Thèse de doctorat

Université de Limoges
École doctorale S2IM
Faculté des Sciences et Techniques

Thèse pour obtenir le grade de
Docteur de l'Université de Limoges

Discipline : Informatique

Présentée et soutenu par
Adrien Hauteville

Le 04 décembre 2017

Nouveaux protocoles et nouvelles attaques pour la cryptologie basée sur les codes en métrique rang.

Thèse dirigée par :
Philippe Gaborit, Professeur des universités, Université de Limoges
Olivier Ruatta, Maître de conférences, Université de Limoges
Jean-Pierre Tillich, Directeur de recherche, INRIA, Paris

JURY :

Président du jury
M. Nicolas Sendrier, Directeur de recherche, INRIA, Paris

Rapporteurs
M. Ayoub Otmani, Professeur des universités, Université de Rouen
M. Gilles Zémor, Professeur des universités, Bordeaux

Examinateurs
M. Pierre Loidreau, Directeur de recherche, IRMA, Rennes
M. Carlos Aguilar-Melchor, Maître de conférence, INPT, Toulouse

Remerciements

Je voudrais d'abord remercier mes directeurs de thèse Philippe Gaborit, Olivier Ruatta et Jean-Pierre Tillich pour la confiance et le soutien qu'ils m'ont apportés tout au long de cette thèse. J'ai beaucoup appris sur le métier de chercheur grâce à eux.

Je remercie ensuite les membres de mon jury de thèse qui m'ont fait part de remarques pertinentes pour la rédaction de cette thèse et pour la suite de mes travaux.

Je tiens à remercier tous mes amis et collègues doctorants pour la bonne ambiance et les discussions pas toujours sérieuses qui ont accompagné ces trois année de thèse.

Enfin, je souhaite remercier toute ma famille et en particulier mes parents pour leur soutien durant cette thèse.

Table des matières

Notations

Dans tout le document, nous utiliserons les notations suivantes :

- $\mathbb{N}$: ensemble des entiers naturels $\{0, 1, 2, \ldots\}$.
- $\mathbb{Z}$: ensemble des entiers positifs et négatifs $\{\ldots, -1, 0, 1, \ldots\}$.
- $\mathbb{F}_q$: corps fini à q éléments.
- $\mathbb{F}_{q^m}$: corps fini à q^m éléments vu comme une extension de degré m de $\mathbb{F}_q$.
- $\mathbb{F}_q^n$: $\mathbb{F}_q$-espace vectoriel de dimension n.
- $\mathbb{F}_q^{m\times n}$: ensemble des matrices de m lignes et n colonnes à coefficients dans $\mathbb{F}_q$.

Les vecteurs sont représentés par des lettres minuscules en gras et sont par défaut écrits en ligne. Exemple : $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$.

Les matrices sont représentées par des lettres majuscules en gras. Exemple : $\boldsymbol{M} \in \mathbb{F}_q^{m\times n}$.

- la matrice identité de taille n est noté $\boldsymbol{I}_n$.
- s'il n'y a pas ambiguïté sur les dimensions des matrices, la matrice nulle est notée $\mathbf{0}$.
- la transposée de la matrice $\boldsymbol{M}$ est notée $\boldsymbol{M}^T$.
- la concaténation horizontale de deux matrices $\boldsymbol{A}$ et $\boldsymbol{B}$ (quand elle est possible) est notée $(\boldsymbol{A}|\boldsymbol{B})$.
- soient $x_1, \ldots, x_n$ n éléments d'un $\mathbb{F}_q$-espace vectoriel E, le sous-espace vectoriel de E engendré par ces élément est noté $\langle x_1, \ldots, x_n\rangle_{\mathbb{F}_q}$ ou simplement $\langle x_1, \ldots, x_n\rangle$ s'il n'y a pas ambiguïté.

Nous rappelons ici les notations de Landau pour représenter les ordres de grandeur entre fonctions :

- $f = o\big(g\big)$: asymptotiquement, $|f(n)| \leqslant \varepsilon|g(n)|$ pour tout $\varepsilon > 0$. On dit que f est négligeable devant g.
- $f = \mathcal{O}\big(g\big)$: asymptotiquement, il existe $k > 0$ tel que $|f(n)| \leqslant k|g(n)|$. On dit que f est bornée par g.
- $f = \Theta\big(g\big)$: asymptotiquement, il existe $k_1, k_2 > 0$ tels que $k_1|g(n)| \leqslant |f(n)| \leqslant k_2|g(n)|$. On dit que f est de l'ordre de g.

Le nombre de sous-espaces vectoriels de dimension d d'un espace vectoriel de dimension n sur $\mathbb{F}_q$ est donné par le binôme de Gauss :

$$\begin{bmatrix} n \\ d \end{bmatrix}_q = \prod_{i=0}^{d-1} \frac{q^n - q^i}{q^d - q^i} = \Theta\big(q^{d(n-d)}\big)$$

Par défaut, la fonction $\log$ représente le logarithme en base 2. On notera la partie entière inférieure (respectivement supérieure) d'un réel x par $\lfloor x \rfloor$ (respectivement $\lceil x \rceil$).

Première partie

Introduction à la Cryptologie et à la Métrique Rang

Chapitre 1

Introduction

La cryptologie désigne de manière générale tout ce qui concerne la sécurité et la confidentialité des communications et des informations. Son domaine ne se limite pas seulement aux domaines du chiffrement ou de la cryptanalyse, mais aussi à la signature électronique, à l'authentification, au vote électronique, ou à tout protocole nécessitant de sécuriser des données.

La cryptographie (et son opposée la cryptanalyse) est une branche importante de la cryptologie. La cryptographie concerne tout ce qui a trait au chiffrement des données et des communications, c'est-à-dire le fait de rendre inintelligibles ces données aux personnes ne possédant pas de clés de déchiffrement. La cryptanalyse, au contraire, désigne toute méthode cherchant à extraire de l'information de données chiffrées, sans en posséder la clé.

La cryptographie se divise en deux grandes familles. La première (historiquement) est la cryptographie à clé secrète, aussi appelée cryptographie symétrique. Dans cette dernière, la clé de chiffrement et de déchiffrement est identique. Cela implique que les utilisateurs doivent au préalable s'échanger la clé secrète avant de pouvoir communiquer. En revanche, les algorithmes à clé secrète sont extrêmement rapides.

La seconde famille est la cryptographie à clé publique, aussi appelée cryptographie asymétrique. Dans ce domaine, les clés de chiffrement et de déchiffrement sont différentes. Avec ces protocoles, un utilisateur peut révéler la clé de chiffrement (d'où son nom de clé publique) afin que n'importe qui puisse lui envoyer un message. S'il garde secret la clé de

déchiffrement, il est le seul à pouvoir lire ces messages. L'idée même de la cryptographie asymétrique est très récente [28].

Le chiffrement à clé publique est fondé actuellement sur des problèmes mathématiques réputés difficiles, essentiellement en théorie des nombres, tels que la factorisation des entiers (sur laquelle se base RSA) ou le problème du logarithme discret (cryptographie à base de courbes elliptiques). Ces problèmes ne sont pas à l'abri de progrès algorithmiques qui permettraient de les résoudre en temps raisonnable. Par exemple on a montré en 2014 que le problème du logarithme discret sur des corps de petite caractéristique peut se résoudre en temps quasi-polynomial [4].

De plus, les protocoles basés sur ces problèmes sont facilement cassables par un ordinateur quantique (c'est-à-dire qu'il existe un algorithme quantique résolvant ces problèmes en temps polynomial). Il est donc nécessaire de rechercher d'autres protocoles résistants à l'ordinateur quantique et basés sur d'autres problèmes. La cryptologie post-quantique est la branche de la cryptologie s'intéressant à ce type de problèmes, elle repose sur des problèmes difficiles (NP-complets ou NP-durs), tels que le décodage des codes en métrique de Hamming, le décodage des codes en métrique rang ou la recherche de vecteurs de petite taille dans un réseau euclidien.

La métrique rang présente l'avantage de fournir des cryptosystèmes avec une clé de faible taille pour une sécurité élevée avec un faible coût de chiffrement et de déchiffrement car les algorithmes n'utilisent que des opérations d'algèbre linéaire.

Cette thèse est découpée en trois parties. Dans la première partie, nous introduirons les codes en métrique rang. Dans la deuxième partie, nous présenterons de nouvelles attaques sur des problèmes génériques ainsi qu'une attaque structurelle sur des cryptosystèmes existants. Nous analysons également les conséquences de l'ordinateur quantique sur la complexité de ces attaques. Dans la dernière partie, nous présenterons un générateur aléatoire, ainsi qu'une nouveau protocole de chiffrement à clé publique qui a l'avantage d'avoir des clés parfaitement aléatoires. En associant ce nouveau protocole avec la signature RankSign, nous proposons le premier IBE basé sur les codes.

Chapitre 2

Présentation des codes en métrique rang

Dans ce chapitre, nous présentons les notions essentielles et les prérequis de base à la compréhension de cette thèse. Nous introduisons la métrique rang à travers les codes $\mathbb{F}_{q^m}$-linéaires [35], qui sont des sous-cas des codes matriciels, puis nous définissons les problèmes difficiles en métrique rang et les algorithmes pour les résoudre, ainsi que plusieurs familles intéressantes de codes.

2.1 Codes $\mathbb{F}_{q^m}$-linéraires

Définition 2.1.1. *Un code $\mathcal{C}$ $\mathbb{F}_{q^m}$-linéaire de longueur n et de dimension k est un sous-espace vectoriel de $\mathbb{F}_{q^m}^n$ de dimension k. On écrit que $\mathcal{C}$ est un $[n,k]_{q^m}$ code linéaire ou, s'il n'y a pas d'ambiguïté, un $[n,k]$ code. Un élément de $\mathcal{C}$ est appelé un mot de code de $\mathcal{C}$ ou plus simplement un mot de $\mathcal{C}$.*

On peut représenter un code linéaire $\mathcal{C}$ de type $[n,k]$ de deux façons différentes mais équivalentes, soit par une matrice génératrice, soit par une matrice de parité.

Définition 2.1.2. *Soit $\mathcal{C}$ un code de type $[n,k]_{q^m}$ et $\boldsymbol{G} \in \mathbb{F}_{q^m}^{n\times n}$ une matrice dont les lignes forment une base de $\mathcal{C}$. $\boldsymbol{G}$ est appelée matrice génératrice du code et on a :*

$$\mathcal{C} = \{\boldsymbol{x}\boldsymbol{G}, \boldsymbol{x} \in \mathbb{F}_{q^m}^k\}$$

Soit $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ une matrice génératrice du noyau de $\boldsymbol{G}$, c'est-à-dire $\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}$. $\boldsymbol{H}$ est appelée matrice de parité du code et on a :

$$\mathcal{C} = \{\boldsymbol{y} \in \mathbb{F}_{q^m}^n, \boldsymbol{H}\boldsymbol{y}^T = \boldsymbol{0}\}$$

Remarque : Le code généré par une matrice de parité d'un code $\mathcal{C}$ est appelé code dual de $\mathcal{C}$ et est noté $\mathcal{C}^\perp$. Les matrices de parité de $\mathcal{C}^\perp$ correspondent aux matrices génératrices de $\mathcal{C}$.

Un code $\mathcal{C}$ possède plusieurs matrices génératrices ou de parité. Une matrice génératrice $\boldsymbol{G}$ (respectivement de parité $\boldsymbol{H}$) est dite systématique (ou sous forme systématique) si elle est de la forme $(\boldsymbol{I}_k|\boldsymbol{A})$ (respectivement $(\boldsymbol{B}|\boldsymbol{I}_{n-k})$. Le nombre de bits nécessaires pour représenter $\mathcal{C}$ sous la forme de sa matrice génératrice ou de parité sous forme systématique est le même. Il faut représenter $k(n-k)$ coefficients de $\mathbb{F}_{q^m}$, soit un total de $k(n-k)m\lceil\log q\rceil$ bits.

Pour définir la métrique rang sur les codes $\mathbb{F}_{q^m}$-linéaires, il faut définir la matrice associée à un mot de $\mathbb{F}_{q^m}^n$.

Définition 2.1.3. *Soit $(\beta_1, \ldots, \beta_m)$ une base de $\mathbb{F}_{q^m}/\mathbb{F}_q$. À tout vecteur $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$, on peut associer la matrice $\boldsymbol{M}_{\boldsymbol{x}} = (m_{ij})_{\substack{i\in\{1..m\}\\ j\in\{1..n\}}}$ telle que*

$$\forall j \in \{1..n\}, x_j = \sum_{i=1}^{m} m_{ij}\beta_i$$

- *La distance rang entre deux mots $\boldsymbol{x}$ et $\boldsymbol{y}$ de $\mathbb{F}_{q^m}$ est notée $d_R(\boldsymbol{x}, \boldsymbol{y})$ et vaut $\mathrm{Rg}(\boldsymbol{M}_{\boldsymbol{x}} - \boldsymbol{M}_{\boldsymbol{y}})$.*
- *Le poids d'un mot $\boldsymbol{x}$ est noté $|\boldsymbol{x}|_r$ et vaut $d_R(\boldsymbol{0}, \boldsymbol{x}) = \mathrm{Rg}(\boldsymbol{x})$.*

Cette définition est indépendante du choix de la base et définit bien une distance.

Démonstration. Une distance doit vérifier trois propriétés :

- symétrie : $d_R(\boldsymbol{A}, \boldsymbol{B}) = \mathrm{Rg}(\boldsymbol{A} - \boldsymbol{B}) = \mathrm{Rg}(\boldsymbol{B} - \boldsymbol{A}) = d_R(\boldsymbol{B}, \boldsymbol{A})$.
- séparation : $d_R(\boldsymbol{A}, \boldsymbol{B}) = 0 \iff \mathrm{Rg}(\boldsymbol{A} - \boldsymbol{B}) = 0 \iff \boldsymbol{A} - \boldsymbol{B} = \boldsymbol{0} \iff \boldsymbol{A} = \boldsymbol{B}$

— inégalité triangulaire : montrons d'abord que

$$\mathrm{Rg}(\boldsymbol{A}+\boldsymbol{B}) \leqslant \mathrm{Rg}(\boldsymbol{A}) + \mathrm{Rg}(\boldsymbol{B})$$

Notons $\langle \boldsymbol{M} \rangle$ l'espace vectoriel engendré par les lignes de la matrices $\boldsymbol{M}$. Par définition, $\mathrm{Rg}(\boldsymbol{M}) = \dim\langle \boldsymbol{M} \rangle$. Il est évident que $\langle \boldsymbol{A}+\boldsymbol{B} \rangle \subset \langle \boldsymbol{A} \rangle + \langle \boldsymbol{B} \rangle$
$\Rightarrow \mathrm{Rg}(\boldsymbol{A}+\boldsymbol{B}) \leqslant \dim(\langle \boldsymbol{A} \rangle + \langle \boldsymbol{B} \rangle)$
Or la réunion d'une base de $\langle \boldsymbol{A} \rangle$ et d'une base de $\langle \boldsymbol{B} \rangle$, de cardinal respectif $\mathrm{Rg}(\boldsymbol{A})$ et $\mathrm{Rg}(\boldsymbol{B})$, est trivialement une famille génératrice de $\langle \boldsymbol{A}+\boldsymbol{B} \rangle$, d'où $\dim(\langle \boldsymbol{A} \rangle + \langle \boldsymbol{B} \rangle) \leqslant \mathrm{Rg}(\boldsymbol{A}) + \mathrm{Rg}(\boldsymbol{B})$.

$$\begin{aligned} \text{Pour conclure, } d_R(\boldsymbol{A},\boldsymbol{C}) &= \mathrm{Rg}(\boldsymbol{A}-\boldsymbol{C}) \\ &= \mathrm{Rg}(\boldsymbol{A}-\boldsymbol{B}+\boldsymbol{B}-\boldsymbol{C}) \\ &\leqslant \mathrm{Rg}(\boldsymbol{A}-\boldsymbol{B}) + \mathrm{Rg}(\boldsymbol{B}-\boldsymbol{C}) \\ &= d_R(\boldsymbol{A},\boldsymbol{B}) + d_R(\boldsymbol{B},\boldsymbol{C}) \end{aligned}$$

Pour montrer que cette distance est indépendante de la base choisie, il suffit de voir que la matrice associée à un vecteur $\boldsymbol{x}$ dans une autre base s'obtient en multipliant $\boldsymbol{M}_{\boldsymbol{x}}$ par la matrice de passage entre ces deux bases, ce qui ne modifie pas le rang. □

On définit la distance minimale d'un code comme la plus petite distance entre deux mots distincts de ce code. Par linéarité, cette distance est égale au poids minimal des mots non nuls de ce code.

Définition 2.1.4. *Soit $\mathcal{C}$ un code $\mathbb{F}_{q^m}$-linéaire. La distance minimale de $\mathcal{C}$ est l'entier d tel que*

$$d = \min\{|\boldsymbol{x}-\boldsymbol{y}|_r, (\boldsymbol{x},\boldsymbol{y}) \in \mathcal{C}^2, \boldsymbol{x} \neq \boldsymbol{y}\} = \min\{|\boldsymbol{x}|_r, \boldsymbol{x} \in \mathcal{C} \backslash \{\boldsymbol{0}\}\}$$

Une notion importante en théorie des codes est la notion de support d'un mot. Cette notion intervient dans les algorithmes de décodage ou de recherche de mots de petit poids.

Définition 2.1.5. *Soit $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$. Le support de $\boldsymbol{x}$, noté $\mathrm{Supp}(\boldsymbol{x})$, est le $\mathbb{F}_q$-espace vectoriel de $\mathbb{F}_{q^m}$ généré par les coordonnées de $\boldsymbol{x}$.*

$$\mathrm{Supp}(\boldsymbol{x}) = \langle x_1, \ldots, x_n \rangle_{\mathbb{F}_q}$$

On a $|\boldsymbol{x}|_r = \dim \mathrm{Supp}(\boldsymbol{x})$.

Le nombre de supports de dimension w est donc égal au nombre de sous-espaces de dimension w de $\mathbb{F}_{q^m}$ $\begin{bmatrix} m \\ w \end{bmatrix}_q = \prod_{i=0}^{w-1} \frac{q^m - q^i}{q^w - q^i} = \Theta(q^{w(m-w)})$.

2.2 Codes matriciels

La métrique rang est également définie sur une famille plus vaste de codes, les codes matriciels. Dans ce cas là, les mots de codes sont des matrices à coefficients dans $\mathbb{F}_q$, ce qui permet de définir les codes $\mathbb{F}_{q^m}$-linéaires comme une sous-famille de codes matriciels.

Définition 2.2.1. *Un code matriciel $\mathcal{C}$ de dimension K et de longueur $m \times n$ est un sous-espace vectoriel de $\mathbb{F}_q^{m \times n}$ de dimension K. On écrit que $\mathcal{C}$ est un $[m \times n, K]_q$ code matriciel ou, s'il n'y a pas ambiguïté, un $[m \times n, K]$ code.*

Un code matriciel $[m \times n, K]$ peut être représenté par une matrice génératrice de taille $K \times mn$ où chaque ligne représente une matrice d'une base de $\mathcal{C}$. Le nombre de bits pour représenter $\mathcal{C}$ sous forme systématique vaut donc $K(mn - K) \lceil \log q \rceil$. La métrique rang est définie de la même manière pour les codes matriciel, la distance entre deux mots est le rang de leur différence et le poids d'un mot est son rang.

Les entiers m et n jouent un rôle symétrique dans le cas des codes matriciels. On peut aisément inverser leur rôle en considérant le transposé d'un code.

Définition 2.2.2. *Soit $\mathcal{C}$ un code matriciel de type $[m \times n, K]_q$. Le transposé de $\mathcal{C}$, noté $\mathcal{C}^T$ est le code matriciel de type $[n \times m, K]_q$ obtenu en transposant les mots de $\mathcal{C}$.*

$$\mathcal{C}^T = \{\boldsymbol{M}^T, \boldsymbol{M} \in \mathcal{C}\}$$

On peut associer un code matriciel à un code $\mathbb{F}_{q^m}$-linéaire de la même façon qu'on associe une matrice à un mot de $\mathbb{F}_{q^m}^n$.

Définition 2.2.3. *Soient $(\beta_1, \ldots, \beta_m)$ une base de $\mathbb{F}_{q^m}/\mathbb{F}_q$ et $\mathcal{C}$ un code $\mathbb{F}_{q^m}$-linéaire de type $[n, k]$. Le code matriciel $\mathcal{C}^M$ associé à $\mathcal{C}$ (relativement à la base β) est défini par*

$$\mathcal{C}^M = \{\boldsymbol{M}_{\boldsymbol{c}} \in \mathbb{F}_q^{m \times n}, \boldsymbol{c} \in \mathcal{C}\}$$

Soit $\boldsymbol{G} = \begin{pmatrix} \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_k \end{pmatrix}$ *une matrice génératrice de* $\mathcal{C}$*. Alors la famille* $(\boldsymbol{M}_{\beta_j \boldsymbol{g}_j})$ *est une base de* $\mathcal{C}^M$*.*

Le code $\mathcal{C}^M$ *est donc un* $[m \times n, km]_q$ *code matriciel qui possède exactement la même distribution de poids que* $\mathcal{C}$*.*

Remarque : Le nombre de bits permettant de représenter un $[n, k]_{q^m}$ code linéaire est $k(n-k)m \lceil \log q \rceil$ alors qu'un code matriciel avec les mêmes paramètres nécessite $km(nm - km) \lceil \log q \rceil = k(n-k)m^2 \lceil \log q \rceil$ bits pour être représenté en mémoire. La $\mathbb{F}_{q^m}$-linéarité permet donc de gagner un facteur m en taille, ce qui est très utile en cryptographie pour diminuer la taille des clés.

Pour définir le support d'un mot dans le cas des codes matriciels, on peut soit considérer le sous-espace vectoriel de $\mathbb{F}_q^n$ engendré par les lignes de $\boldsymbol{M}$ ou le sous-espace vectoriel de $\mathbb{F}_q^m$ engendré par les colonnes de $\boldsymbol{M}$. Selon les paramètres du code et l'algorithme, il est parfois plus utile de considérer le support lignes que le support colonnes.

2.3 Problèmes difficiles en métrique rang

Comme pour la métrique de Hamming, les principaux problèmes sur lesquels est fondée la cryptographie à base de codes en métrique rang sont le problème du décodage par syndrome et la recherche de mots de petit poids. Les cryptosystèmes se basent sur la difficulté en pratique de ces problèmes.

Définition 2.3.1. *Soient* $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ *une matrice de parité d'un code* $\mathbb{F}_{q^m}$*-linéaire* $\mathcal{C}$*,* $\boldsymbol{s} \in \mathbb{F}_{q^m}^{n-k}$ *et* w *un entier positif. Le problème du décodage par syndrome, appelé problème RSD (Rank Syndrome Decoding), est de trouver* $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ *tel que :*

$$\begin{cases} \boldsymbol{H}\boldsymbol{x}^T &= \boldsymbol{s}^T \\ |\boldsymbol{x}|_r &= w \end{cases}$$

Remarque : Une version équivalente de ce problème est : étant donné un mot $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$, trouver un mot de code $\boldsymbol{c} \in \mathcal{C}$ tel que $d_R(\boldsymbol{c}, \boldsymbol{y}) = w$. En effet, si $\boldsymbol{c}$ est solution du

problème précédent, alors $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ avec $|\boldsymbol{e}|_r = w$ et $\boldsymbol{e}$ est solution du problème RSD avec $\boldsymbol{s} = \boldsymbol{y}\boldsymbol{H}^T$.

Définition 2.3.2. *Soient $\mathcal{C}$ un code $\mathbb{F}_{q^m}$-linéaire et w un entier positif. Le problème de la recherche de mots de petit poids est de trouver un mot $\boldsymbol{c} \in \mathcal{C}$ tel que $|\boldsymbol{c}|_r = w$.*
Ce problème est un cas particulier du problème RSD avec $\boldsymbol{s} = \boldsymbol{0}$.

Contrairement à la métrique de Hamming, il n'est pas démontré que ces problèmes sont NP-complets [8, 78]. Cependant on peut réduire de manière probabiliste ces problèmes en métrique de Hamming à la métrique rang [47]. Ces problèmes sont donc considérés difficiles, les meilleurs algorithmes qui les résolvent sont exponentiels. Dans le cas des codes matriciels, ces problèmes sont équivalents au problème NP-complet MinRank [19]. L'ajout de la linéarité sur $\mathbb{F}_{q^m}$ est une obstruction (a priori) à la NP-complétude du problème.

2.4 Algorithmes génériques de résolution du problème RSD

Dans cette section, nous présentons l'algorithme GRS[43] de résolution du problème RSD. L'idée générale est de trouver un sous-espace F contenant le support de $\boldsymbol{x}$ puis d'exprimer les coordonnées de $\boldsymbol{x}$ dans une base de F et d'utiliser les équations de parité pour résoudre un système linéaire d'inconnues les coefficients des coordonnées de $\boldsymbol{x}$ dans cette base.

2.4.1 Première version de l'algorithme

Cet algorithme tire pleinement parti de la notion de support de l'erreur. Soit $\boldsymbol{s}^T = \boldsymbol{H}\boldsymbol{x}^T$ une instance du problème RSD avec $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ une matrice de parité d'un code $\mathcal{C}$ de type $[n, k]$ et $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ de poids w. L'algorithme est légèrement différent selon que $n \geqslant m$ ou non.

Nous traiterons d'abord le cas $n \geqslant m$, qui est le cas le fréquent dans les cryptosystèmes en métrique rang. Supposons qu'on connaisse un $\mathbb{F}_q$-espace vectoriel F de dimension r

contenant le support E de $\boldsymbol{x}$. Nous calculerons plus loin la probabilité de cet événement en tirant au hasard un espace de dimension r.

Soit $(F_1, \ldots, F_r)$ une base de F. Exprimons les coordonnées de $\boldsymbol{x}$ dans cette base.

$$\forall j \in \{1, \ldots, n\}, x_j = \sum_{i=1}^{r} \lambda_{ij} F_i, \text{ avec } \lambda_{ij} \in \mathbb{F}_q^{nr}$$

Réécrivons à présent les équations données par la matrice $\boldsymbol{H}$ pour faire apparaître les λ_{ij} :

$$\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{s}^T \tag{2.1}$$

$$\iff \begin{cases} H_{11}x_1 & +\cdots+ & H_{1n}x_n & = & s_1 \\ \vdots & & \vdots & & \vdots \\ H_{n-k,1}x_1 & +\cdots+ & H_{n-k,n}x_n & = & s_{n-k} \end{cases}$$

$$\iff \begin{cases} \sum_{j=1}^{r}(\lambda_{j1}H_{11}F_j & +\cdots+ & \lambda_{jn}H_{1n}F_j) & = & s_1 \\ \vdots & & \vdots & & \vdots \\ \sum_{j=1}^{r}(\lambda_{j1}H_{n-k,1}F_j & +\cdots+ & \lambda_{jn}H_{n-k,n}F_j) & = & s_{n-k} \end{cases} \tag{2.2}$$

Il faut à présent réécrire ce système sur $\mathbb{F}_q$ en projetant chaque équation sur une base $(\beta_1, \ldots, \beta_m)$ de $\mathbb{F}_{q^m}/\mathbb{F}_q$. En appelant φ_i la projection sur β_i, il vient, pour tout $i \in \{1, \ldots, m\}$:

$$\begin{cases} \sum_{j=1}^{r}\left(\lambda_{j1}\varphi_i(H_{11}F_j) & +\cdots+ & \lambda_{jn}\varphi_i(H_{1n}F_j)\right) & = & \varphi_i(s_1) \\ \vdots & & \vdots & & \vdots \\ \sum_{j=1}^{r}\left(\lambda_{j1}\varphi_i(H_{n-k,1}F_j) & +\cdots+ & \lambda_{jn}\varphi_i(H_{n-k,n}F_j)\right) & = & \varphi_i(s_{n-k}) \end{cases} \tag{2.3}$$

On obtient donc un système à nr inconnues et $(n-k)m$ équations sur $\mathbb{F}_q$. Comme on suppose que F contient le support de $\boldsymbol{x}$, ce système admet au moins une solution. Pour s'assurer que cette solution est unique, il est nécessaire d'avoir plus d'équations que d'inconnues, ce qui implique la condition suivante :

$$nr \leqslant m(n-k) \Leftrightarrow r \leqslant m - \left\lceil \frac{km}{n} \right\rceil$$

La probabilité p que F contienne le support de $\boldsymbol{x}$ se calcule par dénombrement. Un sous-espace de $\mathbb{F}_q^m$ de dimension r contient $\begin{bmatrix} r \\ w \end{bmatrix}_q$ sous-espaces de dimension w. Le nombre

total d'espaces de dimension w étant $\begin{bmatrix} m \\ w \end{bmatrix}_q$, on obtient la formule suivante :

$$p = \frac{\begin{bmatrix} r \\ w \end{bmatrix}_q}{\begin{bmatrix} m \\ w \end{bmatrix}_q} = \Theta\big(q^{-w(m-r)}\big)$$

La complexité en moyenne de cet algorithme est égale à l'inverse de cette probabilité fois le coût de la résolution d'un système à $(n-k)m$ inconnues dans $\mathbb{F}_q$. En prenant $r = m - \left\lceil \frac{km}{n} \right\rceil$, on obtient :

$$\mathcal{O}\big(\underbrace{(m^3(n-k)^3)}_{\text{coût de l'algèbre linéaire}} \; q^{w\left\lceil \frac{km}{n} \right\rceil} \big) \text{ opérations dans } \mathbb{F}_q.$$

À présent, traitons le cas $m > n$. L'idée est de chercher l'erreur sous forme matricielle en considérant les équations de parité du code matriciel associé à $\mathcal{C}$.

Soit $\boldsymbol{M}_{\boldsymbol{x}} \in \mathbb{F}_q^{m \times n}$ la matrice associée à $\boldsymbol{x}$. Supposons que l'on connaisse un sous-espace vectoriel F de $\mathbb{F}_q^n$ contenant le support-ligne de $\boldsymbol{M}_{\boldsymbol{x}}$. Soit $(F_1, \ldots, F_r)$ une base de F. On peut exprimer $\boldsymbol{M}_{\boldsymbol{x}}$ dans cette base, chaque ligne de la matrice est une combinaison linéaire des éléments de cette base :

$$\boldsymbol{M}_{\boldsymbol{x}} = \begin{pmatrix} \sum_{i=1}^{r} \lambda_{i1} F_i \\ \vdots \\ \sum_{i=1}^{r} \lambda_{im} F_i \end{pmatrix}$$

Les (λ_{ij}) sont les mr nouvelles inconnues du système. De même que dans le cas précédent, on peut réécrire les équations de parité sur $\mathbb{F}_q$ pour obtenir un système linéaire à $(n-k)m$ équations. On choisit r de façon à avoir plus d'équations que d'inconnues, ce qui implique

$$mr \leqslant m(n-k) \iff r \leqslant n-k$$

Comme l'espace F et le support-ligne de $\boldsymbol{M}_{\boldsymbol{x}}$ sont des sous-espaces de $\mathbb{F}_q^n$, la probabilité

que F contienne le support-ligne de $\boldsymbol{M}_{\boldsymbol{x}}$ vaut

$$\frac{\begin{bmatrix} r \\ w \end{bmatrix}_q}{\begin{bmatrix} n \\ w \end{bmatrix}_q} = \Theta\big(q^{-w(n-r)}\big).$$

En prenant $r = n - k$, on obtient une complexité en moyenne de $\mathcal{O}\big(m^3(n-k)^3 q^{wk}\big)$ opérations dans $\mathbb{F}_q$.

Remarque : Cette technique revient à appliquer l'algorithme précédent au code matriciel transposé associé à $\mathcal{C}$.

2.4.2 Utilisation de la $\mathbb{F}_{q^m}$-linéarité

L'algorithme précédent n'utilise pas la $\mathbb{F}_{q^m}$-linéarité du code, en effet cet algorithme est applicable aux codes matriciels. Pour améliorer l'algorithme et utiliser la linéarité sur $\mathbb{F}_{q^m}$, l'idée est de se ramener à la recherche de mots de petit poids dans un code bien choisi.

Soit $\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{s}^T$ une instance du problème RSD avec $|\boldsymbol{x}|_r = w$ et $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$. Soient $\boldsymbol{G}$ une matrice génératrice du code $\mathcal{C}$ de matrice de parité $\boldsymbol{H}$ et $\boldsymbol{y}$ une solution quelconque du système $\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{s}^T$. On sait qu'il existe $\boldsymbol{m} \in \mathbb{F}_{q^m}^k$ tel que $\boldsymbol{y} = \boldsymbol{m}\boldsymbol{G} + \boldsymbol{x}$.

Soit $\mathcal{C}'$ le code engendré par $\mathcal{C}$ et $\boldsymbol{y}$:

$$\mathcal{C}' = \mathcal{C} + \mathbb{F}_{q^m}\boldsymbol{y}$$

Ce code contient $\boldsymbol{x}$ et la matrice $\boldsymbol{G}' = \left(\begin{array}{c} \boldsymbol{G} \\ \hline \boldsymbol{y} \end{array}\right)$ obtenue en concaténant $\boldsymbol{y}$ et $\boldsymbol{G}$ est une matrice de parité. Par linéarité par rapport à $\mathbb{F}_{q^m}$, tous les mots de la forme $\alpha\boldsymbol{x}, \alpha\mathbb{F}_{q^m}^*$ appartiennent à $\mathcal{C}'$ et tous ces mots sont de poids w mais de supports distincts. On peut donc supposer sans perte de généralité qu'il existe $\boldsymbol{x}' \in \mathcal{C}'$ tel que $1 \in E' = \mathrm{Supp}(\boldsymbol{x}')$ et que $\boldsymbol{x}'$ soit un multiple de $\boldsymbol{x}$.

Au lieu de choisir au hasard un espace vectoriel F de dimension r comme précédemment, on impose à F de contenir 1. La probabilité p qu'un espace F ainsi choisi contienne E'

vaut

$$p = \frac{\begin{bmatrix} r-1 \\ w-1 \end{bmatrix}_q}{\begin{bmatrix} m-1 \\ w-1 \end{bmatrix}_q} = \Theta\big(q^{(w-1)(m-r)}\big)$$

En effet, l'application $\psi : F \mapsto F/\mathbb{F}_q$ de l'ensemble des sous-espaces vectoriels de dimension k de $\mathbb{F}_{q^m}$ contenant 1 vers l'ensemble des sous-espaces vectoriels de dimension $k-1$ de l'espace vectoriel-quotient $\mathbb{F}_{q^m}/\mathbb{F}_q$ est une bijection. De plus $F \supset E'$ si et seulement si $F/\mathbb{F}_q \supset E'/\mathbb{F}_q$. Or,

$$\dim F/\mathbb{F}_q = \dim F - 1 = r-1, \dim E'/\mathbb{F}_q = w-1, \dim \mathbb{F}_{q^m}/\mathbb{F}_q = m-1$$

donc la probabilité de ce dernier événement est $\dfrac{\begin{bmatrix} r-1 \\ w-1 \end{bmatrix}_q}{\begin{bmatrix} m-1 \\ w-1 \end{bmatrix}_q}$.

Le reste de l'algorithme se déroule comme précédemment. Comme $\mathcal{C}'$ est de dimension $k+1$, il faut choisir $r = m - \left\lceil \frac{(k+1)m}{n} \right\rceil$, ce qui nous donne au final une complexité en moyenne de $\mathcal{O}\big(m^3(n-k)^3 q^{(w-1)\left\lceil \frac{(k+1)m}{n} \right\rceil}\big)$ pour trouver $\boldsymbol{x}'$. Pour finir, on résout le système $\alpha \boldsymbol{x}' \boldsymbol{H}^T = \boldsymbol{s}$ d'inconnue α pour calculer $\boldsymbol{x}$.

Remarque : Dans le cas $m > n$, on ne peut pas utiliser directement cette idée car la linéarité par rapport à $\mathbb{F}_{q^m}$ ne permet pas de choisir un élément du support-ligne de $\boldsymbol{x}$ vu sous forme matricielle. Pour obtenir le facteur $w-1$ dans ce cas là, on utilisera la méthode présentée dans la partie 3.1.

2.5 Borne sur les codes en métrique rang

Définition 2.5.1 (Borne de Singleton). *Soient $\mathcal{C}$ un code linéaire de type $[n,k]_{q^m}$ et d sa distance minimale.*

- *si $m \leqslant n$, alors $d \leqslant \left\lfloor \frac{(n-k)m}{n} \right\rfloor + 1$,*
- *si $m > n$, alors $d \leqslant n - k + 1$,*

ce qu'on peut résumer par $d \leqslant \left\lfloor \frac{(n-k)m}{\max(m,n)} \right\rfloor + 1$.

Remarque : Dans le cas d'un code matriciel de type $[m \times n, K]_q$, la borne de Singleton vaut $d \leqslant \left\lfloor \frac{nm-K}{\max(m,n)} \right\rfloor + 1$.

La borne de Singleton correspond à la valeur à partir de laquelle le problème RSD devient facile, c'est-à-dire qu'il est résoluble par un algorithme probabiliste en temps polynomial. En effet, en reprenant le système d'équations 2.3 avec $r \geqslant \left\lfloor \frac{(n-k)m}{n} \right\rfloor + 1$ alors le système a plus d'inconnues que d'équations donc possède une solution avec une probabilité non négligeable (cette probabilité est supérieure à une constante $c \approx 0.289$ quels que soient les paramètres du code).

Définition 2.5.2 (Borne de Gilbert-Varshamov en métrique rang)**.** *Soit $B(m, n, r, q)$ le volume d'une boule fermée de rayon r dans $\mathbb{F}_{q^m}^n$. Par définition, ce volume correspond au nombre de matrices de taille $m \times n$ à coefficients dans $\mathbb{F}_q$ et de rang inférieur ou égal à r.*

On définit la borne de Gilbert-Varshamov, notée $RGV(n, k, m, q)$ pour Rank Gilbert-Varshamov, comme étant le plus petit entier r tel que

$$B(m, n, r, q) \geqslant q^{m(n-k)}.$$

Soit $\mathcal{C}$ un code de type $[n, k]_{q^m}$. La borne de Gilbert-Varshamov correspond à l'entier à partir duquel l'espérance du nombre de solutions du problème RSD pour le code $\mathcal{C}$ dépasse 1, quel que soit le syndrome $\boldsymbol{s}$. Une valeur asymptotique de cette borne quand m ou n tend vers l'infini est donnée par la formule

$$RGV(n, k, m, q) \sim \frac{m + n - \sqrt{(m-n)^2 + 4km}}{2} \text{ [63]}.$$

Dans le cas particulier $m = n$, on a la simplification suivante

$$RGV(n, k, n, q) \sim n\left(1 - \sqrt{\frac{k}{n}}\right)$$

2.6 Familles intéressantes de codes en métrique rang

Dans cette section, nous présentons des familles de codes en métrique rang qu'on sait décoder efficacement et qui peuvent être utilisés en cryptographie.

2.6.1 Code de Gabidulin

Les codes de Gabidulin ont été introduits en 1985 [35], mais sont connus depuis 1978 [26]. Cette famille de codes est similaire aux codes de Reed-Solomon en métrique de Hamming, leur définition fait intervenir les q-polynômes à la place des polynomes. Elle possède également une forte structure algébrique. Les q-polynômes ont été introduits par Ore [69], on rappelle ici les définitions de base.

Définition 2.6.1 (q-polynômes)**.** *L'ensemble des q-polynômes sur $\mathbb{F}_{q^m}$ est l'ensemble des polynômes de la forme*

$$\left\{ P(X) = \sum_{i \in \mathbb{N}} p_i X^{q^i}, (p_i) \in \mathbb{F}_{q^m}^{\mathbb{N}} \text{ de support fini} \right\}$$

Le q-degré d'un q-polynôme P, noté $\deg_q P$, est le plus grand entier r tel que $p_r \neq 0$.

Muni de l'addition et de la composition $P \circ Q(X) = P(Q(X))$, l'ensemble des q-polynômes est un anneau unitaire non-commutatif de neutre X.

Définition 2.6.2 (Codes de Gabidulin)**.** *Soient $k, n, m \in \mathbb{N}$ tels que $k \leqslant n \leqslant m$. Soit $\boldsymbol{x} = (x_1, \ldots, x_n)$ une famille $\mathbb{F}_q$-libre de $\mathbb{F}_{q^m}$. Le code de Gabidulin $\mathcal{G}_{\boldsymbol{x}}(n, k, m)$ est le code $[n, k]_{q^m}$ suivant :*

$$\left\{ P(\boldsymbol{x}), \deg_q P < k \right\} \text{ où } P(\boldsymbol{x}) \text{ désigne l'évaluation des coordonnées de } \boldsymbol{x} \text{ par } P.$$

Une matrice génératrice de ce code est

$$\boldsymbol{G} = \begin{pmatrix} x_1 & \ldots & x_n \\ x_1^q & \ldots & \\ \vdots & & \\ x_1^{q^{k-1}} & \ldots & x_n^{q^{k-1}} \end{pmatrix}$$

Ces codes peuvent décoder efficacement jusqu'à $\left\lfloor \frac{n-k}{2} \right\rfloor$ erreurs [35]. Ils peuvent donc être utilisés pour le cryptosystème de McEliece [37]. Ces cryptosytèmes ont été attaqués en raison de leur structure algébrique [73].

2.6.2 Code LRPC

Les codes LRPC (Low Rank Parity-Check) ont été introduits en 2013 [40]. À l'instar des MDPC en métrique de Hamming [68] ou du cryptosystème NTRU en métrique euclidienne [56], leur définition et leur algorithme de décodage repose sur l'existence d'une base de mots de code de petit poids du code dual.

Définition 2.6.3 (Low Rank Parity-Check codes)**.** *Soit F un sous-espace de $\mathbb{F}_{q^m}$ de dimension d. Soit $\boldsymbol{H} = (h_{ij})_{\substack{1\leqslant i\leqslant n-k \\ 1\leqslant j\leqslant n}} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ de rang plein telle que les coefficients de $\boldsymbol{H}$ engendrent F :*

$$F = \langle h_{ij}, i \in \{1..n-k\}, j \in \{1..n\} \rangle$$

Soit $\mathcal{C}$ le code $[n,k]_{q^m}$ de matrice de parité $\boldsymbol{H}$. $\mathcal{C}$ est un code LRPC de poids d.

Remarque : on dit que $\boldsymbol{H}$ est une matrice **uniforme** de poids d et de support F.

Une sous-famille intéressante des codes LRPC est la famille des codes LRPC doublement circulants (DC-LRPC) qui sont représentés en mémoire avec moins de bits.

D'abord, ils nous faut rappeler la définition des matrices circulantes.

Définition 2.6.4 (Matrices circulantes)**.** *Soit n un entier. Une matrice carrée $\boldsymbol{M} = (m_{ij})_{\substack{i\in\mathbb{Z}/n\mathbb{Z} \\ j\in\mathbb{Z}/n\mathbb{Z}}}$ de taille n est dite circulante si et seulement si*

$$m_{ij} = m_{i+1,j+1}$$

Autrement dit, les matrices circulantes sont de la forme :

$$\begin{pmatrix} m_{00} & m_{01} & \dots & m_{0,n-1} \\ m_{0,n-1} & m_{00} & \ddots & m_{0,n-2} \\ \vdots & \ddots & \ddots & \vdots \\ m_{01} & m_{02} & \dots & m_{00} \end{pmatrix}$$

On notera $\mathcal{M}_c(n,K)$ l'ensemble des matrices circulantes de taille n à coefficients dans le corps K.

Définition 2.6.5. *Un code LRPC doublement circulant de poids d est une code LRPC de type $[2n,n]_{q^m}$ qui possède une matrice de parité $\boldsymbol{H}$ de la forme*

$$\boldsymbol{H} = (\boldsymbol{A}|\boldsymbol{B})$$

où $\boldsymbol{A}$ et $\boldsymbol{B}$ sont deux matrices inversibles circulantes d'ordre n, toutes deux à coefficients dans un même sous-espace F de $\mathbb{F}_{q^m}$ de dimension d.

Les codes LRPC doublement circulants ont une structure algébrique plus riche comme le montre la proposition suivante :

Proposition 2.6.6. *Soit $\mathcal{R} = \mathbb{F}_{q^m}[X]/(X^n-1)$ l'algèbre des polynômes à coefficients dans $\mathbb{F}_{q^m}$ modulo $X^n - 1$. L'ensemble des matrices circulantes $\mathcal{M}_c(n, \mathbb{F}_{q^m})$ est une $\mathbb{F}_{q^m}$-algèbre isomorphe à $\mathcal{R}$ dont l'isomorphisme canonique est :*

$$\begin{aligned} f : \quad \mathcal{R} \quad &\to \mathcal{M}_c(n, \mathbb{F}_{q^m}) \\ \sum_{i=0}^{n-1} p_i X^i &\mapsto \begin{pmatrix} p_0 & \cdots & p_{n-1} \\ \vdots & \ddots & \vdots \\ p_1 & \cdots & p_0 \end{pmatrix} \end{aligned}$$

Décodage des LRPC

L'algorithme de décodage des codes LRPC utilise le fait qu'on connaît une matrice de parité de petit poids du code. À partir des équations de parités obtenues par cette matrice, on peut retrouver de manière probabiliste le support de l'erreur, ce qui permet de décoder. En choisissant correctement les paramètres du code, on peut rendre négligeable la probabilité d'échec de l'algorithme.

Soit $\boldsymbol{H}$ une matrice de parité de petit poids d d'un code LRPC $\mathcal{C}$ de type $[n, k]_{q^m}$. Soit $\boldsymbol{H}\boldsymbol{e}^T = \boldsymbol{s}^T$ une instance du problème RSD avec $|\boldsymbol{e}|_r = w$. Soient F le support de dimension d de $\boldsymbol{H}$ et E le support de dimension w de $\boldsymbol{e}$. L'algorithme de décodage se déroule en trois étapes :

- Premièrement on calcule le support S de $\boldsymbol{s}$. Par hypothèse, S est inclus dans l'espace-produit $\langle EF \rangle = \langle e_i f_j \rangle_{\mathbb{F}_q}$ où $(e_i)_{i\in[1..w]}$ et $(f_j)_{j\in[1..d]}$ sont une base respective de E et F. Supposons que $S = \langle EF \rangle$ et que $\dim S = \dim\langle EF \rangle = wd$.
- Deuxièmement, on calcule l'intersection $I = \cap_{j\in[1..d]} f_j^{-1} S$. Par hypothèse, $E \subseteq I$. Supposons qu'il y ait égalité, ce qui nous permet de retrouver le support de $\boldsymbol{e}$.
- Troisièmement, on exprime les coordonnées de $\boldsymbol{e}$ dans une base de E on l'on résout le système à nw inconnues et $(n-k)m$ équations obtenu à partir des équations de parité pour calculer $\boldsymbol{e}$.

Cet algorithme repose sur les hypothèses que l'espace produit $\langle EF \rangle$ est de dimension wd, que S est égal à $\langle EF \rangle$ et que le support E est égal à l'intersection des $f_j^{-1}S$. En choisissant les paramètres du code tels que $wd \leqslant \min(m, n-k)$, on peut majorer la probabilité d'échec par $q^{-(n-k+1-wd)}$.

Par un précalcul, on peut ramener la résolution du système de la troisième étape à un produit matriciel, ce qui permet d'obtenir une complexité finale en $\mathcal{O}\big(w^2(4d^2m + n^2)\big)$ opérations dans $\mathbb{F}_q$. Toutes les précisions concernant les calculs sont données dans l'article originel [40].

2.6.3 Codes simples

Les codes simples sont une famille de codes pouvant décoder asymptotiquement jusqu'à la borne de Gilbert-Varshamov. Ces codes sont très similaires à une famille de codes en métrique sous-espace (une métrique très proche de la métrique rang) dont on trouve une description dans l'article [77].

Définition 2.6.7 (Simple codes)**.** *Un code $\mathbb{F}_{q^m}$-linéaire $\mathcal{C}$ est dit (n, k, t)-simple (ou simplement simple si t, k et n sont clairement identifiés), s'il possède une matrice de parité $\boldsymbol{H}$ de la forme :*

$$\boldsymbol{H} = \left(\begin{array}{c|c} & \boldsymbol{0_t} \\ \boldsymbol{I}_{n-k} & \\ & \boldsymbol{R} \end{array} \right)$$

où $\boldsymbol{I}_{n-k}$ est la matrice identité de taille $(n-k) \times (n-k)$, $\boldsymbol{0_t}$ est la matrice nulle de taille $t \times k$ et $\boldsymbol{R}$ une matrice quelconque à coefficients dans $\mathbb{F}_{q^m}$ de taille $(n-k-t) \times k$. $\mathcal{C}$ est appelé code simple aléatoire si $\boldsymbol{R}$ est choisi uniformément au hasard parmi les matrices de cette taille.

Ces codes sont appelés codes simples car leur algorithme de décodage est assez trivial.

Proposition 2.6.8. *Soit $\mathcal{C}$ un code (n, k, t)-simple aléatoire avec $t < \frac{m+n-\sqrt{(m-n)^2+4km}}{2}$ et w un entier. Si $w \leqslant t$, alors $\mathcal{C}$ peut décoder une erreur de poids w avec une probabilité d'échec $p \sim \frac{1}{q^{t-w+1}}$ quand $q \to \infty$.*

Démonstration. Soient $\boldsymbol{e}$ une erreur de poids w choisie uniformément au hasard et $E = \mathrm{Supp}(\boldsymbol{e})$. Soit $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^T$ le syndrome associée à $\boldsymbol{e}$.

En raison de la forme particulière de $\boldsymbol{H}$, les t premières coordonnées de $\boldsymbol{s}$ sont égales au$t$ premières coordonnées de $\boldsymbol{e}$.

$$e_i = s_i \text{ pour tout } i \in \{1..t\}$$

Avec une probabilité de $1 - \frac{1}{q^{t-w+1}} + o\left(\frac{1}{q^{t-w+1}}\right)$, ces t coordonnées engendrent E. Dans ce cas, on peut exprimer les $n-t$ coordonnées de $\boldsymbol{e}$ restantes dans une base $E_1, \ldots, E_w$ de E :

$$\forall i \in \{t+1..n\}, e_i = \sum_{j=1}^{w} \lambda_{ij} E_j$$

avec $\lambda_{ij} \in \mathbb{F}_q$.

Les $(n-k-t)$ dernières coordonnées de $\boldsymbol{s}$ permettent de calculer les $(n-t)$ dernières coordonnées de $\boldsymbol{e}$. En effet, on a le système d'équations suivant :

$$(\boldsymbol{I}_{n-k-t}|\boldsymbol{R}) \begin{pmatrix} e_{t+1} \\ \vdots \\ e_n \end{pmatrix} = \begin{pmatrix} s_{n-k-t+1} \\ \vdots \\ s_{n-k} \end{pmatrix}$$

En réécrivant ce système sur $\mathbb{F}_q$ et en exprimant les e_i en fonctions des λ_{ij}, on obtient un système de $m(n-k-t)$ équations à $w(n-t)$ inconnues.

Montrons que ce système a plus d'équations que d'inconnues :
la valeur $\frac{m+n-\sqrt{(m-n)^2+4km}}{2}$ est la plus petite racine du polynôme $P(x) = (n-k-x)m - (n-x)x$. Or, on a par hypothèse

$$w \leqslant t < \frac{m+n-\sqrt{(m-n)^2+4km}}{2} \Rightarrow (n-k-t)m > (n-t)t \geqslant (n-w)t.$$

en remplaçant x par t dans la deuxième inégalité.

Par construction, ce système admet toujours une solution qui est unique si ce système est de rang plein, ce qui arrive avec une probabilité de l'ordre de $1 - \frac{1}{q^{(n-k-t)m-(n-t)w+1}} > 1 - \frac{1}{q^{t-w+1}}$. □

Le succès du décodage dépend essentiellement de la probabilité $1-p$ de retrouver le support E de l'erreur grâce t premières coordonnées du syndrome $\boldsymbol{s}$. Cette probabilité peuvent être rendue aussi proche de 1 que l'on veut en augmentant q ou en décodant des erreurs de poids $< t$.

Dans le cas particulier $m = n$ et $w = t \approx n\left(1 - \sqrt{\frac{k}{n}}\right)$ (qui correspond à la borne de Gilbert-Varshamov), un code simple peut décoder jusqu'à la borne de Gilbert-Varshamov, ce qui est meilleur que les codes de Gabidulin qui peuvent décoder des erreurs de poids inférieur ou égal à $\frac{n-k}{2}$. Cependant l'avantage des codes de Gabidulin est d'avoir un algorithme de décodage déterministe.

2.7 Cryptographie à base de codes en métrique rang

Les codes peuvent être utilisés dans le domaine de la cryptographie à clé publique. En 1978, McEliece a proposé un cryptosystème de chiffrement [66] basé sur le problème du décodage d'un code aléatoire en métrique de Hamming. Ce problème a été prouvé NP-difficile la même année [8]. Cependant, l'idée de ce cryptosystème n'est pas liée à la métrique choisie et s'adapte parfaitement à la métrique rang.

L'idée est de choisir un code dont on connaît un algorithme de décodage efficace puis de le masquer par un procédé algorithmique afin de le faire ressembler à un code aléatoire. La description de ce code apparemment aléatoire constitue la clé publique et la description du code utilisé pour le décodage (par exemple une matrice de parité de petit poids pour les LRPC) la clé secrète.

Pour chiffrer un message m, on envoie le mot de code bruité $\boldsymbol{y} = m\boldsymbol{G} + \boldsymbol{e}$ où $\boldsymbol{G}$ est une matrice génératrice du code masqué et $\boldsymbol{e}$ erreur de poids w (w est un paramètre du cryptosystème).
Pour déchiffrer, la connaissance du code non masqué est nécessaire.

Voici plus explicitement le fonctionnement de l'algorithme :

clé publique : un code $\mathcal{C}$ de type $[n, k]$ sur $\mathbb{F}_{q^m}$, de matrice génératrice $\boldsymbol{G}$ apparemment aléatoire, dont on connaît un algorithme efficace de décodage. Un paramètre w désignant le poids des erreurs que l'on peut décoder.

clé secrète : une description de $\mathcal{C}$ permettant de décoder efficacement. Éventuellement une fonction secrète linéaire f permettant de retrouver le message à partir du mot décodé.

Input : $\boldsymbol{m}$: message
$\boldsymbol{G}$: matrice génératrice du code
Output : $\boldsymbol{y}$: chiffré de $\boldsymbol{m}$
Data : w : poids des erreurs que $\mathcal{C}$ peut corriger

begin
  $\boldsymbol{e} \leftarrow$ mot aléatoire de poids w ;
  $\boldsymbol{y} \leftarrow mG + e$;
  return y
end

Algorithme 1 : Chiffrement

Input : $\boldsymbol{y}$: chiffré d'un message $\boldsymbol{m}$
Output : $\boldsymbol{m}$
Data : un algorithme $DECODER$ de décodage du code $\mathcal{C}$
la fonction linéaire f

begin
  $m \leftarrow DECODER(y)$;
  $m \leftarrow f(m)$;
  return m
end

Algorithme 2 : Déchiffrement

La sécurité du cryptosystème McEliece étant basée sur la difficulté de décoder un code aléatoire, il est important que l'attaquant ne puisse pas retrouver la structure sous-jacente du code.

La première famille de codes en métrique rang utilisée pour un cryptosystème de McEliece a été la famille des codes de Gabidulin [37]. Cette famille de codes a été attaquée à plusieurs reprises [50, 51] et des améliorations du masquage ont été apportées [7, 36] pour contrer ces attaques. Une nouvelle attaque structurelle faisant intervenir les q-polynômes a été découverte par Overbeck en 2005 [72] a finalement cassé ces améliorations. Récemment, un nouveau type de masquage a permis d'empêcher cette attaque [65].

Pendant longtemps, les codes de Gabidulin ont été les seuls codes en métrique rang dont on connaissait un algorithme de décodage. En 2013, les codes LRPC ont été découverts (ou inventés) [40] et sont de bon candidats pour le cryptosystème de McEliece.

En effet, l'algorithme de décodage des codes LRPC repose de manière cruciale sur la connaissance d'une matrice de parité de petit poids. Cela permet de facilement masquer la structure d'un code LRPC en révélant une autre matrice de parité, typiquement sous forme systématique. Être capable de retrouver la structure d'un code LRPC revient donc à calculer des mots de petits poids dans le code dual. ces attaques ont été étudiées dans l'article [44].

Le schéma du cryptosystème est composé des trois algorithmes suivants :

1. **Génération des clés :** on choisit uniformément au hasard un code LRPC $\mathcal{C}$ de type $[n,k]_{q^m}$ de poids d et de matrice de parité $\boldsymbol{H}$.
   Clé secrète : une matrice $\boldsymbol{H}$ de petit poids d.
   Clé publique : la matrice génératrice $\boldsymbol{G}_{syst}$ de $\mathcal{C}$.
2. **Chiffrement :** soit $\boldsymbol{m} \in \mathbb{F}_{q^m}^k$ le message que l'on veut chiffrer. On choisit uniformément au hasard une erreur $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$ de poids w et on envoie le chiffré $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}_{syst} + \boldsymbol{e}$.
3. **Déchiffrement :** on calcule le syndrome $\boldsymbol{s}^T = \boldsymbol{H}\boldsymbol{c}^T = \boldsymbol{H}\boldsymbol{e}^T$ afin de retrouver le vecteur $\boldsymbol{e}$ à l'aide de l'algorithme de décodage des codes LRPC. Puis on résout le système linéaire $\boldsymbol{m}\boldsymbol{G}_{syst} = \boldsymbol{c} - \boldsymbol{e}$ pour obtenir le message $\boldsymbol{m}$.

Afin de réduire la taille de la clé publique, les auteurs de [44] proposent d'utiliser des codes DC-LRPC et donnent les trois jeux de paramètres suivants :

n	k	m	q	d	r	probabilité d'échec	taille de la clé publique	sécurité
82	41	41	2	5	4	2^{-22}	1681 bits	80
106	53	53	2	6	5	2^{-24}	2809 bits	128
74	37	23	2^4	4	4	2^{-88}	3404 bits	100

Les codes LRPC ont aussi été utilisés dans le schéma de signature RankSign présenté dans l'article [46]. L'idée générale de la signature est d'utiliser la structure d'un code pour résoudre n'importe quelle instance du problème RSD à un poids w fixé. La structure du code n'est connue que par le signataire et la description publique du code permet de vérifier la validité d'une signature. On relie de manière déterministe le document à signer à un syndrome d'une instance du problème RSD par une fonction de hachage cryptographique.

La famille de codes utilisée dans l'article est celle des codes LRPC augmentés, noté LRPC$^+$. Cette famille de code est plus vaste que celle des codes LRPC mais reste proche par sa définition.

Définition 2.7.1. *Soient $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ la matrice de parité d'un code LRPC de poids d et $\boldsymbol{R} \in \mathbb{F}_{q^m}^{(n-k)\times t}$ une matrice quelconque. Soient $\boldsymbol{P} \in GL_{n-k}(\mathbb{F}_{q^m})$ et $\boldsymbol{Q} \in GL_{n+t}(\mathbb{F}_q)$ deux matrices inversibles. Soit $\boldsymbol{H'} = \boldsymbol{P}(\boldsymbol{R}|\boldsymbol{H})\boldsymbol{Q}$ la matrice de parité d'un code $\mathcal{C}$ de type $[n+t, k+t]_{q^m}$. Par définition, ce code est un code LRPC$^+$.*

L'intérêt d'utiliser les codes LRPC$^+$ pour la signature est double. Premièrement, l'ajout de t colonnes aléatoires permet de rendre plus difficile la recherche de mots de petit poids dans le code dual, donc de masquer plus efficacement un code LRPC$^+$. Deuxièmement, l'algorithme résolvant le problème RSD diffère de celui de décodage des codes LRPC et les codes LRPC$^+$ sont conçus pour s'adapter à cet algorithme. En effet, pour pouvoir résoudre le problème RSD pour n'importe quel syndrome, il faut que le poids de l'erreur cherchée soit supérieur à la borne de Gilbert-Varshamov de ce code, ce que ne permet pas l'algorithme de décodage. L'algorithme de RankSign est assez technique et une description même succincte dépasse le cadre de cette partie. Le lecteur peut se

référer à l'article [46] pour une description complète. Nous contenterons de présenter le schéma global de signature RankSign.

RankSign est composé de trois algorithmes :

1. **Génération des clés :** on choisit uniformément au hasard un code LRPC$^+$ $\mathcal{C}$ de type $[n+t, k+t]_{q^m}$ de matrice de parité $\boldsymbol{H}' = \boldsymbol{P}(\boldsymbol{R}|\boldsymbol{H})\boldsymbol{Q}$.
   Clé secrète : le triplet de matrices $\boldsymbol{P}$, $(\boldsymbol{R}|\boldsymbol{H})$ et $\boldsymbol{Q}$.
   Clé publique : la matrice $\boldsymbol{H}'$.
2. **Signature :** soit M le message à signer. On calcule $\boldsymbol{s} = hash(M) \in \mathbb{F}_{q^m}^{n-k}$. On utilise la clé secrète pour calculer $\boldsymbol{e}$ de poids r tel que $\boldsymbol{H}'\boldsymbol{e}^T = \boldsymbol{s}^T$. On publie la signature $\boldsymbol{e}$ du message M.
3. **Vérification :** une signature valide $\boldsymbol{e}$ doit vérifier $|\boldsymbol{e}|_r = r$ et $\boldsymbol{H}'\boldsymbol{e}^T = hash(M)$.

La sécurité de RankSign repose sur le problème d'indistingabilité d'un code LRPC augmenté.

Définition 2.7.2 (Ind-LRPC$^+$). *Étant donné un code LRPC$^+$ $\mathcal{C}$ de type $[n+t, k+t]_{q^m}$ de poids d, le problème Ind-LRPC$^+$ consiste à distinguer $\mathcal{C}$ d'un code aléatoire de type $[n+t, k+t]_{q^m}$.*

La difficulté de ce problème est étudiée en détail dans l'article [46].

Deuxième partie

Attaques Structurelles sur les Codes DC-LRPC et Amélioration de la Complexité de l'Attaque Générique

Chapitre 3

Nouvelles attaques sur les codes LRPC doublement circulants

Dans ce chapitre, nous présentons une généralisation de l'algorithme de recherche de mots de petits poids de la partie 2.4.2 introduit dans l'article [55]. Cet algorithme permet d'obtenir une meilleure complexité dans le cas où l'on connaît plusieurs éléments du support d'un mot de petit poids (et non plus un seul élément comme dans le cas général). Dans le cas où $n \geqslant m$, on obtient directement une amélioration en ne modifiant que légèrement l'algorithme. En revanche, dans le cas $m > n$, il est nécessaire d'utiliser les codes matriciels, ce qui complique assez fortement l'algorithme.

3.1 A low weight codeword finding algorithm using additional knowledge on the codeword

In this section, we assume that we have additional knowledge about the codeword of weight w we want to find in the form of linear combinations of its columns. More precisely we are looking for an algorithm whose input and output are specified in Algorithm 3.

The case of a matrix code obtained from an $\mathbb{F}_{q^m}$-linear code is a particular case of such an additional knowledge : we can assume that one of the columns of the codeword we are looking for is the column $(10\dots0)^T$. The folding attack that we present in Section 3.2

Input :
(i) an $[m \times n, k.m]$ matrix code $\mathcal{C}$ over $\mathbb{F}_q$ that has at least one codeword $\boldsymbol{c} = (c_{ij})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}$ of rank weight w
(ii) a elements $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_a$ in $\mathbb{F}_q^m$ that are linear combinations of columns of $\boldsymbol{c}$.
(iii) the coefficients λ_{ij}'s of these linear combinations, that is if we denote by $\boldsymbol{c}_{.,j} = (c_{ij})_{1 \leq i \leq m}$ the j-th column of $\boldsymbol{c}$, then $\boldsymbol{c}'_i = \sum_{j=1}^{n} \lambda_{ij} \boldsymbol{c}_{.,j}$ for $i \in \{1, \ldots, a\}$.
Assumes : $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_a$ are linearly independent.
Output : a codeword of $\mathcal{C}$ of rank weight w.

Algorithme 3 : Low rank codeword finding with additional information

will provide another example where we have the knowledge of two independent linear combinations of the columns and will use in an essential way the algorithm we give here.

3.1.1 The case $n \geqslant m$

We use here a variation of the support trapping algorithm [42]. The case when $a = 1$ and when the matrix code is obtained from an $\mathbb{F}_{q^m}$-linear code is already treated in [42, Prop. 3.1]. Generalizing this argument to the more general setting considered here just consists in choosing in the error trapping algorithm recalled in Section 2.4.2 an F as a random subspace of dimension r that contains the subspace generated by the a elements $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_a$. This leads to the following proposition.

Proposition 3.1.1. *The support trapping algorithm outlined above has expected complexity* $\mathcal{O}\big((n-k)^3 m^3 q^{(w-a)\lceil \frac{km}{n} \rceil}\big)$ *when applied on a matrix code over* $\mathbb{F}_q$ *of type* $[m \times n, k.m]$.

The complexity given follows almost immediately from proposition 2.4.2.

Indeed we look for a support E' which contains the linear space V generated by $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_a$ and we choose F as a random subspace of $\mathbb{F}_q^m$ that contains V. The probability p that $F \supset E'$ is obtained by replacing $\mathbb{F}_q$ by V in the proposition, which gives us

$$p = \frac{\begin{bmatrix} r-a \\ w-a \end{bmatrix}_q}{\begin{bmatrix} m-a \\ w-a \end{bmatrix}_q} = \Theta\big(q^{(w-a)(m-r)}\big)$$

The expected complexity of the support trapping algorithm is now given by the inverse of the probability that we computed multiplied by the complexity of solving a linear system with $(n-k)m$ equations.

3.1.2 The case $m > n$

This will be treated essentially by a variation on the error trapping algorithm applied to the transposed code which uses in a suitable way the additional knowledge about the codeword we want to find. The technical difficulty we face here can be described as follows. If we had additional knowledge about $\boldsymbol{c}$ in the form of a independent elements belonging to the row space of $\boldsymbol{c}$, then we could immediately apply the algorithm given in Section 2.4.1 to the transposed code. However it turns out that in the case we are interested in, the knowledge about $\boldsymbol{c}$ that we have concerns the column space of $\boldsymbol{c}$. In this case, when we transpose $\boldsymbol{c}$ to reverse the role of n and m, this translates into some knowledge of the row space of $\boldsymbol{c}^T$ and we can not use the algorithm of Section 2.4.1 anymore. This is why we are going to consider a slightly more complicated algorithm which is able to use some knowledge on the column space of $\boldsymbol{c}$. It will be essential for our attack that is given in Section 3.2 to work to have an efficient algorithm for finding low-rank codewords by exploiting some knowledge about the low-rank codeword we are looking for. Even if the underlying code is defined for $m < n$ it turns out that we are reducing this problem to another low-rank finding problem in a new code where $m > n$.

This new decoding algorithm is in essence a support trapping algorithm working on the transposed code. It will also be able to use in a simple way additional knowledge about the low rank word we are looking for.

The point is now that by applying a version of the support trapping algorithm of [42] that makes use in a suitable way of the additional knowledge we have about the support. More generally it will have an exponential asymptotic complexity of order $\mathcal{O}\big((n-k)^3 m^3 q^{(w-a)k}\big)$ for an $[m \times n, k.m]$ matrix code over $\mathbb{F}_q$ when we know a independent linear combinations of the columns of the matrix codeword of rank w we are looking for.

This algorithm can be described as follows

Step 1 (transformation of the code) : We first transform the matrix code $\mathcal{C}$ by multiplying it on the right by an $n \times n$ invertible matrix $\boldsymbol{P}$ such that $\boldsymbol{c}$ gets transformed into a

matrix $\boldsymbol{c}'$ whose i first columns are precisely the $\boldsymbol{c}'_i$'s defined before. In other words, we consider the code $\mathcal{C}' = \mathcal{C}\boldsymbol{P}$. If $\boldsymbol{c}$ is a word of rank weight w then $\boldsymbol{c}'$ is still a word of rank weight w. Moreover by assumption on the independence of the $\boldsymbol{c}'_i$'s for $i \in \{1, \ldots, a\}$ we can further multiply $\mathcal{C}'$ on the left by an $m \times m$ invertible matrix $\boldsymbol{Q}$ such that $\boldsymbol{c}'$ gets transformed in a matrix $\boldsymbol{c}$" whose first a columns are the first a elements $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_a$ of the canonical basis of $\mathbb{F}_q^m$, that is $\boldsymbol{e}_i$ has only zero entries with the exception of the i-th entry which is equal to 1. Let $\mathcal{C}$" be the resulting code obtained by these operations, that is

$$\mathcal{C}" = \boldsymbol{Q}\mathcal{C}\boldsymbol{P}$$

Notice that $\boldsymbol{c}$" still has rank w.

Step 2 : (setting up the unknowns of the linear system) We are now basically going to apply a variation of the support trapping algorithm of [42] on $\mathcal{C}"^T$ by choosing a subspace V of $\mathbb{F}_q^n$ of dimension r (r will be specified later on) for which we hope that it contains the subspace generated by the columns of $\boldsymbol{c}"^T$. A basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$ of this space is chosen such that

$$v_{j,i} = 0 \text{ for } i, j \text{ in } \{1, \ldots, a\} \text{ and } i \neq j \tag{3.1}$$

$$v_{i,i} = 1 \text{ for } i \text{ in } \{1, \ldots, a\} \tag{3.2}$$

$$v_{j,i} = 0 \text{ for } i \text{ in } \{a+1, \ldots, r\} \text{ and } j \text{ in } \{1, \ldots, a\} \tag{3.3}$$

where $v_{j,i}$ denotes the j-th coordinate of $\boldsymbol{v}_i$. The entries $v_{j,i}$ are chosen uniformly at random for i in $\{a+1, \ldots, r\}$ and j in $\{a+1, \ldots, n\}$. The entries of $v_{j,i}$ for i in $\{1, \ldots, a\}$ and j in $\{a+1, \ldots, n\}$ will be chosen afterwards. Denote by $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_m$ the m columns of $\boldsymbol{c}"^T$. Let us introduce the $x_{s,t}$'s in $\mathbb{F}_q$ that are such that

$$\boldsymbol{C}_s = \sum_{t=1}^{r} x_{s,t}\boldsymbol{v}_t \text{ for } s \text{ in } \{1, \ldots, m\}. \tag{3.4}$$

Notice now the following point

Lemme 3.1.2. *For $s > a$ and all i in $\{1, \ldots, a\}$ we have $x_{s,i} = 0$. If we denote by $C_{i,j}$ the i'th element of the j-th column $\boldsymbol{C}_j$ of $\boldsymbol{c}"^T$ then $C_{i,j} = 0$ for all i, j in $\{1, \ldots, a\}$ with the exception of the diagonal elements $C_{i,i}$ that are equal to 1.*

Démonstration. Denote by $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_n$ the n rows of $\boldsymbol{c}"^T$. Notice that

$$\boldsymbol{R}_i = \boldsymbol{e}_i, \text{ for } i \text{ in } \{1, \ldots, a\} \tag{3.5}$$

where the $\boldsymbol{e}_i$'s are as before the canonical basis of $\mathbb{F}_q^m$. This implies directly that $C_{i,j} = 0$ for all i, j in $\{1, \ldots, a\}$ with the exception of the diagonal elements $C_{i,i}$ that are equal to 1. Moreover, by using (3.5) together with (3.1),(3.2) and (3.3) we know that $x_{s,i} = 0$ for $s > a$ and all i in $\{1, \ldots, a\}$. □

This motivates us to define as unknowns the $(m-a)(r-a)+a(n-a)$ quantities $x_{s,t}$ and $C_{i,j}$ for s in $\{a+1, \ldots, m\}$, t in $\{a+1, \ldots, r\}$, i in $\{a+1, \ldots, n\}$ and j in $\{1, \ldots, a\}$. Moreover these unknowns satisfy $nm - km = (n-k)m$ linear equations obtained from the fact $\boldsymbol{c}"^T$ belongs to $\mathcal{C}"^T$ which is a matrix code of dimension km. They can be obtained by computing a parity-check matrix of this code, then expressing the linear equations that the entries of $\boldsymbol{c}"^T$ have to satisfy and then replacing these entries by the aforementioned unknowns by using (3.4) and Lemma 3.1.2. We choose r such that the number of equations, that is $(n-k)m$ is at least equal to the number of unknowns, that is

$$(n-k)m \geq (m-a)(r-a) + a(n-a)$$

This can be obtained by choosing

$$r \stackrel{\text{def}}{=} \left\lfloor \frac{m}{m-a}(n-k) + a\frac{m-n}{m-a} \right\rfloor$$

Step 3 : (solving the linear system) The last point just consists in solving the linear system, this yields $\boldsymbol{c}"^T$ and from this we deduce $\boldsymbol{c}"$ and then $\boldsymbol{c}$ by

$$\boldsymbol{c} = \boldsymbol{Q}^{-1}\boldsymbol{c}"\boldsymbol{P}^{-1}$$

The last point to understand is under which condition V contains the subspace generated by the columns of $\boldsymbol{c}"^T$. This depends on how we specify the entries $v_{j,i}$ for i in $\{1, \ldots, a\}$ and j in $\{a+1, \ldots, n\}$. We choose them such that (3.4) is verified for s in $\{1, \ldots, a\}$. This can obviously be done by choosing

$$\boldsymbol{v}_i = \boldsymbol{C}_i \text{ for } i \in \{1, \ldots, a\} \tag{3.6}$$

Lemme 3.1.3. *Let V be chosen by a basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$ such that its a first elements are given by (3.6) and as specified in Step 2 for the other elements. Let W be the subspace generated by the columns of $\boldsymbol{c}"^T$. Let W_0 be the subspace of W that is formed by the elements whose first a entries are all equal to 0. In the same way, we denote by V_0 the subspace that is formed by the elements of V whose first a entries are all equal to 0. We have $W \subset V$ iff $W_0 \subset V_0$.*

Démonstration. It is clear that $W \subset V$ implies $W_0 \subset V_0$.

Now assume that $W_0 \subset V_0$. Notice that W is generated by W_0 and by the first a columns of $\boldsymbol{c}$", that is $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_a$. Since V is generated by the same first a columns of $\boldsymbol{c}$", $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_a$ and by V_0 we have that $W \subset V$. □

Putting all these considerations together we obtain that

Théorème 3.1.4. *Let $\mathcal{C}$ be an $[m \times n, k.m]$ matrix code which has at least one codeword of rank weight w for which we know a independent linear combinations of its columns as specified in Algorithm 3. Assume that $n \leq m$ and let $r \stackrel{def}{=} \left\lfloor \frac{m}{m-a}(n-k) + a\frac{m-n}{m-a} \right\rfloor$. Then the algorithm given in this section outputs a codeword of weight w with complexity $\mathcal{O}\left((n-k)^3 m^3 q^{(w-a)(n-r)}\right)$.*

Démonstration. This follows immediately from Lemma 3.1.3 and Proposition 2.4.2 that show that we will try an expected number of $\dfrac{\begin{bmatrix} n-a \\ w-a \end{bmatrix}_q}{\begin{bmatrix} r-a \\ w-a \end{bmatrix}_q} = \Theta\left(q^{(w-a)(n-r)}\right)$ spaces V before finding the right one if there is only one codeword $\boldsymbol{c}$ which has the right form. This is of course an upper bound if there are more than one codeword that have the right form. Each try of a tentative space V takes time $\mathcal{O}\left((n-k)^3 m^3\right)$ whose complexity is dominated by Step 3 when we solve a linear system with $(n-k)m$ equations and a number of unknowns that is less than the number of equations. □

3.2 Folding and projecting attack

In this section we present a key recovery attack on the LRPC cryptosystem [40].

3.2.1 Folded and projected codes

We present here two new ingredients of the attacks that follow, namely the notion of folded code and the notion of projected code. The first attack uses only folding but the second attack uses both. The notion of projected codes uses the polynomial framework

for dealing with quasi-cyclic codes [62, 59]. Quasi-cyclic codes are a generalization of double-circulant codes : they are defined by a parity check matrix formed only from circulant blocks. Such a quasi-cyclic code of length $N = \ell n$ defined over a finite field $\mathbb{K}$, where the size of the circulant blocks is n, can also be viewed as code over the ring $\mathbb{K}[X]/(X^n - 1)$. This is a specific instance of cellular codes that are codes defined over a ring $\mathcal{R} = \mathbb{K}[X]/(f(X))$ where $f(X)$ is a polynomial of $\mathbb{K}[X]$.

More generally we consider codes over a finite field $\mathbb{K}$ derived from codes defined over a ring

$$\mathcal{R} \stackrel{\text{def}}{=} \mathbb{K}[X]/(f(X))$$

where f is some polynomial in $\mathbb{K}[X]$ of degree n. They are derived from the following $\mathbb{K}$-isomorphism $\psi : \mathcal{R} \to \mathbb{K}^n$:

$$a(X) = \sum_{i=0}^{n-1} a_i X^i \mapsto \psi(a(X)) = (a_0, \ldots, a_{n-1}).$$

They are called cellular codes and are defined by

Définition 3.2.1 (cellular code). *Consider a submodule M of $\mathcal{R}^\ell$ of rank s. Let $\psi^\ell : \mathcal{R}^\ell \to \mathbb{K}^{\ell n}$ that maps an element $(f_1, \ldots, f_l)$ of $\mathcal{R}^\ell$ to $\mathbb{K}^{\ell n}$ by mapping each f_i to $\psi(f_i)$. The cellular code associated to M is given by $\psi^\ell(M)$. It is said to have index ℓ and it is a $\mathbb{K}$-linear code of length ℓn.*

Remark 3.2.2. *In order to avoid cumbersome notation, we identified M with $\psi^\ell(M)$ in Section 3.2. Sometimes it will better to view the cellular code $\psi^\ell(M)$ as M and we will freely do this.*

To obtain a generator matrix of the cellular code from a generator matrix

$$\boldsymbol{G}_M = \begin{pmatrix} a_{1,1}(X) & \ldots & a_{1,\ell}(X) \\ \vdots & \ddots & \vdots \\ a_{s,1}(X) & \ldots & a_{s,\ell}(X) \end{pmatrix}$$

of the rank s-submodule we introduce the following mapping $\phi : \mathcal{R} \to \mathbb{K}^{n \times n}$:

$$a(X) = \sum_{i=0}^{n-1} a_i X^i \mapsto \phi(a(X)) = \begin{pmatrix} \psi(a(X)) \\ \psi(Xa(X)) \\ \vdots \\ \psi(X^{n-1}a(X)) \end{pmatrix}$$

It is a bijective morphism of $\mathbb{K}$-algebras. When $f(X) = X^n - 1$ this is precisely the mapping that appears in Proposition 2.6.6. A generator matrix of the associated cellular code is now given by

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{A}_{1,1} & \dots & \boldsymbol{A}_{1,\ell} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{s,1} & \dots & \boldsymbol{A}_{s,\ell} \end{pmatrix}$$

where $\boldsymbol{A}_{i,j} = \phi(a_{i,j}(X))$. This implies that the dimension k of the cellular code satisfies $k \leq ns$.

Définition 3.2.3 (projected code). *Consider a cellular code $\mathcal{C}$ of index ℓ defined over $\mathcal{R} \stackrel{def}{=} \mathbb{K}[X]/(f(X))$ and let $g(X)$ be a divisor of $f(X)$ in $\mathbb{K}[X]$. The projected cellular code $\overline{\mathcal{C}}^g$ is obtained by viewing a codeword $\boldsymbol{c}$ of $\mathcal{C}$ as an element of R^l : $\boldsymbol{c} = (\boldsymbol{c}_1, \dots, \boldsymbol{c}_\ell)$ and applying the surjective morphism Π from $\mathbb{K}[X]/(f(X))$ to $\mathbb{K}[X]/(g(X))$ defined by $\Pi(a(X)) = a(X) \pmod{g(X)}$ to every entry $\boldsymbol{c}_i$.*

In the particular case where $f(X) = X^n - 1$ and $g(X) = X^m - 1$ where m is a divisor of n, projecting corresponds to folding in the sense of [32, 33].

Définition 3.2.4 (folded code). *Consider a quasi-cyclic code $\mathcal{C}$ of index ℓ and length $n\ell$. Let m be a divisor of n. Its folded code of order m is a quasi-cyclic code of index ℓ and length $m\ell$ obtained by mapping each codeword $\boldsymbol{c} = (c_0, \dots, c_{n\ell-1})$ of $\mathcal{C}$ to the codeword $\boldsymbol{c}' = (c'_0, \dots, c'_{m\ell-1})$ where*

$$c'_i = \sum_{s=0}^{\frac{n}{m}-1} c_{an+b+sm}$$

and a and b are the quotient and the remainder of the euclidean division of i by m :

$$i = am + b \text{ with } a \text{ and } b \text{ integer and } b \in \{0, \dots, m-1\}.$$

This really amounts to sum the coordinates that belong to the same orbit of a (permutation) autorphism of order n/m that leaves the quasi-cyclic code invariant.

There are two points which make these two notions very interesting in the cryptographic setting. The first point is that these two reductions of the code do not lead to a trivial code at the end (one could have feared to end up with the full space after projecting or folding). This comes from the following proposition that is proved in [64, Corollary 1]

Proposition 3.2.5. *Consider a cellular code $\mathcal{C}$ of index ℓ defined over $\mathcal{R} \stackrel{def}{=} \mathbb{K}[X]/(f(X))$ and let $g(X)$ be a divisor of $f(X)$ in $\mathbb{K}[X]$. The length of the projected code $\overline{\mathcal{C}}^g$ is $\ell \deg g$ whereas the dimension of $\overline{\mathcal{C}}^g$ is less than or equal to $s\ell$ where s is the rank of the cellular code.*

The second point is that this operation of folding behaves nicely with respect to projecting a quasi-cyclic code defined over an extension field $\mathbb{F}_{q^m}$ with respect to the rank distance over $\mathbb{F}_q$ when the divisor g belongs to $\mathbb{F}_q[X]$

Proposition 3.2.6 ([64, Prop.3])**.** *Consider a cellular code $\mathcal{C}$ of index ℓ defined over $\mathcal{R} \stackrel{def}{=} \mathbb{F}_{q^m}[X]/(f(X))$ and let $g(X)$ be a divisor of $f(X)$ in $\mathbb{F}_q(X)$. Denote by Π the associated projection operation. We have*

$$\mathrm{Rank}(\Pi(\boldsymbol{c})) \leq \mathrm{Rank}(\boldsymbol{c})$$

for any $\boldsymbol{c} \in \mathcal{C}$ where we view these codewords as matrices in $\mathbb{F}_q^{m \times \ell \deg f}$ or in $\mathbb{F}_q^{m \times \ell \deg g}$ by taking the matrix form of these codewords as defined in Section 2.2.

Notice that this proposition can always be applied to folded codes. These two propositions allow to search for a codeword $\boldsymbol{c}$ of rank w in a quasi-cyclic code $\mathcal{C}$ of index ℓ and length $n\ell$ defined over $\mathbb{F}_{q^m}$ by projecting it with respect to a divisor of $X^n - 1$ that belongs to $\mathbb{F}_q[X]$ (or by folding it) and looking for a word of rank $\leq w$ in the projected or folded code. Roughly speaking, the first proposition ensures that we are not looking for a word w in the entire space. From the second proposition, we expect that as long w is below the Gilbert-Varshamov bound of the folded code, the codeword of weight $\leq w$ we will find in the projected code corresponds to the projection of $\boldsymbol{c}$. This allows to recover easily $\boldsymbol{c}$.

3.2.2 A first attack based on folding

Let $\mathcal{C}$ be a DC-LRPC $[2k, k]$ code of weight d over $\mathbb{F}_{q^m}$ obtained from a parity-check matrix $\boldsymbol{H}$. To recover $\boldsymbol{H}$ it is clearly sufficient to find a codeword of rank weight d in the dual $\mathcal{C}^\perp$ of $\mathcal{C}$. Let $\mathcal{C}'$ be the folding of order 1 of $\mathcal{C}^\perp$.

It is in general a $[2, 1]$ code. This folding reveals some additional information about the subspace F of $\mathbb{F}_{q^m}$ generated by the coefficients of H. We namely have

Proposition 3.2.7. *Let $\boldsymbol{c}' = (c'_1, c'_2)$ be in $\mathcal{C}'$. There exists $\boldsymbol{c}$ of weight d in $\mathcal{C}^\perp$ such that the $\mathbb{F}_q$-subspace generated by the coordinates of $\boldsymbol{c}$ contains c'_1 and c'_2.*

Démonstration. If $\mathcal{C}'$ is the all-zero code or $\boldsymbol{c}' = 0$ the conclusion follows directly.

Assume now that this is not the case. In this case, $\mathcal{C}'$ is of dimension 1. Consider a codeword $\boldsymbol{c}$ of $\mathcal{C}^\perp$ which is of weight d. Let $\boldsymbol{c}$" be the folded version of $\boldsymbol{c}$. We have in this case $\boldsymbol{c}' = \alpha\boldsymbol{c}$" for some $\alpha \in \mathbb{F}_{q^m}^\star$. Note that $\boldsymbol{c}'$ is the folded version of $\alpha\boldsymbol{c}$. We observe now two things and this finishes the proof

- $d = \mathrm{Rank}(\boldsymbol{c}) = \mathrm{Rank}(\alpha\boldsymbol{c})$ where we view these codewords as matrices in $\mathbb{F}_q^{m\times n}$ as explained in Section 2.2.
- c'_1 and c'_2 are in the $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ generated by the coordinates of $\alpha\boldsymbol{c}$.

□

We can use (c'_1, c'_2) in the decoding algorithm described in Section 3.1. From $\boldsymbol{c}$ we recover immediately a parity-check matrix of the form $\beta\boldsymbol{H}$, where $\beta \in \mathbb{F}_{q^m}\backslash\{0\}$, by building a parity-check matrix from $\boldsymbol{c}$ and its cyclic shifts. This gives an attack of complexity $\mathcal{O}\big(k^3m^3q^{(d-2)\lceil\frac{m}{2}\rceil}\big)$. However for the parameters proposed in [40, 44], this does not improve the attacks already considered there. However, this proposition together with another projection of the code will lead to a feasible attack against a certain parameter of [40, 44] as we now show.

3.2.3 An improved attack based on folding and projecting

To improve the attack, we search for a word of weight d in a projected code. This new attack depends on the factorization of $X^k - 1$. The length of the projected code we are interested in will be smaller than m and we will use the algorithm of Subsection 3.1.2 instead. The attack can be described as

Step 1 : Compute $\mathcal{C}'$ the folding of order 1 of $\mathcal{C}^\perp$ and extract a codeword (c'_1, c'_2) in it.
Step 2 : Compute the projected code $\overline{\mathcal{C}^\perp}^D$ with respect to a certain divisor $D(X)$ of $X^k - 1$ in $\mathbb{F}_q[X]$.
Step 3 : Find a codeword $\boldsymbol{c}$" in $\overline{\mathcal{C}^\perp}^D$ of weight w such that the $\mathbb{F}_q$ space generated by its coordinates contains c'_1 and c'_2 by using the algorithm of Subsection 3.1.2.
Step 4 : Let F be the $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ generated by the coordinates of $\boldsymbol{c}$". Find the

codeword $\boldsymbol{c}$ in $\mathcal{C}^\perp$ of rank weight w whose support is F (meaning that the $\mathbb{F}_q$-subspace generated by its coordinates should belong to F.)

What justifies the third step is the fact that Proposition 3.2.7 generalizes easily to the projected code, whereas what justifies Step 4 is the fact that it is extremely likely that $\boldsymbol{c}$" is the projection of a codeword in $\mathcal{C}^\perp$ of weight d we are looking for. We recover in this case such a codeword by the process of Step 4. The complexity of this attack is dominated by the third step and is given by Theorem 3.1.4.

In [40], some parameters for the LRPC cryptosystem are suggested. They are recalled in the following table

n	k	m	q	d	security
74	37	41	2	4	80
94	47	47	2	5	128
68	34	23	2^4	4	100

In each case the factorization of $X^k - 1$ in $\mathbb{F}_q[X]$ is given by
(i) $X^{37} - 1 = (X - 1)\sum_{i=0}^{36} X^i$
(ii) $X^{47} - 1 = (X - 1)PQ$ with $\deg P = \deg Q = 23$
(iii) $X^{34} - 1 = (X - 1)^2(P_1 \dots P_8)^2$ with $\deg P_i = 2$, for all $i \in [\![1; 8]\!]$.

In the first case, the polynomial $X^{37}-1$ has only two divisors, so we can only use the first attack. In the second case, we can choose $D = P$ or Q to obtain a folded code of dimension 23. According to Theorem 3.1.4, the complexity of the attack is $\mathcal{O}\big(23^3 47^3 2^{3\times 22)}\big) \approx 2^{96.2}$, that is a gain around 2^{32} compared to the best attack considered in [40] and about 2^{20} compared to the best attack found in [64, Subsec. 3.2].

The third case is the most interesting. Here we can freely choose the dimension of the projected code. Keep in mind that we want the Gilbert-Varshamov bound greater than d which is the case when the dimension k" of the projected code is $\geqslant 4$. We choose $k" = 4$ and we have in this case an attack of complexity $2^{43.6}$ which clearly leads to a feasible attack.

In [44], a new set of parameters is proposed, as follows :

n	k	m	q	d	security
82	41	41	2	5	80
106	53	53	2	6	128
74	37	23	2^4	4	100

In each case the factorization of $X^k - 1$ in $\mathbb{F}_q[X]$ is given by
(i) $X^{41} - 1 = (X-1)PQ$ with $\deg P = \deg Q = 20$
(ii) $X^{53} - 1 = (X-1)\sum_{i=0}^{52} X^i$
(iii) $X^{37} - 1 = (X-1)P_1 \dots P_4$ with $\deg P_i = 9$, for all $i \in [\![1; 4]\!]$.

The first case allow a non-trivial projection but it is not sufficient to obtain a better complexity than 80. In the second case, we can only use the folding attack, and its complexity is greater than 128.

In the third case, we can choose a projected code of dimension 9 and we have an attack of complexity $2^{87,1}$, that is a gain around 2^{13}.

As we can see, it is crucial to choose k such that $X^k - 1$ has the minimum of factors in $\mathbb{F}_q[X]$, it can always be factorisable in $(X-1)(\sum_{i=0}^{k-1} X^i)$ so one have to choose $\sum_{i=0}^{k-1} X^i$ irreducible in $\mathbb{F}_q$. This implies k prime but it is not sufficient, as the third case of the parameters proposed in [44] proves it.

Chapitre 4

Amélioration de l'attaque générique du problème RSD

Dans ce chapitre, nous présentons dans une première partie une amélioration de l'algorithme GRS [42] décrit dans la section 2.4.1. Dans une seconde partie, nous étudierons le gain qu'apportent les algorithmes quantiques.

4.1 Nouvelle attaque générique du problème RSD

Rappelons d'abord le problème RSD 2.3.1 que l'on cherche à résoudre et l'idée générale de l'algorithme GRS. Soit $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ la matrice de parité d'un code $\mathcal{C}$ de type $[n,k]_{q^m}$ et $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ un vecteur de poids w. Le but est de retrouver $\boldsymbol{x}$ en ne connaissant que $\boldsymbol{H}$ et le syndrome $\boldsymbol{s} = \boldsymbol{x}\boldsymbol{H}^T$ de $\boldsymbol{x}$.

L'idée de l'algorithme GRS est de rechercher le support E de $\boldsymbol{x}$ en supposant que celui-ci est inclus dans un sous-espace F de $\mathbb{F}_{q^m}$ de dimension $r = m - \left\lceil \frac{km}{n} \right\rceil$ puis en résolvant un système linéaire découlant des équations de parité. Pour un code $[n,k]_{q^m}$, le nombre d'équations sur $\mathbb{F}_q$ est de $(n-k)m$. Si l'hypothèse $E \subset F$ est vérifiée, le système admet une solution qui permet de retrouver $\boldsymbol{x}$, sinon on teste un autre espace F.

En moyenne, la complexité C de l'algorithme est égale à l'inverse de la probabilité que F contienne E fois le coût de résolution du système linéaire. Cette probabilité est égale

à $\dfrac{\begin{bmatrix} r \\ w \end{bmatrix}_q}{\begin{bmatrix} m \\ w \end{bmatrix}_q} = \Theta(q^{-w(m-r)})$ ce qui nous donne la formule suivante :

$$C = \mathcal{O}\big((n-k)^3 m^3 q^{w\lceil \frac{km}{n} \rceil}\big)$$

Il est donc très important d'améliorer la probabilité de trouver le support E car cela peut permettre de gagner une facteur exponentiel.

Une première amélioration de l'algorithme est de se ramener à la recherche de mots de petits poids dans un code contenant $\boldsymbol{x}$ et $\mathcal{C}$. Soit $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$ tel que $\boldsymbol{y}\boldsymbol{H}^T = \boldsymbol{s}$. Soit $\mathcal{C}'$ le code engendré par $\boldsymbol{y}$ et $\mathcal{C}$:

$$\mathcal{C}' = \mathcal{C} + \mathbb{F}_{q^m}\boldsymbol{y}$$

Par construction, $\boldsymbol{x} \in \mathcal{C}'$ et par $\mathbb{F}_{q^m}$-linéarité, $\alpha\boldsymbol{x} \in \mathcal{C}'$ pour tout $\alpha \in \mathbb{F}_{q^m}$. L'idée est de rechercher l'un de ces multiple $\boldsymbol{x}'$ de $\boldsymbol{x}$, puis de résoudre l'équation $\boldsymbol{x}'\boldsymbol{H}^T = \beta\boldsymbol{s}$ d'inconnue β pour obtenir $\boldsymbol{x}$.

Une première méthode, décrite dans la section 2.4.2 et dans l'article d'origine [42], est de rechercher $\boldsymbol{x}'$ tel que $1 \in \mathrm{Supp}(\boldsymbol{x}') = E'$ en ne testant que les espaces F de dimension $r' = m - \left\lceil \frac{(k+1)m}{n} \right\rceil$ contenant 1. Dans ce cas, la probabilité que F contienne E' sachant que $1 \in F$ et $1 \in E$ vaut $\dfrac{\begin{bmatrix} r'-1 \\ w-1 \end{bmatrix}_q}{\begin{bmatrix} m-1 \\ w-1 \end{bmatrix}_q} = \Theta(q^{-(w-1)(m-r')})$. La complexité moyenne de l'attaque est donc

$$C' = \mathcal{O}\big((n-k)^3 m^3 q^{(w-1)\lceil \frac{(k+1)m}{n} \rceil}\big)$$

La méthode que nous présentons ici consiste à choisir F aléatoirement comme dans le premier algorithme. Si F contient un multiple $\alpha E, \alpha \in \mathbb{F}_{q^m}$ du support E, alors nous pouvons calculer le mot de code $\alpha\boldsymbol{x}$ de $\mathcal{C}'$. Il faut donc calculer la probabilité p que F contienne un espace vectoriel de cette forme. Il y en a au plus $\frac{q^m-1}{q-1}$ car $\alpha E = \beta E$ si $\alpha/\beta \in \mathbb{F}_q^*$. La proposition suivante montre que cette égalité est très souvent vérifiée.

Proposition 4.1.1. *Soit E un $\mathbb{F}_q$-sous-espace de $\mathbb{F}_{q^m}$ de dimension w et $\alpha \in \mathbb{F}_{q^m}^*$ tel que $E = \alpha E$. Alors E est un $\mathbb{F}_q(\alpha)$-espace vectoriel et $[\mathbb{F}_q(\alpha : \mathbb{F}_q]$ divise w. En particulier, si $w \wedge m = 1$, $\alpha \in \mathbb{F}_q^*$.*

Démonstration. Soit $m' = [\mathbb{F}_q(\alpha) : \mathbb{F}_q]$. Puisque $\alpha E = E$, on a par récurrence immédiate $\mathbb{F}_q(\alpha)E = E$ donc E est un $\mathbb{F}_q(\alpha)$-sous-espace vectoriel de $\mathbb{F}_{q^m}$. D'où

$$w = \dim_{\mathbb{F}_q} E = m' \dim_{\mathbb{F}_q(\alpha)} E$$

ce qui prouve le premier point.

De plus, m' divise m ce qui implique que m' divise $m \wedge w$. Si $w \wedge m = 1$, alors $m' = 1$ ce qui prouve le second point. □

La condition $w \wedge m = 1$ est facilement obtenable, il suffit en particulier que m soit premier, ce qui est le cas dans de nombreux cryptosystèmes. Dans le cas où $w \wedge m = m' > 1$, on peut calculer la probabilité qu'un $\mathbb{F}_q$-espace vectoriel de dimension w soit un $\mathbb{F}_{q^{m'}}$-espace vectoriel de dimension $\frac{w}{m'}$ par un argument de dénombrement.

Le nombre de sous-espaces vectoriels de $\mathbb{F}_{q^m}$ de dimension $\frac{w}{m'}$ sur $\mathbb{F}_{q^{m'}}$ est $\begin{bmatrix} m/m' \\ w/m' \end{bmatrix}_{\mathbb{F}_{q^{m'}}} = \Theta(q^{w\frac{m-w}{m'}})$.

Le nombre de sous-espaces vectoriels de $\mathbb{F}_{q^m}$ de dimension w sur $\mathbb{F}_q$ est $\begin{bmatrix} m \\ w \end{bmatrix}_q = \Theta(q^{w(m-w)})$.

La probabilité recherchée vaut donc $\Theta(q^{-w(m-w)(1-\frac{1}{m'})})$, ce qui est négligeable.

Nous supposerons donc que le nombre d'espaces vectoriels différents de forme $\alpha E, \alpha \in \mathbb{F}_{q^m}$ est $\frac{q^m-1}{q-1}$.

On peut alors approximer la probabilité p par le produit du nombre d'espaces vectoriels de la forme αE par la probabilité que F contienne un sous-espace fixé de dimension w. Cette approximation est correcte si on a

$$\frac{q^m - 1}{q - 1} \begin{bmatrix} r \\ w \end{bmatrix}_q \ll \begin{bmatrix} m \\ w \end{bmatrix}_q \iff q^{m+w(r-w)} \ll q^{w(m-w)}$$

Le reste de l'algorithme est identique à l'algorithme GRS. On choisit r de façon à avoir plus d'équations de parité que d'inconnues, ce qui implique $r \leqslant \left\lfloor \frac{m(n-k-1)}{n} \right\rfloor = m - \left\lceil \frac{(k+1)m}{n} \right\rceil$. On obtient ainsi une complexité en moyenne $\mathcal{O}\big((n-k)^3 m^3 q^{w\lceil \frac{(k+1)m}{n} \rceil - m}\big)$.

Dans le cas $m \leqslant n$, cette stratégie est toujours plus efficace que la méthode consistant à choisir un sous-espace F contenant 1. Le gain est de l'ordre $q^{m-\left\lceil \frac{(k+1)m}{n} \right\rceil} = q^{m-\lceil mR' \rceil}$ où R' est le taux de $\mathcal{C}'$. Dans le cas où $m > n$, cette stratégie peut aussi apporter un gain, comme nous allons le voir dans la partie suivante.

4.1.1 Application de l'attaque à des cryptosystèmes existants

Dans cette sous-section, nous étudions les conséquences de cette nouvelle attaque sur la sécurité de du cryptosystème de McEliece [66]. On rappelle que la sécurité des messages est basée sur la difficulté de décoder un code aléatoire. La nouvelle attaque diminue donc la sécurité des schémas existants.

Dans [65], l'auteur propose d'utiliser des codes de Gabidulin . Le tableau suivant donne des exemples de paramètres utilisés, avec la sécurité et la complexité de cette nouvelle attaque. Les paramètres sont :

- n : la longueur du code.
- k : la dimension du code.
- m : le degré de l'extension du corps $\mathbb{F}_2$
- w : poids de l'erreur.

n	k	m	w	sécurité annoncée	complexité de l'attaque
50	50	32	3	81	75
64	40	96	4	139	177
112	80	112	4	259	243

Comme on peut le constater, la nouvelle attaque casse les jeux de paramètres 1 et 3.

Dans [44], les auteurs utilisent les codes LRPC doublement circulant. Contrairement à l'attaque structurelle précédente, nous considérons la complexité de l'attaque sur les messages et non sur la clé publique.

q	n	k	m	w	sécurité annoncée	complexité de l'attaque
2	82	41	41	4	80	75
2	106	53	53	5	128	116
2^4	74	37	23	4	110	77

Pour tous les paramètres proposés, m est inférieur à n, si bien que la nouvelle est plus performante que la sécurité annoncée dans tous les cas.

4.1.2 Analyse de la complexité asymptotique de l'attaque et perspectives

Il est également important de considérer le comportement asymptotique de la complexité de la nouvelle attaque, afin d'évaluer l'évolution de la taille des clés des cryptosystèmes en fonction de la puissance de calcul brute des ordinateurs, et de le comparer à celui des attaques déjà existantes.

On se place dans le cas classique d'un code de taux $1/2$ ($k = \frac{n}{2}$) avec $m = n$. Les deux problèmes que l'on considère sont le calcul de la distance minimum d'un code aléatoire, c'est-à-dire d'un mot de poids $w = RGV(n, \frac{n}{2}, n, q)$ et le décodage d'une erreur de poids $w/2$ (pour avoir unicité de la solution). Pour ces paramètres, on a

$$RGV \sim_{n\to+\infty} n\left(1 - \sqrt{\frac{1}{2}}\right) \simeq 0.293n$$

On obtient donc :

- pour l'algorithme GRS, une complexité moyenne de $\mathcal{O}\left(n^6 q^{0.073n^2}\right)$ pour le décodage et de $\mathcal{O}\left(n^6 q^{0.146n^2}\right)$ pour la recherche d'un mot de poids minimum.
- pour la nouvelle attaque, on obtient une complexité moyenne de $\mathcal{O}\left(n^6 q^{0.073n^2-n}\right)$ pour le décodage et de $\mathcal{O}\left(n^6 q^{0.146n^2-n}\right)$ pour la recherche d'un mot de poids minimum.

Comme on peut le constater, bien que le gain soit exponentiel, le terme dominant dans l'exposant est identique. La question se pose de savoir s'il est possible de diminuer le coefficient du terme dominant de l'exposant, notamment en adaptant les améliorations de la métrique de Hamming au cas de la métrique rang. Ces recherches ont été effectuées dans le cadre de cette thèse, cependant en raison de la nature différente du support en métrique rang, on ne connaît pas d'équivalent au paradoxe des anniversaires en métrique rang. Sans nouvelle technique, il semble difficile d'améliorer cette complexité.

4.2 Algorithmes quantiques en métrique rang

In this section we evaluate the complexity of solving the rank (metric) syndrome decoding problem with a quantum computer. We will use for that a slight generalization of Grover's quantum search algorithm [51, 53] given in [18] what we will use in the following form. We will use the NAND circuit model as in [9], which consists in a directed acyclic graph where each node has two incoming edges and computes the NAND of its predecessors.

Théorème 4.2.1. *[18] Let f be a Boolean function $f : \{0,1\}^b \to \{0,1\}$ that is computable by a NAND circuit of size S. Let p be the proportion of roots of the Boolean function*

$$p \stackrel{def}{=} \frac{\#\{x \in \{0,1\}^b : f(x) = 0\}}{2^b}.$$

Then there is a quantum algorithm based on iterating a quantum circuit $\mathcal{O}\left(\frac{1}{\sqrt{p}}\right)$ many times that outputs with probability at least $\frac{1}{2}$ one of the roots of the Boolean function. The size of this circuit is $\mathcal{O}\left(S\right)$.

Basically this tool gives a quadratic speed-up when compared to a classical algorithm. Contrarily to what happens for the Hamming metric [9], where using this tool does not yield a quadratic speed-up over the best classical decoding algorithms, the situation is here much clearer : we can divide the exponential complexity of the best algorithms by two. The point is that the algorithms of [42] and [55] can be viewed as looking for a linear subspace which has the right property, where linear spaces with appropriate parameters are drawn uniformly at random and this property can be checked in polynomial time. The exponential complexity of these algorithms is basically given by $\mathcal{O}\left(\frac{1}{p}\right)$ where p is the fraction of linear spaces that have this property. More precisley we have

$$\frac{1}{p} = \mathcal{O}\left(q^{(w-1)(k+1)}\right)$$

for $m > n$, see [55]) and

$$\frac{1}{p} = \mathcal{O}\left(q^{(w-1)\lfloor\frac{(k+1)m}{n}\rfloor}\right)$$

when $m \leq n$, see [42]. Checking whether the linear space has the right property can be done by
(i) solving a linear system with $(n-k-1)m$ equations and with about as many unknowns

over $\mathbb{F}_q$,
(ii) checking whether a matrix over $\mathbb{F}_q$ of size $r \times r'$ is of rank equal to w where $(r, r') = (m - \lceil \frac{(k+1)m}{n} \rceil, n)$ in the case $m \leq n$ and $(r, r') = (n - k - 1, m)$ in the case $m > n$.

If we view q as a fixed quantity, there is a classical NAND circuit of size $\mathcal{O}\big((n-k)^3 m^3\big)$ that realizes these operations. In other words, by using Theorem 4.2.1 we obtain

Proposition 4.2.2. *For fixed q, there is a quantum circuit with $\mathcal{O}\big((n-k)^3 m^3\big)$ gates that solves the rank metric syndrome decoding problem in time $\mathcal{O}\big((n-k)^3 m^3\big) q^{(w-1)(k+1)/2}$ when $m > n$ and in time $\mathcal{O}\big((n-k)^3 m^3 q^{(w-1)\lceil \frac{(k+1)m}{n} \rceil/2}\big)$ when $m \leq n$.*

Troisième partie

Nouveaux Protocoles à Base de Codes en Métrique Rang

Chapitre 5

RankSynd : un PRNG prouvé sûr basé sur les codes en métrique rang

5.1 Introduction

Pseudo-random number generators (PRNG) are an essential tool in cryptography. They can be used for one-time cryptography or to generate random keys for cryptosystems. A long series of articles have demonstrated that the existence of a PRNG is equivalent to the existence of one-way functions [82, 60, 54]. Basically, a one-way function is a function which is easy to compute but hard to invert.

There are two types of PRNG in cryptography. The first one is based on block cipher schemes, like AES for instance, used in OFB mode. This gives in general very fast random generators. The second type includes PRNG proven to be secure by reduction to a hard problem. The problems considered can be based on classical problems from cryptography, like factorization or discrete logarithm, [12, 11] or they may be based on linear algebra, like coding theory [34] or lattices [3] or multivariate quadratic systems [6].

Recent works [39, 67] have proven that PRNG based on the syndrome decoding (SD) problem could be almost as fast as PRNG based on AES. However the PRNG based on the SD problem have to store huge matrices. This problem can be solved with the use of quasi-cyclic codes but there is currently no proof of the hardness of the SD problem for quasi-cyclic codes. Moreover recent quantum attacks on special ideal lattices [25], clearly raise the issue of the security of quasi-cyclic structures for lattices and codes,

even if a straight generalization of this quantum attack from cyclic structures to quasi-cyclic structures seems currently out of reach.

5.2 One-way functions based on rank metric

We use here the hardness of the RSD problem to build a family of one-way functions based on this problem. Let us start by recalling the definition of a strongly one-way function (see [34, Definition 1]) :

Définition 5.2.1. *A collection of functions* $\{f_n : E_n \to \mathbb{F}_2^{k_n}\}$ *is called strongly one way if :*

- *there exists a polynomial-time algorithm which computes* $f_n(x)$ *for all* $x \in E_n$
- *for every probabilistic polynomial-time algorithm* A*, for all* $c > 0$ *and for sufficiently large* n*,* $Prob\big(A(f_n(x)) \in f_n^{-1}(f_n(x))\big) < \dfrac{1}{n^c}$

We will consider the following family :
$E_{n,k} = \{(\boldsymbol{H}, \boldsymbol{y}) : \boldsymbol{H} \in \mathbb{F}_{q^n}^{(n-k)\times n}, \boldsymbol{y} \in \mathbb{F}_{q^n}^n, w_R(\boldsymbol{y}) = w_n\}$

$$\begin{aligned} f : \quad E_{n,k} &\to \mathbb{F}_{q^n}^{(n-k)\times(n+1)} \\ (\boldsymbol{H}, \boldsymbol{y}) &\mapsto (\boldsymbol{H}, \boldsymbol{H}\boldsymbol{y}^T) \end{aligned}$$

We take $m = n$ so that the first algorithm of [42] does not improve the complexity of [71]. These functions should be strongly one-way if we choose $w_n \approx d_{GV}(n, k)$ which corresponds to the range where there is basically in general a unique preimage.

5.3 A PRNG based on rank metric codes

5.3.1 Description of the generator

Now that we have a family of one-way functions based on a hard problem, our goal is to use them to build a PRNG which will inherit that hardness. We begin by letting $k = Rn$ and $w = \omega n$ for some constant R and ω. The security and the complexity of computing the pseudo-random sequence associated to this generator will then be expressed as a function of n, with R and ω as parameters.

First it is necessary to expand the size of the input, so that the number of syndromes becomes larger than the number of words of weight w_n. By definition, these two numbers are equal when $w = d_{GV}$ so that we can choose $\omega < \frac{d_{GV}}{n}$. The size of the input is $n(n-k)n\lceil \log q \rceil = n^3(1-R)\lceil \log q \rceil$ for $\boldsymbol{H}$ plus $w_n(2n-w_n)\lceil \log q \rceil = n^2(2\omega - \omega^2)\lceil \log q \rceil$ for $\boldsymbol{y}$ and the size of the output is $n^3(1-R)\lceil \log q \rceil + n^2(1-R)\lceil \log q \rceil$. So the function f_n expands the size of the input by $n^2(1-R-2\omega+\omega^2)\lceil \log q \rceil = \mathcal{O}(n^2)$ bits. To compute $f_n(\boldsymbol{H}, \boldsymbol{y})$ one has to perform a product matrix-vector in a field of degree n, which costs $\mathcal{O}(n^3)$ operations in $\mathbb{F}_q$.

Secondly we need an algorithm which computes a word $\boldsymbol{y} \in \mathbb{F}_{q^n}^n$ of weight ωn with $n^2(2\omega - \omega^2)\lceil \log q \rceil$ bits. This can be done very easily. According to Definition 2.1.3, $\boldsymbol{y}$ can be seen as an $n \times n$ matrix M over $\mathbb{F}_q$ of rank ωn. Let $\beta = (\beta_1, \ldots, \beta_{\omega n})$ be a basis of the subspace generated by the rows of M. We can represent β by a matrix $B \in \mathbb{F}_q^{\omega n \times n}$. There exists a unique matrix $A \in \mathbb{F}_q^{nn}$ such that $M = AB$. In order to ensure the unicity of this representation, we need to take B in its echelon form B_{ech}, then $M = A' B_{ech}$ for some matrix A'. Unfortunately, it is not so easy to enumerate all the echelon matrices efficiently. To avoid this problem, we only generate words with a certain form, as it is done for SYND [39].

Définition 5.3.1 (Regular Rank Words). *A word $\boldsymbol{y} \in \mathbb{F}_{q^n}^n$ of weight r is said to be regular if its associated matrix $M \in \mathbb{F}_q^{n \times n}$ is of the form*

$$M = A \begin{pmatrix} 1 & & \\ & \ddots & C \\ & & 1 \end{pmatrix}$$

with $A \in \mathbb{F}_q^{n \times r}$ and $C \in \mathbb{F}_q^{r \times (n-r)}$

The probability that a word of weight r is regular is equal to the probability that a $r \times r$ matrix over $\mathbb{F}_q$ is invertible. This probability is greater than a constant $c > 0$ for all r and q. Thus it is not harder to solve the RSD problem in the general case than to solve the RSD problem by restraining it to the regular words, since if a polynomial algorithm could solve the RSD problem in the case of regular words then it would also give an algorithm solving the RSD problem with a probability divided by a constant, hence the

RSD problem with regular words remains hard.

Input : $n^2(2\omega - \omega^2)\lceil \log q \rceil$ bits
Output : $\boldsymbol{y} \in \mathbb{F}_{q^n}^n, w_R(\boldsymbol{y}) = \omega n$
Data : A basis $(\beta_1, \ldots, \beta_n)$ of $\mathbb{F}_{q^n}/\mathbb{F}_q$
begin
  compute $x \in \mathbb{F}_q^{n^2(2\omega-\omega^2)}$ with the input bits;
  compute $A \in \mathbb{F}_q^{n \times \omega n}$ with the first ωn^2 coordinates of x;
  compute $B \in \mathbb{F}_q^{\omega n \times (n-\omega n)}$ with the last coordinates of x;
  $B \leftarrow (I_{\omega n}|B)$ `/* this is the concatenation of two matrices */`;
  $M \leftarrow AB$;
  $\boldsymbol{y} \leftarrow (\beta_1, \ldots, \beta_n)M$;
  return $\boldsymbol{y}$;
end

Algorithme 4 : Expansion Algorithm

The most expensive step of this algorithm is the matrix product which takes ωn^3 operations in $\mathbb{F}_q$, so its overall complexity is $\mathcal{O}(n^3)$.

With these two functions, we can construct an iterative version of the generator which

can compute as many bits as we want.

Input : a vector $\boldsymbol{x} \in \mathbb{F}_q^K$ where K is the security parameter
Output : N pseudo-random bits
Data : a random matrix in systematic form $\boldsymbol{H} \in \mathbb{F}_{q^n}^{(1-R)n \times n}$, an initialization vector $\boldsymbol{v} \in \mathbb{F}_q^{n^2(2\omega-\omega^2)-K}$
begin
  $\boldsymbol{y} \leftarrow$ Expansion($\boldsymbol{x} \| \boldsymbol{v}$);
  repeat
    $s \leftarrow \boldsymbol{H}\boldsymbol{y}^T$;
    split s into two strings of bits s_1 and s_2, with s_1 of length $n^2(2\omega - \omega^2)\lceil \log q \rceil$;
    output s_2;
    $\boldsymbol{y} \leftarrow$ Expansion(s_1);
  until *the number of bits generated* $> N$;
end

Algorithme 5 : Our Pseudo-Random Generator

5.3.2 Security of the generator

We recall that a distribution is pseudo-random if it is polynomial-time indistinguishable from a truly random distribution. If our generator were not pseudo-random, then there would exist a distinguisher D_R which distinguishes a sequence produced by our generator from a truly random sequence with a non-negligible advantage. We can use this distinguisher to build another distinguisher for the Fischer-Stern generator [34]. That generator is proven pseudo-random if syndrome decoding in the Hamming metric is hard [8]. It takes as input a parity-check matrix $M \in \mathbb{F}_2^{k \times n}$ of a random code and a vector $\boldsymbol{x} \in \mathbb{F}_2^n$ of Hamming weight d, with d smaller than the Gilbert-Varshamov bound (in the Hamming metric) of the code and outputs $(M, M\boldsymbol{x}^T)$.

We need a method to embed an $\mathbb{F}_q$-linear code into an $\mathbb{F}_{q^m}$-linear code. We use the same technique as in [47].

Définition 5.3.2. *Let $m \geqslant n$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_{q^m}^n$. We define the embedding of*

$\mathbb{F}_q^n$ into $\mathbb{F}_{q^m}^n$ by :

$$\begin{aligned} \psi_{\boldsymbol{\alpha}} : \quad \mathbb{F}_q^n &\to \mathbb{F}_{q^m}^n \\ (x_1, \ldots, x_n) &\mapsto (\alpha_1 x_1, \ldots, \alpha_n x_n) \end{aligned} \tag{5.1}$$

For every $\mathbb{F}_q$-linear code $\mathcal{C}$, we denote by $C(\mathcal{C}, \boldsymbol{\alpha})$ the $\mathbb{F}_{q^m}$-linear code generated by the set $\psi_{\boldsymbol{\alpha}}(\mathcal{C})$.

Our distinguisher works as follow :

- it takes as input $M \in \mathbb{F}_2^{(n-k)\times n}$ and $s \in \mathbb{F}_2^{n-k}$.
- it chooses a vector $\boldsymbol{\alpha} \in \mathbb{F}_{2^m}^n$ at random until the coordinates of $\boldsymbol{\alpha}$ are $\mathbb{F}_2$-linearly independent.
- it gives to D_R the input $(\psi_{\boldsymbol{\alpha}}(M), s)$.
- it returns the same value as D_R.

If (M, s) is an output of the Fisher-Stern generator, then there exists an $\boldsymbol{x}$ such that $s = M\boldsymbol{x}^T$ and $w_H(\boldsymbol{x}) = d$. Hence $s = \psi_{\boldsymbol{\alpha}}(M)\psi_{\boldsymbol{\beta}}(\boldsymbol{x})^T$ with $\boldsymbol{\beta} = \boldsymbol{\alpha}^{-1} = (\alpha_1^{-1}, \ldots, \alpha_n^{-1})$.

Let $\mathcal{C}$ be the code of parity-check matrix M. Since $\mathcal{C}$ is a random code, its Hamming minimum distance d is on the Gilbert-Varshamov bound, so $d \approx d_{GV}$.

Note that $w_H(\psi_\beta(\boldsymbol{x})) = d$. According to Theorem 8 of [47], if we choose $m > 8n$, the probability that the rank minimum distance d_R of $C(\mathcal{C}, \boldsymbol{\alpha})$ is different from d decreases exponentially with n. According to Lemma 7 of [47], the rank weight of $\psi_\beta(\boldsymbol{x})$ satisfies $w_R(\psi_\beta(\boldsymbol{x})) = w_H(\boldsymbol{x}) = d$. This implies that the distinguisher D_R accepts (M, s) with a non-negligible advantage.

If (M, s) is purely random, D_R sees only a random distribution and accepts the inputs with probability 1/2.

Thus the existence of a distinguisher for our generator implies the existence of a distinguisher for the Fisher-Stern generator, which contradicts Theorem 2 of [34]. This implies that our generator is pseudo-random.

5.4 Performances and examples of parameters

5.4.1 Asymptotic behaviour

Consider the case of a random parity check matrix without any structure, $n = m$, $q = 2$ and the case where w is on the rank Gilbert-Varsahmov bound (which is equal in this case to $n(1 - \sqrt{k/n})$). In that case the cost of a syndrome computation is in $\mathcal{O}(n^3)$ operations in the base field $\mathbb{F}_2$ and the number of random bits that are obtained is in $\mathcal{O}(n^2)$, hence the number of operations in the base field $\mathbb{F}_2$ per bit is $\mathcal{O}(n)$. Now since the complexity of the best attacks is in $2^{\mathcal{O}(n^2)}$, for a given security parameter λ, we obtain $n = \mathcal{O}(\sqrt{\lambda})$ and the cost of the protocol is $\mathcal{O}(\sqrt{\lambda})$ per bit, when for other Hamming based approach the cost is in $\mathcal{O}(\lambda)$ in the case of random parity check matrices without additional structure.

Concerning the amount of data, there is also an asymptotic improvement when compared to the Hamming metric. The data for this protocol is given by the matrix $\boldsymbol{H}$ of the code. If it is written in systematic form, $\boldsymbol{H}$ is described by its $k(n-k)$ coefficients in $\mathbb{F}_{2^n}$, so $k(n-k)n$ bits, hence the size of the data N is in $\mathcal{O}(n^3) = \mathcal{O}(\lambda^{3/2})$ whereas the size of the data is in $\mathcal{O}(\lambda^2)$ for the Hamming metric with random matrices [34].

5.4.2 Parameters

We propose two sets of parameters. In the first table, we optimize the size of the required data, that is to say the matrix of the code and the initialization vector. These parameters are useful in constrained environments where the lack of memory is the main issue. The drawback of small data size is that the performance of the generator is very low.

n	n-k	$d_{GV}(n,k)$	w	security	data size	key size	cycles/bytes
31	18	11	10	128	7646 bits	128	273
41	25	16	12	192	17048 bits	192	144
47	30	19	15	256	24899 bits	256	183
61	39	25	23	512	54103 bits	512	977

In the second table, we optimize the speed of the generator at the cost of data size (but still with small matrix sizes compared to SYND and XSYND). We obtain speeds comparable

to those of SYND [39] and we are less efficient than XSYND [67]. But we can always increase the parameters to improve the speed of the generator. Moreover our parameters are chosen at random and have no cyclic structure. In practice as for SYND and XSYND our results are slower than optimized versions of AES in counter mode, but at the price of increasing the size of the matrix it is possible to do better : the parameters we propose in this second table show how it is possible to optimize the speed at the cost of a larger matrix size. Note that the parameters we propose here are only examples and that it should be possible to have faster speed.

n	n-k	$d_{GV}(n,k)$	w	security	data size	key size	cycles/bytes
43	36	24	14	128	14038 bits	128	48
61	50	35	17	192	35143 bits	192	51
83	73	54	25	256	63859 bits	256	51
127	115	87	42	512	183652 bits	512	76

All the lengths n are prime, although there is no evidence that a specific attack against composite length would exist. As quantum attacks divide the exponent of the complexity of the attack by two, our two last parameters are quantum resilient.

We made a non-optimized implementation of our scheme with the MPFQ library, which showed that the theoretical complexity we give, fitted with what we obtained in practice. Moreover the main operation of our system, the syndrome computation (a matrix-vector product) is highly parallelizable, which can be used to further improve the performances.

Chapitre 6

Nouvel IBE en métrique rang

6.1 Identity based Encryption

The notion of identity-based encryption (IBE) was introduced by Shamir [76]. This gives an alternative to the standard notion of public-key encryption. In an IBE scheme, the public key associated with a user can be an arbitrary identity string, such as his e-mail address, and others can send encrypted messages to a user using this arbitrary identity without having to rely on a public-key infrastructure.

The main technical difference between a public key encryption (PKE) and IBE is the way the public and private keys are bound and the way of verifying those keys. In a PKE scheme, verification is achieved through the use of a certificate which relies on a public-key infrastructure. In an IBE, there is no need of verification of the public key but the private key is managed by a Trusted Authority (TA).

Difficulty in designing an IBE. There are two main difficulties in designing an IBE in comparison with a PKE

1. In a PKE, one often generates a public key from a secret key and normally, well-formed public keys are exponentially sparse. In an IBE scheme, any identity should be mapped to a public key and there is no known technique to randomly generate a point in an exponentially sparse space. Regev's public key encryption is an example [74]. In order to circumvent this problem, Gentry *et. al.* proposed a "dual"

of a public-key cryptosystem, in which public keys are first generated in a primal space such that they are *dense* : every point in the primal space corresponds to a public-key and thus via a random oracle, one can map any identity to a valid public key.

2. For some PKE, the public keys are dense and one can thus map any identity to a well-formed public key. However, the difficulty is to extract the corresponding secret key from the public key. ElGamal's public key encryption [29] is an example because from a public key y in a cyclic group generated by g, there is no trapdoor for the discrete log problem that allows to find the corresponding secret key x such that $g^x = y$. In order to circumvent this problem, bilinear maps have been used [16].

Beside the technical difficulties in the design, achieving security in IBE is much more complicated than in PKE. The main difference is that in IBE, except the challenge identity that the adversary aims to attack, any other identities can be corrupted. Therefore the simulator has to be able to generate secret keys for all identities but the challenge identity. Under the above difficulties in the design and in proving the security, it took nearly twenty years for finding efficient methods to implement IBE.

There are currently three classes of IBE schemes : from elliptic curve pairings introduced by Sakai, Ohgishi and Kasahara [75] and by Boneh and Franklin in [16] ; from the quadratic residue problem by Cocks in 2001 [23] ; and from hard problems on lattices by Gentry, Peikert, and Vaikuntanathan [49]. These pioneer works inspired then many other ideas to improve the efficiency or to strengthen the security, in particular to avoid the use of the random oracle. We can name some very interesting schemes in the standard model : pairing-based schemes [14, 15, 81, 48, 20, 80] and lattice-based schemes [21, 1, 17]. It is still not known how to devise an IBE scheme from the quadratic residue problem without random oracles. We explain below a new method to achieve the first IBE scheme in coding theory, with the help of rank metric codes and in the random oracle model.

Achieving IBE in Euclidean Metric. Let us first recall the technique in lattices that helps to construct IBE schemes. One of the major breakthroughs in lattice cryptography was the work of Gentry, Peikert, and Vaikuntanathan [49], that showed how to use a short trapdoor basis to generate short lattice vectors without revealing the trapdoor. This

was used to give the first lattice-based construction of a secure identity-based encryption scheme.

Let us start with Regev's scheme [74]. Associated to a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, one generates the public key as $\boldsymbol{p} = \boldsymbol{s}A + \boldsymbol{e}$ for $\boldsymbol{s} \in \mathbb{Z}_q^n$ and a short vector $\boldsymbol{e}$. The set of possible public keys are points near a lattice point and are thus exponentially sparse. Gentry, Peikert, and Vaikuntanathan introduced a dual version of the Regev's scheme in exchanging the role of public key and of secret key in defining the public key as $\boldsymbol{u} \stackrel{\text{def}}{=} A\boldsymbol{e}^T \mod q$ for short $\boldsymbol{e} \in \mathbb{Z}^m$. The public keys are now dense, any identity could be mapped via a random oracle to a point in $\mathbb{Z}_q^n$ which will then be used as the corresponding public key. The key property is, with a carefully designed trapdoor $\boldsymbol{T}$, from a random public key $\boldsymbol{u} \in \mathbb{Z}_q^n$, the preimage $\boldsymbol{e}$ of the function $f_{\boldsymbol{A}}(\boldsymbol{e}) := A\boldsymbol{e} \mod q$ can be sampled in a space of well-defined short vectors used as the secret keys.

Achieving IBE in Rank Metric : Our technique. It seems very hard to give a rank metric analogue version of the above lattice technique. The main reason is due to the difficulty of obtaining a robust analysis of such a presampling function. However, we can overcome this difficulty in another way which perfectly fits the rank metric. We still keep the public key as $\boldsymbol{p} = \boldsymbol{s}A + \boldsymbol{e}$ for $\boldsymbol{e}$ of low rank (say at most r) in $\mathbb{F}_{q^m}^n$, and for $\boldsymbol{A}$ and $\boldsymbol{s}$ drawn uniformly at random in $\mathbb{F}_{q^m}^{(n-k)\times n}$ and $\mathbb{F}_{q^m}^{n-k}$ respectively, where $\mathbb{F}_{q^m}$ is the finite field over q^m elements. The main feature of the rank metric which will be used in what follows is that we can choose the bound r to be above the Gilbert Varshamov (RGV) bound for rank codes and this gives us two ingredients to design an IBE :

- with r carefully chosen above the RGV bound, we can still invert the function $f(\boldsymbol{s}, \boldsymbol{e}) = \boldsymbol{s}A + \boldsymbol{e}$. This relies on the RankSign system with a trapdoor to compute the pre-image of the function f [45].
- with overwhelming probability, any point $\boldsymbol{p}$ has a preimage $(\boldsymbol{s}, \boldsymbol{e})$ such that $f(\boldsymbol{s}, \boldsymbol{e}) = \boldsymbol{p}$. We can thus map an arbitrary identity to a valid public key $\boldsymbol{p}$, by using a random oracle as in the case of the GPV scheme.

Rank metric vs. Hamming and Euclidean Metric. The rank metric and th Hamming metric are very different metrics. This difference is reflected for instance in the size of balls : when the number of elements of a ball of radius r in the Hamming metric for $\{0,1\}^n$ is bounded above by 2^n, for rank metric the number of elements is exponen-

tial but with a quadratic exponent depending on r. In practice, it means that even if it is possible to construct a trapdoor function for the Hamming distance such as the CFS signature scheme [24], the dimension of the dual code used there has to be sublinear in its length, whereas for rank metric it is possible to obtain such a trapdoor function for constant rate codes. This latter property makes it very difficult to use such a trapdoor function for the Hamming distance in order to build an IBE scheme whereas it is tractable for the rank metric.

Moreover one strong advantage of rank metric is the potential size of public keys. If one considers the general syndrome decoding problem $\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{s}$ (for the hardest case), because of the complexity of the best known attacks for rank metric (see [43]), and for λ a security parameter, the size of $\boldsymbol{H}$ is in $\mathcal{O}\left(\lambda^{\frac{3}{2}}\right)$ for rank metric when it is in $\mathcal{O}\left(\lambda^2\right)$ for Hamming and Euclidean metrics.

6.1.1 Complexity of the rank decoding problem

As explained earlier in the introduction the complexity of practical attacks grows very fast with the size of parameters. There exist two types of generic attacks on the problem :

Combinatorial attacks : these attacks are usually the best ones for small values of q (typically $q = 2$) and when n and k are not too small ; when q increases, the combinatorial aspect makes them less efficient. The best attacks generalize the basic information set decoding approach in a rank metric context. Interestingly enough, the more recent improvements based on the birthday paradox do not seem to generalize in rank metric because of the different notion of support.

In practice, when $m \leqslant n$, the best combinatorial attacks have complexity $\mathcal{O}\left((n-k)^3 m^3 q^{(r-1)\lfloor \frac{(k+1)m}{n} \rfloor}\right)$ [43].

Algebraic attacks : the particular nature of rank metric makes it a natural field for algebraic attacks and solving by Groebner basis, since these attacks are largely independent of the value of q and in some cases may also be largely independent on m. These attacks are usually the most efficient ones when q increases. There exist different types of algebraic modeling which can then be solved with Groebner basis techniques ([61], [31], [30], [43]). Algebraic attacks usually consider algebraic systems on the base field $\mathbb{F}_q$, it implies that the number of unknowns is quadratic in the length of the code. Since the general complexity of Groebner basis attacks

is exponential in the number of unknowns, it induces for cryptographic parameters, general attacks with a quadratic exponent in the length of the code, as for combinatorial attacks.

6.2 A New Public Key Encryption

6.2.1 Public-Key Encryption

Let us briefly recall that a public-key encryption scheme $\mathcal{S}$ is defined by three algorithms : the key generation algorithm KeyGen which, on input the security parameter, produces a pair of matching public and private keys (pk, sk) ; the encryption algorithm $\mathsf{Enc}_{pk}(m; r)$ which outputs a ciphertext c corresponding to the plaintext $m \in \mathcal{M}$, using random coins $r \in \mathcal{R}$; and the decryption algorithm $\mathsf{Dec}_{sk}(c)$ which outputs the plaintext m associated to the ciphertext c.

It is now well-admitted to require *semantic security* (*a.k.a. polynomial security* or *indistinguishability of encryptions* [52], denoted IND) : if the attacker has some *a priori* information about the plaintext, it should not learn more with the view of the ciphertext. More formally, this security notion requires the computational indistinguishability between two messages, chosen by the adversary, one of which has been encrypted. The issue is to find which one has been actually encrypted with a probability significantly better than one half. More precisely, we define the advantage $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{S}}(\mathcal{A})$, where the adversary $\mathcal{A}$ is seen as a 2-stage Turing machine $(\mathcal{A}_1, \mathcal{A}_2)$ by

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{S}}(\mathcal{A}) \stackrel{\text{def}}{=} 2 \times \Pr\left[\begin{array}{l}(pk, sk) \leftarrow \mathsf{KeyGen}, (m_0, m_1, s) \leftarrow \mathcal{A}_1(pk), \\ b \stackrel{R}{\leftarrow} \{0,1\}, c = \mathsf{Enc}_{pk}(m_b) : \mathcal{A}_2(m_0, m_1, s, c) = b\end{array}\right] - 1.$$

This advantage should ideally be a negligible function of the security parameter.

6.2.2 Description of the cryptosystem RankPKE

First, we need to define what we call a homogeneous matrix which will be used for encryption.

Définition 6.2.1 (Homogeneous Matrix). *A matrix* $\boldsymbol{M} = (m_{ij})_{\substack{1\le i\le a\\ 1\le j\le b}} \in \mathbb{F}_{q^m}^{a\times b}$ *is homogeneous of weight* d *if all its coefficients belong to the same* $\mathbb{F}_q$*-vector subspace of dimension* d*, that is to say*

$$\dim_{\mathbb{F}_q}\langle m_{ij}\rangle = d$$

We now introduce a public-key encryption scheme, called RankPKE. Let $\boldsymbol{A}$ be drawn uniformly at random in $\mathbb{F}_{q^m}^{(n-k)\times n}$. We need it to be of full rank. This happens with overwhelming probability (i.e. $1-\mathcal{O}\big(q^{-m(k+1)}\big)$). Let W_r be the set of all the words of rank r and of length n , i.e. $W_r = \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \|\boldsymbol{x}\| = r\}$. The system RankPKE works as follows :

- RankPKE.KeyGen :
  - generate $\boldsymbol{A} \xleftarrow{\$} \mathbb{F}_{q^m}^{(n-k)\times n}$
  - generate $\boldsymbol{s} \xleftarrow{\$} \mathbb{F}_{q^m}^{n-k}$ and $\boldsymbol{e} \xleftarrow{\$} W_r$
  - compute $\boldsymbol{p} = \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e}$
  - define $\boldsymbol{G} \in \mathbb{F}_{q^m}^{k'\times n'}$ a generator matrix of a public code $\mathcal{C}$ which can decode (efficiently) errors of weight up to wr, where w is defined just below.
  - define $sk = \boldsymbol{s}$ and $pk = (\boldsymbol{A}, \boldsymbol{p}, \boldsymbol{G})$
- RankPKE.Enc$((\boldsymbol{A}, \boldsymbol{p}, \boldsymbol{G}), \boldsymbol{m})$:
  Let $\boldsymbol{m} \in \mathbb{F}_{q^m}^{k'}$ be the message we want to encrypt. We generate a random homogeneous matrix $\boldsymbol{U} \in \mathbb{F}_{q^m}^{n\times n'}$ of weight w. Then we can compute the ciphertext $(\boldsymbol{C}, \boldsymbol{x})$ of $\boldsymbol{m}$ as :

$$\left(\begin{array}{c}\boldsymbol{A}\\ \hline \boldsymbol{p}\end{array}\right)\boldsymbol{U} + \left(\begin{array}{c}\boldsymbol{0}\\ \hline \boldsymbol{m}\boldsymbol{G}\end{array}\right) = \left(\begin{array}{c}\boldsymbol{C}\\ \hline \boldsymbol{x}\end{array}\right)$$

- RankPKE.Dec$(\boldsymbol{s}, (\boldsymbol{C}, \boldsymbol{x}))$:
  - use the secret key $\boldsymbol{s}$ to compute :

$$\begin{aligned}\left(\begin{array}{c|c}\boldsymbol{s} & -1\end{array}\right)\left(\begin{array}{c}\boldsymbol{C}\\ \hline \boldsymbol{x}\end{array}\right) &= \boldsymbol{s}\boldsymbol{C} - \boldsymbol{x} = \boldsymbol{s}\boldsymbol{A}\boldsymbol{U} - \boldsymbol{p}\boldsymbol{U} - \boldsymbol{m}\boldsymbol{G}\\ &= \boldsymbol{s}\boldsymbol{A}\boldsymbol{U} - (\boldsymbol{s}\boldsymbol{A}+\boldsymbol{e})\boldsymbol{U} - \boldsymbol{m}\boldsymbol{G} = -\boldsymbol{e}\boldsymbol{U} - \boldsymbol{m}\boldsymbol{G}\end{aligned}$$

— since $\boldsymbol{U}$ is homogeneous, we have $\|\boldsymbol{eU}\| \leqslant wr$. Therefore, by using the decoding algorithm of $\mathcal{C}$, we recover $\boldsymbol{m}$.

The expansion rate of this cryptosystem is $\frac{n-k+1}{R}$ where $R = \frac{k'}{n'}$ is the rate of $\mathcal{C}$.

6.2.3 Security

Définition 6.2.2 (Rank Support Learning (RSL)). *Let $\boldsymbol{A}$ be a random full-rank matrix of size $(n-k) \times n$ over $\mathbb{F}_{q^m}$ and V be a subspace of $\mathbb{F}_{q^m}$ of dimension w. Let $\mathcal{O}$ be an oracle which gives samples of the form $(\boldsymbol{A}, \boldsymbol{Au})$, where $\boldsymbol{u} \xleftarrow{\$} V^n$. The $\mathsf{RSL}_{q,m,n,k,w}$ problem is to recover V given only access to the oracle.*
We say that the problem is (N, t, ε)-hard if for every probabilistic algorithm $\mathcal{A}$ running in time t, we have

$$\mathrm{Prob}[\mathcal{A}(\boldsymbol{A}, \boldsymbol{AU}) = V] \leqslant \varepsilon, \quad \boldsymbol{U} \xleftarrow{\$} V^{n \times N}$$

When we want to stress the fact that we care about the problem where we are allowed to make exactly N calls to the oracle, we denote this the $\mathsf{RSL}_{q,m,n,k,w,N}$ problem. The pair $(\boldsymbol{A}, \boldsymbol{AU})$ is referred to as an instance of the $\mathsf{RSL}_{q,m,n,k,w,N}$ problem.

The corresponding decisional problem, namely DRSL*, is to distinguish $(\boldsymbol{A}, \boldsymbol{AU})$ from $(\boldsymbol{A}, \boldsymbol{Y})$ where $\boldsymbol{Y} \xleftarrow{\$} \mathbb{F}_{q^m}^{(n-k) \times N}$.*

Proposition 6.2.3. *The $\mathsf{RSL}_{q,m,n,k,w,N}$ is as hard as the $\mathsf{RSD}_{q,m,n,k,w}$ problem.*

Démonstration. Let $\boldsymbol{A}$ be a full-rank $(n-k) \times n$ matrix over $\mathbb{F}_{q^m}$ and $\boldsymbol{x} \in \mathbb{F}_{q^m}^{n-k}$ of rank w. Let $\boldsymbol{s} = \boldsymbol{Ax}$. $(\boldsymbol{A}, \boldsymbol{s})$ is an instance of the $\mathsf{RSD}_{q,m,n,k,w}$ problem.
Let $\boldsymbol{S}$ be a matrix obtained by the concatenation of N times the vector $\boldsymbol{s}$. $(\boldsymbol{A}, \boldsymbol{S})$ is an instance of the $\mathsf{RSL}_{q,m,n,k,w,N}$ problem.
If we are able to solve any instances of the $\mathsf{RSL}_{q,m,n,k,w,N}$ problem, then we can recover the support of $\boldsymbol{x}$ and solve the instance $(\boldsymbol{A}, \boldsymbol{s})$.
We can use this technique to solve any instances of the $\mathsf{RSD}_{q,m,n,k,w}$ problem, which proves that the $\mathsf{RSL}_{q,m,n,k,w,N}$ is as hard as the $\mathsf{RSD}_{q,m,n,k,w}$ problem in the worst case. □

Security of the DRSL **and** DRSD **problems.**
We have already seen in the previous section that the DRSD problem is hard. As for other problems in cryptography (like DDH [13, 28]), the best known attacks on the

$\mathsf{DRSL}_{q,m,n,k,w,N}$ problem consist in solving the same instance of the $\mathsf{RSL}_{q,m,n,k,w,N}$ problem, so we make the assumption that the $\mathsf{DRSL}_{q,m,n,k,w,N}$ problem is difficult.

Théorème 6.2.4. *Under the assumption that* DRSL *is hard, the scheme* RankPKE *is semantically secure.*

Démonstration. We proceed by a sequence of games.

Game $\mathbf{G}_0$: This is the real IND-CPA attack game. The RankPKE.KeyGen is run and then, a 2-stage poly-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is fed with the public key $pk = (\boldsymbol{A}, \boldsymbol{p}, \boldsymbol{G}')$. Then, $\mathcal{A}_1$ outputs a pair of messages $(\boldsymbol{m}_0, \boldsymbol{m}_1)$. Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext $c^\star := (C^\star, \boldsymbol{x}^\star)$ of $\boldsymbol{m}^\star = \boldsymbol{m}_b$. This ciphertext $c^\star$ comes from a random homogeneous matrix $\boldsymbol{U} \in \mathbb{F}_{q^m}^{n \times n'}$ of weight w and then $c^\star = \mathsf{RankPKE.Enc}((\boldsymbol{A}, \boldsymbol{p}, \boldsymbol{G}'), \boldsymbol{m}_b)$. On input $c^\star$, $\mathcal{A}_2$ outputs bit b'. We denote by S_0 the event $b' = b$ and use the same notation S_n in any game $\mathbf{G}_n$ below.

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{RankPKE}}(\mathcal{A}) = \mid 2\Pr[\mathrm{S}_0] - 1 \mid$$

Game $\mathbf{G}_1$: In this game, we replace $\boldsymbol{p} = \boldsymbol{s}A + \boldsymbol{e}$ in RankPKE.KeyGen by $\boldsymbol{p} \xleftarrow{\$} \mathbb{F}_{q^m}^n$. Under the hardness of the DRSD problem, the two games $\mathbf{G}_1$ and $\mathbf{G}_0$ are indistinguishable :

$$\mid \Pr[\mathrm{S}_1] - \Pr[\mathrm{S}_0] \mid \leq \epsilon_{\mathsf{drsd}},$$

where ϵ_{drsd} is the bound on the successful probability of the attacks against the problem DRSD.

Game $\mathbf{G}_2$: In this game, we replace $(C^\star, \boldsymbol{x}^\star)$ in $\mathbf{G}_1$ by $(C^\star \xleftarrow{\$} \mathbb{F}_{q^m}^{(n-k)\times n'}, x^\star \xleftarrow{\$} \mathbb{F}_{q^m}^{n'})$. As $\boldsymbol{x}^\star$ is perfectly random, $\boldsymbol{x}^\star - \boldsymbol{m}^\star\boldsymbol{G}$ is also perfectly random. In other words, this game replaces $\left(\dfrac{\boldsymbol{A}}{\boldsymbol{p}}\right)\boldsymbol{U} = \left(\dfrac{C^\star}{\boldsymbol{x}^\star - \boldsymbol{m}^\star\boldsymbol{G}}\right)$ by a perfectly random matrix. Therefore, the indistinguishability of the two games $\mathbf{G}_2$ and $\mathbf{G}_1$ follows from the hardness of the DRSL problem, applying it to the matrix $\boldsymbol{A}' = \left(\dfrac{\boldsymbol{A}}{\boldsymbol{p}}\right)$ which is perfectly random because $\boldsymbol{A}$ and $\boldsymbol{p}$ are both perfectly random. Thus

$$\mid \Pr[\mathrm{S}_2] - \Pr[\mathrm{S}_1] \mid \leq \epsilon_{\mathsf{drsl}},$$

where ϵ_{drsl} is the bound on the successful probability of the attacks against the DRSL problem.

Advantage Zero.

In this last game, as the ciphertext challenge $(C^\star, \boldsymbol{x}^\star)$ is perfectly random, b is perfectly hidden to any adversary $\mathcal{A}$.

$$|\Pr[\mathrm{S}_2]| = \frac{1}{2}$$

□

6.3 On the difficulty of the rank support learning problem

The purpose of this section is to give some evidence towards the difficulty of the support learning problem $\mathsf{RSL}_{q,m,n,k,w,N}$ by

- explaining that it is the rank metric analogue of a problem in Hamming metric (the so called support learning problem) which has already been useful to devise signature schemes and for which after almost twenty years of existence only algorithms of exponential complexity are known ;
- explaining that it is a problem which is provably hard for $N = 1$ and that it becomes easy only for very large values of N ;
- giving an algorithm which is the analogue in the rank metric of the best known algorithm for the support learning problem which is of exponential complexity. This complexity is basically smaller by a multiplicative factor which is only of order $q^{-\beta N}$ (for some $\beta < 1$) than the complexity of solving the rank syndrome decoding problem $\mathsf{RSD}_{q,m,n,k,w}$;
- relating this problem to finding a codeword of rank weight w in a code where there are q^N codewords of this weight. It is reasonable to conjecture that the complexity of finding such a codeword gets reduced by a multiplicative factor which is at most q^N compared to the complexity of finding a codeword of rank weight w in a random code of the same length and dimension which has a single codeword of this weight ;

— showing that this problem can also rephrased in terms of decoding a random code but defined over a larger alphabet ($\mathbb{F}_{q^{mN}}$ instead of $\mathbb{F}_{q^m}$).

6.3.1 A related problem : the support learning problem

The rank support learning problem can be viewed as the rank metric analogue of the support learning problem which can be expressed as follows.

Problem 6.3.1 (Support Learning). *Let $\boldsymbol{A}$ be a random full-rank matrix of size $(n-k)\times n$ over $\mathbb{F}_q$ and I be a subset of $\{1,\dots,n\}$ of size w. Let V be the subspace of $\mathbb{F}_q^n$ of vectors with support I, that is the set of vectors $\boldsymbol{u}=(u_i)_{1\leq i\leq n}\in\mathbb{F}_q^n$ such that $u_i=0$ when $i\notin I$. Let $\mathcal{O}$ be an oracle which gives samples of the form $(\boldsymbol{A},\boldsymbol{Au})$, where $\boldsymbol{u}\xleftarrow{\$}V$. The support learning problem is to recover I given only access to the oracle.*
We say that the problem is (N,t,ε)-hard if for every probabilistic algorithm $\mathcal{A}$ running in time t, we have

$$\mathrm{Prob}[\mathcal{A}(\boldsymbol{A},\boldsymbol{AU})=V]\leqslant\varepsilon,\quad \boldsymbol{U}\xleftarrow{\$}V^N$$

When we want to stress the fact that we care about the problem where we are allowed to make exactly N calls to the oracle, we denote this the $\mathsf{SL}_{q,n,k,w,N}$ problem. The pair $(\boldsymbol{A},\boldsymbol{AU})$ is referred to as an instance of the $\mathsf{SL}_{q,n,k,w,N}$ problem.

When $N=1$ this is just the usual decoding problem of a random linear code with parity check matrix $\boldsymbol{A}$. In this case, the problem is known to be NP-complete [8]. When N is greater than 1, this can be viewed as a decoding problem where we are given N syndromes of N errors which have a support included in the same set I. This support learning problem with $N>1$ has already been considered before in [58]. Its presumed hardness for moderate values of N was used there to devise a signature scheme [58], the so called KKS-scheme. Mounting a key attack in this case (that is for the Hamming metric) without knowing any signature that has been computed for this key really amounts to solve this support learning problem even it was not stated exactly like this in the article. However, when we have signatures originating from this scheme, the problem is of a different nature. Indeed, it was found out in [22] that signatures leak information. The authors showed there that if we know M signatures, then we are given $\boldsymbol{A}$, $\boldsymbol{AU}$ but also M vectors in $\mathbb{F}_q^n$, $\boldsymbol{v}_1,\dots,\boldsymbol{v}_M$ whose support is included in I. The knowledge of those auxiliary $\boldsymbol{v}_i$'s help a great deal to recover I : it suffices to compute the union of their

support which is very likely to reveal the whole set I. When the $\boldsymbol{v}_i$'s are random vectors in $\mathbb{F}_q^n$ of support included in I it is clearly enough to have a logarithmic number of them (in the size of the support I) to recover I. However this does not undermine the security of the support learning problem and just shows that the KKS-signature scheme is at best a one-time signature scheme.

Some progress on the support learning problem itself was achieved almost fifteen years later in [70]. Roughly speaking the idea there is to consider a code that has q^N codewords of weight at most w which correspond to all possible linear combinations of the $\boldsymbol{u}_i$'s and to use generic decoding algorithms of linear codes (which can also be used as low-weight codewords search algorithms) to recover one of those linear combinations. The process can then be iterated to reveal the whole support I. The fact that there are q^N codewords of weight $\leq w$ that are potential solutions for the low weight codeword search algorithm implies that we may expect to gain a factor of order q^N in the complexity of the algorithm when compared to finding a codeword of weight w in a random linear code which has a single codeword of weight w. Actually the gain is less than this in practice. This seems to be due to the fact that we have highly correlated codewords (their support is for instance included in I). However, still there is some exponential speedup when compared to the single codeword case. This allowed to break all the parameters proposed in [58, 57] but also those of [5] which actually relied on the same problem. However, as has been acknowledged in [70], this does not give a polynomial time algorithm for the support learning problem, it just gives an exponential speedup when compared to solving a decoding problem with an error of weight w. The parameters of the KKS scheme can easily be chosen in order to thwart this attack.

6.3.2 Both problems reduce to linear algebra when N is large enough

As explained before when $N = 1$ the support learning problem is NP-complete. The rank support learning problem is also hard in this case since it is equivalent to decoding in the rank metric an $\mathbb{F}_{q^m}$-linear code for which there is a randomized reduction to the NP-complete decoding problem in the Hamming metric [47]. It is also clear that both problems become easy when N is large enough and for the same reason : they basically amount to computing a basis of a linear space.

In the Hamming metric, this corresponds to the case when $N = w$. Indeed in this case, notice that the dimension of the subspace V is w. When the $\boldsymbol{u}_i$'s are generated randomly with support included in I they have a constant probability $K(q)$ (which is increasing with q and bigger than 0.288 in the binary case) to generate the space $\boldsymbol{A}V$. Once we know this space, the problem becomes easy. Indeed let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ be the canonical generators of $\mathbb{F}_q^n$ (i.e. $\boldsymbol{e}_i$ has only one non-zero entry which is its i-th entry that is equal to 1). We recover I by checking for all positions i in $\{1, \ldots, n\}$ whether $\boldsymbol{A}\boldsymbol{e}_i$ belongs to $\boldsymbol{A}V$ or not. If it is the case, then i belongs to I, if this is not the case, i does not belong to I.

There is a similar algorithm for the rank support learning problem. This should not come as a surprise since supports of code positions for the Hamming metric really correspond to subspaces of $\mathbb{F}_{q^m}$ for the rank metric metric as has been put forward in [43] (see also [55] for more details about this). The difference being however that we need much bigger values of N to mount a similar attack to the Hamming metric case. Indeed what really counts here is the space that can be generated by the $\boldsymbol{A}\boldsymbol{u}_i$'s where the $\boldsymbol{u}_i$'s are the columns of $\boldsymbol{U}$. It is nothing but the space $\boldsymbol{A}V^n$. Let us denote this space by W. This space is not $\mathbb{F}_{q^m}$-linear, however it is $\mathbb{F}_q$-linear and it is of dimension nw viewed as an $\mathbb{F}_q$-linear subspace of $\mathbb{F}_{q^m}^n$. When $N = nw$ we can mount a similar attack, namely we compute the space generated by linear combinations over $\mathbb{F}_q$ of $\boldsymbol{A}\boldsymbol{u}_1, \ldots, \boldsymbol{A}\boldsymbol{u}_{nw}$. They generate W with constant probability $K(q)$. When we look all $\mathbb{F}_q$-linear subspaces V' of $\mathbb{F}_{q^m}$ of dimension 1 (there are less than q^m of them) and check whether the subspace W' of dimension n given by $\boldsymbol{A}V'^n$ is included in $W = \boldsymbol{A}V^n$ or not. By taking the sum of the spaces for which this is the case we recover V. Actually the complexity of this algorithm can be improved by using in a more clever way the knowledge of W, but this is beyond the scope of this article and this algorithm is just here to explain the deep similarities between both cases and to convey some intuition about when the rank support learning problem becomes easy.

This discussion raises the issue whether there is an algorithm "interpolating" standard decoding algorithms when $N = 1$ and linear algebra when $N = w$ in the Hamming metric case and $N = nw$ in the rank metric case. This is in essence what has been achieved in [70] for the Hamming metric and what we will do now here for the rank metric.

6.3.3 Solving the subspace problem with information-set decoding

There are two ingredients in the algorithm for solving the support learning problem in [70]. The first one is to set up an equivalent problem which amounts to finding a codeword of weight $\leq w$ in a code which has q^N codewords of this weight. The second one is to use standard information set decoding techniques to solve this task and to show that it behaves better than in the case where there is up to a multiplicative constant a single codeword of this weight in the code. We are going to follow the same route here for the rank metric.

We begin by introducing the following $\mathbb{F}_q$-linear code

$$\mathcal{C} \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \boldsymbol{A}\boldsymbol{x} \in W_U\}$$

where W_U is the $\mathbb{F}_q$-linear subspace of $\mathbb{F}_{q^m}^{n-k}$ generated by linear combinations of the form $\sum_i \alpha_i \boldsymbol{A}\boldsymbol{u}_i$ where α_i belongs to $\mathbb{F}_q$ and the $\boldsymbol{u}_i$'s are the N column vectors forming the matrix $\boldsymbol{U}$. This code has the following properties.

Lemme 6.3.1. *Let $\mathcal{C}' \stackrel{def}{=} \{\sum_i \alpha_i \boldsymbol{u}_i : \alpha_i \in \mathbb{F}_q\}$. We have*

1. $\dim_{\mathbb{F}_q} \mathcal{C} \leq km + N$
2. $\mathcal{C}' \subset \mathcal{C}$
3. *all the elements of $\mathcal{C}'$ are of rank weight $\leq w$.*

[43] gives several algorithms for decoding $\mathbb{F}_{q^m}$-linear codes for the rank metric. The first one can be generalized in a straightforward way to codes which are just $\mathbb{F}_q$-linear as explained in more detail in [55]. This article also explains how this algorithm can be used in a straightforward way to search for low rank codewords in such a code. Here our task is to look for codewords of rank $\leq w$ which are very likely to lie in $\mathcal{C}'$ which would reveal a linear combination $\boldsymbol{c} = \sum_i \alpha_i \boldsymbol{u}_i$. This reveals in general V when $\boldsymbol{c}$ is of rank weight w simply by computing the vector space over $\mathbb{F}_q$ generated by the entries of $\boldsymbol{c}$. When the rank of $\boldsymbol{c}$ is smaller this yields a subspace of V and we will discuss later on how we finish the attack.

Let us concentrate now on analyzing how the first decoding algorithm of [43] behaves when we use it to find codewords of $\mathcal{C}$ of rank $\leq w$. For this, we have to recall how the support attack of [43] works.

We assume that we want to find a codeword of weight w in an $\mathbb{F}_q$-linear code which is a $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ of dimension K. For the purpose of this algorithm, a codeword $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{F}_{q^m}^n$ is also viewed as a matrix $(c_{ij})_{\substack{1 \le i \le m \\ 1 \le j \le n}}$ over $\mathbb{F}_q$ by writing the c_i's in a arbitrary $\mathbb{F}_q$ basis $(\beta_1, \ldots, \beta_m) \in \mathbb{F}_{q^m}^m$ of $\mathbb{F}_{q^m}$ viewed as vector space over $\mathbb{F}_q$: $c_i = \sum_{j=1}^m c_{ij}\beta_j$. There are $nm - K$ linear equations which specify the code that are satisfied by the c_{ij}'s of the form

$$\sum_{1 \le i,j \le m} h_{ij}^s c_{ij} = 0 \tag{6.1}$$

for $s = 1, \ldots, mn - K$. Algorithm 6 explains how a codeword of weight $\le w$ is produced by the approach of [43]. The point of choosing r like this in this algorithm, i.e.

$$r \stackrel{\text{def}}{=} m - \left\lceil \frac{K}{n} \right\rceil \tag{6.2}$$

is that r is the smallest integer for which the linear system (6.3) has more equations than unknowns (and we therefore expect that it has generally only the all-zero solution).

Théorème 6.3.2. *Assume that* $w \le \min\left(\left\lfloor \frac{K}{n} \right\rfloor, \left\lfloor \frac{N}{n} \right\rfloor + 1\right)$ *and that* $\frac{w + \left\lfloor \frac{K}{n} \right\rfloor}{2} \ge \left\lfloor \frac{N}{n} \right\rfloor$. *Let*

$$\begin{aligned} e_- &= \left(w - \left\lfloor \frac{N}{n} \right\rfloor\right)\left(\left\lfloor \frac{K}{n} \right\rfloor - \left\lfloor \frac{N}{n} \right\rfloor\right) \\ e_+ &= \left(w - \left\lfloor \frac{N}{n} \right\rfloor - 1\right)\left(\left\lfloor \frac{K}{n} \right\rfloor - \left\lfloor \frac{N}{n} \right\rfloor - 1\right) + n\left(\left\lfloor \frac{N}{n} \right\rfloor + 1\right) - N \end{aligned}$$

Algorithm 6 outputs an element of $\mathcal{C}'$ *with complexity* $\tilde{\mathcal{O}}\left(q^{\min(e_-, e_+)}\right)$. *The complete proof of this theorem is found in [41].*

Remark 6.3.3.

1. *When* N *and* $K = km + N$ *are multiple of* n, *say* $N = \delta n$ *and* $K = \alpha R n + \delta$ *(with* $\alpha \stackrel{def}{=} \frac{m}{n}$, $R = \frac{k}{n}$*) the complexity above simplifies to* $\tilde{\mathcal{O}}\left(q^{\alpha R n(w-\delta)}\right)$. *In other words the complexity gets reduced by a factor* $q^{\alpha R \delta n} = q^{\alpha R N}$ *when compared to finding a codeword of weight* w *in a random* $\mathbb{F}_q$*-linear code of the same dimension and length.*
2. *This approach is really suited to the case* $m \le n$. *When* $m > n$ *we obtain better complexities by working on the transposed code (see [55] for more details about this approach).*

$r \leftarrow m - \lceil \frac{K}{n} \rceil$;
while *true* **do**
  $W \leftarrow$ random $\mathbb{F}_q$-subspace of dimension r of $\mathbb{F}_{q^m}$;
  Compute a basis $\boldsymbol{f}^1 = (f_i^1)_{1 \leq i \leq m}, \ldots, \boldsymbol{f}^r = (f_i^r)_{1 \leq i \leq m}$ of W;
  Make the assumption that the entries c_j of $\boldsymbol{c}$ can be written in the $\boldsymbol{f}^1, \ldots, \boldsymbol{f}^r$ basis as

$$c_j = \sum_{l=1}^{r} x_{lj} \boldsymbol{f}^l$$

  Rewrite the linear equations (6.1) by writing $c_{ij} = \sum_{l=1}^{r} x_{lj} f_i^l$ to obtain $mn - K$ equations of the form

$$\sum_{1 \leq i,j \leq m} h_{ij}^s \sum_{l=1}^{r} x_{lj} f_i^l = 0 \tag{6.3}$$

  Define $(x_{ij})_{\substack{1 \leq i \leq r \\ 1 \leq j \leq n}}$ by $c_{ij} = \sum_{l=1}^{r} x_{lj} f_i^l$;
  Solve this system (in the x_{ij}'s);
  if *this system has a non zero solution* **then**
    if $\left(\sum_{l=1}^{r} x_{lj} \boldsymbol{f}^l\right)_{1 \leq j \leq n}$ *has rank weight* $\leq w$ **then**
      return $(\sum_{l=1}^{r} x_{lj} \boldsymbol{f}^l)_{1 \leq j \leq n}$;
    end
  end
end

Algorithme 6 : algorithm that outputs a codeword of weight $\leq w$.

6.3.4 Link between rank support learning and decoding over the rank metric

We have exploited here that for solving the rank support learning problem, it can be rephrased in terms of finding a codeword of low rank weight in a code that has many codewords of such low rank weight (namely the code $\mathcal{C}$ that has been introduced in this section). $\mathcal{C}$ is not a random code however, it is formed by a random subcode, namely the code $\mathcal{C}_0 = \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \boldsymbol{A}\boldsymbol{x} = 0\}$ plus some non random part, namely $\mathcal{C}'$ which contains precisely the low rank codeword we are after. In other words $\mathcal{C}$ decomposes as

$$\mathcal{C} = \mathcal{C}_0 \oplus \mathcal{C}'$$

where $\mathcal{C}_0$ is a truly random code and $\mathcal{C}'$ is a subcode of $\mathcal{C}$ that contains the codewords of $\mathcal{C}$ of low-rank. $\mathcal{C}$ is therefore not really a random code.

There is a way however to rephrase the rank support learning problem as a problem of decoding a *random* code. The trick is to change the alphabet of the code. We define the code $\mathcal{C}_N$ as

$$\mathcal{C}_N = \{\boldsymbol{x} \in \mathbb{F}_{q^{mN}} : \boldsymbol{A}\boldsymbol{x} = 0\}.$$

In other words, $\mathcal{C}_N$ is a code defined over the extension field $\mathbb{F}_{q^{mN}}$ but with a random parity-check matrix with entries defined over $\mathbb{F}_{q^m}$.

There are several ways to equip $\mathbb{F}_{q^{mN}}^n$ with a rank metric. One of them consists in writing the entries c_i of a codeword $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{F}_{q^{mN}}^n$ of $\mathcal{C}_N$ as column vectors $(c_{ij})_{1 \leq j \leq mN} \in \mathbb{F}_q^{mN}$ by expressing the entry c_i in a $\mathbb{F}_q$ basis of $\mathbb{F}_{q^{mN}}$ $(\beta_1, \ldots, \beta_{mN})$, i.e. $c_i = \sum_{1 \leq j \leq mN} c_{ij}\beta_j$ and replacing each entry by the corresponding vector to obtain an $mN \times n$ matrix. The rank of this matrix would then define the rank weight of a codeword. However, since $\mathbb{F}_{q^{mN}}$ is an extension field of $\mathbb{F}_{q^m}$ there are also other ways to define a rank metric. We will choose the following one here. First we decompose each entry c_i in an $\mathbb{F}_{q^m}$-basis $(\gamma_1, \ldots, \gamma_N)$ of $\mathbb{F}_{q^{mN}}$:

$$c_i = \sum_{j=1}^{N} \alpha_{(i-1)N+j}\gamma_j$$

where the α_i's belong to $\mathbb{F}_{q^m}$. The rank weight of $(c_1, \ldots, c_n)$ is then defined as the rank weight of the vector $(\alpha_i)_{1 \leq i \leq nN} \in \mathbb{F}_{q^m}^{nN}$ where the rank weight of the last vector is defined as we have done up to here, namely by replacing each entry α_i by a column

vector $(\alpha_{ij})_{1\leq j\leq m}$ obtained by taking the coordinates of α_i in some $\mathbb{F}_q$-basis of $\mathbb{F}_{q^m}$. In other words, the rank weight of $(c_1,\ldots,c_n)$ is defined as the rank of the associated $m\times nN$ matrix.

Let us now introduce the rank decoding problem with random parity check matrices defined over a smaller field.

Définition 6.3.4 (Rank Decoding with parity-check matrices defined over a subfield (RDPCSF))**.** *Let $\boldsymbol{A}$ be a random full-rank matrix of size $(n-k)\times n$ over $\mathbb{F}_{q^m}$ and $\boldsymbol{e}\in\mathbb{F}_{q^{mN}}^n$ be a random word of rank weight w.The $\mathsf{RDPCSF}_{q,m,n,k,w,N}$ problem is to recover $\boldsymbol{e}$ from the knowledge of $\boldsymbol{A}\in\mathbb{F}_{q^m}^{(n-k)\times n}$ and $\boldsymbol{A}\boldsymbol{e}\in\mathbb{F}_{q^{mN}}^{n-k}$.*

It turns out that the support learning problem and the rank decoding problem with parity-check matrices defined over a smaller field are equivalent

Théorème 6.3.5. *The problems $\mathsf{RSL}_{q,m,n,w,N}$ and $\mathsf{RDPCSF}_{q,m,n,w,N}$ are equivalent : any randomized algorithm solving one of this problem with probability $\geq\epsilon$ in time t can be turned into an algorithm for the other problem solving it with probability $\geq\epsilon$ in time $t+P(q,m,n,w,N)$, where P is a polynomial function of its entries.*

Démonstration. Let us consider an instance $(\boldsymbol{A},\boldsymbol{AU})$ of the $\mathsf{RSL}_{q,m,n,w,N}$ problem. Denote the j-th column of $\boldsymbol{U}$ by $\boldsymbol{u}_j$. Define now $\boldsymbol{e}\in\mathbb{F}_{q^{mN}}^n$ by $\boldsymbol{e}=\sum_{j=1}^N\gamma_j\boldsymbol{u}_j$, where $(\gamma_1,\ldots,\gamma_N)$ is some $\mathbb{F}_{q^m}$-basis of $\mathbb{F}_{q^{mN}}$. From the definition of the rank weight we have chosen over $\mathbb{F}_{q^{mN}}^n$, it is clear that the rank weight of $\boldsymbol{e}$ is less than or equal to w. The pair $(A,\sum_{j=1}^N\gamma_j\boldsymbol{A}\boldsymbol{u}_j)$ is then an instance of the $\mathsf{RDPCSF}_{q,m,n,w,N}$ problem. It is now straightforward to check that we transform in this way a uniformly distributed instance of the $\mathsf{RSL}_{q,m,n,w,N}$ problem into a uniformly distributed instance of the $\mathsf{RDPCSF}_{q,m,n,w,N}$ problem. The aforementioned claim on the equivalence of the two problems follows immediately from this and the fact that when we know the space generated by the entries of the $\boldsymbol{u}_j$'s, we just have to solve a linear system to recover a solution of the decoding problem (this accounts for the additive polynomial overhead in the complexity). □

Note that this reduction of the rank support learning problem to the problem of decoding a linear code over an extension field $\mathbb{F}_{q^{mN}}$ defined from a random parity-check matrix defined over the base field $\mathbb{F}_{q^m}$ works also for the Hamming metric : the support learning problem $\mathsf{SL}_{q,n,w,N}$ also reduces to decoding a linear code over an extension field $\mathbb{F}_{q^{mN}}$

defined from a random parity-check matrix defined over the base field $\mathbb{F}_{q^m}$ but this time for the Hamming metric over $\mathbb{F}_{q^{mN}}^n$. All these considerations point towards the same direction, namely that when N is not too large, the rank support learning problem should be a hard problem. It is for instance tempting to conjecture that this problem can not be solved q^N faster than decoding errors of rank weight w for an $[n, k]$ random linear code over $\mathbb{F}_{q^m}$. A similar conjecture could be made for the support learning problem.

6.4 Identity Based Encryption

Identity-based encryption schemes. An identity-based encryption (IBE) scheme is a tuple of algorithms $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ providing the following functionality. The trusted authority runs Setup to generate a master key pair (mpk, msk). It publishes the master public key mpk and keeps the master secret key msk private. When a user with identity ID wishes to become part of the system, the trusted authority generates a user decryption key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$, and sends this key over a secure and authenticated channel to the user. To send an encrypted message m to the user with identity ID, the sender computes the ciphertext $C \xleftarrow{\$} \mathsf{Enc}(mpk, ID, m)$, which can be decrypted by the user as $m \leftarrow \mathsf{Dec}(d_{ID}, C)$. We refer to [16] for details on the security definitions for IBE schemes.

Security. We define the security of IBE schemes through a game with an adversary. In the first phase, the adversary is run on input of the master public key of a freshly generated key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$. In a chosen-plaintext attack ($\mathsf{IND-CPA}$), the adversary is given access to a key derivation oracle $\mathcal{O}$ that on input an identity $ID \in \{0,1\}^*$ returns $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$. At the end of the first phase, the adversary outputs two equal-length challenge messages $m_0, m_1 \in \{0,1\}^*$ and a challenge identity $ID \in \{0,1\}^\star$. The adversary is given a challenge ciphertext $C \xleftarrow{\$} \mathsf{Enc}(mpk, ID, m_b)$ for a randomly chosen bit b, and is given access to the same oracle $\mathcal{O}$ as during the first phase of the attack. The second phase ends when the adversary outputs a bit b'. The adversary is said to win the $\mathsf{IND-CPA}$ game if $b' = b$ and if it never queried the key derivation oracle for the keys of any identity that matches the target identity.

Définition 6.4.1. *An* IBE *scheme is* $\mathsf{IND-CPA}$*-secure if any poly-time adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *making at most a polynomial number of queries to the key derivation oracle, only*

has a negligible advantage in the IND − CPA *game described above,* i.e., *the following advantage is negligible :*

$$2 \times \Pr\left[\begin{array}{l}(mpk, msk) \xleftarrow{\$} \mathsf{Setup}, (ID, m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{O}}(mpk), \\ b \xleftarrow{\$} \{0,1\}, c = \mathsf{Enc}(mpk, ID, (m_b)) : \mathcal{A}_2^{\mathcal{O}}(m_0, m_1, s, c) = b\end{array}\right] - 1.$$

6.4.1 Trapdoor Functions From RankSign

We now adapt the RankSign system to construct a trapdoor function, which is sufficient to convert our PKE to an IBE. Associated to a matrix $\boldsymbol{A} \in \mathbb{F}_{q^m}^{(n-k)\times n}$, we define the function $f_{\boldsymbol{A}}$ as follows :

$$\begin{array}{rcl} f_{\boldsymbol{A}} : \mathbb{F}_{q^m}^{n-k} \times \mathbb{F}_{q^m}^{n} & \rightarrow & \mathbb{F}_{q^m}^{n} \\ (\boldsymbol{s}, \boldsymbol{e}) & \mapsto & \boldsymbol{s}\boldsymbol{A} + \boldsymbol{e} \end{array}$$

The matrix $\boldsymbol{A}$ will be generated with a trapdoor $\boldsymbol{T}$ such that $f_{\boldsymbol{A}}$ is a trapdoor function : from a random $\boldsymbol{p} \in \mathbb{F}_{q^m}^{n}$, with the trapdoor $\boldsymbol{T}$, one can sample $(\boldsymbol{s}, \boldsymbol{e}) = f_{\boldsymbol{A}}^{-1}(\boldsymbol{p})$ such that $\boldsymbol{e}$ is indistinguishable from a random element in W_r, the set of all the words of rank r and of length n, as defined in RankPKE. These properties will be sufficient for us to construct an IBE and reduce its security to the security of RankPKE. We now describe how we can get such a trapdoor function by relying on the RankSign system [45].

RankSign

RankSign is a signature scheme based on the rank metric. Like other signature schemes based on coding theory [24], RankSign needs a family of codes with an efficient decoding algorithm. It takes on input a random word of the syndrome space (obtained from the hash of the file we want to sign) and outputs a word of small weight with the given syndrome. This is an instance of the RSD problem, with the difference that the matrix $\boldsymbol{H}$ has a trapdoor which makes the problem easy. The public key is a description of the code which hides its structure and the secret key, on the contrary, reveals the structure of the code, which allows the signer to solve the RSD problem. RankSign does not compute a codeword of weight below the Gilbert-Varshamov bound, but instead a codeword of weight r between the Gilbert-Varshamov bound and the Singleton bound. The idea is to use a family of the augmented Low Rank Parity Check codes (denoted $LRPC^+$), and an

adapted decoding algorithm (called the General Errors/Erasures Decoding algorithm) to produce such a codeword from any syndrome. The decoding algorithm is probabilistic and the parameters of the code have to be chosen precisely in order to have a probability of success very close to 1. We refer to [45] for a complete description of the decoding algorithm and the signature algorithm.

Définition 6.4.2 (Augmented Low Rank Parity Check Codes)**.** *Let $\boldsymbol{H}$ be an $\mathbb{F}_{q^m}^{(n-k)\times n}$ homogeneous matrix of full-rank and of weight d and $\boldsymbol{R} \in \mathbb{F}_{q^m}^{(n-k)\times t}$ be a random matrix. Let $\boldsymbol{P} \in GL_{n-k}(\mathbb{F}_{q^m})$ and $\boldsymbol{Q} \in GL_{n+t}(\mathbb{F}_q)$ be two invertible matrices (remark that the coefficients of $\boldsymbol{Q}$ belong to the base field). Let $\boldsymbol{H}' = \boldsymbol{P}(\boldsymbol{R}|\boldsymbol{H})\boldsymbol{Q}$ be the parity-check matrix of a code $\mathcal{C}$ of type $[n+t, t+k]$. By definition, such a code is an $LRPC^+$ code. If $t = 0$, $\mathcal{C}$ is an LRPC code.*

The public key of RankSign is the matrix $\boldsymbol{H}'$, the secret key is the structured matrix $(\boldsymbol{R}|\boldsymbol{H})$ and the trapdoor is the pair of matrices $(\boldsymbol{P}, \boldsymbol{Q})$.

We can now describe the trapdoor function $f_{\boldsymbol{A}}^{-1}$. Let $\boldsymbol{p} \in \mathbb{F}_{q^m}^{n+t}$ and $\boldsymbol{H}'$ the public key of an instance of RankSign. We choose $\boldsymbol{A}$ as a generator matrix of a code with parity-check matrix $\boldsymbol{H}'$, i.e. as a full-rank matrix over $\mathbb{F}_{q^m}$ of size $(k+t) \times (n+t)$ which is such that $\boldsymbol{H}'\boldsymbol{A}^T = 0$. First, we compute $\boldsymbol{H}'\boldsymbol{p}$ and then we apply RankSign with trapdoor T to this syndrome to obtain a vector $\boldsymbol{e}$ of weight r such that $\boldsymbol{H}'\boldsymbol{p}^T = \boldsymbol{H}'\boldsymbol{e}^T$. Finally, we solve the linear system $\boldsymbol{s}\boldsymbol{A} = \boldsymbol{p} - \boldsymbol{e}$ of unknown $\boldsymbol{s}$ and the secret key associated to $\boldsymbol{p}$ is set to be $\boldsymbol{s}$. The security of the RankSign system is based on the assumption that $\boldsymbol{H}'$ is computationally indistinguishable from a random matrix.

Définition 6.4.3 (LRPC$^+$ problem[45])**.** *Given an augmented LRPC code, distinguish it from a random code with the same parameters.*

The hardness of this problem is studied in [45]. Currently the best attacks consist in recovering the structure of the LRPC by looking for small-weight words in the code, and the best algorithms for that are generic algorithms whose complexity is exponential [55].

Proposition 6.4.4. *Let $\boldsymbol{H}'$ be a public RankSign matrix and $\boldsymbol{A}$ be a generator matrix of the associated code. The two following distributions are computationally indistinguishable :*

Let $\mathcal{D}_0$ the distribution $(\boldsymbol{p}, \boldsymbol{s}, \boldsymbol{e})$ where $\boldsymbol{p} \xleftarrow{\$} \mathbb{F}_{q^m}^{n+t}$, $\boldsymbol{e} \in W_r$ is sampled from RingSign Algorithm such that $\boldsymbol{H}'\boldsymbol{e}^T = \boldsymbol{H}'\boldsymbol{p}^T$ and $\boldsymbol{s}$ is the solution of the linear system $\boldsymbol{x}\boldsymbol{A} = \boldsymbol{p} - \boldsymbol{e}$ of unknown $\boldsymbol{x}$.

Let $\mathcal{D}_1$ be the distribution $(\boldsymbol{p}', \boldsymbol{s}', \boldsymbol{e}')$ with $\boldsymbol{s}' \xleftarrow{\$} \mathbb{F}_{q^m}^{k+t}$, $\boldsymbol{e}' \xleftarrow{\$} W_r$ and $\boldsymbol{p}' = \boldsymbol{s}'\boldsymbol{A} + \boldsymbol{e}'$.

Precisely, the maximum advantage ϵ of the adversaries to distinguish $\mathcal{D}_0$ et $\mathcal{D}_1$ is bounded by : $\epsilon \leq \frac{2}{q} + \epsilon_{\mathsf{drsd}}$

Démonstration. Let $\mathcal{D}_2$ be the distribution $(\boldsymbol{s}, \boldsymbol{e})$ where $\boldsymbol{s} \xleftarrow{\$} \mathbb{F}_{q^m}^{n-k}$ and $\boldsymbol{e}$ is a signature of $\boldsymbol{s}$ by RankSign with the public key $\boldsymbol{H}'$ (*i.e,* $\|\boldsymbol{e}\| = r$ and $\boldsymbol{H}'\boldsymbol{e}^T = \boldsymbol{s}$). Let $\mathcal{D}_3$ be the distribution $(\boldsymbol{H}'\boldsymbol{e}'^T, \boldsymbol{e}'^T)$ with $\boldsymbol{e}' \xleftarrow{\$} W_r$.

According to the proof of Theorem 2 of [45], a sample $(\boldsymbol{H}'\boldsymbol{e}'^T, \boldsymbol{e}'^T) \leftarrow \mathcal{D}_3$ is distributed exactly as $\mathcal{D}_2$ except if $(\boldsymbol{H}'\boldsymbol{e}'^T, \boldsymbol{e}'^T)$ is not T-decodable and the probability that the latter occurs is less than $\frac{2}{q}$. Therefore an adversary can not distinguish $\mathcal{D}_2$ from $\mathcal{D}_3$ with an advantage larger than $\frac{2}{q}$.

Now, we can prove the proposition. First, let us examine the distribution $\mathcal{D}_0$. Since $\boldsymbol{H}'$ is a linear map and $\boldsymbol{p} \xleftarrow{\$} \mathbb{F}_{q^m}^{n+t}$, $\boldsymbol{s} = \boldsymbol{H}'\boldsymbol{p}^T$ is uniformly distributed among $\mathbb{F}_{q^m}^{n-k}$. This implies $(\boldsymbol{\sigma}, \boldsymbol{e}) \leftarrow \mathcal{D}_2$. Moreover, $\boldsymbol{p} - \boldsymbol{e}$ is uniformly distributed among the words of the code generated by $\boldsymbol{A}$, hence $\boldsymbol{s} \xleftarrow{\$} \mathbb{F}_{q^m}^{k+t}$.

According to the indistinguishability of $\mathcal{D}_2$ and $\mathcal{D}_3$, the distribution of $\boldsymbol{e}'$ and $\boldsymbol{e}$ are computationally indistinguishable. $\boldsymbol{s}'$ and $\boldsymbol{s}$ are both uniformly distributed. Finally, based on the assumption that the DRSD problem is hard, $\boldsymbol{p}'$ and $\boldsymbol{p}$ are indistinguishable.

Summing up these two steps, the advantage of an adversary to distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$ is bounded by $\frac{2}{q} + \epsilon_{\mathsf{drsd}}$. □

□

6.4.2 Scheme

Our IBE system uses a random oracle H which maps the identity into the public keys space $\mathbb{F}_{q^m}^{n+t}$ of our encryption scheme.

- IBE.Setup
  - choose the parameters (n, m, k, d, t) of the scheme according to RankSign. The secret master key is the triplet of matrices $\boldsymbol{P}, (\boldsymbol{R}|\boldsymbol{H})$ and $\boldsymbol{Q}$ such that $\boldsymbol{H}$ is a parity-check matrix of an $[n, k]$ LRPC code of weight d over $\mathbb{F}_{q^m}$, $\boldsymbol{R} \xleftarrow{\$} \mathbb{F}_{q^m}^{(n-k)\times t}$, $\boldsymbol{P} \xleftarrow{\$} GL_{n-k}(\mathbb{F}_{q^m})$ and $\boldsymbol{Q} \xleftarrow{\$} GL_{n+t}(\mathbb{F}_q)$. Let $\boldsymbol{A}$ be a full rank

$(k+t)\times(n+t)$ matrix over $\mathbb{F}_{q^m}$ such $\boldsymbol{H}'\boldsymbol{A}^T = 0$ with $\boldsymbol{H}' = \boldsymbol{P}(\boldsymbol{R}|\boldsymbol{H})\boldsymbol{Q}$ and the trapdoor $\boldsymbol{T}$ is $(\boldsymbol{P}, \boldsymbol{Q})$.
  - define $\boldsymbol{G} \in \mathbb{F}_{q^m}^{k'\times n'}$ to be a generator matrix of a public code $\mathcal{C}'$ which can decode (efficiently) errors of weight up to wr as in RankPKE.KeyGen.
  - return $mpk = (\boldsymbol{A}, \boldsymbol{G})$ and $msk = \boldsymbol{T}$
- IBE.KeyDer$(\boldsymbol{A}, \boldsymbol{T}, id)$:
  - compute $\boldsymbol{p} = H(id)$
  - compute $(\boldsymbol{s}, \boldsymbol{e}) = f_{\boldsymbol{A}}^{-1}(\boldsymbol{p})$ by using the trapdoor $\boldsymbol{T}$
  - store $(id, \boldsymbol{s})$ and return $\boldsymbol{s}$
- IBE.Enc$(id, \boldsymbol{m})$:
  - compute $\boldsymbol{p} = H(id)$
  - return $\boldsymbol{c} =$ RankPKE.Enc$((\boldsymbol{A}, \boldsymbol{p}, \boldsymbol{G}), \boldsymbol{m})$
- IBE.Dec$(\boldsymbol{s}, \boldsymbol{c})$: return RankPKE.Dec$(\boldsymbol{s}, \boldsymbol{c})$.

6.4.3 Security

We now state the security of the IBE system.

Théorème 6.4.5. *Under the assumption that the* LRPC$^+$ *problem is hard and the* RankPKE *is secure, the* IBE *system described above is* IND − CPA*-secure in the random oracle model :*

$$\epsilon_{\mathsf{ibe}} \leq \frac{2q_H}{q} + \epsilon_{\mathsf{lrpc}^+} + q_H(\epsilon_{\mathsf{drsd}} + \epsilon_{\mathsf{pke}})\ ^{1}$$

where $\epsilon_{\mathsf{lrpc}^+}, \epsilon_{\mathsf{pke}}, \epsilon_{\mathsf{ibe}}$ *are respectively the bound on the advantage of the attacks against the* LRPC$^+$ *problem, the* RankPKE *system and the* IBE *system, and* q_H *is the maximum number of distinct hash queries to* H *that an adversary can make.*

Démonstration. We proceed by a sequence of games.

Game $\mathbf{G}_0$: This is the real IND-CPA attack game. The IBE.Setup is run and then, a 2-stage poly-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is fed with the public key $mpk = (\boldsymbol{A}, \boldsymbol{G})$. $\mathcal{A}_1$

1. As in the lattice-based IBE scheme of Gentry, Peikert, and Vaikuntanathan [49], we lose a factor q_H in the reduction from PKE to IBE. Moreover, because of the lack of a statistical indistinguishability in the preimage sampling as in [49], we also lose an additional cost of $\frac{2q_H}{q} + q_H\epsilon_{\mathsf{drsd}}$ which require us to use a large q. Fortunately, the efficiency of our scheme is $\mathcal{O}\big(\log q\big)$.

can ask queries to H and key queries. Then, $\mathcal{A}_1$ outputs a challenge identity $id^\star$, which is different from the key queries $\mathcal{A}_1$ already asked, and a pair of messages $(\boldsymbol{m}_0, \boldsymbol{m}_1)$. Next a challenge ciphertext is produced by flipping a coin b and producing a ciphertext $c^\star = \mathsf{IBE.Enc}(id^\star, \boldsymbol{m}_b)$.

On input $c^\star$, $\mathcal{A}_2$ can continue to ask queries to H and key queries which are different from $id^\star$, and finally outputs bit b'. We denote by S_0 the event $b' = b$ and use the same notation S_n in any game $\mathbf{G}_n$ below.

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{IBE}}(\mathcal{A}) = \mid 2\Pr[\mathrm{S}_0] - 1 \mid$$

We assume without loss of generality that, for any identity id that $\mathcal{A}$ wants to corrupt, $\mathcal{A}$ already queried H on id. In particular, we can assume that $\mathcal{A}$ will query the challenge identity $id^\star$ to H.

As this is the real attack game, for a key query on an identity id, the $\mathsf{IBE.KeyDer}(\boldsymbol{A}, \boldsymbol{T}, id)$ is run and the secret key is given to $\mathcal{A}$. We recall this algorithm :

- compute $\boldsymbol{p} = H(id)$
- compute $(\boldsymbol{s}, \boldsymbol{e}) = f_{\boldsymbol{A}}^{-1}(\boldsymbol{p})$ by using the trapdoor $\boldsymbol{T}$:
  - compute $\boldsymbol{H}'\boldsymbol{p}$ and then we apply RankSign with trapdoor $\boldsymbol{T}$ to this syndrome to obtain a vector $\boldsymbol{e}$ of weight r such that $\boldsymbol{H}'\boldsymbol{p} = \boldsymbol{H}'\boldsymbol{e}$.
  - solve the linear system $\boldsymbol{s}\boldsymbol{A} = \boldsymbol{p} - \boldsymbol{e}$ of unknown $\boldsymbol{s}$ and the secret key associated to $\boldsymbol{p}$ is set to be $\boldsymbol{s}$.
- store $(id, \boldsymbol{s})$ and return $\boldsymbol{s}$.

Game $\mathbf{G}_1$: In this game, we modify the answers to the key queries so that it does not require the trapdoor $\boldsymbol{T}$ anymore. In order to make the answers coherent, we also need to simulate the queries to the hash queries to H. We maintain a list List_H, initially set to empty, to store the tuples $(id, \boldsymbol{p}, \boldsymbol{s})$ where $\boldsymbol{p}$ is the value that we respond to the H query on id, and $\boldsymbol{s}$ is the secret key which corresponds to the public key $\boldsymbol{p}$ we generate. The simulation is given in the following way :

- Hash queries : on $\mathcal{A}$'s jth distinct query id_j to H :
  - randomly choose a vector $\boldsymbol{e}_j$ of weight r
  - randomly choose $\boldsymbol{s}_j$
  - define $\boldsymbol{p}_j = H(id) = \boldsymbol{s}_j\boldsymbol{A} + \boldsymbol{e}_j$
  - add the tuple $(id_j, \boldsymbol{p}_j, \boldsymbol{s}_j)$ to List_H and return $\boldsymbol{p}_j$ to $\mathcal{A}$.

— Secret key queries : when $\mathcal{A}$ asks for a secret key for the identity id, we retrieve the tuple $(id, \boldsymbol{p}, \boldsymbol{s})$ from the List_H and return $\boldsymbol{s}$ to $\mathcal{A}$.

Now, looking back at the Proposition 6.4.4, we remark that the set of q_H samples $(\boldsymbol{p}_j, \boldsymbol{s}_j, \boldsymbol{e}_j)$ in the previous game come from the distribution $\mathcal{D}_0^{q_H}$ and the set of q_H samples $(\boldsymbol{p}_j, \boldsymbol{s}_j, \boldsymbol{e}_j)$ in this game come from the distribution $\mathcal{D}_1^{q_H}$. We thus have :

$$|\Pr[\mathrm{S}_1] - \Pr[\mathrm{S}_0]| \leq \frac{2q_H}{q} + q_H \epsilon_{\mathsf{drsd}}$$

Game G_2: As the objective is to reduce the security of the IBE to the security of RankPKE, in this game, we define the matrix $\boldsymbol{A}$ to be a random matrix as in the RankPKE. Because the simulation in the previous game does not use the trapdoor $\boldsymbol{T}$, we can keep the simulation for hash queries and key queries exactly unchanged. By the assumption that the LRPC^+ problem is hard, this game is indistinguishable from the previous game :

$$|\Pr[\mathrm{S}_2] - \Pr[\mathrm{S}_1]| \leq \epsilon_{\mathsf{lrpc}^+}$$

Game G_3: We can now reduce the security of the IBE in the previous game to the security of RankPKE. We are given the public key $\boldsymbol{p}^\star$ of RankPKE and try to break the semantic security of RankPKE. Intuitively, we proceed as follows. We will try to embed the given public key $\boldsymbol{p}^\star$ of RankPKE to $H(id^\star)$. The IBE for $id^\star$ becomes thus a RankPKE with the same distribution of public keys. We can then use the given challenge ciphertext of RankPKE as the challenge ciphertext to $\mathcal{A}$ and whenever $\mathcal{A}$ can break IBE, we can break RankPKE. The difficulty in this strategy is that we should correctly guess the challenge identity $id^\star$. In a selective game where $\mathcal{A}$ has to announce $id^\star$ at the beginning of the game, we know this identity. However, in the adaptive game that we consider, we need make a guess on the challenge identity among all the identities queried to H. This explains why we lose a factor q_H in the advantage to attack RankPKE.

Now, formally, on input a random matrix $\boldsymbol{A}$ and a public key $\boldsymbol{p}^\star$ for the RankPKE, we choose an index i among $1, \ldots q_H$ uniformly at random and change the answer for the ith query to H and for the challenge as follows :

— Hash queries : on $\mathcal{A}$'s jth distinct query id_j to H : if $j = i$, then add the tuple $(id_j, \boldsymbol{p}^\star, \perp)$ to List_H and return $\boldsymbol{p}^\star$ to $\mathcal{A}$. Otherwise for $j \neq i$, do the same as in the previous game.

— Secret key queries : when $\mathcal{A}$ asks for a secret key for the identity id, retrieve the tuple $(id, \boldsymbol{p}, \boldsymbol{s})$ from the List_H. If $\boldsymbol{s} \neq \perp$, return $\boldsymbol{s}$ to $\mathcal{A}$, otherwise output a random bit and abort.

— Challenge ciphertext : when $\mathcal{A}$ submits a challenge identity $id^\star$, different from all its secret key queries, and two messages m_0, m_1, if $id^\star = id_i$, i.e., $(id^\star, \boldsymbol{p}^\star, \perp) \notin \mathsf{List}_H$, then output a random bit and abort. Otherwise, we also submits the messages m_0, m_1 to the challenger and receive a challenge ciphertext $c^\star$. We return then $c^\star$ to $\mathcal{A}$.

When $\mathcal{A}$ terminates and returns a bit b, we also outputs b. We now analyze the advantage to break RankPKE :

— We do not abort if we made a good guess, *i.e,* $id^\star = id_i$. As i is perfectly hidden from $\mathcal{A}$, the probability that we do not abort is $\frac{1}{q_H}$.

— Conditioned on not aborting, the view we provides to $\mathcal{A}$ is exactly the same as in the previous game. We get thus the same advantage in attacking RankPKE as $\mathcal{A}$'s advantage in attacking IBE

We finally have :

$$| \, 2 \Pr[\mathrm{S}_3] - 1 \, | \leq q_H \epsilon_{\mathsf{pke}}$$

□

6.5 Parameters

In this section, we explain how to construct a set of parameters and give an analysis of the best known attacks against the IBE scheme.

6.5.1 General parameters for RankSign and RankEnc

First, we have to carefully choose the parameters of the algorithm RankSign [45] used for the presampling phase. In the case where only RankPKE is used, the constraints are much weaker. Remember that RankSign is a probabilistic signature algorithm and the probability of returning a valid signature depends on the choice of the parameters. These parameters are :

- q, m : the cardinality of the base field and the degree of the extension field.
- n : the length of the hidden LRPC code used to sign.
- t : the number of random columns added to the LRPC to hide it.
- k, d : the dimension of the LRPC code and the weight of the LRPC code.
- r : the weight of the signature.

The conditions these parameters must verify are [45]

$$n = d(n-k), (r-t)(m-r) + (n-k)(rd-m) = 0, r = t + \frac{n-k}{d}$$

Let us explain the choice of our parameters. First we need to fix d for two reasons :

- if we look at the three conditions, they are homogeneous if d is constant. Thus, we can make other set of parameters from one set by multiply all the parameters (except for d) by a constant.
- d is the weight of the $LRPC^+$ code used for the public master key. It is very important to choose d not too small to ensure the security of the public master key.

Once d is fixed, we can easily test all the valid parameters and choose the most interesting ones, whether we need to optimize the security or the key size.

Then we need to choose the parameters of RankPKE. We need a code which can correct wr errors, where w is the weight of the matrix $\boldsymbol{U}$. We use (n', k', t')-simple codes because because they can asymptotically decode up to d_{GV} errors. In all cases, we have chosen $n' = m$ for simplicity, even if this is not a necessary condition.

Let us describe the size of the keys and of the messages, as well as the computation time of our cryptosystem :

- public master key $\boldsymbol{A}$ is a $(k+t)\times(n+t)$ matrix over $\mathbb{F}_{q^m}$: $(k+t)(n-k)m\lceil\log_2 q\rceil$ bits (under systematic form).
- public key $\boldsymbol{p}_{id}$ is an element of $\mathbb{F}_{q^m}^{n+t}$: $(n+t)m\lceil\log_2 q\rceil$ bits.
- secrete key $\boldsymbol{s}_{id}$ is an element of $\mathbb{F}_{q^m}^{n-k}$: $(n-k)m\lceil\log_2 q\rceil$ bits.
- plaintext $\boldsymbol{m}$ is an element of $\mathbb{F}_{q^m}^{k'}$: $k'm\lceil\log_2 q\rceil$ bits.
- ciphertext is a $(k+t+1)\times n'$ matrix over $\mathbb{F}_{q^m}$: $(k+t+1)n'm\lceil\log_2 q\rceil$ bits.
- to generate the secret key, we need to invert a syndrome with RankSign which takes $(n-k)(n+t)$ multiplications in $\mathbb{F}_{q^m}$ ([45]).
- encryption consists in a multiplication of two matrices of respective sizes $(k+t+1)\times(n+t)$ and $(n+t)\times n'$, which takes $(k+t+1)(n+t)n'$ multiplications in $\mathbb{F}_{q^m}$.

— decryption consists in a multiplication matrix-vector and the decoding of an error of weight wr with a (n', k', t')-simple code, which takes $(k+t+1)n'$ multiplications in $\mathbb{F}_{q^m}$ and $\mathcal{O}\big(((n'-t')wr)^3\big)$ operations in $\mathbb{F}_q$.

A multiplication in $\mathbb{F}_{q^m}$ costs $\tilde{\mathcal{O}}\big(m \log q\big)$ operations in $\mathbb{F}_2$ [79].

6.5.2 Practical evaluation of the security

In order to analyze the security of the IBE, we recall the result of the Theorem 6.4.5 : $\epsilon_{\mathsf{ibe}} \leq \frac{2q_H}{q} + \epsilon_{\mathsf{lrpc}^+} + q_H(\epsilon_{\mathsf{drsd}} + \epsilon_{\mathsf{pke}})$. We want $\epsilon_{\mathsf{ibe}} \leqslant 2^{-\lambda}$, where λ is the security parameter. Since the first term only depends on q and on the number of queries, we need $q > q_H 2^{\lambda+1}$. We stress that the size of the data and the computation time are linear in the logarithm of q. In consequence, it is not a problem to have q exponential in the security parameter. Moreover, since all combinatorial attacks are polynomial in q, they are utterly inefficient to break the IBE.

The second type of attacks are the algebraic attacks. An adversary can either attack the public master key $\boldsymbol{A}$ by solving an instance of LRPC^+ problem, a public key $\boldsymbol{p}$ of an user by solving an instance of DRSD or a ciphertext by solving an instance of RSL. By using the results in [10], we can estimate the complexity of the attacks and adapt the parameters in consequence.

We give an example of a set of parameters in the following table. We take the standard values $\lambda = 128$ for the security parameter and $q_H = 2^{60}$.

n	$n-k$	m	q	d	t	r	d_{GV}	d_{Sing}	Public master key size (Bytes)
100	20	96	2^{192}	5	12	16	11	20	4,239,360

n'	k'	t'	w	Probability of failure
96	9	66	4	2^{-576}

The decoding algorithm for the simple codes is probabilistic, that is why there is a probability p_f that the decoding fails. However, $p_f \approx \frac{1}{q^{t'-wr+1}}$, since we have a very large q in this example, p_f is negligible. These parameters are large but still tractable, for a first code-based IBE scheme in post-quantum cryptography.

Chapitre 7

Conclusions et perspectives

Cette thèse a permis d'étudier les codes en métrique rang à la fois du point de vue de la cryptanalyse et de la cryptographie.

Du point de vue des attaques, nous sommes parvenus à améliorer les attaques génériques en exploitant pleinement la $\mathbb{F}_{q^m}$-linéarité, ce qui a permis de diminuer la complexité du meilleur algorithme connu. Nous avons aussi exploité la structure des codes cycliques en métrique rang pour aboutir à une attaque spécifique à ces codes, ce qui a donné lieu à un premier article [55]. Cette dernière attaque n'exclut pas l'utilisation des codes cycliques en cryptographie mais montre qu'il existe des clés faibles. Il faut donc soigneusement choisir les paramètres des codes cycliques en métrique rang afin d'éviter cette attaque.

Enfin nous avons aussi analysé les améliorations qu'apportent les algorithmes quantiques dans la résolution des problèmes génériques afin de pouvoir choisir des paramètres résistants à l'ordinateur quantique.

Du point de vue de la cryptographie, nous avons adapté le générateur aléatoire de Fischer-Stern [34] à la métrique rang. Grâce aux propriétés de la métrique rang, nous obtenons des meilleures performances qu'en métrique de Hamming.

Nous avons aussi conçu un nouveau cryptosystème de chiffrement à clé publique RankPKE dont les clés sont choisies aléatoirement. La sécurité de ce cryptosystème est basée sur le nouveau problème $DRSL$. Cela montre la richesse de la métrique rang et l'importance

de chercher et d'étudier de nouvelles primitives en cryptographie. Ce cryptosystème associé à la signature RankSign a permis la création du premier IBE basé sur des problèmes difficiles de théorie des codes. Ces résultats ont été publiés dans [38].

De nombreuses questions restent ouvertes concernant la métrique rang.

Du point de vue des attaques, il est intéressant de se demander si les techniques utilisées pour améliorer les algorithmes de décodage générique ou de recherche de mots de petit poids en métrique de Hamming peuvent s'adapter en métrique rang. Nonobstant nos efforts, nous n'avons pas trouvé d'équivalent au paradoxe des anniversaires en métrique rang. Cela est en partie dû au fait que le support correspond à un espace vectoriel et non à un ensemble de coordonnées non nulles.

Concernant les primitives cryptographiques, il n'existe pas à l'heure actuelle de fonction de hachage en métrique rang résistante aux collisions. Une autre voie à explorer dans la continuité de l'IBE est d'obtenir d'autres primitives plus complexes, comme l'offuscation ou le chiffrement fonctionnel.

En ce qui concerne les problèmes difficiles en métrique rang, on peut se demander s'il existe une réduction search-to-decision du problème RSD qui utiliserait directement le théorème de Goldreich-Levin comme en métrique de Hamming. D'autre part, bien qu'une réduction probabiliste à un problème NP-complet suffise à prouver que les problèmes de recherche de mots de petit poids et de décodage par syndrome sont difficiles, il serait théoriquement important de savoir si ces problèmes sont ou non NP-complets (ou NP-difficiles dans leur version search).

Pour la suite de nos recherches, de nouveaux standards en cryptographie post-quantique vont être sélectionnés par le NIST (National Institute of Standards and Technology) à la suite d'un concours similaire à celui qui a permis d'établir l'AES. Cela ouvre la voie à de nombreuses opportunités de recherche pour l'avenir. Une autre piste à explorer est la conception de cryptosystèmes dont la sécurité repose sur des problèmes génériques (par exemple le décodage d'un code quasi-cyclique aléatoire [2, 27]) et non sur le masquage d'une familles de codes structurés.

Publications Personnelles

dans des conférences internationales à comité de relecture :

Hauteville, A., & Tillich, J. P. (2015, June). New algorithms for decoding in the rank metric and an attack on the LRPC cryptosystem. In Information Theory (ISIT), 2015 IEEE International Symposium on (pp. 2747-2751). IEEE.

Gaborit, P., Hauteville, A., & Tillich, J. P. (2016, February). Ranksynd a PRNG based on rank metric. In International Workshop on Post-Quantum Cryptography (pp. 18-28). Springer International Publishing.

Gaborit, P., Hauteville, A., Phan, D. H., & Tillich, J. P. (2017, August). Identity-based encryption from codes with rank metric. In Annual International Cryptology Conference (pp. 194-224). Springer, Cham.

en cours de rédaction

Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, Jean-Pierre Tillich. Improvement of Generic Attacks on the Rank Syndrome Decoding Problem. 2017. hal-01618463

Bibliographie

[1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, May 2010.

[2] Carlos Aguilar Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Efficient encryption from random quasi-cyclic codes. *CoRR*, abs/1612.05572, 2016.

[3] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Advances in Cryptology–EUROCRYPT 2012*, pages 719–737. Springer, 2012.

[4] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 1–16, Copenhagen, Denmark, May 2014. Springer.

[5] Paulo S.L.M Barreto, Rafael Misoczki, and Marcos A. Jr. Simplicio. One-time signature scheme from syndrome decoding over generic error-correcting codes. *Journal of Systems and Software*, 84(2) :198–204, 2011.

[6] Côme Berbain, Henri Gilbert, and Jacques Patarin. Quad : A practical stream cipher with provable security. In *Advances in Cryptology-EUROCRYPT 2006*, pages 109–128. Springer, 2006.

[7] Thierry P. Berger and Pierre Loidreau. Security of the Niederreiter form of the GPT public-key cryptosystem. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2002*, page 267. IEEE, June 2002.

[8] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3) :384–386, May 1978.

[9] Daniel J. Bernstein. Grover vs. McEliece. In Nicolas Sendrier, editor, *Post-Quantum Cryptography 2010*, volume 6061 of *LNCS*, pages 73–80. Springer, 2010.

[10] Luk Bettale. *Cryptanalyse algébrique : outils et applications*. PhD thesis, Université Pierre et Marie Curie - Paris 6, 2012.

[11] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on computing*, 15(2) :364–383, 1986.

[12] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM journal on Computing*, 13(4) :850–864, 1984.

[13] Dan Boneh. The decision Diffie-Hellman problem. In *International Algorithmic Number Theory Symposium*, pages 48–63. Springer, 1998.

[14] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, May 2004.

[15] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, August 2004.

[16] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, August 2001.

[17] Xavier Boyen. Lattice mixing and vanishing trapdoors : A framework for fully secure short signatures and more. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 499–517. Springer, May 2010.

[18] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortsch. Phys.*, 46 :493, 1998.

[19] Jonathan F. Buss, Gudmund S. Frandsen, and Jeffrey O. Shallit. The computational complexity of some problems of linear algebra. *J. Comput. System Sci.*, 58(3) :572–596, June 1999.

[20] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3) :265–294, July 2007.

[21] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, May 2010.

[22] Pierre-Louis Cayrel, Ayoub Otmani, and Damien Vergnaud. On Kabatianskii-Krouk-Smeets signatures. In *Arithmetic of Finite Fields - WAIFI 2007*, volume 4547 of *LNCS*, pages 237–251, Madrid, Spain, June 21–22 2007.

[23] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 360–363. Springer, December 2001.

[24] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 157–174. Springer, December 2001.

[25] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In *Advances in Cryptology - EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 559–585, Vienna, Austria, May 2016. Springer.

[26] Philippe Delsarte. Bilinear forms over a finite field, with applications to coding theory. *J. Comb. Theory, Ser. A*, 25(3) :226–241, 1978.

[27] Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Ouroboros : a simple, secure and efficient key exchange protocol based on coding theory. In *International Workshop on Post-Quantum Cryptography*, pages 18–34. Springer, 2017.

[28] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6) :644–654, 1976.

[29] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, August 1984.

[30] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. Computing loci of rank defects of linear matrices using gröbner bases and applications to cryptology. In *ISSAC, 2010, Proceedings*, pages 257–264, 2010.

[31] Jean-Charles Faugère, Françoise Levy-dit Vehel, , and Ludovic Perret. Cryptanalysis of Minrank. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *LNCS*, pages 280–296, 2008.

[32] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. Structural weakness of compact variants of the McEliece cryptosystem. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2014*, pages 1717–1721, Honolulu, HI, USA, July 2014.

[33] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. Structural cryptanalysis of McEliece schemes with compact keys. *Des. Codes Cryptogr.*, 79(1) :87–112, 2016.

[34] Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *LNCS*, pages 245–255. Springer, 1996.

[35] Ernest Mukhamedovich Gabidulin. Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1) :3–16, 1985.

[36] Ernst M. Gabidulin, Alexei V. Ourivski, Bahram Honary, and Bassem Ammar. Reducible rank codes and their applications to cryptography. *IEEE Trans. Inform. Theory*, 49(12) :3289–3293, 2003.

[37] Ernst M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and their applications to cryptography. In *Advances in Cryptology - EUROCRYPT'91*, number 547 in LNCS, pages 482–489, Brighton, April 1991.

[38] Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. Identity-based encryption from rank metric. In *Advances in Cryptology - CRYPTO2017*, volume 10403 of *LNCS*, pages 194–226. Springer, August 2017.

[39] Philippe Gaborit, Cédric Lauradoux, and Nicolas Sendrier. SYND : a fast code-based stream cipher with a security reduction. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 186–190, Nice, France, June 2007.

[40] Philippe Gaborit, Gaétan Murat, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes and their application to cryptography. In *Proceedings of the Workshop on Coding and Cryptography WCC'2013*, Bergen, Norway, 2013. Available on `www.selmer.uib.no/WCC2013/pdfs/Gaborit.pdf`.

[41] Philippe Gaborit, Duong Hieu Phan, Adrien Hauteville, and Jean-Pierre Tillich. Identity-based encryption from codes with rank metric, full version. IACR Cryptology ePrint Archive, Report2017/623, 2017. `http://eprint.iacr.org/`.

[42] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. *CoRR*, abs/1301.1026, 2013.

[43] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. *IEEE Trans. Information Theory*, 62(2) :1006–1019, 2016.

[44] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. New results for rank-based cryptography. In *Progress in Cryptology - AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 1–12, 2014.

[45] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. Ranksign : An efficient signature algorithm based on the rank metric. In *Post-Quantum Cryptography 2014*, volume 8772 of *LNCS*, pages 88–107. Springer, 2014.

[46] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. Ranksign : An efficient signature algorithm based on the rank metric (extended version on arxiv). In *Post-Quantum Cryptography 2014*, volume 8772 of *LNCS*, pages 88–107. Springer, 2014.

[47] Philippe Gaborit and Gilles Zémor. On the hardness of the decoding and the minimum distance problems for rank codes. *IEEE Trans. Information Theory*, 62(12) :7245–7252, 2016.

[48] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464. Springer, May / June 2006.

[49] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[50] Keith Gibson. Severely denting the Gabidulin version of the McEliece public key cryptosystem. *Des. Codes Cryptogr.*, 6(1) :37–45, 1995.

[51] Keith Gibson. The security of the Gabidulin public key cryptosystem. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 212–223. Springer, 1996.

[52] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2) :270–299, 1984.

[53] L. K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79 :325, 1997.

[54] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4) :1364–1396, 1999.

[55] Adrien Hauteville and Jean-Pierre Tillich. New algorithms for decoding in the rank metric and an attack on the LRPC cryptosystem, 2015. abs/1504.05431.

[56] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU : A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *LNCS*, pages 267–288. Springer, 1998.

[57] Gregory Kabatianskii, Evgenii Krouk, and Sergei Semenov. *Error Correcting Coding and Security for Data Networks : Analysis of the Superchannel Concept.* John Wiley & Sons, 2005.

[58] Gregory Kabatianskii, Evgenii Krouk, and Ben. J. M. Smeets. A digital signature scheme based on random error-correcting codes. In *IMA Int. Conf.*, volume 1355 of *LNCS*, pages 161–167. Springer, 1997.

[59] Kristine Lally and Patrick Fitzpatrick. Algebraic structure of quasicyclic codes. *Discrete applied Math.*, 111(1) :157–175, 2001.

[60] Leonid A Levin. One way functions and pseudorandom generators. *Combinatorica*, 7(4) :357–363, 1987.

[61] Françoise Lévy-dit Vehel and Ludovic Perret. Algebraic decoding of codes in rank metric. In *proceedings of YACC06*, Porquerolles, France, June 2006. available on `http://grim.univ-tln.fr/YACC06/abstracts-yacc06.pdf`.

[62] San Ling and Patrick Solé. On the algebraic structure of quasi-cyclic codes. I. finite fields. *IEEE Trans. Inform. Theory*, 47(7) :2751–2760, 2001.

[63] Pierre Loidreau. Asymptotic behaviour of codes in rank metric over finite fields. *Des. Codes Cryptogr.*, 71(1) :105–118, 2014.

[64] Pierre Loidreau. On cellular code and their cryptographic applications. In I. Landjev G. Kabatiansky, editor, *Proceedings of ACCT14 (algebraic and combinatorial coding theory)*, pages 234–239, Svetlogorsk, Russia, September 2014.

[65] Pierre Loidreau. A new rank metric codes based encryption scheme. In *International Workshop on Post-Quantum Cryptography*, pages 3–17. Springer, 2017.

[66] Robert J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[67] Mohammed Meziani, Gerhard Hoffmann, and Pierre-Louis Cayrel. Improving the Performance of the SYND Stream Cipher. In *Progress in Cryptology - AFRICACRYPT 2012*, volume 7374 of *LNCS*, pages 99–116. Springer, 2012.

[68] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece : New McEliece variants from moderate density parity-check codes. *IACR Cryptology ePrint Archive, Report2012/409*, 2012, 2012.

[69] Oystein Ore. On a special class of polynomials. *Transactions of the American Mathematical Society*, 35(3) :559–584, 1933.

[70] Ayoub Otmani and Jean-Pierre Tillich. An efficient attack on all concrete KKS proposals. In *Post-Quantum Cryptography 2011*, volume 7071 of *LNCS*, pages 98–116, 2011.

[71] Alexei V. Ourivski and Thomas Johansson. New technique for decoding codes in the rank metric and its cryptography applications. *Problems of Information Transmission*, 38(3) :237–246, 2002.

[72] Raphael Overbeck. Extending Gibson's attacks on the GPT cryptosystem. In Oyvind Ytrehus, editor, *WCC 2005*, volume 3969 of *LNCS*, pages 178–188. Springer, 2005.

[73] Raphael Overbeck. A new structural attack for GPT and variants. In *Mycrypt*, volume 3715 of *LNCS*, pages 50–63, 2005.

[74] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[75] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000.

[76] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, August 1984.

[77] Danilo Silva, Frank R. Kschischang, and Ralf Kötter. Communication over finite-field matrix channels. *IEEE Trans. Information Theory*, 56(3) :1296–1305, 2010.

[78] Alexander Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory*, 43(6) :1757–1766, November 1997.

[79] Joachim von zur Gathen and Jurgen Gerhard. *Modern computer algebra*. Cambridge University Press, 2003.

[80] Brent Waters. Dual system encryption : Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, August 2009.

[81] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005.

[82] Andrew C Yao. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 80–91. IEEE, 1982.