Master : Cryptologie Et Sécurité Informatique

Rapport

Réalisé par :

ELMNAJJA DINA

ELM'RABTI WAFAE

Professeur :

ESSAIDI NOUREDDINE

Année universitaire :2024/2025

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Dans le monde numérique d'aujourd'hui, où les échanges d'information se font à une vitesse vertigineuse et sur une échelle mondiale, la préservation de l'intégrité des données est devenue une priorité absolue. Cependant, les canaux de communication ne sont pas toujours fiables et peuvent être sujets à des perturbations et des erreurs de transmission. C'est là qu'interviennent les codes de correcteurs d'erreurs, des outils essentiels conçus pour détecter et corriger les erreurs qui se produisent lors de la transmission de données sur des canaux perturbés. Ces codes sont une manifestation tangible de la théorie de l'information, une discipline fondamentale qui étudie la quantification, le stockage et la communication de l'information.

Les codes de correcteurs d'erreurs sont omniprésents dans notre quotidien, jouant un rôle crucial dans une variété d'applications, allant des communications sans fil et des réseaux informatiques aux systèmes de stockage de données et aux transmissions satellitaires.

Cette introduction se penche sur les principes théoriques sous-jacents des codes de correcteurs d'erreurs, explorant leurs mécanismes de fonctionnement, leurs algorithmes de détection et de correction, ainsi que leur impact sur la fiabilité des systèmes de communication numérique. En comprenant la nature complexe et les applications pratiques des codes de correcteurs d'erreurs, nous pouvons mieux apprécier leur importance dans la préservation de l'intégrité des données et dans l'assurance de la stabilité des systèmes de communication modernes.

CHAPITRE 1

FONDEMENTS DE LA CRYPTOGRAPHIE

1.1 Cryptographie

La cryptographie tire son origine du grec "kryptos", signifiant "caché", et de "graphien", qui signifie "écrire". Elle constitue un ensemble de techniques visant à protéger une communication au moyen d'un code graphique secret. La cryptographie englobe l'étude des méthodes permettant l'envoi de messages codés de manière à ce que seul le destinataire puisse les décoder. Le message à transmettre est communément appelé le texte clair, tandis que le message codé ou encrypté est également désigné sous le terme de cryptogramme. Le processus de conversion du texte clair en message codé est appelé chiffrement ou codage, tandis que le processus inverse est désigné sous le nom de déchiffrement ou décodage.

Pour réaliser un codage, il est nécessaire de suivre une méthode précise appelée système cryptographique ou cryptosystème. Ce dernier requiert fréquemment l'utilisation d'une clé. La cryptanalyse, quant à elle, consiste en l'étude des méthodes visant à découvrir le sens d'un message codé, sans avoir connaissance du message original.

1.2 Histoire de la Cryptographie

L'être humain a toujours eu le besoin de cacher des informations. Que ce soit un secret ne devant pas être divulgué dans son entourage qui compromettrait des individus ou encore d'informations tactiques lors des différentes batailles et guerres ayant marqué l'Histoire. Dissimuler des informations a toujours été une nécessité. Voici une liste non exhaustive de différentes techniques utilisées au fil des siècles qui marquent l'évolution de la cryptographie à travers les âges.

Les premières traces de cryptographie remontent à l'Antiquité, plus précisément aux alentours du $XVI^{ème}$ siècle avant J.-C. Un potier en Irak avait gravé sur une

table en argile sa recette en supprimant les consonnes et en modifiant l'orthographe des mots.

Par la suite, entre le $X^{ème}$ et le $VII^{ème}$ siècle avant J.-C., les Grecs utilisaient des scytales, des sortes de bâtons en bois. Quand l'émetteur voulait communiquer, il enroulait une bande de cuir sur la scytale et y inscrivait le message (une lettre par bout de bande). Une fois la bande déroulée, les lettres n'étaient plus ordonnées et n'avaient donc plus aucun sens. Le seul moyen de pouvoir comprendre le message était d'enrouler la bande sur une scytale de même diamètre pour que les lettres puissent s'aligner correctement.

FIGURE 1.1 – une Scytale

Au I^{er} siècle avant J.-C., le cryptage de César faisait son apparition. Ce chiffrement était utilisé par Jules César pour communiquer de façon secrète. C'est un des premiers chiffrements par substitution. Son principe est simple, il suffit de substituer chaque caractère du message d'origine par un autre dans l'alphabet, qui se trouve toujours à une distance fixe.

Au fur et à mesure que le temps passe, comme les méthodes utilisées précédemment deviennent de plus en plus faciles à comprendre et à détourner, d'autres mesures ont dû être prises afin de pouvoir empêcher le décryptage des messages. Au $XVI^{ème}$ siècle, le chiffre de Vigenère fit son apparition. Il s'agit également d'un chiffrement par substitution, mais il est plus avancé que le chiffrage de César. Plutôt que d'utiliser un décalage fixe, le chiffre de Vigenère se base sur une clé qui va déterminer le décalage pour chaque caractère.

A partir du siècle dernier, la cryptographie a joué un rôle clé lors des différentes guerres. La technologie a commencé à être utilisée avec par exemple, les messages radio ou les télégrammes. Les informations pouvaient être beaucoup plus facilement interceptées que dans le passé. Ce fut le cas du ministre des affaires étrangères allemand Arthur Zimmermann, le 16 janvier 1917. Il a envoyé un télégramme à l'ambassadeur allemand au Mexique pour lui proposer un complot afin de garder les Etats-Unis hors de la guerre. Cependant, le message a été intercepté par les Britanniques car les Allemands ont utilisé le « **code diplomatique 0075** » qui était un code de substitutions comprenant 1000 substitutions possibles .

1.3 Machine Enigma

Après la première Guerre Mondiale, la machine Enigma a été créée et les Allemands, ayant compris l'importance que peuvent avoir les informations sensibles, ont investi dans une version militaire plus complexe de cette machine. Toutefois, bien que le fonctionnement de la machine soit complexe, des chercheurs polonais ont étudié le fonctionnement de la machine pour tenter de décrypter les messages. Par la suite, le célèbre mathématicien, cryptologue et informaticien britannique Alan Turing a collaboré au décryptage des messages codés par la machine. Grâce à cela, les Britanniques ont pu déchiffrer des messages, ce qui a été un sérieux avantage qui leur a permis de gagner la guerre. Il a même été estimé que la découverte de Turing a raccourci la deuxième guerre mondiale de deux ans.

En 1977, le standard de chiffrement DES est proposé comme standard par le NIST. C'est un chiffrement par bloc de textes de 56 bit (en réalité 64, mais un bit est utilisé pour contrôler la parité). Il est resté comme standard jusqu'à la fin des années 1990. En effet, à la fin des années 70, les ordinateurs n'étaient pas suffisamment puissants pour déchiffrer un message codé avec DES, mais avec la multiplication de leur puissance, le déchiffrage est devenu possible dans un temps raisonnable. Une information chiffrée en DES peut être déchiffrée en 30h avec 1000 PC en parallèle cadencés à 1GHz. C'est pourquoi, à la fin des années 90, le NIST a lancé un concours pour remplacer le DES et définir un nouveau standard de cryptage qui deviendra AES. Les gagnants sont les deux chercheurs belges Joan Daemen et Vincent Rijmen avec leur chiffrement Rijndael.

A l'époque du DES, plus précisément en 1976, les deux cryptologues Whitfield Diffie et Martin Hellman publient un article « **New Directions in Cryptography** » qui propose une nouvelle façon de crypter les données. C'est à partir de leur article que le cryptage asymétrique est né avec le chiffrement de Diffie-Hellman. Jusque-là, tous les cryptosystèmes étaient symétriques. Bien que ce type de cryptage fonctionne bien, garder la clé secrète ou bien tout simplement la communiquer peut devenir contraignant. Avec leur système, le problème de clé secrète n'est plus un problème car le fonctionnement se base sur une clé pouvant être connue de tout le monde.

1.4 Principes de base de la Cryptographie

Les principes de base de la cryptographie reposent sur des concepts fondamentaux visant à sécuriser les communications et à protéger les données sensibles.

- **Confidentialité :** La cryptographie vise à assurer la confidentialité des informations en les rendant illisibles pour toute personne non autorisée. Cela est généralement réalisé en chiffrant les données à l'aide d'un algorithme et d'une clé de chiffrement, de sorte que seuls les destinataires autorisés disposant de la clé appropriée puissent les déchiffrer.

- **Intégrité :** La cryptographie garantit l'intégrité des données en s'assurant qu'elles n'ont pas été altérées ou modifiées de manière non autorisée pendant la transmission. Cela peut être réalisé en ajoutant une empreinte numérique (hash) aux données ou en utilisant des mécanismes de vérification d'intégrité comme les signatures numériques.

- **Authentification :** La cryptographie permet de s'assurer de l'identité des parties impliquées dans une communication. Cela peut être réalisé en utilisant des certificats numériques et des infrastructures à clé publique (PKI) pour vérifier l'identité des parties et garantir l'authenticité des échanges.

- **Non-répudiation :** La cryptographie peut également être utilisée pour empêcher les parties impliquées dans une communication de nier leur participation ou l'envoi de certaines informations. Les signatures numériques permettent de lier de manière indissociable un message à l'identité de son expéditeur, empêchant ainsi ce dernier de nier l'envoi du message.

- **Disponibilité :** Bien que moins directement lié, un principe important de la cryptographie est d'assurer la disponibilité des données pour les utilisateurs autorisés. Cela signifie garantir que les données sont accessibles lorsque cela est nécessaire et protégées contre les attaques visant à les rendre indisponibles.

1.5 Cryptographie Symétrique et Asymétrique

1.5.1 Cryptographie Symétrique

La cryptographie symétrique (ou cryptographie à clé secrète) est la forme la plus ancienne de cryptographie . Ce chiffrement fonctionne en principe avec une clé secrète, bien qu'il existe certains chiffrements symétriques qui n'utilisent pas de clé,comme par exemple le chiffre de César. Dans le cas des chiffrements avec clé, le principe est le suivant : L'émetteur du message chiffre les données grâce à une clé.Cette clé est généralement une chaîne de caractères. Le message est chiffré et sans la clé il est quasi impossible (le niveau d'impossibilité dépend du niveau de protection du chiffrement utilisé ainsi que de la complexité de la clé utilisée) de retrouver le message d'origine. L'émetteur doit donc transmettre la clé aux personnes à qui il désire transmettre le message s'il veut que son message puisse être lu.

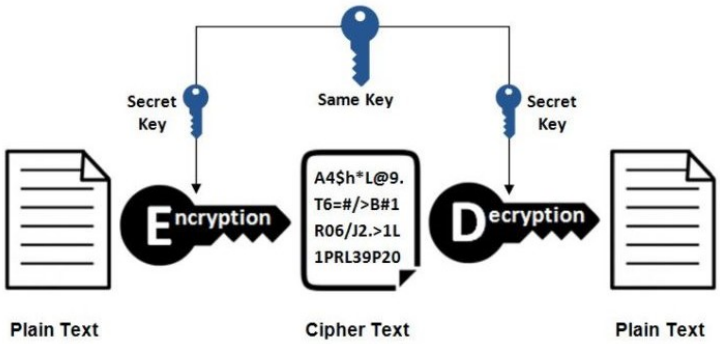


Figure 1.2 – Chiffrement Symétrique

1.5.2 Cryptographie Asymétrique

La cryptographie asymétrique ou cryptographie à clé publique fonctionne de façon totalement différente à la cryptographie symétrique. Si l'on peut comparer la cryptographie symétrique à un coffre-fort auquel seules les personnes possédant la clé peuvent accéder, la cryptographie asymétrique pourrait être comparée à une boite aux lettres dans laquelle on peut déposer des informations, et seule la personne possédant la clé peut accéder au contenu de la boîte. La boite aux lettres serait la clé publique (donc accessible à tout le monde), alors que la clé pour ouvrir la boite serait la clé privée. En effet, dans la cryptographie asymétrique, il y a une clé publique et une clé privée.

Les nombres premier sont l'élément clé pour rendre les algorithmes de cryptographie asymétrique indéchiffrables (ou presque). En prenant comme exemple la multiplication de 5 x 7, il est assez facile de répondre instantanément 35. L'opération inverse, à savoir, retrouver 35 à partir de facteurs premiers est assez facile. Cependant, factoriser 1591 par exemple, est beaucoup plus compliqué, alors que calculer 37 x 43 se fait très facilement. C'est sur cette difficulté de factorisation que se reposent les algorithmes asymétriques.

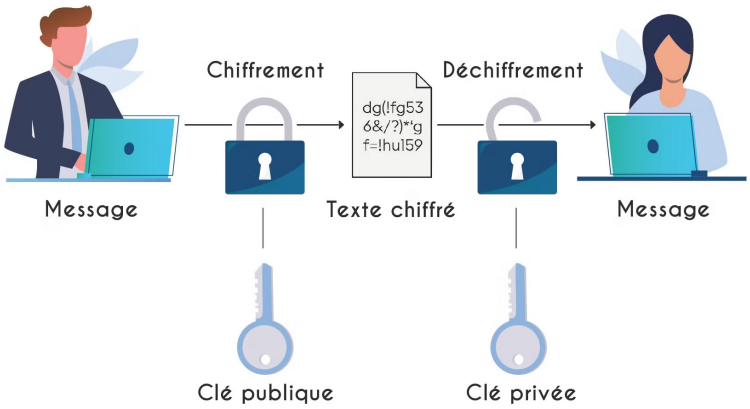


FIGURE 1.3 – Chiffrement Asymétrique

1.5.3 Méthodes de Chiffrement

Chiffrement par flot

Les algorithmes basés sur le principe de chiffrement par flot chiffrent ou déchiffrent un message à la volée. Leur fonctionnement se base sur un générateur de nombres pseudo-aléatoires et un mécanisme de substitution bit à bit. Les algorithmes se basant sur ce principe sont réputés rapides.

Chiffrement par blocs

Le chiffrement par blocs fonctionne différemment. Au lieu de prendre chaque bit un par un, les messages sont découpés en blocs (la taille des blocs dépend de la clé). Ensuite, chaque bloc est additionné à la clé et un traitement de type permutation, opération XOR ou autre est appliqué à chaque bloc.

1.5.4 Algorithmes

Chiffrement de César

Le chiffrement de substitution est un exemple extrêmement simple de cryptographie conventionnelle. Il substitue une information par une autre. Cette opération s'effectue généralement en décalant les lettres de l'alphabet. Le code secret de Jules César est à la base de la cryptographie conventionnelle. Dans ce cas, l'algorithme constitue à décaler les lettres de l'alphabet et la clé correspond au nombre de caractères de décalage. Par exemple, si vous codez le mot « SECRET » à l'aide de la valeur 3 de la clé de César, l'alphabet est décalé de manière à commencer à la lettre D. Ainsi, l'alphabet

ABCDEFGHIJKLMNOPQRSTUVWXYZ

si vous décalez le début de 3 lettres, vous obtenez

DEFGHIJKLMNOPQRSTUVWXYZABC

où D = A, E = B, F = C, etc.

Avec ce procédé, le texte en clair « SECRET » est crypté en « VHFUHW ». Pour autoriser un autre utilisateur à lire le texte chiffré, indiquez-lui que la valeur de la clé est égale à 3. Evidemment, ceci est considéré comme une cryptographie extrêmement vulnérable de par les standards actuels. Mais, cette méthode convenait à César et illustre le mode de fonctionnement de la cryptographie conventionnelle.

Principe de Kerckhoffs

En 1883, le cryptologue hollandais Auguste Kerckhoffs publiait un essai intitulé « La cryptologie militaire ». Cet essai présente une liste des six règles à respecter en cryptographie afin d'assurer un système confidentiel. Bien que ce soit des règles anciennes (elles font par exemple référence à la correspondance par télégramme), elles sont toujours applicables de nos jours.

- Le système doit être matériellement, sinon mathématiquement, indéchiffrable ;
- Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;
- La clé doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants ;
- Il faut qu'il soit applicable à la correspondance télégraphique ;
- Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes ;
- Enfin, il est nécessaire, vu les circonstances qui en commandent l'application,que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.

Chiffrement de Vernam

Le chiffre de Vernam ou encore l'algorithme à masque jetable est une méthode de chiffrement qui est théoriquement indéchiffrable. Toutefois, pour que cet algorithme fonctionne de façon optimale, les trois contraintes suivantes doivent absolument être respectées :

1. La clé doit être absolument aléatoire

2. La clé doit être de la même longueur que l'information à chiffrer

3. La clé ne doit être utilisée qu'une seule et unique fois (d'où le nom « chiffrement à masque jetable »).

Cet algorithme est une version améliorée des algorithmes de décalage simple (ou monoalphabétique). En effet, il inclut la notion de clé. Il est aussi une version améliorée du chiffrement de Vigenère qui fonctionne de la même façon, à la différence qu'il n'a pas les trois contraintes décrites précédemment, ce qui rend l'algorithme vulnérable. Plutôt que d'avoir un seul décalage fixe, chaque caractère de la clé définit le décalage du caractère à chiffrer. On parle alors de chiffrement par substitution polyalphabétique. C'est pour cette raison que la clé doit avoir la même taille que le message à chiffrer et donc le chiffrement est réalisé par flot. Ainsi, un caractère n'aura pas la même valeur une fois chiffré, grâce à la clé.

Protocole d'échange de Clé de Diffie-Hellman

Alice et Bob veulent s'échanger des informations de façon confidentielle en utilisant un système de communication non sécurisé (téléphone, internet,...), Charlie quant à lui désire savoir ce qu'ils veulent se dire. Alice et Bob vont donc devoir crypter (ou chiffrer) leur messages. La plupart des algorithmes de chiffrement des messages nécéssitent l'accord préalable d'Alice et Bob sur une clé qu'ils seront les seuls à connaitre (dite clé secrète). Cela pose évidemment un problème si Alice et Bob sont très éloignés et ne disposent pas de moyens de communication sûrs. Bob et Alice doivent donc se mettre d'accord sur un chiffre (qui leur servira de clé secrète) de telle sorte que même si Charlie entend toute la conversation il ne puisse pas le connaître. En 1976, Diffie et Hellmann ont proposé une solution pour y parvenir. Tout d'abord Alice et Bob se mettent d'accord sur un grand nombre premier appelé cryptographique p, et un générateur g de $\mathbb{Z}/p\mathbb{Z}$. Charlie connaît lui aussi p et g. Le protocole repose sur l'hypothèse que dans $\mathbb{Z}/p\mathbb{Z}$ l'application

$\{0, ..., p-1\} \to \mathbb{Z}/\mathbf{p}\mathbb{Z}\ x \to \mathbf{g^x}$

est à sens unique. Pour tout nombre x, on sait comment calculer rapidement g^x modulo p, mais à partir de cette valeur on ne sait pas calculer rapidement x (on ne sait pas non plus prouver que c'est impossible !). Si on prend x assez grand, la méthode consistant à essayer tout les nombres jusqu'à tomber sur x. Alice choisit alors secrètement un nombre privé x et calcule g^x dans $\mathbb{Z}/p\mathbb{Z}$ elle envoie le résultat à Bob. Calculer x est difficile pour Charlie et pour Bob également. Bob choisit lui aussi secrètement un nombre privé y, calcule g^y dans $\mathbb{Z}/p\mathbb{Z}$ et l'envoie à Alice. Charlie ne sait pas en déduire y. Finalement Alice élève à la puissance x le nombre que Bob lui a envoyé elle obtient : $(g^y)^x$ modulo p Bob, de même, élève à la puissance y le résultat que lui a envoyé Alice et obtient $(g^x)^y$ modulo p, comme $(g^y)^x = (g^x)^y = g^{xy}$ ce nombre : g^{xy} peut leur servir de clé secrète, et Charlie ne peut pas le calculer à partir de g^x et g^y.

Supposons qu'Alice et Bob choisissent le nombre premier $p = 1259$ et $g = 3$. Alice choisit $x = 144$ et calcule $3^{144} = 572 \pmod{1259}$. Alice envoie 572 à Bob. Bob choisit $y = 731$ et calcule $3^{731} = 900 \pmod{1259}$. Bob envoie 900 à Alice. Alice calcule $900^{144} = 572^{731} = 26 \pmod{1259}$. Donc, Alice et Bob peuvent utiliser la clé $K = 26$.

AES

Le chiffrement AES est le standard actuel en termes de cryptographie. Il est pour le moment indéchiffrable à moins d'utiliser une méthode de force brute. Une clé de 128 bits est utilisée pour la version standard d'AES. Initialement, le chiffrement de Rijndael (gagnant du concours AES) prévoyait en plus des chiffrements par clés de 192 et 256 bits. Les tailles de clés différentes ne changent pas le fonctionnement de l'algorithme. La seule différence se trouve dans le nombre de fois que les quatre opérations de la deuxième phase sont réalisées. Le Tableau 1.1 indique le nombre d'itérations (Nr) effectuées. Ce nombre dépend du nombre de colonnes que contient la matrice contenant la clé (Nk) ainsi que de son nombre de lignes (Nb). Ainsi, dans AES 128 bits, le nombre de tours de boucle sera égal à Nr - 1. Son fonctionnement se déroule en plusieurs étapes (généralement appelés « rounds »).

	Nk	Nb	Nr
128	4	4	10
192	6	4	12
256	8	4	14

TABLE 1.1 – Tableau du nombre d'itérations par rapport à la clé

RSA

Le chiffrement RSA a été inventé en 1977 par les mathématiciens Ronald Rivest, Adi Shamir et Leonard Adleman. Les initiales de leur nom ont donné RSA. Même si l'idée de base est la même que celle de Diffie-Hellman, (échanger une clé avec un grand nombre de personnes), son fonctionnement est différent bien qu'il soit basé sur la difficulté à factoriser de très grands nombres premiers. Il est massivement utilisé à travers le monde. En effet, c'est entre autres le chiffrement utilisé lors de connexions sécurisées sur un navigateur Web. Avec tous les utilisateurs du réseau Internet, il serait inimaginable d'utiliser un chiffrement symétrique. C'est pour cette raison que la clé privée est calculée avec RSA. Comme les algorithmes asymétriques sont plus lents que les symétriques, RSA ne calcule que la clé qui servira à chiffrer les données avec un chiffrement symétrique tel qu'AES.

Pour générer la clé publique qui se compose des nombres n et e, Alice doit choisir deux très grands nombres premiers distincts p et q d'une taille d'au moins 2512 bits chacun dans le but de former une clé de 21024 bits afin d'être suffisamment sûre. Ensuite, elle devra calculer n qui est égal à $p \times q$. Elle doit, après cela, calculer $\varphi(n) = (p-1) \times (q-1)$, qui va permettre de calculer la clé de décryptage d. Ensuite, elle doit choisir un exposant e qui est premier avec $\varphi(n)$. La clé publique va être constituée de e et n. Bob pourra ainsi grâce à e et n chiffrer un message m avec le calcul suivant : $m^e \mod n$. Ensuite, pour calculer la clé de décryptage, il faut que $e \times d \mod \varphi(n) = 1$. En résumé, pour trouver d, il faut faire le calcul suivant : $e^{-1} \mod \varphi(n)$. Pour trouver ce résultat, l'algorithme d'Euclide étendu peut être utilisé.

Prenons comme exemple $p = 7$, $q = 11$. Pour calculer n, il faut multiplier p et q, ce qui nous donne $n = 77$. Pour calculer $\varphi(n)$, il faut faire $(7 - 1) \times (11 - 1)$ ce qui donne 60. Maintenant, il faut choisir l'exposant e, qui doit être premier avec $\varphi(n)$. Prenons donc par exemple $e = 13$ (PGCD $(13, 60) = 1$ donc premiers entre eux). Voici donc la clé publique $e = 13$ et $n = 77$.

Pour calculer d, il suffit de faire $13^{-1} \mod 60$, ce qui donne 37. Pour vérifier que d est correct, il suffit de calculer $e \times d \mod \varphi(n) = 1$. Dans cet exemple cela revient à calculer $13 \times 37 \mod 60 = 1$.

1.6 Hachage

Les fonctions de hachage permettent de chiffrer un message sous la forme d'une chaîne de caractères de taille fixe, peu importe la taille du message d'origine, généralement entre 128 et 512 bits. Elles sont comparables à une empreinte, car un message aura toujours la même empreinte en appliquant une fonction de hachage. Elles sont à sens unique, ce qui signifie qu'il est facile de hacher un message, mais qu'il n'est en principe pas possible de calculer son inverse, à moins d'utiliser une méthode de force brute. Pour qu'une fonction de hachage soit sûre, elle doit être résistante aux collisions. En partant du principe qu'il y a une infinité de messages possibles pouvant être chiffrés, il est évident que plusieurs messages vont donner le même résultat une fois hachés. La résistance est le fait que les collisions ne puissent pas être retrouvées.

Les fonctions de hachage les plus utilisées de nos jours sont MD5 et SHA-1 : MD5 une sortie de 128 bits pour une entrée de 128+512 et SHA-1 avec une sortie de 160 bits pour une entrée de 160+512. Quelques attaques existent sur cette fonction mais elle est encore très utilisée pour des applications où la sécurité n'est pas primordiale. Elle est de nos jours essentiellement utilisée pour garantir l'authenticité de fichiers sur internet.

1.6.1 MD5

La fonction de hachage MD5 (Message Digest 5) a été développée par Ronald Rivest, l'un des créateurs de RSA. Elle utilise des blocs de 512 bits et génère des messages chiffrés de 128 bits. Chaque bloc est découpé en 16 sous-blocs de 32 bits (A, B, C et D). Les quatre calculs suivants sont réalisés 64 fois.

1. $F = (B \wedge C) \vee (\neg B \wedge D)$
2. $F = (D \wedge B) \vee (\neg D \wedge C)$
3. $F = B \oplus C \oplus D$
4. $F = C \oplus (B \vee \neg D)$

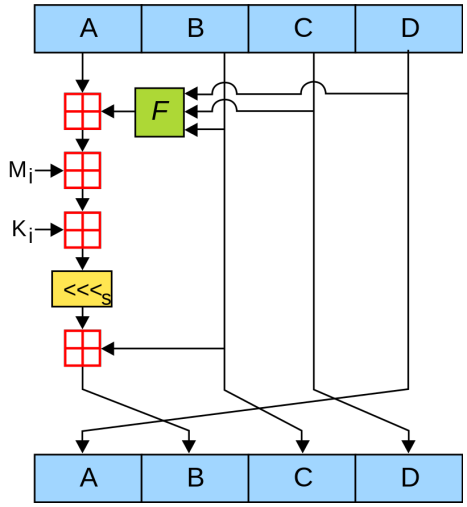


FIGURE 1.4 – Schéma représentant un tour dans MD5

Le résultat final obtenu donnera le message haché. MD5 est actuellement considéré comme une méthode de hachage non sûre. En réalisant une attaque par force brute, cela permet de retrouver un message original assez rapidement. La complexité du message haché est de 2^{64} calculs, or certains algorithmes permettent de réduire les calculs à 2^{30} opérations.

1.6.2 SHA-1

La fonction de hachage SHA-1 (Secure Hash Algorithm) a été développée par le NIST en 1995. SHA-1 utilise des blocs de 512 bits et produit des messages hachés de 160 bits. Les blocs sont découpés en 16 sous-blocs comme pour MD5 de 32 bits chacun. Ils sont ensuite étendus en 80 blocs. Pour produire les 80 blocs, les 4 opérations suivantes sont réalisées 80 fois.

1. F = (B AND C) OR (¬(B) AND D)
2. F = B XOR C XOR D
3. F = (D AND C) XOR (B AND D) XOR (C AND D)
4. F= B XOR C XOR D

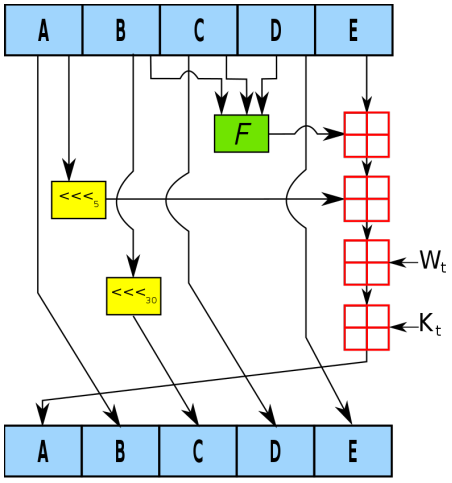


FIGURE 1.5 – Schéma représentant un tour dans SHA-1

A ce jour, SHA-1 est théoriquement cassable par force brute. Toutefois, il est encore très difficile de le casser facilement. Un mot de passe simple de type « 12345 » haché avec SHA-1 pourra être retrouvé facilement. En revanche un mot de passe contenant des caractères aléatoires prendra beaucoup plus de temps à être retrouvé. Malgré cela, il reste très utilisé.

1.7 Certificats numériques

Lors de l'utilisation des systèmes de cryptographie de clé publique, les utilisateurs doivent constamment vérifier qu'ils cryptent vers la clé du bon utilisateur, ce qui constitue un problème. Dans un environnement où le libre échange de clés via des serveurs publics est sécurisé, toute attaque menée par une personne intermédiaire, encore appelée un intercepteur, représente une menace éventuelle. Dans ce type d'attaque, une personne place une fausse clé comportant le nom et l'ID utilisateur du destinataire. Les données cryptées (et interceptées) vers le détenteur réel de cette clé erronée sont dorénavant entre de mauvaises mains.

Dans un environnement de clé publique, il est essentiel de s'assurer que la clé publique vers laquelle vous cryptez les données est celle du destinataire concerné et non une contre façon. Vous pouvez crypter uniquement vers les clés qui vous ont été distribuées physiquement. Supposons maintenant que vous devez échanger des informations avec des personnes que vous ne connaissez pas, comment savoir que vous êtes en possession de la bonne clé ? Les certificats numériques ou certificats simplifient la tâche qui consiste à déterminer si une clé publique appartient réellement à son détenteur supposé.

Un certificat correspond à une référence. Il peut s'agir par exemple de votre permis de conduire, de votre carte de sécurité sociale ou de votre certificat de naissance. Chacun de ces éléments contient des informations vous identifiant et déclarant qu'une autre personne a confirmé votre identité. Certains certificats, tels que votre passeport, représentent une confirmation de votre identité suffisamment importante pour ne pas les perdre, de crainte qu'une autre personne ne les utilise pour usurper votre identité.

Un certificat numérique contient des données similaires à celles d'un certificat physique. Il contient des informations associées à la clé publique d'une personne, aidant d'autres personnes à vérifier qu'une clé est authentique ou valide. Les certificats numériques permettent de contrecarrer les tentatives de substitution de la clé

d'une personne par une autre.

Un certificat numérique se compose de trois éléments :

- Une clé publique.
- Des informations sur le certificat. (Informations sur l'« identité » de l'utilisateur, telles que son nom, son ID utilisateur, etc.)
- Une ou plusieurs signatures numériques. La signature numérique d'un certificat permet de déclarer que ses informations ont été attestées par une autre personne ou entité. La signature numérique ne garantit pas totalement l'authenticité du certificat. Elle confirme uniquement que les informations d'identification signées correspondent ou sont liées à la clé publique.

Ainsi, un certificat équivaut en réalité à une clé publique comportant un ou deux types d'ID joints ainsi qu'une estampille agréée par d'autres personnes fiables.

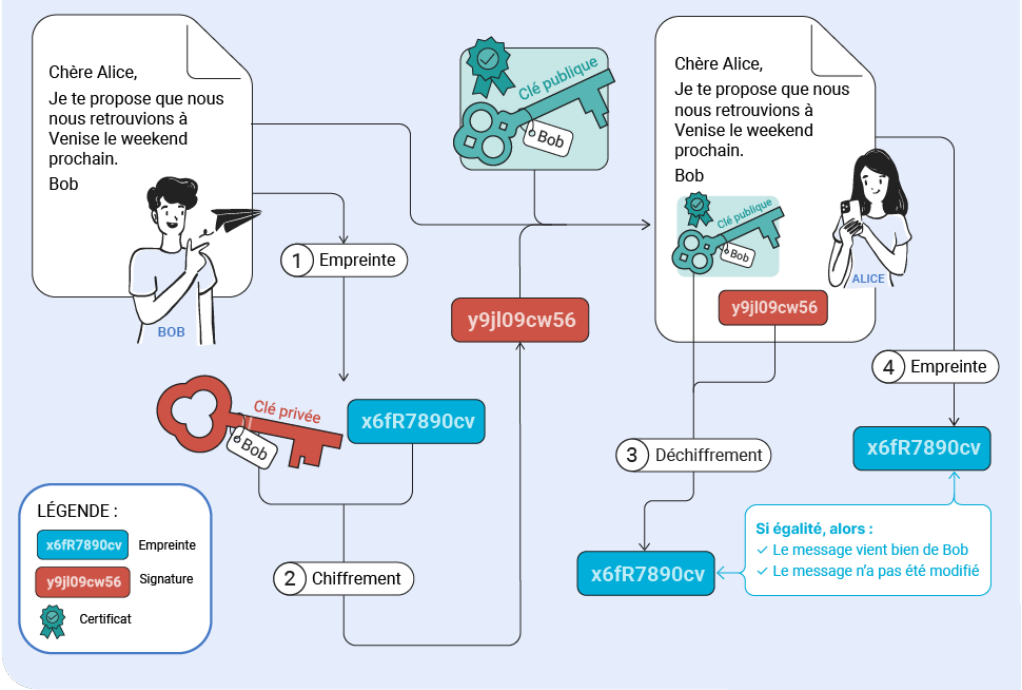


FIGURE 1.6 – Schéma de Certificat Numérique

1.8 Applications de la Cryptographie

Jusqu'au 20ième siècle la cryptographie (légale) était essentiellement réservée aux militaires et aux diplomates et accessoirement aux industriels et banquiers. Le développement des moyens de communications électromagnétiques faciles à intercepter, des stockages de données confidentielles dans des sites assez faciles à pénétrer et sur des supports (bandes ou disques, magnétiques optiques) faciles à dupliquer ont généré une demande de cryptosystèmes sûrs, faciles d'emploi et rapides pour des usages civils et commerciaux. Les codes symétriques ou à clefs secrètes, type DES (Data Encryption Standard), IDEA (International Digital Encryption Algorithm), AES (Advanced Encryption Standard) ont été créés pour couvrir ces besoins. L'application première de la cryptographie reste encore la transmission sécurisée de données sensibles mais d'autres applications commencent à se développer. Le commerce en ligne (e-commerce : paiement par carte bancaire...), les transactions

bancaires et boursières (e-banking :la consultation des données bancaires, ordres de bourse, virements de compte à compte,...), l'administration en ligne (e-administration : déclaration d'impôts, paiement de la TVA pour les entreprise,...), la télévision payante (chaîne payante, pay per view,...), la protection des télécommunications (internet, WIFI, Bluetooth, GSM, téléphonie mobile,...), l'identification automatique (avions, camions suivi par GPS,...), le tatouage ou watermarking des données numériques ainsi que la gestion des droits numériques (digital right management), etc. ont entraîné une explosion de l'utilisation de la cryptographie mais aussi une extension de son champ d'applications. Cette extension a été facilitée par la mise au point des procédés de transferts de clefs de Diffie et Hellman et des codes asymétriques ou à clefs publiques, de type RSA ou El Gamal. Les codes à clef publique permettent de résoudre avec des protocoles légers les questions de transfert de clefs, d'identification, d'authentification, de signature entre correspondants. Aujourd'hui les plus gros utilisateurs de cryptosystèmes sont les institutions financières (banques, bourses,...), les télécommunications (téléphonie mobile, WIFI, Internet, télévision payante,...), les sites d'achats en ligne, ... Il est probable que le tatouage ou watermarking des données numériques ainsi que la gestion des droits numériques (digital right management ou DRM) vont entraîner une demande accrue de cryptographie.

CHAPITRE 2

LES CODES DE CORRECTEURS D'ERREURS

2.1 Codes correcteurs d'erreurs

Les codes correcteurs d'erreurs sont un outil visant à améliorer la fiabilité des transmissions sur un canal bruité. La méthode qu'ils utilisent consiste à envoyer sur le canal plus de données que la quantité d'information à transmettre. Une redondance est ainsi introduite. Si cette redondance est structurée de manière exploitable, il est alors possible de corriger d'éventuelles erreurs introduites par le canal. On peut alors, malgré le bruit, retrouver l'intégralité des informations transmises au départ.

Une grande famille de codes correcteurs d'erreurs est constituée des codes par blocs. Pour ces codes, l'information est d'abord coupée en blocs de taille constante et chaque bloc est transmis indépendamment des autres, avec une redondance qui lui est propre. La plus grande sous-famille de ces codes rassemble ce que l'on appelle les codes linéaires.

Dans un code linéaire, les messages que l'on veut coder sont lus sous la forme d'un k-uplet d'éléments d'un corps fini K. Cet élément de K^k est ensuite transformé en élément de K^n par une application linéaire. La longueur n est choisie plus grande que la dimension k et c'est ainsi que la redondance est ajoutée. Ce que l'on appellera code est en fait le sous-espace vectoriel C de dimension k de K^n correspondant à l'espace image de l'application linéaire. La matrice génératrice G du code est la matrice associée à cette application linéaire. On appellera taux de transmission de ce code le rapport $\frac{k}{n}$ entre la longueur du message transmis et la quantité d'information utile qu'il contient.

Dans ce manuscrit, le terme code désignera, sauf mention explicite, un code linéaire sur le corps F_2 ou une de ses extensions de la forme F_{2^m}.

2.2 Que signifie décoder ?

La notion de décodage dans un code C désigne l'action d'associer un mot du code (un élément du sous-espace vectoriel) à un mot de l'espace vectoriel. En général, on cherche à décoder en associant à un mot le mot de code qui lui est le plus proche. Cependant, il est nécessaire de décider du sens à donner à l'expression "**le plus**

proche". Lors du décodage d'un message transmis le long d'un canal bruité, on s'intéresse principalement au décodage à vraisemblance maximale. Cela consiste à associer toujours le mot de code qui a la plus grande probabilité d'avoir donné ce mot en étant transmis sur le canal. Bien entendu, le décodage aura un sens différent selon la nature du canal.

Le modèle de canal le plus souvent utilisé est le canal binaire symétrique. Un tel canal ne possède pas de mémoire et les données qui y sont transmises sont binaires. Il est caractérisé par sa probabilité de transition p, qui est la probabilité qu'un bit émis (0 ou 1) soit changé en son opposé. Pour ce canal, le décodage à vraisemblance maximale consiste à trouver le mot de code qui a le plus de coordonnées en commun avec le mot reçu. Pour cela, on introduit la notion de poids de Hamming d'un élément de K^n : c'est le nombre de coordonnées non nulles de cet élément. Ce poids est donc compris entre 0 et n et définit une norme au sens mathématique du terme. La distance de Hamming est la distance déduite de cette norme : la distance entre deux éléments est le nombre de coordonnées où ils diffèrent.

On cherche donc à trouver le mot de code le plus proche pour la distance de Hamming. Cependant, effectuer un décodage complet (c'est-à-dire savoir décoder n'importe quel mot de l'espace) à vraisemblance maximale, s'avère en général être un problème très difficile.

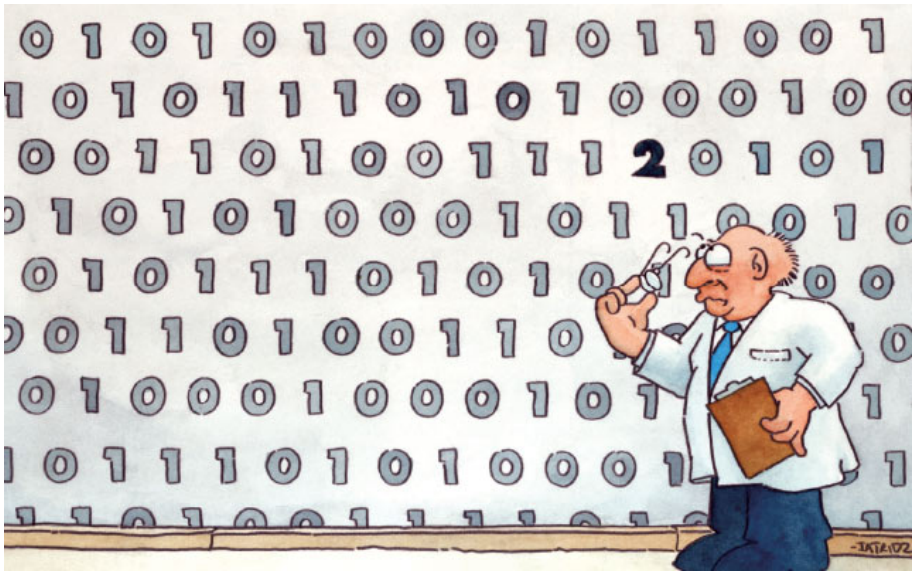
On s'intéresse donc aussi à d'autres formes de décodage, moins fortes : par exemple le décodage borné. Cela consiste à renvoyer un mot de code quelconque se trouvant à une distance inférieure à une borne ρ fixée. Evidemment, selon la valeur de ρ certains mots de l'espace peuvent s'avérer non décodables. On appellera rayon de recouvrement la valeur minimale de ρ pour laquelle l'espace est entièrement décodable : si on construit une « boule » de rayon ρ (pour la distance de Hamming) autour de chaque mot de code, c'est la valeur minimale pour laquelle les boules recouvrent tout l'espace. Ainsi, en effectuant un décodage borné jusqu'au rayon de recouvrement on a aussi un décodage complet, mais pas toujours à vraisemblance maximale. De plus, calculer le rayon de recouvrement d'un code est encore une fois un problème difficile.

De la même façon, on introduit la distance minimale d'un code. C'est la plus petite distance d séparant deux mots du code. Comme le code est linéaire c'est aussi le poids minimum d'un mot de code non nul. On écrira $[n, k, d]$ les paramètres d'un code de longueur n, dimension k et distance minimale d. En effectuant un décodage borné jusqu'à la distance $\left\lfloor \frac{d-1}{2} \right\rfloor$, on a un décodage unique qui se trouve donc être un décodage à vraisemblance maximale.

Pour un code parfait le rayon de recouvrement vaut exactement $\frac{d-1}{2}$ et le décodage borné devient alors aussi un décodage complet à vraisemblance maximale, malheureusement on ne connaît que peu de tels codes.

On cherche donc à construire des codes possédant une bonne distance construite (une borne inférieure sur la distance minimale du code, liée à la structure de ce code) et tels qu'il existe un algorithme efficace (polynomial en général) permettant de décoder de l'une des façons précédentes. La plupart du temps ces algorithmes permettront de décoder jusqu'à la moitié de la distance construite du code.

2.3 Problématique



Les mathématiques sont partout dans l'encodage de l'information, laquelle se fait en transformant les informations en suites de bits, soit des 0 et des 1. Mais, lors du transfert de l'information, certains bits peuvent être corrompus, un 0 étant transformé en 1, ou le contraire. Comment récupérer l'information ?

Les ordinateurs fonctionnent à l'aide de transistors que l'on peut modéliser simplement comme des interrupteurs qui ont deux positions : ouvert et fermé, que l'on peut représenter par 0 et 1, d'où l'idée de coder l'information en utilisant des bits.

Il est courant que des bits soient corrompus lors du transfert de l'information. Pourtant, il est souvent essentiel que le message puisse être décodé correctement. C'est le cas, par exemple, lorsqu'on communique avec une sonde interplanétaire. La solution retenue est d'utiliser **un code correcteur d'erreurs**. Le principe d'un tel code est d'ajouter de l'information redondante. Ainsi, si une partie de l'information est perdue, on peut la récupérer ailleurs.

Prenons un exemple. On veut envoyer quatre bits : 0010. On pourrait choisir de répéter chaque bit deux fois : 00 00 11 00. Ainsi, si on reçoit 00 10 11 00, on détecte qu'il y a eu une erreur. Par contre, on ne peut corriger... même s'il y a eu exactement une erreur. En effet, le mot envoyé aurait pu être 00 00 11 00, ou encore 00 11 11 00. Un tel code est un code détecteur d'erreurs. Si on choisit plutôt de répéter chaque bit trois fois, et donc d'encoder notre mot comme

000 000 111 000,

alors on peut corriger une erreur. Ainsi, si on reçoit **000 010 111 000**, on déduit que le mot envoyé est **0010**. Mais, si on reçoit **000 011 111 000**, on corrige à 0110 et on a quand même reçu un mot erroné.

Un code correcteur d'erreurs ne permet de récupérer le mot initial que si un nombre pas trop grand d'erreurs se sont produites.

Sinon, l'information est perdue. Certains codes correcteurs d'erreurs peuvent corriger plus d'une erreur. Le choix du code correcteur utilisé dépend de la fiabilité du canal de transmission.

Mais, revenons à notre exemple. On est parti d'un mot de quatre bits et on l'a allongé à 12 bits pour s'assurer de toujours pouvoir corriger une erreur.

2.4 Quelques Codes

Dans cette section, nous présentons quelques exemples de codes par blocs les plus célèbres et les algorithmes utilisés pour les décoder.

2.4.1 Code à répétitions

Ce code est la forme la plus simple de code par blocs : chaque bit à transmettre est simplement répété d fois dans le message transmis. Ainsi pour $d = 3$ sur $\mathbb{F}_2$ le message 1101 devient 111111000111. Ce code peut aussi être défini sur n'importe quel autre corps et sa matrice génératrice est toujours la matrice $1 \times d$ remplie de 1. Cette construction donne une distance minimale de d mais une dimension de 1 pour une longueur d aussi. On construit donc des codes de la forme $[d, 1, d]$. Le décodage est ensuite très simple puisqu'il suffit de garder le bit majoritaire dans chaque bloc. La capacité de correction ainsi obtenue est donc de $\left\lfloor \frac{d-1}{2} \right\rfloor$. Le principal problème est qu'avec ce code le taux de transmission est de $\frac{1}{d}$ ce qui est très peu. On peut toutefois noter que ce code est parfait quand on choisit d impair.

2.4.2 Code de Hamming

Ce code est lui aussi très simple mais a un bien meilleur taux de transmission. Pour le définir il est nécessaire d'introduire la notion de matrice de parité : c'est une matrice H de taille $(nk) \times n$ qui a pour noyau le code C tout entier. Ainsi si $c \in C$, on aura toujours $Hc^T = 0$, et uniquement dans ce cas-là. Remarquons qu'un même code C peut avoir plusieurs matrices de parité différentes, de même qu'il peut avoir plusieurs matrices génératrices différentes. On appellera syndrome d'un mot c l'élément S obtenu en calculant $S = Hc$. Ainsi les mots de C seront tous les éléments de F_q^n ayant un syndrome nul.

Le code de Hamming est donc un code binaire défini par sa matrice de parité plutôt que par sa matrice génératrice. C'est la matrice de dimension $r \times (2r1)$ qui contient toutes les colonnes non nulles distinctes que l'on peut écrire sur r bits. Ainsi pour $r = 3$ on a :

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Le code de Hamming est l'ensemble des mots de longueur $2r - 1$ dans le noyau de H. C'est donc un espace de dimension $2r - 1 - r$. De plus, la distance minimale de ce code est 3 car il n'existe aucun mot de code de poids 1 ou 2, puisque cela signifierait qu'il y a une colonne nulle ou deux colonnes égales dans H. On a donc un code $[2r - 1, 2^r - 1 - r, 3]$ dans lequel on doit donc pouvoir décoder une erreur. Effectivement, on peut facilement décoder une erreur seule en utilisant la matrice de parité : si on reçoit un mot bruité $c_0 = c + e$ où c est un mot du code et e une erreur de poids 1, on calcule $S = Hc_0 = Hc + He = He$, puisque c est dans le code et donc dans le noyau de H. L'erreur e étant de poids 1, S est donc égal à la colonne de H où se situe l'erreur et comme toutes les colonnes sont distinctes, on peut retrouver e et donc c.

Cette technique de décodage passant par le calcul d'un syndrome est la méthode utilisée pour décoder quasiment tous les codes par blocs et est un moyen facile d'obtenir une information ne dépendant que de e et pas du message transmis contenu dans c. Pour $r = 1$, le code de Hamming n'est pas intéressant puisqu'il ne contient que le mot nul et pour $r = 2$ c'est le code à répétition de longueur 3. En revanche, pour n'importe quelle autre valeur de r, c'est un code parfait.

2.4.3 Codes de Reed-Solomon

Les codes de Reed-Solomon ont été développés par Reed et Solomon dans les années 50 mais avaient déjà été construits par Bush un peu avant, dans un autre contexte. Ces codes sont certainement les codes par blocs les plus utilisés pour la correction d'erreurs en étant présents dans les CD, les DVD et la plupart des support de données numériques. Ils sont trés utilisés car ils sont extrémaux du point de vue de la capacité de correction.

1. **Construction**
   La façon la plus simple de voir ces codes est en tant que code d'évaluation : chaque élément du support du code est associé à un élément du corps $\mathbb{F}_q$ sur lequel est défini le code et chaque mot de code est l'évaluation d'une fonction $f \in \mathbb{F}$ sur le support. Pour les Reed-Solomon, on prend $\mathbb{F}_q = \mathbb{F}_{2^m}$ et $\mathbb{F}$ l'ensemble des polynômes de degré strictement inférieur à k sur $\mathbb{F}_{2^m}$. Ainsi, on a bien un code linéaire de longueur $n \leq 2^m$ et de dimension k. Une matrice génératrice du code peut alors s'écrire :

$$G = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{pmatrix}$$

   Par sa nature, ce code a une distance minimale d'au moins $n - k + 1$ car deux polynômes de degré $< k$ distincts ne peuvent pas être égaux en plus de $k - 1$ positions distinctes. Cette distance est même exactement égale à $n - k + 1$ puisque l'évaluation d'un polynôme de la forme $\prod_{i=1}^{k-1}(X - \alpha_i)$ est de poids $n-k+1$. On a donc des codes sur $\mathbb{F}_{2^m}$ de la forme $[n, k, n-k+1]$ qui peuvent donc avoir à la fois un bon taux de transmission et une bonne capacité de correction.

   Notons que cette distance minimale est la meilleure que l'on puisse atteindre avec ces paramètres : une distance minimale supérieure entrerait en contradiction avec la dimension du code.

   Ces codes ont aussi l'avantage de pouvoir corriger des rafales d'erreurs du fait de leur structure sur $\mathbb{F}_{2^m}$. En effet, une fois écrits en binaire, si une erreur atteint plusieurs bits à la suite, il y a de bonnes chances pour que cela n'affecte qu'un seul bloc de m bits et donc ne crée au final qu'une seule erreur.
2. **Décodage**
   Plusieurs algorithmes polynomiaux permettent de décoder efficacement $\frac{n-k}{2}$ erreurs dans un code de Reed-Solomon. Celui de Berlekamp-Massey est le plus

connu et le plus utilisé, mais l'algorithme le plus simple à expliquer est, selon nous, celui de Berlekamp-Welch .

On suppose que l'on a reçu un mot $c_0 = c + e$ où e est une erreur de poids w et c un mot de code, évaluation d'un polynôme P_c de degré $< k$. On appelle polynôme localisateur de e le polynôme unitaire L_e qui s'annule en tous les éléments du support où e est non nul. Le degré de L_e est donc w. Puisque e n'est pas connu ce polynôme est aussi inconnu. Si on multiplie le mot reçu par l'évaluation de ce polynôme on trouve :

$$\forall i \in [1; n] \quad c_i^{'} \times L_e(\alpha_i) = (P_c(\alpha_i) + e_i) \times L_e(\alpha_i) = P_c(\alpha_i) \times L_e(\alpha_i). \qquad (1\text{-}1)$$

Le polynôme P_c de degré $< k$ est lui aussi inconnu (c'est ce que l'on cherche à trouver), et on linéarise le système en introduisant le polynôme $N = P_c \times L_e$ qui est donc de degré au plus $k - 1 + w$. On obtient alors le système linéaire suivant :

$$\forall i \in [1; n] \quad c_i^{'} \times L_e(\alpha_i) = N(\alpha_i). \qquad (1\text{-}2)$$

Il est composé de n équations en $k+2w$ inconnues (w inconnues pour L_e qui est unitaire de degré w et $k+w$ pour N). De plus, toute solution du système (1-1) donne aussi une solution pour ce système. Il suffit donc qu'il y ait un nombre fini de solutions à (1-2) pour pouvoir retrouver une solution. Ce sera le cas si $n \geq k + 2w$ et donc si $w \leq \frac{n-k}{2}$. On peut donc facilement corriger jusqu'à la moitié de la distance minimale des Reed-Solomon. C'est le maximum que l'on puisse faire si on veut garder un décodage unique.

Au-delà de cette borne de $\frac{n-k}{2}$ erreurs, on peut encore décoder en temps polynomial en utilisant l'algorithme de Guruswami-Sudan . Cet algorithme permet d'effectuer du décodage par liste, c'est-à-dire qu'il renvoie la liste de tous les mots plus proches que la distance fixée. Avec cet algorithme, on peut corriger jusqu'à $n - \sqrt{nk}$ erreurs, et même s'il peut renvoyer une liste de solutions, dans la pratique la liste ne contiendra en général qu'une seule solution.

2.4.4 Codes Cycliques

Les codes cycliques sont des codes correcteurs d'erreurs plus avancées qui offrent plusieurs avantages par rapport aux méthodes plus simples comme le code de parité. Leur utilisation est répandue dans diverses applications en raison de ces avantages spécifiques. Les codes cycliques présentent une efficacité accrue en termes de détection et de correction d'erreurs par rapport aux codes de parité. De plus, ils sont capables de détecter et de corriger un plus large éventail d'erreurs, notamment les erreurs en rafale qui peuvent être problématiques dans les communications numériques. Leur structure mathématique permet une mise en œuvre efficace à l'aide d'algorithmes de décodage rapides, ce qui est essentiel pour les applications en temps réel telles que les télécommunications.

2.4.5 Codes de Goppa

1. **Construction**

   Les codes de Goppa sont des codes linéaires sur un corps fini $\mathbb{F}_p$. Cependant, leur construction passe par l'utilisation d'une extension $\mathbb{F}_{p^m}$. Un code de Goppa $\Gamma(L, g)$ est défini par son polynôme de Goppa g de degré t à coefficients dans $\mathbb{F}_{p^m}$ et son support $L \subset \mathbb{F}_{p^m}$ de n éléments. Si on note $\alpha_0, \ldots, \alpha_{n-1}$ les éléments

de son support, sa matrice de parité est alors obtenue à partir de la matrice suivante :

$$H_{\text{aux}} = \begin{bmatrix} \frac{1}{g(\alpha_0)} & \frac{1}{g(\alpha_1)} & \cdots & \frac{1}{g(\alpha_{n-1})} \\ \frac{\alpha_0}{g(\alpha_0)} & \frac{\alpha_1}{g(\alpha_1)} & \cdots & \frac{\alpha_{n-1}}{g(\alpha_{n-1})} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\alpha_0^{t-1}}{g(\alpha_0)} & \frac{\alpha_1^{t-1}}{g(\alpha_1)} & \cdots & \frac{\alpha_{n-1}^{t-1}}{g(\alpha_{n-1})} \end{bmatrix}$$

Chaque élément de cette matrice est ensuite décomposé en m éléments de $\mathbb{F}_p$, placés en colonnes, en utilisant une projection de $\mathbb{F}_{p^m}$ dans $\mathbb{F}_p^m$. On passe ainsi d'une matrice de taille $t \times n$ sur $\mathbb{F}_{p^m}$ à une nouvelle matrice de parité H de taille $mt \times n$ sur $\mathbb{F}_p$.

Les éléments du code $\Gamma(L, g)$ seront donc tous les éléments c de $\mathbb{F}_p^n$ tels que $H \times c^T = 0$. C'est donc un code de longueur n et dimension $k \geq n - mt$. De plus, un tel code a une distance minimale au moins égale à $t + 1$. En effet, toute sous-matrice carrée $t \times t$ de H_{aux} est inversible car H_{aux} s'écrit comme le produit d'une matrice de Vandermonde et d'une matrice diagonale inversible :

$$H_{\text{aux}} = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ \alpha_0^{t-1} & \cdots & \alpha_{n-1}^{t-1} \end{bmatrix} \times \begin{bmatrix} \frac{1}{g(\alpha_0)} & 0 & \cdots & 0 \\ 0 & \frac{1}{g(\alpha_1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{g(\alpha_{n-1})} \end{bmatrix}$$

Il n'existe donc pas de mot de code de poids inférieur ou égal à t. Les codes de Goppa sont donc de la forme $[n, k \geq n - mt, d \geq t + 1]$ sur $\mathbb{F}_p$, avec comme seule contrainte de longueur $n \leq 2^m$.

si on fixe $m = 1$, on se retrouve avec des codes qui ont exactement les mêmes performances que les Reed-Solomon.

2. **Décodage d'un code de Goppa**

On suppose avoir reçu un vecteur $a = (a_1, \ldots, a_n)$ de $\mathbb{F}^n$ et on cherche à retrouver le mot $\hat{a}$ du code C qui lui correspond, sous l'hypothèse qu'il n'y a eu moins de $\frac{r}{2}$ erreurs de transmission.

Pour tout i, soit $f_i(T)$ le polynôme de degré $< r$ tel que $1 \equiv f_i(T)(T - z_i) \pmod{G(T)}$. Le syndrome de a est le polynôme $S(T) = \sum_{i=1}^{n} a_i f_i(T)$. C'est aussi celui du vecteur d'erreur $\epsilon = a - \hat{a}$.

Soit $I \subset \{1, \ldots, n\}$ l'ensemble des indices où $\hat{a}_i \neq \hat{a}_i$. On suppose qu'il y a eu strictement moins de $\frac{r}{2}$ erreurs, c'est-à-dire que $\text{Card}(I) < \frac{r}{2}$, et on veut les corriger. On définit des polynômes $\sigma(T)$ et $\omega(T)$ par les formules :

$$\sigma(T) = \prod_{i \in I}(T - z_i), \quad \omega(T) = \sum_{i \in I} \epsilon_i \prod_{\substack{j \in I \\ j \neq i}} (T - z_j).$$

On a la congruence $S(T)\sigma(T) \equiv \omega(T) \pmod{G(T)}$. Par suite, les polynômes σ et ω peuvent être obtenus à l'aide de la méthode décrite au paragraphe précédent.

Pour terminer le décodage, il faut vérifier que σ est bien de la forme $\prod_{i \in I}(T - z_i)$ pour une partie I de $\{1, \ldots, n\}$ à déterminer explicitement, puis calculer les coefficients ϵ_i.

2.4.6 Codes LRPC

Les codes LRPC (Low Rank Parity Check) sont une famille de codes en métrique rang tels que l'ensemble des coordonnées d'une de leurs matrices de parité H appartient à un sous-espace F de $\mathbb{F}_{q^m}$ de petite dimension.

Un code LRPC C de rang d, longueur n et dimension k sur $\mathbb{F}_{q^m}$ est un code qui admet une matrice de parité $H \in \mathbb{F}_{q^m}^{n\times(n-k)}$ telle que l'espace vectoriel engendré par ses coefficients h_{ij} est de dimension d. On note $F = \langle h_{ij} \rangle$ le sous-espace de $\mathbb{F}_{q^m}$ engendré par les coordonnées de H.

1. **Décodage**
   L'algorithme de décodage des codes LRPC utilise la connaissance de l'espace F de petite dimension afin de retrouver le support de l'erreur E à partir du syndrome. L'idée générale est la suivante : à partir du syndrome $s = (s_1, \ldots, s_{n-k})$, on peut calculer $S = \langle s_1, \ldots, s_{n-k} \rangle$ qui est un sous-espace de l'espace produit EF. Si on a l'égalité $S = EF$, alors on peut en déduire le support de l'erreur E et il ne reste plus qu'à résoudre un système d'équations linéaires pour retrouver le vecteur erreur e. L'ensemble de ces étapes est détaillé ci-dessous.
2. **Algorithme de décodage**
   Soit C un code LRPC de longueur n et de dimension k de rang d, de matrice génératrice G et de matrice de parité H. Soit F le sous-espace de $\mathbb{F}_{q^m}$ engendré par les coordonnées h_{ij} de H et soit $\{F_1, \ldots, F_d\}$ une base de F.
   Soit $y = mG + e$ le mot reçu, où m est le message et e est une erreur de rang r. L'algorithme de décodage est présenté à la table 2.1.

Paramètre	Clé publique	Clé secrète	Chiffré
hqc-128	3024	40	6017
hqc-192	5690	40	11364
hqc-256	8698	40	17379
hqc-RMRS-128	2607	40	5191
hqc-RMRS-192	4906	40	9794
hqc-RMRS-256	7535	40	15047

TABLE 2.1 – Tailles en octets pour HQC

2.4.7 Codes BCH

Un code cyclique C de longueur n sur $\mathbb{F}_q$ est un code BCH de distance construite δ, avec $2 \leq \delta \leq n$, s'il existe un entier b tel que son polynôme générateur est égal à

$$g(x) = \text{ppcm}\{M_b(x), M_{b+1}, \ldots, M_{b+\delta-2}\},$$

c'est-à-dire si $g(x)$ est le polynôme monique de plus petit degré sur $\mathbb{F}_q$ qui a α^b, α^{b+1}, …, $\alpha^{b+\delta-2}$ parmi ses zéros, où α est une racine primitive n-ième de l'unité.

Si $b = 1$, il s'agit d'un code BCH au sens strict.

Si $n = q^m - 1$, le code BCH est appelé primitif, car α est un élément primitif de $\mathbb{F}_{q^m}$.

i.e. si $g(x)$ est le polynôme monique de plus petit degré sur $\mathbb{F}_q$ qui a α^b, α^{b+1}, …, $\alpha^{b+\delta-2}$ parmi ses zéros, où α est une racine primitive n-ième de l'unité.

Si $b = 1$, il s'agit d'un code BCH au sens strict.

Si $n = q^m - 1$, le code BCH est appelé primitif, car α est un élément primitif de $\mathbb{F}_{q^m}$.

Ainsi, un code BCH de distance construite δ a une séquence de $\delta - 1$ puissances successives de α comme zéros,il a distance minimale au moins δ.

La matrice de parité d'un code BCH de longueur n sur $\mathbb{F}_q$ de distance construite δ, par rapport à l'entier b, est

$$H := \begin{bmatrix} 1 & \alpha^{b} & \alpha^{2b} & \cdots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \cdots & \alpha^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{b+\delta-2} & \alpha^{2(b+\delta-2)} & \cdots & \alpha^{(n-1)(b+\delta-2)} \end{bmatrix}$$

où α est une racine primitive n-ième de l'unité dans $\mathbb{F}_{q^m}$ et H est considérée comme matrice sur $\mathbb{F}_q$, en remplaçant tout entrée par la colonne de m élément sur $\mathbb{F}_q$ correspondante. Après cette substitution, on obtient $m(\delta - 1)$ lignes, mais on ne peut pas dire si elles sont linéairement indépendantes. Ainsi, la dimension du code est au moins $n - m(\delta - 1)$. On peut résumer ces résultats dans un théorème :

Théorème : Un code BCH sur $\mathbb{F}_q$ de longueur n et distance construite δ a paramètres $[n, \geq n - m(\delta - 1), \geq \delta]$, où m est l'ordre de q dans $(\mathbb{Z}/n\mathbb{Z})^*$.

2.5 Principes de Base de Codes Correcteurs d'Erreurs

Les principes de base des codes correcteurs d'erreurs sont fondamentaux pour assurer la fiabilité des transmissions de données dans les systèmes de communication.

Détection des erreurs : Les codes correcteurs d'erreurs sont conçus pour détecter la présence d'erreurs dans les données transmises. Ils utilisent des techniques telles que la parité, les sommes de contrôle (checksums) ou les codes de détection d'erreurs cycliques (CRC) pour identifier les erreurs lors de la réception des données.

Correction des erreurs : En plus de détecter les erreurs, certains codes correcteurs d'erreurs sont capables de les corriger. Ces codes ajoutent des informations redondantes aux données transmises, ce qui permet au récepteur de déterminer et de corriger les erreurs de transmission sans avoir besoin de renvoyer les données.

Redondance : Les codes correcteurs d'erreurs exploitent le principe de redondance pour détecter et corriger les erreurs. En ajoutant des bits de redondance aux données, ces codes permettent au récepteur de reconstruire les données originales même en cas d'erreurs de transmission.

Capacité de correction : La capacité d'un code correcteur d'erreurs à détecter et à corriger les erreurs dépend de sa capacité de correction. Certains codes sont capables de corriger un nombre limité d'erreurs, tandis que d'autres sont plus robustes et peuvent corriger un plus grand nombre d'erreurs.

Complexité et efficacité : Les codes correcteurs d'erreurs varient en termes de complexité et d'efficacité. Certains codes sont simples à mettre en œuvre mais offrent une capacité de correction limitée, tandis que d'autres sont plus complexes mais peuvent corriger un plus grand nombre d'erreurs avec une meilleure efficacité.

2.6 Codes Correcteurs et Cybersécurité

Les codes correcteurs d'erreurs sont essentiels en cybersécurité pour garantir l'intégrité et la fiabilité des données transmises sur les réseaux. Cette technologie joue un rôle crucial en détectant et en corrigeant les erreurs de transmission, qu'elles soient accidentelles ou malveillantes, qui pourraient compromettre la sécurité des systèmes informatiques et des communications numériques. En assurant la détection précoce et la correction des erreurs, ces codes contribuent à maintenir un niveau élevé de sécurité dans divers domaines, tels que les communications sans fil, le stockage de données, les systèmes de contrôle industriels et les communications gouvernementales sensibles. Leur utilisation permet de prévenir les altérations malveillantes des données et de protéger les systèmes contre les attaques potentielles, renforçant ainsi la confiance dans l'intégrité des données et la sécurité des réseaux informatiques.

CHAPITRE 3

CODES CORRECTEURS D'ERREURS ET CRYPTOGRAPHIE

3.1 Cryptosystème de McEliece

C'est le plus ancien cryptosystème à clef publique utilisant des codes correcteurs d'erreurs. Il a été imaginé, comme son nom l'indique, par McEliece en 1978, juste après l'invention des premiers cryptosystèmes à clef publique, à peu près en même temps que RSA , le système le plus utilisé aujourd'hui. Comme tous les cryptosystèmes à clef publique, ce système est constitué de 3 algorithmes :

- La génération de clefs
- Le chiffrement (utilisant la clef publique)
- Le déchiffrement (utilisant la clef secrète).

McEliece a suggéré d'utiliser les codes de Goppa, qui sont des codes linéaires avec un algorithme rapide de décodage.

3.1.1 Génération de clef

On commence par générer un code de Goppa corrigeant t erreurs et sa matrice de parité G de taille $k \times n$. On va maintenant mélanger cette matrice pour la rendre indistinguable d'une matrice aléatoire : pour cela, on a besoin d'une matrice de permutation aléatoire P de taille $n \times n$ et d'une matrice inversible aléatoire Q de taille $k \times k$. La clé publique sera la matrice $G_{\text{pub}} = Q \times G \times P$, qui est indistinguable d'une matrice aléatoire. Par contre, la clé secrète est composée des trois matrices Q, P et G qui permettent de retrouver la structure du code de Goppa et donnent donc accès à l'algorithme de décodage.

3.1.2 Chiffrement

Soit m un message de k bits qu'Alice veut chiffrer. Alice calcule le mot c de longueur n associé à m :

$$c = m \times G_{\text{pub}}$$

Ensuite, elle génère une erreur aléatoire e de longueur n et de poids t. Le chiffré sera simplement le mot de code bruité :

$$c_0 = c + e$$

3.1.3 Déchiffrement

Pour déchiffrer, Bob connaît Q, P et G, il calcule :

$$c_0 \times P^{-1} = mG_{\text{pub}}P^{-1} + eP^{-1} = mQ \times G + eP^{-1}$$

$mQ \times G$ est un mot du code de Goppa et eP^{-1} est une erreur de poids t. Bob peut décoder cette erreur et retrouver le message initial mQ. Il ne reste plus qu'à multiplier par Q^{-1} pour retrouver le message m et avoir fini de déchiffrer.

3.2 Variante de Niederreiter

Cette variante du cryptosystème de McEliece a été mise au point par Niederreiter en 1986. Elle est exactement équivalente du point de vue de la sécurité et est un peu plus efficace en temps de calcul. Elle fonctionne comme le chiffrement de McEliece, mais en utilisant la matrice de parité du code et en utilisant l'erreur pour contenir le message.

3.2.1 Génération de clef

Comme pour McEliece, on commence par générer un code de Goppa et sa matrice de parité H de taille $(n-k) \times n$. On génère une permutation aléatoire P de taille $n \times n$ et une matrice inversible Q de taille $(n-k) \times (n-k)$. La clé publique est :

$$H_{\text{pub}} = Q \times H \times P$$

3.2.2 Chiffrement

Pour chiffrer, Alice commence par coder le message m en un mot e_m de poids t et de longueur n. On peut donc mettre au plus $\log_2\left(\frac{n}{t}\right)$ bits d'information dans m. Le chiffré transmis est le syndrome S de ce mot :

$$S = H_{\text{pub}} \times e_m^T$$

3.2.3 Déchiffrement

Pour déchiffrer, on procède comme avec le système de McEliece. Bob commence par calculer :

$$Q^{-1} \times S = H \times Pe_m^T$$

Encore une fois, Pe_m^T est de poids t lui aussi et il sait donc retrouver l'erreur correspondante au syndrome $Q^{-1} \times S$. Il retrouve donc Pe_m^T et donc aussi e_m. Bob en déduit alors le message clair m.

Ici, la clef publique fera $n(n-k)$ bits (ou $k(n-k)$ sous forme systématique), les blocs sont de longueur $\log_2\left(\frac{n}{t}\right)$ et le taux de transmission est $\frac{\log_2\left(\frac{n}{t}\right)}{n-k}$. Le chiffrement est lui beaucoup moins coûteux avec juste le codage en mot de poids constant (qui sera malheureusement souvent l'étape la plus coûteuse) et la somme de t colonnes de H pour un coût de $t(n-k)$. Le déchiffrement a lui un coût très similaire, mais

un certain gain sur les produits de matrice puisque l'on travaille sur des tailles un peu plus petites.

Pour les paramètres $[2048, 1685, 67]$, cela donne un taux de transmission de 0.66 (donc un peu moins que McEliece), des blocs de 363 bits pour 240 bits d'information (donc beaucoup plus courts que ceux de McEliece) et une matrice de 750 kbits (ou encore 600 kbits sous forme systématique).

Ce système a donc l'avantage d'avoir des blocs beaucoup plus courts et un chiffrement bien plus rapide, sans pour autant perdre beaucoup sur les autres points vis-à-vis du cryptosystème de McEliece.

3.3 Utilisation du chiffrement de Niederreiter

Le cryptosystème de Niederreiter par rapport à la plupart des autres cryptosystèmes à clef publique, l'avantage d'avoir un chiffrement très rapide. On peut donc imaginer utiliser une fonction similaire à f_c et ainsi bénéficier à la fois de la rapidité du système et de sa sécurité , dans le système de Niederreiter le chiffrement se fait en utilisant une matrice de parité d'un code de Goppa. On convertit le message à chiffrer en un mot de longueur n et de poids w donné que l'on multiplie par la matrice de parité. Le résultat est un syndrome qui va servir de chiffré. La matrice de parité est supposée être indistinguable d'une matrice aléatoire et seule la connaissance de sa structure de code de Goppa et de la permutation utilisée pour la brouiller permet de déchiffrer le syndrome. Pour utiliser cette fonction de chiffrement en tant que fonction de compression on n'a pas besoin de connaître la trappe (au contraire, on veut même être certain que personne ne peut inverser la fonction de compression) et on peut donc prendre une véritable matrice aléatoire H de taille $r \times n$. La fonction de compression se décompose alors en deux étapes :

1. convertir les $s = \log_2 \left(\binom{n}{w}\right)$ bits d'entrée en un mot de longueur n et poids w.
2. multiplier ce mot par la matrice H pour obtenir un syndrome de longueur r.

3.4 Cryptographie basée sur les codes correcteurs d'erreurs

3.4.1 Cryptographie en métrique de Hamming

Hamming Quasi-Cyclic (HQC) On présente le cryptosystème HQC, qui utilise la structure des codes doublement circulant. Ce schéma permet d'obtenir de petites tailles de clés (table 3.1) ainsi que des temps d'exécution rapides (table 3.2) tout en ayant une réduction de sécurité au problème du décodage de codes aléatoires : en effet le code structuré C permettant de le décodage est une donnée publique, contrairement au cryptosystème de McEliece.

Paramètre	Clé publique (octets)	Clé secrète (octets)	Chiffré (octets)
hqc-128	3024	40	6017
hqc-192	5690	40	11364
hqc-256	8698	40	17379
hqc-RMRS-128	2607	40	5191
hqc-RMRS-192	4906	40	9794
hqc-RMRS-256	7535	40	15047

TABLE 3.1 – Tailles en octets pour HQC

Paramètre	Génération de clés (kilocycles)	Chiffrement (kilocycles)	Déchiffrement (kilocycles)
hqc-128	175	286	486
hqc-192	386	636	966
hqc-256	633	1076	1577
hqc-RMRS-128	160	272	556
hqc-RMRS-192	350	598	1021
hqc-RMRS-256	589	1013	1649

TABLE 3.2 – Performances en kilocycles pour HQC

1. Données publiques

— Un code $[n, k]$ C de matrice de génératrice G pouvant corriger jusqu'à σ erreurs.

2. Génération de clés

— On tire aléatoirement $h \in \mathbb{F}_2^n$ et $(x, y) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ tels que $\|x\| = \|y\| = w$.

— (x, y) est la clé secrète et $(h, s = x + hy)$ est la clé publique.

3. Chiffrement d'un message $m \in \mathbb{F}_2^k$

— On tire aléatoirement $e \in \mathbb{F}_2^n$ et $(r_1, r_2) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ tels que $\|e\| = w_e$ et $\|r_1\| = \|r_2\| = w_r$.

— On calcule $u = r_1 + hr_2$ et $v = mG + sr_2 + e$.

— Le chiffré de m est le couple (u, v).

4. Déchiffrement

— On décode $v - uy$ en m en utilisant l'algorithme de décodage de C.

3.4.2 Cryptographie en métrique rang

Schémas de chiffrement

Le premier cryptosystème basé sur les codes correcteurs d'erreurs en métrique de rang a été proposé en 1991 par Gabidulin, Paramonov et Tretjakov. Ce cryptosystème est une variante du système de McEliece utilisant les codes de Gabidulin. Cependant, il a été attaqué en raison de la structure algébrique des codes de Gabidulin.

Les codes LRPC (Low Rank Parity Check) ont ensuite été utilisés pour instancier le cryptosystème de McEliece. La sécurité de ce système repose sur deux problèmes : le problème de décodage en métrique de rang et le problème d'indistinguabilité des codes LRPC. L'utilisation de la métrique de rang permet d'obtenir des tailles de clés et des temps d'exécution très compétitifs, comme le montrent les performances de ROLLO, un cryptosystème soumis à l'appel à projets du NIST. Étant donné que l'algorithme de décodage des codes LRPC a une probabilité d'échec non nulle, ROLLO propose deux jeux de paramètres : ROLLO-I pour l'échange de clés avec une probabilité d'échec de déchiffrement de l'ordre de 2^{-30} et ROLLO-II pour le chiffrement avec une probabilité d'échec de déchiffrement de l'ordre de 2^{-128}.

RQC (Rank Quasi-Cyclic) est une alternative à ROLLO, également soumise lors de l'appel à projets du NIST. Ce cryptosystème a une structure similaire à HQC. Les tailles de clés et les performances sont moins compétitives que celles de ROLLO,

mais RQC présente deux avantages : d'une part, l'algorithme de décodage utilisé lors du déchiffrement est celui des codes de Gabidulin, ce qui permet d'éviter les échecs de déchiffrement. D'autre part, la sécurité repose uniquement sur une variante du problème de décodage de syndrome pour les codes.

3.5 Pourquoi faire de la cryptographie avec des codes ?

La cryptographie basée sur des codes joue un rôle fondamental dans la sécurisation des communications et la protection des données sensibles. En cryptant les informations, elle rend les données illisibles pour toute personne non autorisée, assurant ainsi la confidentialité des échanges. Cette confidentialité est essentielle dans de nombreux contextes, notamment pour protéger les informations personnelles, financières ou stratégiques contre les accès non autorisés.

De plus, les codes cryptographiques peuvent être conçus avec une complexité élevée, rendant leur déchiffrement extrêmement difficile sans la clé appropriée. Cette complexité renforce la sécurité des systèmes contre les attaques potentielles, telles que les tentatives de décryptage par force brute ou les attaques de type cryptanalyse.

La cryptographie offre également des mécanismes pour vérifier l'authenticité des informations et garantir leur intégrité. Par exemple, les codes de hachage sont souvent utilisés pour créer des empreintes numériques uniques des données, permettant de détecter toute altération ultérieure. Cela permet de s'assurer que les données n'ont pas été modifiées ou corrompues lors de leur transmission.

En outre, la cryptographie simplifie la gestion des clés en permettant aux utilisateurs d'échanger des informations de manière sécurisée sans avoir à partager directement leurs clés secrètes. Cela évite les risques associés à la divulgation involontaire de clés et facilite la collaboration sécurisée entre différentes parties.

Avec des applications pratiques dans de nombreux domaines, tels que les communications en ligne, les transactions financières et la sécurité informatique, la cryptographie basée sur des codes est essentielle dans notre monde numérique moderne pour assurer la confidentialité, la sécurité et la fiabilité des informations échangées.

3.6 Sécurité

Les codes correcteurs d'erreurs sont des outils fondamentaux en cryptographie pour garantir la confidentialité et l'intégrité des données. Leur utilisation sécurisée repose sur plusieurs principes et défis :

- **Confidentialité et Intégrité des Données :** Les codes correcteurs d'erreurs sont utilisés pour chiffrer les données et vérifier leur intégrité lors de leur transmission ou de leur stockage. En ajoutant de la redondance aux données, les codes permettent de détecter et de corriger les erreurs qui pourraient résulter de perturbations ou d'attaques malveillantes.

- **Résistance aux Attaques :** La sécurité des systèmes utilisant des codes correcteurs d'erreurs dépend de leur capacité à résister à différentes formes d'attaques, telles que les attaques par canal auxiliaire et les attaques par canal caché. Les paramètres des codes doivent être choisis de manière à garantir une sécurité adéquate tout en minimisant les risques d'attaques réussies.

- **Complexité des Algorithmes :** Les algorithmes de codage et de décodage doivent être suffisamment complexes pour assurer la sécurité du système. Cependant, une complexité excessive peut entraîner des performances médiocres ou des exigences de ressources élevées, ce qui compromettrait la praticité du système.

- **Masquage de la Structure du Code :** Pour garantir la sécurité, il est essentiel de masquer la structure du code utilisé. Cela rendrait difficile pour un attaquant de retrouver les paramètres du code et d'exploiter ses vulnérabilités.

- **Choix des Familles de Codes :** Le choix de la famille de codes est crucial pour la sécurité du système. Il doit offrir un équilibre entre la capacité de décodage efficace et la résistance aux attaques cryptographiques. Les familles de codes bien étudiées et éprouvées, telles que les codes de Goppa, sont souvent privilégiées pour leur robustesse et leur capacité à résister aux attaques.

3.7 Application des Codes Correcteurs d'Erreurs et des Fonctions de Hachage en Cryptographie

La combinaison des fonctions de hachage et des codes correcteurs d'erreurs offre une approche robuste pour garantir à la fois l'intégrité des données et la fiabilité des communications. En examinant de près cet exemple, nous pouvons apprécier comment cette association assure l'intégrité des données et la fiabilité de la communication dans des protocoles sécurisés tels que SSL/TLS et IPsec.

- **Utilisation des Fonctions de Hachage pour l'Intégrité :**

  Dans ces protocoles, des fonctions de hachage sont utilisées pour créer des empreintes ou des condensés des données échangées. Ces empreintes sont ensuite incluses dans les messages ou les paquets de données. Lorsque les données sont reçues, les destinataires recalculent les empreintes à partir des données reçues et les comparent aux empreintes reçues. Si les empreintes correspondent, cela garantit que les données n'ont pas été altérées pendant la transmission.

- **Utilisation des Codes Correcteurs d'Erreurs pour la Fiabilité :**

  Les protocoles SSL/TLS et IPsec incorporent également des mécanismes de détection et de correction d'erreurs pour assurer la fiabilité de la communication. Des codes correcteurs d'erreurs sont ajoutés aux données pour détecter et corriger les erreurs introduites pendant la transmission. Cela garantit que même en présence de perturbations sur le canal de communication, les données peuvent être récupérées de manière fiable sans perte d'intégrité.

CHAPITRE 4

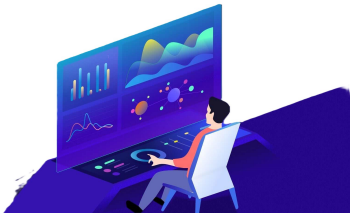
SIMULATION

4.1 Outils Pour le Développement D'applications Desktop

4.1.1 Qt Designer

Qt Designer est un outil de conception graphique intégré dans le framework Qt, qui permet de créer des interfaces utilisateur pour des applications Qt. Il offre une interface intuitive permettant de concevoir des fenêtres, des boîtes de dialogue, des boutons, des zones de texte et d'autres éléments d'interface en les glissant-déposant simplement sur une zone de travail. Avec Qt Designer, les développeurs peuvent créer des interfaces graphiques sans avoir à écrire manuellement le code correspondant. Une fois la conception terminée, Qt Designer génère automatiquement le code XML ou Python nécessaire pour intégrer l'interface utilisateur dans l'application Qt, ce qui simplifie considérablement le processus de développement.

4.2 Interface Desktop



Nous avons développé deux applications desktop distinctes pour répondre à différents besoins en cryptographie et en correction d'erreurs.La premiére application permet à l'utilisateur de choisir entre l'utilisation d'un code cyclique ou d'un code de Goppa pour encoder des messages.

- **Code Cyclique** : Si l'utilisateur sélectionne l'option du code cyclique, il peut entrer un message ainsi que la position de l'erreur pour encoder le message, le résultat s'affiche sous la forme du "**Encoded message**" et "**Received message with error**". L'application fournit une interface intuitive pour saisir ces informations et procéder à l'encodage.
- **Code de Goppa** : Si l'utilisateur opte pour le code de Goppa, l'application affiche une autre interface dédiée où l'utilisateur peut entrer un message, spécifier le nombre de lignes et la longueur n du message. Ces paramètres sont utilisés pour générer la matrice H nécessaire à l'encodage du message. Cette fonctionnalité permet de bénéficier de la robustesse et de la sécurité accrues offertes par les codes de Goppa.

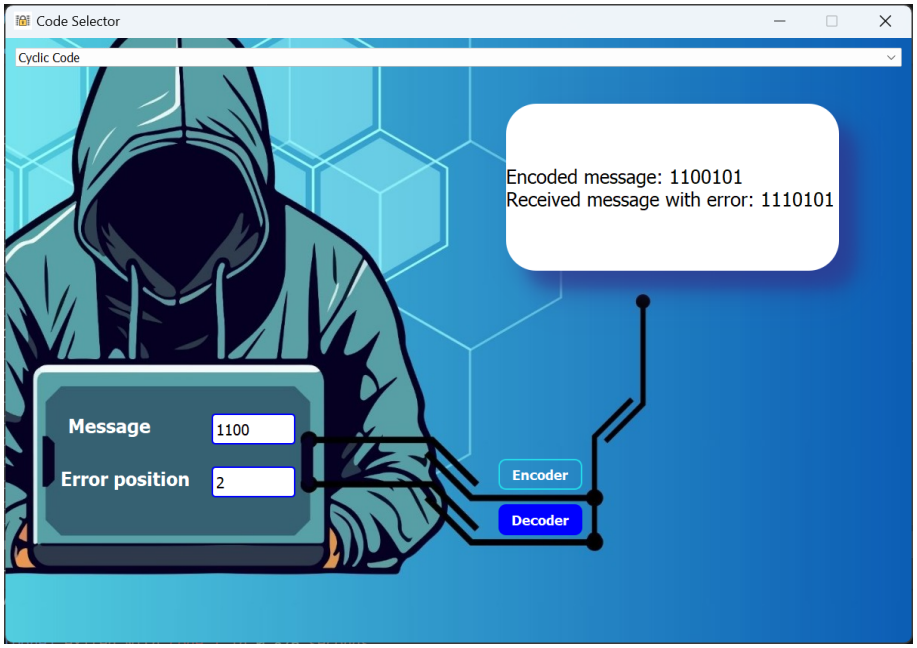


FIGURE 4.1 – Code Cyclique

Dans le codes cyclique, nous utilisons le polynôme générateur $G(x) = x^3 + x$, correspondant à la séquence binaire $[1, 0, 1, 0]$. Pour illustrer le processus de codage et de décodage, nous avons entré le message 1100.

Détail des calculs

1. **Message d'origine** : 1100.
2. **Calcul du reste avec** $G(x) = x^3 + x$:

- **Message polynomial** : $M(x) = x^3 + x^2$ (correspondant à 1100).
- **Multiplication par** x^3 : $M(x) \times x^3 = x^6 + x^5$.

Pour trouver le reste, nous effectuons la division de $x^6 + x^5$ par $x^3 + x$:

- **Division initiale :**

$x^6 + x^5 \div (x^3 + x) = x^3$
Multiplication : $x^3 \times (x^3 + x) = x^6 + x^4$

Soustraction : $(x^6 + x^5) - (x^6 + x^4) = x^5 - x^4$

- **Division suivante :**

$x^5 - x^4 \div (x^3 + x) = x^2$
Multiplication : $x^2 \times (x^3 + x) = x^5 + x^3$
Soustraction : $(x^5 - x^4) - (x^5 + x^3) = -x^4 - x^3$

- **Division suivante :**

$-x^4 - x^3 \div (x^3 + x) = -x$
Multiplication : $-x \times (x^3 + x) = -x^4 - x^2$
Soustraction : $(-x^4 - x^3) - (-x^4 - x^2) = -x^3 + x^2$

- **Division suivante :**

$-x^3 + x^2 \div (x^3 + x) = -1$
Multiplication : $-1 \times (x^3 + x) = -x^3 - x$
Soustraction : $(-x^3 + x^2) - (-x^3 - x) = x^2 + x$
$R(x) = x^2 + x$

Le message encodé devient alors :
$E(x) = M(x) \times x^3 + R(x) = x^6 + x^5 + x^2 + x$
En notation binaire, cela donne 1100101.
3. **Introduction de l'erreur** :

- **Message encodé original :** 1100101.
- **Erreur introduite à la position 2 :** 1100101 → 1110101.

Le message reçu avec l'erreur est donc **1110101**.

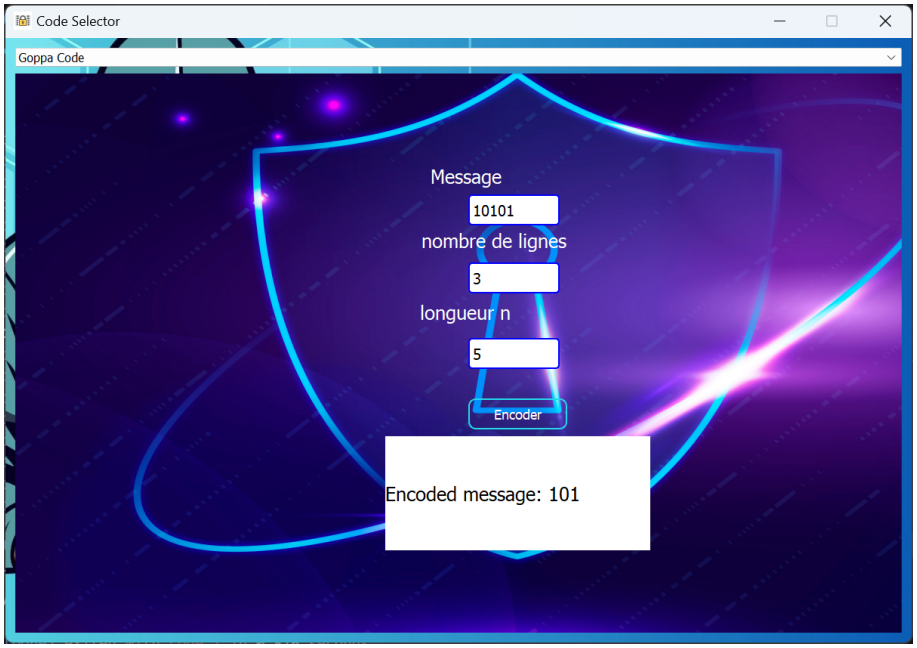


FIGURE 4.2 – Code Goppa

dans le code goppa , prenons un exemple concret où l'utilisateur entre le message 10101 et spécifie que le nombre de lignes est 3 et la longueur du message est 5.

Entrée du message et paramètres :

- **Message :** 10101
- **Nombre de lignes :** 3
- **Longueur du message n :** 5

Génération de la matrice H :

La matrice H est une matrice de parité utilisée pour l'encodage dans le code de Goppa. Sa taille est généralement $k \times n$, où k est le nombre de lignes et n est la longueur du message. Dans ce cas, nous générons une matrice H de taille 3×5. La matrice H est :

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 6 & 8 & 10 \\ 3 & 6 & 9 & 12 & 15 \end{bmatrix}$$

Et puisque nous travaillons avec des codes binaires, chaque élément de la matrice est modulo 2. Alors la matrice H devient :

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Encodage du message :

Pour encoder le message, nous multiplions le message (sous forme de vecteur ligne) par la matrice H. Supposons que le message soit représenté comme un vecteur ligne m :

$$m = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Le message encodé c est obtenu par la multiplication matricielle $c = m \cdot H^t$, réalisée modulo 2 :

$$\begin{aligned} c &= \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}^t \\ &= \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} (1 \times 1 + 0 \times 0 + 1 \times 1 + 0 \times 0 + 1 \times 1) & \cdot \\ (1 \times 0 + 0 \times 0 + 1 \times 0 + 0 \times 0 + 1 \times 0) & \cdot \\ (1 \times 1 + 0 \times 0 + 1 \times 1 + 0 \times 0 + 1 \times 1) & \cdot \end{pmatrix} \\ &= \begin{pmatrix} 3 & 0 & 3 \end{pmatrix} \end{aligned}$$

Ainsi, le message encodé est $\begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$.

Cet exemple montre comment l'utilisateur peut entrer un message et des paramètres pour générer la matrice H et encoder le message en utilisant le code de Goppa. Le processus d'encodage garantit que le message est transformé de manière à détecter et corriger les erreurs potentielles lors de la transmission, assurant ainsi une communication sécurisée et fiable.

La deuxième application que nous développons est destinée au chiffrement de messages à l'aide de l'algorithme de Vernam. Dans cette application, l'utilisateur peut saisir un message ainsi qu'une clé de chiffrement. Ensuite, l'application applique l'algorithme de Vernam pour chiffrer le message en combinant chaque caractère du message avec le caractère correspondant de la clé. Cette méthode de chiffrement offre un niveau élevé de sécurité car elle utilise une clé aléatoire de la même longueur que le message, rendant la cryptanalyse extrêmement difficile. Une fois le message chiffré, l'utilisateur peut le transmettre en toute sécurité, sachant qu'il est protégé par un chiffrement robuste.

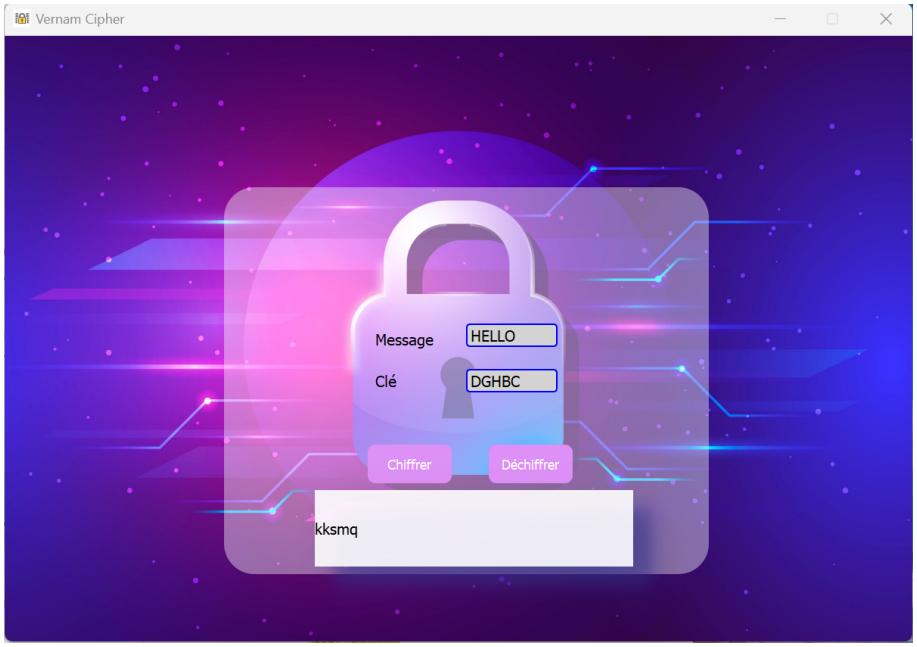


FIGURE 4.3 – Code de Vernam

Prenons un exemple concret où le message est "**hello**" et la clé de chiffrement est "**dghbc**".

Message : hello
Clé de chiffrement : dghbc

Étapes du Chiffrement :

- **Conversion en Valeurs Numériques**
  Chaque caractère est converti en sa position dans l'alphabet (A=0, B=1, ..., Z=25).
  Message : hello
  h = 7
  e = 4
  l = 11
  l = 11
  o = 14

Clé de chiffrement : dghbc
d = 3
g = 6
h = 7
b = 1
c = 2

Addition Modulo 26
On additionne les valeurs correspondantes du message et de la clé, puis on prend le résultat modulo 26.
h (7) + d (3) = 10 (k)
e (4) + g (6) = 10 (k)
l (11) + h (7) = 18 (s)
l (11) + b (1) = 12 (m)
o (14) + c (2) = 16 (q)

- **Conversion en Caractères Alphabétiques**
  Convertir les valeurs numériques résultantes en caractères alphabétiques.
  10 = k
  10 = k
  18 = s
  12 = m
  16 = q

Message Chiffré
Le message chiffré est donc : "kksmq".

Étapes du déchiffrement :

- **Conversion en Valeurs Numériques**

Message Chiffré :kksmq
k = 10
k = 10
s = 18
m = 12
q = 16

Clé de chiffrement : dghbc
d = 3
g = 6
h = 7
b = 1
c = 2

Soustraction modulo 26 : Soustraire les valeurs correspondantes du message chiffré et de la clé de chiffrement, puis prendre le résultat modulo 26 pour obtenir les valeurs numériques des caractères du message original.

k (10) - d (3) = 7 (h)
k (10) - g (6) = 4 (e)
S (18) - h (1) = 11 (l)
m (12) - b (1) = 11 (l)
q (16) - c (2) = 14 (o)

- **Conversion en caractères alphabétiques**

  7 = h
  4 = e
  11 = l
  11 = l
  14 = o
  Message déhiffré
  Le message déchiffré est donc "hello".

CONCLUSION

L'entrelacement entre les codes correcteurs d'erreurs et la cryptographie représente une fusion stratégique entre deux domaines essentiels de la sécurité des communications. Initialement développés pour garantir l'intégrité des données lors des transferts numériques, les codes correcteurs d'erreurs ont élargi leur champ d'application pour renforcer la confidentialité et la sécurité des échanges cryptographiques. Leur capacité à détecter et à corriger les altérations accidentelles ou malveillantes contribue de manière significative à la robustesse des systèmes de cryptographie, rendant plus ardue toute tentative d'interception ou de manipulation des données.

Par ailleurs, les avancées récentes dans les techniques de correction d'erreurs, en particulier dans le domaine émergent des codes correcteurs d'erreurs quantiques, ouvrent de nouvelles perspectives pour la sécurité des communications quantiques. Ces progrès offrent la promesse de systèmes de communication ultra-sécurisés, basés sur les principes de la physique quantique, et capables de résister aux attaques même des ordinateurs quantiques.

En fin de compte, la collaboration étroite entre les chercheurs en théorie de l'information et en cryptographie continue de repousser les limites de la sécurité des communications dans un monde de plus en plus connecté et numérique. Cette synergie entre les codes correcteurs d'erreurs et la cryptographie est un pilier essentiel de l'infrastructure de sécurité moderne, garantissant la confidentialité et l'intégrité des informations échangées à l'ère de la communication numérique.

BIBLIOGRAPHIE

[1] Philippe GABORIT, Professeur des Universités, Université de Limoges-Gilles ZEMOR, Professeur des Universités, Université de Bordeaux- Cryptographie à base de codes correcteurs d'erreurs en métrique rang et applications[Lien]

[2] Alexander Barg,Thierry Berger Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique[Lien]

[3] Daniel LAMAS- Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES [Lien]

[4] Nicolas Aragon- Cryptographie à base de codes correcteurs d'erreurs en métrique rang et application[Lien]

[5] Grini Abdelâli -Utilisation des codes correcteurs d'erreurs en cryptographie à clef publique[Lien]