Table des matières

Liste des Algorithmes

Liste des figures

Liste des tableaux

Liste des Abréviations

AES: Advanced Encryption Standard

BCH: Bose, Chaudhuri et Hocquenghem

BIKE: Bit-flipping Key Encryption

CCA: Chosen Ciphertext Attack

CPA: Chosen Plaintext Attack

DES: Data Encryption Standard

DFR: Decoding Failure Rate

GJS: Guo, Johannson et Stankovski

IND-CCA: Indistinguishability under Adaptative Chosen ciphertext Attacks

LDPC: Low Density Parity Check

MDPC: Moderate Density Parity Check

NIST: National Institute of Standards and Technology

QC-LDPC: Quasi-cyclic Low Density Parity Check

QC-MDPC: Quasi-cyclic moderate Density Parity Check

RC4: Rivest Cipher 4

RSA: Rivest Shamir Adleman

TLS: Transport Layer Security

TTL: Time-to-live (ttl)

Résumé

En cryptographie, les techniques les plus utilisées sont ceux basées sur des problèmes de théories des nombres et de logarithmes discrets. Cependant en 1994, Peter Shor, d'AT&T Research Labs, introduisit un algorithme quantique qui peut, en temps polynomial probabiliste, factoriser de grands nombres entiers et résoudre le logarithme discret sur machine quantique. C'était une percée spectaculaire car il a présenté l'un des premiers exemples de scénario dans des techniques quantiques.

Dès lors, la communauté scientifique à penser à de nouvelles primitives capables de résister à la machine quantique: cette nouvelle famille est appelée cryptographie post-quantique.

Ainsi, plusieurs pistes sont explorées, notamment la cryptographie basée sur les codes. Le premier cryptosystème basé sur les codes correcteurs d'erreurs fut introduit par McEliece. Celui-ci s'appuyant sur des problèmes difficiles de théorie des codes ne présente pas cette vulnérabilité face à l'ordinateur quantique. D'origine, McEliece utilise les codes de Goppa qui sont des codes très bien structurés mais présentent une taille de clés très grande, ce qui fait qu'ils sont très peu utilisés en pratique. Il était donc impératif de réduire la taille des clés de ce cryptosystème. Les récentes propositions suggèrent l'utilisation des codes QC-MDPC qui offrent une taille de clés très raisonnable pour le schéma de McEliece.

Cependant, le principal algorithme de décodage de ces codes MDPC (le *Bit-flipping*) présente une probabilité d'échec au décodage (DFR) qui sera exploitée dans l'attaque de GJS. Cette attaque exploite le spectre de distance pour récupérer la clé sécrète. Il est donc nécessaire d'améliorer le décodage pour sécuriser le cryptosystème. Ainsi, plusieurs variantes de l'algorithme *Bit-flipping* ont été proposées : il s'agit du *Black-Gray* et du *Back-flipping*.

Nous avons pour notre part travailler sur un algorithme, qu'on a proposé, combinant le *Black-Gray* et le *Bit-flipping*. En effet, il a été démontrer que le Black-Gray est plus performant que le Back-flipping lorsqu'il s'agit d'une implémentation en temps constant, même si ce dernier offre le DFR le plus bas. Ainsi, l'idée est de comparer le DFR de notre algorithme avec celui du *Black-Gray* et du *Back-flipping*.

Abstract

In cryptography, the most used techniques are those based on problems of number theory and discrete logarithms. However, in 1994, Peter Shor, from AT&T Research Labs, introduced a quantum algorithm that can, in probabilistic polynomial time, factor large integers and solve the discrete logarithm on a quantum machine. It was a spectacular breakthrough as it presented one of the first examples of scenario in quantum techniques.

From then on, the scientific community thought of new primitives capable of resist the quantum machine: this new family is called post-quantum cryptography.

Thus, several avenues are explored, including code-based cryptography. The first cryptosystem based on error-correcting codes was introduced by McEliece. This one, based on difficult problems of code theory, does not present this vulnerability to the quantum computer. Originally, McEliece uses Goppa codes which are very well structured codes but have a very large key size, so they are very little used in practice. It was therefore imperative to reduce the key size of this cryptosystem. Recent proposals suggest the use of QC-MDPC codes that offer a very reasonable key size for the McEliece scheme.

However, the main decoding algorithm for these MDPC codes (*Bit-flipping*) has a decoding failure probability (DFR) that will be exploited in the GJS attack. This attack exploits the distance spectrum to recover the secret key. It is therefore necessary to improve decoding to secure the cryptosystem. Thus, several variants of the *Bit-flipping* algorithm have been proposed: *Black-Gray* and *Back-flipping*.

For our part, we have worked on an algorithm, which we have proposed, combining *Black-Gray* and *Bit-flipping*. Indeed, it has been demonstrated that *Black-Gray* is more efficient than *Back-flipping* when it is a constant time implementation, even if the latter offers the lowest DFR. So, the idea is to compare the DFR of our algorithm with that of *Black-Gray* and *Back-flipping*.

Introduction Générale

Etymologiquement, la cryptologie est la science du secret. Le concept est apparu dans l'antiquité, notamment avec le chiffrement de César. La cryptologie comprend la cryptographie, domaine qui regroupe l'ensemble des techniques et méthodes utilisées pour transformer un message clair en un message inintelligible, et la cryptanalyse, domaine où l'on cherche des failles pour retrouver de l'information, partielle ou totale, à partir du message crypté. Dans le passé, la cryptographie et la cryptanalyse ont été largement utilisées pour et par les armées afin d'assurer la confidentialité dans les transmissions, mais également de collecter des informations sensibles sur leurs adversaires. Récemment, avec le développement des technologies de l'information et de la communication, la cryptographie est devenue incontournable.

Cependant, l'avènement de l'ordinateur quantique fait naître beaucoup d'inquiétudes pour la communauté cryptographique. En effet, les capacités de calcul d'un tel ordinateur permettraient de résoudre certains problèmes plus rapidement que les ordinateurs actuels, y compris ceux sur lesquels se fonde la cryptographie moderne. Il urge alors de trouver des solutions, qui, si l'ordinateur quantique est construit, permettraient toujours d'assurer la confidentialité. Et depuis, plusieurs domaines cryptographiques sont explorés, notamment les codes correcteurs d'erreurs. Ces derniers se fondent sur un principe simple: on ajoute de la redondance dans le message à transmettre et on obtient ainsi un message plus long, mais dont l'information excédentaire peut être exploité pour détecter voire corriger des erreurs de transmission.

En 1978, McEliece propose le premier cryptosystème à clé publique basé sur les codes correcteurs d'erreurs [1]. Ce cryptosystème a été introduit initialement en utilisant les codes de Goppa qui, étant indistinguables de codes aléatoires, offrent une bonne résistance aux attaques structurelles. Cependant, les codes de Goppa ont une capacité de correction très faible et nécessitent des clés publiques de taille très importantes pour être viables.

Afin de résoudre ce problème de la taille de la clé, Misoczki, Tillich, Sendrier et Barreto préconisent l'utilisation des codes quasi-cycliques MDPC (Moderate Density Parity Check) pour le schéma de McEliece [2]. Lorsqu'on ajoute une structure quasi-cyclique aux codes MDPC, on obtient ainsi des codes qui sont représentés uniquement à partir de la première ligne de leur matrice de définition. Ces codes sont appelés codes quasi-cycliques MDPC, qu'on notera QC-MDPC.

Ce manuscrit est composé de quatre chapitres. Le premier chapitre abordera les généralités sur la cryptographie et la théorie des codes correcteurs d'erreurs. Le deuxième chapitre sera d'abord consacré à l'étude du schéma original de McEliece utilisant les codes de Goppa. Ensuite, nous aborderons dans ce chapitre les codes QC-MDPC. Et enfin, nous appliquerons ces derniers au schéma de McEliece. Le troisième chapitre sera consacré aux attaques connues sur le schéma de McEliece instancié aux codes QC-MDPC. Une étude exhaustive sur ces attaques sera ainsi faite dans ce chapitre. Le dernier chapitre sera consacré à l'analyse des algorithmes de décodage des codes QC-MDPC.

Chapitre 1

Généralités sur la cryptographie et les codes correcteurs d'erreurs

Ce chapitre sera l'occasion d'aborder les notions de cryptographie et de codes correcteurs d'erreurs. Ces derniers sont très utilisés en cryptographie mais également en théorie de l'information et de la communication. Ce sont des codes qui ont pour objectif de permettre la transmission de l'information malgré l'ajout éventuel d'erreurs lors de cette transmission. Afin d'y parvenir, on ajoute une redondance au message à transmettre qui, lorsqu'un nombre suffisamment faible d'éléments de ce message étendu est perdu ou altéré, permettra de reconstituer le message initialement envoyé. Cette reconstitution est appelée le décodage.

1.1. Généralités sur la cryptographie

1.1.1. Définitions

La cryptographie vient du grec *kryptos* qui veut dire caché et de *graphien* qui signifie écrire. La cryptographie est donc un ensemble des techniques permettant de protéger une communication au moyen d'un code graphique secret.

Elle est l'étude des méthodes d'envoi de messages codés de telle sorte que seul le destinataire puisse le décoder. Le message qu'on veut envoyer est appelé le **texte clair** et le message codé, ou **encrypté**, est également appelé **cryptogramme**.

Le processus de conversion d'un texte clair en message codé s'appelle **chiffrement**, ou codage. Et le processus inverse est appelé **déchiffrement**, ou décodage. Pour effectuer un codage, on suit une méthode précise appelée système cryptographique, ou **cryptosystème**. Un codage se fait donc à l'aide d'un cryptosystème, et celui-ci nécessite très souvent l'utilisation d'une clé.

***Figure 1**: Schéma général de la cryptographie*

La cryptanalyse est l'étude des méthodes qui permettent de découvrir le sens d'un message codé, sans connaître le message original. Il y a plusieurs situations possibles. On peut vouloir simplement trouver le sens du message codé, sans chercher la clé de codage. Mais, en général on voudra d'abord trouver quel est le système de codage, puis la clé de codage utilisée. Lorsqu'on a trouvé tous les éléments de la méthode utilisée pour coder les messages, on dit qu'on a cassé, ou brisé, le système cryptographique en question. Plus un système est « difficile » à briser, plus il est qualifié de « sûr ».

La cryptologie est l'ensemble formé de la cryptographie et de la cryptanalyse. Elle fait partie d'un ensemble de théories et techniques liées à la transmission de l'information (théorie des ondes électromagnétiques, théorie du signal, théorie de l'information, …).

1.1.2. Principe de Kerckhoff

Dans son article *La cryptographie militaire* [7], Auguste Kerckhoff fustige le manque de précaution lors de l'utilisation des techniques cryptographiques dans les armées.

Cependant, la partie la plus connue de cet article est de *Desiderata de la cryptographie militaire*, où il énonce six conditions pour avoir un système cryptographique fiable et utilisable par les armées. De ces six points, le plus important est le second: «il faut qu'il (le système cryptographique) n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ».

Autrement dit, tout système cryptographique devrait avoir sa fiabilité du fait de la difficulté de trouver sa clé, et non du secret de l'algorithme. De plus, en rendant les algorithmes publics, les cryptanalystes du monde entier peuvent essayer de le casser ou l'améliorer.

1.1.3. Enjeux de la cryptographie

Dès que plusieurs entités sont impliquées dans un échange de messages sécurisés, des règles doivent déterminer l'ensemble des opérations cryptographiques à réaliser, leur séquence, afin de sécuriser la communication.

Ainsi, lorsque l'on parle de ‘‘sécuriser un échange’’, on souhaite prêter attention aux trois services suivants:

La confidentialité: seulement le destinataire autorisé doit être capable d'extraire le contenu du message de son état crypté. Par ailleurs, l'obtention de l'information à propos du contenu du message ne doit pas être possible.

L'intégrité: Par analyse de l'intégrité le destinataire peut déterminer si le message a été modifié pendant la transmission.

L'authentification: le destinataire doit avoir la capacité d'identifier le destinataire du message, aussi bien que de savoir si c'est l'auteur présumé qui a en effet envoyé le message.

1.1.4. Notion de chiffrement

Dans la cryptographie moderne, toute sécurité est basée sur la clé (ou les clés), et non dans les détails des algorithmes. Cela signifie qu'un algorithme peut être publié et analysé, mais la clé doit être protégée [8].

Ainsi, dans les systèmes cryptographiques utilisant des clés, il faut bien distinguer les deux types existants: le chiffrement symétrique et le chiffrement asymétrique.

1.1.4.1. Chiffrement symétrique

Le chiffrement symétrique (aussi appelé chiffrement à clé privée, ou à clé secrète) se base sur l'utilisation d'une clé qui doit rester secrète et qui ne doit être connue que par les personnes devant crypter et décrypter les messages. La sécurité de la méthode de chiffrement réside donc dans la difficulté à trouver cette clé.

***Figure 2**: Chiffrement symétrique*

Les algorithmes de chiffrement symétriques sont très rapides en termes de calcul. Cependant, leur principal inconvénient est le problème de la distribution de la clé entre l'expéditeur et le destinataire.

On peut distinguer deux types de chiffrements symétriques :

- Le chiffrement par flux qui se fait bit à bit sans attendre la réception entière de données. Le **RC4** (Rivest Cipher 4) est l'algorithme le plus connu pour ce type de chiffrement.
- Le chiffrement par bloc qui consiste à diviser les données en blocs de taille généralement fixe (entre 64 et 128 bits). Les blocs sont ensuite chiffrés les uns après les autres. Les algorithmes les plus répandus de ce type de chiffrement sont : le **DES** (Data Encryption Standard) et l'**AES** (Advanced Encryption Standard) qui est actuellement l'algorithme le plus utilisé et le plus sûr.

1.1.4.2. Chiffrement asymétrique

Inventé en 1977 par Diffie et Hellmann [9], le chiffrement asymétrique, aussi connu sous le nom de chiffrement à clé publique, élimine la problématique de la transmission de la clé du chiffrement symétrique.

Le principe de ce chiffrement est basé sur l'usage d'un couple de clés, l'une publique qui est connu par tout le monde et l'autre privée qui doit rester confidentielle.

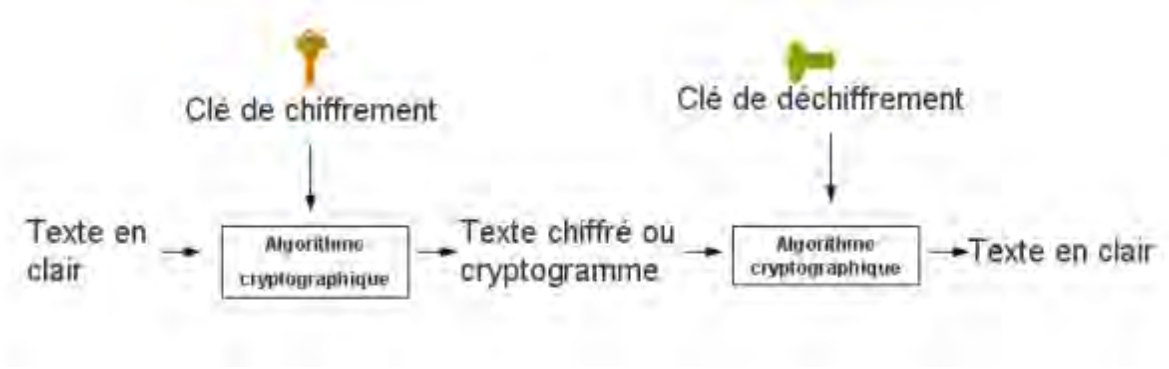

***Figure 3**: Chiffrement asymétrique*

Au niveau des performances, le chiffrement asymétrique est 1000 fois plus lent que le chiffrement symétrique. Cependant, avec le chiffrement asymétrique, la distribution des clés

est grandement facilitée car l'échange des clés secrètes n'est plus nécessaire. Chaque utilisateur conserve sa clé secrète sans jamais la divulguer. Seule la clé publique devra être distribuée.

Les algorithmes les plus fréquents du chiffrement asymétrique sont : le **RSA** (Rivest Shamir Adleman) [10] qui est très utilisé dans le commerce électronique et plus généralement pour échanger des données confidentielles sur internet, **ElGamal**, **Merkle-Hellman**, etc.

1.1.5. Fonction de hachage

Les fonctions de hachage sont des fonctions qui prennent en entrée une chaine de bits de longueur arbitraire et produisent un résultat de taille fixe appelé empreinte ou hash. Elles sont en général utilisées dans les signatures numériques [11].

Les fonctions de hachage doivent être résistantes aux collisions, c'est-à-dire qu'on ne peut pas trouver facilement deux messages différents ayant la même empreinte. Bien évidemment, la taille de l'empreinte étant fixe, il existera toujours des collisions du fait qu'il y a un nombre limité d'empreintes alors que le nombre de messages initiaux est illimité.

1.2. Codes correcteurs d'erreurs

1.2.1. Théorie des codes

En 1948, dans son article intitulé ''*A mathematical theory of communication*'', Claude Shannon posa les bases de ce que l'on appelle aujourd'hui la théorie de l'information [12]. La théorie de l'information décrit les aspects les plus fondamentaux des systèmes de communication, essentiellement à l'aide de la théorie des probabilités. Elle étudie les moyens de transmettre un message depuis une source (voix, signal numérique, onde électromagnétique, ...) jusqu'à un destinataire et via un canal de transmission (liaison radio, ligne téléphonique, ...).

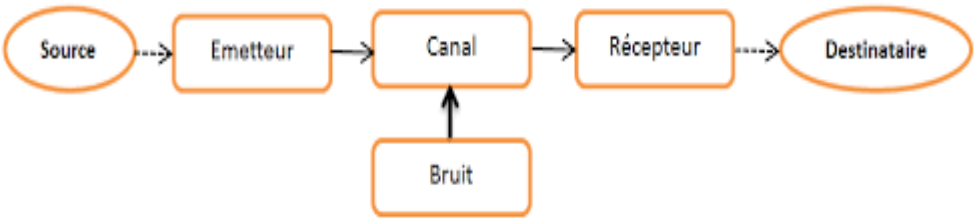

***Figure 4**: Synoptique d'un système de communication*

Cependant, lorsqu'une information est transmise à travers un canal, celui-ci génère des erreurs ou des effacements. Le canal est alors bruité. Ce problème sera résolu en 1950, lorsque Richard W. Hamming, dans son article ''*Error detecting and error correcting codes*'' introduisit la théorie des codes [13]. Hamming proposa ainsi les premiers codes correcteurs d'erreurs. Ces derniers sont un outil qui permet d'assurer la transmission fiable d'informations à travers un canal. L'idée étant de rajouter de la redondance au message transmis, de sorte que cet excèdent d'information aide à retrouver le message initial. Une grande famille des 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.

1.2.2. Canal binaire symétrique

La modélisation des canaux est un enjeu majeur pour la théorie de l'information afin de comprendre les propriétés du bruit qu'ils vont produire et de l'estimer. En effet, le choix de l'algorithme de décodage et le bon fonctionnement de celui-ci dépendent du modèle du canal utilisé.

Dans notre manuscrit, nous allons considérer le modèle du canal binaire symétrique. C'est un canal binaire caractérisé par la probabilité d'erreur p qu'au cours de la transmission un bit (0 ou 1) soit modifié en son opposé. Ces modifications se produisent indépendamment sur chacun des bits transmis. Nous illustrons ce canal par la figure ci-dessous.

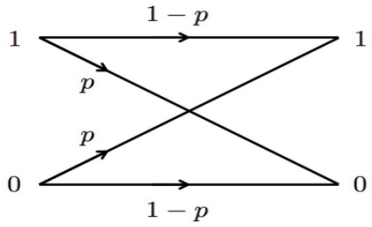

***Figure 5**: Canal binaire symétrique de probabilité d'erreur p*

1.2.3. Codes linéaires

1.2.3.1. Définition et exemple

Soit $\mathbb{F}$ un corps fini et $n > 0$.

Un **code linéaire binaire** $\mathcal{C}$ de longueur n et de dimension k est un sous-espace vectoriel engendré par k vecteurs libres de $\mathbb{F}_2^n$ [15]. Le nombre de mots d'un tel code est donc 2^k. On notera ce code un $[n, k]$-code linéaire.

En d'autres termes, un code linéaire est un ensemble de vecteurs de longueurs n, qui est stable par addition et par multiplication par un scalaire, dont une base contient k vecteurs linéairement indépendants.

Exemple : Le code {0000, 1101, 0110, 1011} est un $[n, k]$-code linéaire binaire de longueur 4 et de dimension 2.

Le poids de Hamming de v, noté $wt(v)$ est le nombre de ses coefficients non nuls.

La distance de Hamming entre v et v' est définie par :

$$d(v, v') = wt(v' - v).$$

La distance minimale d'un code est la plus petite distance entre deux mots distincts du code.

Parallèlement, nous définissons le support d'un vecteur comme les indices des coordonnées non nulles de celui-ci.

Le support de $v \in \mathbb{F}_2^n$ est défini comme :

$$Supp(v) = \{j : v_j = 1\} \subseteq \{1, \dots, n\}$$

Le poids de Hamming d'un vecteur $v \in \mathbb{F}_2^n$ se définit de manière équivalent comme :

$$wt(v) = |Supp(v)|.$$

Un code linéaire, étant un sous-espace vectoriel, peut être décrit grâce à une base de vecteurs représentée sous forme matricielle. Ainsi, on distingue la matrice génératrice et la matrice de contrôle de parité.

1.2.3.2. Matrice génératrice

Soit $\mathcal{C}$ un [n, k]-code linéaire binaire. On appelle **matrice génératrice** de $\mathcal{C}$, notée G, toute matrice de taille $k \times n$ dont les lignes forment une base de $\mathcal{C}$. On a alors :

$$\mathcal{C} = \{\vartheta \mathrm{G} \mid \vartheta \in \mathbb{F}_2^k \}$$

La matrice G définie une bijection de $\mathbb{F}_2^k \longrightarrow \mathcal{C}$ par $\vartheta \longmapsto \vartheta G$ et ainsi on représente 2^k messages distincts par des mots du code $\mathcal{C}$. Chaque mot de longueur k est codé par un mot du code $\mathcal{C}$ de longueur n.

Une matrice génératrice G d'un code $\mathcal{C}$ n'est pas unique. Puisque les k colonnes de G sont linéairement indépendantes, en effectuant des opérations élémentaires sur les lignes, G peut être transformée en $G^* = (I_k|A)$, I_k est la matrice identité d'ordre k et A une matrice $k \times (n - k)$.

G^* est appelée matrice génératrice canonique de $\mathcal{C}$.

Ainsi, lorsqu'un code linéaire $\mathcal{C}$ possède une matrice génératrice de la forme $G^* = (I_k|A)$, on dit que le code $\mathcal{C}$ est **systématique** [16].

1.2.3.3. Code dual

Soit $\mathcal{C}$ un [n, k]-code linéaire binaire.

Le code dual (ou code orthogonal), noté $\mathcal{C}^{\perp}$, est l'espace-vectoriel orthogonal de $\mathcal{C}$ pour le produit scalaire sur $\mathbb{F}_2^n$ [18]. On note :

$$\mathcal{C}^{\perp} = \{x \in \mathbb{F}_2^n \mid x.y = 0, \forall\, y \in \mathcal{C}\}$$

1.2.3.4. Matrice de contrôle de parité

Soit $\mathcal{C}$ un [n, k]-code linéaire binaire.

On appelle matrice de contrôle de parité de $\mathcal{C}$, toute matrice génératrice du code dual $\mathcal{C}^{\perp}$, notée H. On a :

$$\mathcal{C} = \{x \in \mathbb{F}_2^n \mid x.H^t = 0\}$$

La matrice de contrôle de parité, comme son nom l'indique, permet le contrôle d'erreurs. Ainsi, pour toute matrice génératrice G, il existe une matrice de contrôle H, telle que les lignes de H soient orthogonales aux lignes de la matrice G. Autrement dit, GH^t = 0.

Comme pour la matrice génératrice G, on peut écrire la matrice de contrôle de parité sous forme systématique. Dans ce cas on a :

$$H = (-A^t | I_{n-k})$$

1.2.3.5. Fonction syndrome

Soit H une matrice de contrôle de parité d'un code $\mathcal{C}$, le syndrome $s \in \mathbb{F}_2^{n-k}$ de tout vecteur $x \in \mathbb{F}_2^n$ est définit comme suit :

$$s = H.x^T$$

Remarque : En multipliant un mot de code du code $\mathcal{C}$ par sa matrice de contrôle de parité H, on obtient le vecteur, c'est-à-dire : $\forall\, c \in \mathcal{C},\ Hc^T = 0$.

1.2.4. Quelques exemples de codes linéaires

1.2.4.1. Code de Hamming

Un code de Hamming est un code correcteur linéaire. Il est utilisé dans les transmissions de données car il permet de détecter et de corriger une erreur survenue dans un bloc transmis [17].

Soit m un entier positif supérieur ou égal à 2. **Un code de Hamming binaire**, dit de paramètre r, est un code linéaire défini sur $\mathbb{F}_2$, de longueur $n = 2^r - 1$, de dimension $k = 2^r - 1 - r = n - r$ et de distance minimale $d = 3$, Les colonnes d'une matrice de contrôle d'un code de Hamming binaire sont tous les vecteurs non nuls de $\mathbb{F}_2^r$ [18].

Le code de Hamming est un code parfait, ce qui signifie que pour une longueur de code donnée, il n'existe pas d'autre code plus compact ayant la même capacité de correction. En ce sens, son rendement est maximal.

1.2.4.2. Codes de Goppa classique

En 1970, le mathématicien russe, Valery Denisovich Goppa découvre la relation entre la géométrie algébrique et les codes. Ce qui a pour conséquence la naissance des codes dits de Goppa classique, définis sur les corps finis [20].

Définition : Soient $\mathcal{L} = \{\alpha_0, \dots, \alpha_{n-1}\} \in \mathbb{F}_q^n$ et $G(X) \in \mathbb{F}_q[X]$ un polynôme de degré t tels que $\forall\, i \in \{0,2, \dots, n-1\}$, $g(\alpha_i) \neq 0$. Le code de Goppa de longueur n défini sur $\mathbb{F}_q$ est donné par :

$$\Gamma_q(\mathcal{L}, G) = \{C = (c_0, c_2, \dots, c_{n-1}) \in \mathbb{F}_q^n : \sum_{i=0}^{n} \frac{c_i}{X - \alpha_i} \equiv 0 \ mod\ G(X)\}$$

On dit alors que $\mathcal{L}$ est le support et G le polynôme de Goppa de $\Gamma_q(\mathcal{L}, G)$.

Grâce à la précédente équation, on peut retrouver la matrice de contrôle du code de Goppa à partir de la matrice suivante :

$$H = \begin{pmatrix} \frac{1}{g(\alpha_0)} & \cdots & \frac{1}{g(\alpha_{n-1})} \\ \vdots & \vdots & \vdots \\ \frac{\alpha_0^{t-1}}{g(\alpha_0)} & \cdots & \frac{\alpha_{n-1}^{t-1}}{g(\alpha_{n-1})} \end{pmatrix}$$

Nous verrons dans le chapitre suivant que les codes de Goppa sont très utilisés en cryptographie, notamment dans le cryptosystème de McEliece.

1.2.4.3. Codes cycliques

Les codes cycliques constituent l'une des classes les plus importantes des codes linéaires, et sont les plus utilisés en pratique. En effet, ils présentent de nombreux avantages : leur mise en œuvre (codage/décodage) est facile, ils offrent une gamme étendue de codes (comme les codes quasi-cycliques, qu'on abordera dans cette partie), et enfin permettent de corriger différent types d'erreurs.

a. Définition

Un code $\mathcal{C}$ de longueur n est dit cyclique s'il est linéaire et s'il vérifie la propriété suivante :

$$(c_0, c_1, \dots, c_{n-1}) \in \mathcal{C} \iff (c_{n-1}, c_0, c_1, \dots c_{n-2}) \in \mathcal{C}$$

En d'autres termes, dans un code cyclique, tout décalage cyclique d'un mot de code forme encore un mot de code [15].

Exemple : Le code $\mathcal{C}$ = {000, 110, 011, 101} est un code binaire cyclique.

b. Représentation polynomiale

Pour faciliter l'écriture et afin d'étudier les propriétés algébriques de ces codes, il est plus commode d'écrire les mots d'un code cyclique sous forme polynomiale.

Soit $\mathcal{C}$ un code linéaire de longueur n, à chaque mot c de $\mathcal{C}$, on associe un polynôme $c(x)$ de $\mathbb{F}_2[X]$ défini par :

$$c = (c_o, c_1, \dots, c_{n-1}) \longrightarrow c(X) = c_0 + c_1 X + c_2 X^2 + \cdots + c_{n-1} X^{n-1}$$

Ainsi, l'action du décalage circulaire sur un mot de code revient à la multiplication par X modulo $X^n - 1$ sur le polynôme correspondant :

$$(c_{n-1}, c_0, \dots, c_{n-2}) \longrightarrow c(X) = c_{n-1} + c_0 X + \cdots + c_{n-2} X^{n-1}$$

Avec,

$$c_{n-1} + c_0X + \cdots + c_{n-2}X^{n-1} = X.c(X)[\text{modulo } (X^n - 1)]$$

Autrement dit, le polynôme $c_{n-1} + c_0X + \cdots + c_{n-2}X^{n-1}$ est égal au reste du polynôme $X.c(X)$ dans la division euclidienne par $X^n - 1$ [19].

c. Matrice circulante et quasi-circulante

Une matrice carrée est dite circulante si elle est construite de sorte que ses lignes soient les décalages cycliques successifs de la première ligne [15].

Une telle matrice s'écrit comme suit :

$$M = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_n \\ a_n & a_0 & a_1 & \cdots & a_{n-1} \\ a_{n-1} & a_n & a_0 & \cdots & a_{n-2} \\ & & & \ddots & \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{bmatrix}$$

La matrice circulante est donc entièrement définie par la donnée d'une seule de ses lignes.

Remarque : Un code cyclique admet au moins une matrice génératrice (ou de contrôle de parité) circulante.

Une matrice est dite quasi-circulante lorsqu'elle est formée de blocs circulants. Une telle matrice s'écrit comme suit :

$$M = \begin{bmatrix} a_0 & a_1 & a_2 & b_0 & b_1 & b_2 \\ a_2 & a_0 & a_1 & b_2 & b_0 & b_1 \\ a_1 & a_2 & a_0 & b_1 & b_2 & b_0 \end{bmatrix}$$

On remarque que cette matrice est formée de deux blocs circulants. On a donc :

$M = [A \quad | \quad B]$, avec

$$A = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_2 & a_0 & a_1 \\ a_1 & a_2 & a_0 \end{bmatrix} \quad et \quad B = \begin{bmatrix} b_0 & b_1 & b_2 \\ b_2 & b_0 & b_1 \\ b_1 & b_2 & b_0 \end{bmatrix}$$

d. Codes quasi-cycliques

Au sein des codes correcteurs d'erreurs et de la théorie de l'information, les codes quasi-cycliques tiennent une place particulière. Ces codes sont une généralisation des codes cycliques et possèdent en plus d'excellents paramètres : ils ont une grande capacité de correction.

Définition : Un code est dit quasi-cyclique lorsqu'il admet une matrice génératrice (ou de contrôle de parité) quasi-circulante.

Remarque : Un code quasi-cyclique admet au moins une matrice génératrice sous forme systématique et quasi-circulante.

Dans les chapitres suivant de notre manuscrit, on utilisera les codes quasi-cycliques qu'on va appliquer à la cryptographie.

1.2.4.4. Autres codes

Il existe beaucoup d'autres codes linéaires (les codes **BCH**, les codes **Reed-Muller**, les codes **Reed-Solomon**, les codes **alternants**, les codes **Golay**, etc.) toutes présentant leurs intérêts, soit du point de vue de la capacité de correction d'erreurs, soit de la simplicité du codage ou du décodage.

On ne fait ici que mentionner leur existence car leur connaissance n'est pas nécessaire à la compréhension de notre manuscrit. Il n'est pas en revanche exclu que des applications cryptographiques de la théorie algébrique des codes puissent passer par l'utilisation de ces codes.

Chapitre 2

Cryptosystème de McEliece et codes QC-MDPC

Le chapitre précèdent a été l'occasion de voir les généralités sur la cryptographie mais également d'introduire les codes correcteurs d'erreurs. Ainsi, dans ce chapitre on va s'intéresser, d'abord, à un système de chiffrement cryptographique basé sur les codes correcteurs d'erreurs : le cryptosystème de McEliece [1]. Puis, on va étudier les codes MDPC (Moderate Density Parity Check) quasi-cycliques et enfin appliquer ce type de codes au schéma de McEliece.

2.1. Le cryptosystème de McEliece

En 1978, McEliece eu l'idée d'utiliser la théorie des codes correcteurs d'erreurs à des fins cryptographiques, et plus précisément pour un algorithme de chiffrement asymétrique. Ainsi, le premier cryptosystème à clé publique basé sur les codes correcteurs d'erreurs est né. L'inconvénient majeur de ce cryptosytème est la taille des clés publiques et privées relativement grande. C'est pourquoi le chiffrement de McEliece a été mis de côté pendant de nombreuses années.

Par ailleurs, la plupart des cryptosystèmes utilisés aujourd'hui sont basés sur des problèmes issus de la théorie des nombres comme : la factorisation d'un entier en ses diviseurs premiers pour RSA [10], la résolution du logarithme discret dans un corps fini [9] ou encore sur des courbes elliptiques. Cependant, en 1999, Peter W. Shor dans son article ‘‘*Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer’’* a explicité des algorithmes pouvant résoudre ces problèmes avec un ordinateur quantique [21]. Le niveau de sécurité de ces cryptosystèmes est donc remis en question si toute fois un tel ordinateur voit le jour. Il était donc nécessaire de trouver un cryptosystème capable de résister à l'ordinateur quantique. Ainsi, un regain d'intérêt pour le cryptosystème de McEliece est apparu ces dernières années, car les chiffrements basés sur les codes font partie de la courte liste des cryptosystèmes supposés résister à l'ordinateur quantique.

Le cryptosystème de McEliece étant un chiffrement à clé publique, Nous allons d'abord voir comment générer ses différentes clés, ensuite l'étape de chiffrement, puis celle de déchiffrement et enfin sa sécurité.

2.1.1. Génération de clés

L'article original de McEliece qui présente son cryptosystème utilise les codes de Goppa classiques binaires [1]. Et à l'heure actuelle, cette version résiste toujours aux attaques connues, à taille des paramètres près [22].

L'algorithme de la génération de clés se présente comme suit :

Algorithme 1 : Génération des clés de McEliece

Entrées : Un $[n, k]$-code C capable de corriger t erreurs.

Sorties : Une clé publique K_P et une clé secrète K_S,

Début

$G \longleftarrow$ Une matrice $k \times n$ génératrice du code C,

$S \longleftarrow$ Une matrice aléatoire inversible $k \times k$,

$P \longleftarrow$ Une matrice de permutation $n \times n$, choisie aléatoirement,

$G' \longleftarrow SGP$,

$K_P \longleftarrow (G', t)$,

$K_S \longleftarrow (S^{-1}, G, P^{-1})$,

Retourner (K_P, K_S).

On remarque ainsi que la clé publique est la matrice $G' = SGP$ et la clé privée est obtenue en calculant les inverses de la matrice de permutation P et de la matrice inversible S. La clé secrète est constituée ainsi des matrices $S^{-1}, G \; et \; P^{-1}$. Pour avoir une sécurité raisonnable, les tailles des matrices doivent être assez importantes, cela implique des tailles de clés conséquentes. C'est l'inconvénient principal du cryptosystème de McEliece, et c'est pour cela qu'aujourd'hui ce système de chiffrement n'est pas souvent utilisé dans la vie courante.

2.1.2. Chiffrement

Pour la phase de chiffrement, nous aurons besoin du message à chiffrer m et de la clé publique K_P. Il suffira alors de multiplier le message clair m par la matrice de la clé publique G' et de modifier aléatoirement t composantes du résultat. On ajoute une erreur de poids t au mot de code engendré par le message clair m. L'algorithme de chiffrement se présente ainsi :

Algorithme 2: Chiffrement de McEliece

Entrées : Le message clair m et la clé publique $K_P = (G', t)$,

Sorties : Le message chiffré m'.

Début

$c' \leftarrow mG'$,

$e \leftarrow$ Un vecteur aléatoire de poids t,

$m' \leftarrow c' + e$,

Retourner m'.

En 2008, Biswas et Sendrier, dans leur article [22], mettent au point une nouvelle idée pour l'implémentation de l'algorithme de chiffrement. En effet, ils proposent, au lieu de choisir aléatoirement un vecteur d'erreur, on pouvait utiliser ce dernier pour faire passer de l'information. En plus de cela, ils proposent d'utiliser la matrice génératrice sous sa forme systématique.

2.1.3. Déchiffrement

Pour le déchiffrement, nous utiliserons le message chiffré m' et la clé secrète K_S. Ainsi, le déchiffrement consiste à annuler l'effet de permutation, puis à décoder le mot de code obtenu pour supprimer les erreurs ajoutées durant le chiffrement. Cela revient donc à annuler l'effet de la matrice inversible S. L'algorithme de déchiffrement se présente ainsi :

Algorithme 3: Déchiffrement de McEliece

Entrées : Le message chiffré m' et la clé secrète $K_S = (S^{-1}, G, P^{-1})$.

Sorties : Le message clair m.

Début

$c'' \leftarrow m'P^{-1}$,

$m'' \leftarrow$ décoder c'',

$m \leftarrow m''S^{-1}$,

Retourner m.

On remarque ainsi que le déchiffrement repose essentiellement sur un algorithme de décodage. On peut alors vérifier qu'à partir des bonnes clés et un bon algorithme de chiffrement et de déchiffrement, on est bien en mesure de retrouver le message clair m. Soient $G' = SGP$ la matrice génératrice de la clé publique et e le vecteur d'erreur ajouté pendant le chiffrement.

On a: $$m' = mG' + e = mSGP + e,$$

$$m'P^{-1} = mSG + eP^{-1}.$$

Comme e est un vecteur d'erreurs de poids t et P une permutation, alors eP^{-1} est un autre vecteur d'erreur de poids toujours égal à t, donc :

$$dec(m'P^{-1}) = mS,$$

$$dec(m'P^{-1})S^{-1} = m.$$

Notons que la clé publique G' correspond à un $[n, k, 2t + 1]$-code linéaire qui est à permutation égale à la clé secrète choisie. Dans [1], McEliece utilise des codes de Goppa binaires pour lesquels un algorithme de décodage efficace a été présenté par Patterson [50]. Pour appliquer l'algorithme de Patterson, le polynôme générant le code de Goppa doit être connu. Par conséquent, dans le cas des codes de Goppa binaires, on peut considérer la clé publique comme étant $\big(S, G, P, g(X)\big)$, où $g(X)$ est le polynôme de Goppa pour le code choisi.

Par ailleurs, une dizaine d'années après la publication du cryptosystème de McEliece, Niederreiter a proposé un schéma similaire reposant sur une matrice de parité au lieu d'une matrice génératrice du code [23]. L'équivalence de la sécurité entre les deux cryptosystèmes a été montrée dans [24]. Sans perte de généralité, on présente uniquement le cryptosystème de McEliece.

2.1.4. Exemple

Soit G la matrice génératrice du code binaire C:

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

On choisit la matrice inversible S (matrice de brouillage) et la matrice de permutation P:

$$S = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \qquad et \qquad P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

On calcule ensuite la clé publique $G' = SGP$. Ainsi, on trouve :

$$G' = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Chiffrement : Soit $m = (111011101)$ le message à envoyer. Pour crypter ce message, on va le chiffrer par blocs : $m_1 = (111), m_2 = (011)\ et\ m_3 = (101)$. Si on suppose que le canal de transmission introduit une erreur simple de poids 1, au lieu d'envoyer le message chiffré $c = mG'$, c'est un autre message c' qui est envoyé :

$$c' = mG' + e, avec \quad e = (00100).$$

On obtient ainsi :

$$c'_1 = m_1G' + e = (00011),$$

$$c'_2 = m_2G' + e = (01010),$$

$$c'_3 = m_3G' + e = (10111).$$

D'où le message chiffré est :

$$c' = c'_1 + c'_2 + c'_3 = (000110101010111).$$

Déchiffrement : A la réception, on a : $m' = (11110)$.

On calcule donc : $c'' = m'P^{-1}$.

On a :

$$S^{-1} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \qquad et \qquad P^{-1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$c'' = m'P^{-1} = (11110) \times \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} = (11101).$$

Or

$$c'' = m'P^{-1}$$

$$(mG' + e) \times P^{-1}$$

$$(m \times G \times S \times P + e) \times P^{-1}$$

$$m \times S \times G + eP^{-1}$$

$$m \times S \times G + e',$$

Où $e' = eP^{-1}$ est un vecteur d'erreur de poids 1 (puisque P^{-1} est une matrice de permutation).

Le syndrome de c'' est :

$$c'' = m'(SG)^t$$

$$(101)$$

Puisque e' est de poids 1 et

$$(001) \times G' + e = c'' = (11110)$$

$$\Longrightarrow m = (001).$$

2.1.5. Avantages et inconvénient du cryptosystème de McEliece

Le cryptosystème de McEliece présente plusieurs avantages. On peut notamment mentionner que les opérations de chiffrement et de déchiffrement, correspondant à des opérations d'encodage et de décodage, peuvent être faites de manière très rapide. Mais surtout, comme il est basé sur un problème difficile lié à la théorie des codes linéaires, il est résistant aux attaques effectuées à l'aide d'ordinateurs quantiques permettant d'attaquer facilement des cryptosystèmes basés sur les nombres premiers ou le logarithme discret.

Cependant, le cryptosystème de McEliece est très peu utilisé en pratique pour plusieurs raisons, principalement à cause de sa clé publique qui est une matrice dont la taille est de $(nk)log_2(q^m)$. Cette taille est beaucoup trop importante par rapport aux tailles des clés publiques d'autres systèmes asymétriques : pour une sécurité de $2^{128} bits$, par exemple, le cryptosystème classique RSA nécessite une clé publique (représentant un grand nombre) de quelques milliers de bits alors que l'on dépasse le million de bits pour un cryptosystème de McEliece.

Ainsi pour résoudre ce problème de la taille de clés pour le cryptosystème de McEliece, plusieurs autres codes ont été proposés, notamment les codes quasi-cycliques MDPC sur lesquels va reposer notre travaille.

2.1.6. Problème difficile en théorie des codes

Les multiplications par une matrice inversible, puis par une matrice de permutation masquent la structure du code de Goppa, engendrant un code de Goppa équivalent, mais qui semble être un code aléatoire. Pendant très longtemps, on a pensé que la sécurité du cryptosystème de McEliece reposait sur la difficulté de distinguer un code de Goppa d'une matrice aléatoire et que l'on ne savait pas décoder un code sans en connaitre sa structure. En réalité, la première assertion est fausse. En effet, Faugère, Otmani, Tillich et Perret ont montré qu'on peut distinguer un code de Goppa avec un fort rendement d'un code aléatoire [49]. Même si l'on arrive à distinguer un code de Goppa d'un code aléatoire, on ne peut retrouver la matrice de permutation P et la matrice inversible S, c'est-à-dire retrouver la bonne structure du code de Goppa originalement utilisé. Sans la connaissance de cette structure, il est difficile de décoder ce code.

2.1.7. Sécurité du cryptosystème de McEliece

Un attaquant a deux approches pour tenter de décrypter le chiffrement de McEliece : soit il essaie une *attaque générique* en tentant de décoder un code quelconque sans en connaître la structure, ce qui est un problème difficile, soit il essaie une *attaque structurelle* en tentant dans ce cas de trouver une décomposition valide de la clé publique. Cette dernière attaque est dédiée aux variantes de McEliece qui utilisent des codes fortement structurés.

Attaque par décodage : Ces attaques consistent généralement à appliquer un algorithme de décodage sur le chiffré sans que la structure du code privé soit connue. La première proposition parlant de ce type d'attaque a été faite par McEliece lui-même comme méthode pour attaquer son propre cryptosystème, tous deux développés dans le même article [1]. Cette attaque porte le nom d'ISD (*Information Set decoding*) [51] (décodage par ensemble d'information, en français).

Attaque structurelle : Les attaques structurelles visent des variantes précises de McEliece. Ce type d'attaques tente de retrouver la clé secrète de la clé publique, en profitant au maximum de la structure des codes utilisés. Comme on a pu le voir précédemment, l'inconvénient majeur du cryptosystème de McEliece est la taille de ses clés. De ce fait, beaucoup d'auteurs ont essayé d'utiliser des codes très structurés, dans le but de limiter la clé publique à une sous-matrice de la matrice génératrice, qui permet d'en reconstruire le reste.

2.2. Les Codes quasi-cycliques MDPC

2.2.1. Etat de l'art

La famille de codes proposée par McEliece dans [1] est la famille des codes de Goppa puisqu'il existe un algorithme de décodage efficace et que leur distance minimale est relativement grande. De plus, les codes de Goppa sont indistinguables des codes aléatoires. Par cela, on entend qu'il est ‘‘calculatoirement’’ difficile de distinguer un code de Goppa d'un code aléatoire. Et comme on l'a vu précédemment, l'inconvénient majeur du schéma de McEliece est la taille des clés utilisées qui est relativement grande.

Pour résoudre ce problème relatif à la taille des clés publiques et privées, plusieurs propositions ont été faites en ce sens. Ainsi, en 1994, Vladmir M. sildenikov a proposé d'utiliser les codes de Reed-Muller [25], mais ce schéma de chiffrement fut attaqué dans [26]. En 1996, Janwa et Moreno dans [27] proposent l'utilisation des codes géométriques alternants, mais ce cryptosystème a lui aussi été prouvé non sûr dans [28] et dans [29].

Puis, la communauté cryptographique fait recourt aux codes très peu structurés pour résoudre le problème d'indistinguabilité du code public. C'est ainsi que dans [30] les auteurs étudient les codes LDPC qu'ils appliquent au schéma de McEliece. Cependant, ces codes possèdent de par leur définition de nombreux mots de poids faibles dans leur dual. Cette propriété sera exploitée comme distingueur par les mêmes auteurs.

Plus tard, certaines propositions ont porté sur l'utilisation des codes quasi-cycliques qui sont entièrement décrits grâce à une seule ligne de leur matrice génératrice (ou de parité). La première proposition ayant recourt aux codes quasi-cycliques est celle de Philippe Gaborit [31] qui suggère d'utiliser les sous codes BCH. Cependant, les auteurs de [32] ont explicité un algorithme qui permet de retrouver la clé privée à partir de la clé publique.

C'est après cela que Marco Baldi et Franco Chiaraluce présentent une nouvelle variante du cryptosystème de McEliece [33] avec des codes quasi-cycliques LDPC dont les mots de poids faibles dans le dual semblent être cachés. Mais la technique adoptée pour masquer la présence de ces mots a été exploité dans [32] pour retrouver la clé privée. En 2008, avec Barreto, ils proposent encore une autre variante du schéma de McEliece [34] et toujours avec les codes quasi-cycliques LDPC, en utilisant une nouvelle technique pour masquer les mots de poids faibles dans le dual.

Ces nombreuses propositions sur les codes quasi-cycliques LDPC prouvent que ces codes constituent un candidat sérieux pour la cryptographie post-quantique. Cependant, le problème principal de l'utilisation des codes LDPC quasi-cycliques est que les lignes de poids faibles de la matrice de contrôle de parité peuvent être considérées comme des mots de code de poids faibles dans le dual du code public. Ainsi, une attaque directe contre la variante LDPC utilisant le schéma de McEliece permet de trouver deux mots de codes de poids faibles et les utiliser pour construire une matrice de contrôle de parité. Telle est la conclusion de [35]. Ces mêmes auteurs trouvent également qu'avec cette variante quasi-cyclique, pour une sécurité de 80-bits, on obtient une clé publique dont la taille est de 12096 bits. C'est pour ces raisons que les auteurs de [2] proposent d'utiliser les codes MDPC quasi-cycliques.

Les codes MDPC sont définis de sorte que leurs propriétés soient adaptées à leur utilisation dans le cryptosystème de McEliece. Ces codes sont très peu structurés car ils sont simplement définis par une matrice de parité creuse dont les lignes ont un poids fixe (et modéré). Comparés aux codes LDPC, les codes MDPC sont encore assez bons pour empêcher l'efficacité des algorithmes de décodage standard.

Ainsi, toujours dans [2], les auteurs ont évalué la sécurité des codes quasi-cycliques MDPC. Et pour une sécurité de 80-bits, ils obtiennent une clé publique dont la taille est de 4801 bits.

2.2.2. Définitions

Les codes LDPC (Low Density Parity Check) ont été introduits en 1960 par Robert G. Gallager [35]. Ces codes sont constitués d'une matrice de contrôle de parité à densité faible. Pendant presque 35ans, les codes LDPC ont été oubliés par la communauté de la théorie des codes. Il faudra alors attendre jusqu'en 1996 pour voir David J. C. McKay et R. M. Neal [36] redécouvrir ces codes (dans un résultat apparemment indépendant du travail de Gallager). Ce travail sera

suivi de plusieurs autres, attestant leur excellente caractéristique de correction d'erreurs et de les promouvoir à la sélection au groupe de bons codes linéaires pour les applications de télécommunication.

Les codes MDPC (Moderate Density Parity Check), quant à eux, ont été introduits dans [2]. Ce sont des codes LDPC de densité plus élevé que ce qui est habituellement adopté pour les applications de télécommunication. Ce qui conduit en générale à une plus grande capacité de correction d'erreurs. Cependant, dans la cryptographie à base de codes, on ne s'intéresse pas nécessairement à corriger de nombreuses erreurs, mais seulement un certain nombre d'erreurs qui assure un niveau de sécurité adéquat, une condition que satisfait les codes MDPC. Ces derniers admettent une matrice de contrôle de parité à densité modérée, c'est-à-dire que les équations de parité définies par cette matrice sont de poids modéré.

Définition (Code LDPC)**:** *Un code* $[n, r, w]$-*LDPC est un code de longueur* n*, co-dimension* r *et qui admet une matrice de contrôle dont les lignes ont un poids fixe* w*, où* $w = O(1)$.

Définition (Code MDPC)**:** *Un code* $[n, r, w]$-*MDPC est un code de longueur* n*, co-dimension* r *et qui admet une matrice de contrôle dont les lignes ont un poids fixe* w*, où* $w = O(\sqrt{n})$.

Lorsqu'on ajoute la structure quasi-cyclique à un code $[n, r, w]$-$MDPC$, on notera ce code $[n, r, w]$-QC-$MDPC$.

2.2.3. Les codes QC-$MDPC$ pour le schéma de McEliece

Dans cette partie, nous allons utiliser les codes $[n, r, w]$-QC-$MDPC$ à des fins cryptographiques et principalement pour le cryptosystème de McEliece. Ces codes sont d'un intérêt particulier, puisque la propriété quasi-cyclique permet de représenter le code à utiliser par une unique ligne de sa matrice génératrice ou de parité. Ils possèdent également d'autres avantages. En effet, la taille des clés est très raisonnable, par exemple, pour le cryptosystème de McEliece et une sécurité de $80bits$ la clé publique est d'environ $5000bits$. Ensuite, les opérations nécessaires pour chiffrer, déchiffrer ou pour échanger une clé de session sont très simples et peu coûteuses, puisqu'elles correspondent, la plupart du temps, à des opérations binaires. Aussi, le formalisme polynomial de la description de ces codes peut être exploité pour diminuer le coût en mémoire et en temps des opérations [15].

Ainsi, nous allons ci-dessous décrire entièrement le schéma de cryptage QC-MDPC de McEliece. Chaque algorithme est décrit d'une manière assez générique afin de faciliter la compréhension.

2.2.3.1. Génération des clés

Construire un code $[n, r, w]$-QC-$MDPC$ revient à construire sa matrice de contrôle de parité. Ainsi, en considérant $n = n_0 r$, où r est premier (avec $r = n - k$), cette matrice sera de la forme :

$$H = [\, H_0 \mid H_1 \mid \ldots \mid H_{n_0-1} \,],$$

Où chaque H_i est lui-même une $r \times r$ matrice circulante. Pour construire une telle matrice de contrôle de parité, il suffit de générer sa première ligne. Cela se fait par sélections des moyennes aléatoires d'un vecteur binaire h de longueur n de poids $w = O(\sqrt{n})$:

$$h = [(h_0, h_1, \ldots, h_{n_0-1}), (h_{n_0}, h_{n_0+1}, \ldots, h_{2n_0+1}), \ldots, (h_{(r-1)n_0}, h_{(r-1)n_0+1}, \ldots, h_{rn_0-1})].$$

La composante $(h_{in_0}, \ldots, h_{(i+1)n_0-1})$ est la première ligne de H_i et le reste de ses composantes est obtenue par des décalages cycliques séquentiels de sa première rangée. De cette façon, chacun de ses composantes ont leur propre poids w_i, où $w_i = \sum_{i=0}^{n_0-1} w_i$.

En appliquant le fait que $HG^T = 0$, la relation liant la matrice génératrice G et la matrice de contrôle de parité H, ainsi que l'hypothèse que H_{n_0-1} est inversible, on peut calculer la matrice génératrice sous forme systématique correspondante à cette matrice de contrôle : $G = [I_k | Q]$, où

$$Q = \begin{bmatrix} \left(H_{n_0-1}^{-1}.H_0\right)^T \\ \left(H_{n_0-1}^{-1}.H_1\right)^T \\ \vdots \\ \left(H_{n_0-1}^{-1}.H_{n_0-2}\right)^T \end{bmatrix}$$

et I_k la matrice identité de dimension $k \times k$.

On doit préciser qu'en effet, les auteurs de [37] montrent que l'emploi d'une matrice publique quasi-circulante dont la taille des blocs n'est pas un nombre premier facilite la résolution du décodage générique. Ce qui nous amène donc à considérer des blocs dont la taille est un nombre premier (c'est-à-dire de considérer r premier) afin de se prémunir contre ce type d'attaques.

Notons aussi que, G est une matrice $k \times n$ et que pour tout vecteur $x \in F_2^k$, les k premiers bits du produit xG sont exactement égaux à x lui-même. Nous pouvons maintenant présenter les

algorithmes de la génération de clés, de chiffrement et de déchiffrement pour le cas $QC\text{-}MDPC$ telle que décrit dans [38].

Algorithme 4: Génération des clés de McEliece dans le cas QC-MDPC

Entrées : Le paramètre de sécurité n, le poids w, la co-dimension r et le seuil de correction d'erreur t.

Sorties : La clé publique G et la clé secrète H.

Début

1. Générer une matrice de contrôle de parité $H \in F_2^{r \times n}$ et pouvant corriger t erreurs d'un code $[n, r, w]$-$QC\text{-}MDPC$.
2. Calculer la matrice génératrice correspondante $G = [I|Q]$.

Retourner (G, H).

Notons ici que la valeur t dans ce qui précède dépend de l'algorithme de décodage utilisé.

Tailles des clés : La taille de clé publique pour le cas $QC\text{-}MDPC$ de McEliece est de$n \times k$. Cependant, la totalité de G ne doit pas nécessairement être stockée. Comme G contient toujours la matrice identité I_k, seule la sous-matrice Q doit être stockée. La taille de Q étant $k(n-k)$. Cependant, en utilisant le fait que les blocs $\left(H_{n_0-1}^{-1}.H_i\right)^T$ sont eux-mêmes des matrices circulantes, les besoins en stockage peuvent être encore réduits, car il suffit de stocker les premières lignes seulement. Au total, cela nécessite donc $(n_0 - 1)k$ bits de stockage. Dans le cas où $n_0 = 2$, seuls k bits doivent être stockés. De même, la clé secrète H est $n \times k$ bits, mais seulement la première rangée (composée de n bits), doit être stockés.

Durée d'exécution : En général, le calcul de la matrice de contrôle de parité nécessite la production de n bits de caractère aléatoire, d'analyser ce caractère aléatoire en n_0 et en effectuant $n_0 k$ décalages cycliques ultérieurs. Or n_0 est généralement considéré comme égal à 2, et donc la valeur de k détermine en grande partie ce coût. De plus, il n'est pas nécessaire que la clé secrète entière soit générée parce que, la première ligne fournit suffisamment d'informations pour générer la matrice génératrice correspondante. Le calcul de la matrice génératrice nécessite une inversion de matrice de dimension $r \times r$, puis de $n_0 - 1$ multiplications et des transpositions de matrices de dimension $r \times r$ également. Si nous faisons l'hypothèse très prudente que l'inversion de la matrice et chaque multiplication de la matrice

prennent r^3 opérations, le calcul de G prend environ $n_0 r^3$ opérations. En réalité, une implémentation optimisée de cet algorithme s'exécuterait dans le temps de manière approximativement linéaire en r.

2.2.3.2. Chiffrement

Le chiffrement dans le schéma $QC\text{-}MDPC$ de McEliece peut être décrit succinctement comme une multiplication de matrice suivie d'un Xor avec un vecteur d'erreur. Une description générique de cet algorithme est décrite ci-dessous.

Algorithme 5: Chiffrement de McEliece dans le cas QC-MDPC

Entrées : La clé publique G et le message clair $m \in F_2^k$.

Sorties : Le texte chiffré $c \in F_2^n$

Début

1. Générer un vecteur d'erreur aléatoire $e \in F_2^n$ tel que $wt(e) = t$.
2. Calculer $c \longleftarrow mG + e$.

Retourner c.

Taille du texte chiffré : Un texte chiffré dans ce schéma est de taille n.

Temps d'exécution : La multiplication matrice/vecteur peut être effectuée très rapidement. Par exemple, rappelons que $G = [I_k|Q]$, et donc que les k premiers éléments de mG sont exactement égaux à m, aucun calcul n'est donc nécessaire pour calculer ces bits. L'opération Xor prend des quantités de ressources insignifiantes.

2.2.3.3. Déchiffrement

Le déchiffrement nécessite comme sous-routine un algorithme de décodage $QC\text{-}MDPC$ à correction d'erreurs t avec connaissance de la clé secrète H. Indiquons ce décodeur par Ψ_H.

Algorithme 6: Déchiffrement de McEliece dans le cas QC-MDPC

Entrées : Le texte chiffré $c \in F_2^n$ et la dimension k.

Sorties : Le vecteur $m \in F_2^k$ tel que $d(mG, c) \leq t$.

Début

1. Calculer $mG = \Psi_H(c) = \Psi_H(mG + e)$.
2. Extraire le message clair m à partir des k premières positions de mG.

Retourner m.

Durée d'exécution : L'algorithme de déchiffrement prend un temps et des ressources essentiellement égaux à ceux de l'algorithme de chiffrement.

2.2.4. Décodage

Lors du déchiffrement, un algorithme de décodage est nécessaire pour retrouver le message chiffré. Ainsi, en ce qui concerne les algorithmes de décodage des codes $QC\text{-}MDPC$, nous avons deux types d'algorithmes distincts : il y a d'une part celui introduit dans [39] par Berlekamp, McEliece et Von Tilborg, et d'autre part celui de Gallager [35]. Le premier offre une meilleure capacité de correction des erreurs, mais est plus complexe sur le plan des calculs. En particulier, pour le traitement de codes volumineux, le second, appelée aussi algorithme *Bit-flipping* (algorithme de retournement de bits) semble être approprié. C'est d'ailleurs pour cette raison que nous allons utiliser dans ce qui suit l'algorithme *Bit-flipping* de Gallager [35] pour décoder les codes $QC\text{-}MDPC$. Cet algorithme de décodage est itératif et probabiliste, c'est-à-dire qu'il est constitué d'une série d'opérations qui seront répétés plusieurs fois et qu'il retourne la bonne solution avec une certaine probabilité. Il fournit une capacité de correction d'erreurs pour les codes $LDPC$ qui augmente linéairement avec la longueur du code et diminue plus ou moins linéairement avec le poids *w* de la matrice de contrôle de parité. Notons qu'à l'origine, Gallager a introduit cet algorithme pour le décodage des codes $LDPC$, qui admettent des équations de parité de poids très faibles. Seulement, l'algorithme peut aussi être utilisé pour le décodage des

codes $MDPC$ puisque ceux-ci admettent aussi des équations de parité de poids faibles (légèrement supérieures à ceux des codes $LDPC$). De ce fait, lors du passage des codes $LDPC$ aux codes $MDPC$, on s'attend à une dégradation de la capacité de correction des erreurs. Seulement, comme on l'avait déjà précisé, en cryptographie, on ne s'intéresse pas nécessairement à corriger un grand nombre d'erreurs, mais uniquement un nombre qui assure un niveau de sécurité adéquat.

L'algorithme *Bit-flipping* de Gallager fonction comme suit : étant donnés $x \in \mathbb{F}_2^n$ un mot de code bruité et C un mot de code $[n, r, w]$-$MDPC$ dont une matrice de contrôle de parité creuse est $H \in \mathbb{F}_2^{r \times n}$, on va utiliser le syndrome $s = xH^T \in \mathbb{F}_2^r$ pour identifier les positions du vecteur x qui ne vérifient pas les équations de parités de H. Ainsi, si une coordonnée du syndrome, par exemple i telle que $1 \leq i \leq r$, est non nulle, cela revient à écrire : $\langle x, h_i \rangle = 1$. En d'autres termes, x ne vérifie pas la i-ème équation de parité définie par H. Ceci se produit, si, et seulement si, un nombre impair de coordonnées du vecteur x qui intervient dans cette équation sont erronées. Nous pouvons ainsi présenter l'algorithme comme suit :

Algorithme 7: *Bit-flipping*

Entrées : Une matrice de contrôle de parité H et le mot de code bruité $x \in \mathbb{F}_2^n$.

Sorties : Un mot de code C.

$s \longleftarrow xH^T$ // *Calcul du syndrome*

Tant que $s \neq 0$ **faire**

  Pour $j \in \{1, \ldots, n\}$ **faire**

    $\sigma_i \longleftarrow \langle s, h_i \rangle \in Z$ // *Calcul du* $i-$ *ème compteur*

    Si $\sigma_i > b$ **alors**

      $x_j \longleftarrow x_j \oplus 1$ //*Retourner le bit correspondant*

    Fin si

  Fin pour

  $s \longleftarrow xH^T$ //*on met à jour le syndrome*

Fin tant que

Retourner x

Dans l'algorithme original de Gallager, un seuil de valeur par itération est pré-calculé. Les formules utilisées pour calculer ces seuils garantissent une certaine probabilité de l'échec du décodage. Cette solution est utilisée pour décoder les codes $QC\text{-}MDPC$, mais il n'y a aucune certitude quant à la probabilité d'un échec du décodage. Ceci va constituer donc une faille pour le chiffrement de McEliece et sera exploitée par plusieurs auteurs.

Une autre solution proposée dans [40] est de ne retourner que les bits qui violent un nombre maximum d'équations de contrôle de parité, jusqu'à quelques unités.

Dans [41], plusieurs autres réglages ont été proposés afin d'accélérer la durée moyenne de fonctionnement de l'algorithme. Au lieu d'utiliser le même syndrome s pour toutes les positions lors d'une itération, les auteurs ont proposé de mettre à jour le syndrome après chaque retournement de situation.

2.2.5. Sécurité

La proposition d'utiliser les codes QC-MDPC est sûre selon deux hypothèses :

- ✓ Indiscernabilité des codes QC-MDPC :
  Cela signifie qu'il est difficile de trouver la matrice de contrôle de parité d'un Code QC-MDPC. Il s'agit de trouver un mot de longueur n et de poids w appartenant à la matrice de contrôle de parité. La valeur de w doit donc être suffisamment grande pour que le nombre de possibilités soit trop important pour être épuisé. Toutefois, la valeur de w ne devrait pas être trop importante puisque l'algorithme de décodage itératif offre une capacité de correction des erreurs qui diminue plus ou moins linéairement avec le poids w des équations de contrôles de parité. Précédent les codes proposés pour réduire la taille de la clé souffrent d'une structure algébrique, tandis que les codes LDPC et Les codes MDPC sont libres de cette structure. Cela tend également à renforcer la sécurité concernant le problème de la distinction.
- ✓ Décodage du syndrome des codes quasi-cycliques :
  Cela signifie que le décodage de codes aléatoires quasi-cycliques est probablement un problème difficile, tout comme le décodage de codes linéaires aléatoires. Il est donc peu probable qu'un algorithme en temps polynomial ou plus rapide existe pour résoudre ce problème.

2.2.6. Taille de la clé

Grâce à la suppression de la matrice de brouillage S et de la matrice de permutation P dans ce schéma et à la construction quasi-cyclique, la taille de la clé est considérablement réduite. La clé privée se compose uniquement de la matrice de contrôle de parité H et la clé publique se compose de la matrice de générateur G correspondante et du paramètre de capacité de correction d'erreur t.

Le stockage de la matrice de contrôle de parité H d'un code $[n, r, w]$-QC-$MDPC$ nécessite $n\ bits$. Cette matrice de contrôle de parité peut être entièrement construite en stockant uniquement les $n\ bits$ dans la première ligne de la matrice de contrôle de parité. Les $r - 1$ lignes restantes peuvent être obtenues en effectuant des déplacements quasi cycliques de la rangée stockée.

Le stockage de la matrice génératrice G d'un code $[n, r, w]$-QC-$MDPC$ sous forme systématique nécessite $n - r\ bits$. Cette matrice génératrice peut être entièrement construite en stockant uniquement les $n - r\ bits$ dans la première colonne de la partie non systématique de la matrice génératrice. Les $r - 1$ colonnes restantes de la partie non systématique peuvent être construites en effectuant des décalages quasi-cycliques de la colonne stockée.

Ainsi, divers paramètres ont été proposés pour différents niveaux de sécurité du cryptosystème de McEliece avec des codes QC-MDPC. Ces paramètres sont présentés dans le tableau ci-dessous [2] avec leurs tailles de clé respectives.

n_0	n	r	w	t	Niveau de Sécurité	Sécurité
2	9602	4801	90	84	4801	80-bit
3	10779	3593	153	53	7186	80-bit
4	12316	3079	220	42	9237	80-bit
2	19714	9857	142	134	9857	128-bit
3	22299	7433	243	85	14866	128-bit
4	27212	6803	340	68	20409	128-bit
2	65542	32771	274	264	32771	256-bit
3	67593	22531	465	167	45062	256-bit
4	81932	20483	644	137	61449	256-bit

Tableau 1: Paramètres recommandés pour les codes QC-MDPC

Le tableau ci-dessous [2] compare les tailles des clés publiques entre le système de cryptographie McEliece utilisant les codes QC-MDPC et celui utilisant les codes de Goppa.

Sécurité	QC-MDPC	Goppa
80	4801	460 647
128	9857	1 537 536
256	32771	7 667 855

Tableau 2: Comparaison des clés publiques entre les codes QC-MDPC et les codes de Goppa

Chapitre 3

Attaques sur les codes QC-MDPC

Nous avons vu précédemment que l'utilisation des codes quasi-cycliques MDPC dans le cryptosystème de McEliece permet de construire un schéma de chiffrement post-quantique dont les clés ont une taille raisonnable. Cependant, ces codes étant très bien structurés, ils sont vulnérables aux attaques de types structurelles. Ces attaques permettent aux cryptanalystes de retrouver la clé secrète en vue de déchiffrer le message envoyé. La principale faille de ces types d'attaques est l'algorithme de décodage *bit-flipping*.

Nous allons, dans ce chapitre, faire une étude exhaustive de l'attaque de récupération de clé de Guo-Johansson-Stankovski ainsi que de l'attaque temporelle sur les codes quasi-cycliques MDPC.

3.1. Attaque de Guo-Johannson-Stankovski

L'attaque de récupération de clé dans [42] sur les codes QC-MDPC est une attaque de réaction reposant sur le fait que lors de l'étape du déchiffrement, le décodage itératif peut échouer avec une faible probabilité. Les auteurs ont ainsi identifié une dépendance entre la clé secrète et l'échec du décodage. Cette attaque suppose seulement qu'un adversaire est en mesure de dire quand une erreur s'est produite, par exemple parce qu'une demande de renvoi a été effectuée. Elle peut être utilisée pour construire ce que l'on appelle le spectre de distance pour la clé secrète. L'attaque se déroule en deux étapes. La première étape consiste à calculer le spectre de distance de la clé secrète (ou d'une partie de la clé secrète), basée sur l'observation d'un grand nombre de vecteurs d'erreur qui ont abouti à un échec du décodage. La deuxième étape consiste à reconstruire la clé secrète en fonction du spectre de distance.

3.1.1. Spectre de distance

Définition : Le spectre de distance d'un vecteur $h \in \mathbb{F}_2^r$, noté $d(h)$, est l'ensemble des distances θ telles qu'il existe deux bits de h non nuls à la distance θ. Les distances étant comptées de façon cyclique.

$$d(h) = \left\{ \begin{array}{l} \\ \theta : 1 \leq \theta \leq \left\lceil \frac{r}{2} \right\rceil, \exists (i,j), h_i = h_j = 1, \\ \\ \end{array} \begin{array}{r} 0 \leq i < j < r, \\ \\ \min\{j-i, r-(j-i)\} = \theta, \end{array} \right\}$$

Exemple : considérons le vecteur $h = 1001000001$, avec $r = 10$ et $w = 3$. On peut remarquer que $d(h) = \{1,3,4\}$. En effet, le premier et le dernier bit de h sont voisins (la distance étant comptée de manière cyclique), ce qui donne 1 comme spectre. Le premier et le quatrième bit sont distants de 3. Le quatrième et le dernier bit quant à eux sont distants de 4.

Notons que pour un décalage cyclique dans un sens comme dans l'autre du vecteur h, on retrouve le même spectre de distance.

3.1.2. Description de l'attaque

Le but de cette procédure d'attaque est de récupérer le premier circulant de la matrice secrète H, qui est désignée par H_0, de sa matrice publique G correspondante. Comme la matrice secrète H_0 est quasi-cyclique, il suffit de récupérer uniquement la première ligne de la matrice correspondante, h_0. L'idée principale de cette attaque est de vérifier le nombre d'échecs de décodage signalés en utilisant un certain ensemble de modèles d'erreurs différents. L'attaque génère ces schémas d'erreurs selon un certain critère.

L'idée clé est d'examiner la procédure de décodage pour différents modèles d'erreurs. L'attaque de récupération de la clé sécrète H fonctionne comme ainsi. Soit Ψ_d l'ensemble de tous les vecteurs binaires de longueur $n = 2r$, contenant exactement t nombre de ''1''. Ces t ''1'' sont placés en $\frac{t}{2}$ paires aléatoires avec une distance d entre eux sur la première moitié du vecteur d'erreur v choisi. La seconde moitié du vecteur d'erreur sera entièrement constituée de zéros, puisque l'attaque se concentre uniquement sur la récupération du premier circulant H_0. L'équation suivante donne une description mathématique de la génération de l'ensemble des vecteurs d'erreurs Ψ_d et du vecteur d'erreur v :

$$\Psi_d = \{v = (e, f) \mid w_H(f) = 0, et\ \exists\ s_1, s_2, \dots, s_t\ distinct, s.t.\ e_{s_i} = 1,$$

$$et\ s_{2i} = (s_{2i-1} + d) mod\ r\ pour\ i = 1, \dots, \frac{t}{2}\}.$$

Exemple :

Pour un code C de longueur $n = n_0 \times p = 2 \times 6 = 12$ et $t = 2$, les sous-ensembles Ψ_d suivants avec les modèles d'erreurs sont crées

$$\Psi_1 = \{110000\ 000000, 011000\ 000000, 001100\ 000000, 000110\ 000000,$$
$$000011\ 000000, 100001\ 000000\}$$

$$\Psi_2 = \{101000\ 000000, 010100\ 000000, 001010\ 000000, 000101\ 000000,$$
$$100010\ 000000, 010001\ 000000\}$$

$$\Psi_3 = \{100100\ 000000, 010010\ 000000, 001001\ 000000\}$$

Les sous-ensembles Ψ_d seront construits pour $d = 1, 2, \dots, \frac{r}{2}$. Les modèles d'erreur de ces sous-ensembles seront utilisés pour générer des informations statistiques sur le taux de réussite du décodage. Comme décrit précédemment, ce taux de réussite dépend de l'emplacement des bits d'erreur.

L'attaque effectuera M essais de décodage sur le code $QC - MDPC$, où chaque essai choisit un vecteur d'erreur à partir de Ψ_d. Cela sera fait pour $d = 1, \dots, \frac{r}{2}$, où $U = \frac{r}{2}$ est la limite supérieure. Ainsi, un total de $M \times U$ essais sera effectué, et une probabilité d'erreur de décodage peut être calculée pour chaque essai.

La motivation derrière cette attaque est qu'il existe une corrélation entre la probabilité d'erreur de décodage calculée et la fréquence d'occurrence d'une distance d dans le premier circulant H_0. En d'autres termes, plus cette distance se produit dans la matrice H_0, ou dans le vecteur h_0, plus la probabilité d'erreur de décodage calculée sera faible. Le nombre de fois que la distance se produit dans un vecteur donné est appelé multiplicité de cette distance dans ce vecteur, et est désignée par $\mu(d)$. Par conséquent, un profil de distance de h_0 peut être construit, où il donne toutes les distances disponibles en h_0, et leurs multiplicités correspondantes. Le profil de cette distance est indiqué par $D(h_0)$, et est donné par :

$$D(h_0) = \{d\ répété\ \mu(d) fois, pour\ d = 1, \dots, U\}$$

L'algorithme pour le calcul du spectre tel que présenté dans [42] est illustré ci-dessous :

Algorithme 8: Calcul du spectre de distance

Entrées : Les paramètres $n, r, w\ et\ t$ du code $QC - MDPC$ sous-jacent, le nombre d'essais M de décodage par distance.

Sorties : Le spectre de distance $D(h_0)$.

Pour toutes les distances d **faire**

Essayez de décoder les M essais en utilisant le modèle d'erreur conçu

Effectuer un test statistique pour décider de la multiplicité $\mu(d)$

Si $\mu(d) > 0$ **alors**

Ajouter d avec la multiplicité $\mu(d)$ au spectre de distance $D(h_0)$.

Exemple :

$d = 4$

$$\left.\begin{matrix} 1001 \\ 1001 \\ \vdots \\ 1001 \end{matrix}\right\} \frac{t}{2}, placés\ aléatoirements\ dans\ v \Longrightarrow v = [00100100001001 \cdots 001001 \cdots]$$

$$h_0 = [\cdots 000110100100 1100 \cdots]$$

$$\Longrightarrow \mu(d)_{h_0} = 3$$

Cet exemple montre la façon dont le vecteur d'erreur est généré lors de l'attaque et pour une distance donnée, ainsi que la façon dont la multiplicité de cette distance apparaîtra en h_0.

3.1.3. Reconstruction de la Clé secrète

Une fois que l'attaque est faite, et que $D(h_0)$ est construit avec succès, la reconstruction de h_0, et donc de H_0, peut être faite facilement. Tout d'abord, on place un ‘‘1’’ dans la première position du vecteur de reconstruction, qu'on notera h_{recons}. Un deuxième ‘‘1’’ est placé à une

position i_0, où i_0 est égal à la première distance en $D(h_0)$. Le troisième ‘‘1’’ est placé à une position i_1 quelconque et la distance entre celui-ci et les deux précédemment calculées. Si ces distances calculées apparaissent dans le profil de distance, alors la troisième est maintenue dans sa position; sinon, une nouvelle position est testée. Cette opération est répétée jusqu'à ce que les distances calculées, avec cette position i_1, apparaissent dans $D(h_0)$. La même procédure est répétée pour les quatrième, cinquième, sixième et ainsi de suite, où toutes les distances calculées entre la $n^{ième}$ et les $n-1$ ‘‘1’’ précédents doivent apparaître dans $D(h_0)$.

À la fin, h_0 sera entièrement reconstruit, et H_0 pourra alors être reconstruit en décalant cycliquement chaque entrée de h_0 sur toutes les lignes. Ensuite, le reste de la clé secrète H peut être reconstruit sous la forme H_0 en utilisant l'algèbre linéaire [26] [25].

Nous pouvons illustrer la procédure de reconstruction de la clé secrète grâce à l’algorithme suivant présenté dans [42].

Algorithme 9: Récupération de clé à partir du spectre de distance

---

Entrées : Le spectre de distance $D(h_0)$, la clé secrète partielle h_0 et la profondeur actuelle l
Sorties : La clé secrète h_0 récupérée ou le message "Il n’existe aucune clé secrète de ce type".
Paramètres récursives initiaux : Le spectre de distance $D(h_0)$, un ensemble vide pour la clé secrète, la profondeur actuelle 0.

Si $l = w$ **alors**
    Retourner h_0. //*clé secrète trouvée*
Pour toutes les clés potentielles de bits i **faire**
    Pour toutes les distances au bit clé i existent en $D(h_0)$ **faire**
        Ajouter le bit de clé i à la clé secrète h_0
        Effectuer un appel récursif avec les paramètres $D(h_0)$, h_0 et $l+1$
        Si l'appel récursif trouve la solution h_0 **alors**
            si h_0 est la clé secrète **alors**
                Retourner h_0. //*clé secrète trouvée*
        Supprimer le bit de clé i de la clé secrète h_0.

Retourner "Il n’existe aucune clé secrète de ce type".

Exemple :

Soit H_0 une matrice de contrôle de parité dont la première ligne notée h_0 comporte des 1 aux positions 2, 5, 7 et 10.

$$h_0 = [\,0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0\,]$$

On a alors : $b_2 = 1,\ b_5 = 1,\ b_7 = 1\ et\ b_{10} = 1$

Le spectre de distance $D(H_0)$ est constitué de toutes les distances d pour les quelles $1 \leq d \leq \frac{r}{2}$

$$D(H_0) = \{d(b_2, b_5), d(b_2, b_7), d(b_2, b_{10}), d(b_5, b_7), d(b_5, b_{10}), d(b_7, b_{10})\}$$

$$= \{3, 5, 4, 2, 5, 3\} = \{2, 3, 3, 4, 5, 5\}$$

Connaissant le spectre de distance $D(H_0)$, on peut reconstruire h_0.

$$h_0 = [\,1 \;\; 0 \;\; 1 \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ?\,]$$

$$d(b_1, b_3) = 2$$

Toutes les distances $d \in D(H_0) \rightarrow D(H_0) = \{\cancel{2}, 3, 3, 4, 5, 5\}$

$$h_0 = [\,1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; ? \;\; ? \;\; ? \;\; ? \;\; ? \;\; ?\,]$$

$$d(b_1, b_6) = 5 \quad \text{et} \quad d(b_3, b_6) = 3$$

Toutes les distances $d \in D(H_0) \rightarrow D(H_0) = \{\cancel{2}, \cancel{3}, 3, 4, \cancel{5}, 5\}$

$$h_0 = [\,1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; ? \;\; ? \;\; ?\,]$$

$$d(b_1, b_9) = 4, \quad d(b_3, b_9) = 6 \quad \text{et} \quad d(b_6, b_9) = 3$$

Pas toutes les distances $d \in D(H_0) \rightarrow echec\ du\ test$

$$h_0 = [\,1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; ? \;\; ?\,]$$

$$d(b_1, b_{10}) = 3, \quad d(b_3, b_{10}) = 5 \quad \text{et} \quad d(b_6, b_{10}) = 4$$

Toutes les distances $d \in D(H_0) \rightarrow D(H_0) = \{\cancel{2}, \cancel{3}, \cancel{3}, 4, \cancel{5}, \cancel{5}\}$

On obtient ainsi : $h_0 = [\,1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0\,]$.

En faisant un décalage cyclique, on retrouve l'expression originale :

$$h_0 = [\,0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0\,]$$

Cette procédure peut être effectuée pour chaque matrice circulante $H_i.$ dans H. Après avoir obtenu tous les $h_i.$ on peut les combiner et construire la matrice secrète de contrôle de parité H.

Le nombre d'opérations binaires nécessaires pour rompre le système cryptographique QC-MDPC avec pour la sécurité 80-bit est estimée à 228 opérations. La décision difficile de Gallager L'algorithme de décodage était l'un des premiers algorithmes de décodage itératif pour les codes LDPC. Des algorithmes de décodage plus avancés avec de meilleures performances de décodage rendent cette la cryptanalyse pour trouver la clé privée. Il faut envoyer davantage de cryptogrammes pour obtenir suffisamment des informations statistiques pour construire le spectre de distance $D(H_i)$. Ces algorithmes de décodage d'augmenter le facteur de travail de cette cryptanalyse d'un facteur d'environ $2^{15.6}$ augmentant ainsi le facteur de travail total à $2^{43.6}$. Cependant, ce facteur de travail est encore trop faible pour que ce cryptosystème soit sécurisé.

La variante CCA-2, plus sûre, qui utilise essentiellement l'idée de brouiller les bits du message d'entrée m autour, peut également être cryptée avec cette attaque. Bien que l'attaque nécessite quelques modifications mineures et que le facteur de travail de cette variante soit plus élevé (opérations sur $2^{55.3}\ bits$), le niveau de sécurité est encore trop faible pour être d'une utilité pratique.

3.2. Attaque temporelle

L'attaque de GJS, dans [42], profite d'un minutieux calcul du taux d'échec au décodage pour les différentes classifications du vecteurs d'erreur. Pour la plupart des décodages une erreur se produit lorsqu'il faut plus qu'un certain nombre d'itérations. De ce fait, une attaque similaire peut être réalisée si un adversaire est capable de connaître le nombre d'itérations que l'algorithme effectue pour le décodage. Cela signifie que l'attaque du GJS peut être reformulée comme une attaque par canal auxiliaire très forte lorsque l'algorithme de décodage n'est pas en temps constant.

Et comme le poids du syndrome fait fuir des informations, tout paramètre corrélé à cette quantité pourrait être utilisé pour cette attaque. Un paramètre intéressant et souvent facile à mesurer est le temps.

L'algorithme de décodage des codes QC-MDPC est un algorithme itératif. Le nombre de tours nécessaires pour corriger les erreurs est variable. Cela a été étudié dans [5]. Pour les paramètres habituels pour 80 bits de sécurité, l'algorithme corrige généralement l'erreur en 3 tours, mais

dans certains cas, il en faut 4, 5 itérations, voire plus. Les mises en œuvre habituelles s'arrêtent après un certain nombre de tours (environ 10), c'est ce qui a été utilisé pour l'attaque de [42].

C'est ce qui a motivé Lequesne dans [3] à essayer de réaliser une attaque de synchronisation (attaque temporelle). Le scénario est le même que celui de l'attaque de GJS [42], mais au lieu d'observer le poids du syndrome, l'attaquant peut mesurer le nombre d'itérations nécessaires pour décoder son message. Pour obtenir le spectre, il utilise exactement le même algorithme de collecte de données : pour chaque distance dans le spectre, il calcule le nombre moyen d'itérations nécessaires pour corriger une erreur contenant cette distance. En moyenne, le nombre d'itérations est légèrement plus faible lorsque la distance apparaît dans le spectre de la clé. Cela fonctionne bien et il est possible de récupérer entièrement le spectre de distance avec 100 millions d'échantillons sur Schéma QC-MDPC de sécurité 80 bits [3].

Il s'agit de la première attaque de synchronisation sur le schéma QC-MDPC. Elle indique que nous avons besoin d'un décodage en temps constant, comme pour les QcBits [4], mais avec un algorithme de décodage.

Cette attaque a d'abord été testé sur une version simplifiée du système n'utilisant qu'un seul bloc, afin de le comparer au comportement attendu. Mais le résultat est frappant [3]. En utilisant les paramètres habituels pour une sécurité de 80 bits, avec cent mille échantillons, l'attaquant a pu récupérer tout le spectre et même les multiplicités, c'est-à-dire les distances qui apparaissent plusieurs fois dans la clé. Lorsque l'on pousse à un milliard d'échantillons, il n'est pas un lieu de confusion.

Lorsque l'expérience est effectuée sur le véritable schéma QC-MDPC avec deux blocs, on obtient des résultats similaires. L'attaque est effectuée sur chaque bloc séparément, c'est-à-dire que pour chaque modèle d'erreur, on ajoute le poids du syndrome sur les compteurs de toutes les distances présentes dans la première moitié de l'erreur pour récupérer le spectre du premier bloc. Comme il n'y a pas de corrélation entre les deux blocs la présence du deuxième bloc agit comme un bruit ajouté au poids du syndrome. La seule différence est donc qu'on aura besoin de plus d'échantillons pour réduire la variance et bien distinguer les distances dans le spectre. A noter qu'il est possible de calculer le spectre des deux blocs en même temps, de sorte qu'il n'est pas nécessaire de doubler le nombre d'échantillons pour récupérer le deuxième bloc.

Cette attaque a également été réalisée lorsqu'une autre erreur est ajoutée au syndrome, comme dans le schéma d'Ouroboros [6]. Encore une fois, cela n'ajoute qu'un bruit aléatoire et il est

possible de récupérer le spectre avec quelques millions d'échantillons pour les paramètres de sécurité 80 bits.

L’attaque temporelle est beaucoup plus rapide que l'attaque de réaction de GJS. Cette rapidité s’explique par les différences dans le nombre d'itérations qui sont beaucoup plus courantes que les erreurs de décodage. Cela permet d'obtenir plus d'informations sur les corrélations à collecter par itération.

Chapitre 4

Analyse des algorithmes de décodage des codes QC-MDPC

Les codes QC-MDPC permettent la conception du cryptosystème de McEliece avec des clés et une sécurité qui réduit à l'évidence les problèmes de décodage pour les codes quasi-cycliques. En particulier, ces codes sont parmi les plus prometteurs qui sont proposés à l'appel du NIST pour la normalisation de la cryptographie post-quantique.

La première génération d'algorithme de décodage souffre d'un petit, mais pas négligeable, taux d'échec du décodage (DFR, pour *Decoding Failure Rate,* de l'ordre de 10^{-7} à 10^{-10}). Ce qui a permis l'attaque de récupération de clés de Guo, Johansson et Stankovski qui exploite une petite corrélation entre les messages erronés et la clé secrète du cryptosystème de McEliece [42]. Cependant, cette attaque semble ne pas avoir d'incidence sur l'établissement interactif de communications sécurisées (le protocole TLS par exemple), mais l'utilisation de clés publiques statiques pour des applications asynchrones (les courriers électroniques par exemple) est rendue dangereuse.

Il est donc intéressant de comprendre et d'améliorer le décodage des codes QC-MDPC pour les applications cryptographiques. En particulier, la recherche de paramètres pour lesquels le DFR est manifestement négligeable (généralement aussi bas que 2^{-64} à 2^{-128}) augmenterait l'applicabilité du cryptosystème [43].

4.1. Optimisation de l'algorithme *Bit-flipping*

L'algorithme *Bit-flipping* a été décrit dans le chapitre 2. Cependant, il est nécessaire de rappeler de manière brève le fonctionnement de l'algorithme. Ainsi, pour un mot de code $x \in \mathbb{F}_2^n$, on a les étapes suivantes :

- Calculer le syndrome $s = Hx^T$ du mot reçu x.
- Comptez les équations de contrôle de parité insatisfaites, notées $\#_{upc}$ associées à chaque bit de x.
- Inversez les bits de x qui violent plus de b équations, où b est le seuil d'inversion de bit.
- Recalculez le syndrome pour le nouveau x mis à jour.

Ce processus est répété jusqu'à ce que le syndrome devienne nul ou qu'un nombre maximum prédéfini d'itérations soit atteint, à partir duquel une erreur de décodage est renvoyée.

Le nombre d'équations de contrôle de parité insatisfaites est égal au nombre de bits partagés dans une ligne de la matrice de contrôle de parité H et du syndrome s. A noté que le syndrome ne dépend, par définition, que de l'erreur e qui est ajoutée à un mot de code c :

$$S = Hx^T = H(c + e)^T = Hc^T + He^T = He^T$$

Puisque $Hc^T = 0$, par définition.

Ainsi, plusieurs techniques ont été élaborés pour accélérer le calcul du syndrome et réduire le taux d'échec du décodage.

Les décodeurs *Bit-flipping* dans la littérature recalculent le syndrome après chaque itération pour décider si le décodage a réussi ou non. Le coût d'un seul calcul du syndrome peut être estimé à environ deux fois le coût d'un encodage dans le cas des codes QC-MDPC avec $n_0 = 2$.

Les auteurs de [44] proposent une optimisation qui peut être appliquée à tous les décodeurs bit-flipping, basée sur l'observation suivante : si le nombre d'équations de contrôle de parité non satisfaites dépasse un seuil qu'on notera b, le bit correspondant dans le texte chiffré est inversé et le syndrome change. Soulignons que le syndrome ne change pas arbitrairement, mais le nouveau syndrome est égal à l'ancien syndrome accumulé avec la ligne h_j de la matrice de contrôle de parité correspondant au bit inversé à la position j.

En gardant la trace des bits qui sont retournés et en mettant à jour le syndrome en conséquence, le recalcule du syndrome peut être omis. Comme seuls quelques bits sont inversés à chaque itération lors du décodage, la mise à jour du syndrome nécessite beaucoup moins d'ajouts qu'un calcul de syndrome ordinaire.

Il y a deux façons d'appliquer le calcul du syndrome pour les optimisations du paragraphe précédent. L'une d'elles consiste à stocker toutes les modifications du syndrome dans un registre séparé et d'ajouter les modifications à la fin d'une itération de décodage au syndrome. Ainsi, le calcul du syndrome est accéléré mais le comportement du décodage reste inchangé. L'autre possibilité est d'appliquer directement les changements au syndrome à chaque fois qu'un bit du texte chiffré est retourné. Cela accélère également le calcul du syndrome mais affecte également le comportement du décodage puisque le syndrome modifié est utilisé pour déterminer les équations de contrôle de parité non satisfaites de bits de texte chiffré suivants.

4.2. Les variantes de l'algorithme *Bit-flipping*

Les algorithmes itératifs bit-flipping ont un taux d'échec de décodage beaucoup plus élevé lorsqu'ils sont utilisés pour décoder les codes MDPC que ceux des équivalents LDPC. Ceci est dû au fait que la matrice de contrôle de parité d'un code LDPC est sensiblement plus clairsemée qu'une matrice correspondante pour le code MDPC. Autrement dit chaque ligne de la matrice pour le code MDPC contient un plus grand nombre d'entrées non nulles et donc le nombre de bits inclus dans le calcul de chaque bit du syndrome est considérablement augmenté. Raison pour laquelle il est plus difficile de décider si un bit est erroné ou non. Dans la construction originale de Gallager, une seule décision était prise par bit en une itération, et en inversant un bit qui n'est pas en erreur, la probabilité d'une défaillance de décodage est plus élevée.

4.2.1. La variante *Black-Gray*

La variante *Black-Gray* [45] de l'algorithme *Bit-flipping* utilise plusieurs étapes dans une itération de décodage unique afin de réduire la probabilité de retourner les bits inutiles et réduire le taux d'échec. Il utilise deux seuils prédéfinis δ et d pour trier les bits avec un nombre élevé d'équations de contrôles de parité insatisfaisants en deux ensembles : l'ensemble noir et l'ensemble gris.

L'algorithme 10 illustre le décodeur *Black-Gray*. Cette variante fonctionne selon un nombre fixe d'itérations (X_{BG}). L'algorithme comporte trois principales étapes. L'étape I consiste à effectuer une itération de l'algorithme *Bit-flipping* et marquer les bits inversés avec des étiquettes B pour *Black* (noire) et G pour *Gray* (grise). L'étape II consiste à déverrouiller les bits d'erreur noirs qui atteignent un certain seuil. La dernière étape, l'étape III, consiste aussi à déverrouiller les bits d'erreur gris qui atteignent un certain seuil.

Algorithme 10*: Black-gray*

Entrées : Matrice de contrôle de parité H, mot de code c et le maximum d'itérations X_{BG}.

Sorties : L'erreur e.

Exception : ''échec du décodage'' **retourner** $\perp$.

Procedure Black-Gray (c, H)

$s = Hc^T,\ \ e = 0,\ \ \delta = 4$

$B = \phi,\ \ G = \phi$

Pour $itr = (O, \dots, X_{BG} - 1)$ **Faire**

$th = CalculeSeuil(s)$

$upc[n-1:0] = CalculeUPC(s, H)$ // étape I

Pour $(i = 0, \dots, n-1)$ **Faire**

Si $upc[i] \geq th$ **Alors**

$e[i] = e[i] \oplus 1$

$B = B \cup i$ // mise à jour des ensembles noirs

Sinon Si $upc_i >= th - \delta$ **Alors**

$G = G \cup i$ // mise à jour des ensembles gris

$s = H(c^T + e^T)$ //mise à jour du syndrome

$upc[n-1:0] = CalculeUPC(s, H)$ // étape II

Pour $b \in B$ **Faire**

Si $upc[b] > (\frac{d+1}{2})$ **Alors**

$e[b] = e[b] \oplus 1$

$s = H(c^T + e^T)$ //mise à jour du syndrome

$upc[n-1:0] = CalculeUPC(s, H)$ // étape III

Pour $g \in G$ **Faire**

Si $upc[g] > (\frac{d+1}{2})$ **Alors**

$e[g] = e[g] \oplus 1$

$s = H(c^T + e^T)$ //mise à jour du syndrome

Si $(wt(s) \neq 0)$ **Alors**

Retourner $\perp$

Sinon

Retourner e

L'algorithme fonctionne comme suit : les bits avec un nombre maximum d'équations de contrôles de parité non satisfaites sont classés comme des bits *Black* (noirs) et retournés immédiatement. Les bits dont le nombre d'équations de contrôles parité non satisfaits s'écartant du maximum d'un nombre inférieur au seuil δ, sont classées comme des bits *Gray* (gris), mais ne sont pas retournées. Après la boucle initiale, le syndrome et le nombre d'équations de contrôles de parité non satisfaits sont recalculés pour chaque bit. Ensuite, chaque bit dans les deux ensembles de *Black* et de *Gray* sont analysés à nouveau : si le nombre d'équations de contrôles de parité non satisfaites dépasse un seuil défini par le deuxième paramètre d, les bits noirs qui dépassent ce seuil sont ramenés à leur état initial et les bits gris sont retournés. Après chaque étape, le nombre d'équations de contrôles de parité non satisfaisants est mis à jour en conséquence [46].

4.2.2. La variante *Back-flipping*

L'algorithme *Back-flipping* est une variante de l'algorithme originale de Gallager. Cet algorithme fut introduit dans [47] dans le but de réduire le DFR du décodage *Bit-flipping*. Dans ce qui suit, il est décrit comme dans [48].

Comme les positions avec des compteurs plus élevés dans l'algorithme original de Gallager ont des probabilités d'être erronées. Ces positions sont retournées lorsque le compteur est au-dessus d'un seuil, la quantité au-dessus n'a pas d'importance et une partie des informations fiables est perdu. L'idée de *Back-flipping* est d'utiliser ces informations tout en conservant la simplicité du décodeur de *Bit-flipping*.

Parmi les décisions de retournement de bits, la plupart sont bonnes (suppression d'erreurs) et certaines sont mauvais (ajout d'erreurs). Les mauvaises décisions conduisent ainsi à un échec au décodage. Pour exploiter les informations de fiabilité, un décodeur pourrait réduire l'impact des décisions les moins fiables et renforcer l'impact des décisions les plus fiables. *Back-flipping* est un nouvel algorithme de retournement de bits qui utilise le temps τ pour exploiter les informations de fiabilité fournies par les compteurs à chaque retournement de bits. Chaque flip (retournement) a une durée de vie (fini). Lorsque son temps est écoulé, le flip est annulé. Les positions avec un compteur plus élevé restent le plus longtemps retournées que les positions avec un compteur juste au-dessus du seuil. La conception de *Back-flipping* est basée sur les principes suivants :

- Les décisions les plus fiables auront plus d'influence dans le processus de décodage,
- Toutes les mauvaises décisions seront annulées à un moment donné,
- La sélection conservatrice des seuils empêche les mauvaises décisions d'entrer en cascade.

En outre, on constate aisément que, par rapport à l'algorithme *Bit-flipping*, l'algorithme *Back-flipping* ne nécessite que quelques opérations supplémentaires pour gérer un tableau des délais D. De plus, en ce qui concerne le seuil, le ttl ou time-to-live (durée de vie) est très bien approximé par une fonction affine pour tout ensemble de paramètres fixé et son calcul a un coût négligeable en pratique [48].

Algorithme 11*: Back-flipping (*ou *Back-flip)*

Entrées : La matrice de contrôle de parité H, le syndrome s et un entier $u \geq 0$.

Sorties : L'erreur e.

Prérequis : $|s - eH^T| \leq u \quad ou \quad \tau >$ max $_\tau$.

$e \longleftarrow 0\ ;\ \tau \leftarrow 1\ ;\ D \leftarrow 0$

Tant que $|s - eH^T| > u \quad et \quad \tau \leq$ max $_\tau$ **faire**

Pour j telque $D_j = \tau$ **faire**

$e_j \leftarrow 0$

$\tau \longleftarrow \tau + 1$

$s' \leftarrow s - eH^T$

$T \leftarrow seuil(|s'|,\ t - |e|)$

Pour $j \in \{0, \dots, n-1\}$ **faire**

Si $|s' \cap h_j| \geq T$ **alors**

$e_j \leftarrow 1 - e_j\ ;\quad D_j \leftarrow \tau + ttl(|s' \cap h_j| - T)$

Retourner e

Règle de sélection des seuils $(\boldsymbol{S, t_0})$. Comme la durée de vie d'un *flip* est toujours fini, un mauvais *flip* sera toujours annulé par la suite. Cependant, il est nécessaire pour éviter d'ajouter d'autres mauvais *flips* pendant la période où il est retourné. Pour y parvenir, les seuils sont utilisés avec $s = |s'|$ et $t' = t - |e|$. Il s'agit de la meilleure estimation du poids d'erreur, qui suppose que chaque flip a supprimé une erreur. Lorsque de nombreuses erreurs ont été ajoutées,

le seuil correspondant est plus élevé que pour l'algorithme original de Gallager, cela ralentira le décodage en laissant le temps d'annuler les mauvaises décisions tout en ne prenant que de nouveaux *flips*. Dans le cas typique, ou la plupart des décisions de *flip* étaient bonnes, le seuil est proche à l'optimal, et le décodage converge rapidement.

Time-to-live $ttl(\delta)$. Empiriquement, il apparaît que la durée de vie devrait augmenter avec la différence δ entre le compteur de position et l'itération seuil. Il devrait également être limité, car sinon des valeurs de comptage aberrantes pourraient conduire à ajouter des erreurs plus difficiles à détecter : des positions correctes avec un compteur élevé peuvent devenir des erreurs avec un compteur bas une fois retournés, leur compteur doit changer radicalement avant d'être corrigées par un algorithme reposant uniquement sur un seuil. La fonction ttl dépend des paramètres du code (en particulier w et t) ainsi que du nombre maximum d'itérations du décodeur.

4.3. Notre proposition

Les variantes *Black-Gray* et *Back-flipping* peuvent décoder efficacement les codes MDPC pour un nombre d'itérations faible (11 itérations au maximum). Or, plus le nombre d'itérations est faible, plus la probabilité de réussir une attaque de réaction (attaque de GJS) est négligeable. Aussi, il a été démontré dans [45] que la réduction du nombre d'itérations permettrait une implémentation en temps constant. Notons qu'une implémentation en temps constant permet d'éviter les attaques temporelles sur le cryptosystème. En effet, ces types d'attaques exploitent la différence du temps d'exécution qui existe entre deux itérations successives pour récupérer la clé secrète. Cependant, si le *Back-flipping* permet d'obtenir un DFR plus bas que le *Black-Gray*, ce dernier est plus performant lorsqu'il s'agit d'une implémentation en temps constant [45].

Ainsi, notre proposition consiste à améliorer le décodage de l'algorithme *Black-Gray* (puisqu'étant plus performant lors d'une implémentation en temps constant) pour obtenir un DFR plus bas ou avoisinant celui de l'algorithme *Back-flipping*.

Le *Black-Gray* comprend trois étapes principales pour chaque itération. Donc plus le nombre d'itérations est grand plus on a d'étapes dans la phase de décodage. L'idée ici est de réduire le nombre d'étapes de l'algorithme et voir comment cela agit sur le DFR. Nous avons ainsi réalisé une combinaison entre le *Black-Gray* et le *Bit-flipping*. C'est-à-dire lors du décodage, nous avons effectué la première itération avec le *Black-Gray* et les itérations restantes avec le *Bit-*

flipping. Nous appellerons cette variante le *Black-Gray-flipping*. La combinaison citée précédemment nous a permis de réduire significativement le nombre d'étapes du décodage, puisque le *Bit-flipping* ne s'effectue qu'en une seule étape pour chaque itération. Par exemple pour cinq itérations, on aura $3 + 1 + 1 + 1 + 1 = 7$ étapes, alors qu'en utilisant le *Black-Gray* seul, on aura $3 + 3 + 3 + 3 + 3 = 15$ étapes. La réduction du nombre d'étapes effectuées par l'algorithme de décodage augmente donc la vitesse d'exécution mais également les performances du décodeur. Reste à savoir comment cela va impacter sur le DFR ?

Pour cela nous avons réalisé des simulations pour comparer le DFR entre le *Black-Gray*, le *Back-flipping* et le *Black-Gray-flipping* pour 11, 7 et 5 itérations.

4.4. Simulations et résultats

4.4.1. Choix des paramètres de simulations

Notre objectif ici est de simuler les trois algorithmes cités précédemment afin de comparer leur DFR. Nous avons ainsi réalisé 2000 essais sur 100 codes QC-MDPC différents et cela pour 11, 7 et 5 itérations. Les paramètres utilisés pour ces codes QC-MDPC sont les suivants :

$$k = r = 4801, \quad n_0 = 2, \quad n = n_0 \times r = 2 \times 4801 = 9602, \quad w = 90 \quad et \quad t = 84$$

Dans ce cas, un bloc de texte en claire de 4801 bits est encodé en un mot de code de 9602 bits, auquel il faut ajouter un certain nombre d'erreurs. La matrice de contrôle de parité se compose ainsi de deux blocs 4801×4801 circulants respectivement pour H_0 et H_1.

Pour la variante *Black-Gray*, les deux paramètres à prendre en compte sont d et δ. Dans [45], les auteurs proposent d'utiliser $\delta = 4$ pour une meilleure optimisation de l'algorithme. Pour le paramètre d, nous avons pris plusieurs valeurs entre 10 et 100 et les simulations montrent que la valeur $d = 60$ donne le DFR le plus bas.

Pour la variante *Back-flipping*, nous considérons les mêmes paramètres que précédemment pour les codes QC-MDPC. Il faudra également fixer un temps maximum de fonctionnement de l'algorithme. Lorsque ce temps est atteint, l'algorithme s'arrête et renvoi un certain nombre d'erreur. Nous avons appelé ce temps maximum de fonctionnement $max\ \_\tau$ et nous l'avons fixé à $max\ \_\tau = 5$ [48].

4.4.2. Simulations avec 11 itérations

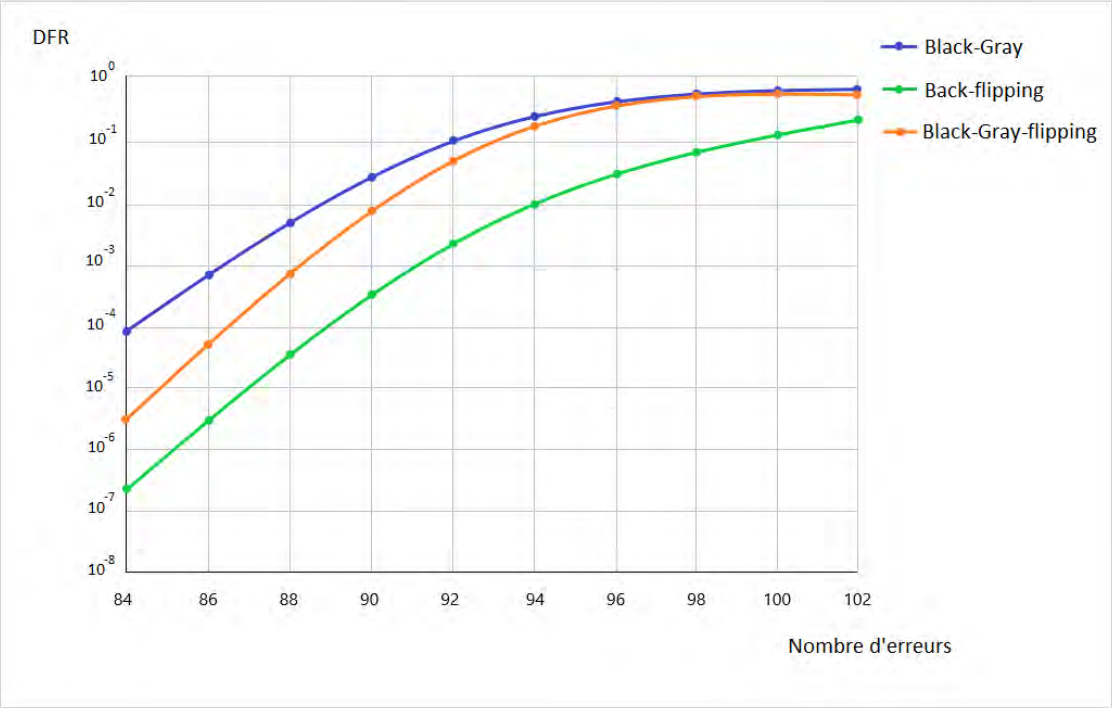

Figure 6 *: Comparaison du DFR des algorithmes Black-Gray, Black-Gray-flipping et Back-flipping pour 11 itérations*

Les simulations pour les algorithmes *Black-Gray*, *Back-flipping* et *Black-Gray-flipping* pour 11 itérations donnent les résultats consignés sur la figure 6 ci-dessus. Ces résultats montrent que l'algorithme *Back-flipping* offre le DFR le plus bas. Cependant, le *Black-Gray-flipping* dépasse de loin le *Black-Gray* original en terme de DFR.

Ceci peut être expliqué par la réduction du nombre d'étapes du décodage pour l'algorithme *Black-Gray-flipping*, mais également par le nombre réduit de bits erronés qui sont retournés. En effet, pour ce dernier, comme le *Black-Gray* divise les bits en erreurs en deux groupes *Black* et *Gray* lors de la première itération, seuls les bits en erreurs se rapprochant du seuil δ sont retournés par le *Bit-flipping* lors des itérations suivantes.

L'algorithme *Back-flipping* utilise le facteur temps pour retourner les bits d'erreurs. Ainsi, comme chaque *flip* a une durée de vie fini, donc un mauvais *flip* est par la suite annulé. Ceci permet d'obtenir un DFR assez bas comparé aux autres algorithmes.

4.4.3. Simulations avec 7 itérations

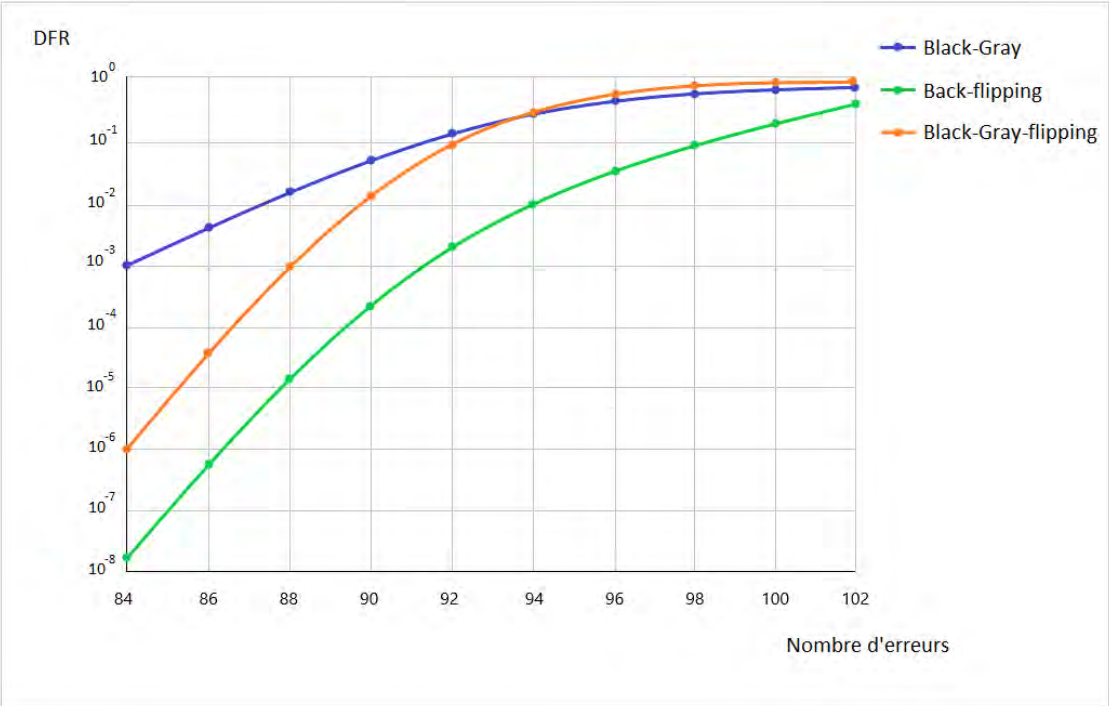

Figure 7 *: Comparaison du DFR des algorithmes Black-Gray, Black-Gray-flipping et Back-flipping pour 7 itérations*

Lorsque l'on réduit le nombre d'itérations à 7, le *Black-Gray* voit son DFR augmenter mais reste toujours performant. En effet, le nombre de bits d'erreurs retournés par l'algorithme est réduit significativement lorsque les itérations sont à leur tour réduites. Ce qui fait que beaucoup d'erreurs ne sont pas corrigées.

Le DFR pour le *Black-Gray-flipping* et le *Back-flipping* diminue avec le nombre d'itérations. Cependant, cette diminution est plus importante pour le *Back-flipping*. En effet, ce dernier est conçu pour fonctionner efficacement avec un minimum d'itérations.

4.4.4. Simulations avec 5 itérations

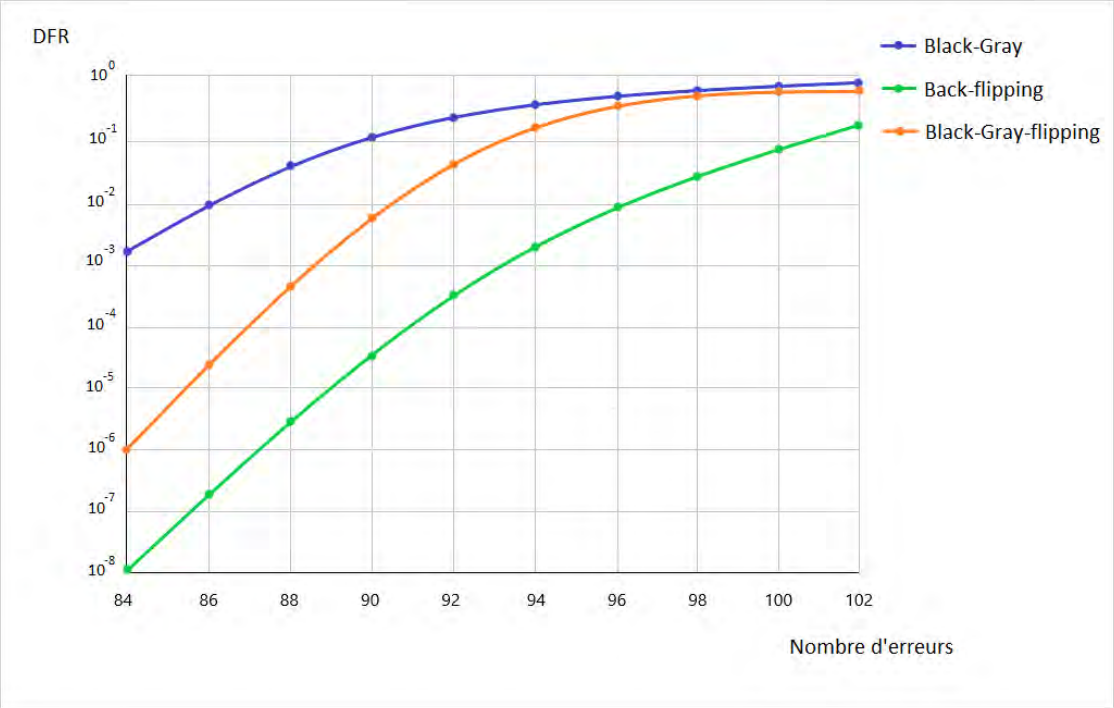

Figure 8 *: Comparaison du DFR des algorithmes Black-Gray, Black-Gray-flipping et Back-flipping pour 5 itérations*

La réduction du nombre d'itérations à 5 fournit des résultats presque similaires que lorsqu'on était à 7 itérations pour les algorithmes *Black-Gray* et le *Black-Gray-flipping*. Les 4 dernières itérations du *Black-Gray-flipping* réalisées à partir du *Bit-flipping* sont à l'origine de cette stabilité du DFR, car le *Bit-flipping* n'étant pas efficace lorsque les itérations sont très réduites.

Le *Back-flipping* est toujours plus performant que les autres algorithmes en terme de DFR. Nous noterons ici une très légère diminution du DFR qui va atteindre 10^{-8}.

L'utilisation de l'algorithme *Bit-flipping* pour le décodage des codes QC-MDPC n'est possible qu'avec un nombre élevé d'itérations pour ainsi parvenir à retrouver efficacement le mot de code. Cependant, plus le nombre d'itérations est grand, plus la probabilité qu'un échec au décodage se produit. Ce qui entraine un DFR assez grand, exploitable pour mener une attaque de réaction (comme celle de GJS) ou une attaque temporelle. Les algorithmes *Black-Gray* et *Back-flipping* sont ainsi conçu pour fonctionner avec un nombre réduit d'itérations. Ils

permettent de décoder efficacement les codes QC-MDPC avec un minimum d'itérations, même si le DFR qu'ils fournissent reste toujours non négligeable.

Cependant, s'agissant d'une implémentation en temps constant, le *Black-Gray* reste plus performant que le *Back-flipping*, même si ce dernier fournit un DFR plus bas. Ce qui nous a poussé à modifier l'algorithme *Black-Gray*. La combinaison entre le *Black-Gray* et le *Bit-flipping* nous a donné l'algorithme *Black-Gray-flipping*.

Nous avons ainsi analysé le DFR pour les différents algorithmes. On note que même si le *Black-Gray-flipping* n'égale pas le DFR du *Back-flipping*, il surpasse de loin le *Black-Gray* original. Maintenant, il reste à étudier les performances de cet algorithme (le *Black-Gray-flipping*) lorsqu'il est implémenté en temps constant.

Conclusion Générale

Le cryptosystème de McEliece est un très bon candidat pour la cryptographie post-quantique. Les codes de Goppa instanciés à ce cryptosystème, permettent d'obtenir un schéma assez sécurisé face à un ordinateur quantique. Cependant, à cause de la taille de clés énormes de ces codes, il est impossible de les utilisés en pratique. D'où la nécessité d'utiliser d'autres types de codes.

Ainsi, dans ce manuscrit, nous avons étudié les codes QC-MDPC avec le schéma de McEliece. Ces codes nous ont permis d'obtenir une taille de clés raisonnable qui peut être facilement implémenté sur du matériel. Néanmoins, l'algorithme de décodage utilisé pour décoder les codes MDPC (le *Bit-flipping*) présente un DFR (taux d'échec au décodage) qui a été exploité pour mener une attaque de récupération de la sécrète, qui est une attaque de décodage. Pour se prémunir de ces types d'attaques, d'autres algorithmes ont été proposés. Il s'agit des algorithmes *Black-Gray* et *Back-flipping*. Cependant, si ce dernier offre un DFR négligeable, le *Black-Gray* quant à lui permet une implémentation en temps constant plus performante.

Nous avons ainsi modifié le décodage de l'algorithme *Black-Gray*, pour obtenir un autre algorithme qu'on a appelé ici le *Black-Gray-flipping*, en vue de comparer son DFR avec les autres algorithmes que sont le *Black-Gray* et le *Back-flipping*. Cette modification nous a permis de réduire le nombre d'étapes de décodage. En simulant les différents algorithmes pour 11, 7 et 5 itérations, nous avons ainsi démontrer que le *Black-Gray-flipping* permet d'obtenir un DFR plus bas que le *Black-Gray* et qui reste assez proche du *Back-flipping*.

Une orientation possible des futures recherches serait, d'une part, d'optimiser les variantes *Black-Gray* et *Back-flipping* en vue de réduire d'avantage le DFR, et d'autre part, d'effectuer des implémentations physiques sécurisées sur FPGA et sur AVX.

BIBLIOGRAPHIE

[1] R. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN Prog. Rep., Jet Prop. Lab., California Inst. Technol., Pasadena, CA, pages 114 116, Jan. 1978.

[2] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto, "MDPCMcEliece: New McEliece variants from moderate density parity-check codes," in *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, 2013, pp. 2069–2073.

[3] Matthieu Lequesne. Side Channel Key Recovery Attacks on QC-MDPC Codes. Internship report – MPRI M2, 2017.

[4] T. Chou. Qcbits: Constant-time small-key code-based cryptography. In Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings, pages 280–300, 2016.

[5] J. Chaulet and N. Sendrier. Worst case QC-MDPC decoder for McEliece cryptosystem. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 1366–1370, 2016.

[6] Deneuville, J.C., Gaborit, P., Z´emor, G.: Ouroboros: A simple, secure and efficient key exchange protocol based on coding theory. In: PQCrypto 2017. LNCS, vol. 10346, pp. 18-34. Springer (2017).

[7] A. Kerckhoffs. La cryptographie militaire - partie 1. *Journal des sciences militaires*, jan. 1883. http://www.petitcolas.net/fabien/kerckhoffs/

[8] Bruce Schneier. *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)*. John Wiley & Sons Inc., 1996.

[9] Whitfield Diffie and Marin E. Hellman: New Directions in Cryptography. *IEEE Transactions on Information Theory*, nov. 1976.

[10] Ronald Riverst, Adi Shamir and Leonard Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM, Vol. 21*, 1978.

[11] Niels Ferguson, Bruce Schneier and Tadayoshi Kohno. Cryptography Engineering: Design, Principles and Pratical Applications. 2010.

[12] Claude Elwood Shannon: A mathematical theory of communication. The Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, July, October 1948.

[13] Richard W. Hamming: Error detecting and error correcting codes. Bell System Technical Journal, 29(2):147-160, April 1950.

[14] Alain Soyeur, François Capaces et Emmanuel Vieillard-Baron. Cours de Mathématiques Sup MPSI PCSI PTSI TSI. En partenariat avec l'association Sésamath et Les-Mathématiques.net. 23 Mars 2011.

[15] Julia Chaulet. *Etude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques*. Thèse de doctorat, University Pierre et Marie Curie, March 2017.

[16] El Mamoun SOUIDI. Théorie des codes correcteurs d'erreurs I. Cours de Master spécialisé Codes, Cryptographie et sécurité de l'information, Université Mohamed V – AGDAL. 2011.

[17] Pierre-Louis Cayrel. *Construction et optimisation de cryptosystèmes basés sur les codes correcteurs d'erreurs*. Thèse de doctorat, Université de Limoges, 2008.

[18] Tania Richmond. *Implantation sécurisé de protocoles cryptographiques basés sur les codes correcteurs d'erreurs*. Thèse de doctorat, Université Jean Monnet, Octobre 2016.

[19] Geatan Murat. *Résultats de polynômes de Ore et Cryptosystèmes de McEliece sur des Codes Rang faiblement structurés*. Thèse de doctorat, Université de Limoges, Décembre 2014.

[20] Valery Denisovich Goppa: A new class of linear error-correcting codes. Problemy Peredachi Informatsii, 6(3):24-30, September 1970.

[21] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Review, Vol. 41, No. 2, pp 303-332, 1999.

[22] Bhaskar Biswas et Nicolas Sendrier. McEliece cryptosystem implementation: Theory and practice. *Dans* Johannes Buchmann, et Jintai Ding, éditeurs. *Post-Quantum Cryptography*, volume 5299 de *Lecture Notes in Computer Science*, pages 47-62. Springer Berlin / Heidelberg, 2008.

[23] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, pages 15(2):159-166, 1986.

[24] Yuan Xing Li, Robert Deng, and Xin Mei Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Trans. On Information Theory*, 40(1):271-273, janvier 1994.

[25] Vladimir Michilovich Sidelnikov. A public-key cryptosytem based on Reed-Muller codes. *Discrete Math. Appl.*, 4(3):191–207, 1994.

[26] Lorenz Minder and Amin Shokrollahi. Cryptanalysis of the Sidelnikov cryptosystem. In *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Comput. Sci.*, pages 347–360, Barcelona, Spain, 2007.

[27] Heeralal Janwa and Oscar Moreno. McEliece public key cryptosystems using algebraic geometric codes. *Des. Codes Cryptogr*, 8(3):293–307, 1996.

[28] Cédric Faure and Lorenz Minder. Cryptanalysis of the McEliece cryptosystem over hyperelliptic curves. In *Proceedings of the eleventh International Workshop on Algebraic and Combinatorial Coding Theory*, pages 99–107, Pamporovo, Bulgaria, June 2008.

[29] Alain Couvreur, Irene Márquez-Corbella, and Ruud Pellikaan. A polynomial time attack against algebraic geometry code based public key cryptosystems. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2014*, pages 1446–1450, June 2014.

[30] Chris Monico, Joachim Rosenthal, and Amin A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, page 215, Sorrento, Italy, 2000.

[31] Philippe Gaborit. Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, Bergen, Norway, March 2005.

[32] Ayoub Otmani, Jean-Pierre Tillich, and Léonard Dallot. Cryptanalysis of McEliece cryptosystem based on quasi-cyclic LDPC codes. In *Proceedings of First International Conference on Symbolic Computation and Cryptography*, pages 69–81, Beijing, China, April 28-30 2008. LMIB Beihang University.

[33] Marco Baldi and Franco Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2591–2595, Nice, France, June 2007.

[34] Marco Baldi and Franco Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2591–2595, Nice, France, June 2007.

[35] R. G. Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press, 1963.

[36] D. J.C. MacKay and R. M. Neal. Near shannon limit performance of Low Density Parity Check codes. *Electronics Letters*, 32:1645-1646, 1996.

[37] Pierre-Alain Fouque and Gaëtan Leurent. Cryptanalysis of a hash function based on quasi-cyclic codes. In *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008. Proceedings*, volume 4964 of *Lecture Notes in Comput. Sci.*, pages 19–35. Springer, 2008

[38] Atsushi Yamada. QC-MDPC KEM: A Key Encapsulation Mechanism Based on the QC-MDPC McEliece Encryption Scheme. ISARA Corporation 30 Novembre 2017.

[39] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3): 384–386, 1978.

[40] R. Misoczki, J-P. Tillich, N. Sendrier, and P. S. L. M. Barreto, "MDPCMcEliece: New McEliece variants from moderate density parity-check codes," *IACR Cryptology ePrint Archive, Report2012/409*, vol. 2012, 2012.

[41] S. Heyse, I. von Maurich, and T. Güneysu, "Smaller keys for code-based cryptography: QC-MDPC McEliece implementations on embedded devices," in *Cryptographic Hardware and Embedded Systems – CHES 2013*, ser. Lecture Notes in Comput. Sci., G. Bertoni and J. Coron, Eds., vol. 8086. Springer, 2013, pp. 273–292.

[42] Qian Guo, Thomas Johansson, Paul Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. In: Asiacrypt 2016. LNCS, vol. 10031, pp. 789-815. Springer (2016).

[43] Nicolas Sendrier and Valentin Vasseur. On the decoding failure rate of QC-MDPC bit-flipping decoders. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography 2019*, volume 11505 of *LNCS*, pages 404-416, Chongquing, China, May 2019. Springer.

[44] Ingo Von Maurich, Tobias Oder, and Tim Güneysu. Implementing QC-MDPC McEliece encryption. ACM Trans. Embed. Comput. Syst., 14(3):44:1–44:27, April 2015.

[45] Drucker N, Gueron S, Kostic D. On Constant-Time QC-MDPC Decoding with Negligible Failure Rate. IACR Cryptology ePrint Archive. 2019, 1289.

[46] Markus Punnar. Bachelor's Thesis: Cryptosystem for Post-Quantum Age Based on Moderate-Density Parity-Check (MDPC) Codes. UNIVERSITY OF TARTU, May 2020.

[47] Carlos Aguilar Melchor, Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Guneysu, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor. BIKE. Second round submission to the NIST post-quantum cryptography call, April 2019.

[48] Nicolas Sendrier et Valentin Vasseur. About Low DFR for QC-MDPC Decoding. International Conference on Post-Quantum Cryptography, PQCrypto 2020: Post-Quantum Cryptography pp 20-34.

[49] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, et Jean-Pierre Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. *Dans* Henri Gilbert, éditeur. *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 de *Lecture Notes in Computer Science*, pages 279-298. Springer Berlin / Heidelberg, 2010.

[50] N. Patterson. The algebraic decoding of Goppa codes. IEEE Transactions on Information Theory, 21(2) :203–207, Mar. 1975.

[51] E. Prange. The use of information sets in decoding cyclic codes. IRE Transactions, IT-8 : S5–S9, 1962.