N° d'ordre : 74/2017-C/MT

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediène

Faculté de Mathémathiques

THESE

Présentée pour l'obtention du diplôme de DOCTORAT 3eme Cycle

En : MATHEMATIQUES

Spécialité : Mathématiques Fondamentales et Cryptographie

Par : MOUFEK Hamza

Sujet

Les Codes Correcteurs pour la Cryptographie

Soutenue publiquement, le 23/10/2017, devant le jury composé de :

M. K. BETINA	Professeur à l'USTHB	Président
Mme. K. GUENDA	Maître de Conférence/A à l'USTHB	Directrice de thèse
M. M.O. HERNANE	Professeur à l'USTHB	Examinateur
M. A. ALI-PACHA	Professeur à l'USTO Oran	Examinateur
M. L. NOUI	Professeur à l'Université de Batna 2	Examinateur
Mme. F. MAMACHE	Maître de Conférence/A à l'USTHB	Examinatrice
Mme. F. BELKREDIM	Maître de Conférence/A à l'U. de Chlef	Examinatrice

A la mémoire de Dr. Aini Laoudi.
A mes parents.

Remerciements

Cette partie est sans doute la plus difficile à écrire dans une thèse. Il n'est pas toujours évident de trouver les mots justes pour remercier toutes les personnes qui ont contribué d'une manière ou d'une autre à l'aboutissement d'une thèse.

Tout d'abord, je remercie chaleureusement Dr. Kenza GUENDA, ma directrice de thèse, avec qui au cours de ces quatre ans de thèse, j'ai passé des moments agréables, vu sa générosité et son enthousiasme scientifique. Elle était disponible tout le temps, elle m'a aidé à bénéficier d'un stage de courte durée chez Pr. Ayoub Otmani, où j'ai appris beaucoup de choses sur les primitives cryptographiques basées sur la théorie des codes. Elle a aussi présenté nos travaux de recherche aux États-Unis, au Portugal et récemment au Canada. Qu'elle trouve ici l'expression de ma profonde reconnaissance et de mon admiration.

Je remercie Pr. Kamel Betina pour m'avoir fait l'honneur d'accepter de présider le jury de ma thèse.

Je remercie Pr. Adda Ali-Pacha, Dr. Fatma Zohra Belkredim, Pr. Mohand Ouamar Hernane, Dr. Fatiha Mamache et Pr. Lemnouar Noui pour avoir accepté de faire partie du jury de ma thèse.

Des remerciements particuliers à Dr. Fatiha Mamache pour son aide, son soutien et surtout ses conseils précieux avant et durant ma thèse.

Je souhaite ensuite remercier Ayoub Otmani, Professeur à l'université de Rouen qui m'a accueilli au sein du laboratoire LITIS pour une durée de trois mois. Grâce à ses explications au tableau et ses conseils, j'ai pu avancer dans mes travaux de recherche.

Je souhaite remercier très sincèrement Pr. Mohand Ouamar Hernane, Pr. Abderrahmane Nitaj et Pr. Alain Togbe d'avoir organisé des écoles de recherche CIMPA à Tipaza, Nouakchott et Abidjan. C'était une bonne occasion pour moi de rencontrer et discuter avec des spécialistes dans le domaine de la cryptographie et la théorie des nombres, cette expérience était bénéfique pour ma thèse et m'a permis d'améliorer mes connaissances.

Je n'oublie pas bien sûr de remercier Dr. Pierre-Louis Cayrel, qui était disponible par mail et il répondait à toutes mes questions rapidement, il m'a même invité pour lui rendre visite à Saint-Etienne afin d'avancer dans mes recherche.

Je remercie aussi mes coauteurs Pierre-Louis Cayrel, T. Aaron Gulliver et Mahdjoubi Roumaissa.

Je n'oublie pas mes amis Adnène, Nabil, Sofiane et Zakaria qui m'ont accompagné et

conseillé depuis mes débuts.

Je tiens à remercier Mme Meguedmi, Mme Batoul, Mme Benahmed, Mme Zoubir, Mme Saadallah et Mme Djoumakh pour leurs encouragements.

Je tiens à remercier aussi toute ma famille et tous mes amis pour leur aide et leur soutien durant ces années.

Enfin, j'aimerais exprimer ma reconnaissance à mes parents pour leur soutien continu dans mes études.

Résumé

Cette thèse est consacrée à l'étude des primitives cryptographiques basés sur les codes correcteurs d'erreurs. L'avantage principal de ce type de primitives est la rapidité des processus de chiffrement et de déchiffrement, mais il n'est pas encore utilisé dans la pratique car il nécessite une grande taille de clé.

Dans ce manuscrit nous avons proposé des systèmes de chiffrement à clé publique basés sur différents codes (les codes convolutifs, QC-MDPC, QC-LDPC et les codes en métrique rang) pour résister à des attaques qui ont été réalisées. De plus nous avons réussi à réduire la taille de la clé publique.

Nous avons aussi construit un schéma d'identification basé sur les codes QC-MDPC qui est plus performant par rapport aux schémas existants.

Mots-clés : Cryptographie basée sur les codes, Cryptosystème de McEliece, les codes convolutifs, les codes quasi-cycliques, identification.

Abstract

The dissertation is devoted to the study of cryptographic primitives based on error-correcting codes. The main advantage of this type of primitives is the fast encryption and decryption process, but it is not used in practice because it requires a large key size.

In this manuscript we proposed several public key encryption systems based on different codes (convolutional codes, QC-MDPC, QC-LDPC and rank metric codes) in order to resist to known attacks. In addition, we managed to reduce the public key size.

We also constructed an identification scheme based on QC-MDPC codes, that is more efficient compared to existing protocols.

Keywords: Code based cryptography, McEliece cryptosystem, Convolutional codes, Quasi-cyclic codes, identification.

Table des matières

Table des figures

Liste des tableaux

Introduction

Après les algorithmes de Shor [84,85], l'apparence de l'ordinateur quantique rend vulnérable les systèmes de chiffrement à clé publique les plus populaires, dont leur sécurité est basée sur les problèmes difficiles de la théorie des nombres (factorisation, logarithme discret ...etc) .
Pour cette raison, nous nous intéressons aux algorithmes cryptographiques post-quantiques qui résistent à toutes les attaques quantiques connues.
Parmi les domaines de la cryptographie post-quantique, nous mentionnons : la cryptographie à base de code, la cryptographie basée sur les réseaux euclidiens et la cryptographie multivariée. Cela motive l'utilisation de la cryptographie basée sur les codes.

En 1978, Robert J. McEliece [60] a inventé le premier système de chiffrement post-quantique basé sur la théorie des codes algébriques. Sa sécurité est basée sur deux problèmes difficiles : le problème du décodage des codes linéaires aléatoires (qui est prouvé NP-complet [10]) et l'indistinguabilité des codes de Goppa des codes aléatoires. Niederreiter [73], en 1986, a introduit un nouveau schéma basé sur les codes de Reed-Solomon généralisés (GRS) en utilisant une matrice de contrôle de parité. Les deux systèmes ont une sécurité équivalente [51]. Les mauvaises nouvelles sont arrivées en 1992, lorsque Sidelnikov et Shestakov [88] révèlent une attaque algébrique sur les codes GRS. L'avantage principal du cryptosystème de McEliece est la rapidité des processus de chiffrement et de déchiffrement (il est plus rapide que RSA), mais il n'est pas utilisé dans la pratique car il nécessite une grande taille de clé. Pour réduire la taille des clés, plusieurs solutions ont été suggérées en utilisant des codes quasi-cycliques [3, 9, 34], des codes quasi-dyadiques [64] ...etc. De nombreuses autres familles de codes ont été utilisées pour construire un cryptosystème basé sur : les codes de Reed-Muller [87], les codes convolutionnels [54], les codes MDPC [63] ...etc.
Différentes attaques ont été réalisées contre ces protocoles. Il existe deux catégories d'attaques : les attaques structurelles et les attaques par décodage. Par exemple, Minder et Shokrollahi [62] ont introduit une attaque algébrique contre le schéma basé sur les codes Reed-Muller, Faugère et al. [24] ont cassé presque tous les systèmes basés sur des structures quasi cycliques ou quasi-dyadiques (à l'exception du cas binaire de [64]), Landais et Tillich [50] ont réalisé une attaque efficace sur le McEliece PKC basé sur les codes convolutionnels [54], le système proposé par Baldi et al. [1], basé sur les codes GRS, a été attaqué dans [21] et très récemment Bardet et al. [5] ont réussi à briser la variante de

McElice basée sur des codes polaires [86].

Aujourd'hui, seuls deux systèmes sont sécurisés (à notre connaissance) : celui basé sur les codes Goppa [60] et celui basé sur les codes QC-MDPC [63].

Cette thèse est organisée de la façon suivante :

Le premier chapitre sera consacré aux préliminaires sur la théorie des codes correcteurs d'erreurs.

Au deuxième chapitre, nous avons donnés quelques rappels sur les notions de base de cryptographie, que nous avons utilisés dans les chapitres qui suivent.

Dans le troisième chapitre, nous avons proposé une nouvelle variante du protocole d'identification de Stern. Notre schéma est basé sur la difficulté de trouver la matrice de contrôle à partir de la matrice génératrice publique et sur la difficulté du décodage d'un code linéaire aléatoire. En outre, notre système a une taille de clé réduite par rapport à d'autres protocoles.

Dans le quatrième chapitre, nous introduisons une nouvelle variante du cryptosystème de McEliece qui est basée sur les codes convolutionnels et les générateurs autorétrécissants , nous prouvons que notre système est sûr contre certaines attaques robustes.

Au cinquième chapitre nous présentons un nouveau système de chiffrement, inspiré du cryptosystème inventé par McEliece. Notre schéma se base sur les codes QC-LDPC et QC-MDPC. L'avantage de notre système est la sécurité contre les attaques basées sur les matrices creuses. De plus, la taille des clés est réduite.

Au chapitre 6, nous construisons une nouvelle variante du cryptosystème GPT pour augmenter le niveau de sécurité contre les attaques récentes. Notre cryptosystème est basé sur un problème difficile ; Le décodage par syndrome en métrique rang (RSD). Cette variante présente une construction d'un code produit composé de deux types de codes différents, un code LRPC (Low Rank Parity Check) et un code convolutionnel PUM (mémoire unitaire partielle).

Dans le dernier chapitre, nous proposons une nouvelle version du système de chiffrement McEliece basé sur la forme Smith des codes convolutifs. Nous utilisons la forme Smith pour cacher une partie du code dans la matrice publique et nous laissons l'autre partie secrète. La partie secrète sera ensuite utilisée pour le déchiffrement. Nous cachons cette partie en multipliant à gauche par une matrice aléatoire et on rajoute une matrice aléatoire qui satisfait quelques conditions. Notre système a une clé publique de petite taille par rapport au cryptosystème de McEliece original et résiste à l'unique attaque par décodage contre la structure convolutionnelle présentée lors de la conférence PQCrypto 2013 par Landais et Tillich. De plus, l'attaque par recherche exhaustive est infaisable contre notre système.

CHAPITRE 1

GÉNÉRALITÉS SUR LES CODES

Par codes, on peut entendre plusieurs concepts distincts : les codes pour la cryptographie, les codes pour la compression, les codes pour la correction d'erreurs…etc.
Dans ce chapitre, nous nous intéressons aux codes correcteurs d'erreurs. Nous introduisons les définitions de base et les résultats essentiels des codes linéaires et cycliques.
Les codes correcteurs servent à protéger l'information d'erreurs de transmission ou de stockage. L'une des grandes familles de codes est les codes en bloc qui sont très utilisés dans les applications téléinformatiques classiques puisque leur codage/décodage est plus simple et plus rapide.

1.1 Les codes en bloc

Le codage par bloc consiste à découper un message en plusieurs blocs (mots) de la même taille k en traitant chaque bloc indépendamment l'un après l'autre, pour le transformer en un mot codé de taille n. Le code introduit une redondance (c'est la différence $n-k$), cette multiplication de l'information est la clé de tout code correcteur.
Le rapport $F = d/n$ représente la fiabilité de la transmission et la quantité $R = k/n$ est appelée rendement(taux de codage) du code.
Soit $\mathbb{F}_q$ un ensemble fini à q éléments (bits, alphabets) et soit k et n deux entiers strictement positifs avec $k \leq n$. L'ensemble des messages sera une partie E de $\mathbb{F}_q^n$.
Un code en bloc est une application injective :

$$\begin{array}{rccc} \phi & : & E & \longrightarrow & \mathbb{F}_q^n \\ & & m = (m_1, m_2, \ldots, m_k) & \longrightarrow & c = (c_1, c_2, \ldots, c_n) \end{array}$$

appelée "application de codage" ou "encodeur".
L'application ϕ est donc une bijection de $\mathbb{F}_q^k$ sur C et C peut être considéré comme l'ensemble de tous les messages possibles. C est appelé "code" et ses éléments sont les "mots

du code".
L'ensemble des mots du code, constitue généralement un sous-ensemble très réduit de $\mathbb{F}_q^n$.
Un alphabet A est un ensemble fini, dont les éléments sont appelés symboles. Dans la suite de notre chapitre, on prend le corps fini $\mathbb{F}_q$ comme alphabet.

Définition 1.1.1. *Soit $\mathbb{F}_q$ un alphabet, n un entier naturel non nul. Un code en blocs est une partie C de $\mathbb{F}_q^n$ de cardinal $|C| = M$. Tout élément de C est appelé mot du code et n est la longueur du code C.*

1.1.1 Distance minimale d'un code

Un paramètre important d'un code est la distance minimale entre les mots de code. Commençons par définir la notion de distance entre deux mots.

Définition 1.1.2. *Soit A un alphabet. Soit d_H une application telle que :*

$$\begin{aligned} d_H : A^n \times A^n &\longrightarrow \mathbb{R}_+ \\ (x,y) &\longrightarrow d_H(x,y) = card\{i : x_i \neq y_i\} \end{aligned} \tag{1.1}$$

c'est-à-dire le nombre de symboles distincts entre deux mots.
d_H est appelée distance de Hamming.

Cette distance vérifie les propriétés usuelles des distances.
Un invariant important d'un code est la distance minimale entre les mots de code.

Définition 1.1.3. *Soient C un code sur A et d_H la distance de Hamming sur $\mathbb{F}_q^n$.*
La distance minimale de C notée d est le minimum des distances de Hamming entre les mots du code. On a donc :

$$d(C) = min\{d_H(x,y) : x,y \in C \, et \, x_i \neq y_i\}$$

Un code de longueur n, de distance minimale d, contenant M mots, est un (n,M,d)-code sur $\mathbb{F}_q$.

Cette distance est une propriété essentielle d'un code, plus elle est grande plus on est capable de corriger plus d'erreurs.
Un code C peut détecter $d-1$ erreurs et corriger jusqu'à $e = \lfloor (d-1)/2 \rfloor$ erreurs.
Notons $A_q(n,d)$ le nombre maximum de mots que peut contenir un code de longueur n et de distance minimale d et q représente le cardinal de l'alphabet.
Il existe des liens entre n, d, M, q et $A_q(n,d)$. En voici quelques uns :

Proposition 1.1.1. *soit n et d la longueur et la distance minimale d'un code C et e la capacité de correction d'erreurs, alors on a :*

$$\frac{q^n}{V_q(n,d-1)} \leq A_q(n,d) \leq \frac{q^n}{V_q(n,e)}$$

L'inégalité de gauche s'appelle "borne de Gilbert-Varshamov" et celle de droite, c'est "la borne de Hamming".

La détermination des nombres $A_q(n,d)$ est un problème difficile non encore résolu. A l'heure actuelle, on ne connait que des encadrements de ces nombres.
La borne de Hamming nous permet de définir la notion de code parfait.
Un code est dit parfait si et seulement si la borne de Hamming est atteinte.
Donc un code parfait ne contient aucune redondance inutile.

Remarque 1.1.1. *Il y a peu de codes parfait, par exemple le code de Hamming et le code à répétition de longueur impaire sont parfaits.*

Une autre majoration du nombre $A_q(n,d)$ est donnée par la proposition suivante :

Proposition 1.1.2. *Soit C un (n,M,d)-code. Alors on a l'inégalité suivante :*

$$A_q(n,d) \leq q^{n-d+1}$$

Cette inégalité est appelée "borne de Singleton".

La théorie de la complexité est un domaine qui étudie formellement la quantité de ressources (en temps et en espace) nécessaire pour la résolution de problèmes au moyen de l'exécution d'un algorithme.
Etant donné un code C de cardinal M, pour calculer sa distance minimale, il faut faire $\binom{M}{2} = \frac{M(M-1)}{2}$ calculs de distances. Ce code est de complexité $\mathcal{O}(nM^2)$.
Pour pouvoir travailler avec des codes, il faut chercher à les construire en mettant plus de structures afin que ce calcul soit minimisé.

1.2 Les codes linéaires

La première structure que nous ajoutons est la structure linéaire. A cause de leurs propriétés algébriques, les codes linéaires sont les plus étudiés d'un point de vue mathématique.
Les codes linéaires correspondent à une très large majorité de type de codes correcteurs. Ce sont les codes les plus utilisés dans la pratique. Leur puissance est due à leur richesse mathématique et de la présence d'algorithmes efficaces d'encodage et de décodage. Ils sont utilisés dans plusieurs domaines technologiques, parmi lesquels on cite : Les transmissions spatiales, les supports de stockage(CD,DVD)... etc.
Soit $\mathbb{F}_q$ un corps fini à q éléments où $q = p^m$ avec p premier et m un entier positif.

Un code linéaire de longueur n et de dimension k sur F_q, est tout sous-espace vectoriel C de F_q^n.
Un code linéaire de dimension k sur F_q contient q^k mots.
Le poids de Hamming d'un code linéaire, noté $wt(x)$, est le nombre de coordonnées non nulles d'un mot (ou sa distance de Hamming par rapport au mot nul), c'est-à-dire :

$$wt(x) = card\{i : x_i \neq 0\} = d_H(x, 0)$$

Le poids minimal d'un code linéaire C est le nombre :

$$w(C) = min\{wt(x)/x \in C\, et\, x \neq 0\}$$

Proposition 1.2.1. *La distance minimale d'un code linéaire est égale au plus petit poids non nul de ce code.*

Ce résultat est très important en pratique car $wt(C)$ est beaucoup plus facile à calculer que $d(C)$, donc il est plus rapide de déterminer la distance de Hamming quand il s'agit de code linéaire (si $|C| = M$ on doit calculer $M - 1$ poids seulement).
La proposition suivante donne une majoration de la distance minimale :

Proposition 1.2.2. *Soit un code* $[n, k, d]$. *On a l'inégalité suivante :*

$$d \leq n - k + 1$$

Cette inégalité représente la borne de Singleton d'un code linéaire.

Si la borne de Singleton est atteinte, le code est dit MDS (Maximum Distance Separate).

Il y a deux manières de représenter un code linéaire à l'aide des matrices. Soit on introduit un homomorphisme dont le code est l'espace vectoriel image, on obtient ainsi la notion de matrice génératrice, soit on introduit un homomorphisme dont le code est le noyau ; on obtient ainsi la notion de matrice de contrôle de parité.

1.2.1 Matrice génératrice

Tout code linéaire est défini par sa matrice génératrice. Les codes linéaires sont des codes dont chaque mot du code est obtenu après transformation linéaire des bits du mot initial. Ces codes sont caractérisés par une matrice dont les lignes forment une base du code, cette matrice est appelée matrice génératrice.
Un code linéaire C a précisément $\frac{1}{k!}\prod_{i=0}^{k-1}(2^k - 2^i)$ bases différentes, donc un code peut avoir plusieurs matrices génératrices.
Pour former un mot de code, on calcule le produit d'un vecteur ligne par la matrice génératrice. Alors la relation entre le code et la matrice génératrice est définie par l'ensemble :

$$C = \{c \in F_q^n / \exists x \in F_q^k : c = xG\}$$

La matrice génératrice peut avoir une forme systématique (canonique) qui rend le codage lisible.
Une matrice génératrice d'un (n,k)-code linéaire est sous forme systématique (normalisée) si la matrice formée par les k première colonnes est la matrice identité. Si un code possède une matrice sous une telle forme, on dit que ce code est systématique.
Puisqu'un code linéaire est un sous-espace d'un espace vectoriel, il est le noyau d'une certaine transformation linéaire. En particulier, il existe une $(n-k)\times n$ matrice H, appelée matrice de contrôle de parité pour le (n,k)-code C.

Définition 1.2.1 (Matrice de contrôle)**.** *Soit C un (n,k)-code sur $\mathbb{F}_q$, il existe une $(n-k)\times n$ matrice H, appelée matrice de contrôle de parité, sur $\mathbb{F}_q$ qui est de rang $n-k$ et vérifie :*

$$C = Ker\phi = \{c \in \mathbb{F}_q^n / c.H^t = 0\}$$

La matrice de contrôle permet de savoir si un mot reçu est un mot du code en calculant son syndrome. Si le syndrome du mot est nul, alors ce mot appartient au code.
Pour un code linéaire C, le syndrome d'un mot de code x est $s = x.H^t$; où H représente la matrice de contrôle du code.
Les codes linéaires fournissent une solution adéquate au problème du décodage au plus proche voisin i.e. de manière à minimiser le poids de l'erreur, au moyen des classes latérales.

Définition 1.2.2. *Soit C un (n,k)-code linéaire et u un élément de $\mathbb{F}_q^n$. On appelle classe latérale de C l'ensemble $u+C$ défini par : $u+C = \{u+x; x \in C\}$.*
Deux éléments de $\mathbb{F}_q^n$ sont dans la même classe latérale si et seulement s'ils possèdent le même syndrome : $x^T H = y^T H \Leftrightarrow x-y \in C$ i.e. x et y sont dans la même classe modulo C.
Dans chaque classe modulo C (ensemble sous la forme $u+C$), on considère l'élément ayant le poids minimal. On l'appelle le leader de la classe.

Le syndrome est un objet fondamental dans le décodage des codes linéaires. On en déduit alors un algorithme de décodage que nous décrivons ci-dessous.

1.2.2 Méthode du décodage par syndrome

Soit un code linéaire C de paramètres (n,k,d), de matrice de parité H et de matrice génératrice G. Le destinataire reçoit un mot bruité $y = x+e$, où e est un vecteur d'erreur contenant au plus $(d-1)/2$ positions d'erreur. Le fait qu'il y a une bijection entre les syndromes et les classes latérales, on obtient l'algorithme de décodage suivant :

1. On calcule le syndrome $s(e)$ de chaque leader de classe e et on construit une table de syndromes en associant à chaque syndrome z le leader de la classe dont il est issu $f(z)$,

2. On calcule $z = s(y)$,

3. On décode y comme $y - f(z)$ au moyen de la table des syndromes.

4. On obtient alors le mot du code x tel que $y = x + f(z)$.

La complexité de ce type de décodage est $\mathcal{O}(2^{n-k})$. En théorie de la complexité, un problème NP-complet est un problème de décision vérifiant les propriétés suivantes :

- Il est possible de vérifier une solution efficacement (en temps polynomial) ; la classe des problèmes vérifiant cette propriété est notée NP.
- Tous les problèmes de la classe NP se ramènent à celui-ci via une réduction polynomiale ; cela signifie que le problème est au moins aussi difficile que tous les autres problèmes de la classe NP.

Un problème NP-difficile est un problème qui remplit la seconde condition, et donc peut être dans une classe de problème plus large et donc plus difficile que la classe NP. En 1978, Berlekamp, McEliece et Van Tilborg ont montré que le problème de la recherche de mots de poids et de syndrome fixé est un problème NP-complet.
Une classe très importante des codes linéaires est obtenue en rajoutant la structure de cyclicité.
La notion de codes cycliques apparait pour la première fois en 1957 par l'intermédiaire de Prange [78]. Leur popularité s'explique grâce à leurs propriétés structurelles qui offrent une protection contre les erreurs isolées et les erreurs en rafale.
La structure d'un code cyclique est telle que si tout mot du code est décalé de manière cyclique, le résultat est aussi un mot du code.
Soit T l'application de décodage circulaire (shift), donnée par :

$$\begin{array}{rccc} T & : & \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_q^n \\ & & (x_1, x_2, \ldots, x_n) & \longrightarrow & (x_n, x_1, \ldots, x_{n-1}) \end{array}$$

Définition 1.2.3. *C est un code cyclique si et seulement si pour tout $c \in C$, $T(c) \in C$, c'est-à-dire que l'ensemble des mots du code est stable par décalage circulaire.*
On peut dire que $(c_n, c_1, \ldots, c_{n-1})$ est le shift de $(c_1, c_2, \ldots, c_n)$.

Remarque 1.2.1. *les codes cycliques sont aussi appelés CRC (Cyclic Redundant Codes), ou codes polynomiaux.*

1.3 D'autres familles de codes

1.3.1 Les codes BCH

Les codes BCH (reprenant les initiales de ses inventeurs : Bose, Ray-Chaudhuri et Hocquenghem) ont été mis au jour par Hocquenghem [45] autour de 1959, et de façon indépendante par Bose et Ray-Chaudhuri [12] en 1960. Ils furent généralisés à tous les corps finis par Gorenstein et Zierler.

Avant de définir ce qui est un code BCH, nous allons tout d'abord introduire quelque notions.
Un groupe infini est dit monogène s'il est engendré par un de ses éléments. Cet élément est appelé générateur du groupe.

Définition 1.3.1. *On appelle élément primitif de $\mathbb{F}_q$ tout générateur du groupe multiplicatif $\mathbb{F}_q^*$.*

Proposition 1.3.1. *Tout corps fini $\mathbb{F}_q$ admet un élément primitif β, c'est-à-dire un élément β tel que : $F_q = \{0, 1 = \beta^0, \beta, \beta^2, \ldots, \beta^{q-2}\}$ et $\beta^i = 1$ si et seulement si $(q-1)/i$.*

D'un élément primitif β de $\mathbb{F}_q$, on obtient la racine n-ième de l'unité : $\xi = \beta^{(q^m-1)/n}$ et donc l'ensemble des racines n-ièmes de l'unité dans $\mathbb{F}_{q^m}$ est :

$$U_n =< \xi >= K \in F_{q^m}/K^n = 1$$

Un sous ensemble $W \subseteq U_n$ est appelé consécutif (par rapport à ξ), s'il existe des entiers $b \geq 0$ et $\delta \geq 2$ tels que :

$$W = \{\xi^b, \xi^{b+1}, \ldots, \xi^{b+\delta-2}\}.$$

Définition 1.3.2. *Soit $W = \{\xi^b, \xi^{b+1}, \ldots, \xi^{b+\delta-2}\}$ un sous ensemble consécutif de U_n pour un certain $b \geq 0$ et un certain δ avec $n > \delta \geq 2$. Définissons le polynôme : $g = ppcm\{M_{\xi^{b+i}}/i \in \delta - 1\}$ où $M_{\xi^{b+i}}$ est le polynôme minimal de ξ^{b+i} sur $\mathbb{F}_q$.*
Le code C avec le polynôme générateur g est appelé le code BCH engendré par W.
Si $n = q^n - 1$, le code est dit primitif puisque dans ce cas ξ est également un élément primitif de $\mathbb{F}_{q^m}$. En outre, si $b = 1$, le code C est appelé un code BCH au sens strict.

Définition 1.3.3. *On considère un code cyclique C sur $\mathbb{F}_q$ avec $g(x)$ son polynôme générateur. La variété de g (appelée aussi variété de C) sur $\mathbb{F}_q$ est définie comme suit :*

$$V(C) := V(g)$$

Chaque élément de $V(C)$ est appelé racine de C sur $\mathbb{F}_q$.

La dimension k d'un code BCH engendré par l'ensemble W d'ordre $\delta - 1$ satisfait l'inégalité :

$$k \geq n - m(\delta - 1) = n - m.|W|$$

Un paramètre important pour les codes cycliques est la distance construite.

Définition 1.3.4. *Soit C un code cyclique de longueur n sur $\mathbb{F}_q$ où q est premier avec n. Supposons que la variété de C contient $\delta - 1$ puissances consécutives de ξ, la racine n-ième de l'unité où $\delta \geq 2$. Alors la distance minimale de C est au moins δ, i.e. : $d(C) \geq \delta$. C'est-à-dire : $W = \{\xi^b, \xi^{b+1}, \ldots, \xi^{b+\delta-2}\} \subseteq V(C) \Rightarrow d(C) \geq \delta$.*
δ est dite distance construite de C.

1.3.2 Les codes de Reed-Solomon

Les codes Reed-Solomon sont considérés comme étant un sous-groupe de codes cycliques de la famille des codes BCH. Reed et Solomon [81] ont développé et publié leurs travaux sur une famille de codes qui portera leurs noms. Les codes de Reed-Solomon ont été découvert plus tôt dans le contexte des matrices orthogonales par Bush [13] en 1952. on appelle codes de Reed-Solomon cycliques sur $\mathbb{F}_q$, les codes BCH de longueur $n = q - 1$.

Proposition 1.3.2. *soit C un code de Reed-Solomon sur $\mathbb{F}_q$ de distance construite δ, alors : C a une distance minimale $d = \delta$ et une dimension $k = n - d + 1$.*

Les codes de Reed-Solomon ont été appliqués dans le monde de la conservation d'informations numérique, ainsi que dans les systèmes de communication. On retrouve par exemple le code $RS(255, 233, 33)$ qui est utilisé par la NASA dans les communications spatiales.

1.3.3 Les codes de Reed-Muller

Les codes de Reed-Müller sont des codes correcteurs linéaires. Cette famille de codes, initialement binaire, doit son nom aux travaux de D.E. Muller [72] qui proposa le principe du code et à Irving S. Reed [80] qui proposa une technique de décodage, publiés en 1954. On notera $RM(r, m)$ le code de Reed-Muller d'ordre r et de longueur 2^m, avec $0 \leq r \leq m$. Un code de Reed-Muller est défini de la façon suivante : $RM(0, m) = \{00\ldots0, 11\ldots1\}$, chaque mot est de longueur 2^m.
$Rm(m, m) = \{0, 1\}^{2^m}$
$RM(r, m) = \{(x, x + y) : x \in RM(r, m-1), y \in RM(r-1, m-1)\}$ pour $0 < r < m$. Ces codes sont très utiles pour la transmission sur des canaux très brouillés. En particulier, le code $RM(1, m)$ a une distance égale à la moitié de la longueur des mots du code. Le cas de $RM(1, 5)$ a été utilisé pour la transmission d'images de la planète Mars via le satellite Mariner-9. Ce code pouvait corriger jusqu'à 7 erreurs par mot du code.

1.3.4 Les codes de Goppa

Un code de Goppa et défini à partir d'un polynôme unitaire $g(x) = \sum_{i=0}^{t} g_i x^i$, dit polynôme de Goppa et d'un ensemble $L = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\} \subseteq \mathbb{F}_{q^m}$, avec $g_i \in \mathbb{F}_{q^m}$ pour tout $i = 0, \ldots, t$ et $g(\alpha_i) \neq 0$ pour tout $\alpha_i \in L$.
Étant donné un mot de code $c = (c_1, c_2, \ldots, c_n)$ sur $\mathbb{F}_q$, nous avons la fonction

$$R_c(x) = \sum_{i=1}^{n} \frac{c_i}{x - \alpha_i}$$

où $\frac{1}{x - \alpha_i}$ est l'unique polynôme tel que $(x - \alpha_i).\frac{1}{x - \alpha_i} \equiv 1 \mod g(x)$ avec un degré inférieur ou égal à $t - 1$.

Un code de Goppa $\Gamma(L,g)$ est constitué de tous les vecteurs c tels que $R_c(x) \equiv 0 \mod g(x)$ (c'est à dire que le polynôme $g(x)$ divise $R_c(x)$).
Le code de Goppa est défini par sa matrice de parité construite à partir de la matrice :

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

Jusqu'à présent, on ignore la dimension et la distance minimale exactes des code de Goppa, sauf dans certains cas particuliers ; on connaît juste des bornes sur ces quantités.

1.3.5 Les codes convolutionnels

Un code convolutionnel diffère d'un code en bloc par le fait que chaque bloc de n éléments en sortie ne dépend pas seulement des k entrées à un instant donné mais aussi des m blocs précédents.
Soit $\mathbb{F}_2$ le corps a deux éléments, binaire. Notons $\mathbb{F}_2[D]$ l'anneau des polynômes sur $\mathbb{F}_2$, $\mathbb{F}_2((D))$ le corps des séries de Laurent sur $\mathbb{F}_2$ et $\mathbb{F}_2(D)$ le corps des fonctions rationnelles sur $\mathbb{F}_2$.

Définition 1.3.5. *Un (n,k) code convolutionnel est un sous-espace de dimension k sur $\mathbb{F}_2(D)^n$.*

Dans le cadre de notre étude algébrique, nous nous intéressons aux codes possédant une matrice génératrice polynomiale.

Définition 1.3.6. *Soit C un (n,k) code convolutionnel sur $\mathbb{F}_2$.*
Une matrice génératrice $G(D)$ de C est une matrice $k \times n$ dont les lignes forment une base de C. Cette matrice est dite matrice génératrice polynômiale (MGP) si ses coefficients sont des polynômes.

Soit $X(D) = (x_1(D), x_2(D), \ldots, x_k(D))$ le vecteur d'entrée, à chacune des k entrées correspond une suite infinie que l'on écrit comme une série en l'indéterminée D.

$$x_i(D) = \sum_{t=0}^{\infty} x_i(t) D^t, \qquad 1 \leq i \leq k$$

et $C(D) = (c_1(D), c_2(D), \ldots, c_k(D))$ le vecteur de sortie, à chacune des n sorties correspond une suite infinie que l'on écrit :

$$c_i(D) = \sum_{t=0}^{\infty} c_i(t) D^t, \qquad 1 \leq i \leq n$$

Pour savoir si un mot appartient à un code , nous avons besoin de la matrice de contrôle (parité).

Définition 1.3.7. *Soit $G(D)$ une matrice génératrice du code C. Une matrice $H(D)$ de rang $n-k$ composée de $(n-k)\times n$ polynômes est une matrice de contrôle du code C si et seulement si :*

$$G(D)\, H^T(D) = 0$$

La matrice de parité d'un code convolutionnel sous sa forme systématique est une $(n-k)\times n$ matrice polynomiale, telle que :

$$H(D) = \begin{pmatrix} h_{1,1}(D) & \dots & h_{1,k}(D) & h_0(D) & & \\ \vdots & \dots & \vdots & & \ddots & \\ h_{n-k,1}(D) & \dots & h_{n-k,k}(D) & & & h_0(D) \end{pmatrix}$$

où les $h_{i,j}$ (D) correspondent aux polynômes générateurs de la matrice de contrôle du code tels que :

$$\begin{cases} h_{i,j}(D) = f_{j,i+k}(D) \\ h_0(D) = f_{1,1}(D) \end{cases} \quad \forall i = 1,\dots,n-k \text{ et } \forall j = 1,\dots,k.$$

1.3.6 Les codes QC-LDPC/MDPC

Les codes LDPC (Low Density Parity Check) ont été introduits en 1963 par Gallager [35]. Il a même proposé un algorithme de décodage efficace pour ces codes. Les codes LDPC sont basés sur des matrices de contrôle pseudo aléatoires de faible densité. Les codes MDPC (Moderate Density Parity Check) sont des codes à densité peu élevé par rapport aux codes LDPC.
Un code LPDC est un code dont la matrice de contrôle de parité H est de faible densité (matrice creuse). C'est à dire que les éléments de cette matrice sont majoritairement nuls.
On note T l'opérateur de décalage standard sur $\mathbb{F}^n$.
Un code linéaire est dit l-quasi-cyclique si et seulement s'il est invariant par rapport à T^l.
Maintenant, nous donnons une définition des codes quasi-cycliques LDPC/MDPC.

Définition 1.3.8. *Soit w le poids de chaque ligne d'une matrice de contrôle de parité $H \in \mathbb{F}_2^{r\times n}$. La densité est le taux $\frac{w}{n}$.*
Un (n,k,w)-LDPC/MDPC code C est un code linéaire de longueur n et de dimension k qui admit une matrice de contrôle de parité H à faibe/moyenne densité d'un poids des lignes w, constant.
Pour un code MDPC, les poids des lignes sont supposés être aux environs de $\mathcal{O}(\sqrt{n!})$.
Quand C est quasi-cyclique, il est appelé un (n,k,w)-QC-LDPC/QC-MDPC code avec $n = n_0 r$.

La matrice de contrôle de parité d'un code QC-LDPC/QC-MDPC est définie par la concaténation de blocks H_i tels que :

$$H = [H_0|\dots|H_{n_0-1}]$$

où H_i est une matrice circulante de taille $r \times r$ donnée par :

$$H_i = \begin{bmatrix} h_0 & h_1 & h_2 & \dots & h_{n_0-1} \\ h_{n_0-1} & h_0 & h_1 & \dots & h_{n_0-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & h_3 & \cdots & h_0 \end{bmatrix} \tag{1.2}$$

avec $i = 0, \dots, n-1$.
La matrice génératrice d'un code QC-LDPC/MDPC sous forme systématique est donnée par :

$$G = \left[\begin{array}{c|c} & (H_{n_0-1}^{-1} H_0)^T \\ & H_{n_0-1}^{-1} H_1)^T \\ I & \vdots \\ & H_{n_0-1}^{-1} H_{n_0-2})^T \end{array}\right]. \tag{1.3}$$

1.4 Conclusion

Dans ce chapitre, nous avons présenté les codes que nous allons mentionné dans les autre parties de la thèse. Au chapitre suivant, nous rappelons les notions de base sur la cryptographie à clé publique.

Chapitre 2

Introduction la cryptographie

La cryptologie est à la fois une discipline ancienne, mais également une science moderne, dont son nom signifie "science du secret". Elle est la conjonction de deux disciplines, la cryptographie qui est la science des écritures secrètes et la cryptanalyse qui consiste à dévoiler les faiblesses des systèmes cryptographiques. Le but de la cryptographie moderne ne se limite pas à garantir la confidentialité des données (protéger ses données de lectures non désirées), mais aussi à assurer leur authenticité (être sûr de l'origine des données) et leur intégrité (s'assurer que les données n'ont pas été modifiées).
Pour assurer ces usages, cette discipline scientifique s'intéresse à des primitives cryptographiques telles que la signature numérique, les fonctions de hachage, le chiffrement, la génération d'aléas...etc.
Un système cryptographique est constitué de :

1. Un espace de clés $\mathcal{K}$.
2. Un ensemble $\mathcal{M}_k$ de messages clairs, pour chaque $k \in \mathcal{K}$.
3. Un ensemble de cryptogrammes $\mathcal{C}_k$.
4. Un algorithme de chiffrement $\mathcal{E}_k$.
5. Un algorithme de déchiffrement $\mathcal{D}_k$.

tel que $\mathcal{D}_k(\mathcal{E}_k(m)) = m$ pour chaque texte clair $m \in \mathcal{M}_k$.
Il existe deux type de cryptographie : la cryptographie à clé secrète (symétrique) et la cryptographie à clé publique (asymétrique).
Dans un algorithmes de chiffrement symétrique les clefs de chiffrement et de déchiffrement sont les mêmes. Pour ce type de chiffrement, le partage d'une clé commune doit se faire par le biais d'un canal sûr, ce qui représente un inconvénient majeur car le canal de

transmission n'est pas toujours sécurisé.
C'est en 1976 qu'une solution au problème d'échange de clefs a été introduite par W. Diffie et M. Hellman [23]. Ils ont proposé aussi un modèle théorique de la cryptographie à clé publique.
Dans un schéma de chiffrement à clé publique, deux clés coexistent, l'une privée et l'autre publique. Ce type de chiffrement est tel que, la clé publique est utilisé dans l'étape de chiffrement et la clé privée est utilisée pour le déchiffrement.

2.1 Quelques fonctions de base en cryptographie

Dans cette section, nous définissons les fonctions cryptographiques de base que nous allons utiliser tout au long de cette thèse.

2.1.1 Fonction à sens unique

Une fonction f est à sens unique s'il est facile de calculer $f(x)$ à partir de x, mais il est difficile, étant donné $y = f(x)$ pour un élément x quelconque, de trouver un x tel que $f(x) = g$. Une fonction à sens unique f est dite à trappe, si à l'aide de cette trappe on peut trouver facilement un antécédent pour f.

2.1.2 Fonction de hachage

Une fonction de hachage a pour but de pouvoir donner pour n'importe quel message de taille quelconque, un condensé de taille fixe. Donc, on peut définir une telle fonction de la manière suivante :

Définition 2.1.1. *Soit Σ un alphabet. On définit une fonction de hachage comme une application :*

$$h : \Sigma^* \longrightarrow \Sigma^n, n \in \mathbb{N}$$

Les fonctions de hachage possèdent plusieurs propriétés.

- Une fonction de hachage peut être appliquée à un bloc de données de n'importe quelle taille.
- Une fonction de hachage h produit une sortie de longueur fixe.
- Pour toute donnée x, $h(x)$ doit être facile à calculer.
- Pour tout code donné a, il est impossible (avec les moyens dont on dispose) de trouver x tel que $h(x) = a$ "calcul de préimage".
- Pour tout bloc donné x, il est impossible de trouver y différent de x tel que $h(y) = h(x)$ "calcul de seconde préimage".

- Il est impossible (avec les moyens dont on dispose) de trouver un couple (x, y) tel que $h(x) = h(y)$ "collision".

Une bonne fonction de hachage au sens cryptographique doit vérifier au moins trois propriétés : la résistance aux collisions, la résistance en seconde préimage et la résistance en préimage.

2.2 Problèmes difficiles en théories des codes

Dans cette section, nous définissons quelques problèmes difficile en théorie des codes sur lesquels est basé la construction de nos systèmes de chiffrement.

1. **Problème de décodage par syndrome**
   Entrée : Soit H une matrice $(n \times k; n)$ binaire, w un entier et $s \in \mathbb{F}_2^{n-k}$ un syndrome.
   Sortie : Un mot $e \in \mathbb{F}_2^n$ tel que $wt(e) \leq w$ et $He^T = s$.

2. **Équivalence de Codes Poinçonnés**
   Soit M une matrice de taille $k \times n$ et H une $k \times m$ matrice où $m \leq n$ sur un corps $\mathbb{F}$, exist-il une matrice non-singulière T de taille $k \times k$ et une application injective τ telle que $(TM)_\tau = TM_\tau = H$.

2.3 Cryptosystème de McEliece

Peu de temps après la publication de l'article de Diffie et Hellman [23] fondant la cryptographie à clé publique, McEliece [60] inventa le premier système de chiffrement basé sur la théorie algébrique des codes correcteurs d'erreurs. Dans ces cryptosystèmes, la clé publique est constituée par la matrice génératrice d'un code linéaire ; quoique McEliece a utilisé la famille des codes de Goppa binaires irréductibles.
La sécurité du système repose sur l'indistingabilité entre les codes de Goppa et les codes aléatoires ainsi que le problème du décodage d'un code aléatoire qui est NP-complet.
L'inconvénient de ce système réside sur la grande taille de la clé publique, pour cela des travaux ont été réalisés pour réduire la taille de cette clé en utilisant de différentes façons. Actuellement, plusieurs schémas ont été proposés en modifiant la structure du code de Goppa, par exemple les codes de Reed Muller [47, 87], les codes LDPC [2, 3], les codes convolutionnels [54] ...etc.

Comme tous les systèmes de chiffrement à clef publique, ce cryptosystème comprend trois opérations : la génération d'une paire de clé (une privée et l'autre publique), le chiffrement et le déchiffrement.
La clé privée est le triplet (G, S, P) où G est une matrice génératrice d'un code linéaire choisi aléatoirement qui est capable de corriger t erreurs et qui possède un algorithme de décodage rapide et efficace, S est une $k \times k$ matrice aléatoire inversible et P est une $n \times n$

matrice de permutation inversible.
La clé publique est le couple (G_{pub}, t) où G_{pub} est la matrice définie par :

$$G_{pub} = SGP$$

Un élément important de la cryptographie basée sur les codes est le choix des paramètres du code qui influent sur la sécurité du cryptosystème. Par exemple McEliece a suggéré de choisir un $(1024, 524, 101)$-code de Goppa binaires [60].
Les étapes du schéma de chiffrement de McEliece sont les suivantes :

Algorithme de génération des clés

1. Choisir aléatoirement une matrice génératrice d'un code de Goppa binaire.
2. Choisir aléatoirement une $k \times k$ matrice non singulière S.
3. Choisir aléatoirement une $n \times n$ matrice de permutation inversible P.
4. Calculer $G_{pub} = SGP$.

La clé publique est le couple $Pk = (G_{pub}, t)$ et la clé secrète est le triplet $sk = (G, S, P)$.

Algorithme de chiffrement

1. Générer aléatoirement un vecteur e de longueur n et de poids au plus t.
2. Calculer $c = mG_{pub} + e$

Algorithme de déchiffrement

1. Calculer $u = cP^{-1}$.
2. Déterminer $u^{'}$ en corrigeant les erreurs de u.
3. Calculer $m = u^{'}S^{-1}$.

2.4 Conclusion

Dans le présent chapitre, nous avons décrit ce qui est une fonction à sens unique, qui est un outil nécessaire en cryptographie. Nous avons également défini la fonction de hachage, qui est utilisé dans plusieurs primitives cryptographiques telles que les signature. A la fin du chapitre nous avons présenté le premier système de chiffrement asymétrique basé sur les codes correcteurs d'erreurs.

Chapitre 3

Un nouveau schéma d'identification basé sur les codes

Ce chapitre a fait l'objet d'une publication dans un proceeding IEEE à la conférence inernationale "7^{th} Seminar on Detection Systems : Architectures and Technologies (DAT 2017)" [69].

Dans ce chapitre, nous proposons un nouveau schéma d'identification basé sur les codes, nous prouvons qu'il est à divulgation nulle de connaissance. De plus nous avons réduit la taille des clés par rapport aux variantes existantes.

3.1 Introduction

Un schéma d'identification est une série de messages échangés entre deux entités (un prouveur et un vérifieur) dont le prouveur cherchera à s'identifier auprès du vérifieur. Les schémas d'identification à clé publique sont généralement appliqués en cryptographie pour atteindre l'objectif d'authentification d'entité. Ils ont des applications dans l'authentification et les services de contrôle d'accès tels que la connexion à distance, les achats par carte de crédit ...etc.

Les schémas d'identification à divulgation nulle de connaissance (zero-knowledge identification schemes) ont été inventés par Fiat et Shamir [25] en 1986. Le concept de ce type de protocoles est que le prouveur essayera de convaincre le vérifieur qu'il connaît un secret sans dévoiler son contenu.

L'idée d'utiliser des codes correcteurs pour construire un schéma d'identification est dûe à Harari [42], mais son schéma n'était pas pratique.
A Crypto'93, J. Stern [90] a introduit le premier protocole d'identification efficace à divulgation nulle de connaissance basé sur les codes correcteurs d'erreurs. La sécurité de ce schéma repose sur la difficulté du décodage par syndrome qui est prouvé NP-complet. Le

schéma de Stern est basé sur le cryptosystem de Niederreiter. L'avantage de ce schéma est sa résistance aux attaques menées par un ordinateur quantique (s'il apparaîtra un jour), contrairement aux systèmes basés sur des problèmes difficiles d'arithmétique tels que la factorisation et le logarithme discret, qui sont obsolète face à un tel puissant ordinateur. Un autre avantage de ce schéma d'identification basé sur les codes est qu'il nécessite des opérations simples, donc il est facile à implémenter. Toutefois, ces principaux inconvénients sont le coût élevé de communication et la grande taille de la clé publique, ce qui rend le schéma de Stern impraticable. Il existe une version duale de ce protocole proposée par Pascal Véron (il a utilisé une matrice génératrice d'un code au lieu d'une matrice de contrôle) . Par rapport au schéma de Stern, Véron réussi à diminuer légèrement la complexité de communication mais la taille de la clé publique est plus grande (environ 15 KB) .

Dans ce chapitre, nous proposons une amélioration du schéma d'identification de Stern en réduisant la taille de la clé. Notre schéma est à divulgation nulle de connaissance et il est basé sur deux problèmes difficiles : le problème du décodage par syndrome et le problème de trouver la matrice de contrôle d'un code MDPC à partir de sa matrice génératrice. Le contenu de ce chapitre est organisé comme suit. Dans la section 3.2, nous décrivons le protocole de Stern. Dans les sections 3.3 et 3.4, nous rappelons ce qui est une preuve à divulgation nulle de connaissance et nous définissons le modèle de l'oracle aléatoire. Nous présentons notre schéma proposé dans la section 3.5. La dernière partie de ce chapitre est consacrée à l'étude de la sécurité du protocole ainsi que ses avantages par rapport aux schémas prédécesseurs.

3.2 Schéma d'identification de Stern

Soit n et k deux entiers tels que $n > k$. Le schéma de Stern utilise une matrice de contrôle H d'un code linéaire choisi aléatoirement, de taille $(n-k) \times n$, qui est publique pour tous les utilisateurs.

Chaque utilisateur (prouveur) reçoit une clé secrète qui est un mot de n bits de poids prescrit p. Il calcul un identifiant (pour le publier) comme suit :

$$i = Hs^t$$

Cet identifiant peut être utilisé pour plusieurs authentifications. Le protocole d'identification s'appuie fortement sur la notion de commitment (engagement). Un commitment sera utilisé comme une fonction à sens unique. Maintenant, nous décrivons les étapes du protocole de Stern. La génération des clés se fait de la manière suivante :

i **Paramètres du système :** $n, k, q, w \in \mathbb{N}^+$ et $H_{pub} \in \mathbb{F}_q^{(n-k)\times n}$.

ii **Clé publique :** $H_{pub}e^T = s \in \mathbb{F}_q^{n-k}$

iii **Clé privée :** $e \in \mathbb{F}_q^n$ de poids w.

Le schéma comprend r tours, chacun d'eux s'exécute selon l'algorithme suivant :
Le prouveur P choisit aléatoirement :

- un vecteur y de n bits
- une permutation σ de l'ensemble $\{1,\ldots,n\}$,

et envoie à V trois commitments :

$$c_1 = (\sigma, H_{pub}y^T), c_2 = y\sigma, c_3 = (y+e)\sigma$$

Le vérifieur V envoie à P une valeur aléatoire $b \in \{0,1,2\}$ aléatoirement.
P reçoit b et nous avons donc trois possibilités :

- ⋆ Si $b = 0$: P révèle c_2 et σ
  V vérifie que c_1 et c_2 reçus sont correctement calculés.
- ⋆ Si $b = 1$: P révèle c_3 et σ
  V vérifie que c_1 et c_3 reçus sont correctement calculés.
- ⋆ Si $b = 2$: P révèle c_2 et c_3
  V vérifie c_2, c_3 et le poids de $e\sigma$.

Le détenteur de clé secrète peut prouver sa connaissance de e en utilisant deux facteurs de mélange : une permutation et un vecteur aléatoire. Cependant, pour un tour d'exécution, un tricheur peut s'identifier avec une probabilité de 2/3. Afin de réduire la probabilité de triche, le protocole doit être exécuté r fois jusqu'à l'obtention du niveau de sécurité espéré. Le coût de communication par tour est $n(log_2(q) + log_2(n))$ plus trois fois la taille des commitments utilisés.

3.3 Preuve à divulgation nulle de connaissance

S. Goldwasser et al. [39] ont introduit une méthode probabiliste de preuve, tel que un observateur sera convaincu de la preuve avec une très grande probabilité.
Les auteurs ont produit un modèle qui permet de prouver l'appartenance d'un élément I à un certain langage L de classe NP. Ils ont énoncé une hypothèse qui dit que le prouveur possède une puissance de calcul illimitée et que son temps d'exécution est exponentiel et celui du vérifieur est polynomial.
Soit le langage $L = \{(m,x), x \in \mathbb{Z}_m^\times$ n'est pas un résidu quadratique$\}$ et soit $I = (m,x)$. Il n'existe pas d'algorithme efficace qui vérifie si un nombre est un résidu quadratique modulo m lorsque la factorisation de m est inconnue. L'algorithme suivant permet à un prouveur de démontrer au vérifieur si le couple (m,x) appartient ou non à L, sans dévoiler la factorisation de m.

★ V choisit n nombres quelconques $r_1,\ldots,r_n$ dans $\mathbb{Z}_m^\times$

★ V choisit un vecteur aléatoire $(t_1, \ldots, t_n) \in \{0,1\}^n$

★ V envoie à P le vecteur $t_1, \ldots, t_n$ où $t_i = x^{e_i} r_i^2 \mod m$

★ P possédant une puissance de calcul illimitée, peut déterminer les résidus quadratiques t_i. Il en déduit alors le vecteur $r_1, \ldots, r_n$ de la manière suivante :

$$r_i = \begin{cases} \sqrt{t_i x^{-1}} \mod n & \text{si } t_i \text{ n'est pas un résidu quadratique} \\ \sqrt{t_i} \mod n & \text{sinon} \end{cases}$$

et l'envoie à V qui contrôle la validité de la réponse.

Si $(m, x) \in L$, Alors P peut toujours déterminer la suite $r_1, \ldots, r_n$, sinon $t_1, \ldots, t_n$ est une suite aléatoire de résidus quadratiques et la probabilité pour que P reconstruit la suite $r_1, \ldots, r_n$ vaut $\frac{1}{2^n}$.

A. Fiat et A. Shamir [25] ont démontré l'importance des systèmes interactifs de preuve à divulgation nulle de connaissance, en les appliquant pour la première fois au problème de l'identification. Dans leur modèle, ils ont utilisé les notations suivante :

♣ $\overline{P}$ représente un prouveur honnête.

♣ $\tilde{P}$ représente un fraudeur utilisant un algorithme probabiliste polynomial.

♣ P représente soit $\overline{P}$, soit $\tilde{P}$.

♣ $\overline{V}$ représente un vérifieur honnête.

♣ $\tilde{V}$ représente un vérifieur malhonnête utilisant un algorithme probabiliste polynomial.

♣ X, Y représente l'exécution du processus d'identification où X est le prouveur et Y le vérifieur.

Les conditions prises sont les suivantes :

♣ P et V sont considérés comme un couple interactif de machines de Turing. Ils ont en commun un ruban de données I, deux rubans de communication CP et CV, deux rubans privés de travail WP et WV, et deux rubans privés d'aléa RP et RV.

♣ $\overline{P}$ a accès à un ruban privé possédant son secret $\mathfrak{s}$. Ce dernier satisfait un prédicat publique $P(I, \mathfrak{s})$ vérifiable en temps polynomial.

♣ P et V possèdent des états d'exécution et de repos distincts.

♣ Au début du processus, V est en état de repos et P est en état d'exécution.

♣ Lorsque P atteint un état de repos, V passe à un état d'exécution et vice-versa.

- ♣ V possède deux états spéciaux : un état d'échec et un état de succès, à partir desquels aucun passage d'un état à l'autre n'est possible. Lorsque V atteint l'un de ces états, le protocole s'arrête.
- ♣ Le temps d'exécution de V est polynomialement borné.
- ♣ Si P s'exécute et V est au repos, ce dernier n'effectue aucun passage d'un état à l'autre.
- ♣ P s'exécute en un temps fini.
- ♣ Il existe un polynôme $\mathfrak{p}$ tel que P n'écrive jamais plus de $\mathfrak{p}(n)$ caractères sur le ruban de communication, lorsque la donnée I est de longueur n.

Un protocole de preuve à divulgation nulle de connaissance satisfait les conditions suivantes :

- ♦ Le prouveur ne peut pas tromper le vérifieur.
- ♦ Le vérifieur ne doit pas obtenir le moindre indice.
- ♦ Le vérifieur n'apprend rien du prouveur que la véracité de la proposition.

Un ruban de communication est la concaténation de toutes les valeurs échangées entre P et V, lors du déroulement d'une preuve. Les valeurs échangées ont le comportement de variables aléatoires suivant une certaine loi de distribution.

On note $(P,V)[I]$ l'état dans lequel se trouve V après l'exécution de (P,V). Lorsque le ruban de communication commun de données est I, nous définirons ce qui est un couple interactif de machines de Tuting.

Définition 3.3.1. *Un couple interactif de machines de Tuting polynomiales probabilistes, $\overline{P}$ et $\overline{V}$, est un système interactif de preuve pour le prédicat $P(I,\mathfrak{s})$, si pour $0 \leq \epsilon \leq \frac{1}{2}$, les conditions suivantes sont satisfaites :*
Complétude : *Pour toute donnée I satisfaisant $P(I,\mathfrak{s})$*

$$Pr((\overline{P},\overline{V})[I] = "succès") \geq 1-\epsilon$$

Consistance : *Pour tout $\tilde{P}$, quelle que soit la donnée I*

$$Pr((\tilde{P},\overline{V})[I] = "succès") \leq \epsilon$$

Définition 3.3.2. *Un système interactif de preuve est à divulgation nulle de connaissance, si pour tout vérifieur V, il existe une machine de Turing polynomiale probabiliste, capable de simuler un ruban de communication dont la loi de distribution est parfaitement indiscernable de la loi de distribution d'un ruban de communication provenant d'une véritable identification avec V.*

3.4 Modèle de l'oracle aléatoire

Le modèle de l'oracle aléatoire a été introduit par Bellare et Rogaway [8] en 1993 pour modéliser une fonction de hachage comme un oracle publiquement accessible, retournant une chaine de n bits uniformément aléatoires à chaque nouvelle requête. Dans ce modèle, le challenger et l'adversaire peuvent faire des requêtes à un oracle aléatoire noté O. Ce dernier répond en choisissant une valeur aléatoire, avec pour une seule condition, que si on lui pose deux fois la même requête, alors il doit donner deux fois la même réponse. C'est à dire, le modèle fixe un entier n et un domaine d'entrée Dom pour l'oracle. Au début du jeu de sécurité, l'oracle choisit au hasard une fonction $f : Dom \longrightarrow \{0,1\}^n$ et à toutes les requêtes $x \in Dom$ faites par une partie, il répond par $f(x)$.
Dans une preuve de sécurité, si on fait appel à des oracles aléatoires, on dit que la preuve se fait dans le modèle de l'oracle aléatoire (dans ce cas ces oracles aléatoires correspondent à des fonctions de hachage dans le cryptosystème). Si on n'utilise pas d'oracle aléatoire, on dit qu'on est dans le modèle standard.
Canetti et al. [15] ont montrer dans leurs travaux qu'une preuve dans le modèle de l'oracle aléatoire n'implique pas que le système est sûr dans la vie pratique (vulnérables avec n'importe quelle fonction de hachage). Néanmoins, il est très utilisé dans les preuves de sécurité par la communauté scientifique.

Définition 3.4.1. *Un oracle est un algorithme (machine de Turing ou une fonction f) auquel on peut soumettre une entrée x et recevoir une sortie $f(x)$ et que cet algorithme se comporte comme une boîte noire pour le requérant. C'est à dire que le requérant n'exécute pas lui même f mais a simplement accès à toute $f(x)$ correspondante au x de son choix.*

Généralement on peut supposer que :

- Le requérant ne connaît pas une expression algébrique (ainsi que tous les paramètres utilisés) de la fonction comme dans le cas d'une fonction de déchiffrement, ainsi quand un requérant est capable de déchiffrer le chiffré de son choix sans forcément savoir comment on déchiffre, on dit qu'il a accès à un oracle de déchiffrement.
- Le requérant connaît une expression algébrique de la fonction mais les sorties de cette fonction se comportent de façon aléatoire, on dit que c'est un oracle aléatoire (c'est le cas d'une fonction de hachage idéalisée).

On considère $\mathcal{P} = \{f : \{0,1\}^* \longrightarrow \{\}^\infty\}$. f prend en entrée un mot de longueur finie et produit une suite de longueur infinie.

Définition 3.4.2. *Une fonction $f \in \mathcal{P}$ est dite aléatoire si pour tout $x \in \{0,1\}^*$ chaque bit de la suite infinie $f(x)$ est pris au hasard c'est à dire $p(0) = p(1) = 1/2$.*

Définition 3.4.3. *Un oracle aléatoire est une procédure qui permet pour $x \in \{0,1\}^*$ de produire $f(x)$ où $f \in \mathcal{P}$ est une fonction aléatoire sans révéler d'aucune façon le procédé de calcul.*

3.5 Notre schéma proposé

La construction de notre protocole d'identification est basé sur les codes QC-MDPC. Leur avantage est la réduction de la taille des clés, ils jouent aussi un rôle très important dans la sécurité du schéma, vu la forme de leur matrice génératrice et celle de contôle de parité.

Le protocole que nous avons construit, procède selon les étapes qui suivent :

Soit H une $(n-k)\times n$ matrice de contrôle d'un code QC-MDPC choisie aléatoirement et G la matrice génératrice du code. La génération des clés pour notre schéma est :
Clé publique commune : une $k\times n$ matrice génératrice G et une fonction de hachage h.
Clé secrète du prouveur : m un vecteur binaire de longueur k, e un vecteur binaire de longueur n et H une matrice de contrôle binaire de taille $(n-k)\times n$.
Clé publique du prouveur : $c = mG + e$, $y = Hs^T$ et $t = w(e)$.
P calcul aléatoirement :

- un vecteur binaire u de longueur k,
- une permutation σ de l'ensemble $\{1,\dots,n\}$,

et envoie à V trois commitments :

$$c_1 = h(\sigma(y)), c_2 = h(\sigma((u\oplus m)G)), c_3 = h(\sigma(uG\oplus c))$$

Le vérifieur V envoie à P un entier aléatoire $b\in\{0,1,2\}$.
P reçoit b et donc il y a trois possibilités :

- Si $b=0$: P révèle y et u
  V vérifie que c_1 et c_3 reçu sont correctement calculés.
- Si $b=1$: P révèle $\sigma(u\oplus m)$ et $\sigma(e)$
  V vérifie que c_2 et c_3 reçu sont correctement calculés et $w(e)=t$.
- Si $b=2$: P révèle y et $u\oplus m$
  V vérifie que c_1 et c_2 reçu sont correctement calculés.

3.6 Analyse de la sécurité de notre protocole

Pour qu'un schéma d'identification soit à divulgation nulle de connaissance, il doit vérifier trois propriétés : la complétude, la consistance et une propriété de divulgation nulle de connaissance.

Dans notre preuve de sécurité, nous utilisons l'approche présenté dans [93].
On note un prouveur malhonnête par $\tilde{P}$ et un vérifieur malhonnête par $\tilde{V}$.
Complètude : Il est facile de prouver que si P connait un triplet valide (m, H, e) pour une donnée publique I, Il peut répondre correctement à toutes les questions du vérifieur. Par conséquent, nous avons :

$$Pr[(\overline{P}, \overline{V})[I] = success] = 1$$

D'où, la propriété de complétude de notre schéma est satisfaite.

Consistance : Nous prouvons maintenant le lemme qui suit.

Lemme 3.6.1. *Si V accepte la preuve de $\tilde{P}$ avec une probabilité au moins $\left(\frac{2}{3}\right)^r + \epsilon$, alors il existe un algorithme probabiliste M de temps polynomial qui extrait, avec une forte probabilité (proche de 1), soit un triplet secret valide (m, H, e) ou trouve des collisions pour la fonction de hachage.*

Démonstration. Soit $\mathcal{T}$ l'arbre d'exécution du protocole $(\tilde{P}, V)$ correspondant à toutes les questions possibles du vérifieur V lorsque $\tilde{P}$ possède un ruban aléatoire RA. V peut poser trois questions possibles à chaque étape. Tout d'abord, nous allons montrer qu'à partir d'un sommet possédant trois fils, on peut exhiber soit un triplet valide (m, H, e), soit une collision pour la fonction de hachage. Ensuite, nous montrons qu'une machine de Turing polynômiale probabiliste M, peut trouver de tels sommets dans $\mathcal{T}$ avec une probabilité proche de un.

Soit v un sommet possédant trois fils. Ceci correspond à une situation où trois commitments c_1, c_2 et c_3 ont été calculés où le tricheur a été en mesure de répondre aux trois quetions du vérifieur.
Soit $y^{'}$ et $u^{'}$ les réponses à la question $b = 0$, $z^{'}$ et $e^{'}$ les réponses à la question $b = 1$ et $y^{''}$, $u^{'} \oplus m^{'}$ les réponses à la question $b = 2$. Nous avons :

$$h(\pi(y^{'})) = c_1 = h(\pi(y^{''}))$$

$$h(\pi(z^{'})G) = c_2 = h(\pi((u^{'} \oplus m^{'})G))$$

$$w(e^{'}) = t$$

$$h(\pi(u^{'}G \oplus c)) = c_3 = h(\pi^{'}(u^{'}G \oplus c))$$

Par conséquent, soit $\tilde{P}$ qui a été en mesure de violer la propriété de liaison du commitment, ou nous avons $c = (u^{'} \oplus m^{'})G \oplus \pi^{-1}(e^{'})$, où $\pi^{-1}(e^{'})$ est un mot de longueur n et de poids t. Alors $(u^{'} \oplus m^{'}, \pi^{-1}(e^{'}))$ est un couple secret valide et peut être utilisé afin de se faire passer pour l'identité de $\tilde{P}$.
Or l'hypothèse du lemme implique que la probabilité pour que $\mathcal{T}$ ait un sommet avec trois fils est au moins ϵ. En effet, considérons RA comme un ensemble de l éléments, où A choisit aléatoirement les valeurs qu'il transmet, et soit l'ensemble $Q = \{0, 1, 2\}$. Ces deux ensembles sont considérés comme des espaces de probabilité munis de la distribution

uniforme.
Un couple $(a,b) \in (RA \times Q)^r$ représente les mises en gage et les questions et réponses communiquées entre $\tilde{P}$ et V, lors d'un processus d'identification. Nous appelons (a,b) une paire valide, si l'exécution de $(\tilde{P}, V)$ conduit à l'état de réussite (succès).
Soit $V \subset (RA \times Q)^r$ contenant toutes les paires valides. D'après l'hypothèse du lemme, nous avons :

$$\frac{|V|}{|(RA \times Q)^r|} \geq \left(\frac{2}{3}\right)^r + \epsilon$$

On considère $\Omega_r \subset RA^r$ tel que :

- Si $a \in \Omega_r$, alors $2^r + 1 \leq |\{b, (a,b) \text{ soit valide }\}| \leq 3^r$

- Si $a \in RA^r \setminus \Omega_r$, alors $0 \leq |\{b, (a,b) \text{ soit valide }\}| \leq 2^r$

Alors $V = \{(a,b) \text{ valides }, a \in \Omega_r\} \cup \{(a,b) \text{ valides }, a \in RA^r \setminus \Omega_r\}$, par conséquent :

$$|V| \leq |\Omega_r|.3^r + (\mu^r - |\Omega_r|).2^r$$

Notons que $|RA^r| = \mu^r$ et $|Q^r| = 3^r$. Donc

$$\begin{array}{rcl} \frac{|V|}{|(RA\times Q)^r|} & \leq & \left(\frac{|\Omega_r|}{|RA^r|} + 2^r\left(3^{-r} - \frac{|\Omega_r|}{|(RA\times Q)^r|}\right)\right) \\ & \leq & \frac{|\Omega_r|}{RA^r} + \left(\frac{2}{3}\right)^r \end{array}$$

Il s'en suit que : $\frac{|\Omega_r|}{RA^r} \geq \epsilon$. Cela montre que la probabilité qu'un intrus (fraudeur) puisse répondre à au moins 2^r+1 des questions du vérifieur, par le choix de valeurs aléatoires, est plus grande que ϵ. Maintenant, si plus de $2^r + 1$ questions sont répondues correctement par $\tilde{P}$, cela signifie que $T(RA)$ a au moins $2^r + 1$ feuilles, i.e. $T(RA)$ possède au moins un sommet ayant trois fils. Or, si aucun sommet de $T(RA)$ ne possède trois fils, donc ce dernier ne peut avoir au plus que 2^r feuilles. Alors,

$$Pr(T(RA) \text{ possède un sommet ayant trois fils}) \geq \epsilon$$

Par conséquent, en recombinant $\tilde{P}$, $\frac{1}{\epsilon}$ fois, il est possible de trouver un arbre d'exécution avec un sommet ayant trois fils avec une probabilité arbitraire constante proche de un. □

Ce lemme implique que la fonction de hachage h n'est pas résistante aux collisions, ce qui contredit l'hypothèse de non solvabilité en temps polynomial, du problème SD. Pour tout I, on en déduit que :

$$Pr((\tilde{P}, V)[I] = \text{"succès"}) \leq \left(\frac{2}{3}\right)^r$$

Ceci conclut la démonstration de la propriété de consistance.

Zero-Knowledge : La propriété à divulgation nulle de connaissance pour notre protocole d'identification est prouvée dans le modèle de l'oracle aléatoire.

Proposition 3.6.1. *Notre protocole d'identification est à divulgation nulle de connaissance (zero-knowledge) dans le modèle de l'oracle aléatoire.*

Démonstration. Nous allons prouver qu'aucune information secrète ne peut être déduite en temps polynomial pendant l'exécution du protocole.
Afin de produire le comportement d'un vérifieur malhonnête, il faut supposer que ce dernier adoptera une certaine stratégie en fonction des mises en gage que le prouveur lui dévoile. Même si en pratique V ne peut rien déduire de ces valeurs, il est nécessaire dans le cadre théorique, de ne rien laisser au hasard, pour bien prouver la sécurité du schéma. On note $St(c_1, c_2, c_3) \in \{0, 1, 2\}$ la stratégie d'un vérifieur quelconque. Notons

$$\begin{array}{rccc} \xi: & \mathbb{F}_2^k & \longrightarrow & \mathbb{F}_2^k \\ & u & \longrightarrow & u+m \end{array}$$

$$\begin{array}{rccc} \Sigma: & \mathbb{F}_2^k & \longrightarrow & \mathbb{F}_2^n \\ & u & \longrightarrow & uG \end{array}$$

ξ est un automorphisme de $\mathbb{F}_2^k$ et Σ est un isomorphisme de $\mathbb{F}_2^k$ sur le code C engendré par la matrice génératrice G.

L'algorithme probabiliste polynomial M suivant, produit un ruban de communication dont la distribution de probabilité est indistinguable de celle d'un ruban de communication provenant d'une identification réelle.

1. M Choisit aléatoirement une question, $b \in \{0, 1, 2\}$

   Le cas $b = 0$ peut être anticipé par le choix de π, une permutation aléatoire, un vecteur aléatoire y', un mot aléatoire u' et le calcul de $h_1 = h(\pi(y'))$ et $h_3 = h(\pi(u'G + c))$. Nous remarquons que (y', u') et (y, u) sont indistinguables.

   Si $b = 1$, M choisit $x \in C$ un mot de code aléatoire, $e' \in W$ où $W = \{\delta \in \mathbb{F}_2^n; w_H(\delta) = t\}$ et calcule $h_2 = \pi(x)$ et $h_3 = \pi(x + e')$ et substitut h_l par une chaine aléatoire. Soit $L = (h_1|h_2|h_3)$ la concaténation de h_1, h_2 et h_3, et $A = (\pi(y)|\pi(e').\pi(e'))$ a la même distribution de probabilité que $\sigma(e)$, de plus soit z n'importe quel mot de code de C, alors :

   $$\begin{array}{rcl} Pr[(u \oplus m)G = z] & = & Pr[(u = \xi^{-1}(\Sigma^{-1}(z))] \\ & = & 2^{k-n} \\ & = & Pr[x = z] \end{array}$$

   Le cas $b = 2$ peut être anticipé par le choix de π une permutation aléatoire, $v = u'' \oplus m'$, un vecteur aléatoire y'', $h_1 = h(\pi(y''))$ et $h_2 = h(\pi(vG))$. Nous remarquons que (y'', v) et $(y, u \oplus m)$ sont indistinguables.

2. M calcule $b^{'} = St(L)$.

3. Si $b = b^{'}$, alors M écrit sur le ruban $\mathfrak{R}$, les quantités L, b et A, sinon M retourne à l'étape 1.

Par conséquent, en $3r$ tours en moyenne, M simule un ruban de communication indiscernable d'un autre ruban qui correspond à une exécution de processus d'identification équitable qui prend r tours. Ceci conclut la preuve. □

Notre protocole est basé sur la difficulté de trouver la matrice de contrôle de parité à partir de la matrice publique et du problème SD.

Nous mentionnons qu'attaquer la variante MDPC de McEliece ou Niederreiter n'est pas plus facile que de résoudre le problème de décodage par syndrome pour un code aléatoire et de casser le système de McEliece basé sur les codes QC-MDPC n'est pas plus facile que de résoudre le problème du décodage par syndrome des codes aléatoire quasi-cycliques.
Donc, l'utilisation des codes QC-MDPC dans notre système d'identification est sécurisé.
Afin de démontrer l'efficacité de notre schéma, nous l'avons comparé à d'autres protocoles.

Paramètres proposés :

Pour un niveau de sécurité de 80, nous suggérons comme paramètres : $n_0 = 2$, $n = 174$, , $k = 87$ et $w = 11$.
Une comparaison des tailles des clés pour ces paramètres avec d'autres protocoles est donnée dans le tableau 3.1 :

	Stern [90]	Veron [93]	CVE [17]	Notre schéma
Tours	28	28	16	16
Taille de la matrice	122500	122500	32768	7569
Clé publique	350	700	512	174
Clé privée	700	1050	1024	348

TABLE 3.1 – Comparaison de la taille des clés (bits)

Cela montre que le système proposé a la plus petite taille de clé.

3.7 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle variante du protocole d'identification de Stern. Notre schéma est basé sur la difficulté de trouver la matrice de contrôle à partir de la matrice génératrice publique et sur la difficulté du décodage d'un code linéaire aléatoire. En outre, notre système a une clé réduite par rapport à d'autres protocoles.

Chapitre 4

Nouvelle variante du cryptosystème de McEliece

Ce chapitre a fait l'objet de deux publications. La première a été présentée à la conférence internationale "20^{th} Conference on Applications of Computer Algebra", New York, 2014 [66]. La seconde a été publiée comme chapitre d'un livre dans un proceeding Springer dédié à la conférence inernationale "4th International Castel Meeting on Coding Theory and Applications", Portugal [67].

Dans ce chapitre, nous introduisons une nouvelle variante du cryptosystème de McEliece qui est basée sur les codes convolutionnels et les générateurs auto-rétrécissants, nous prouvons que notre système est sûr contre certaines attaques robustes.

4.1 Construction d'un code poinçonné

Le poinçonnage permet de générer de nouveaux codes convolutifs à partir d'anciens. Cette technique consiste à ne pas transmettre certains bits suivant un motif de poinçonnage approprié.

Pour obtenir la matrice génératrice du code poinçonné, il faut tout d'abord construire la matrice du code regroupé de profondeur M. Pour cela, nous devons réaliser une décomposition polyphase des polynômes générateurs de la matrice génératrice du code origine.

Définition 4.1.1. *Soit $a(D) \in \mathbb{F}_2$ un polynôme d'indéterminée en D.*

$$a(D) = a_0 + a_1 D + a_2 D^2 + \cdots + a_\mu D^\mu$$

Alors le polynôme $a(D^r)$ est tel que :

$$a(D^r) = a_0 D^{r.0} + a_1 D^{r.1} + a_2 D^{r.2} + \cdots + a_\mu D^{r.\mu}$$

Pour tout $j = 0,\dots,M-1$, La (j,M)-ème décomposition polyphase de $a(D)$ est donnée par :

$$q_j(D) = D^{-j/M}.a_{[j]M}(D^{1/M}),$$

où $[j]M$ la classe des entiers j modulo M, qui correspond aux entiers $j+l.M$ avec $l = 0,1,\dots,\lfloor\frac{\mu-j}{M}\rfloor$. $a_{[j]M}(D^{1/M}$ est le polynôme composé des coefficients $[j]M$ de $a(D)$. $q_j(D)$ est appelée la j-ème composante polyphase de $a(D)$.

Cette décomposition sert à construire une matrice dite PCPC.
La M-ième matrice polycyclique pseudo-circulante (PCPC) $Q^{[M]}(D)$ associée à $a(D)$ qui est de taille $M \times M$, est définie par :

$$Q^{[M]}(D) = \begin{pmatrix} q_0(D) & q_1(D) & \dots & q_{M-2}(D) & q_{M-1}(D) \\ Dq_{M-1}(D) & q_0(D) & \dots & q_{M-3}(D) & q_{M-2}(D) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ Dq_2(D) & Dq_3(D) & \dots & q_0(D) & q_1(D) \\ Dq_1(D) & Dq_2(D) & \dots & Dq_{M-1}(D) & q_0(D) \end{pmatrix}$$

Définition 4.1.2. *On note $g_{i,j}(D)$ un polynôme générateur de la matrice génératrice $G(D)$, alors la matrice génératrice du code $C^{[M]}$, notée $G^{[M]}(D)$, peut être obtenue en remplaçant les polynômes $g_{i,j}(D)$ par leurs M-ième matrices PCPC et en entrelaçant les lignes et les colonnes à la profondeur M. Donc, en partant d'un code convolutionnel de paramètres (n,k,K), on a construit un code équivalent de paramètres (M_n, M_k, K_p) appelé code regroupé.*

Avec la matrice de poinçonnage P (motif de poinçonnage) dont le nombre de ses coefficients correspond au nombre de colonnes de la matrice génératrice du code regroupé, nous allons obtenir la matrice génératrice du code poinçonné. Soit ϕ la fonction définie par :

$$(i,j) \longrightarrow \phi(i,j) = i + n(j-1)$$

En associant chaque coefficient $P_{i,j}$ à la colonne $\phi(i,j)$ de $G^{[M]}(D)$ (c'est la colonne qu'on va supprimer), nous obtiendrons enfin la matrice génératrice du code poinçonné de paramètres $(N^{'}, M_k, K_p)$. On doit s'assurer que la matrice obtenue n'est pas catastrophique pour ne pas avoir des problèmes lors du décodage.

4.2 Générateur rétrécissant

Dans [20] Coppersmith, Krawczyk et Mansour ont introduit le principe de ce nouveau générateur pseudo aléatoire. Le principe consiste à construire la sortie du générateur en utilisant deux LSFRs générant les m séquences $(x_0, x_1 \dots)$ et $(y_0, y_1 \dots)$ en fonction de la règle suivante :

- si $y_i = 1$, le bit x_i est envoyé en sortie.
- si $y_i = 0$, le bit x_i est supprimé et aucun bit n'est envoyé en sortie.

En1994 Meier et Statfelbach [61] ont présenté une version modifiée qui est plus simple du générateur rétrécissant, appelée générateur auto-rétrécissant, de plus ce système est plus sûr que son prédécesseur. Depuis l'invention de ces deux méthodes de génération, beaucoup de chercheurs ont analysé la sécurité de ce type de générateurs pseudo-aléatoires.
Par exemple, Zenner et al. [100] ont réduit la complexité d'attaque à $\mathcal{O}(2^{0,695n})$, ensuite Krause [48] a créer une attaque BDD de complexité $\mathcal{O}(2^{0,656n})$, qui a été prouvée après dans l'article de Hell et Johanson [44]. En 2006, Zhang et Feng [101] ont suggérer une attaque dont on peut trouver l'état initial du LFSR avec une complexité de $\mathcal{O}(2^{0,556n})$.
Pour éviter ces attaques de complexité qui ne cesse pas à diminuer, nous utiliserons dans ce papier une autre méthode de génération introduite en 2011 dans [49] qui assure une sécurité plus performante.
Kanso [49] a modifié la méthode de Meier et Statfelbach [61] en ulilisant un seul LFSR de longueur n qui fonctionne comme suit : Soit A un LFSR qui génère la séquence $a_t = a_0, a_1, a_2 \ldots$
Au temps i , on considère le triplet $(a_{3i}, a_{3i+1}, a_{3i+2})$, si le bit $a_{3i} \oplus a_{3i+1} = 1$, la sortie est a_{3i+2}, sinon aucune sortie n'est produite.
Nous avons choisi ce procédé pour modifier la structure de notre code en lui donnant une structure aléatoire en remplissant la matrice de poinçonnage, ligne par ligne, par les n éléments en sortie du générateur. Donc la longueur de la séquence de sortie du LFSR doit être assez grande pour générer au moins n bits en sortie.

Exemple 4.2.1. *Soit $G(D)$ une matrice génératrice d'un $(3,2,3)$- code convolutionnel.*

$$G(D) = \begin{pmatrix} D^3+D+1 & D^2+1 & D^2+D+1 \\ D^2+D & D & D+1 \end{pmatrix}$$

En regroupant les matrices PCPC des polynômes générateurs, nous obtenons la matrice :

Polynômes générateurs	Décomposition polyphase d'ordre 2	La matrice PCPC
D^3+D+1	$\begin{pmatrix} 1 & 1+D \end{pmatrix}$	$\begin{pmatrix} 1 & 1+D \\ D+D^2 & 1 \end{pmatrix}$
D^2+1	$\begin{pmatrix} 1+D & 0 \end{pmatrix}$	$\begin{pmatrix} 1+D & 0 \\ 0 & 1+D \end{pmatrix}$
D^2+D+1	$\begin{pmatrix} 1+D & 1 \end{pmatrix}$	$\begin{pmatrix} 1+D & 1 \\ D & 1+D \end{pmatrix}$
D^2+D	$\begin{pmatrix} D & 1 \end{pmatrix}$	$\begin{pmatrix} D & 1 \\ D & D \end{pmatrix}$
D	$\begin{pmatrix} 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ D & 0 \end{pmatrix}$
$D+1$	$\begin{pmatrix} 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ D & 1 \end{pmatrix}$

TABLE 4.1 – Les matrices PCPC correspondantes aux polynômes générateurs

$$G^{'}(D) = \begin{pmatrix} \begin{pmatrix} 1 & 1+D \\ D+D^2 & 1 \end{pmatrix} & \begin{pmatrix} 1+D & 0 \\ 0 & 1+D \end{pmatrix} & \begin{pmatrix} 1+D & 1 \\ D & 1+D \end{pmatrix} \\ \begin{pmatrix} D & 1 \\ D & D \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ D & 0 \end{pmatrix} & \begin{pmatrix} 1 & 1 \\ D & 1 \end{pmatrix} \end{pmatrix}$$

Pour trouver la matrice du code $C^{[2]}$, nous devons effectuer un entrelacement de profondeur 2 sur les ligne de la matrice $G^{'}(D)$, puis sur ses colonnes.

$$G^{[2]}(D) = \begin{pmatrix} 1 & 1+D & 1+D & 1+D & 0 & 1 \\ D & 0 & 1 & 1 & 1 & 1 \\ D+D^2 & 0 & D & 1 & 1+D & 1+D \\ D & D & D & D & 0 & 1 \end{pmatrix}$$

A ce stade, nous avons besoin du motif de poinçonnage. Pour cela, nous utilisons une suite aléatoire de bits d'un LFSR.
Soit le LFSR de longueur 35 :
10100110100001110001100000000110100
Par la méthode de Kanso, nous obtenons comme sortie : 111011
Donc, le motif de poinçonnage est : $T = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$
La position des colonnes de la matrice $G^{[2]}(D)$ qu'il faut enlever est : $\phi(2,2) = 5$.
Alors, notre matrice poinçonnée est donnée par :

$$G_p(D) = \begin{pmatrix} 1 & 1+D & 1+D & 1+D & 1 \\ D & 0 & 1 & 1 & 1 \\ D+D^2 & 0 & D & 1 & 1+D \\ D & D & D & D & 1 \end{pmatrix}$$

4.3 Description du nouveau système de chiffrement

Algorithme 1 : Génération des clefs

1. Choisir une matrice génératrice G d'un code convolutionnel de paramètres : sa longueur n, sa dimension k et sa mémoire m.

2. Ecrire la décomposition polyphase des éléments de G puis former la matrice polycyclique pseudocirculante $Q^{[M]}(D)$.

3. Remplacer les polynômes $g_{i,j}(D)$ de G par leurs M-ième matrices PCPC et en entrelaçant les lignes et les colonnes à la profondeur M afin d'obtenir une matrice $G^{[M]}(D)$.

4. Faire entrer une séquence aléatoire de bits dans un générateur auto-rétrécissant, et remplir la matrice P par les éléments (bits) de sortie.

5. Appliquer la fonction ϕ à la matrice $G^{[M]}(D)$ pour obtenir la matrice secrète G_p .

6. Choisir aléatoirement deux matrices : U matrice de permutation et S une matrice inversible.

7. Calculer la matrice publique $G^{'} = SG_pU$.

Algorithme 2 : Chiffrement
Pour envoyer un message x à quelqu'un, on calcul : $c = xG^{'} + e$
où e est un vecteur d'erreur de poids inférieur où égal à t, choisi de façon aléatoire.

Algorithme 3 : Déchiffrement
Pour déchiffrer le message, il faut :

1. Calculer $z = cU^{-1}$

2. Déterminer $z^{'}$ en corrigeant les erreurs de z

3. Calculer $x = z^{'}S^{-1}$

Pour corriger les erreurs de z, on utilise l'algorithme de décodage de Viterbi, puisque un algorithme qui décode le code parent, il décode également le code poinçonné.

4.4 Algorithme de Viterbi

C'est un algorithme de maximum de vraisemblance qui consiste à trouver le plus court chemin dans le treillis du code qui correspond au code le plus proche du message reçu.
Cet algorithme se déroule comme suit :

4.4.1 Etapes de l'algorithme

1. D'abord, Nous examinons chaque paquet de n bits du mot (message) reçu puis nous calculons la distance de Hamming entre le mot de la branche (arc) et le mot de code reçu.

2. Ensuite, Nous remplaçons chaque mot de la branche par la distance correspondante calculée à l'étape précédente.

3. Cette distance ajoutée à celle associée au nœud d'origine (état présent) est portée sur le nœud ainsi atteint (état suivant). Un nœud comportera deux indications puisqu'il y a deux façons de l'atteindre.

4. Dans cette étape, nous gardons pour chaque nœud la distance minimale, ce qui revient à ne conserver qu'un certain chemins (que nous appellerons les survivants)

5. Après, Nous recommençons les opérations précédentes (de 1 à 4) jusqu'à ce que le message reçu soit terminer.

6. Enfin, pour trouver le code le plus proche, nous traçons le chemin partant du nœud correspondant à la distance minimale totale allant vers le nœud d'origine.
   Cette trajectoire nous donnera le mot de code le plus proche(probable) du mot de code d'origine.

Exemple 4.4.1. *Soit* $x = (01011101001011)$ *le mot de code d'origine et* $y = (01010101001011)$ *le mot reçu. En appliquant les étapes énoncées ci-dessus, nous obtenons à la fin le plus court chemin qui correspond au mot d'origine (représenté en rouge). Les étapes de l'algorithme de Viterbi de 1 à 5 sont représentées par les figures ci-dessous.*

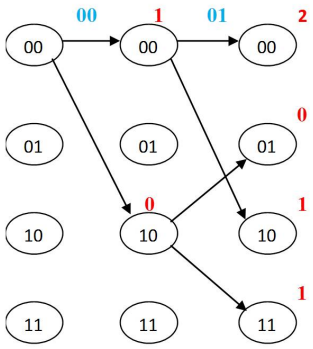

FIGURE 4.1 – Phase initiale

Durant cette phase il n'y a donc pour un état suivant qu'une seule distance et aucun chemin ne sera rejeté.

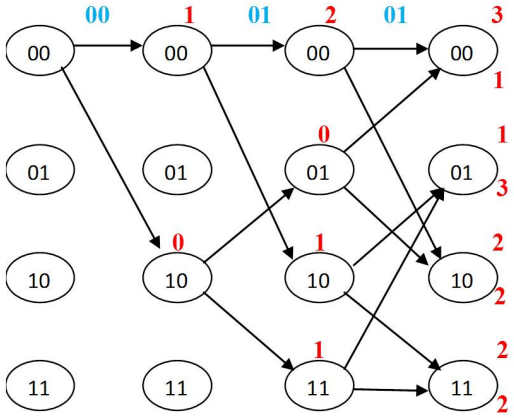

FIGURE 4.2 – Phase 2

Tous les états sont atteignables de deux manières et à chaque étape nous décidons de ne conserver que les trajets correspondants pour chaque état possible à une distance minimale : les survivants.

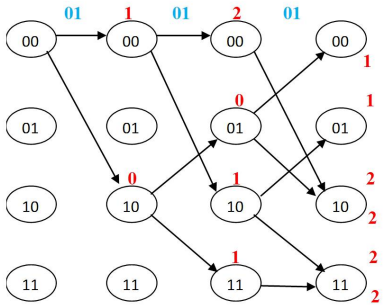

FIGURE 4.3 – Conservation des survivants de la phase 2

Etape suivante :

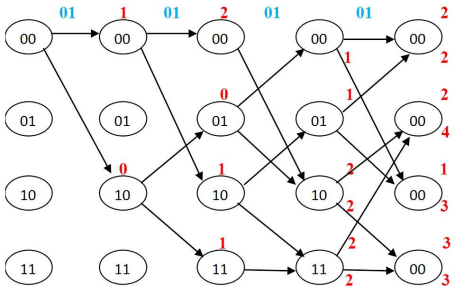

FIGURE 4.4 – Phase 3

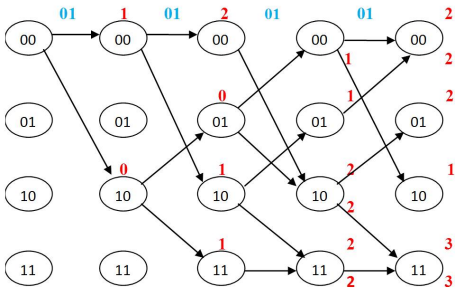

FIGURE 4.5 – Conservation des survivants de la phase 3

Nous continueons notre représentation, jusqu'à ce que le message reçu soit terminer. Nous obtenons à la fin :

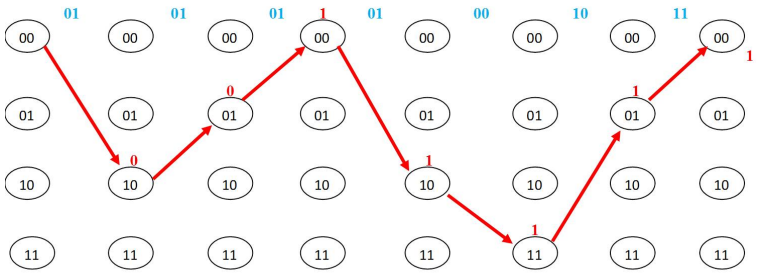

FIGURE 4.6 – Phase finale

et là on constate qu'il y a une seule erreur comise sur le 5 éme bit.

4.5 Etude de la sécurité

Il existe plusieurs approches pour tenter d'attaquer le système de chiffrement de McEliece. Parmi eux, on trouve des méthodes algébriques et des méthodes probabilistes. Dans cette section nous donnons une preuve que notre système est sécurisé contre les attaques structurelles et les attaques par décodage.

4.5.1 Attaques structurelles

L'objectif d'une attaque structurelle est de trouver un code équivalent au code public dont un algorithme de décodage polynomial est connu. Pour cela, nous avons utilisé un motif de poinçonnage pour cacher la structure du code, dont l'attaquant ne peut pas imaginer. De plus, l'équivalence des codes poinçonnés est un problème NP-complet [98]. Cela rend impossible la cryptanalyse du système.

Les cryptosystèmes à clé publique sont difficiles à concevoir. La fonction de cryptage doit être une fonction à sens unique. Dans notre cas, l'inversion de la fonction à sens unique est liée au problème du décodage d'un code convolutif arbitraire.

Définition 4.5.1. *Entrée : une matrice aléatoire H de taille $(n-k)\times n$, un vecteur S de $(n-k)$ et un poids w.*
Sortie : un vecteur e de n bits de poids de Hamming $\leq w$ tel que $H.e^T = S$.

Proposition 4.5.1. *Le décodage par syndrome d'un code convolutionnel est NP-complet.*

Démonstration. La preuve est basée sur le fait que nous pouvons réduire notre problème à Appariement à 3 dimensions (three dimensional matching) [38]. Le résultat peut donc être prouvé de la même manière que [26, Section II]. □

Landais et Tillich ont introduit un algorithme qui est utilisé pour trouver une matrice génératrice équivalente à la matrice secrète. En appliquant cette méthode à notre matrice publique, nous obtenons une matrice sous la forme donnée à la figure 4.7.

FIGURE 4.7 – La matrice génératrice obtenue après l'attaque de G. Landais et J-P. Tillich

Ce qui n'est pas équivalent à la matrice G_p, car si on poinçonne le code C de distance minimale d dans les positions non nulles, on obtient un code de distance $d^{'} < d$.

4.5.2 Attaques exhaustives

Dans notre schéma, la clé privée (G_p, S, P, T) est obtenue au hasard. Puisque le nombre de matrices inversibles dans $\mathbb{F}_q$ est $\prod_{i=1}^{k}(q^k - q^{i-1})$ et le nombre de matrices de permutation de taille n est égal à $n!$. Ensuite, pour trouver les deux matrices sélectionnées, nous devons essayer $n!\prod_{i=1}^{k}(q^k - q^{i-1})$ matrices, donc la probabilité de trouver la bonne matrice est négligeable.
Pour trouver les positions de poinçonnage, nous devons rechercher la séquence initiale générée par le LFSR. Soit L l'ensemble des positions des colonnes supprimées lors du poinçonnage.
Pour trouver un sous-ensemble $L^{'} \subseteq L$, il est nécessaire de connaître une partie de la séquence générée par le LFSR.
Soit A une séquence de r bits générée par un LFSR de longueur s.
Pour construire r triplets de bits i.e .; $s = 3r$, nous obtenons un total de $2^{0.862s}$ états possibles pour le LFSR. Donc, pour un grand r, nous avons un nombre important d'états possibles LFSR.

4.5.3 Décodage par ensemble d'information

Certaines attaques de ce type ont été présentées par [7] et [58].
Le facteur de travail global requis par l'algorithme Canteaut-Chabaud [16] est :

$$W_{p,\sigma} = \frac{\Omega_{p,\sigma} E(N)}{A_w}$$

où A_w est le nombre de mots du code de poids w, $E(N)$ est l'espérance du nombre d'itérations N nécessaire jusqu'à ce que $\{X_i\}_{i\in N}$ atteint l'état de réussite $(2p)_S$ et

$$\Omega_{p,\sigma} = 2p\sigma\begin{pmatrix} K/2 \\ p \end{pmatrix} + 2p(n-k-\sigma)\frac{\begin{pmatrix} K/2 \\ p \end{pmatrix}^2}{2^\sigma} + K\left(p\begin{pmatrix} K/2 \\ p \end{pmatrix} + 2^\sigma\right) + \frac{k(n-k)}{2}$$

Pour un niveau de sécurité de l'ordre de 2^{80} mesuré par l'algorithme de Canteaut-Chabaud, nous proposons l'ensemble des paramètres suivants :
Soit C un $(400, 343)$-code convolutionnel, après une poinçonnage de profondeur 7 dans 56 positions, on obtient un code poinçonné de longueur 2744 et de dimension 2401.
Pour un code de taux $R = 3/4$, nous proposons une poinçonnage de profondeur 3 d'un $(570, 421)$-code convolutionnel en 26 positions. Par la suite, nous aurons un $(1684, 1263)$-code poinçonné.

Niveau de sécurité	n	k	M	Nbre de colonnes supprimées	k'	N'	Taux
80	305	150	4	20	600	1200	1/2
	284	71	5	30	355	1420	1/4
	570	421	3	26	1263	1684	3/4
	125	1050	2	100	250	2000	1/8
100	316	154	5	40	770	1540	1/2
	625	155	3	15	465	1860	1/4
	730	540	3	30	1620	2160	3/4
	68	550	5	30	340	2720	1/8

TABLE 4.2 – Les paramètres proposés pour notre cryptosystème

Dans le système originel de McEliece, pour un $(1024, 524)$-code de Goppa, le niveau de sécurité est de 65 et est de 107 pour un code de Goppa de longueur 2048 et de dimension 1024. Pour les mêmes paramètres, nous obtenons respectivement les niveaux de sécurité 69 et 125, 6.
Maintenant nous allons étudier et comparer le comportement de notre système contre une autre attaque dangereuse ; à savoir l'attaque par renvoi de message. Cette attaque a été introduite dans [14] et est décrite dans la section suivante.

4.5.4 Attaque par renvoi de message

Supposons maintenant que, suite à un accident ou une action de la part d'un cryptanalyste, les deux messages chiffrés $c_1 = mSGP+e_1$ et $c_2 = mSGP+e_2$ avec $e_1 \neq e_2$ sont envoyés. Nous appelons cette procédure, la condition du renvoi de mesage. Pour $(1684, 1263)$-code poinçonné de distance libre $d_{free} = 131$, la probabilité telle que deux vecteur d'erreurs ont 1 à la position l est :

$$Pr(e_1(l) = e_2(l)) = \left(\frac{65}{1684}\right)^2 \approx 0.0015$$

La probabilité que exactement i coordonnées sont simultanément brouillées par e_1 et e_2 est :

$$p_i = Pr(|\{l : e_1(l) = 1\} \cap \{l : e_2(l) = 1\}|) = \frac{\begin{pmatrix} 65 \\ i \end{pmatrix}\begin{pmatrix} 1619 \\ 65-i \end{pmatrix}}{\begin{pmatrix} 1684 \\ 65 \end{pmatrix}}$$

Par conséquent, la cardinalité de L_1 est :

$$E(|L_1|) = \sum_{i=0}^{65}(130 - 2i)p_i \approx 125$$

Par exemple, avec une cardinalité $|L_1| = 124$, nous aurons $|L_0| = 1560$ et seulement 5 entrées de L_0 sont affecter par une erreur. Nous voyons que la probabilité de deviner 1263 colonnes non indexées de celles indexées par L_0 est :

$$\frac{\begin{pmatrix} 1555 \\ 1263 \end{pmatrix}}{\begin{pmatrix} 1560 \\ 1263 \end{pmatrix}} \approx 0.0004336$$

Par conséquent, après environ 2306 essais, un mot de code non modifié est sélectionné, qui peut être décodé à l'aide de la matrice génératrice publique.
Dans la figure 4.8, nous présentons la probabilité de deviner les k colonnes non indexées de celles indexées par L_0 en fonction du nombre d'erreurs pour notre système et celui de McEliece.
Ces résultats sont plus sûrs que le résultat obtenu par la même attaque citée dans [60]. De plus, lorsque le nombre d'erreurs augmente, la sécurité du système de chiffrement sera élevée.

4.6 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle construction du schéma de chiffrement asymétrique de McEliece qui est basée sur les codes convolutionnels et les générateurs auto-rétrécissants. Nous avons prouvé que notre système est sûr contre certaines attaques critiques.

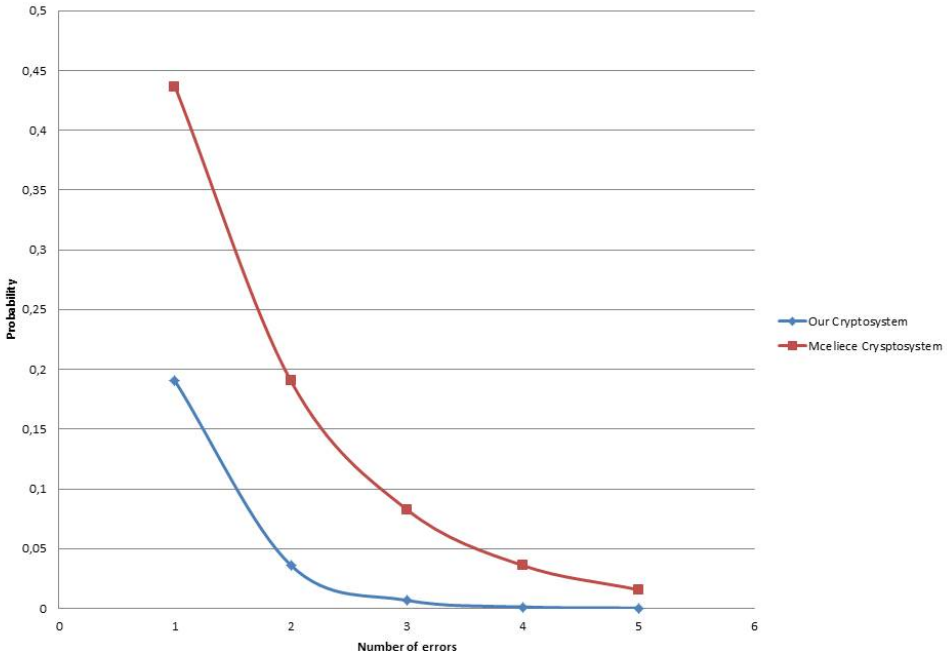

FIGURE 4.8 – la probabilité de deviner les k colonnes non indexées de celles indexées par L_0 en fonction du nombre d'erreurs

Chapitre 5

Une nouvelle variante du cryptosystème de McEliece basée sur les codes QC-LDPC et QC-MDPC

Ce chapitre a fait l'objet d'une publication dans le journal international "IEEE Communications Letters" [71].

Dans ce chapitre nous présentons un nouveau système de chiffrement, inspiré du cryptosystème inventé par McEliece. Notre schéma se base sur les codes QC-LDPC et QC-MDPC. L'avantage de notre système est la résistance aux attaques basées sur les matrices creuses. De plus, la taille de clés est réduite.

5.0.1 Le schéma de chiffrement proposé

Le cryptosystème proposé est similaire au cryptosystème de McEliece. Il comporte donc trois étapes : La génération d'une paire de clés (une clé privée et une clé publique), le chiffrement et le déchiffrement. La clé privée est le triplet (G, S, P), où G est une matrice génératrice d'un code binaire comportant deux codes, de matrices génératrices G_1 et G_2 de longueur n_1 et n_2 qui peuvent corriger t_1 et t_2 erreurs, respectivement. S est une $k \times k$ matrice inversible aléatoire et P est une $n \times n$ matrice de permutation aléatoire où $n = n_1 + n_2$. la clé publique est (G_{pub}, t_1, t_2), où G_{pub} est la matrice $G_{pub} = SGP$.

Étape 1 : Génération des clés

1. Choisir un code QC-LDPC C_1 de longeur n_1 et de dimension k_1, et un code QC-MDPC C_2 de longueur n_2 et de dimension k_2.

2. Construire une matrice génératrice systématique G_1 pour C_1.

3. Construire une matrice $G = (G_1|G_2)$ où G_2 est la matrice génératrice du code C_2 avec $n_2 - k_2 = r$ impair.

4. Générer une suite aléatoire de bits en utilisant l'algorithme de Kanso pour remplir la matrice S de telle sorte qu'elle soit une matrice inversible et dense.

5. Générer aléatoirement une matrice inversible creuse P en utilisant l'approche de David MacKay [56].

6. Calculer la matrice publique $G_{pub} = SGP$.

Remarque 5.0.1. *Pour assurer la sécurité du schéma proposé, il est nécessaire que la matrice S soit dense. Si la sortie du générateur auto-rétrécissant modifié ne fournit pas une telle matrice, le processus de génération S sera répété. Puisque la sortie du générateur auto-rétrécissement modifié est* 1 *avec une probabilité de* 1/2*, la probabilité d'obtenir une matrice dense est élevée.*

Étape 2 : Chiffrement

Pour chiffrer un message m :

1. On génère aléatoirement un vecteur e de poids au plus t_1 dans les premières n_1 positions et t_2 dans les dernières n_2 positions.

2. Calculer $c = mG_{pub} + e$.

Étape 3 : Déchiffrement

Pour déchiffrer le texte chiffré c et trouver le message :

1. Calculer $u = cP^{-1} = mSG + eP^{-1}$.

2. Déterminer $z = mS$ par la correction d'erreurs de u en utilisant l'algorithme de décodage modifié de Gallager [35] introduit dans [19], pour C_1 et C_2.

3. Calculer $m = zS^{-1}$.

5.1 Etude de la securité du système proposé

Dans cette section, nous considérons les attaques structurelles et celles par décodage, la recherche exhaustive et les attaques par renvoi de message. Nous montrons que le schéma proposé est sécurisé contre toutes ces attaques. En outre, bien que le cryptosystème original de McEliece soit vulnérable à une attaque par renvoi de message, l'approche proposée est nettement plus résistante.

5.1.1 Attaque par recherche exhaustive

Dans cette attaque, l'attaquant recherche de façon aléatoire la clé privée (G, S, P). Puisque le nombre des matrices inversibles de taille $k \times k$ dans $\mathbb{F}_q$ est $\prod_{i=1}^{k}(q^k - q^{i-1})$, La recherche de la matrice sélectionnée nécessite le test pour $\prod_{i=1}^{k}(q^k - q^{i-1})^2$ matrices. La probabilité de trouver la matrice correcte est inférieurement bornée par $1/\left(q^{(k^2-k)/2}(q-1)^{(k^2-k)/2}\right)$. D'où, la complexité est exponentielle, donc cette attaque n'est pas réalisable pour des paramètres typiques du système.

5.1.2 Les attaques structurelles

L'objectif d'une attaque structurelle est de trouver un code équivalent au code privé avec un algorithme de décodage efficace (se fait en un temps polynomial). Dans le schéma proposé, une matrice dense S est utilisée pour contrer l'attaque algébrique dans [75]. De plus, la matrice génératrice G_2 d'un code QC-MDPC est employée pour augmenter la densité de sorte que le schéma soit résistant aux attaques sur les matrices creuses.

5.1.3 Attaque par décodage par ensemble d'information

Le décodage par ensemble d'information est une technique introduite par Prange [79] et Amélioré par la suite dans [7, 16, 58, 59].

Le facteur de travail global requis par l'algorithme de May, Meurer, et Thomae (MMT) [58] est

$$WF_{MMT} = \min_{p} \frac{\begin{pmatrix} n \\ w \end{pmatrix}}{\begin{pmatrix} n-k-l \\ w-p \end{pmatrix}\begin{pmatrix} k+l-p/2 \\ p/2 \end{pmatrix}} \tag{5.1}$$

où $l = \log_2\begin{pmatrix} k+l \\ p/2 \end{pmatrix}$. Cette attaque s'exécute en un temps $\mathcal{O}(2^{0.054n})$, donc la complexité est exponentielle. Ainsi, ce type d'attaque est infaisable pour n et k grands.

5.1.4 Attaque par renvoie de message

Pour un code de paramètres $n_0 = 4$, $n_1 = 3024$, $n_2 = 1008$, $k_1 = 2268$, $k_2 = 756$ et $d = 53$, La probabilité que deux vecteurs d'erreur aient un 1 à la position l est

$$Pr(e_1(l) = e_2(l)) = \left(\frac{t}{n}\right)^2 = \left(\frac{26}{16128}\right)^2 \approx 2.6 \times 10^{-6}.$$

Soit $L_0 = \{l \in \{1,\ldots,n\} : c_1(l) \oplus c_2(l) = e_1(l) \oplus e_2(l) = 0\}$ et $L_1 = \{l \in \{1,\ldots,n\} : c_1(l) \oplus c_2(l) = e_1(l) \oplus e_2(l) = 1\}$. La probabilité que exactement i coordonnées sont simultanément

brouillées par e_1 et e_2 est

$$\begin{aligned} p_i &= Pr(|\{l : e_1(l) = 1\} \cap \{l : e_2(l) = 1\}|) \\ &= \frac{\binom{26}{i}\binom{16128-26}{26-i}}{\binom{16128}{26}} \end{aligned}$$

et par conséquent

$$E(|L_1|) = \sum_{i=0}^{26} (52 - 2i)p_i \approx 100.$$

Par exemple, si $|L_1| = 99$, $|L_0| = n - |L_1| = 16029$ et seulement 8 entrées de L_0 sont affectées par une erreur. La probabilité de deviner $4 \times 3024 = 12096$ colonnes correctes de celles indexées par L_0 est 0.00001307, Donc après environ 76523 essais, on obtient un mot de code qui peut être décodé en utilisant la matrice génératrice publique. Dans la table 5.1, on calcule la probabilité de deviner k colonnes correctes de celles indexées par L_0 par rapport au nombre d'erreurs dans le cryptosystème. Comme prévu, l'augmentation du nombre d'erreurs améliore la sécurité. Ces résultats sont meilleurs que ceux obtenus pour cette attaque [14] sur le cryptosystème de McEliece [60]. Les résultats du tableau 5.1 indiquent qu'il n'est pas nécessaire d'utiliser un vecteur d'erreur de poids important.

Nombre d'erreurs	Probabilité de deviner k colonnes correctes de celles indexées par L_0	Nombre d'essais
1	0.245	4
2	0.0602	16
3	0.0148	67
4	0.00362	276
5	0.000888	1126
8	$1.31\ 10^{-5}$	76523
10	$7.84\ 10^{-7}$	1275200
15	$6.89\ 10^{-10}$	1450441408

TABLE 5.1 – La probabilité de deviner k colonnes correctes de celles indexées par L_0 par rapport au nombre d'erreurs.

5.1.5 Autres attaques

Deux nouvelles attaques utilisant une technique de quadrature ont été récemment proposées. Löndahl et al. [55] ont introduit une attaque de récupération de clé et une attaque de récupération de message contre un cryptosystème de McEliece basé sur les codes quasi-cycliques. Shooshtari et al. [83] ont appliqué une technique de quadrature au

texte chiffré afin de trouver un mot de code de poids faible au carré dans une dimension inférieure. Ensuite, le mot de code original de faible poids a été obtenu à partir des mots de code à poids faible au carré. Dans le système proposé, r est impair, Donc ce type d'attaque n'est pas possible parce que l'attaque de [55] ne s'applique qu'aux codes quasi-cycliques dont la longueur et la dimension est un multiple d'une puissance de 2, et l'attaque de [83] peut être employée quand la longueur du texte chiffré est un multiple d'une puissance de 2.

Recemment, Bardet et al. [6] ont présenté une approche pour trouver des faiblesse sur les clés pour le schéma de cryptage basé sur les codes QC-MDPC. Cette attaque ne peut pas être appliquée au cryptosystème proposé parce que la partie QC-LDPC de la clé publique ne satisfait pas la condition d'orbite faible donnée dans [6].

Un adversaire peut être capable de déterminer le nombre d'itérations du décodeur par un canal latéral, et d'obtenir ainsi des informations sur la matrice de contrôle de parité creuse. Par conséquent, un attaquant peut obtenir des informations sur la clé secrète en observant le processus de décodage. L'ajout de fausses itérations jusqu'à ce qu'un nombre prescrit soit atteint fournira un décodeur de temps constant pour contrer cette attaque.

Guo et al. [41] ont présenté une attaque de récupération de clé efficace sur le schéma QC-MDPC (appelé attaque par réaction), En utilisant le fait que le déchiffrement utilise le décodage itératif. Dans le système proposé, un décodeur de bit-flipping [19] peut être utilisée, ce qui augmente la complexité de l'attaque d'un facteur supérieur à 2^{26}. De plus, l'utilisation des codes QC-LDPC code rend cette attaque impossible.

5.1.6 Paramètres proposés

Toutes les variantes du cryptosystème de McEliece nécessitent une très grande clé publique. Dans cette section, des paramètres sont suggérés pour le schéma proposé. Pour un niveau de sécurité de 80 bits , les paramètres du code sont : $n_0 = 4$, $n_1 = 3024$, $n_2 = 1008$, $k_1 = 2267$ et $k_2 = 756$, donc la clé publique a une longueur de 5290 bits. Une comparaison de la taille des clés pour ces paramètres avec les cryptosystèmes de McEliece basés sur les codes de Goppa [11], les codes de Goppa quasi-dyadiques (QD) [64] et les codes QC-LDPC [2], est donnée dans la table 5.2. Ces résultats montrent que le système proposé a la plus petite taille de clé.

Schéma	Proposé	Goppa [11]	QD-Goppa [64]	QC-LDPC [2]
Taille des clés	5290	460647	20480	6084

TABLE 5.2 – Taille de la clé en bits de quatre cryptosystèmes pour un niveau de sécurté de 80.

5.2 Conclusion

Une nouvelle variante du cryptosystème de McEliece basée sur les codes QC-LDPC et QC-MDPC a été présentée dans ce chapitre.
Un générateur auto-rétrécissant modifié a été utilisé pour obtenir des bits aléatoires pour construire la matrice génératrice. Ce système s'est avéré sécurisé contre les attaques structurelles et celles par décodage connues. En outre, il a une clé plus courte en le comparant avec d'autres schéma.

Chapitre 6

Nouveau système de chiffrement de type GPT

Ce chapitre a fait l'objet d'une publication à la conférence internationale "ICCC 2015" [69].

Les codes en métrique rang ont été introduits en 1978 par Deslarte [22], puis en 1985 par Gabidulin, où une discussion sur la différence et la similitude entre la métrique rang et la métrique de Hamming a été entamée, d'où, vient le nom des codes de Gabidulin.
La métrique rang convient à un modèle de canal où les mots de code peuvent être vus comme des matrices à coefficients dans un corps fini, où les erreurs arrivent sur les lignes ou sur les colonnes.
La théorie de la métrique rang constitue une généralisation des travaux de l'auteur sur les codes qui corrigent les tableaux d'erreurs.
Dans [28], une généralisation de la borne de Singleton et une nouvelle famille de codes atteignant cette borne ont été présentées. Ces codes sont équivalents aux codes de Reed-Solomon généralisés pour la métrique de Hamming. Gabidulin a introduit aussi dans son article un algorithme de décodage efficace qui fonctionne en temps polynomial.
La métrique rang n'est pas vraiment intéressante pour la détection d'erreurs, mais elle est bien adaptée à la correction des codes entrelacés, le codage réseau, le codage espace-temps(spatio-temporel) et notamment en cryptographie (schéma de chiffrement, signature, protocole d'identification ...), où elle prend tout son intérêt.
En cryptographie, l'outil principal consistait à masquer la structure des codes de Gabidulin de manières différents et d'utiliser en général, un système de chiffrement de type McEliece (ou Niederreiter).
En 1991, Gabidulin, Paramonov et Tretjakov [30] ont introduit le premier cryptosystème basé sur les codes correcteurs d'erreurs utilisant la métrique rang.

Dans ce qui suit, nous allons présenté des propriétés importantes sur les codes en métrique rang.

6.1 Quelques propriétés de la métrique rang

Les résultats de cette section sont tirés de [28] et [53].
Soit $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ une base de $\mathbb{F}_{q^m}$ sur $\mathbb{F}_q$. Alors tout élément $b \in \mathbb{F}_{q^m}$ s'écrit de manière unique sous la forme $b = \sum_{i=1}^{m} b_i \beta_i$ où $b_j \in \mathbb{F}_q$.
Un vecteur b de longueur n sur $\mathbb{F}_{q^m}$ peut être représenté comme une matrice de taille $m \times n$ sur $\mathbb{F}_q$. Ce vecteur est en bijection avec l'ensemble des matrices $m \times n$ modulo $\sum_{i=1}^{m} b_i \beta_i$.
Alors une norme peut être définie sur $\underbrace{\mathbb{F}_{q^m} \times \mathbb{F}_{q^m} \times \cdots \times \mathbb{F}_{q^m}}_{n \text{ fois}}$.

Définition 6.1.1. *Soit $x = (x_1, x_2, \ldots, x_j, \ldots, x_n)$ un mot de longueur n sur $\mathbb{F}_{q^m}$. Pour tout j, un vecteur x_j peut s'écrire comme vecteur de $\mathbb{F}_q$.*

$$x_j = \begin{pmatrix} x_{1j} \\ \vdots \\ x_{mj} \end{pmatrix}$$

Alors x est vu comme une matrice dans $\mathbb{F}_q$.

$$X = \begin{pmatrix} x_{11} & \ldots & x_{1j} & \ldots & x_{1n} \\ \vdots & & \vdots & & \vdots \\ x_{i1} & \ldots & x_{ij} & \ldots & x_{in} \\ \vdots & & \vdots & & \vdots \\ x_{m1} & \ldots & x_{mj} & \ldots & x_{mn} \end{pmatrix}$$

$x = (x_{ij})_{\substack{1 \le i \le m \\ 1 \le j \le n}}$ est alors une matrice à m lignes et n colonnes dans $\mathbb{F}_q$.
Soit X la matrice de taille $m \times n$ de l'extension des coordonnées x_j de x sur la base β, où $x_j = \sum_{i=1}^{m} x_{ij} \beta_i$.
On définit la norme rang (ou le rang) du mot x notée $Rg(x)$ (ou $Rg(x|\mathbb{F}_q)$ comme étant le rang de la matrice X, i.e, Rg(x)=Rg(X).

Cette définition satisfait les conditions d'une norme sur l'espace $\mathbb{F}_{q^m}^n$. On peut associer à cette norme une distance appelée distance rang.

Définition 6.1.2. *Soit x et y deux vecteurs de $\mathbb{F}_{q^m}^n$. On définit la distance rang par :*

$$d_r(x, y) = Rg(x - y)$$

La métrique rang est reliée à la métrique de Hamming par la proposition suivante :

Proposition 6.1.1. *Soit $x \in \mathbb{F}_{q^m}^n$. Si $w(x)$ désigne le poids de Hamming du mot x, on a : $Rg(x) \leq w(x)$.*

Cette proposition indique que la métrique rang est plus grossière que la métrique de Hamming.
Maintenant, on va définir un paramètre important pour tout code en métrique rang.

Définition 6.1.3. *La distance rang minimale d'un code C de longueur n sur $\mathbb{F}_{q^m}$ est la valeur minimale des rangs des mots qui le constituent.*

$$d_r = \underset{x \in C}{Min}(Rg(x))$$

Pour un code linéaire C, nous avons $d \leq n-k+1$, donc d'après la proposition ci-dessus, nous avons également $d_r \leq n-k+1$.
Dans le cas de la métrique rang, nous avons un théorème similaire que celui de la métrique de Hamming, qui nous donne une condition nécessaire et suffisante sur un code linéaire pour que sa distance minimale soit égale à une valeur donnée d.

Théorème 6.1.1. *Soit C un code linéaire sur $\mathbb{F}_{q^m}$, de longueur n, de dimension k et de distance minimale d.*
Soit H une matrice de parité de C. Alors la distance rang minimale du code C est égale à d si et seulement si :

1. *Pour toute matrice Y de taille $(d-1) \times n$, à coefficients dans $\mathbb{F}_q$ et de rang $d-1$, on a :*

$$Rg(YH^T) = d-1$$

2. *Il existe une matrice Y_0 de taille $d \times n$ à coefficients dans $\mathbb{F}_q$ et de rang d, qui vérifie :*

$$Rg(Y_0H^T) < d$$

Comme en métrique de Hamming, on trouve les même notions pour les codes en métrique rang telle que la notion de matrice génératrice et le code dual qui sont définis de la même façon.
On trouve aussi des équivalents des bornes telles que la borne de Singleton et la borne de Gilbert-Varshamov.
Avant de donner l'expression de la borne de Singleton, nous introduisons d'abord la notion du code groupe, qui est une généralisation de la définition d'un code linéaire.

Définition 6.1.4 (Code groupe)**.** *Un code groupe de longueur n et de taille M sur un alphabet $\mathbb{F}_{q^m}$ est un ensemble de M mots $x = (x_1, x_2, \ldots, x_n)$ de longueur n sur $\mathbb{F}_{q^m}$, qui forment un groupe additif.*

Définition 6.1.5 (Borne de Singleton)**.** *Soit C un code de longueur n, de taille M et de distance rang minimale d_r sur $\mathbb{F}_{q^m}$. La borne de Singleton du code C est donnée par :*

$$M \leq q^{(m-d_r+1)m}$$

Si $n \leq m$ et la borne de Singleton est atteinte, on dit que le code C est MDR (Maximum Rank Distance).

Remarque 6.1.1. *En métrique rang, il n'existe pas de code parfait.*

En métrique rang, on peut évaluer la taille des boules en comptant les matrices à m lignes et n colonnes de rang fixé.
Soit x un vecteur de longueur n à coefficients dans $\mathbb{F}_{q^m}$. Soit $S(x,t)$, la sphère centrée en x de rayon t et $B(x,t)$ la boule centrée en x et de rayon t. On a :
Le nombre d'éléments $S(x,t)$ dans une sphère de rayon t dans $\mathbb{F}_{q^m}^n$ est égal au nombre de matrices q-aires de rang t.

$$|S(x,t)| = \left(\prod_{j=0}^{t-1}\left(q^n - q^j\right)\right)\begin{bmatrix} \mathrm{m} \\ \mathrm{t} \end{bmatrix}_q$$

où $\begin{bmatrix} \mathrm{m} \\ \mathrm{t} \end{bmatrix}_q = \prod_{j=0}^{t-1} \frac{q^m - q^j}{q^t - q^j}$ est appelé binôme de Gauss, qui est le nombre d'espaces vectoriels de dimension t dans $\mathbb{F}_{q^m}$ (c'est une généralisation du binôme de Newton).
Le volume d'une boule $B(x,t)$ est donné par :

$$|B(x,t)| = \sum_{i=0}^{t} |S(x,i)|$$

Les cardinaux de $S(x,t)$ et de $B(x,t)$ vérifient les encadrements suivants :

$$q^{(m+n-2)t-t^2} \leq |S(x,t)| \leq q^{(m+n+1)t-t^2}$$

$$q^{(m+n-2)t-t^2} \leq |B(x,t)| \leq q^{(m+n+1)t-t^2+1}$$

En métrique rang, il existe aussi une borne d'existence des codes.

Proposition 6.1.2 (Borne de Gilvert-Varshamov)**.** *Si $M \times |B(0,d-1)| < q^{mn}$, alors il existe un code de taille $M+1$ et de distance minimale d sur $\mathbb{F}_{q^m}$.*

Dans les deux propositions suivantes, nous donnons la répartition probabiliste des rangs des mots composent soit un code aléatoire ou un code MRD.

Proposition 6.1.3. *Soit C un code $(n,M,d)_r$ aléatoire. Alors le nombre A_i de mots de rang i dans C vérifie en moyenne :*

$$E(A_i) = \frac{M \times |S(0,i)|}{q^{mn}}$$

Proposition 6.1.4. *Soit $A_s(n,d)$ le nombre de mots de rang s d'un code MRD sur $\mathbb{F}_{q^m}$. Alors :*

$$A_{d+l}(n,d) = \begin{bmatrix} n \\ d+l \end{bmatrix}_q \sum_{t=0}^{l} (-1)^{t+l} \begin{bmatrix} d+l \\ l+t \end{bmatrix}_q q^{\binom{l-t}{2}} \left(q^{m(t+1)} - 1\right)$$

6.2 Les codes en métrique rang

6.2.1 Les codes de Gabidulin

Ces codes ont été introduits en 1985 par Ernest Gabidulin [28], ils atteignent la borne de Singleton pour la métrique rang (se sont des codes MRD sur $\mathbb{F}_{q^m}$ de longueur $n \leq m$). Avant de définir les codes de Gabidulin, nous introduisons l'automorphisme de Frobenius.

Définition 6.2.1. *Pour un corps fini $\mathbb{F}_{q^m}$, l'automorphisme de Frobenius, noté θ, est l'application définie par :*

$$\begin{array}{rccc} \theta : & \mathbb{F}_{q^m} & \longrightarrow & \mathbb{F}_{q^m} \\ & x & \longrightarrow & x^q \end{array}$$

Pour tout $g_j \in \mathbb{F}_{q^m} \cong \mathbb{F}_q^m$, on note $g_j^{[i]} = \theta^i(g_j)$ la puissance i-ème de l'automorphisme de Frobenius évalué en g_j

Définition 6.2.2. *Soit $g = (g_1, g_2, \ldots, g_n)$ un vecteur de $\mathbb{F}_{q^m}^n$. On suppose que la famille $\{g_1, g_2, \ldots, g_n\}$ est libre dans $\mathbb{F}_q^m$.*
Le code de Gabidulin $Gab_k(g)$ (ou $Gab[n,k]$) de dimension k et de vecteur générateur g est le code généré par la matrice :

$$G = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^{[1]} & g_2^{[1]} & \cdots & g_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{[k-1]} & g_2^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}$$

La représentation de la matrice génératrice d'un code de Gabidulin sous forme systématique est donnée en utilisant les q-polynômes.

Définition 6.2.3. *Soit $\mathbb{F}_{q^m}$ un corps fini. On appelle q-polynôme de degré l sur $\mathbb{F}_{q^m}$, un polynôme*

$$P(x) = \sum_{i=0}^{l} a_i X^{q^i}, \text{ où } a_i \in \mathbb{F}_{q^m}$$

La matrice génératrice d'un code de Gabidulin sous forme systématique est donnée ci-dessous.

$$G_{syst} = \left(\begin{array}{cccc|ccc} 1 & 0 & \ldots & 0 & P_1(g_{k+1}) & \ldots & P_1(g_n) \\ 0 & 1 & \ddots & 0 & P_2(g_{k+1}) & \ldots & P_2(g_n) \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & \ldots & 1 & P_k(g_{k+1}) & \ldots & P_k(g_n) \end{array}\right)$$

où pour $i = 1, \ldots, k$, P_i est l'unique q-polynôme de degré k vérifiant pour tout $j \leq k$, $P_i(g_j) = \delta_{i,j}$.
Le code de Gabidulin peut être vu comme un code d'évaluation de q-polynômes de q-degré strictement inférieur à k sur le vecteur g :

$$Gab_k(g) = \{(P(g_1), P(g_2), \ldots, P(g_n)) |\ P \text{ un } q\text{-polynôme de degré } k-1\}$$

Le dual du code $Gab_k(g)$ est de matrice génératrice :

$$H = \begin{pmatrix} h_1 & h_2 & \dots & h_n \\ h_1^{[1]} & h_2^{[1]} & \dots & h_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{[d-2]} & h_2^{[d-2]} & \dots & h_n^{[d-2]} \end{pmatrix}$$

où $h = \left(\lambda_1^{[d]}, \lambda_2^{[d]}, \dots, \lambda_n^{[d]}\right)$ et les δ_i sont solutions de l'équation :

$$\sum_{i=1}^{n} \lambda_i g_i^{[j]} = 0$$

6.2.2 Algorithme de décodage des codes de Gabidulin

Gabidulin a utilisé la méthode de décodage par syndrome pour décoder le code qu'il a construit. Vu la ressemblance entre les codes de Gabidulin et les codes de Reed-Solomon, des algorithmes de décodage ont été conçus en s'inspirant des travaux réalisés pour les codes de Reed-Solomon. Par exemple, on trouve P. Loidreau [52] qui a proposé un algorithme pour décoder les codes de Gabidulin, en s'inspirant d'un algorithme de décodage des codes de Reed-Solomon (Algorithme de Berlekamp-Welch).
L'algorithme de décodage proposé par Gabidulin, procède de la manière suivante :

1. Calcul du syndrome $s = HY^T\beta^T$, où Y est une matrice de taille $l \times n$ à coefficients dans $\mathbb{F}_q$ et $\beta = (\beta_1, \beta_2, \dots, \beta_l)$ constitué de l éléments linéairement indépendants sur $\mathbb{F}_q$ tel que $e = \beta Y$.

2. Détermination d'un vecteur β, souvent appelé base des solutions. On construit un q-polynôme de q-degré l, qui a pour racine l'espace vectoriel engendré par les composantes de β.On cherche à déterminer les racines du q-polynôme.

3. Détermination de la matrice $X = HY^T$, en résolvant le système $s = X\beta^T$.

4. Détermination de la matrice Y, puis calcul de $e = \beta Y$.

6.2.3 Les codes convolutionnels en métrique rang

Dans cette partie, nous présentons la structure des codes convolutionnels en métrique rang introduite dans [95], que nous avons utilisée pour construire notre cryptosystème. Nous avons utilisé des codes convolutionnels de mémoire 1 (ou ce qu'on appelle codes d'unité à mémoire partielle "PUM"). Par analogie avec la métrique de Hamming, nous introduisons la forme des mots de la matrice génératrice, la matrice de contrôle de parité ainsi que la distance rang libre.

Matrice de contrôle du code PUM Partail Unit Memory)

Soit $\mu_H \geq 1$ et H une matrice de contrôle de parité semi-infinie d'un code convolutionnel C sur $\mathbb{F}_{q^m}$, de taux $R = k/n \geq \mu_H/(\mu_H+1)$.
Chaque matrice $H^{(i)}$ doit être une matrice de contrôle d'un code de Gabidulin $Gab[n,k]$, i.e.

$$H^{(i)} = qvan_{n-k}(h^{(i)}) = qvan_{n-k}\left(\left(h_0^{(i)} h_1^{(i)} \dots h_{n-1}^{(i)}\right)\right)$$

$\forall i \in [0, \mu_H]$, où $h_0^{(i)}, h_1^{(i)}, \dots, h_{n-1}^{(i)} \in \mathbb{F}_{q^m}$ sont linéairement indépendants sur $\mathbb{F}_q$.
De plus, soit la matrice H^c qui définie un code $Gab[n, n-(\mu_H+1)(n-k)]$ et $H^{(r(i))} \stackrel{\text{def}}{=} \left(H_0^{(i)} H_1^{(i)} \dots H_{n-1}^{(i)}\right)$ définie un code $Gab[(i+1)n, in+k], \forall i \in [0, \mu_H]$.
Alors la matrice H est une matrice de contrôle d'un code convolutionnel sur $\mathbb{F}_{q^m}$ de taux $R \geq \mu_H/(\mu_H+1)$.

Matrice génératrice d'un code PUM

Soit $k + k^{(1)} \leq n \leq m$, où $k^{(1)} \leq k$ et soit $g_0, g_1, \dots, g_{n-1} \in \mathbb{F}_{q^m}$ linéairement indépendants sur $\mathbb{F}_q$. Pour $k^{(1)} = k$, on définit un code $UM(n,k)$ et pour $k^{(1)} < k$, un code $PUM(n,k|k^{(1)})$ sur $\mathbb{F}_{q^m}$ par une matrice génératrice terminée à zéro-forcé G_{term}, avec $\mu = 1$ et les $k \times n$ sous matrices $G^{(0)}$ et $G^{(1)}$. Où

$$G_{term} = \begin{pmatrix} G^{(0)} & G^{(1)} & \dots & G^{(\mu)} & & \\ & G^{(0)} & G^{(1)} & \dots & G^{(\mu)} & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & G^{(0)} & G^{(1)} & \dots & G^{(\mu)} \end{pmatrix},$$

$$G^{(0)} = \begin{bmatrix} G^{(00)} \\ G^{(01)} \end{bmatrix} = \left[\begin{array}{cccc} g_1 & g_2 & \cdots & g_n \\ g_1^{[1]} & g_2^{[1]} & \cdots & g_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{[k^{(1)}-1]} & g_2^{[k^{(1)}-1]} & \cdots & g_n^{[k^{(1)}-1]} \\ \hline g_1^{[k^{(1)}]} & g_2^{[k^{(1)}]} & \cdots & g_n^{[k^{(1)}]} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{[k-1]} & g_2^{[k-1]} & \cdots & g_n^{[k-1]} \end{array}\right]$$

et

$$G^{(1)} = \begin{bmatrix} G^{(10)} \\ 0 \end{bmatrix} = \left[\begin{array}{cccc} g_1^{[k]} & g_2^{[k]} & \cdots & g_n^{[k]} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{[k+k^{(1)}-1]} & g_2^{[k+k^{(1)}-1]} & \cdots & g_n^{[k+k^{(1)}-1]} \\ \hline 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{array}\right]$$

Avant de donner la définition de la distance rang libre, nous rappelons d'abord les notions de somme distance rang (sum rank distance) et de la somme poids rang (sum rank weight).

Définition 6.2.4. *Soient a et b deux vecteurs appartenant à $\mathbb{F}_{q^m}^{nN}$ tels que :*

$$a = (a^{(0)}a^{(1)}\dots a^{(N-1)}) \text{ et } b = (b^{(0)}b^{(1)}\dots b^{(N-1)}),$$

avec $a^{(i)}$, $b^{(i)} \in \mathbb{F}_{q^m}^{n}$, $\forall i \in [0, N-1]$. La somme poids rang du vecteur a est la sommes des rangs des sous-vecteurs :

$$w_{r,\Sigma}(a) \overset{def}{=} \sum_{i=0}^{N-1} w_r(a^{(i)}) = \sum_{i=0}^{N-1} Rg(a^{(i)})$$

La somme distance rang entre a et b est la somme poids rang de la différence des vecteurs :

$$d_{\Sigma,r}(a,b) \overset{def}{=} w_{\Sigma,r}(a,b) = \sum_{i=0}^{N-1} Rg(a^{(i)} - b^{(i)}).$$

Une mesure importante pour les codes convolutionnels est la distance rang libre.

Définition 6.2.5 (Distance rang libre)**.** *La distance rang libre d'un code convolutionnel C est la somme distance rang minimale entre deux mots de code différents $a, b \in C$.*

$$d_{f,r} = \min_{\substack{a,b \in C \\ a \neq b}} \{d_{\Sigma,r}(a,b)\} = \min_{\substack{a,b \in C \\ a \neq b}} \left\{ \sum_{i=0}^{\infty} Rg(a^{(i)}, b^{(i)}) \right\}$$

6.2.4 Les codes LRPC

Dans cette partie, nous présentons une autre famille de codes en métrique rang : les codes LRPC (Low Rank Parity Check), pour lesquels il existe un algorithme de décodage probabiliste. Cette famille de codes est une analogie aux codes LDPC pour la métrique rang.
Les codes LRPC ont été introduits récemment par P. Gaborit et al. [32] pour être appliqués par la suite en cryptographie.

Définition 6.2.6. *Un code avec une matrice de contrôle de faible rang d, de longueur n et de dimension k sur $\mathbb{F}_{q^m}$ est un code en métrique rang dont la matrice de parité H de dimension $(n-k) \times n$ est telle que le sous espace de $\mathbb{F}_{q^m}$ engendré par les coefficients $(h_{ij})_{1 \leq i \leq n-k; 1 \leq j \leq n}$ de H a pour dimension au plus d. Nous appellerons cette dimension le poids (rang) de H.*

On note F le sous-espace de $\mathbb{F}_{q^m}$ engendré par les coefficients de H et $\{F_1, F_2, \dots, F_n\}$ une base de F.
Dans la pratique cela signifie que pour tout $1 \leq i \leq n-k; 1 \leq j \leq n$, il existe des $(h_{ijk})_{1 \leq k \leq d} \in \mathbb{F}_q$ tels que :

$$h_{ij} = \sum_{k=1}^{d} h_{ijk} F_k$$

6.2.5 Algorithme de décodage des codes LRPC

L'idée utilisée pour le décodage des codes LRPC ressemble beaucoup à la méthode de décodage des codes BCH, où l'on retrouve le polynôme localisateur d'erreur, qui donne le support de l'erreur, puis la valeur des coordonnées de l'erreur.
Considérons un code $C[n,k]_r$ sur $\mathbb{F}_{q^m}$ de faible rang d, notons G une matrice génératrice de taille $k \times n$ et H une matrice de parité de taille $(n-k) \times n$, telles que toutes les coordonnées h_{ij} de H appartiennent à un espace F de rang d dont une base est $\{F_1, F_2, \ldots, F_d\}$, et supposons que H est choisie de sorte qu'il existe une matrice de décodage inversible associée D_H.
On suppose que le mot reçu est $y = xG + e$ avec x et e appartiennent à $\mathbb{F}_{q^m}^n$ où $e = (e_1, e_2, \ldots, e_n)$ est le vecteur erreur de rang r dont le support est $E = \{E_1, E_2, \ldots, E_r\}$ ce qui veut dire que chaque composante e_i peut être écrite sous la forme suivante :

$$e_i = \sum_{j=1}^{r} e_{ij} E_j$$

où les coefficients e_{ij} sont dans $\mathbb{F}_q$.
Les étapes de cet approche sont décrits comme suit :

1. **Génération de l'espace syndrome**
   Calculer le vecteur syndrome $H.y^t = s = (s_1, \ldots, s_{n-k})$ et l'espace syndrome $S =< s_1, \ldots, s_{n-k} >$

2. **Détermination du support de l'erreur**
   Définir $S_i = F_i^{-1} S$, le sous-espace où tous les générateurs de S sont multipliés par F_i^{-1}. Générer le support de l'erreur $E = S_1 \cap S_2 \cap \cdots \cap S_d$. Pour cela déterminer $S_1 \cap S_2$ et calculer sa dimension. Si cette dimension est égale à celle de E alors $E = S_1 \cap S_2$, sinon calculer $(S_1 \cap S_2) \cap S_3$ et ainsi de suite. Générer ensuite une base $\{E_1, ..., E_r\}$ de E.

3. **Retrouver le vecteur erreur** e
   Écrire e_i,$1 \leq i \leq n$, dans le support de l'erreur c'est à dire $e_i = \sum_{j=1}^{r} e_{ij} E_j$, et résoudre le système $H.e^t = s$ où les équations $H.e^t$ et les coordonnées du syndrome s_i sont écrites comme des éléments de l'espace produit $P =< E.F >$ dans la base $\{F_1E_1, \ldots, F_1E_r, \ldots, F_dE_1, ..., F_dE_r\}$. Le système a nr inconnues (les e_{ij}) et $(n-k)rd$ équations à partir du syndrome.

4. **Retrouver le message** x
   Retrouver x à partir de $xG = y - e$.

L'algorithme probabiliste utilisé pour le décodage des codes LRPC, présenté ci-dessus, permet de décoder un vecteur erreur aléatoire e de rang faible r tel que $rd \leq n-k$ avec une probabilité de $q^{-(n-k+1-rd)}$ et une complexité de l'ordre de $\mathcal{O}(r^2(4d^2m + n^2))$.

6.3 Applications des codes en métrique rang à la cryptographie

6.3.1 Le cryptosystème GPT

Les trois étapes du premier systèmes de chiffrement basé sur les codes en métrique rang, sont les suivantes :

Génération des clés La clé publique est une matrice de taille $k \times (n + t_1)$ donnée par :

$$G_{pub} = S\,[X \quad G]\,P,$$

où

1. G est une matrice génératrice d'un code de Gabidulin, choisie au hasard, de taille $k \times n$.
2. S est une matrice d'ordere k sur $\mathbb{F}_{q^m}$, inversible.
3. X est une matrice de distorsion de taille $k \times t_1$ sur $\mathbb{F}_{q^m}$, avec $t_1 < t$.
4. P une matrice inversible de taille $(t_1 + n) \times (t_1 + n)$ sur $\mathbb{F}_q$.

La clé privée est (S, G, X, P).

Chiffrement Pour envoyer un message (texte clair) $x = (x_1, \ldots, x_k)$ à quelqu'un, on calcul c de longueur $n + t_1$, tel que :

$$c = xG_{pub} + e$$

où $e = (e_1, \ldots, e_{n+t_1})$ est un vecteur d'erreur de rang $t_2 \leq t$.

Déchiffrement Pour déchiffrer le message chiffré c, il faut calculer

$$cP^{-1} = xS\,[X \quad G] + eP^{-1}$$

Le rang de l'erreur eP^{-1} est plus petite que t. Ceci permet d'appliquer un algorithme de décodage rapidement pour corriger les erreurs, puis on extrait xS et on en déduit le message x tel que $x = (xS)S^{-1}$.
Dans leur système originel, les auteurs n'utilisaient pas de matrice de distorsion. Suite à l'attaque de Gibson [36], Gabidulin et Ourivski [29] ont rajouté la matrice de distorsion X au système GPT pour se prémunir de cette attaque.

Construction de la matrice de distorsion :
Loidreau [52] a présenté dans sa thèse, une version plus simple et équivalente à celle de Gabidulin, de la construction de la matrice de distorsion. Pour avoir une telle matrice, Loidreau [52] a proposé l'algorithme suivant :

- Générer deux matrice A et B telles que $A = (Q|0)$ est une matrice de longueur n, Q une matrice de taille $k \times t_1$ à coefficients dans $\mathbb{F}_{q^m}$ et 0 est une matrice de taille $k \times n - t_1$ remplie de 0. La matrice B est une matrice inversible à coefficients dans $\mathbb{F}_q$, de taille $n \times n$.
- Calculer $X = AB$, vérifiant :

$$\forall c = (c_1, \ldots, c_k) \in \mathbb{F}_{q^m}^k, Rg(cX) \leq t_1$$

Le nombre des matrices de distorsion qu'on peut obtenir est :

$$\prod_{i=0}^{t_1-1} \frac{\left(q^{mk} - q^{mi}\right)}{\left(q^{mt_1} - q^{mi}\right)} \cdot \prod_{i=0}^{n-1} \left(q^n - q^i\right)$$

6.3.2 Le cryptosystème LRPC

Génération de la clé

Choisir aléatoirement un code LRPC C sur $\mathbb{F}_{q^m}$ de faible rang d ; le code est déterminé à partir de la matrice de parité H. Notons G une matrice génératrice de C et D_H une matrice de décodage corrigeant les erreurs de poids $\leq r$. Déterminer une matrice R à coefficients dans $\mathbb{F}_{q^m}$ inversible de dimension $(n-k) \times (n-k)$.

- Clé secrète : La matrice de rang faible H, la matrice masquante R.
- Clé publique : La matrice $G' = RG$.

Chiffrement

Convertir le vecteur information v en un mot de code x, choisir une erreur aléatoire e de rang r sur $\mathbb{F}_{q^m}$. Le chiffrement de v est $c = xG' + e$.

Déchiffrement

Calculer le syndrome $s = H.c^T$ et retrouver le vecteur erreur e, ensuite xR, et en enfin x.

6.4 Notre système basé sur la construction $(u|u+v)$

Dans notre schéma de chiffrement, nous utilisons la construction $(u|u+v)$ d'un code linéaire. Le code sera donc de la forme $C = [C_1, C_2]$, de longueur $2n$ et de dimension $k = k_1 + k_2$, où C_1 est un $[n, k_1, d_1]$ code LRPC et C_2 est un $[n, k_2, d_2]$ code (P)UM. La distance rang minimale du code C est $d = \min\{2d_1, d_2\}$. La matrice génératrice du code C est donnée par :

$$G = \begin{bmatrix} G_1 & G_1 \\ 0 & G_2 \end{bmatrix} \tag{6.1}$$

telle que G_1 (respectivement G_2) est la matrice génératrice du code C_1 (respectivement C_2).
Notre système de chiffrement représente une variante du cryptosystème GPT dont la structure du code utilisé a été changée, ainsi que quelques paramètres, pour résister aux attaques qui existent. Notre schéma proposé fonctionne comme suit.

Génération des clés La clé publique utilisée est une matrice génératrice

$$G_{pub} = S\,[X \quad G]\,P$$

de taille $k \times (n + t_1)$, du code C présenté ci-dessus 6.1.

où

- La matrice S est une matrice inversible aléatoire de taille $k \times k$ à coefficients dans $\mathbb{F}_{q^m}$ avec m pair.
- La matrice X est une matrice de distorsion de taille $k \times t_1$ sur $\mathbb{F}_{q^m}$ de rang plein $t_1 > n - k$.
- La matrice P est une matrice de permutation choisie au hasard, de taille $(t_1 + n) \times (t_1 + n)$ sur $\mathbb{F}_q$.

La clé secrète est le quadruplet (S, G, X, P), l'algorithme de décodage du code LRPC et l'algorithme de décodage par effacement d'un code (P)UM.

Chiffrement Pour envoyer un message $x = (x_1, x_2, \ldots, x_k)$ à quelqu'un, tel que $x_i \in \mathbb{F}_{q^m}$ pour $i = 1, \ldots, k$, l'émetteur calcule :

$$c = xG_{pub} + e,$$

où e est un vecteur d'erreurs artificiel de rang inférieur ou égal à $\lfloor \frac{(n-k)}{2} \rfloor$.

Déchiffrement Pour déchiffrer le message crypté, le récepteur calcule :

$$cP^{-1} = xS\,[X \quad G] + eP^{-1}, \tag{6.2}$$

On note $c' = cP^{-1}$ et $e' = eP^{-1}$, alors l'équation 6.2 devient $c' = xS\,[X \quad G] + e'$, qui est équivalente à

$$c' = xS \begin{bmatrix} X & G_1 & G_1 \\ & 0 & G_2 \end{bmatrix} + e'. \tag{6.3}$$

Donc le récepteur va appliquer l'algorithme de décodage des codes LRPC sur C_1 et l'algorithme de décodage des codes (P)UM codes sur C_2 pour corriger l'erreur e' telle que $rk(e'|\mathbb{F}_q) \leq t$. Ensuite il extrait xS et retrouve le message clair x en calculant $(xS)S^{-1} = x$.

6.5 Analyse de la sécurité

Plusieurs attaques ont été proposées contre le cryptosystème à clé publique de type GPT. Ces attaques peuvent être classées en deux type différents : les attaques structurelles et les attaques par décodage. La première version du GPT-PKC a été attaquée par Gibson [36]. Plus tard, Overbeck [76, 77] a présenté deux attaques structurelles contre le cryptosystème GPT. Dans cette partie, nous montrons que notre schéma de chiffrement est sécurisé contre toutes les attaques connues jusqu'à présent. En outre, nous avons réduit la taille de la clé publique comparée par rapport aux variantes de McEliece et à certains systèmes de type GPT.

6.5.1 Attaques combinatoires

Habituellement, ces attaques sont les plus efficaces lorsque q, n et k prennent de petites valeurs. La première attaque existante a été introduite par Chabaud et Stern [18] mais elle n'est pas applicable. Plus tard, Ourivski et Johanson [74] ont amélioré cette attaque et en ont proposé une nouvelle. Dans [33], Gaborit et al. ont généralisé ces attaques.
Le problème de décodage d'un (n,k) code linéaire C sur $\mathbb{F}_{q^m}$ de distance rang minimale d peut être formulé comme suit :
Étant donné $G \in \mathcal{M}_{k\times n}(\mathbb{F}_{q^m})$ et $c \in \mathbb{F}_{q^m}^n$, trouver $u \in \mathbb{F}_{q^m}^k$ tel que $e = c - uG$ ait le plus petit rang $r = Rk(e|\mathbb{F}_q)$.
L'idée d'Ourivski et de Johanson est basée sur la construction d'un code C_e de matrice génératrice $\begin{pmatrix} G \\ c \end{pmatrix} = \begin{pmatrix} I_k & 0 \\ u & 1 \end{pmatrix}\begin{pmatrix} G \\ e \end{pmatrix}$
Le problème est ensuite réduit à trouver un mot de code de rang minimum r dans le code C_e. La modélisation de ce problème conduit les auteurs à résoudre un système d'un ensemble d'équations quadratiques en utilisant deux stratégies.

- Énumération de base : cette méthode a une complexité en $\mathcal{O}((k+r)^3 q^{(r-1)(m-r)+2})$.
- Énumération des coordonnées : la complexité de cette méthode est $\mathcal{O}((k+r)^3 r^3 q^{(r-1)(k+1)})$.

La meilleure complexité de ce type d'attaques est $\mathcal{O}((n-k)^3 k^3 q^{(r-1)\lfloor\frac{(k+1)m}{n}\rfloor})$, elle est introduite dans [33]. L'idée de leur méthode consiste à utiliser la notion de support d'un mot en métrique rang.
Pour des paramètres bien choisis pour notre système, nous pouvons rendre ces attaques, de complexité exponentielle.

6.5.2 Attaques algébriques

Plusieurs attaques algébriques ont été proposées contre les variantes du cryptosystème GPT. Parmi ces attaques, nous mentionnons :

★ L'attaque de Gibson (1995) [36]

★ L'attaque de Overbeck (2008) [77]

★ L'attaque de Hauteville et Tillich (2015) [43]

Gibson a proposé de résoudre le problème suivant pour effectuer sa cryptanalyse :
Problème : Étant donné (G_{pub}, t_1) la clé publique du cryptosystème GPT, trouver un triplet $(\tilde{G}, \tilde{S}, \tilde{X})$ tel que $G_{pub} = \tilde{S}\tilde{G} + \tilde{X}$.
On peut avoir une solution de ce problème en un temps de $\mathcal{O}((n-k)k^3q^{mt_1}$. L'attaque de Gibson ne marche pas sur notre schéma à cause de l'ajout d'une matrice de distorsion X.

Pour réaliser l'attaque de Overbeck [77], on construit la matrice G' de la manière suivante :

$$\begin{aligned}\tilde{G}^{pub} &= \begin{pmatrix} S & 0 & \dots & 0 \\ 0 & S^{[1]} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & S^{[n-k-1]} \end{pmatrix} \begin{pmatrix} X & G \\ X^{[1]} & G^{[1]} \\ \vdots & \vdots \\ X^{[n-k-1]} & G^{[n-k-1]} \end{pmatrix} P \\ &= S'(X'|G')P\end{aligned}$$

où $S^{[i]}$ est la matrice obtenue par application de la i-ème puissance de l'automorphisme de Frobenius $\theta^i : x \to x^{q^i}$ à tout les coefficients de S.
Loidreau, dans [53], a prouvé que cette attaque ne marche pas si $Rk(X'|G') = n + t_1 - 1$.
Dans notre cas, l'attaque de Overbeck [76] n'est pas applicable due au choix du rang t_1 de la matrice de distorsion X, qui est plus grand ou égal à $n-k$.

Récemment, Hauteville et Tillich [43] ont introduit une nouvelle attaque de récupération de clé contre le cryptosystème LRPC. Cette attaque est basée sur les techniques de pliage et de projection.

Définition 6.5.1. *(Code projeté) Soit C un code cellulaire d'indice l défini sur $\mathcal{R} = \mathbb{K}[X]/(f(X))$ et soit $g(X)$ un diviseur de $f(X)$ dans $\mathbb{K}[X]$. Le code cellulaire projeté $\bar{C}^g$ est obtenu par le visionnement d'un mot de code $c \in C$ comme un élément de $\mathcal{R}^l$: $c(c_1, \dots, c_l)$ et appliquer le morphisme surjectif*

$$\begin{aligned}\Pi : \quad \mathbb{K}[X]/(f(X)) &\to \mathbb{K}[X]/(g(X)) \\ a(X) &\mapsto \Pi(a(X)) = a(X)(mod\ g(X))\end{aligned}$$

à chaque entrée c_i pour $i = 1, \dots, l$.

Définition 6.5.2. *(Code plié) Soit C un code quasi-cyclique d'indice l et de longueur nl, soit m un diviseur de n. Son code plié d'ordre m est le code quasi-cyclique d'indice l et de longueur ml obtenu en faisant correspondre chaque mot de code $c = (c_0, \dots, c_{nl-1}) \in C$ au mot de code $c' = (c'_0, \dots, c'_{ml-1})$ où*

$$c'_i = \sum_{s=0}^{\frac{n}{m}-1} c_{an+b+sm} \quad /i = am + b \quad b \in \{0, \dots, m-1\}.$$

Le but de cette attaque est de trouver un mot de code de faible poids d dans un code projeté sur $\mathbb{K}[X]/(X^n-1)$, cela dépend de la factorisation de X^k-1 selon les étapes décrites dans [43].
Hauteville et al. [43] ont donné plusieurs exemples de factorisations où cette attaque ne fonctionne pas (pas vraiment efficace). Le point commun entre ces exemples est que m est premier. Mais en ce qui concerne notre système de chiffrement, nous avons choisi m pair, alors nous supposons actuellement que notre système est sécurisé contre ce type d'attaque.

6.5.3 Les paramètres du système

Dans la table 6.1 présentée ci-dessous, nous proposons des paramètres pour un niveau de sécurité de 80 pour notre système.
Cela montre que notre schéma possède une petite taille de clé par rapport à d'autres schémas.

m	n	k_1	k_2	t_1	taille de la clé	taille de la clé dans [53]
24	24	6	6	40	14976	18432
24	24	6	6	52	18432	21888

TABLE 6.1 – Comparaison de la taille de la clé publique (en bits) pour un niveau de sécurité de 80 bits.

Nous remarquons que la taille de la clé est grande mais inférieure à la taille des clés proposées dans les variantes du cryptosystème McEliece et certaines variantes du schéma GPT.

6.6 Conclusion

Dans ce chapitre, nous avons introduit un nouveau type de code que nous avons utilisé pour construire une nouvelle version du cryptosystème de type GPT qui est sécurisée face à toutes les attaques connues actuellement (jusqu'à la présentation de ces travaux à la conférence ICCC 2015).

Chapitre 7

Cryptosystème de McEliece basé sur la forme de Smith des codes Convolutifs

Ce chapitre a fait l'objet d'une publication dans le journal international "Cryptologia" [70].

Dans ce chapitre, nous proposons une nouvelle version du système de chiffrement de McEliece basé sur la forme Smith des codes convolutifs. Nous utilisons la forme Smith pour cacher une partie du code dans la matrice publique et nous laissons l'autre partie secrète. La partie secrète sera ensuite utilisée pour le déchiffrement. Nous cachons cette partie en multipliant à gauche par une matrice choisie aléatoirement et on rajoute une matrice aléatoire qui satisfait quelques conditions. Notre système a une petite taille de la clé publique par rapport au cryptosystème de McEliece original et résiste à l'unique attaque par décodage, contre la structure convolutionnelle présentée lors de la conférence PQCrypto 2013 par Landais et Tillich [50]. De plus, l'attaque par recherche exhaustive est infaisable contre notre système.

Une séquence d'un code convolutionnel peut être infinie. En pratique, les séquences d'entrée sont de longueur finie. Il existe trois méthodes possibles pour générer un code convolutif terminé : terminaison, troncature et le tail-biting. Dans notre travail, nous utilisons une construction tail-biting car le taux de code reste inchangé et tous les bits d'information ont la même quantité de protection contre les erreurs. Le code devient un (n, k) code linéaire .

Il existe de nombreux algorithmes pour le décodage des codes convolutionnels. Parmi eux, nous citons l'algorithme de Viterbi [94], l'algorithme de décodage par syndrome [82] et l'algorithme de décodage séquentiel [99].

7.1 La forme Smith d'un code convolutionnel

Soient $k \leq n$ deux entiers positifs et soit $G(D)$ une $k \times n$, $k \leq n$, matrice polynomiale binaire (i.e., $G(D) = (g_{ij}(D))$, où $g_{ij}(D) \in \mathbb{F}_2[D]$, $1 \leq i \leq k, 1 \leq j \leq n$) de rang r. Alors $G(D)$ peut être écrite sous la forme :

$$G(D) = A(D)\Gamma(D)B(D), \tag{7.1}$$

où $A(D)$ et $B(D)$ sont, respectivement, deux $k \times k$ et $n \times n$, matrices binaires polynomiales avec des déterminants unitaires, et $\Gamma(D)$ est une matrice de taille $k \times n$ telle que :

$$\Gamma(D) = \begin{pmatrix} \gamma_1(D) & & & & & & & \\ & \gamma_2(D) & & & & & & \\ & & \ddots & & & & & \\ & & & \gamma_r(D) & & & & \\ & & & & 0 & & & \\ & & & & & \ddots & & \\ & & & & & & 0 \ \ldots \ 0 & \end{pmatrix}, \tag{7.2}$$

La forme donnée en (7.1) est appelée la forme Smith de $G(D)$, et les éléments non nuls $\gamma_i(D) \in \mathbb{F}_2[D]$, $1 \leq i \leq r$, donnés en (7.2) sont appelés les facteurs invariants de $G(D)$.
Pour plus de détails sur les propriétés de la forme Smith d'un code convolutionnel, veuillez consulter [46].

7.2 Notre nouvelle variante

Dans cette section nous décrivons notre nouveau schéma de McEliece basé sur la forme Smith d'un code convolutionnel.

Génération des clés

i Sélectionner une matrice génératrice G d'un code convolutionnel de longueur n et de dimension k telle que $k \leq n$.

ii Ecrire cette matrice génératrice choisie sous sa forme Smith correspondante $G = A\Gamma B$.

iii Choisir aléatoirement une matrice inversible S de taille $k \times k$.

iv Générer une matrice aléatoire Z de taille $k \times n$ telle que $ZB = 0$.

v Pour les deux matrices $SA\Gamma$ et Z, rajouter r colonnes aux c_i positions de telle sorte que les lignes de la matrice $SA\Gamma \oplus Z$ soient de même poids. Nous notons les matrices obtenues, de taille $k \times (n + r)$, par U et $Z^{'}$.

vi Calculer la matrice publique $G^{'} = U \oplus Z^{'}$.

La clé publique est $(G^{'}, t)$ et la clé privée est $(A, S, G, B, \Gamma, Z, Z^{'})$.

Chiffrement

Pour envoyer un message m à quelqu'un, nous générons au hasard $e \in \mathbb{F}_2^{n+r}$ de poids de Hamming $\leq t$ (t représente la capacité de correction) et on calcule : $c = mG^{'} + e$

Déchiffrement

Pour déchiffrer le message, d'abord on élimine les r éléments qui se trouvent aux c_i positions du vecteur c. Nous notons $c^{'}$ le vecteur obtenu. Ensuite on calcule $y = c^{'}B = (mG^{'} + e^{'})B = m(SA\Gamma \oplus Z)B + e^{'}B = mSG + e^{'}B$. ($e' \in \mathbb{F}_2^n$ et $w(e^{'}) \leq w(e)$).
Après le décodage de la quantité $mSG + e^{'}B$ on obtient $y^{'} = mS$. Le texte clair est obtenu après le calcul de $y^{'}S^{-1}$.
Dans le processus de décodage, nous utilisons le décodage par syndrome [82] car il s'agit d'un problème NP-complet. En outre, il nécessite moins d'un tiers du matériel nécessaire à la mise en oeuvre du décodeur classique de Viterbi.

Pour assurer un décodage réussi dans l'étape de décryptage, nous utilisons la méthode présentée par Baldi et al. dans [1] et [4].
Nous écrivons la matrice B sous la forme $B = Q + R$, où Q est une matrice dense de taille $n \times n$ et R est une matrice creuse de taille $n \times n$. La matrice R est obtenue à partir de deux ensembles, $\mathscr{A}$ et $\mathscr{B}$, chacun contient w matrices de taille $z \times n$, $z \leq n$, $\mathscr{A} = \{\mathfrak{a}_1, \mathfrak{a}_2, \ldots, \mathfrak{a}_w\}$, $\mathscr{B} = \{\mathfrak{b}_1, \mathfrak{b}_2, \ldots, \mathfrak{b}_w\}$.
La matrice R est obtenue comme

$$R = \begin{pmatrix} \mathfrak{a}_1(D) \\ \mathfrak{a}_2(D) \\ \vdots \\ \mathfrak{a}_w(D) \end{pmatrix}^T \begin{pmatrix} \mathfrak{b}_1(D) \\ \mathfrak{b}_2(D) \\ \vdots \\ \mathfrak{b}_w(D) \end{pmatrix}$$

Nous nous concentrons sur deux cas distincts, les deux avec $w = 2$: $(i)\mathfrak{a}_1 = \mathfrak{a}, \mathfrak{a}_2 = 0$ et $(ii)\mathfrak{b}_2 = 1 + \mathfrak{b}_1$, où 0 et 1 représentent, respectivement, les zéros et les uns des $z \times n$ matrices.
Supposons maintenant qu'une contrainte soit imposée au vecteur e sous la forme :

$$\mathfrak{a}.e^T = 0. \tag{7.3}$$

Si l'équation (7.3) tient, pour les deux cas sur lesquels nous nous concentrons, la contribution due à Q devient :

$$e.Q = \begin{cases} 0 & \text{if} \quad \mathfrak{a} = \mathfrak{a}_1, \mathfrak{a}_2 = 0, \\ e.\mathfrak{a}_2^T & \text{if} \quad \mathfrak{b}_2 = 1 + \mathfrak{b}. \end{cases} \tag{7.4}$$

Notre objectif est de réduire la matrice eB à eR (R est creuse). Le Design de telles matrices est detaillé dans [1] et [4]. Comme conséquence de réduire eB à eR, Bob est capable de corriger le vecteur d'erreur résultant de eB durant le déchiffrement.

7.3 Analyse de la sécurité

7.3.1 Attaque par force brute

Dans notre système, la clé privée (A, G, B, Γ, Z) est obtenue de manière aléatoire. Dans le cas binaire, trouver la matrice Z nécessite $\mathcal{O}(2^{kn})$ opérations. De plus, le nombre de matrices inversibles dans $\mathbb{F}_2$ est $\prod_{i=1}^{k}(2^k - 2^{i-1})$. La dernière quantité est majorée par $2^{(k^2-k)/2}$. D'où la complexité de l'attaque par recherche exhaustive contre notre cryptosystème est $\mathcal{O}(2^{kn})$. Par conséquent, l'attaque par recherche exhaustive se fait en un temps exponentiel sur notre schéma.

7.3.2 Attaque par décodage

Le décodage par ensemble d'informations (ISD) est l'une des meilleures approches des attaques par décodage. Cette approche (ISD) a été introduite par Prange en 1962 [79], et la complexité de son algorithme est égale à $2^{0.0576n}$ qui a été ensuite abaissée par Stern à $2^{0.0557n}$. Plusieurs algorithmes de décodage ont été proposés pour améliorer l'algorithme de Stern [91]. Tout d'abord, Bernstein et al. [11] l'ont amélioré à $2^{0.0556n}$, puis May et al. [58] ont réduit le temps de fonctionnement de l'algorithme à $2^{0.0537n}$. A la conférence EUROCRYPT 2012, Becker et al. [7] ont introduit un autre algorithme ISD avec une complexité de $2^{0.0494n}$ et très récemment l'algorithme ISD le plus rapide a été présenté par Alexander May et Ilya Ozerov [59] à EUROCRYPT 2015, leur approche a une complexité de $2^{0.0473n}$.

Le cryptosystème basé sur les codes convolutionnels introduit par Londahl et Johansson dans [54] a été attaqué par Landais et Tillich [50]. Ils ont construit un algorithme ISD efficace pour attaquer la structure convolutionnelle. Ils ont prouvé qu'il est possible de récupérer complètement la structure du code convolutif. Ils ont annoncé que le schéma basé sur les codes convolutionnels peut devenir résistant à leur attaque en rajoutant au moins 140 colonnes aléatoires à la matrice publique.
Notre système est sécurisé contre l'attaque de Landais-Tillich parce que notre construction ne révèle aucune information sur les fréquences des positions des mots de code d'un certain poids w en utilisant l'algorithme de Dumer [21]. Dans notre cas, le code publique a le comportement d'un code MDPC car toutes les lignes de la matrice génératrice ont le même poids.

Dans [10], le problème du décodage des codes aléatoires est prouvé NP-complet, plus tard, Finiasz [26] a prouvé la NP-complétude du problème de décodage par syndrome d'un code de Goppa paramétrique.
Dans cette section, nous prouvons que le décodage des codes convolutifs aléatoires est un problème NP-complet.

Définition 7.3.1. *(Convolutional Syndrome Decoding Problem) Entrée : une matrice binaire H de taille $(n-k)\times n$, un vecteur s de $n-k$ bits et un poids w.*

Sortie : un vecteur e de longueur n de poids de Hamming ≤ w tel que $H.e^T = s$.

Pour prouver la NP-complétude du problème de décodage par syndrome des codes convolutifs, nous devons réduire notre problème à un autre appelé Three Dimensional Matching.

Three-Dimensional Matching (3DM)
Entrée : un sous-ensemble $U \subseteq T \times T \times T$ où T est un ensemble fini.
Propriété : il existe un ensemble $V \subseteq U$ tel que $|V| = |T|$ et pas deux éléments de V accepte toute coordonnée.

Proposition 7.3.1. *[10] Three-Dimensional Matching est un problème NP-complet.*

Le théorème suivant montre que la sécurité du schéma proposé est réduite à celle d'un problème difficile.

Théorème 7.3.1. *Le problème du décodage par syndrome des codes convolutifs est NP-Complet.*

Démonstration. On construit une réduction de notre problème à Three Dimensional Matching.
Soit A la matrice d'incidence de taille $|U| \times |3T|$ illustré dans la figure 7.1.

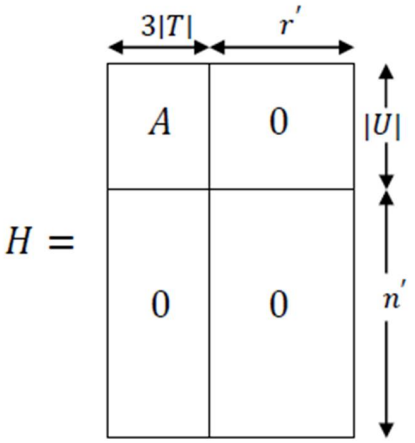

FIGURE 7.1 – La matrice H utilisée pour réduire three dimensional matching au problème du décodage par syndrome d'un code convolutionnel

La matrice A est complétée par des zéros afin que nous puissions obtenir les proportions du code convolutionnel pour corriger jusqu'à $|T|$ erreurs.
Supposons que nous ayons un algorithme en temps polynomial capable de résoudre n'importe quelle instance du problème CSD. Les entrées de l'algorithme sont H et $S = (1,\ldots,1,0,\ldots,0)$ contenant $3|T|$ uns suivis par r' zéros. Une solution de ce problème est quand $|T| = w$ ligne.
Un algorithme en temps polynomial pour CSD implique un algorithme en temps polynomial pour 3DM. D'après la proposition 7.3.1, 3DM est NP-complet, alors nous obtenons que le problème CSD est NP-complet. □

Nous avons montré que la sécurité du schéma proposé est réduite à celle d'un problème difficile, appelé problème de décodage par syndrome d'un code convolutif. Donc, notre système est OW-CPA (one-wayness against selected-plaintext attacks), mais la propriété OW-CPA n'est pas suffisante pour dire que le schéma est sécurisé.

7.3.3 Attaque exploitant la structure des codes convolutionnels

La structure du code convolutionnel utilisé dans notre cryptosystème est cachée par une matrice de brouillage et un vecteur aléatoire. Si la matrice B de la décomposition de Smith sera découverte par l'attaquant, une des informations importantes sera révélée, comme une matrice équivalente de la clé secrète G ou la matrice de contrôle de parité du code convolutif. La matrice B a la forme suivante :

$$B = \begin{pmatrix} G_b \\ (H^{-1})^T \end{pmatrix},$$

Comme B est inversible, un attaquant peut facilement déduire la matrice H de $B^{-1} = (G_b^{-1} H^T)$.
Pour éviter une attaque critique, nous laissons la matrice B secrète afin qu'elle ne fasse même pas partie de la construction du code publique. Cette matrice est utilisée uniquement dans l'étape de déchiffrement.
Beaucoup de systèmes basés sur des codes alternatifs sont brisés. Ces attaques utilisent le fait que la matrice sous-jacente d'un code alternatif est une matrice de Vandermonde. Dans notre cas, une matrice de contrôle de parité d'un code convolutif avec la structure Vandermonde n'existe pas. Cette structure existe dans un cas particulier de codes convolutionnels [40]. Donc, de telles attaques algébriques sont impossibles.
En outre, la sécurité de notre système de chiffrement est réduite à un autre problème difficile qui est le problème d'équivalence d'un code poinçonné. C'est un problème NP-complet [98], ce qui rend très difficile la cryptanalyse de notre système.

7.3.4 Attaque par renvoi de message

Considérons le cas que deux chiffrés $c_1 = mSGP + e_1$ et $c_2 = mSGP + e_2$ avec $e_1 \neq e_2$ sont produit par l'algorithme proposé. C'est ce qu'on appelle une condition de renvoi de message.
Pour un code de paramètres $n = 2048$, $k = 1536$ et $d_{free} = 81$, La probabilité que deux vecteurs d'erreur aient un 1 à la position $l \in \{1, \ldots, n\}$ est

$$Pr(e_1(l) = e_2(l)) = \left(\frac{40}{2048}\right)^2 \approx 0.0003814.$$

Soit $L_0 = \{l \in \{1, \ldots, n\} : c_1(l) \oplus c_2(l) = e_1(l) \oplus e_2(l)\} = 0$, $L_0 = \{l : e_1(l) = 0 = e_2(l\} \cup \{l : e_1(l) = 1 = e_2(l)\}$, $L_0 \approx \{l : e_1(l) = 0, e_2(l) = 0\}$, et $L_0 = \{l \in \{1, \ldots, n\} : c_1(l) \oplus c_2(l) = e_1(l) \oplus e_2(l) = 1\}$.

La probabilité que i coordonnées soient modifiées à la fois par e_1 et e_2 est

$$p_i = Pr(|\{l : e_1(l) = 1\} \cap \{l : e_2(l) = 1\}|) = \frac{\begin{pmatrix} 40 \\ i \end{pmatrix}\begin{pmatrix} 2048-40 \\ 40-i \end{pmatrix}}{\begin{pmatrix} 2048 \\ 40 \end{pmatrix}}$$

Par conséquent,

$$E(|L_1|) = \sum_{i=0}^{40}(80-2i)p_i \approx 78.$$

Par exemple, si $|L_1| = 78$, il s'ensuit que $|L_0| = n - |L_1| = 1970$ et seulement 7 entrées de L_0 sont affectées par une erreur. La probabilité de deviner 1536 colonnes correctes de celles indexées par L_0 est

$$\frac{\begin{pmatrix} 1970-7 \\ 1536 \end{pmatrix}}{\begin{pmatrix} 1970 \\ 1536 \end{pmatrix}} \approx 0.00002424.$$

Par conséquent, après environ 41240 essais, un mot de code est obtenu, qui peut être décodé à l'aide de la matrice génératrice publique.
La table 7.1 présente la probabilité de deviner des k colonnes correctes de celles indexées par L_0 par rapport au nombre d'erreurs dans le cryptosystème. Comme prévu, augmenter le nombre d'erreurs améliore la sécurité. Ces résultats sont meilleurs que ceux obtenus pour la même attaque [14] sur le cryptosystème McEliece donné dans [60].
Les résultats du tableau 1 indiquent qu'il n'est pas nécessaire d'utiliser un vecteur d'erreur d'un poids important dans l'étape de chiffrement.

Nombre d'erreurs	La probabilité de deviner k colonnes correctes de celles indexées par L_0	Nombre d'essais
1	0.2203045685	5
2	0.04844686550	21
5	0.0005096619	1962
7	0.0000242481	41240
10	$2.482039676\ 10^{-7}$	4028944
15	$1.154239440\ 10^{-10}$	8663713661

TABLE 7.1 – La probabilité de deviner k colonnes correctes de celles indexées par L_0

7.3.5 Paramètres proposés

Dans cette partie, nous proposons des paramètres pour notre schéma, pour les deux niveaux de sécurité standards, 80-bit et 128-bit.

Pour une sécurité de 80 bits, on propose un [2048, 1536] code convolutionnel avec 40 erreurs rajoutées.
Pour une sécurité de 128 bits, on propose un [3400, 2550] code convolutionnel avec 68 erreurs rajoutées.
La comparaison des valeurs de la taille des clés des paramètres suggérés avec ceux du cryptosystème de McEliece basé sur les codes de Goppa [11] et le système de McEliece basé sur les codes de Reed-Solomon généralisés (GRS) [1] est donnée dans la table 7.2.

Niveau de sécurité	Notre schéma	McEliece basé sur les codes de Goppa [11]	McEliece basé sur les codes GRS [1]
80	98304	460687	199899
128	326560	1537536	540267

TABLE 7.2 – Comparaison de la taille des clés (bits)

7.4 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle variante du cryptosystème de McEliece basé sur la Forme Smith des codes convolutionnels et inspirée du problème de décodage par syndrome d'un code convolutif et du problème d'équivalence d'un code poinçonné, qui sont tous les deux NP-complets. Notre approche résiste à l'unique attaque existante contre les codes convolutifs proposée par Landais et Tillich et toutes les attaques structurelles connues. En outre, il a une clé de petite taille par rapport à d'autres systèmes.

Conclusion générale

Cette thèse m'a donnée l'occasion de voir de près l'application et l'utilité des codes correcteurs d'erreurs en cryptographie à clé publique. Nous avons fait le design et l'analyse de plusieurs systèmes cryptographiques post-quantiques basés sur les codes. Dans nos constructions, nous avons utilisé les codes convolutionnels, les codes quasi-cycliques LDPC et MDPC, les codes LRPC et les codes en métriques rang (convolutionnels et ceux de Gabidulin).

Actuellement, plusieurs constructions de primitives cryptographiques basées sur les codes ont été inspirées de celles basées sur la théories des nombres. Mais ça ne marche pas toujours, comme on le souhaite. Par exemple, jusqu'à présent il n'existe pas des signatures basées sur les codes qui ont été mis en pratique. L'été passé (Août 2017), le premier système d'IBE (Identity Based Encryption) basé sur les codes a été présenté à la prestigieuse conférence internationale Crypto'17. Ce système est construit à partir des codes en métrique rang, donc la construction d'un tel schéma cryptographique à partir des codes en métrique de Hamming reste un problème ouvert. A partir des schémas d'IBE, on peut parfois tirer un système d'ABE (Attribute Based Encryption), donc après l'existence d'un seul schéma d'IBE, c'est l'occasion de construire le premier schéma d'ABE basé sur les codes, peu importe la métrique utilisée.

A la fin, on peut dire qu'il y a beaucoup de choses qui reste à faire dans le domaine de la cryptographie basée sur la théorie des codes correcteurs d'erreurs.

ANNEXE A

NOTIONS DE COMPLEXITÉ

Le temps d'exécution d'un programme dépend du type d'ordinateur utilisé ; et du langage utilisé (représentation des données). La complexité d'un algorithme est le temps de calcul nécessaire pour obtenir une réponse valide.

Définition A.0.1. *Soient f et g deux fonctions à variable entière et à valeurs positives.*
$f(n) = \mathcal{O}(g(n))$ s'il existe une constante $c > 0$ et un entier n_0 tels que $\forall n \geq n_0, 0 \leq f(n) \leq cg(n)$.
On dit que g domine f ou que f ne croit pas plus vite que g multiplié par une constante.

Il y a trois type de complexité :

- La complexité dans le pire des cas : c'est le temps d'exécution maximum, dans le cas le plus défavorable.
- La complexité dans le meilleur des cas : c'est le temps d'exécution minimum, dans le cas le plus favorable (en pratique, cette complexité n'est pas très utile).
- La complexité en moyenne : elle représente la moyenne des temps d'exécution.

Un algorithme est dit polynômiale si sa complexité dans le pire des cas est de la forme $\mathcal{O}(n^k)$ où k est une constante.
Si sa complexité ne peut être majorée par un polynôme, on dit que l'algorithme est à complexité exponentielle.
Un algorithme est dit quasi (ou sous)-exponentielle si sa complexité est une fonction de la forme $e^{\mathcal{O}(n)}$.

A.0.1 Classes de complexité

Définition A.0.2. *Un problème de décision est un problème dont la réponse est "oui" ou "non".*

La classe de complexité P est l'ensemble des problèmes de décision résoluble en temps polynômiale.
La classe de complexité NP est l'ensemble des problèmes de décision pour lesquels la réponse "oui" peut être vérifiée en temps polynômiale moyennant une information supplémentaire appelée "certificat".
La classe de complexité Co-NP est l'ensemble des problèmes de décision pour lesquels la réponse "non" peut être vérifiée en temps polynômial moyennant un "certificat" approprié.

A.0.2 Réduction et NP-complétude

Soient D_1 et D_2 deux problèmes de décision. On dit que D_1 se réduit polynomialement à D_2, et on note $D_1 \leq_p D_2$, s'il existe une fonction $f : \Sigma^* \longrightarrow \Sigma^*$ calculable en temps polynomial telle que $x \in D_1 \Leftrightarrow f(x) \in D_2$. La fonction f est appelée réduction de D_1 à D_2.

Définition A.0.3. *Soit C une classe de complexité. On dit qu'un langage D_1 est C-complet (pour la réduction $\leq_p$) si les deux conditions suivantes sont satisfaites :*

1. $D_1 \in C$.
2. *pour tout langage $D_2 \in C$, $D_2 \leq_p D_1$.*

Si seule la condition 2 est satisfaite, on dit que D_1 est C-difficile.

Deux problèmes de décision D_1 et D_2 sont polynômialement équivalents s'ils sont(mutuellement) polynômialement réductibles l'un à l'autre , i.e, $D_1 \leq_p D_2$ et $D_2 \leq_p D_1$.

Problème NP-complet :

Un problème de décision est dit NP-complet si :

1. $D \in$ NP.
2. $D_1 \leq_p D$ pour tout $D_1 \in$ NP.

Les problèmes NP-complets sont les problèmes les plus difficiles dans NP.

Problème NP-Dur

Un problème D est NP-Dur s'il existe un problème R qui est NP-Complet et qui se réduit polynômialement à D ($R \leq_p D$).

Bibliographie

[1] Baldi, M. , M. Bianchi, F. Chiaraluce, J. Rosenthal, and D. Schipani. Enhanced public key security for the McEliece cryptosystem. Journal of Cryptology, Springer, vol. 29, pp. 1-27, 2016.

[2] M. Baldi, F. Chiaraluce, R. Garello, and F. Mininni, "Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem," in *Proc. IEEE Int. Conf. on Commun.*, Glasgow, UK, Jun. 2007, pp. 951–956.

[3] Baldi, M. , M. Bodrato, and G. F. Chiaraluce. 2008. A new analysis of the McEliece cryptosystem based on QC-LDPC codes. Security and Cryptography for Networks (SCN), Lecture Notes in Computer Science, Springer, 5229 : 246-262.

[4] Baldi. M. , F.Chiaraluce, J. Rosenthal and D. Schipani. 2015. A modification of the McEliece cryptosystem based on. Generalized Reed-Solomon codes. MEGA 2015, Italy. http://mega2015.science.unitn.it/POSTERS/BaldiChiaraluceRosenthalSchipani.pdf.

[5] M. Bardet, J. Chaulet, V. Dragoi, A. Otmani, J-P. Tillich. Cryptanalysis of the McEliece public key cryptosystem based on polar codes. In : International Workshop on Post-Quantum Cryptography. Springer International Publishing, 2016. p. 118-143.

[6] M. Bardet, V. Dragoi, J.-G. Luque, and A. Otmani, "Weak keys for the quasi-cyclic MDPC public key encryption scheme," *Progress in Cryptology, Proc. AFRICACRYPT*, Lecture Notes in Computer Science, vol. 9646, Springer, Berlin, pp. 346–367, 2016.

[7] A. Becker, A. Joux, A. Mayand, and A. Meurer, "Decoding random binary linear codes in $2^{(n/20)}$: How 1 + 1 = 0 improves information set decoding," in *Advances in Cryptology, Proc. EUROCRYPT*, Lecture Notes in Computer Science, vol. 7237, Springer-Verlag, Berlin, pp. 520–536, 2012.

[8] M. Bellare, P. Rogaway, "Entity authentication and key distribution", CRYPTO'93, 1993.

[9] Berger, T. P., P-L. Cayrel, P. Gaborit and A. Otmani. 2009. Reducing key length of the McEliece cryptosystem, Progress in Cryptology - AFRICACRYPT 2009, Springer, 5580 : 77-97.

[10] E. R. Berlekamp, R. J. McEliece, and H. C. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inform. Theory*, vol. 24, no. 3, pp. 384–386, May 1978.

[11] D. J. Bernstein, T. Lange, and C. Peters, "Attacking and defending the McEliece cryptosystem," in *Proc. Int. Workshop on Post-Quantum Cryptography*, Lecture Notes in Computer Science, vol. 5299, Springer-Verlag, Berlin, pp. 31–46, 2008.

[12] R.C. Bose and D.K. Ray-Chaudhuri, On a class of error correcting binary group codes. Information and control, 3(1), pp. 68-79, 1960.

[13] K.A. Bush, Orthogonal arrays of index unity. Ann. Math. Statist., 23 , pp. 426-434, 1952.

[14] T. Burton, "Failure of the McEliece public-key cryptosystem under message-resend and related-message attack," in *Advances in Cryptology, Proc. CRYPTO*, Lecture Notes in Computer Science, vol. 1294, Springer-Verlag, Berlin, pp. 213–220, 1997.

[15] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In Symposium on Theory of Com- puting, STOC'98, pp. 209-218. ACM, 1998

[16] A. Canteaut and F. Chabaud, "A new algorithm for finding minimum-weight words in a linear code : Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511," *IEEE Trans. Inform. Theory*, vol. 44, no. 1, pp. 367–378, Jan. 1998.

[17] P-L Cayrel, P. Véron and S-M El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. SAC'10, pp. 171-186, LNCS, vol. 6544, Springer, 2010.

[18] F. Chabaud, J. Stern. The Cryptographic Security of the Syndrome Decoding Problem for Rank Distance Codes. ASIACRYPT. pp 368-381.(1996).

[19] J. Chaulet and N. Sendrier, "Worst case QC-MDPC decoder for McEliece cryptosystem," *Proc. IEEE Int. Symp. Inform. Theory*, Barcelona, Spain, Jul. 2016, pp. 1366–1370.

[20] D. Coppersmith, H. Krawczyk, and Y. Mansour, "The shrinking generator," *Advances in Cryptology, Proc. CRYPTO*, Lecture Notes in Computer Science, vol. 773, Springer-Verlag, Berlin, pp. 22–39, 1994.

[21] A. Couvreur, A. Otmani, J. Tillich, and V. Gauthier-Umana. A polynomial-time attack on the BBCRS scheme. In J. Katz, editor, Public-Key Cryptography - PKC 2015, vol. 9020 of LNCS, pp. 175-193. Springer, 2015.

[22] P. Delsarte. Bilinear forms over a finite field, with applications to coding theory, J. Combin. Theory Ser. A vol. 25, pp. 226-241, 1978.

[23] W. Diffie and M. E. Hellman. New directions in cryptography. IEEE Trans. Inform. Theory, 22(6), pp 644-654, 1976.

[24] Faugère, J-C., A. Otmani, L. Perret, and J-P. Tillich. 2010. Algebraic cryptanalysis of McEliece variants with compact keys, Advances in Cryptology - EUROCRYPT 2010, Lecture Notes in Computer Science, Springer, 6110 : 279-298.

[25] A. Fiat and A. Shamir, How To Prove Yourself : Practical Solutions to Identification and Signature Problems, in Advances in Cryptology, CRYPTO 86, Lecture Notes in Computer Science, Springer Berlin Heidelberg, vol. 263, pp. 186-194, 1986.

[26] Finiasz, M. 2009. NP-completeness of certain sub-classes of the syndrome decoding problem, CoRR, abs/0912.0453.

[27] E.M. Gabidulin. Attacks and counter-attacks on GPT public key cryptosystem. Designs Codes and Cryptography. pp. 171-177. Springer, Netherlands(2008).

[28] E.M. Gabidulin. Theory of codes with maximum rank distance. Probl. Inf. Transm., vol. 21, 1-12 (1985).

[29] E. M. Gabidulin and A.V. Ourivski. Modified GPT PKC with Right Scrambler. Proceedings of the 2nd International workshop on Coding and Cryptography, WCC 2001, pp. 233-242, 2001, France.

[30] E.M. Gabidulin, A.V. Paramonov , O.V. Tretjakov. Ideals over a non-commutative ring and their application in cryptology. In : Davies D.W. (ed.) Advances in Cryptology-Eurocrypt '91 LNCS, No. 547, pp. 482-489. Springer, Berlin (1991).

[31] Gaborit, P. 2005. Shorter keys for code based cryptography, International Workshop on Coding and Cryptography (WCC 2005), Norway, 81-91.

[32] P. Gaborit, G. Murat, O. Ruatta, G. Zémor. Low rank parity-check codes and their application to cryptography. WCC 2013. Bergen, Norway.(2013).

[33] P. Gaborit, O. Ruatta and J. Schrek . On the complexity of the Rank Syndrome Decoding problem. IEEE Transactions on Information Theory, vol. 62, pp. 1006-1019, 2016.

[34] Gaborit, P. Shorter keys for code based cryptography, International Workshop on Coding and Cryptography (WCC 2005), Norway, pp. 81-91, 2005.

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

[36] J. K. Gibson. Severely denting the Gabidulin version of the McEliece public key cryptosystem. Design Codes and Cryptography, 6(1), pp. 37-45 (1995).

[37] J. K. Gibson. The security of the Gabidulin publickey cryptosystem. In : U.M. Maurer, ed., Advances in Cryptology - EUROCRYPT'96, LNCS vol. 1070, pp. 212-223. Springer, Heidelberg (1996).

[38] M. R. Garey and D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, First Edition, A Series of Books in the Mathematical Sciences. W. H. Freeman and Co., San Francisco, Calif., 1979.

[39] S. Goldwasser, S. Micali, C. Rackoff. The knowledge complexity of interactive proof systems. Proc. 17th ACM Symp. Theory Computing, 291-304, 1985.

[40] Gluesing-Luerssen, H., and B. Langfeld. 2006. A class of one-dimensional MDS convolutional codes, Journal of Algebra and Its Applications, 5 : 505-520.

[41] Q. Guo, T. Johansson, and P. Stankovski, "A key recovery attack on MDPC with CCA security using decoding errors," *Advances in Cryptology, Proc. ASIACRYPT*, Lecture Notes in Computer Science, vol. 10031, Springer-Verlag, Berlin, pp. 789–815, 2016.

[42] S. Harari : A New Authentication Algorithm, Coding Theory and Applications, LNCS Vol.388, pp.204-211, Springer, 1989.

[43] A. Hauteville, J-P. Tillich. New algorithms for decoding in the rank metric and an attack on the LRPC cryptosystem. IEEE International Symposium on Information Theory - ISIT 2015, Hong Kong, China, pp. 2747 - 2751, (2015)

[44] M. Hell and T. Johansson, "Two new attacks on the self-shrinking generator," *IEEE Trans. Inform. Theory*, vol. 52, no. 8, pp. 3837–3843, Aug. 2006.

[45] A. Hocquenghem, Codes correcteurs d'erreurs. Chiffres, 2(2), pp. 147-156, 1959.

[46] R. Johannesson and K. S. Zigangirov. Fundamentals of convolutional coding, USA, IEEE Press, 2nd Edition, 2015.

[47] G.A. Karpunin, "On the key space of the McEliece cryptosystem based on binary Reed-Muller codes," *Disc. Math. Applic.*, vol. 14, no. 3, pp. 257–262, Jul. 2004.

[48] M. Krause, "BDD-based cryptanalysis of keystream generators," in *Advances in Cryptology, Proc. EUROCRYPT*, Lecture Notes in Computer Science, vol. 2332, Springer-Verlag, Berlin, pp. 222–237, 2002.

[49] A. Kanso, "Modified self-shrinking generator," *Computers Elect. Eng.*, vol. 36, no. 5, pp. 993–1001, Sep. 2010.

[50] Landais, G. and J-P. Tillich. 2013. An Efficient Attack of a McEliece Cryptosystem Variant Based on Convolutional Codes, PQCrypto 2013, Lecture Notes in Computer Science, Springer, 7932 : 102-117.

[51] Li, Y.X., R.H. Deng, and X.M. Wang. 1994. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems, IEEE Transactions on Information Theory, 40 : 271-273.

[52] P. Loidreau. Etude et optimisation de cryptosystèmes à clé publique fondés sur la théorie des codes correcteurs. Thèse, 2001.

[53] P. Loidreau. Métrique rang et cryptographie, HDR thesis. France(2007).

[54] Londahl, C. and T. Johansson. 2012. A new version of McEliece PKC based on convolutional codes, Information and Communications Security - ICICS 2012, Lecture Notes in Computer Science, Springer, 7618 : 461-470.

[55] C. Löndahl, T. Johansson, M. K. Shooshtari, M. Ahmadian-Attari, and M. R. Aref, "Squaring attacks on McEliece public-key cryptosystems using quasi-cyclic codes of even dimension," *Des. Codes Crypt.*, vol. 80, no. 1, pp. 359–377, Aug. 2016.

[56] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[57] M. Marazin, R. Gautier and G. Burel, Algebraic method for blind recovery of punctured convolutional encoders from an erroneous bitstream, IET Signal Proces. 2011.

[58] A. May, A. Meurer, and E. Thomae, "Decoding random linear codes in $\mathcal{O}(2^{0.054n})$," in *Advances in Cryptology, Proc. ASIACRYPT*, Lecture Notes in Computer Science, vol. 7073, Springer-Verlag, Berlin, pp. 107–124, 2011.

[59] A. May and I. Ozerov, "On computing nearest neighbors with applications to decoding of binary linear codes," *Advances in Cryptology, Proc. EUROCRYPT*, Lecture Notes in Computer Science, vol. 9056, Springer, Berlin, pp. 203–228, 2015.

[60] R.J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *DSN Progress Report*, 42-44, pp. 114–116, 1978.

[61] W. Meier and O. Statfelbach, "The self-shrinking generator," in *Advances in Cryptology, Proc. EUROCRYPT*, Lecture Notes in Computer Science, vol. 950, Springer-Verlag, Berlin, pp. 205–214, 1995.

[62] L. Minder and A. Shokrollahi, "Cryptanalysis of the Sidelnikov cryptosystem," in *Advances in Cryptology, Proc. EUROCRYPT*, Lecture Notes in Computer Science, vol. 4515, Springer-Verlag, Berlin, pp. 347–360, 2007.

[63] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto, "MDPC-McEliece : New McEliece variants from moderate density parity-check codes," in *Proc. IEEE Int. Symp. Inform. Theory*, Istanbul, Turkey, pp. 2069–2073, Jul. 2013.

[64] R. Misoczki and P. S. L. M. Barreto, "Compact McEliece keys From Goppa codes," in *Proc. Selected Areas in Cryptography*, Lecture Notes in Computer Science, vol. 5867, Springer-Verlag, Berlin, pp. 376–392, 2009.

[65] C. Monico, J. Rosenthal, and A. Shokrollahi, "Using low density parity check codes in the McEliece cryptosystem," in *Proc. IEEE Int. Symp. Inform. Theory*, Sorrento, Italy, p. 215, Jun. 2000.

[66] H. Moufek and K. Guenda, McEliece cryptosystem based on punctured convolutional codes and the pseudo-random generators, the 20th Conference on Applications of Computer Algebra (ACA), New York, USA, Jul. 2014.

[67] H. Moufek and K. Guenda, New Variant of the McEliece Cryptosystem. In :4th International Castle Meeting, Palmela Castle, Portugal, 2014. Coding Theory and Applications, pp. 291-296. Springer, 2015.

[68] H. Moufek, R. Mahdjoubi, P-L. Cayrel, and K. Guenda, New GPT cryptosystem based on the (u, u+ v)-construction codes. In International Conference on Coding theory and Cryptography ICCC2015, Algiers, Algeria, 2015.

[69] H. Moufek and K. Guenda, New Code Based Identification Scheme, 7th Seminar on Detection Systems : Architectures and Technologies (DAT 2017). IEEE, pp. 1-4, IEEE, Algiers, Algeria, 2017.

[70] H. Moufek and K. Guenda, New Variant of the McEliece Cryptosystem Based on Smith Form of Convolutional Codes. Cryptologia, 2017. (à paraître)

[71] H. Moufek, K. Guenda and T.A. Gulliver, A New Variant of the McEliece Cryptosystem Based on QC-LDPC and QC-MDPC Codes. IEEE Communications Letters, vol. 21, no 4, pp. 714-717, IEEE, 2017.

[72] D. E. Muller, Application of boolean algebra to switching circuit design and to error detection, Electronic Computers, Transactions of the I.R.E. Professional Group on EC-3, no. 3, pp. 6-12, 1954.

[73] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. Prob. Contr. Inform. Theory 15(2), pp.157-166 (1986).

[74] A.V. Ourivski, T. Johansson. New technique for decoding codes in the rank metric and its cryptography applications. Problems of Information Transmission 38(3), pp. 237-246 (2002)

[75] A. Otmani, J. P. Tillich, and L. Dallot, "Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes," *Math. Computer Sci.*, vol. 3, no. 2, pp. 129–140, Apr. 2010.

[76] R. Overbeck. A new structural attack for GPT and variants. In Proc. of Mycrypt 2005, volume 3715 of LNCS, pages 50-63. Springer Verlag (2005).

[77] R. Overbeck. Structural Attacks for Public Key Cryptosystems based on Gabidulin Codes. J. Cryptology, 21(2). pp. 280-301 (2008).

[78] E. Prange, "Cyclic Error-Correcting Codes in Two Symbols," Air Force Cambridge Research Center , Cambridge, MA, Tech. Rep. AFCRC-TN-57-103, 1957

[79] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Trans.*, vol. 8, no. 5, pp. 5–9, Sep. 1962.

[80] I. Reed, A class of multiple-error-correcting codes and the decoding scheme, Information Theory, Transactions of the IRE Professional Group on 4, no. 4, pp. 38-49, 1954.

[81] I. S. Reed and G. Solomon, Polynomial codes over certain

nite fields, SIAM J. Appl. Math., vol. 8, pp. 300-304, 1960.

[82] Schalkwijk, J. P. M., A. J. Vinck and K. A. Post. 1978. Syndrome decoding of rate-k/n convolutional codes, IEEE Transactions on Information Theory, 24 : 553-562.

[83] M. K. Shooshtari, M. Ahmadian-Attari, T. Johansson, and M. R. Aref, "Cryptanalysis of McEliece cryptosystem variants based on quasi-cyclic low-density parity check codes," *IET Inform. Sec.*, vol. 10, no. 4, pp. 194–202, Jul. 2016.

[84] P.W. Shor, Algorithms for quantum computation : discrete logarithms and factoring. In : 35th Annual Symposium on Foundations of Computer Science, 20-22 Nov 1994, Santa Fe, pp. 124-134. IEEE Press, New York (1994).

[85] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, vol. 26, pp. 1484-1509, 1997.

[86] S.R. Shrestha, Y.-S.Kim, New McEliece cryptosystem based on polar codes as a candidate for post-quantum cryptography. In : 2014 14th International Symposium on Communications and Information Technologies (ISCIT), pp. 368-372. IEEE (2014)

[87] V. M. Sidelnikov, "A public-key cryptosytem based on Reed-Muller codes," *Discr. Math. Applic.*, vol. 4, no. 3, pp. 191–207, Jan. 1994.

[88] Sidelnikov, V.M. and S. Shestakov. 1992. On the insecurity of cryptosystems based on generalized Reed-Solomon codes, Discrete Mathematics and Applications, 1 : 439-444.

[89] D. Silva, F.R. Kschischang. Fast Encoding and Decoding of Gabidulin Codes. Proc. IEEE Int. Symp. Inf. Theory, pp.2858-2862, Korea (2009).

[90] J. Stern, A new identification scheme based on syndrome decoding. CRYPTO'93. LNCS, vol. 773, pp. 13-21. Springer, Heidelberg (1994)

[91] Stern, J. 1998. A method for finding codewords of small weight, Coding Theory and Applications, Lecture Notes in Computer Science, Springer, 388 : 106-113.

[92] V. Tomas, J. Rosenthal and R. Smarandache. Decoding of Convolutional Codes Over the Erasure Channel. IEEE Trans. on inf. theory. 58(1) (2012).

[93] P. Véron. Improved identification schemes based on error-correcting codes. Appl. Algebra Eng. Commun. Comput., 8(1), pp. 57-69, Springer, 1996.

[94] Viterbi, A. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory, 13 : 260-269.

[95] A. Wachter-Zeh. Decoding of Block and Convolutional Codes in Rank Metric. Thesis ,University of Rennes 1. France (2013).

[96] A. Wachter-Zeh, V. Afanassiev, V. Sidorenko. Fast decoding of Gabidulin codes. Design Codes Cryptogr., vol. 66(1), pp.57 -73 (2013).

[97] A. Wachter-Zeh, V. Sidorenko, M. Bossert, V. Zyablov. On (Partial) Unit Memory Codes Based on Gabidulin. Codes. Problems of Information Transmission. 47(2), pp. 117-129 (2011).

[98] C. Wieschebrink. Two NP-complete problems in coding theory with an application in code based cryptography. IEEE International Symposium on Information Theory, pp. 1733-1737, 2006.

[99] Wozencraft, J. M. and B. Reiften. 1961. Sequential Decoding, M.I.T. Press and John Wiley and Sons.

[100] E. Zenner, M. Krause and S. Lucks, improved cryptanalysis of the self-shrinking generator, *Information Security and Privacy*, *LNCS*, **2119** (2001), 21–35.

[101] B. Zhang and D. Feng, "New guess-and-determine attack on the self-shrinking generator" in *Advances in Cryptology*, *Proc. ASIACRYPT*, Lecture Notes in Computer Science, vol. 4284, Springer-Verlag, Berlin, pp. 54–68, 2006.