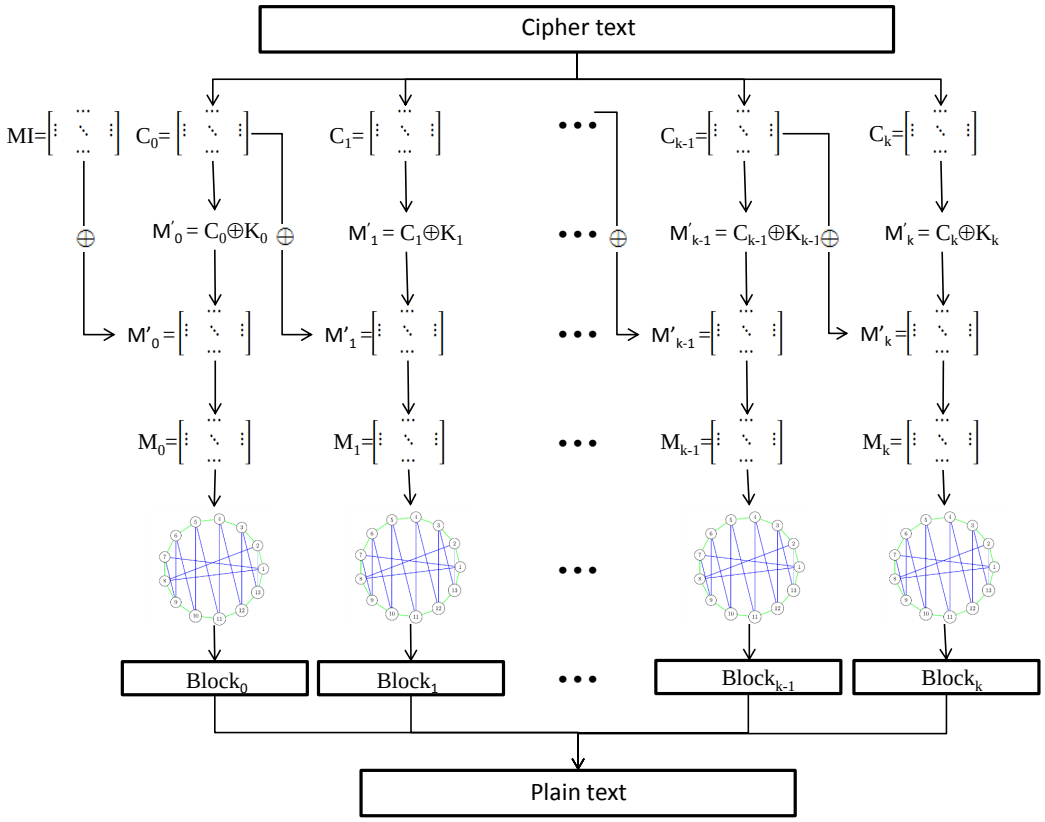


FIGURE 3.3 – Processus de déchiffrement

Pour effectuer ce test, une séquence de chiffrés générés aléatoirement est d'abord convertie en format binaire pour produire un flux de bits de taille supérieure à 10Mo. Ensuite, ce flux est analysé statistiquement en utilisant le test de DIEHARD [Marsaglia, 1998]. Le test de DIEHARD vérifie la valeur p (p-value) des chiffrés aléatoires obtenus, sachant que la valeur de p est compris dans $[0.025, 0.975]$.

Les moyennes des résultats sont résumées dans le tableau 3.1. D'après ces résultats. Le flux binaire généré en utilisant notre proposition a réussi tous les tests de DIEHARD. En outre, notre chiffrement par bloc **BC-GT** fournit un bon comportement aléatoire et statistiquement indiscernable.

3.2.2 Attaque par force brute

L'attaque par force brute est un moyen d'essayer de trouver tous les arrangements clés potentiels en utilisant un outil de prédiction rapide. Compte tenu d'une machine de haute qualité qui prend 10^{-10} Secondes pour tester la validité de la clé. Supposant que les chiffres utilisés dans la clé mère et compris entre 1 et 1000, notre algorithme comporte 100^{256} clés possibles. En essayant l'attaque de force brute, il faudra environ 10^{758} s (10^{-10}

TABLE 3.1 – Test de DIEHARD concernant notre premier crypto-système proposé

Nom de test	p-value	Interprétation
diehard birthdays	0.26375543	PASSED
diehard operm5	0.37541747	PASSED
diehard rank 32x32	0.95699200	PASSED
diehard rank 6x8	0.09885031	PASSED
diehard bitstream	0.93471890	PASSED
diehard opso	0.73703729	PASSED
diehard oqso	0.17764052	PASSED
diehard dna	0.81870913	PASSED
diehard count 1s str	0.97971001	PASSED
diehard count 1s byt	0.49404053	PASSED
diehard parking lot	0.27155245	PASSED
diehard 2dsphere	0.35355239	PASSED
diehard 3dsphere	0.58482092	PASSED
diehard squeeze	0.91687701	PASSED
diehard sums	0.15611362	PASSED
diehard runs	0.64253136	PASSED
diehard craps	0.63765229	PASSED
marsaglia tsang gcd	0.74963931	PASSED
sts monobit	0.92126172	PASSED
sts runs	0.28893885	PASSED
sts serial	0.50145071	PASSED
rgb bitdist	0.69014502	PASSED
rgb minimum distance	0.57112646	PASSED
rgb permutations	0.59422228	PASSED
rgb lagged sum	0.59927829	PASSED
rgb kstest test	0.10026759	PASSED
dab bytedistrib	0.49551450	PASSED
dab dct	0.08738803	PASSED
dab filltree	0.22157233	PASSED
dab filltree2	0.22430630	PASSED
dab monobit2	0.15458308	PASSED

$\times\ 1000^{256}$) pour obtenir la bonne clé secrète. Ainsi, une attaque de force brute avec une recherche de clé exhaustive est impossible dans un temps raisonnable.

Pour révéler un message de 25 caractères c.-à-d. Le cas où un seul bloc est utilisé, il existe 100 possibilités pour déterminer un chiffre de la clé mère, qui représentera le germe du générateur BBS pour la génération du vecteur S_0, or que les valeurs des nombres premiers utilisées comme paramètre d'entrée du générateur sont difficiles à terminer (problème de factorisation). Du fait, il est casé impossible de trouver le pseudo clé tant que pq est grand.

Conclusion

Ce chapitre présente un nouveau système de chiffrement par blocs en utilisant à la fois le mécanisme de "Diviser pour régner" et les concepts fondamentaux de la théorie des graphes afin de faciliter et diminuer la complexité des traitements. De plus notre processus de génération des sous-clés de chiffrement est basé sur le générateur pseudo-

aléatoire BBS, ce qui a permis d'augmenter la robustesse de nos clés. Différents tests statistiques ont été réalisés pour prouver la sécurité de notre algorithme. Tous ces tests ont confirmé la résistance de notre algorithme face aux attaques statistiques.

Chapitre

4 Chiffrement EBC-GT : An enhanced block cipher system using graph theory

Sommaire

Introduction

Le partage de données privées dans un canal non sécurisé est extrêmement critique, car des entités non autorisées peuvent les intercepter et briser leur confidentialité. La conception d'un cryptosystème répondant aux exigences de sécurité en termes de confidentialité, d'intégrité et d'authenticité des données transmises est donc devenue un impératif incontournable. De nombreux travaux ont d'ailleurs été menés dans ce sens. Bien que de nombreux cryptosystèmes aient été proposés dans la littérature publiée, il a été constaté que leur robustesse et leurs performances varient relativement de l'un à l'autre. Adoptant cette réflexion, nous abordons dans ce chapitre le concept de chiffrement par blocs, qui est une solution cryptographique majeure pour garantir la confidentialité, en faisant intervenir les propriétés de la théorie des graphes pour représenter le message en clair.

Notre proposition est en fait un nouveau système de chiffrement par blocs [Bekkaoui *et al.*, 2021] appelé An enhanced block cipher system using graph theory (**EBC-GT**), qui procède en représentant les messages en clair à l'aide de circuits hamiltoniens disjoints, puis en les traitant comme une matrice d'adjacence dans une phase de pré-chiffrement. Le système proposé repose sur un générateur de sous-clés particulier qui a été soigneusement conçu pour produire les clés de chiffrement conformément aux spécifications du système. Les résultats expérimentaux obtenus démontrent que notre système de chiffrement proposé est robuste contre les attaques statistiques, notamment le test DIEHARD, et présente à la fois une bonne confusion et une bonne diffusion.

Le chapitre 4 est organisé de la manière suivante : Dans la section 4.1, nous décrivons l'algorithme de chiffrement par bloc proposé. Dans la section 4.2, nous présentons les analyses de sécurité et de performances pour évaluer notre algorithme de chiffrement par bloc EBC-GT. Enfin, le bilan du chapitre est présenté dans la section conclusion.

4.1 Description du chiffrement par bloc proposé

L'objectif principal du système présenté dans cet article est de proposer une variante robuste du schéma de chiffrement proposé dans [Bekkaoui *et al.*, 2020a] tout en maintenant les niveaux de performance. Le concept principal sur lequel repose notre approche est inspiré de la méthode de conception diviser pour régner, qui consiste à diviser un problème initial en sous-problèmes, puis à traiter indépendamment chaque composant du sous-ensemble résultant. La solution finale du problème initial est alors déduite des solutions trouvées pour les sous-problèmes.

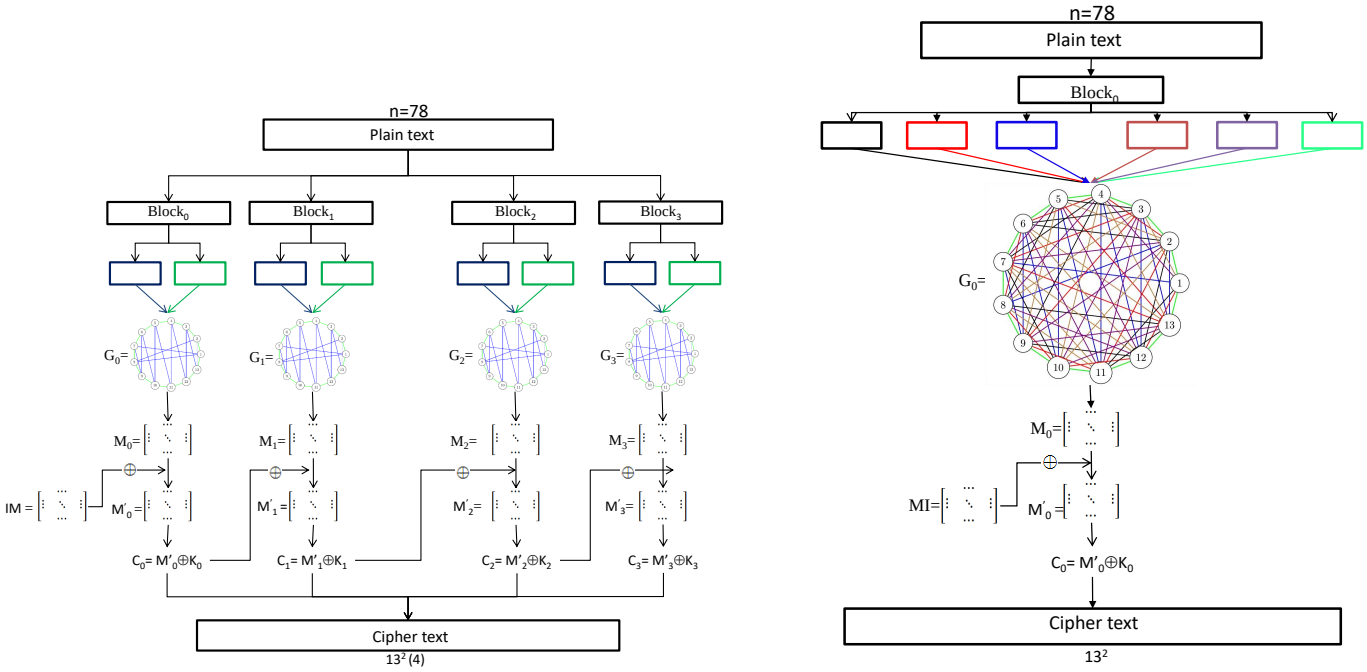
Le système décrit dans ce travail [Bekkaoui *et al.*, 2021] a été conçu de manière à prendre en compte la complexité du traitement que subissent les messages en clair lors de leur chiffrement. En effet, c'est l'objectif de la contribution de ce papier, qui consiste à améliorer le traitement du texte en clair en le rendant plus difficile et plus complexe que [Bekkaoui *et al.*, 2020a], en exploitant principalement tous les circuits hamiltoniens pour représenter le texte en clair.

Le schéma proposé dans [Bekkaoui *et al.*, 2020a] utilisait un bloc de taille 25 caractères, qui peut être représenté par 2 circuits hamiltoniens disjoints dans un graphe d'ordre 13, étant donné qu'un graphe d'ordre 13 contient 6 circuits hamiltoniens disjoints (le théorème 4.3). Contrairement à [Bekkaoui *et al.*, 2020a], qui n'utilisait que 2 des 6 circuits, le concept proposé dans cette approche utilise tous les circuits hamiltoniens disjoints du graphe (6 circuits), ce qui permet de représenter des blocs de 78 caractères en un seul graphe.

Théorème 4.3.

Dans un graphe complet avec n sommets, il existe $(n-1)/2$ circuits Hamiltoniens disjoints, si n est un nombre impair supérieur strictement à 3 [Deo, 2017].

Considérant un message composé de 78 caractères, la formule de partition en blocs dans [Bekkaoui *et al.*, 2020a] serait la suivante : $78 = 25 \times 3 + 3$. Cela signifie que 4 blocs seront transformés en 4 matrices d'adjacence. La formule utilisée dans l'algorithme proposé est limitée à un seul bloc, qui sera à son tour partitionné en 6 sous-blocs pour former un seul graphe avec 6 circuits hamiltoniens disjoints, formant ainsi une seule matrice d'adjacence (FIGURE 4.1 illustre la différence entre les deux systèmes).



(a) (b)

FIGURE 4.1 – Comparaison entre (a) le processus de chiffrement de [Bekkaoui *et al.*, 2020a] et (b) celui proposé dans ce chapitre.

En général, le processus de pré-chiffrement du texte clair se déroule en plusieurs étapes : Tout d'abord, le texte clair est converti en valeurs ASCII et ensuite divisé en plusieurs blocs de taille 78 (référencé par $Block_i$). Cette opération utilise la formule suivante : $n = 78 \times k + r$, où n est la taille du texte clair, r ($r \in [0,77]$) le reste de la division de n sur 78) représente le reste du texte clair après le partitionnement en blocs, et k est le nombre de blocs (se réfère au quotient).

$$\begin{cases} k' = k & \text{Si la division est exacte.} \\ k' = k + 1 & \text{sinon.} \end{cases} \tag{4.1}$$

Où k' représente le nombre total de blocs résultant de la division. Chaque $Block_i$ est partitionné en 6 sous-blocs de taille 13 (chaque sous-bloc est représenté par $subBloc_{ij}$), qui sont ensuite convertis en circuits hamiltoniens où les poids des arêtes du graphe G_i sont représentés par les valeurs ASCII des caractères qui les composent. Enfin, le graphe résultant est converti en une matrice d'adjacence M_i. Dans ce qui suit, nous détaillons le processus de générations des clés, les systèmes de chiffrement et de déchiffrement.

4.1.1 Algorithme de génération des sous clés

La génération de pseudo-clé K_i se produit en quatre étapes, la première concerne la sélection d'un caractère Char de $Block_i$ de façon aléatoire, la deuxième consiste à exploiter la position correspondante à la valeur ASCII du caractère Char d'une part pour construire le vecteur VP nécessaire pour le déchiffrement. Et d'autre part récupérer le nombre N situé dans la même position au niveau de la clé mère KEK qui sera utilisé comme germe S du générateur BBS. La troisième permet de générer un vecteur S_i de taille 13 à partir de germe S. La dernière utilise le vecteur S_i pour générer la pseudo-clé K_i sous forme de matrices carrées d'ordre 13. Les pseudo-clés K_i qui sont générés constituent l'ensemble $SK_{k'}$. Ce processus est illustré par la FIGURE 4.2

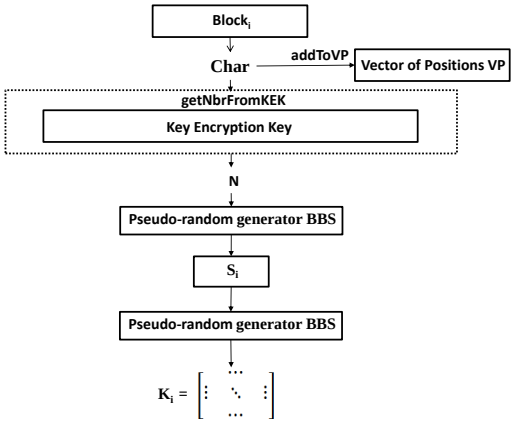


FIGURE 4.2 – Générateur de sous-clés durant le processus de chiffrement.

Algorithme 4.1 Algorithme de génération dessous-clés durant le processus de chiffrement (GenerateSubKeys)

input : Clear message of n characters CMC_n , master key KEK, k' number of blocks
output: sub-keys $SK_{k'}$, the vector VP

begin

```
/* Converts each character of the message into its ASCII value.      */
CMA_n ← convertMessage(CMC_n, n);
   /* Splits the message into k' Block_i forming the set BlockSet_k' ,
   where BlockSet_k' = {Block_0,...,Block_k'-1}.                      */
BlockSet_k' ← parseMessage(CMA_n, k');
 for element Block_i of the set BlockSet_k' do
   /* Randomly selects a character Char from Block_i.                 */
   Char ← getCharFromBlock(Block_i);
      /* Feeds the vector VP with the ASCII value of the character
      Char.                                                           */
   VP ← addToVP(Char);
      /* Returns the content in the position p of the master key
      KEK, where p represents the ASCI code of the character
      concerned.                                                      */
   N ← getNbrFromKEK(KEK, Char);
      /* Generates from the seed N a vector S_i of size 13.          */
   S_i ← generateSeed(N, BBS);
      /* Takes as input the vector S_i and returns the sub-key K_i as a
      square matrix of order 13.                                      */
   K_i ← generateSubKey(S_i, BBS);
      /* Feeds the set SK_k' with the sub-key K_i.                   */
   SK_i ← putSubKey(K_i);
```

end

end

La régénération de K_i pendant le processus de déchiffrement commence par l'utilisation de VP pour recréer une clé Key de taille $13^2k'$ à partir de KEK. Key est ensuite divisée en sous-vecteurs S_i de taille 13 qui sont ensuite utilisés pour générer les sous-clés K_i sous forme de matrices carrées d'ordre 13. Ce processus est décrit dans FIGURE 4.3

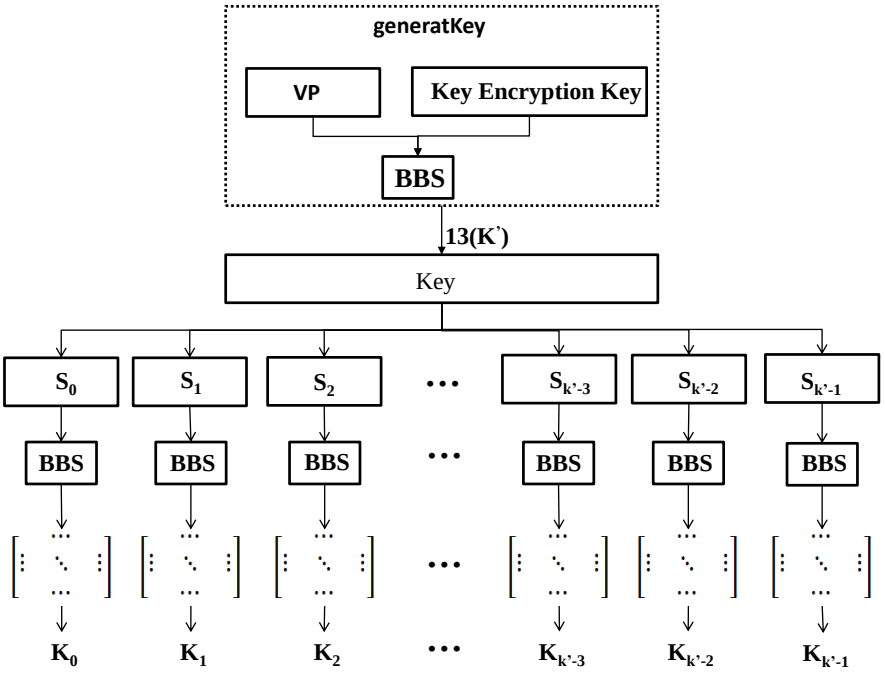


FIGURE 4.3 – Générateur de sous-clés durant le processus de déchiffrement.

Algorithme 4.2 Algorithme de génération des sous-clés durant le processus de déchiffrement (GenerateSubKeys)

input : master key KEK, the vector of positions VP
output: sub-keys $SK_{k'}$

begin
  /* Generates a key Key of size 13k' from the vector of positions VP and the master key KEK. */
  $Key \leftarrow$ ***generateKey***(KEK, VP) ;
  /* Divides the key Key into k' vectors S_i (i = 0,..., k'-1). */
  $S_{k'} \leftarrow$ ***parseKey***(Key) ;
  for *element* S_i *of the set* $S_{k'}$ **do**
    /* Takes as input the vector S_i and returns the sub-key K_i as a square matrix of order 13. */
    $K_i \leftarrow$ **generateSubKey**(S_i) ;
    /* Feeds the set $SK_{k'}$ with the sub-key K_i. */
    $SK_i \leftarrow$ **putSubKey**(K_i) ;
  end
end

4.1.2 Processus de chiffrement

Le processus de chiffrement commence par l'étape de pré-chiffrement décrite ci-dessus. Le mode de chiffrement avec chaînage de blocs (Cipher Block Chaining) est employé comme modes d'opérations par notre système. Le chaînage utilise une méthode de rétroaction, comme le résultat du chiffrement du bloc précédent C_{i-1} est réutilisé pour le chiffrement du bloc courant M_i. Plus précisément, l'opérateur binaire XOR est appliqué entre le bloc actuel M_i et le bloc précédent de texte chiffré C_{i-1} comme indiqué ci-dessous :

$$M'_i = C_{i-1} \oplus M_i \tag{4.2}$$

Nous exécutons ensuite une deuxième opération XOR entre le résultat de l'opération (4.2) et la sous-clé K_i générée par notre générateur pseudo-aléatoire pour calculer le chiffré C_i du bloc courant,

$$C_i = M'_i \oplus K_i \tag{4.3}$$

Pour le tout premier bloc M_0, un bloc ayant un contenu aléatoire, appelé Matrice d'initialisation MI, est générée et utilisée pour l'application de l'opération XOR. Ainsi, chaque bloc chiffré ne dépend pas seulement du bloc de texte en clair correspondant, mais également de tous les blocs chiffrés qui le précèdent.

$$\begin{cases} M'_0 = MI \oplus M_0 \\ C_0 = M'_0 \oplus K_0 \end{cases} \tag{4.4}$$

Les vecteurs résultants $eBlock_i$ (i = 0,. . ., k'-1) générés à partir de tous les blocs sont ensuite concaténés pour former un vecteur unique EM de taille $13^2k'$.

Le processus de chiffrement, comme illustré dans FIGURE 4.4, se termine par la transmission du message chiffré EM en plus du vecteur VP qui est nécessaire pour le processus de déchiffrement.

Algorithme 4.3 Chiffrement : EBC-GT

input : Clear message of n characters CMC_n, master key KEK, Initialization Matrix IM of size 13

output: Encrypted message EM

begin

$SK_{k'} \leftarrow$ ***GenerateSubKeys***(CMC_n, KEK, k') ;
(4.1.1).

/* Converts each character of the message into its ASCII value. */

$CMA_n \leftarrow$ ***convertMessage***(CMC_n) ;

/* Splits the message into k' $Block_i$ forming the set $BlockSet_{k'}$, where $BlockSet_{k'}$ = {$Block_0$,...,$Block_{k'-1}$}. */

$BlockSet_{k'} \leftarrow$ ***parseMessage***(CMC_n) ;

for *element* $Block_i$ *of the set* $BlockSet_{k'}$ **do**

/* Divides each block $Block_i$ into six sub-blocks $subBlock_{ij}$ (j = 0,...,5) of size 13, all forming the set $subBlockSet_i$, where $subBlockSet_i$= {$subBlock_{i0}$,...,$subBlock_{i5}$}. */

$subBlockSet_i \leftarrow$ **parseBlock**$(Block_i)$;

/* Converts the sub-blocks into disjoint hamiltonian circuits in a graph G. */

$G_i \leftarrow$ **blockToGraph**$(subBlockSet_i, 13)$;

/* Transforms the graph G_i into an adjacency matrix M_i of order 13. */

$M_i \leftarrow$ **graphToMatrix**(G_i) ;

if *i=0* **then**

$M'_0 \leftarrow IM \oplus M_0$;

else

$M'_i \leftarrow C_{i-1} \oplus M_i$;

end

$C_i \leftarrow M'_i \oplus SK_i$;

/* Concatenates the rows of the matrix C_i to form the vector $eBlock_i$ of size 13^2. */

$eBlock_i \leftarrow$ **transformMatixToVector**(C_i) ;

end

/* Forms a single vector EM of size $13^2k'$ by concatenating the resulting vectors $eBlock_i$ (i = 0,..., k'-1). */

$EM \leftarrow$ **concatenateEncryptionBlock**$(eBlock_{k'})$;

end

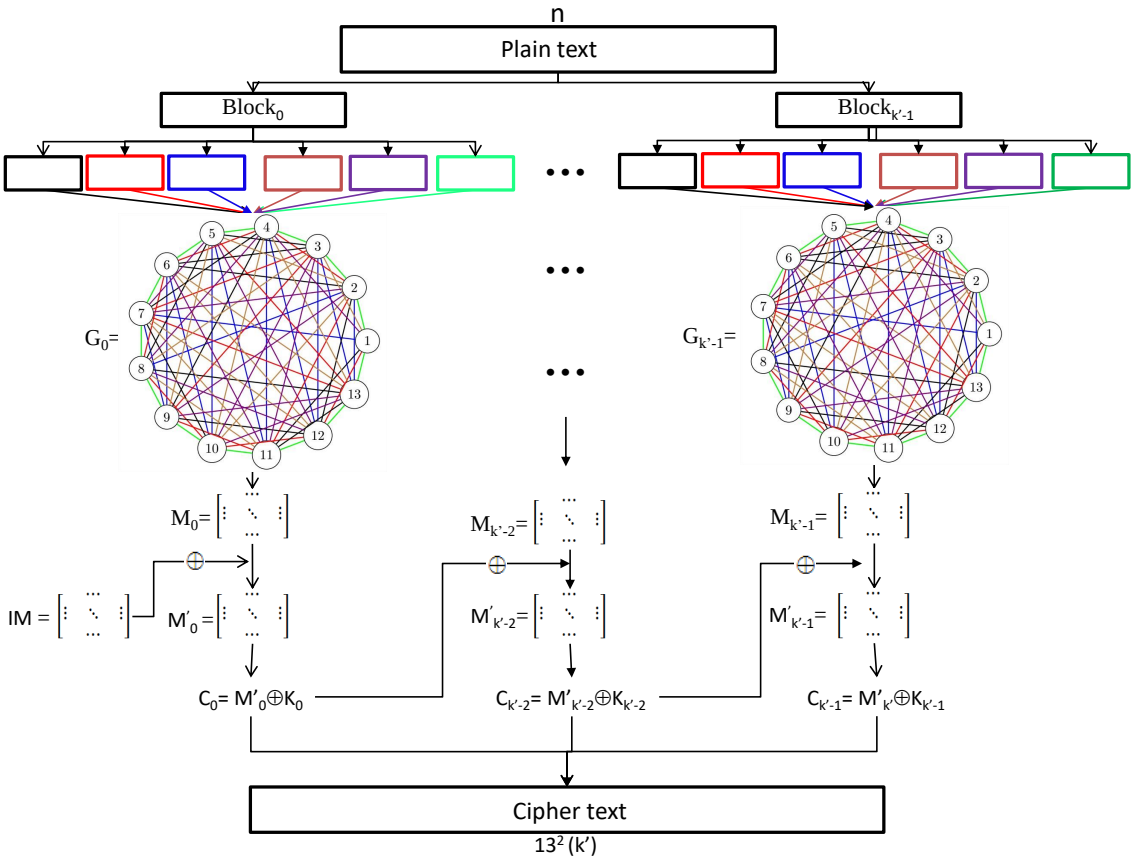


FIGURE 4.4 – Processus de chiffrement

4.1.3 Processus de déchiffrement

En général, le processus de déchiffrement correspond au processus de chiffrement exécuté dans l'ordre inverse (ALGORITHME 4.4).

Lors du processus de déchiffrement décrit dans ce chapitre, le texte chiffré EM correspond à l'entrée de l'algorithme. EM est décomposé en k' vecteurs ($eBlock_i$) qui sont ensuite rassemblés pour former l'ensemble $eBlockSet_k$. Les $eBlock_i$ (i = 0,. . ., k'-1) sont à leur tour converti en matrices C_i. Le nombre de blocs k' est calculé comme suit :

$$k' = m \div 13^2 \tag{4.5}$$

Où m est la taille du texte chiffré.

L'algorithme de génération de sous-clés présenté dans ALGORITHME 4.2 utilise le vecteur fourni VP pour produire une clé de taille $13^2k'$ à partir de la clé mère KEK. Chaque bloc C_i(i = 0,. . ., k'-1) est déchiffré par sa propre clé K_i en utilisant la formule suivante :

$$M_i = C_{i-1} \oplus M'_i \tag{4.6}$$

avec

$$M'_i = C_i \oplus K_i \tag{4.7}$$

et

$$M_0 = IM \oplus M'_0 \tag{4.8}$$

À ce stade, les blocs déchiffrés M_i sont transformés en un graphe G_i puis en $Block_i$. Enfin, le message en clair est formé par la concaténation des $Block_i$ (i = 0,. . ., k'-1) comme indiqué dans la FIGURE 4.5.

Algorithme 4.4 Déchiffrement : EBC-GT

input : Encrypted message EM, master key KEK, the vector of positions VP
output: Clear message of n characters CMC_n

begin
  $SK_{k'} \leftarrow$ ***GenerateSubKeys***(KEK, VP) ;
    (4.1.1).
    /* Divides the encrypted message into k' $eBlock_i$ wich are then concatenate to form a set $eBlockSet_{k'}$. */
  $eBlockSet_{k'} \leftarrow$ ***parseEncryptedMessage***(EM) ;
  for *element* $eBlock_i$ *of the set* $eBlockSet_{k'}$ **do**
    /* Form the matrix C_i of order 13 from the vector $eBlock_i$. */
    $C_i \leftarrow$ **transformVectorToMatrix**$(eBlock_i)$
    $M'_i \leftarrow C_i \oplus K_i$;
    if $i = 0$ **then**
      $M_0 \leftarrow IM \oplus M'_0$;
    else
      $M_i \leftarrow C_{i-1} \oplus M'_i$;
    end
    /* Transforms the adjacency matrix M_i into a graph G_i. */
    $G_i \leftarrow$ **matrixToGraph**(M_i) ;
      /* Returns the $Block_i$ represented by the disjoint hamiltonian circuits inside the graph G_i. */
    $Block_i \leftarrow$ **graphToBlock**(G_i) ;
  end
  /* Forms a single block that forms the plaintext message by concatenating the resulting blocks $Block_i$ (i = 0,..., k'-1). */
  $CMC_n \leftarrow$ **concatenateBlock**$(Block_{k'})$;
end

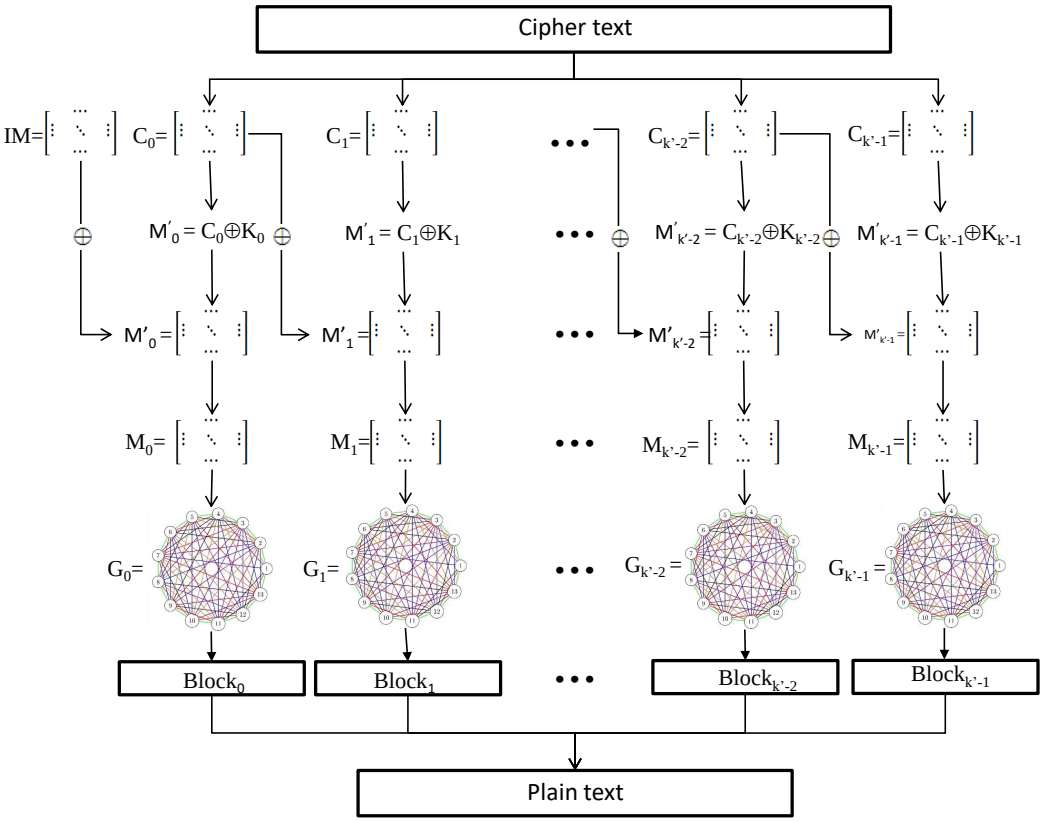


FIGURE 4.5 – Processus de déchiffrement

4.2 Analyse de sécurité et résultats expérimentaux

Dans cette section, nous présentons les analyses de sécurité et de performances pour évaluer notre algorithme de chiffrement par bloc **EBC-GT**. Nous commençons par une étude du test de confusion et du test de diffusion.

4.2.1 Tests de confusion et de diffusion

La confusion et la diffusion sont deux propriétés importantes qui devraient satisfaire un algorithme de chiffrement par bloc. Afin de vérifier ces propriétés, une expression mathématique décrite à l'équation 4.10 est utilisée, permettant de calculer le taux du changement d'un ou plusieurs bits dans le message clair ou dans la clé. Cette équation est inspirée du critère d'avalanche, est une propriété recherchée dans tout algorithme de chiffrement ou fonctions de hachage, il signifie qu'une petite modification du texte clair ou de la clé devrait générer des changements importants dans le texte chiffré. Cette propriété mesure l'impact de changer un ou plusieurs bits en entrée sur la sortie [Castro *et al.*, 2005].

$$bits_{diff} = (1 \div (13^2 \times 16) \times w(C \oplus C') \tag{4.9}$$
$$= (1 \div (2704) \times w(C \oplus C') \tag{4.10}$$

Où w est le poids de Hamming, C et C' sont respectivement les entrées originales et modifiées, et la valeur 16 fait référence au nombre de bits représentant chaque élément du message chiffré.

4.2.1.1 *Test de diffusion*

La propriété de diffusion vise à produire un effet d'avalanche [Castro *et al.*, 2005] entre le message clair et le message chiffré. Le test de sensibilité du changement de bits dans le texte clair permet de vérifier la propriété de diffusion d'un algorithme donné. Étant donné des paires de clairs et de clés secrètes, nous générons le texte chiffré correspondant à chaque paire (clair, clé secrète) à travers notre algorithme EBC-GT, en modifiant un ou plusieurs bits dans le clair généré de manière aléatoire, en gardant la clé inchangée. Par la suite, nous calculons la moyenne du pourcentage de la différence de bits par l'équation 4.10 comme le montre la FIGURE 4.6, plus de 50% des bits dans le texte chiffré sont modifiés. En particulier, la moyenne du pourcentage de différence de bits est comprise entre 48.16% et 51% pour notre système de chiffrement (EBC-GT) et entre 47.1% et 50.80% pour AES-128. Ces pourcentages ont prouvé que notre algorithme EBC-GT offre une bonne diffusion par rapport à AES-128.

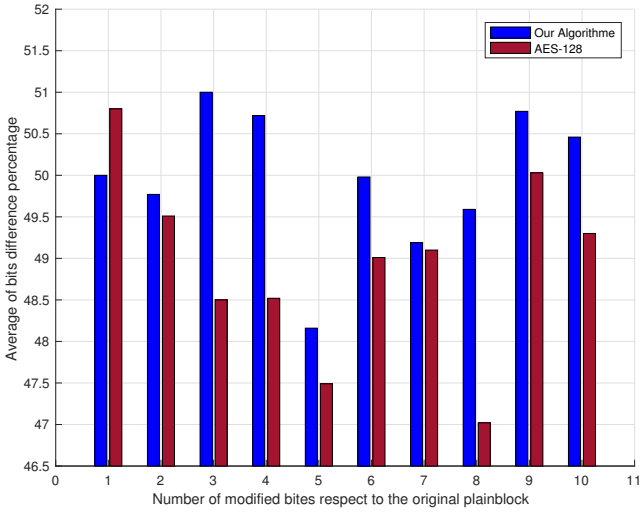


FIGURE 4.6 – Nombre de bits modifiés par rapport au texte clair original

4.2.1.2 Test de confusion

La propriété de confusion crée une relation entre la clé, le clair et le chiffré. Le test de sensibilité de la clé permet d'assurer cette propriété. En effet, nous considérons un ensemble de paires de clairs et de clés secrètes, puis chaque paire est chiffrée en utilisant l'algorithme EBC-GT proposé. Ensuite, nous modifions un ou plusieurs bits dans les différentes clés générées de manière aléatoire, en gardant cette fois ci le clair fixe. Par la suite, nous calculons la moyenne du pourcentage de différence de bits en appliquant l'équation 4.10.

La FIGURE 4.7 représente les résultats obtenus en utilisant le chiffrement EBC-GT et notre générateur pour générer les clés de chiffrement. Elle montre que plus de 50% des bits dans le texte chiffré sont modifiés. Précisément, le pourcentage moyen de différence de bits est compris entre 49.64% et 50.79% pour notre système de chiffrement (EBC-GT), entre 48.25% et 50.73% pour AES-128. Par conséquent, la génération des clés via notre l'algorithme est robuste par rapport à AES-128.

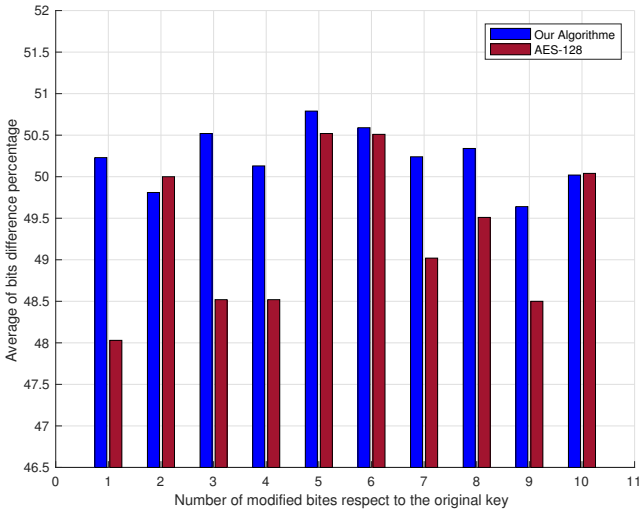


FIGURE 4.7 – Nombre de bits modifiés par rapport à la clé originale

4.2.2 Tests statistiques

Dans cette section, nous analysons la qualité de la production aléatoire du chiffrement par blocs proposés en utilisant le test de DIEHARD [Marsaglia, 1998]. Le but principal de ce test est de prouver que notre algorithme résiste face aux attaques statistiques. Autrement, une sortie de chiffrement par blocs sécurisée devrait être statistiquement indiscernable à partir d'une sortie aléatoire via la fonction de chiffrement.

Pour effectuer ce test, une séquence de chiffrés générés aléatoirement est d'abord convertie en format binaire pour produire un flux de bits de taille supérieure à

10Mo. Ensuite, ce flux est analysé statistiquement en utilisant le test de DIEHARD [Marsaglia, 1998]. Le test de DIEHARD vérifie la valeur p (p-value) des chiffrés aléatoires obtenus, sachant que la valeur de p est compris dans $[0.025, 0.975]$.

Les moyennes des résultats sont résumées dans le TABLEAU 4.1. D'après ces résultats. Le flux binaire généré en utilisant notre proposition a réussi tous les tests de DIEHARD. En outre, notre chiffrement par bloc **EBC-GT** fournit un bon comportement aléatoire et statistiquement indiscernable.

TABLE 4.1 – Test de DIEHARD concernant notre deuxième crypto-système proposé

Test Name	P-value	Interpretation
diehard bitstream	0.59537390	
diehard squeeze	0.97442749	
diehard sums	0.11133210	
diehard count 1s str	0.60934773	
diehard count 1s byt	0.78478421	
diehard parking lot	0.55915630	
diehard birthdays	0.03222200	
diehard operm5	0.75636037	
diehard oqso	0.33566335	
diehard dna	0.45051943	
diehard 2dsphere	0.53656799	
diehard 3dsphere	0.62980562	
diehard rank 32x32	0.40775458	
diehard rank 6x8	0.45554634	
diehard opso	0.44037399	
diehard runs	0.86351847	PASSED
diehard craps	0.15275419	
rgb bitdist	0.69014502	
rgb minimum distance	0.57113046	
rgb permutations	0.60422228	
rgb lagged sum	0.60927830	
rgb kstest test	0.26054914	
dab bytedistrib	0.68169231	
dab dct	0.25149694	
dab filltree	0.88848873	
dab filltree2	0.29185197	
dab monobit2	0.74899931	
sts monobit	0.68441660	
sts runs	0.37246909	
sts serial	0.50145101	
marsaglia tsang gcd	0.47467308	

4.2.3 Attaque par force brute

L'attaque par force brute est un moyen d'essayer de trouver tous les arrangements clés potentiels en utilisant un outil de prédiction rapide. Compte tenu d'une machine de

haute qualité qui prend 10^{-10} Secondes pour tester la validité de la clé. Supposant que les chiffres utilisés dans la clé mère et compris entre 1 et 1000, notre algorithme comporte 1000^{256} clés possibles. En essayant l'attaque de force brute, il faudra environ 10^{758} s ($10^{-10} \times 1000^{256}$) pour obtenir la bonne clé secrète. Ainsi, une attaque de force brute avec une recherche de clé exhaustive est impossible dans un temps raisonnable.

Pour révéler un message de 78 caractères c.-à-d. le cas où un seul bloc est utilisé, il existe 1000 possibilités pour déterminer un chiffre de la clé mère, qui représentera le germe du générateur BBS pour la génération du vecteur S_0, or que les valeurs des nombres premiers utilisées comme paramètre d'entrée du générateur sont difficiles à terminer (problème de factorisation). Du fait, il est casé impossible de trouver le pseudo clé tant que pq est grand.

4.2.4 Tests de performances et comparaison

Le tableau 4.2 représente le test de performance de notre cryptosystème, comparé à d'autres chiffrements par blocs connus tels que le triple DES [Coppersmith *et al.*, 1996] et l'AES [Rijmen et Daemen, 2001] en termes de consommation de temps CPU. Les calculs sont effectués sur un ordinateur équipé d'un processeur Intel Core i7-6600U, 64-bit OS, 2.81 GHz avec 20 GB de RAM. Il est clair d'après TABLEAU 4.2 que notre algorithme est capable d'obtenir de bons résultats en termes de temps d'exécution par rapport aux autres systèmes de chiffrement standard.

TABLE 4.2 – Encryption time Comparison between our block cipher and others block ciphers using different message size

Message Size (Kilo Byte)	AES (ms)	3DES (ms)	Our encryption algorithm
3	248.07	247.47	4.9
10	951.2	614.9	10.4
20	1972	1096	21.2

Conclusion

Ce chapitre présente un nouveau cryptosystème qui exploite les principes de la théorie des graphes, qui offrent un haut degré de sécurité tout en maintenant les performances du traitement des données. Notre proposition de chiffrement par blocs utilise en particulier les circuits hamiltoniens disjoints qui ont été adoptés pour représenter le texte en clair dans une phase de pré-chiffrement. Le processus utilise un générateur de sous-clés spécifiques

qui a été mis en place pour générer les clés de chiffrement selon les exigences du système proposé.

Nous avons effectué différents tests statistiques, notamment les tests DIEHARD, de confusion et de diffusion pour prouver la sécurité et les performances de notre système de chiffrement. Les résultats des expériences ont prouvé le bon comportement de notre conception proposée en termes de robustesse et de temps CPU par rapport à 3DES et AES.

Conclusion générale et perspectives

Les systèmes cryptographiques sont de plus en plus utilisés pour assurer la sécurité et la confidentialité des données transmises à travers les canaux non sécurisés et pour protéger le secret de la vie privée des individus. La confidentialité est un aspect primordial de la sécurité, elle peut être assurée par le mécanisme de chiffrement, à travers lequel les données deviennent inintelligibles à toute personne non habilitée. En effet, les algorithmes de chiffrement consistent à transformer un message clair en un message chiffré de telle sorte que les personnes habilitées peuvent retrouver le texte clair en effectuant l'opération inverse nommée *Déchiffrement*. Cette dernière opération doit être difficile à réaliser pour les personnes non habilitées. Aujourd'hui, la cryptologie moderne a pu employer un ensemble d'outils mathématiques, ce qui a permis des gains plus importants en matière de performance et d'efficacité. La théorie des graphes est un domaine qui a été considéré très prometteur en ce sens, du fait qu'il fournit des concepts capables de résoudre des problèmes dans tous les domaines liés à la notion de réseau.

La théorie des graphes est devenue un élément clé dans de nombreuses applications dans le domaine informatique. C'est un concept récent qui a été intégré avec succès pour fournir des algorithmes cryptographiques plus puissants et difficiles à casser par n'importe quel logiciel moderne. En effet, il s'agit essentiellement de modéliser des problèmes de chiffrement en les exprimant en graphes de sorte qu'il relève d'un problème de la théorie des graphes que nous savons le plus souvent résoudre. Cela a conduit à une large application des concepts de la théorie des graphes en cryptographie, car de nombreux problèmes NP-difficiles proviennent de cette théorie. Dans cette thèse, nous nous sommes intéressés à la conception et à la réalisation de nouvelles approches basées sur la théorie des graphes, du fait que différentes techniques développées dans la littérature souffrent de certaines lacunes selon la nature des données traitées, la précision, la robustesse et le temps de calcul. L'originalité de notre travail réside dans le fait que les systèmes que nous proposons ont pu combiner à la fois le concept de chiffrement par blocs, qui est une catégorie majeure de la cryptographie symétrique et les propriétés de la théorie des graphes pour atteindre un niveau de sécurité relativement élevé. Dans ce travail, l'objectif principal porte sur l'intégration et l'exploitation des concepts liés à la théorie des graphes.

À travers la première contribution, nous avons présenté un nouveau système de chif-

frement par blocs en utilisant à la fois le mécanisme *Diviser pour régner* et les concepts fondamentaux de la théorie des graphes afin de faciliter et de diminuer la complexité des traitements. De plus, notre processus de génération des sous-clés de chiffrement a été baser sur le générateur pseudo-aléatoire *Blum Blum Shub* (BBS), qui a permis de renforcer la robustesse de nos clés générées. Différents tests statistiques ont été réalisés pour prouver la sécurité de notre algorithme. Tous ces tests ont confirmé la résistance de notre algorithme face aux à ce type d'attaques.

La deuxième contribution réalisée dans cette thèse, a permis de mettre en œuvre une nouvelle variante améliorée du premier système. Notre proposition est en fait un nouveau cryptosystème qui exploite les principes de la théorie des graphes, offrant un bon niveau de sécurité tout en maintenant les performances du traitement des données. Notre algorithme de chiffrement par blocs utilise en particulier les circuits hamiltoniens disjoints qui ont été adoptés pour représenter le texte en clair dans une phase de pré-chiffrement. Le processus qui utilise un générateur de sous-clés spécifiques a été mis en place pour générer les clés de chiffrement en respectant les exigences du système proposé. Pour l'évaluation, nous avons effectué différents tests statistiques, notamment les tests DIEHARD, de confusion et de diffusion pour prouver la sécurité et les performances de notre système de chiffrement. Les résultats des expériences ont confirmé un comportement favorable de notre proposition en termes de robustesse et de temps CPU par rapport à *3DES* et *AES*.

Dans le cadre de nos futurs travaux, nous avons l'intention d'une part d'utiliser le générateur pseudo-aléatoire connu sous le nom de *PSOCA*, qui est fondé principalement sur les automates cellulaires, et d'étudier également d'autres propriétés de la théorie des graphes pour une représentation plus discriminante et robuste des données. D'autre part, nous visons à concevoir un nouveau cryptosystème utilisant le graphe comme clé secrète et le produit *CORONA* comme fonction de chiffrement. Par ailleurs, nous chercherons aussi à développer un nouveau générateur pseudo-aléatoire basé sur la théorie des graphes.

Bibliographie

[Agarwal et Uniyal, 2015] Agarwal, S. et Uniyal, A. S. (2015). Prime weighted graph in cryptographic system for secure communication. *International Journal of Pure and Applied Mathematics*, 105(3):325–338.

[Akl, 2019] Akl, S. G. (2019). The graph is the message : design and analysis of an unconventional cryptographic function. *In From Parallel to Emergent Computing*, pages 425–442. CRC Press.

[Akl, 2020] Akl, S. G. (2020). How to encrypt a graph. *International Journal of Parallel, Emergent and Distributed Systems*, 35(6):668–681.

[Al Etaiwi, 2014] Al Etaiwi, W. M. (2014). Encryption algorithm using graph theory. *Journal of Scientific Research and Reports*, 3:2519–2527.

[Alfred J Menezes et Vanstone, 1996a] Alfred J Menezes, P. C. V. O. et Vanstone, S. A. (1996a). *Block ciphers*. CRC press.

[Alfred J Menezes et Vanstone, 1996b] Alfred J Menezes, P. C. V. O. et Vanstone, S. A. (1996b). *Stream ciphers*. CRC press.

[AMOUNAS, 2016] AMOUNAS, F. (2016). An innovative approach for enhancing the security of amazigh text using graph theory based ecc. *International journal of scientific research in science, engineering and technology*.

[Amudha *et al.*, 2018] Amudha, P., Sagayaraj, A. C. et Sheela, A. S. (2018). An application of graph theory in cryptography. *International Journal of Pure and Applied Mathematics*, 119(13):375–383.

[Aumasson *et al.*, 2013] Aumasson, J.-P., Henzen, L., Meier, W. et Naya-Plasencia, M. (2013). Quark : A lightweight hash. *Journal of cryptology*, 26(2):313–339.

[Beaula et Venugopal, 2020] Beaula, C. et Venugopal, P. (2020). Cryptosystem using double vertex graph. *Indian Journal of Science and Technology*, 13(44):4483–4489.

[Bekkaoui *et al.*, 2020a] Bekkaoui, K., Ziti, S. et Omary, F. (2020a). A robust scheme to improving security of data using graph theory. *International Journal of Advanced Computer Science and Applications*, 11(5).

[Bekkaoui *et al.*, 2020b] BEKKAOUI, K., ZITI, S. et OMARY, F. (2020b). A robust scheme to improving security of data using graph theory. *International Journal of Advanced Computer Science and Applications*, 11(5).

[Bekkaoui *et al.*, 2021] BEKKAOUI, K., ZITI, S. et OMARY, F. (2021). Data security : A new symmetric cryptosystem based on graph theory. *International Journal of Advanced Computer Science and Applications*, 12(9).

[Berge et Berge, 1967] BERGE, C. et BERGE, C. (1967). Graphes et hypergraphes. *Dunod, Paris*.

[Bertoni *et al.*, 2008] BERTONI, G., DAEMEN, J., PEETERS, M. et VAN ASSCHE, G. (2008). *On the indifferentiability of the sponge construction*. In EuroCrypt.

[Bertoni *et al.*, 2013] BERTONI, G., DAEMEN, J., PEETERS, M. et VAN ASSCHE, G. (2013). Keccak. *In Annual international conference on the theory and applications of cryptographic techniques*, pages 313–314. Springer.

[Biryukov et Wagner, 1999] BIRYUKOV, A. et WAGNER, D. (1999). Slide attacks. *In International Workshop on Fast Software*, pages 245–259.

[Blum *et al.*, 1986] BLUM, L., BLUM, M. et SHUB, M. (1986). A simple unpredictable pseudo-random number generator. *SIAM Journal on computing*, 15(2):364–383.

[Blum et Micali, 1984] BLUM, M. et MICALI, S. (1984). How to generate cryptographically strong sequences of pseudorandom bits. *SIAM journal on Computing*, 13(4):850–864.

[Bogdanov *et al.*, 2011] BOGDANOV, A., KNEŽEVIĆ, M., LEANDER, G., TOZ, D., VARICI, K. et VERBAUWHEDE, I. (2011). *SPONGENT : A lightweight hash function*. Springer.

[Boyar, 1989] BOYAR, J. (1989). Inferring sequences produced by pseudo-random number generators. *Journal of the ACM (JACM)*, 36(1):129–141.

[Castro *et al.*, 2005] CASTRO, J. C. H., SIERRA, J. M., SEZNEC, A., IZQUIERDO, A. et RIBAGORDA, A. (2005). The strict avalanche criterion randomness test. *Mathematics and Computers in Simulation*, 68(1):1–7.

[Coppersmith *et al.*, 1996] COPPERSMITH, D., JOHNSON, D. B. et MATYAS, S. M. (1996). A proposed mode for triple-des encryption. *IBM Journal of Research and Development*, 40(2):253–262.

[Deo, 2017] DEO, N. (2017). *Graph theory with applications to engineering and computer science*. Courier Dover Publications.

[Diestel, 2000] DIESTEL, R. (2000). *Graph theory*. Springer.

[Diudea *et al.*, 2001] DIUDEA, M. V., GUTMAN, I. et JANTSCHI, L. (2001). *Molecular topology*. Nova Science Publishers Huntington, NY.

[FIPS, 1980] FIPS, P. (1980). Des modes of operation. *Issued December*, 2:63.

[Guo *et al.*, 2011] GUO, J., PEYRIN, T. et POSCHMANN, A. (2011). *The PHOTON family of lightweight hash functions*. Springer.

[Hashem, 2019] HASHEM, S. H. (2019). Proposal hybrid cbc encryption system to protect e-mail messages. *Iraqi Journal of Science*, 60(2):157–170.

[KBalakrishnan, 1995] KBALAKRISHNAN, V. (1995). *Graph theory*. Mcgraw-hill companies.

[Khaleel et Al-Shumam, 2020] KHALEEL, T. A. et AL-SHUMAM, A. A. (2020). A study of graph theory applications in it security. *Iraqi Journal of Science*, 61(10):2705–2714.

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

[Krawczyk, 1992] KRAWCZYK, H. (1992). How to predict congruential generators. *Journal of algorithms*, 13(4):527–545.

[Labelle, 1981] LABELLE, J. (1981). *Théorie des graphes*. BibliothÃ¨que nationale de Canada.

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

[Meier, 1993] MEIER, W. (1993). *On the security of the IDEA block cipher*. Springer.

[Menezes *et al.*, 1996a] MENEZES, A., VAN OORSCHOT, P. et VANSTONE, S. (1996a). Handbook of. *Applied Cryptography ISBN 0-8493-8523-7*.

[Menezes *et al.*, 1996b] MENEZES, A. J., KATZ, J., VAN OORSCHOT, P. C. et VANSTONE, S. A. (1996b). *Handbook of applied cryptography*. CRC press.

[Menezes *et al.*, 1996c] MENEZES, A. J., van OORSCHOT, P. C. et VANSTONE, S. A. (1996c). *Hash functions and data integrity*. Routledge Handbooks Online.

[Menezes *et al.*, 2018] MENEZES, A. J., VAN OORSCHOT, P. C. et VANSTONE, S. A. (2018). *Handbook of applied cryptography*. CRC press.

[Micali et Schnorr, 1991] MICALI, S. et SCHNORR, C.-P. (1991). Efficient, perfect polynomial random number generators. *Journal of cryptology*, 3(3):157–172.

[Nigel P, 2016] NIGEL P, S. (2016). *Cryptography made simple*. Springer.

[Perera et Wijesiri, 2021] PERERA, P. et WIJESIRI, G. (2021). Encryption and decryption algorithms in symmetric key cryptography using graph theory. *Psychology and Education Journal*, 58(1):3420–3427.

[Rangaswamy et Gurusamy, 2018] RANGASWAMY, K. D. et GURUSAMY, M. (2018). Application of graph theory concepts in computer networks and its suitability for the resource provisioning issues in cloud computing-a review. *J. Comput. Sci.*, 14(2):163–172.

[Rigo, 2009] RIGO, M. (2009). Théorie des graphes. *Université de Liège*.

[Rijmen et Daemen, 2001] RIJMEN, V. et DAEMEN, J. (2001). Advanced encryption standard. *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pages 19–22.

[Rijmen *et al.*, 1997] RIJMEN, V., PRENEEL, B. et DE WIN, E. (1997). *On Weaknesses of Non-surjective Round Functions*. Springer.

[Rukhin *et al.*, 2001] RUKHIN, A., SOTO, J., NECHVATAL, J., SMID, M. et BARKER, E. (2001). A statistical test suite for random and pseudorandom number generators for cryptographic applications. Rapport technique, Booz-allen and hamilton inc mclean va.

[Sakiyama *et al.*, 2016] SAKIYAMA, K., SASAKI, Y. et LI, Y. (2016). *Security of block ciphers : from algorithm design to hardware implementation*. John Wiley & Sons.

[Sensarma et Sarma, 2019] SENSARMA, D. et SARMA, S. S. (2019). Application of graphs in security. *International Journal of Innovative Technology and Exploring Engineering*, 8(10):2273–2279.

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

[Smart, 2016a] SMART, N. P. (2016a). *Block Ciphers and Modes of Operation*. Springer.

[Smart, 2016b] SMART, N. P. (2016b). *Hash functions, message authentication codes and key derivation functions*. Springer.

[Smart, 2016c] SMART, N. P. (2016c). *Moderne bstream ciphers*. Springer.

[Smart, 2016d] SMART, N. P. (2016d). *Modular Arithmetic, Groups, Finite Fields and Probability*. Springer.

[Smart, 2016e] SMART, N. P. (2016e). Modular arithmetic, groups, finite fields and probability. *In Cryptography Made Simple*, pages 3–25. Springer.

[Smart, 2016f] SMART, N. P. (2016f). Modular arithmetic, groups, finite fields and probability. *In Cryptography Made Simple*, pages 3–25. Springer.

[Smart, 2016g] SMART, N. P. (2016g). *The "Naive" RSA algorithm*. Springer.

[Smart, 2016h] SMART, N. P. (2016h). *Public key encryption and signature algorithms*. Springer.

[Thomas, 2015] THOMAS, G. (2015). *Design et Analyse de sécurité pour les constructions en cryptographie symétrique*. Limoges.

[West *et al.*, 2001] WEST, D. B. *et al.* (2001). *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River.

[Xu, 2003] XU, J. (2003). *Theory and application of graphs*, volume 10. Springer Science & Business Media.

[Yamuna *et al.*, 2012] YAMUNA, M., GOGIA, M., SIKKA, A. et KHAN, M. J. H. (2012). Encryption using graph theory and linear algebra. *International Journal of Computer Application*, 5(2):102–107.

[Yamuna *et al.*, 2013a] YAMUNA, M., MUKESH, k. D., TUSHAR, K. et TNUJOY, R. (2013a). Encryption of multiple messages using hamiltonian circuit and time dependent function. *International Journal of Advanced Scientific Research and Technology*, 3:2249–6149.

[Yamuna *et al.*, 2013b] YAMUNA, M., SANKAR, A., RAVICHANDRAN, S. et HARISH, V. (2013b). Encryption of a binary string using music notes and graph theory. *International Journal of Engineering and Technology*, 5(3):2920–2925.

[Yamuna *et al.*, 2014] YAMUNA, M., SUWATHI, A. et KRISHNAN, N. (2014). Four digit pin number as a digraph. *International Journal of Computer Application*, (4):100–107.

[Yousif et Kashmar, 2019] YOUSIF, A. et KASHMAR, A. H. (2019). Key generator to encryption images based on chaotic maps. *Iraqi Journal of Science*, 60(2):362–370.