Thèse de doctorat

Université de Limoges

École Doctorale Sciences et Ingénierie des Systèmes, Mathématiques, Informatique

Cryptographie à base de codes correcteurs d'erreurs en métrique rang et applications

Nicolas ARAGON

Remerciements

Tout d'abord je tiens à remercier mes directeurs de thèse, Philippe Gaborit et Gilles Zémor, pour m'avoir permis de découvrir le monde de la recherche grâce à leur soutien et leurs conseils. M'impliquer dans ces projets avec vous a été une expérience très enrichissante pour moi.

J'aimerais également adresser mes remerciements à mes rapporteurs, Alain Couvreur et Ayoub Otmani, pour leurs commentaires qui m'ont permis d'améliorer la qualité de ce manuscrit.

Les échanges que j'ai pu avoir tout au long de ma thèse avec mes co-auteurs et plus largement mes collègues ont également été riches en enseignements, merci pour tous ces moments.

Merci également à mes collègues et amis doctorants, pour les discussions scientifiques (ou non) durant les nombreuses pauses cafés effectuées durant ces trois années.

Enfin je termine en remerciant tout particulièrement Chloé pour son soutien durant cette thèse.

Table des matières

Table des figures

Liste des tableaux

Première partie

Introduction

Chapitre 1

Introduction

Contexte

Cryptographie

La cryptographie se découpe en deux parties distinctes : la cryptographie à clé secrète et la cryptographie à clé publique.

La cryptographie à clé secrète, la plus ancienne, nécessite que les deux parties souhaitant communiquer échangent un secret commun avant la communication : la clé de chiffrement. Cette clé est commune aux deux parties, et est utilisée à la fois pour l'opération de chiffrement et de déchiffrement. La cryptographie à clé secrète existe depuis l'Antiquité : on peut notamment citer le très connu chiffrement de César.

En 1976, Diffie et Hellman, dans "New directions in cryptography" [28] proposent des techniques permettant de réaliser des opérations de chiffrement et signature sans que les deux parties n'aient besoin de posséder un secret commun au préalable : la cryptographie à clé publique. Deux clés distinctes sont utilisées pour les opérations de chiffrement et de déchiffrement : une première clé, publique, permet de chiffrer, et la clé secrète, seulement connue du destinataire des messages, permet de déchiffrer.

Les schémas de chiffrement à clé publique les plus utilisés actuellement reposent sur les problèmes de la factorisation [65] et du logarithme discret [30]. Bien qu'efficaces, ces schémas seraient voués à disparaitre si un ordinateur quantique de grande taille était construit : l'algorithme de Shor [66] permet de résoudre ces deux problèmes en temps polynomial.

C'est dans ce contexte que le National Institute for Standards and Technology (NIST) a lancé en 2017 un appel à projet dans le but de standardiser de nouveaux cryptosystèmes "post-quantiques", c'est-à-dire résistants aux attaques réalisées par un ordinateur quantique.

Les propositions peuvent être séparées en plusieurs catégories, en fonction du problème sur lequel repose leur sécurité :

— Cryptographie basée sur les codes correcteurs d'erreurs

— Cryptographie basée sur les réseaux euclidiens

— Cryptographie multivariée

— Cryptographie basée sur les fonctions de hachage

Dans ce manuscrit nous allons nous intéresser à la cryptographie basée sur les codes correcteurs d'erreurs.

Cryptographie basée sur les codes correcteurs d'erreurs

Les codes correcteurs d'erreurs sont un outil permettant de transmettre de manière fiable des informations sur un canal de communication bruité. Parmi leurs utilisations, on peut citer les supports de stockage de données (CD, DVD, disques durs...) ou encore les télécommunications.

Le principe est le suivant : pour transmettre un message $\boldsymbol{m}$ de k symboles, on commence par **encoder** ce message en un mot de code de n symboles. Au cours de la transmission, un certain nombre de symboles sont altérés : on doit donc appliquer une opération de **décodage** afin de retrouver le message original $\boldsymbol{m}$.

Les codes correcteurs d'erreurs utilisés pour transmettre des informations sont des codes structurés, pour lesquels on connait un algorithme de décodage en temps polynomial. Pour un code aléatoire, le coût du décodage est exponentiel.

En 1978, McEliece décrit dans [58] un cryptosystème basé sur les codes correcteurs d'erreurs. Ce schéma de chiffrement est peu utilisé, notamment à cause de la taille de ses clés, mais la recherche dans le domaine de la cryptographie basée sur les codes correcteurs a permis de développer de nombreux compromis entre sécurité, taille des clés et rapidité de chiffrement. Le schéma fonctionne de la manière suivante :

1. **Génération de clés**
   — On choisit un code $\mathcal{C}$ de longueur n et de dimension k pour lequel on sait décoder t erreurs en temps polynomial. Soit $\boldsymbol{G}$ une matrice génératrice de $\mathcal{C}$.
   — On tire aléatoirement une matrice inversible $\boldsymbol{S}$ de taille $k \times k$ et une matrice de permutation $\boldsymbol{P}$ de taille $n \times n$.
   — On calcule $\hat{\boldsymbol{G}} = \boldsymbol{SGP}$.
   — $\hat{\boldsymbol{G}}$ est la clé publique et $(\boldsymbol{S}, \boldsymbol{G}, \boldsymbol{P})$ est la clé privée.
2. **Chiffrement d'un message $\boldsymbol{m} \in \mathbb{F}_2^k$**
   — On calcule $\boldsymbol{y} = \boldsymbol{m}\hat{\boldsymbol{G}}$.
   — On génère aléatoirement un vecteur erreur $\boldsymbol{e}$ de poids t.
   — Le chiffré de $\boldsymbol{m}$ est $\boldsymbol{c} = \boldsymbol{y} + \boldsymbol{e}$.
3. **Déchiffrement**
   — On calcule $\boldsymbol{c}' = \boldsymbol{c}\boldsymbol{P}^{-1}$.
   — On décode $\boldsymbol{c}'$ en un message $\boldsymbol{m}'$ en utilisant l'algorithme de décodage de $\mathcal{C}$.
   — On calcule $\boldsymbol{m} = \boldsymbol{m}'\boldsymbol{S}^{-1}$.

Dans l'article original, McEliece a proposé l'utilisation des codes de Goppa, qui sont aujourd'hui toujours utilisés. La sécurité repose sur deux problèmes : premièrement, un attaquant ne doit pas être capable de retrouver la matrice $\boldsymbol{G}$ à partir de la matrice $\hat{\boldsymbol{G}}$: autrement dit, un code de Goppa permuté doit être indistinguable d'un code aléatoire. Deuxièmement, un attaquant ne doit pas être capable de décoder efficacement un code aléatoire pour retrouver $\boldsymbol{m}$: ce problème du décodage de codes aléatoires a été prouvé

NP-complet pour la métrique de Hamming en 78 par Berlekamp, McEliece et Van Tilborg [21].

La principale faiblesse du cryptosystème proposé par McEliece est la taille des clés. Pour rendre les clés plus compactes tout en restant en métrique de Hamming, on peut utiliser des codes ayant une structure particulière, comme les codes quasi-cycliques [35], qui ont une représentation plus compacte. Cette technique est par exemple utilisée dans les cryptosystèmes basés sur les codes MDPC (Moderate Density Parity Check) [60, 1].

Une autre piste pour réduire la taille des clés est de changer de métrique. Au lieu d'utiliser la métrique de Hamming, on considère des codes en métrique rang : au lieu de considérer des codes à coefficients dans $\mathbb{F}_2$, on considère des codes à coefficients dans $\mathbb{F}_{q^m}$, avec des notions de poids et de distance différentes. Cette métrique est définie dans la section 2.3, et est utilisée par exemple dans les cryptosystèmes ROLLO [8] et RQC [3].

Contributions

Algorithmes de décodage en métrique rang

Le sécurité des cryptosystèmes basés sur codes correcteurs repose sur le problème de décodage de codes aléatoires. Pour estimer la sécurité des cryptosystèmes, il faut donc connaître précisément la complexité des algorithmes permettant de résoudre ce problème.

Dans [40], deux approches sont présentées :

— Une approche combinatoire, qui consiste à énumérer des espaces vectoriels jusqu'à en trouver un contenant le support de l'erreur recherchée.

— Une approche algébrique, qui propose une mise en équation du problème et différentes méthodes de résolution (par linéarisation ou par bases de Grobner).

Dans le chapitre 3, nous présentons une amélioration de l'algorithme permettant de résoudre le problème 2.3.6 de manière combinatoire. Cette approche permet d'exploiter la propriété de $\mathbb{F}_{q^m}$-linéarité que l'on retrouve dans les codes utilisés pour des applications cryptographiques.

Une analyse de la complexité de ce nouvel algorithme montre que dans le cas où m est plus petit que la longueur du code, ce qui est généralement le cas dans les applications cryptographiques, alors cette approche est plus efficace que l'algorithme combinatoire présenté dans [40]. Cette analyse est donc un outil essentiel à l'estimation de la sécurité des cryptosystèmes en métrique rang.

Le chapitre 4 présente une amélioration de l'algorithme de décodage des codes LRPC 2.3.12. Comparé à l'algorithme de décodage de [39], ce nouvel algorithme permet soit :

— De décoder des erreurs de plus grand poids : l'algorithme augmente la capacité de décodage des codes LRPC à $\frac{2}{3}(n-k)$ là où l'algorithme de décodage classique décode des erreurs de poids jusqu'à $\frac{n-k}{2}$.

— A poids d'erreur égal, de réduire la probabilité d'échec de décodage.

Une des faiblesses du schéma ROLLO [8] est la probabilité d'échec de déchiffrement non nulle. En effet, lors de l'opération de déchiffrement, on doit utiliser l'algorithme de décodage des codes LRPC pour décoder un mot de code bruité. Le fait de disposer de

n	m	r	d	DFR de [39]	DFR de l'algorithme amélioré
149	83	5	8	2^{-109}	2^{-128}
151	107	6	8	2^{-103}	2^{-128}
157	127	7	8	2^{-101}	2^{-132}

Table 1.1 – Exemples de paramètres de ROLLO avec les taux d'échec de déchiffrement des deux algorithmes de décodage des codes LRPC

plusieurs algorithmes de décodage permet donc de trouver de nouveaux compromis entre probabilité d'échec de décodage, taille des paramètres, et complexité de l'algorithme de décodage. Le taux d'échec de déchiffrement dépend de l'application souhaitée : dans [8], pour le cas de l'échange de secret avec des clés éphémères, il a été fixe à au plus 2^{-30}, alors que dans le cas du chiffrement, il a été fixé à au plus 2^{-128}. La table 1.1 montre un exemple de jeux de paramètres pour le chiffrement, soumis lors du second tour du processus de standardisation du NIST, avec les taux d'échec de déchiffrement pour les deux algorithmes de décodage.

Nouveaux schémas

Lors de l'appel à projets du NIST, un seul schéma de signature basée sur les codes en métrique rang a été proposé : Ranksign [43, 11]. Ce schéma de signature est un schéma de type "hacher et signer" : une fonction de hachage permet d'obtenir un syndrome qui sert de challenge, et la structure de la clé secrète (en l'occurence un code LRPC) permet de trouver un antécédent de ce syndrome. De la même manière que dans le cryptosystème de McEliece, la clé publique est donc une version masquée de la matrice LRPC. Pour que l'opération de signature puisse être effectuée en temps polynomial, une proportion non négligeable des syndromes doit posséder un antécédent, ce qui impose de fortes contraintes sur la longueur et le rang des codes LRPC. Ces contraintes ont été utilisées pour attaquer le schéma dans [25]. Suite à cette attaque il n'existait plus aucun schéma de signature en métrique rang.

Dans le chapitre 5 nous présentons un nouveau schéma de signature en métrique rang : Durandal. Contrairement à Ranksign, il ne s'agit pas d'un schéma de type "hacher et signer", mais d'un schéma à preuve de connaissance : il s'agit une adaptation du schéma de signature de Lyubashevsky [56] en métrique rang. Le principe est le suivant : on tire une matrice $\boldsymbol{H}$ aléatoire et une matrice $\boldsymbol{S}$ aléatoire dont les coordonnées appartiennent toutes à un espace vectoriel E de petite dimension r. La clé publique est le couple $(\boldsymbol{H}, \boldsymbol{T} = \boldsymbol{HS})$ et la clé secrète est la matrice $\boldsymbol{S}$. La signature est de la forme $(\boldsymbol{x} = \boldsymbol{Hy}^T, \boldsymbol{z} = \boldsymbol{y} + \boldsymbol{cS})$, où $\boldsymbol{c}$ est un challenge issu d'une fonction de hachage, et, pour vérifier la validité de la signature, on contrôle que $\boldsymbol{Hz}^T - \boldsymbol{Tc}^T = \boldsymbol{x}$. La difficulté pour adapter ce schéma, que ce soit en métrique de Hamming ou en métrique rang, est de faire en sorte que le terme $\boldsymbol{z}$ ne fasse pas fuir d'informations sur la matrice secrète $\boldsymbol{S}$. Dans Durandal, nous proposons d'ajouter un terme à $\boldsymbol{z}$ pour le randomiser. Afin de prouver l'indistinguabilité des signatures, on introduit un nouveau problème : le problème PSSI (Product Space Subspaces Indistinguishability), et on donne une analyse d'un distingueur contre ce problème.

Cela nous permet d'obtenir un schéma de signature efficace, avec une taille de clé publique de l'ordre de 20 kilooctets, et une taille de signature de l'ordre de 4 kilooctets.

Le chapitre 6 propose deux améliorations théoriques pour le schéma HQC (Hamming Quasi-Cyclic), un schéma de chiffrement en métrique de Hamming soumis à l'appel à projets du NIST.

Premièrement, nous proposons une nouvelle analyse de la distribution du poids des vecteurs erreurs utilisés dans HQC. En effet, HQC utilise des vecteurs qui sont des produits de vecteurs de poids donnés, et l'analyse du poids de Hamming du vecteur produit est une donnée essentielle à l'analyse de la probabilité d'échec de déchiffrement du cryptosystème. Nous proposons une estimation plus fine du poids de ces vecteurs que dans la soumission [5], qui permet d'obtenir une meilleure borne supérieure de la probabilité d'échec de déchiffrement et donc de proposer de meilleurs paramètres pour atteindre une probabilité d'échec fixée (typiquement $2^{-\lambda}$, où λ est le paramètre de sécurité).

Deuxièmement, nous proposons l'utilisation de codes de Reed-Muller et de Reed-Solomon concaténés à la place du code produit de codes à répétition et de codes BCH initialement proposé dans [5]. L'utilisation de cette nouvelle famille de codes permet d'obtenir une analyse précise du taux d'échec de déchiffrement, qui donne de meilleurs résultats que la famille de codes proposés à l'origine pour HQC. Nous donnons également des résultats de simulations permettant d'appuyer nos résultats théoriques.

Ces résultats ont été intégrés dans la soumission HQC, et permettent d'obtenir des tailles de clé 17% plus faibles que dans [5], à niveau de sécurité et taux d'échec de déchiffrement égaux.

Cryptanalyse

Le chapitre 7 décrit deux adaptations du schéma de signature de Lyubashevsky [56], respectivement en métrique rang et en métrique de Hamming, et les cryptanalyse. Dans les deux cas, la structure des signatures est exploitée pour retrouver la clé secrète à partir d'un très petit nombre de signatures (une seule en métrique rang et une dixaine en métrique de Hamming).

En métrique rang, on montre que l'adaptation directe du schéma de signature de Lyubashevsky ne peut même pas servir à réaliser une signature à usage unique, étant donné qu'on peut retrouver l'intégralité de la matrice secrète $\boldsymbol{S}$ à partir d'une seule signature, en utilisant l'algorithme de décodage des codes LRPC.

En métrique de Hamming, on ne peut utiliser directement l'adaptation de cette technique, qui consisterait à utiliser un algorithme de décodage de codes comme les codes LRPC ou MDPC, qui sont en quelque sorte l'équivalent des codes LRPC en métrique de Hamming, pour retrouver la clé secrète. En revanche on peut montrer que lorsqu'on obtient une signature, la probabilité qu'une coodonnée de la signature soit à 1 dépend du poids de la colonne associée dans la clé secrète. En compilant les informations obtenues à partir de plusieurs signatures et en utilisant un algorithme de type ISD (Information Set Decoding), on peut retrouver la clé secrète.

Ce chapitre montre la difficulté d'apapter le schéma de signature de Lyubashevsky en cryptographie basée sur les codes correcteurs d'erreurs, mais aussi la nécessité d'une analyse rigoureuse de l'indistinguabilité entre les signatures et des vecteurs aléatoires lors de la conception de tels schémas.

Implémentation de la métrique rang

Dans le cadre de l'appel à projets du NIST, il a fallu réaliser l'implémentation des différents schémas en métrique rang. La réalisation de ces implémentations a soulevé de nombreuses questions, que ce soit sur la représentation des données en mémoire ou sur les choix algorithmiques pour réaliser les différentes opérations associés aux objets mathématiques utilisés en métrique rang (corps finis et espaces vectoriels en particulier).

Avec l'avancement de la compétition, de nouvelles contraintes de performances, mais aussi de sécurité des implémentations se sont posées, rendant l'homogénéisation du code entre les différentes soumissions de plus en plus difficile. Afin de rendre les choses plus simples, la bibliothèque RBC (Rank Based Cryptography) a été développée [6].

Le chapitre 8 décrit les choix qui ont été réalisés pour développer la librairie RBC, en particulier quels algorithmes ont été implémentés pour manipuler les éléments de $\mathbb{F}_{q^m}$ et $\mathbb{F}_{q^m}^n$, ainsi que les sous-espaces vectoriels de $\mathbb{F}_{q^m}$. Nous présentons également le travail qui a été efféctué pour rendre certaines fonctions essentielles aux applications cryptographiques résistantes à certaines attaques par canaux cachés (notamment les attaques visant à observer le temps d'exécution d'une fonction). Ce chapire présente également des comparaisons des performances de la librairie RBC avec les principales librairies permettant de réaliser des calculs sur les corps finis : MPFQ, NTL et RELIC.

Chapitre 2

Codes correcteurs d'erreurs et cryptographie

2.1 Codes correcteurs d'erreurs

2.1.1 Définitions

Definition 2.1.1. *Code linéaire*
Un code linéaire $\mathcal{C}$ sur $\mathbb{F}$ de dimension k et de longueur n est un sous-espace vectoriel de $\mathbb{F}^n$ de dimension k. On dit que $\mathcal{C}$ est un code $[n,k]_{\mathbb{F}}$. Les éléments de $\mathcal{C}$ sont appelés des mots de code.

On peut représenter un tel code de deux manières équivalentes :

Definition 2.1.2. *Matrice génératrice*
Soit $\boldsymbol{G}$ une matrice $\in \mathbb{F}^{k\times n}$ et $\mathcal{C}$ un code $[n,k]_{\mathbb{F}}$. $\boldsymbol{G}$ est une matrice génératrice de $\mathcal{C}$ si les lignes de $\boldsymbol{G}$ forment une base $\mathcal{C}$, c'est-à-dire :

$$\mathcal{C} = \{\boldsymbol{mG}, \boldsymbol{m} \in \mathbb{F}^k\}.$$

Definition 2.1.3. *Matrice de parité*
Soit $\boldsymbol{H}$ une matrice $\in \mathbb{F}^{(n-k)\times n}$ et $\mathcal{C}$ un code $[n,k]_{\mathbb{F}}$. $\boldsymbol{H}$ est une matrice de parité de $\mathcal{C}$ si :

$$\mathcal{C} = \{\boldsymbol{c} \in \mathbb{F}^n, \boldsymbol{Hc}^\intercal = 0\}.$$

Remarque :
Ces définitions sont données pour n'importe quel corps fini $\mathbb{F}$. Pour la suite, nous allons avoir besoin des notions de poids d'un mot (le poids d'un mot $\boldsymbol{x} \in \mathbb{F}^n$ sera noté $\|\boldsymbol{x}\|$) et de distance entre deux mots (la distance entre deux mots $\boldsymbol{x}$ et $\boldsymbol{y}$ sera notée $d(\boldsymbol{x},\boldsymbol{y})$). On définit les notions de poids et de distance en utilisant la métrique de Hamming (cas $\mathbb{F} = \mathbb{F}_q$) et la métrique rang (cas $\mathbb{F} = \mathbb{F}_{q^m}$). Plus de détails sont donnés dans les sections 2.2 et 2.3.

Definition 2.1.4. *Métrique de Hamming*

Soit $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$. Le poids de Hamming de $\boldsymbol{x}$ est le nombre de coordonnées non-nulles de $\boldsymbol{x}$.

La distance de Hamming entre deux mots $\boldsymbol{x}$ et $\boldsymbol{y}$ est le poids de Hamming de $\boldsymbol{x} - \boldsymbol{y}$.

Note :

Dans ce document nous n'utiliserons la métrique de Hamming que dans le cas $\mathbb{F} = \mathbb{F}_2$.

Definition 2.1.5. *Métrique rang*

Soit $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ et soit $(\beta_1, \ldots, \beta_m) \in \mathbb{F}_{q^m}^n$ une base de $\mathbb{F}_{q^m}$ sur $\mathbb{F}_q$. Chaque coordonnée x_i peut être associée à un vecteur $(x_{i1}, \ldots, x_{im}) \in \mathbb{F}_q^m$ tel que :

$$x_i = \sum_{j=1}^{m} x_{ij}\beta_j.$$

On appelle $\boldsymbol{M}(\boldsymbol{x}) = (x_{ij})_{\substack{1\leqslant i\leqslant n \\ 1\leqslant j\leqslant m}}$ la matrice associée au vecteur $\boldsymbol{x}$.

Le poids rang $|\boldsymbol{x}|_r$ est défini de la manière suivante :

$$|\boldsymbol{x}|_r = \mathrm{Rank}\,\boldsymbol{M}(\boldsymbol{x}).$$

Cette définition ne dépend pas du choix de la base $(\beta_1, \ldots, \beta_m)$. La distance entre deux éléments $\boldsymbol{x}$ et $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$ est définie comme :

$$d(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x} - \boldsymbol{y}|_r.$$

On peut maintenant définir la distance minimale d'un code :

Definition 2.1.6. *Distance minimale*

Soit $\mathcal{C}$ un code $[n, k]_{\mathbb{F}}$. On appelle distance minimale de $\mathcal{C}$ la valeur d définie comme suit :

$$d = min(d(\boldsymbol{x}, \boldsymbol{y}), (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{C}^2, \boldsymbol{x} \neq \boldsymbol{y}).$$

On dit que $\mathcal{C}$ est un code $[n, k, d]_{\mathbb{F}}$.

2.2 Codes correcteurs d'erreurs en métrique de Hamming

Dans cette section nous allons définir les problèmes difficiles sur lesquels sont basés les cryptosystèmes en métrique de Hamming et présenter le schéma de chiffrement HQC [2].

2.2.1 Problèmes difficiles

Definition 2.2.1. *Syndrome Decoding (*SD*) problem*

Soit $\boldsymbol{H} \in \mathbb{F}_2^{(n-k)\times n}$ une matrice de parité d'un code $\mathcal{C}$, $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ un syndrome et w un entier positif. Le problème du décodage par syndrome, consiste à trouver, si il existe, $\boldsymbol{x} \in \mathbb{F}_2^n$ tel que :

$$\begin{aligned} \boldsymbol{H}\boldsymbol{x}^\intercal &= \boldsymbol{s} \\ \|\boldsymbol{x}\| &\leqslant w. \end{aligned} \tag{2.1}$$

2.2.2 Codes quasi-cycliques

Les codes quasi-cycliques ont été utilisés en cryptographie pour la première fois en 2005 par Gaborit [35]. Cette idée permet d'obtenir des représentations plus petites des matrices génératrices et de parité : leur utilisation en cryptographie permet donc de réduire la taille des clés.

L'idée est d'utiliser le fait qu'un vecteur $\boldsymbol{x} = (x_0, \dots, x_{n-1}) \in \mathbb{F}_2^n$ peut être associé à un polynôme $\boldsymbol{x}(X)$ dans $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$: $\boldsymbol{x}(X) = \sum_{i=0}^{n-1} x_i X^i$.

On définit le produit des vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$ comme le produit des polynômes $\boldsymbol{x}(X)$ et $\boldsymbol{y}(X)$ dans $\mathcal{R}$. Ce produit peut être associé à un produit vecteur/matrice :

$$\begin{aligned} \boldsymbol{x}\boldsymbol{y} &= \boldsymbol{x}(X)\boldsymbol{y}(X) \mod (X^n-1) \\ &= \sum_{i=0}^{n-1} x_i X^i \boldsymbol{y}(X) \mod (X^n-1) \\ &= \sum_{i=0}^{n-1} x_i (X^i \boldsymbol{y}(X) \mod (X^n-1)) \\ &= (x_0, \dots, x_{n-1}) \begin{pmatrix} \boldsymbol{y} \\ X\boldsymbol{y} \mod (X^n-1) \\ X^2\boldsymbol{y} \mod (X^n-1) \\ \vdots \\ X^{(n-1)}\boldsymbol{y} \mod (X^n-1) \end{pmatrix}. \end{aligned}$$

On appelle cette matrice une matrice circulante :

Definition 2.2.2. *Matrice circulante*

Soit $\boldsymbol{x} = (x_0, \dots, x_{n-1}) \in \mathcal{R}$. *On peut associer au polynôme* $\boldsymbol{x}$ *la matrice circulante suivante :*

$$\mathcal{M}(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{x} \\ X\boldsymbol{x} \mod (X^n-1) \\ X^2\boldsymbol{x} \mod (X^n-1) \\ \vdots \\ X^{(n-1)}\boldsymbol{x} \mod (X^n-1) \end{pmatrix} \in \mathbb{F}_2^{n\times n}.$$

On peut alors écrire le produit $\boldsymbol{x}\boldsymbol{y}$ comme le produit vecteur/matrice :

$$\boldsymbol{x}\boldsymbol{y} = \boldsymbol{x}\mathcal{M}(\boldsymbol{y}) = (\mathcal{M}(\boldsymbol{x})^\intercal \boldsymbol{y}^\intercal)^\intercal = \boldsymbol{y}\boldsymbol{x}$$

Remarque : En écrivant explicitement les coordonnées de $\mathcal{M}(\boldsymbol{x})$ on obtient la matrice suivante :

$$\mathcal{M}(\boldsymbol{x}) = \begin{pmatrix} x_0 & x_1 & \dots & x_{n-1} \\ x_{n-1} & x_0 & \dots & x_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & \dots & x_0 \end{pmatrix}.$$

D'où le nom de matrice circulante.

On peut maintenant définir la notion de code quasi-cyclique :

Definition 2.2.3. *Code quasi-cyclique*

On considère un vecteur $\boldsymbol{c} \in \mathbb{F}_2^{sn}$ que l'on voit comme un ensemble de s vecteurs de longueur n : $\boldsymbol{c} = (\boldsymbol{c}_0, \dots, \boldsymbol{c}_{s-1})$ avec $\boldsymbol{c}_i \in \mathbb{F}_2^n$. Chacun des $\boldsymbol{c}_i$ est vu comme un polynôme dans $\mathcal{R} = \mathbb{F}_2[X]/(X^n - 1)$.

Un code $\mathcal{C}$ $[sn, k]$ est quasi-cyclique d'ordre s si pour tout $\boldsymbol{c} = (\boldsymbol{c}_0, \dots, \boldsymbol{c}_{s-1}) \in \mathcal{C}$ on a $(X\boldsymbol{c}_0, \dots, X\boldsymbol{c}_{s-1}) \in \mathcal{C}$.

Dans la suite de ce document nous allons principalement considérer un cas particulier des codes quasi-cycliques, les codes doublement circulants :

Definition 2.2.4. *Code doublement circulant*

Un code $\mathcal{C}$ doublement circulant est un code quasi-cyclique d'ordre 2.

Un tel code peut être représenté par une matrice génératrice $\boldsymbol{G} \in \mathbb{F}_2^{n \times 2n}$ de la forme $(\boldsymbol{A}_0\boldsymbol{A}_1)$ où les $\boldsymbol{A}_i$ sont des matrices circulantes de taille $n \times n$.

Si la matrice $\boldsymbol{A}_0$ est inversible, alors $\mathcal{C}$ admet une représentation sous forme systématique de la forme $(\boldsymbol{I}\boldsymbol{A})$.

L'intérêt de ces codes pour la cryptographie est leur représentation très compacte : en effet, leur matrice de parité sous forme systématique peut être représentée par n symboles de $\mathbb{F}_2$, contre n^2 pour un code non quasi cyclique.

2.3 Codes correcteurs d'erreurs en métrique rang

Dans cette section nous allons considérer des codes sur $\mathbb{F}_{q^m}$ munis de la métrique rang, définie en 2.1.5.

On aura besoin tout au long de cette section de la notion de support :

Definition 2.3.1. *Support en métrique rang*

Soit $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_{q^m}^n$. On appelle support de $\boldsymbol{x}$ le sous-espace vectoriel de $\mathbb{F}_{q^m}$ engendré par les coordonnées de $\boldsymbol{x}$:

$$\operatorname{Supp} \boldsymbol{x} = \langle x_1, \dots, x_n \rangle.$$

Par définition, on a $\dim(\operatorname{Supp} \boldsymbol{x}) = |\boldsymbol{x}|_r$.

Le nombre de supports de dimension r en métrique rang est donné par le coefficient binomial de Gauss $\begin{bmatrix} m \\ r \end{bmatrix}$. On a (voir [48]) :

$$\begin{bmatrix} m \\ r \end{bmatrix} = \prod_{i=0}^{r-1} \frac{q^m - q^i}{q^r - q^i}$$

On utilisera souvent l'approximation $\begin{bmatrix} m \\ r \end{bmatrix} \approx q^{r(m-r)}$.

2.3.1 Codes en métrique rang

Il existe deux types de codes en métrique rang : les codes matriciels (ou $\mathbb{F}_q$-linéaires) et les codes $\mathbb{F}_{q^m}$-linéaires. Les codes $\mathbb{F}_{q^m}$-linéaires sont les plus utilisés en cryptographie et sont un cas particulier des codes matriciels.

Definition 2.3.2. *Codes $\mathbb{F}_{q^m}$-linéaires*
Un code $\mathbb{F}_{q^m}$-linéaire $\mathcal{C}$ de longueur n et de dimension k est un sous-espace vectoriel de $\mathbb{F}_{q^m}^n$ de dimension k. On dit que $\mathcal{C}$ est un code $[n,k]_{\mathbb{F}_{q^m}}$.

Definition 2.3.3. *Codes matriciels*
Soit $\mathcal{M}_{\mathbb{F}_q}^{m\times n}$ l'ensemble des matrices de taille $m \times n$ à coefficients dans $\mathbb{F}_q$. Un code matriciel $\mathcal{C}$ de longueur $m \times n$ et de dimension K est un sous-espace vectoriel de $\mathcal{M}_{\mathbb{F}_q}^{m\times n}$ de dimension K. On dit que $\mathcal{C}$ est un code $[m \times n, K]_{\mathbb{F}_q}$.

Les codes $\mathbb{F}_{q^m}$-linéaires sont un sous-ensemble des codes matriciels. En effet, à un mot de code $\boldsymbol{c} = (c_1, \dots, c_n) \in \mathbb{F}_{q^m}^n$ d'un code $\mathbb{F}_{q^m}$-linéaire $\mathcal{C}$, on peut associer une matrice $\boldsymbol{M} \in \mathcal{M}_{\mathbb{F}_q}^{m\times n}$ en écrivant chacune des coordonnées dans une base de $\mathbb{F}_{q^m}$ sur $\mathbb{F}_q$, de la même manière que dans la définition 2.1.5. En répétant cette opération pour chacun des mots du code $\mathcal{C}$, on obtient le code matriciel $\mathcal{C}'$, de longueur nm et de dimension km, associé à $\mathcal{C}$.

On peut représenter ces codes de deux manières : soit par une matrice génératrice $\boldsymbol{G}$, soit par une matrice de parité $\boldsymbol{H}$. On commence par le cas des codes $\mathbb{F}_{q^m}$-linéaires : en supposant que le code admette une matrice génératrice sous forme systématique, on a :

$$\boldsymbol{G} = (\boldsymbol{I}_k | A), \boldsymbol{H} = (\boldsymbol{B} | \boldsymbol{I}_{n-k}),$$

où $\boldsymbol{A}$ et $\boldsymbol{B}$ sont des matrices aléatoires à coefficients dans $\mathbb{F}_{q^m}$. Quelle que soit la représentation choisie, il faut $k(n-k)m \lceil log(q) \rceil$ bits pour représenter le code $\mathcal{C}$.

On s'intéresse maintenant à la représentation du code matriciel $\mathcal{C}'$ associé à $\mathcal{C}$. On peut représenter $\mathcal{C}'$ par une matrice génératrice de taille $nm \times km \in \mathbb{F}_q$. En supposant que le code admette une matrice génératrice sous forme systématique, il faut $km(n - k)m \lceil log(q) \rceil = k(n-k)m^2 \lceil log(q) \rceil$ bits pour représenter $\mathcal{C}'$.

On remarque que la représentation d'un code $\mathbb{F}_{q^m}$-linéaire est m fois plus compacte que la représentation du code matriciel associé. Cela explique l'intérêt des codes $\mathbb{F}_{q^m}$-linéaires en cryptographie : leur représentation compacte permet de réduire la taille des clés.

Dans la suite de ce document, les codes en métrique rang que l'on va considérer seront des codes $\mathbb{F}_{q^m}$-linéaires, sauf mention contraire.

2.3.2 Codes idéaux

Les codes idéaux sont une généralisation de la notion de codes quasi-cycliques. La différence est qu'au lieu d'utiliser le polynôme $(X^n - 1)$, on choisit un polynôme $P \in \mathbb{F}_q[X]$ de degré n et on définit $\mathcal{R} = \mathbb{F}_{q^m}[X]/\langle P \rangle$. De la même manière que pour les codes quasi-cycliques, on définit les notions de matrice idéale et de codes idéaux :

Definition 2.3.4. *Matrice idéale*

Soit $\boldsymbol{x} = (x_0, \ldots, x_{n-1}) \in \mathcal{R}$. On peut associer au polynôme $\boldsymbol{x}$ la matrice idéale suivante :

$$\mathcal{M}(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{x} \\ X\boldsymbol{x} \mod P \\ X^2\boldsymbol{x} \mod P \\ \vdots \\ X^{(n-1)}\boldsymbol{x} \mod P \end{pmatrix} \in \mathbb{F}_2^{n \times n}.$$

Definition 2.3.5. *Codes idéaux*

On considère un vecteur $\boldsymbol{c} \in \mathbb{F}_{q^m}^{sn}$ que l'on voit comme un ensemble de s vecteurs de longueur n : $\boldsymbol{c} = (\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{s-1})$ avec $\boldsymbol{c}_i \in \mathbb{F}_{q^m}^n$. Chacun des $\boldsymbol{c}_i$ est vu comme un polynôme dans $\mathcal{R} = \mathbb{F}_{q^m}[X]/\langle P \rangle$.

Un code $\mathcal{C}$ $[sn, k]$ est idéal d'ordre s si pour tout $\boldsymbol{c} = (\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{s-1}) \in \mathcal{C}$ on a :

$$(X\boldsymbol{c}_0, \ldots, X\boldsymbol{c}_{s-1}) \in \mathcal{C}$$

De la même manière qu'en métrique de Hamming, la matrice génératrice d'un code idéal d'ordre 2 $[2n, n]$ admettant une forme systématique peut être représentée par n symboles de $\mathbb{F}_{q^m}$.

2.3.3 Problèmes difficiles

De la même manière qu'en métrique de Hamming, les deux principaux problèmes difficiles en métrique rang sont le problème de décodage par syndrome et le problème de recherche de mot de petit poids :

Definition 2.3.6. *Problème du décodage par syndrome (ou Rank Syndrome Decoding,* RSD*)*

*Soit $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ une matrice de parité d'un code $\mathcal{C}$, $\boldsymbol{s} \in \mathbb{F}_{q^m}^{n-k}$ un syndrome et w un entier positif. Le problème du décodage par syndrome en métrique rang, ou rank syndrome decoding (*RSD*) consiste à trouver, s'il existe, $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ tel que :*

$$\begin{aligned} \boldsymbol{H}\boldsymbol{x}^\intercal &= \boldsymbol{s} \\ |\boldsymbol{x}|_r &\leqslant w \end{aligned} \tag{2.2}$$

Definition 2.3.7. *Problème de recherche de mot de petit poids*

Soit $\mathcal{C}$ un code $[n, k]$ et w un entier. Le problème consiste à trouver un mot de code $\boldsymbol{c} \in \mathcal{C}$ tel que $|\boldsymbol{c}|_r = w$.

Remarque : Si $\boldsymbol{H}$ est une matrice de parité du code $\mathcal{C}$, le problème de recherche de mot de petit poids est une instance du problème de décodage par syndrome avec $\boldsymbol{s} = \boldsymbol{0}$.

Nous allons également montrer que trouver un vecteur erreur $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ solution d'une instance de RSD est équivalent à simplement retrouver le support de $\boldsymbol{x}$.

Definition 2.3.8. *Problème de recherche de support (ou Rank Support Recovery,* RSR*)*

*Soit $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ une matrice de parité d'un code $\mathcal{C}$, $\boldsymbol{s} \in \mathbb{F}_{q^m}^{n-k}$ un syndrome et w un entier positif. Le problème de recherche de support, ou rank support recovery (*RSR*) consiste à trouver un sous-espace vectoriel E de $\mathbb{F}_{q^m}$ de dimension au plus w tel qu'il existe un $\boldsymbol{x} \in E^n$ tel que $\boldsymbol{H}\boldsymbol{x}^\intercal = \boldsymbol{s}$.*

Equivalence entre les problèmes RSD **et** RSR On peut trivialement réduire le problème RSR au problème RSD : en effet, pour retrouver le support E dans une instance de RSR, il suffit de calculer le support d'une solution $\boldsymbol{x}$ de l'instance de RSD associée.

Inversement, on peut également réduire le problème RSD au problème RSR. Supposons que l'on connaisse un support E solution d'une instance de RSD avec un poids w. On veut trouver un vecteur $\boldsymbol{x} = (x_1, \ldots, x_n) \in E^n$ tel que $\boldsymbol{H}\boldsymbol{x}^\intercal = \boldsymbol{s}$.

Soit $(E_1, \ldots, E_w)$ une base de E. On peut écrire les coordonnées de $\boldsymbol{x}$ dans cette base :

$$\forall 1 \leqslant i \leqslant n, x_i = \sum_{j=1}^{w} \lambda_{ij} E_j, \lambda_{ij} \in \mathbb{F}_q.$$

On peut alors réécrire l'équation $\boldsymbol{H}\boldsymbol{x}^\intercal = \boldsymbol{s}$ en fonction des nouvelles inconnues λ_{ij}. On obtient un système de nw inconnues dans $\mathbb{F}_q$ et n équations dans $\mathbb{F}_{q^m}$, soit nm équations dans $\mathbb{F}_q$.

On peut alors trouver une solution à l'instance de RSD en résolvant ce système.

Afin d'écrire des preuves de sécurité pour les cryptosystèmes basés sur les codes correcteurs d'erreurs en métrique rang, on doit introduire des variations sur le problème du décodage en métrique rang, notamment sa version décisionnelle :

Problem 2.3.9. *Version décisionnelle du problème de décodage par syndrome*

Etant donnés une matrice de parité $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ et un vecteur $\boldsymbol{y} \in \mathbb{F}_{q^m}^{(n-k)}$, le problème est de distinguer la paire $(\boldsymbol{H}, \boldsymbol{y} = \boldsymbol{H}\boldsymbol{e})$ où $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$ est un vecteur erreur de poids w d'une paire $(\boldsymbol{H}, \boldsymbol{y})$ tirée aléatoirement dans $\mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^n$.

On introduit également des variations du problème de décodage et de recherche du support pour les codes idéaux 2.3.5 : ce problème sera utile en cryptographie, étant donné que ce type de codes est utilisé pour réduire la taille des clés.

Problem 2.3.10. *Ideal-Rank Syndrome Decoding*

Etant donnés un vecteur $\boldsymbol{h} \in \mathbb{F}_{q^m}^n$, un polynome $P \in \mathbb{F}_q[X]$ de degré n, un syndrome $\boldsymbol{\sigma}$ et un entier w, le problème est de trouver un vecteur $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{F}_{q^m}^{2n}$ de poids $\leqslant w$ tel que $\boldsymbol{x}_1 + \boldsymbol{x}_2\boldsymbol{h} = \boldsymbol{\sigma} \mod P$.

Comme $\boldsymbol{h}$ et P définissent un code idéal $[2n, n]_{q^m}$ sous forme systématique, le problème $\mathrm{I-RSD}$ est un cas particulier du problème RSD.

Problem 2.3.11. *Ideal-Rank Support Recovery*

Etant donnés un vecteur $\boldsymbol{h} \in \mathbb{F}_{q^m}^n$, un polynome $P \in \mathbb{F}_q[X]$ de degré n, un syndrome $\boldsymbol{\sigma}$ et un poids w, le problème est de retrouver le support E de dimension $\leqslant w$ tel qu'il existe $\boldsymbol{e}_1 + \boldsymbol{e}_2\boldsymbol{h} = \boldsymbol{\sigma} \mod P$ où les vecteurs $\boldsymbol{e}_1$ et $\boldsymbol{e}_2$ ont pour support E.

1. **Calcul de l'espace S**
   On calcule $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{y}^\intercal$ et $S = \langle s_1, \dots, s_{n-k} \rangle$.
2. **Récupération du support de l'erreur E**
   Soit $S_i = F_i^{-1}S$. On calcule $E = S_1 \cap \cdots \cap S_d$ et une base $\{E_1, \dots, E_r\}$ de E.
3. **Calcul de $\boldsymbol{e}$**
   En écrivant $e_i = \sum_{i=1}^{n} e_{ij}E_j$ on peut résoudre le système $\boldsymbol{H}\boldsymbol{e}^\intercal = \boldsymbol{s}$ où les équations $\boldsymbol{H}\boldsymbol{e}^\intercal$ et les coordonnées du syndrome sont écrites dans une base de l'espace produit EF. Le système a nr inconnues et $(n-k)rd$ équations dans $\mathbb{F}_q$.
4. **Calcul de $\boldsymbol{m}$**
   On résout le système $\boldsymbol{m}\boldsymbol{G} = \boldsymbol{y} - \boldsymbol{e}$.

FIGURE 2.1 – Algorithme de décodage des codes LRPC

2.3.4 Codes LRPC

Définition

Les codes LRPC (Low Rank Parity Check), introduits dans [39], sont une famille de codes en métrique rang tels que l'ensemble des coordonnées de d'une de leurs matrices de parité $\boldsymbol{H}$ appartient à un sous-espace F de $\mathbb{F}_{q^m}$ de petite dimension.

Definition 2.3.12. *Code LRPC*
Un code LRPC $\mathcal{C}$ de rang d, longueur n et dimension k sur $\mathbb{F}_{q^m}$ est un code qui admet une matrice de parité $\boldsymbol{H} \in \mathbb{F}_{q^m}^{n\times(n-k)}$ telle que l'espace vectoriel engendré par ses coefficients h_{ij} est de dimension d. On note $F = \langle h_{ij} \rangle$ le sous-espace de $\mathbb{F}_{q^m}$ engendré par les coordonnées de $\boldsymbol{H}$.

Décodage

L'algorithme de décodage des codes LRPC utilise la connaissance de l'espace F de petite dimension afin de retrouver le support de l'erreur E à partir du syndrome. L'idée générale est la suivante : à partir du syndrome $\boldsymbol{s} = (s_1, \dots, s_{n-k})$, on peut calculer $S = \langle s_1, \dots, s_{n-k} \rangle$ qui est un sous-espace de l'espace produit EF. Si on a l'égalité $S = EF$, alors on peut en déduire le support de l'erreur E et il ne reste plus qu'à résoudre un système d'équations linéaires pour retrouver le vecteur erreur $\boldsymbol{e}$. L'ensemble de ces étapes est détaillé ci-dessous.

Algorithme de décodage

Soit $\mathcal{C}$ un code LRPC de longueur n et de dimension k de rang d, de matrice génératrice $\boldsymbol{G}$ et de matrice de parité $\boldsymbol{H}$. Soit F le sous-espace de $\mathbb{F}_{q^m}$ engendré par les coordonnées h_{ij} de $\boldsymbol{H}$ et soit $\{F_1, \dots, F_d\}$ une base de F.

Soit $\boldsymbol{y} = \boldsymbol{m}\boldsymbol{G} + \boldsymbol{e}$ le mot reçu, où $\boldsymbol{m}$ est le message et $\boldsymbol{e}$ est une erreur de rang r.

L'algorithme de décodage est présenté figure 2.1.

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

Table 2.1 – Tailles en octets pour HQC

Les détails concernant la probabilité d'échec de l'algorithme et sa complexité sont disponibles dans [39].

2.4 Cryptographie basée sur les codes correcteurs d'erreurs

2.4.1 Cryptographie en métrique de Hamming

Hamming Quasi-Cyclic (HQC) On présente le cryptosystème HQC, qui utilise la structure des codes doublement circulant. Ce schéma permet d'obtenir de petites tailles de clés (table 2.1) ainsi que des temps d'exécution rapides (table 2.2) tout en ayant une réduction de sécurité au problème du décodage de codes aléatoires : en effet le code structuré $\mathcal{C}$ permettant de le décodage est une donnée publique, contrairement au cryptosystème de McEliece. Les détails sur le fonctionnement du schéma et la preuve de sécurité peuvent être trouvés dans [2]. Le chapitre 6 présente une analyse de la probabilité d'échec de décodage et propose l'utilise des codes de Reed-Muller et de Reed-Solomon concaténés comme code $\mathcal{C}$.

1. **Données publiques**
   — Un code $[n, k]$ $\mathcal{C}$ de matrice de génératrice $\boldsymbol{G}$ pouvant corriger jusqu'à σ erreurs.
2. **Génération de clés**
   — On tire aléatoirement $\boldsymbol{h} \in \mathbb{F}_2^n$ et $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ tels que $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = w$.
   — $(\boldsymbol{x}, \boldsymbol{y})$ est la clé secrète et $(\boldsymbol{h}, \boldsymbol{s} = \boldsymbol{x} + \boldsymbol{h}\boldsymbol{y})$ est la clé publique.
3. **Chiffrement d'un message $\boldsymbol{m} \in \mathbb{F}_2^k$**
   — On tire aléatoirement $\boldsymbol{e} \in \mathbb{F}_2^n$ et $(\boldsymbol{r}_1, \boldsymbol{r}_2) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ tels que $\|\boldsymbol{e}\| = w_e$ et $\|\boldsymbol{r}_1\| = \|\boldsymbol{r}_2\| = w_r$.
   — On calcule $\boldsymbol{u} = \boldsymbol{r}_1 + \boldsymbol{h}\boldsymbol{r}_2$ et $\boldsymbol{v} = \boldsymbol{m}\boldsymbol{G} + \boldsymbol{s}\boldsymbol{r}_2 + \boldsymbol{e}$.
   — Le chiffré de $\boldsymbol{m}$ est le couple $(\boldsymbol{u}, \boldsymbol{v})$.
4. **Déchiffrement**
   — On décode $\boldsymbol{v} - \boldsymbol{u}\boldsymbol{y}$ en $\boldsymbol{m}$ en utilisant l'algorithme de décodage de $\mathcal{C}$.

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

TABLE 2.2 – Performances en kilocycles pour HQC

Paramètre	Clé publique	Clé secrète	Chiffré	Echec de déchiffrement
ROLLO-I-128	696	40	696	2^{-28}
ROLLO-I-192	958	40	958	2^{-34}
ROLLO-I-256	1371	40	1371	2^{-33}
ROLLO-II-128	1941	40	2089	2^{-134}
ROLLO-II-192	2341	40	2469	2^{-130}
ROLLO-II-256	2559	40	2687	2^{-136}

TABLE 2.3 – Tailles en octets pour ROLLO

2.4.2 Cryptographie en métrique rang

Schémas de chiffrement

Le premier cryptosystème basé sur les codes correcteurs d'erreurs en métrique rang a été proposé en 1991 par Gabidulin, Paramonov et Tretjakov [34]. Il s'agit d'une instanciation du cryptosystème de McEliece en utilisant les codes de Gabidulin [33]. Ce cryptosystème a été attaqué en raison de la forte structure algébrique des codes de Gabidulin [63].

Les codes LRPC 2.3.12 ont ensuite été utilisés pour instancier le cryptosystème de McEliece [39]. La sécurité de ce cryptosystème repose sur deux problèmes : d'une part, le problème de décodage en métrique rang 2.3.6 et d'autre part, le problème d'indistingabilité des codes LRPC. L'utilisation de la métrique rang permet d'obtenir des tailles de clés et des temps d'exécution très compétitifs, comme le montrent les tables 2.3 et 2.4 qui montrent les performances de ROLLO [8], un cryptosystème soumis à l'appel à projets du NIST. Etant donné que l'algorithme de décodage des codes LRPC a une probabilité d'échec non nulle, ROLLO propose deux jeux de paramètres : ROLLO-I pour l'échange de clés avec une probabilité d'échec de déchiffrement de l'ordre de 2^{-30} et ROLLO-II pour le chiffrement avec une probabilité d'échec de déchiffrement de l'ordre de 2^{-128}.

RQC (Rank Quasi-Cyclic) [4] est une alternative à ROLLO, également soumise lors de l'appel à projets du NIST. Ce cryptosystème a la même structure que HQC présenté dans la section 2.4.1. Les tailles de clés et les performances sont moins compétitives que celles de ROLLO, comme le montrent les tables 2.5 et 2.6 mais RQC dispose de deux avantages : d'une part, l'algorithme de décodage utilisé lors du déchiffrement est celui des codes de Gabidulin au lieu des codes LRPC, ce qui permet d'obtenir un cryptosystème sans échec de déchiffrement. D'autre part, de la même manière que dans HQC, la sécurité ne repose que sur une variante du problème du décodage de syndrome 2.3.6 pour les codes

Paramètre	Génération de clés	Chiffrement	Déchiffrement
ROLLO-I-128	939	113	686
ROLLO-I-192	1142	127	803
ROLLO-I-256	1582	151	1347
ROLLO-II-128	3690	331	1195
ROLLO-II-192	3689	334	1258
ROLLO-II-256	4296	361	1604

Table 2.4 – Performances en kilocycles pour ROLLO

Paramètre	Clé publique	Clé secrète	Chiffré
RQC-128	1834	40	3652
RQC-192	2583	40	5690
RQC-256	4090	40	8164

Table 2.5 – Tailles en octets pour RQC

quasi-cycliques.

Paramètre	Génération de clés	Chiffrement	Déchiffrement
RQC-128	370	530	2580
RQC-192	760	1160	5650
RQC-256	1150	1710	9350

Table 2.6 – Performances en kilocycles pour RQC

Part II

Nouveaux algorithmes de décodage en métrique rang

Chapter 3

Décodage générique en métrique rang

This chapter introduces an improvement of the GRS algorithm [40] that is used to solve the RSD problem in a combinatorial way. Since the security of the rank-based cryptosystems directly relies on the hardness of decoding random $\mathbb{F}_{q^m}$-linear codes, it is very important to evaluate the complexity of the generic decoding algorithms. One of the first algorithm [62] has a complexity of $\mathcal{O}\left((wm)^3 q^{(w-1)(k+1)+2}\right)$ for an error of weight w in a code of length n and of dimension k over the field $\mathbb{F}_{q^m}$. As we can see, this complexity does not take into account the length of the code. The GRS algorithm [40] gives a complexity of $\mathcal{O}\left((n-k)^3 m^3 q^{(w-1)\left\lceil \frac{(k+1)m}{n} \right\rceil}\right)$ in the case $n \leqslant m$ and this article introduces another algorithm based on the q-polynomials of complexity $\mathcal{O}\left((wm)^3 q^{r\left\lceil w\frac{(k+1)(w+1)-n-1}{w}\right\rceil}\right)$. This work improves the complexity of the GRS algorithm and we obtain a complexity of $\mathcal{O}\left((n-k)^3 m^3 q^{w\left\lceil \frac{(k+1)m}{n} \right\rceil - m}\right)$. This work is a joint work with Philippe Gaborit, Adrien Hauteville and Jean-Pierre Tillich and was published in ISIT 2018 [15].

Contents

3.1 Description of the Attacks on the RSD Problem

Our algorithm is an improvement of the GRS algorithm [40]. We first recall the description of this algorithm in the first subsection then we present our improvement in the second one.

3.1.1 The GRS algorithm

The general idea to solve the RSD problem is to guess a subspace F such that $\text{Supp}(\boldsymbol{e}) \subset F$. Then we can express the coordinates of $\boldsymbol{x}$ in a basis of F and solve the linear system given by the parity-check equations. There are two possible cases we will describe more precisely.

— **first case: $n \geqslant m$.**
Let F be a subspace of $\mathbb{F}_{q^m}$ of dimension r and $(F_1, \dots, F_r)$ a basis of F. Let us assume that $\mathrm{Supp}(\boldsymbol{e}) \subset F$.

$$\Rightarrow \forall i \in [1..n], e_i = \sum_{j=1}^{r} \lambda_{ij} F_j.$$

This gives us nr unknowns over $\mathbb{F}_q$.
We can now rewrite the parity-check equations in these unknowns:

$$\boldsymbol{H}\boldsymbol{e}^T = \boldsymbol{s} \tag{3.1}$$

$$\Leftrightarrow \left\{ \begin{array}{ccccc} H_{11}e_1 & + \cdots + & H_{1n}e_n & = & s_1 \\ \vdots & & \vdots & & \vdots \\ H_{n-k,1}e_1 & + \cdots + & H_{n-k,n}e_n & = & s_{n-k} \end{array} \right.$$

$$\Leftrightarrow \left\{ \begin{array}{cccc} \sum_{l=1}^{n} H_{1l} & \sum_{j=1}^{r} \lambda_{lj} F_j & = & s_1 \\ \vdots & & & \vdots \\ \sum_{l=1}^{n} H_{n-k,l} & \sum_{j=1}^{r} \lambda_{lj} F_j & = & s_{n-k}. \end{array} \right. \tag{3.2}$$

Now, we need embed these $n-k$ equations over $\mathbb{F}_{q^m}$ into $(n-k)m$ equations over $\mathbb{F}_q$.
Let φ_i the i^{th} projection with respect to the basis $(\beta_1, \dots, \beta_m)$ from $\mathbb{F}_{q^m}$ on $\mathbb{F}_q$:

$$\begin{array}{cccc} \varphi_i : & \mathbb{F}_{q^m} \to & \mathbb{F}_q \\ & \sum_{i=1}^{m} x_i \beta_i \mapsto & x_i. \end{array}$$

We have:

$$\begin{array}{c} \boldsymbol{H}\boldsymbol{e}^T = \boldsymbol{s} \\ \Leftrightarrow \forall i \in [1..m], \end{array}$$

$$\left\{ \begin{array}{ccc} \sum_{l=1}^{n} \sum_{j=1}^{r} \lambda_{1j} \varphi_i(H_{1l}F_j) & = & \varphi_i(s_1) \\ \vdots & & \vdots \\ \sum_{l=1}^{n} \sum_{j=1}^{r} \lambda_{1j} \varphi_i(H_{n-k,l}F_j) & = & \varphi_i(s_{n-k}). \end{array} \right. \tag{3.3}$$

Since we assume $\mathrm{Supp}(\boldsymbol{e}) \subset F$, this system has at least one solution. We want more equations than unknowns in order to have only one solution with overwhelming probability. So we have the condition:

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

If the system has a solution, we check if its rank is equal to w. Otherwise, we choose another F and restart at the beginning.
The average complexity of the algorithm is $\mathcal{O}\left(\frac{(n-k)^3 m^3}{p}\right)$ where p is the probability that $\mathrm{Supp}(\boldsymbol{e}) \subset F$.

p is equal to the number of subspaces of dimension w in a subspace of dimension r divided by the number of subspaces of dimension w in $\mathbb{F}_{q^m}$.

$$p = \frac{\begin{bmatrix} r \\ w \end{bmatrix}}{\begin{bmatrix} m \\ w \end{bmatrix}} \approx q^{-w(m-r)}.$$

By taking $r = m - \left\lceil \frac{km}{n} \right\rceil$ we obtain a complexity of $\mathcal{O}\left((n-k)^3 m^3 q^{w\left\lceil \frac{km}{n} \right\rceil}\right)$

— **second case:** $m > n$. In this case we consider the matrix code associated to the $\mathbb{F}_{q^m}$-linear code. Instead of searching for a subspace which contains the columns support of the matrix of the error, we search for a subspace F of $\mathbb{F}_q^n$ which contains the rows support of the error. The rest of the algorithm is the same, the only differences are:

  — the number of unknowns is mr, which implies $r \leqslant n - k$.

  — the probability to choose F correctly is $\frac{\begin{bmatrix} r \\ w \end{bmatrix}}{\begin{bmatrix} n \\ w \end{bmatrix}} \approx q^{-w(n-r)}$.

  Thus, the complexity in this case is $\mathcal{O}\left((n-k)^3 m^3 q^{wk}\right)$.

This algorithm does not use the $\mathbb{F}_{q^m}$-linearity of the code. To exploit this structure, the main idea is to consider the code $\mathcal{C}' = \mathcal{C} + \mathbb{F}_{q^m}\boldsymbol{e}$, where $\mathcal{C}$ is the code with parity-check matrix $\boldsymbol{H}$.

The problem is reduced to the search for a codeword of weight w in $\mathcal{C}'$. If $\boldsymbol{e}$ is the only solution of the system 2.2, then the only codewords of $\mathcal{C}'$ of weight w are of the form $\alpha\boldsymbol{e}, \alpha \in \mathbb{F}_{q^m}^*$. These codewords are solutions of the system

$$\begin{cases} \boldsymbol{H}'\boldsymbol{e}'^T = \boldsymbol{0} \\ |\boldsymbol{e}'|_r = w. \end{cases} \tag{3.4}$$

where $\boldsymbol{H}'$ is a parity-check matrix of $\mathcal{C}'$.

Thus, instead of looking for the support E of $\boldsymbol{e}$, we can look for any multiple αE of the support. In [40], the authors specialize one element of the support by searching for small a weight codeword $\boldsymbol{c}$ such that $1 \in \mathrm{Supp}(\boldsymbol{c}) = E$ (but one can choose any non zero element of $\mathbb{F}_{q^m}$). One can use this information to improve the probability that $F \supset \mathrm{Supp}(\boldsymbol{c})$ by choosing F such that $1 \in F$. Let φ be the projection from the $\mathbb{F}_q$-subspaces of $\mathbb{F}_{q^m}$ which contain 1 to the subspaces of the quotient-space $\mathbb{F}_{q^m}/\mathbb{F}_q$ such that $\varphi(V) = V/\mathbb{F}_q$. We have $\dim \varphi(F) = \dim F - 1$ and $\dim \varphi(E) = \dim E - 1$. Moreover, $E \subset F$ if and only if $\varphi(E) \subset \varphi(F)$.

Thus, the probability that $F \supset \mathrm{Supp}(\boldsymbol{c})$ is equal to $\frac{\begin{bmatrix} r-1 \\ w-1 \end{bmatrix}}{\begin{bmatrix} m-1 \\ w-1 \end{bmatrix}} \approx q^{(w-1)(m-r)}$. We use the same techniques as before to obtain the system

$$\left\{\begin{array}{rcl} \sum_{l=1}^{n}\sum_{j=1}^{r}\lambda_{1j}\varphi_i(H'_{1l}F_j) & = & 0 \\ \vdots & & \vdots \\ \sum_{l=1}^{n}\sum_{j=1}^{r}\lambda_{1j}\varphi_i(H'_{n-k-1,l}F_j) & = & 0 \end{array}\right. \tag{3.5}$$

We take $r = \left\lfloor \frac{m(n-k-1)}{n} \right\rfloor = m - \left\lceil \frac{m(k+1)}{n} \right\rceil$ in order to have more equations than unknowns. If $F \supset E$, then the kernel of system 3.5 is of positive dimension and we can compute a non-zero vector $\boldsymbol{e}'$ solution of $\boldsymbol{H}'\boldsymbol{e}'^T = \boldsymbol{0}$. Then if it is of rank w, we solve the equation $\boldsymbol{H}\boldsymbol{e}'^T = \alpha\boldsymbol{H}\boldsymbol{e}^T = \alpha\boldsymbol{s}^T$ of unknown α to recover $\boldsymbol{e}$. Finally, we get an average complexity of $\mathcal{O}\left((n-k)^3 m^3 q^{(w-1)\left\lceil \frac{(k+1)m}{n} \right\rceil}\right)$.

3.1.2 Improvement of this algorithm

The idea of our improvement is still to search for a codeword of weight w in the code $\mathcal{C}'$. However, instead of choosing F such that $1 \in F$, we choose a random F. If it contains a multiple αE of E, then we can compute the codeword $\alpha\boldsymbol{e}$ of $\mathcal{C}'$ as before. The probability that $F \supset \alpha E$ depends on the number of different subspaces of the form $\alpha E, \alpha \in \mathbb{F}_{q^m}^*$.

Proposition 3.1.1. *Let E be an $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ of dimension w. There are at most $\frac{q^m-1}{q-1}$ subspaces of the form $\alpha E, \alpha \in \mathbb{F}_{q^m}^*$, with equality if w and m are coprime.*

Proof. Let S be the cardinality of the set of the subspaces of this form. Let α and $\beta \in \mathbb{F}_{q^m}$ such that $\alpha\beta^{-1} \in \mathbb{F}_q^*$ By linearity, it is obvious that $\alpha E = \beta E$, hence $S \leqslant \frac{q^m-1}{q-1}$.

To prove the equality, let $\alpha \in \mathbb{F}_{q^m}^*$ such that $E = \alpha E$. By recursion, we have $\mathbb{F}_q(\alpha)E = E$ thus E is an $\mathbb{F}_q(\alpha)$-subspace of $\mathbb{F}_{q^m}$. Let $m' = [\mathbb{F}_q(\alpha) : \mathbb{F}_q]$. We have $w = \dim_{\mathbb{F}_q} E = m' \dim_{\mathbb{F}_q(\alpha)} E$ hence m' divides w and m. If $GCD(m, w) = 1$ then $\alpha \in \mathbb{F}_q^*$, which concludes the proof. □

This also proves that if the number S of different subspaces of the form αE is lower than $\frac{q^m-1}{q-1}$ then E is an $\mathbb{F}_{q^{m'}}$-subspace of dimension $\frac{w}{m'}$ of $\mathbb{F}_{q^m}$. The probability of such an event is negligible, so we can assume $S = \frac{q^m-1}{q-1}$.

The probability p that F of dimension $r = m - \left\lceil \frac{m(k+1)}{n} \right\rceil$ contains any subspace of the form $\alpha E, \alpha \in \mathbb{F}_{q^m}^*$ can be approximated by the product of S by the probability that F contains a fixed subspace of dimension w. This approximation is correct if

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

This leads us to the following theorem:

Theorem 3.1.2. *Let $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}, \boldsymbol{s} \in \mathbb{F}_{q^m}^{n-k}$ and w an integer. Our algorithm solves the RSD problem 2.3.6 with average complexity of*

$$\mathcal{O}\left((n-k)^3 m^3 q^{w\left\lceil \frac{(k+1)m}{n} \right\rceil - m}\right)$$

operations in $\mathbb{F}_q$.

We have implemented our algorithm to check if our approximation is correct. We have obtained experimental values very close to the theoretic ones, which confirms our hypothesis.

In the case $m \leqslant n$, our algorithm is always better compared to the combinatorial approach from [40]. The gain in the exponent is equal to $m - \left\lceil \frac{(k+1)m}{n} \right\rceil = m - \lceil mR' \rceil \approx m(1-R')$ where R' is the rate of $\mathcal{C}'$. In the case $m > n$ our algorithm may also be faster for some parameters.

3.2 Examples of broken parameters

In this section, we present two public key encryption cryptosystems [55, 42] whose some parameters are broken by our attack. They are based on the McEliece cryptosystem [58]. The general principle of this cryptosystem is to choose a code with a decoding algorithm decoding up to w errors and to publish a masked version of this code. To encrypt a message $\boldsymbol{m}$, one computes the cipher text $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}_{pub} + \boldsymbol{e}$ where $\boldsymbol{G}_{pub}$ is the public generator matrix of the masked code ans $\boldsymbol{e}$ is an error of wight w. To decrypt, the receiver removes the mask and decodes the error.

Under the assumption that the masked code is indistinguishable from a random code, an attacker has to solve an instance of the RSD problem of weight w to recover the message.

The following tables give some parameters of [43] and [55] which are broken by our attack:

n	k	m	w	claimed security	complexity of our attack
82	41	41	4	2^{80}	2^{75}
106	53	53	5	2^{128}	2^{116}
50	32	50	3	2^{81}	2^{75}
112	80	112	4	2^{259}	2^{247}

Some parameters of these cryptosystems are not affected by our attack since it is not necessarily the best attack if $m > n$.

Our attack also breaks the parameters of the IBE in [36] but they have been updated in the eprint version of the paper.

In the process of the standardization of post-quantum cryptosystems engaged by the NIST, the cryptosystems LAKE, LOCKER, Ouroboros-R, RankSign and RQC are based on rank metric codes and take account on this attack for the choice of their security parameters. See the following link for the documentation on this submissions:

https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions

Chapter 4

Nouvel algorithme de décodage des codes LRPC

This chapter presents an improvement of the decoder for the LRPC codes that was introduced in [39]. This decoder allows to decode errors of higher rank weight, up to $\frac{2}{3}(n-k)$, when the original decoding algorithm only allowed to decode errors of weight up to $\frac{n-k}{2}$. We also present how LRPC codes can be used to build a KEM and a PKE, and we propose parameters making use of the improved decoding algorithm to reduce the Decryption Failure Rate (DFR). This work is a joint work with Philippe Gaborit, Adrien Hauteville, Olivier Ruatta and Gilles Zémor, and was published in IEEE Transactions on Information Theory [13].

Contents

4.1 Some results on the product of two subspaces

Before introducing the decoding algorithm of LRPC codes, we need to introduce some results on the product of two subspaces.

Definition 4.1.1. *Let A and B be two $\mathbb{F}_q$-subspaces of $\mathbb{F}_{q^m}$: we call the* product space *of A and B, and denote it by AB, the $\mathbb{F}_q$*-linear span *of the set of products $\{ab, a \in A, b \in B\}$.*

If A and B have dimensions α and β, and are generated respectively by $\{a_1, \cdots, a_\alpha\}$ and $\{b_1, \cdots, b_\beta\}$, then the product space AB is obviously generated by the set $\{a_ib_j, 1 \leqslant i \leqslant \alpha, 1 \leqslant j \leqslant \beta\}$ and its dimension is therefore bounded above by $\alpha\beta$.

A question of interest that concerns us is the probability that the dimension is not maximal when α and β are relatively small. Let A and B be random $\mathbb{F}_q$-subspaces of $\mathbb{F}_{q^m}$ of dimensions α and β respectively. We suppose $\alpha\beta < m$ and we investigate the typical dimension of the subspace AB.

We rely on the following lemma:

Lemma 4.1.2. *Let A' and B be two subspaces of $\mathbb{F}_{q^m}$ of dimensions α' and β such that $\dim A'B = \alpha'\beta$. Let $A = A' + \langle a \rangle$ where a is a uniformly chosen random element of $\mathbb{F}_{q^m}$. Then*

$$\operatorname{Prob}\dim(AB) < \alpha'\beta + \beta \leq \frac{q^{\alpha'\beta+\beta}}{q^m}.$$

Proof. We have $\dim(AB) < \alpha'\beta + \beta$ if and only if the subspace aB has a non-zero intersection with $A'B$. Now,

$$\begin{aligned}\operatorname{Prob}\dim(A'B \cap aB) \neq \{0\} &\leq \sum_{b \in B, b \neq 0} \operatorname{Prob} ab \in A'B \\ &\leq (|B| - 1)\frac{q^{\alpha'\beta}}{q^m}\end{aligned}$$

since for any fixed $a \neq 0$, we have that ab is uniformly distributed in $\mathbb{F}_{q^m}$. Writing $|B| - 1 \leq |B| = q^\beta$ we have the result. □

Proposition 4.1.3. *Let B be a fixed subspace and suppose we construct a random subspace A by choosing uniformly at random α independent vectors of $\mathbb{F}_{q^m}$ and letting A be the subspace generated by these α random vectors. We have that $\dim AB = \alpha\beta$ with probability at least $1 - \alpha\frac{q^{\alpha\beta}}{q^m}$.*

Proof. Apply the Lemma α times, starting with a random subspace $A' \subset A$ of dimension 1, and adding a new element to A' until we obtain A. □

In practice, our tests show that the probability that $\dim AB = rd$ is slightly larger than the probability that rd elements of $\mathbb{F}_{q^m}$ chosen uniformly at random generate a subspace of dimension rd. This difference becomes rapidly negligible, even for small value such as $\alpha = 3, \beta = 4, m = 20$.

Let B be a fixed subspace of $\mathbb{F}_{q^m}$ containing 1 and let B^2 be the subspace generated by all products of two, possibly equal, elements of B. Let $\beta_2 = \dim B^2$. Let A be a random subspace of $\mathbb{F}_{q^m}$ of dimension α. By Proposition 4.1.3 we have that $\dim(AB^2) = \alpha\beta_2$ with probability at least $1 - \alpha\frac{q^{\alpha\beta_2}}{q^m}$.
Remark: we have $\beta_2 \leqslant \beta(\beta+1)/2$.

Lemma 4.1.4. *Suppose* $\dim(AB^2) = \alpha\beta_2$. *Let* $e \in AB$ *with* $e \notin A$. *Suppose* $eB \subset AB$. *Then there exists* $x \in B$, $x \notin \mathbb{F}_q$, *such that* $xB \subset B$.

Proof. Let (a_i) be a basis of A. From $AB = a_1 B \oplus \cdots \oplus a_\alpha B$, we have

$$e = \sum_i \lambda_i a_i b_i$$

with $\lambda_i \in \mathbb{F}_q$ for all i and $b_j \notin \mathbb{F}_q$ and $\lambda_j \neq 0$ for some j, otherwise $e \in A$ contrary to our assumption. Let b be any element of B. By our hypothesis we have $eb \in AB$, meaning there exist $\mu_j \in \mathbb{F}_q$ such that

$$\sum_i \lambda_i a_i b_i b = \sum_j \mu_j a_j b'_j$$

with $b'_i \in B$. Now the maximality of the dimension of AB^2 implies that

$$\lambda_j a_j b_j b = \mu_j a_j b'_j$$

from which we deduce $b_j b \in B$. Since this holds for arbitrary $b \in B$, we have $b_j B \subset B$. □

Proposition 4.1.5. *Suppose* m *is prime. Let* A *and* B *be random subspaces of dimensions* α *and* β *respectively. Let* (b_i) *be a basis of* B *and let* $S = AB$. *Then with probability at least* $1 - \alpha\frac{q^{\alpha\beta(\beta+1)/2}}{q^m}$ *we have that* $\bigcap_i b_i^{-1} S = A$.

Proof. If not, there exists a subspace $E \supsetneq A$, such that $EB = AB$. By the remark before Lemma 4.1.4 we have that with probability at least

$$1 - \alpha\frac{q^{\alpha\beta(\beta+1)/2}}{q^m}$$

the conditions of Lemma 4.1.4 hold. But then there is $x \notin \mathbb{F}_q$ such that $xB \subset B$. But this implies that $\mathbb{F}_q(x)B \subset B$. But m prime implies that there is no intermediate field between $\mathbb{F}_q$ and $\mathbb{F}_{q^m}$, hence $\mathbb{F}_{q^m} \subset B$, a contradiction. □

A better bound Our goal is to show that with a large probability, when A and B are randomly chosen with sufficiently small dimension, then with probability close to 1 we have that :

$$\bigcap_{b \in B, b \neq 0} b^{-1}AB = A.$$

Without loss of generality we can suppose that $1 \in B$. We shall show that for a random $b \in B$, we have that $b \neq 0$ and

$$AB \cap b^{-1}AB = A$$

with probability close to 1.

When b is a random element of $\mathbb{F}_{q^m}$ we will abuse notation somewhat by letting b^{-1} denote the inverse of b when $b \neq 0$ and by letting it equal 0 when $b = 0$. With this convention in mind we have:

Lemma 4.1.6. *Let B_1 be a subspace of dimension β_1. Let b be a uniformly distributed random element of $\mathbb{F}_{q^m}$. Then the probability that $b \in B_1 + b^{-1}B_1$ is at most:*

$$\frac{2q^{2\beta_1}}{q^m}.$$

Proof. This event can only happen if b is a root of an equation of the form

$$x^2 - b_1 x - b_1' = 0.$$

There are at most $|B_1|^2 = q^{2\beta_1}$ such equations, and each one of them has at most two roots. □

Let B_1 be any vector space containing 1 and of dimension β_1. Let b be a random element uniformly distributed in $\mathbb{F}_{q^m}$ and set $B = B_1 + \langle b \rangle$. Denote $\beta = \dim B = \beta_1 + 1$ (with probability $1 - q^{\beta_1}/q^m$). Since b^{-1} is also uniformly distributed, so is $b^{-1}b_1$ for any $b_1 \in B_1$, $b_1 \neq 0$. Therefore, for any such b_1, the probability that $b^{-1}b_1 \in B_1$ equals $|B_1|/q^m$. By the union bound, the union of these events, over all $b_1 \in B_1 \setminus \{0\}$, has probability at most $(|B_1| - 1)|B_1|/q^m$. We have therefore that either $b = 0$ or $B_1 \cap b^{-1}B_1 \neq \{0\}$ with probability at most

$$(|B_1| - 1)\frac{|B_1|}{q^m} \leq \frac{q^{2\beta_1}}{q^m}.$$

Therefore with probability at least

$$1 - \frac{q^{2\beta_1}}{q^m}$$

we have that $b \neq 0$ and

$$\dim(B_1 + b^{-1}B_1) = 2\beta_1.$$

Now applying Lemma 4.1.6 we obtain:

Lemma 4.1.7. *With probability at least*

$$1 - \frac{3q^{2\beta_1}}{q^m}$$

we have that

$$\dim(B + b^{-1}B) = 2\beta - 1.$$

Proposition 4.1.8. *Let B be a subspace of dimension β containing 1 such that $\dim B + b^{-1}B = 2\beta - 1$ for some $b \in B$. Let A be a randomly chosen subspace of dimension α. With probability at least $1 - \alpha\frac{q^{\alpha(2\beta-1)}}{q^m}$ we have that $AB \cap b^{-1}AB = A$*

Proof. By Proposition 4.1.3 we have that with probability at least

$$1 - \alpha \frac{q^{\alpha(2\beta-1)}}{q^m}$$

$$\dim[A(B + b^{-1}B)] = \alpha(2\beta - 1) = 2\alpha\beta - \alpha.$$

On the other hand, we have that

$$\begin{aligned}\dim A(B + b^{-1}B) &= \dim AB + \dim b^{-1}AB \\ &\quad - \dim(AB \cap b^{-1}AB) \\ &= 2\alpha\beta - \dim(AB \cap ABb^{-1})\end{aligned}$$

hence

$$\dim(AB \cap b^{-1}AB) = \alpha.$$

But this proves the result since $A \subset AB \cap b^{-1}AB$ and $\dim A = \alpha$. □

4.2 LRPC codes and their basic decoding

LRPC codes and their basic decoding algorithm were introduced in section 2.3.4.

4.3 Improved decoding : syndrome space expansion algorithm

In general, the basic decoding algorithm presented in Section 2.3.4 does not work when the syndrome space $S = \langle s_1, ..., s_{n-k} \rangle$ is different from EF. In this section, we present an algorithm that can be used between steps 1) and 2) of the basic decoding algorithm to recover EF in certain cases when $\dim S < \dim EF$. We denote by c the codimension of S in EF, that is to say $c = \dim EF - \dim S$.

In this section, we will always suppose that $\dim EF = rd$ for two reasons:

— first, according to Proposition 4.1.3, the probability p_1 that $\dim EF < rd$ can easily be exponentially low by increasing m. We can also decrease the probability p_2 that $S \neq EF$ by increasing n, however for cryptographic applications of LRPC, the parameters of the code are such that p_1 is negligible with respect to p_2.

— secondly, the case $\dim EF = rd$ is the worst case for the decoding since the probability that $S = EF$ increases when the dimension of EF decreases. Thus, we can analyze the theoretical probability of failure in the worst case scenario. All the tests we have made show that in practice the probability of failure is smaller when $\dim EF < rd$, which confirms our analysis.

In the following, we will consider a random LRPC code with $\dim(F) = d$ and a random error vector of rank r for the different analysis.

Data: The syndrome space S and the vector space F
Result: The expanded syndrome space S', which may be EF, or failure

1. **Expand S with the expansion function**
   Use the expansion function for all possible parameters:
   $S \leftarrow f_{expand}(S)$
2. **Iterative step**
   If $\dim S = rd$, then return S. If $\dim S < rd$, go back to step 1. If $\dim S$ does not increase during step 1, return failure.

Figure 4.1 – The syndrome space expansion algorithm

4.3.1 General idea

This algorithm's aim is to recover vectors of the product space EF which do not belong to the syndrome space S by using the structure of the product space $EF = \langle f_1e_1, f_2e_1, ..., f_de_r\rangle$ and the fact that we know a basis $\langle f_1, ..., f_d\rangle$ of F. We use a function which takes on input the subspace S and outputs a subspace S' such that $S \subsetneq S' \subset EF$ with a very high probability, depending on the parameters of the code. We call such a function an "expansion function".

Let $S_i = f_i^{-1}.S$. Here are two examples of expansion functions:

1. $f_{decode}(S) = (S + f_iS_j) \cap (S + f_kS_l)$ is used to decode errors of larger weight.
2. $f_{prob}(S) = S + FS_{ij}$, where $S_{ij} = S_i \cap S_j$, is used to reduce the decoding failure rate.

We use these expansion functions to describe an iterative algorithm [4.1]. Detailed algorithms for each function are given in the next subsections.

Proposition 4.3.1. *$\dim(S_i \cap E) \geqslant r - c$, with $c = \dim EF - \dim S$.*

Proof. We have $\dim S_i = \dim S = rd - c$ and

$$\begin{aligned}\dim(E + S_i) &= \dim f_i(E + S_i) = \dim(f_iE + S)\\ &\leqslant \dim EF = rd\end{aligned}$$

since f_iE and S are subspaces of EF. Hence:

$$\begin{aligned}\dim(S_i \cap E) &= \dim E + \dim S_i - \dim(E + S_i)\\ &\geqslant r + rd - c - rd\\ &\geqslant r - c.\end{aligned}$$

□

As we will see in the next subsections, each expansion function has advantages and drawbacks, and the choice of which function to use depends on the parameters. In particular, m needs to be high enough for these functions to work. The function f_{decode} needs $m \geq 3rd - 2r$ and allows to decode errors of weight up to $\frac{2}{3}(n-k)$ whereas the function f_{prob} only needs $m \geq 2rd - r$. In this case we can only decode errors of weight up to $\frac{(n-k)}{2}$ but by strongly reducing the decoding failure probability. In particular, this latter case corresponds to parameters used for cryptography, where $rd \approx (n-k) = \frac{n}{2}$ and $m \approx n$.

```
1: Input: The syndrome space $S$ and the vector space $F$
2: Output: The expanded syndrome space $S'$, which may be $EF$, or failure
3: while true do
4:     $tmp \leftarrow \dim S$
5:     for Every $(i,j,k,l)$ such that $i \neq j$, $k \neq l$ and $(i,j) \neq (k,l)$ do
6:         $S \leftarrow (S + f_i.S_j) \cap (S + f_k.S_l)$
7:     if $\dim S = rd$ then return $S$
8:     if $\dim S = tmp$ then return failure
```

Figure 4.2 – Syndrome space expansion using f_{decode}

4.3.2 Increasing the weight of the decoded error vectors

In this subsection we explain in details the function f_{decode} (the algorithm is described figure 4.2).

$$f_{decode} : S \mapsto (S + f_iS_j) \cap (S + f_kS_l).$$

The idea is that each subspace of the form $(S + f_iS_j)$ has a chance to contain vectors x such that $x \in EF$ and $x \notin S$, since $EF \cap f_iS_j \neq \{0\}$. If we find some new vectors of EF but not the whole product space, we can reuse values of (i,j,k,l) already tested because they may lead us to recover some new vectors thanks to the one we have added. Hence the algorithm functions in an iterative way. If during a whole iteration, the dimension of S does not increase, then the algorithm will fail.

Maximum weight of the decoded errors. Without using the syndrome space expansion algorithm, the decoding algorithm for LRPC codes can decode up to $r = \frac{n-k}{d}$ errors, where d is the rank of F. We can use f_{decode} to deal with error vectors of increased weight.

Theorem 4.3.2. *For $d = 2$, under the assumption that $\dim(S_i \cap E) = r - c$ for $1 \leqslant i \leqslant 2$, f_{decode} allows to decode errors of weight up to $r \leqslant \left\lfloor \frac{2}{3}(n-k) \right\rfloor$ with probability $\approx 1 - \frac{q^{3r-2(n-k)}}{q-1}$ for the general case and errors of weight up to $r = \frac{2}{3}(n-k)$ with probability ≈ 0.29 for the case $q = 2$.*

Proof. $d = 2 \implies \dim EF = 2r$.
Thus, the dimension of $EF + f_1f_2^{-1}EF$ is at most $3r$ for $f_1E \subset EF \cap f_1f_2^{-1}EF$. The only expansion that can be done is $S \leftarrow (S + f_1f_2^{-1}S) \cap (S + f_2f_1^{-1}S)$, hence there is no iterative effect.

Since $(S + f_1f_2^{-1}S) = f_1f_2^{-1}(S + f_2f_1^{-1}S)$, the dimension of these two vectors spaces is the same. If $\dim(S + f_1f_2^{-1}S) < 3r$, then it is impossible to find the whole space EF, which leads to a decoding failure. Since the dimension of S is at most $n-k$, the dimension of $S + f_if_j^{-1}S$ is at most $2(n-k)$, which implies $3r \leqslant 2(n-k)$. Hence $r \leqslant \left\lfloor \frac{2}{3}(n-k) \right\rfloor$.

Since we have $rd > (n-k)$, we are going to suppose that $\dim S = n-k$, given that this will happen with a really high probability compared to the overall decoding probability.

Considering that S and $f_1f_2^{-1}S$ behave like independent variables, the probability of success is the probability that the sum of two subspaces of dimension $n-k$ of $EF +$

$f_1 f_2^{-1} EF$ generates the whole space. So we have:

$$\begin{aligned} p_{success} &= \text{Prob}(S + f_1 f_2^{-1} S = EF + f_1 f_2^{-1} EF) \\ &= \text{Prob}(\dim(S + f_1 f_2^{-1} S) = 3r) \\ &= \text{Prob}(2(n-k) - \dim S \cap (f_1 f_2^{-1} S) = 3r) \\ &= \text{Prob}(\dim S \cap (f_1 f_2^{-1} S) = 2(n-k) - 3r). \end{aligned}$$

The formula for computing this probability can be found in [48] (Proposition 3.2). We obtain:

$$\begin{aligned} p_{success} &= \frac{\begin{bmatrix} 2(n-k)-3r \\ n-k \end{bmatrix} q^{(3r-(n-k))^2}}{\begin{bmatrix} n-k \\ 3r \end{bmatrix}} \\ &= \frac{\begin{bmatrix} 3r-(n-k) \\ n-k \end{bmatrix} q^{(3r-(n-k))^2}}{\begin{bmatrix} 3r-(n-k) \\ 3r \end{bmatrix}}. \end{aligned} \tag{4.1}$$

By applying the series expansion for $P_{a,b}(n)$ 4.3.10 with $n = 3r, a = b = 3r - (n-k)$, we obtain:

$$\begin{aligned} p_{success} &\approx 1 - \frac{q^{-3r}(q^{3r-(n-k)} - 1)^2}{q-1} \\ &\approx 1 - \frac{q^{3r-2(n-k)}}{q-1}. \end{aligned}$$

In the case $q = 2$ and $3r = 2(n-k)$, this approximation is incorrect. According to Equation (4.1),

$$\begin{aligned} p_{success} &= \frac{\begin{bmatrix} n-k \\ n-k \end{bmatrix}_2 2^{(n-k)^2}}{\begin{bmatrix} n-k \\ 3r \end{bmatrix}_2} \\ &= \frac{2^{(n-k)^2}}{\begin{bmatrix} n-k \\ 2(n-k) \end{bmatrix}_2} \\ &\longrightarrow 0.29 \text{ when } (n-k) \to \infty \\ &\text{(the convergence is very fast).} \end{aligned}$$

□

In figure 4.3, we present simulation results comparing the observed probability of failure and q^{-1}.

Proposition 4.3.3. *Under the assumption that* $\dim(S_i \cap E) = r - c$ *for* $1 \leqslant i \leqslant d$, f_{decode} *allows to decode errors up to* $r \leqslant \frac{n-k}{d-1}$ *for* $d > 2$, *using the fact that the algorithm functions in an iterative way.*

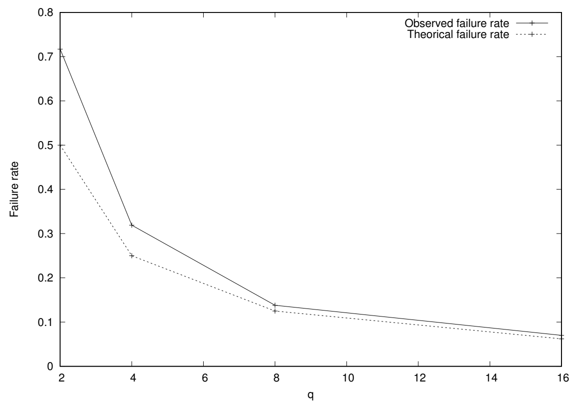

Figure 4.3 – Results for $d = 2$, $n = 30$, $k = 15$, $r = 10$. We can observe the success rate of ≈ 0.29 for $q = 2$.

This result has been verified by simulation, here is a heuristic explaining how it works: for $d > 2$, the algorithm may recover EF as long as $c < r$ (i.e there are elements of E in S_i) using the iterative effect, so the algorithm does not have to recover the whole product space at a time : if it recovers one new vector then it can be added to S and used with different values of (i, j, k, l) (even those that have already been used) to produce more elements of EF, hence the only condition for that method to work is $c < r$ in order to have vectors of E in S_i.

If the $n-k$ syndrome coordinates are independent, then c is $rd-(n-k)$, which gives the result:

$$\begin{aligned} c < r &\Leftrightarrow rd - (n-k) < r \\ &\Leftrightarrow r(d-1) < n-k \\ &\Leftrightarrow r < \frac{n-k}{d-1}. \end{aligned}$$

Although giving a theorical probability of failure is difficult in this case since there is an iterative effect, simulations show that in practice we attain this bound.

Minimal value of m

Proposition 4.3.4. *f_{decode} cannot recover new elements of EF if $m < 3rd - 2r$*

Proof. To produce elements of EF, we compute vector spaces of the form $f_i f_j^{-1} S$. If we suppose that $f_j^{-1}S$ contains exactly $r-c$ independent elements of E, which is the worst case, then we get $r-c$ elements of EF and $r(d-1)$ random elements in $S'_{ij} = f_i f_j^{-1} S$.

By adding this vector space to S, the dimension cannot exceed $rd + r(d-1) = 2rd - r$ i.e the dimension of EF along with the random elements in S'_{ij}.

The intersection between $S + S'_{ij}$ and $S + S'_{kl}$ needs to be exactly of dimension rd in order to recover elements that are $\in EF$, whence:

$$\begin{aligned}
&\dim(S+S'_{ij})+\dim(S+S'_{kl}) \leqslant m+\dim((S+S'_{ij})\cap(S+S'_{kl}))\\
\Leftrightarrow &2(2rd-r) \leqslant m+rd\\
\Leftrightarrow &m \geqslant 4rd-2r-rd\\
\Leftrightarrow &m \geqslant 3rd-2r.
\end{aligned}$$

Hence the result. □

The value of m needed by this expansion function is pretty high and does not fit the parameters used for cryptography (see the parameters in 4.4.5 for example). However, it works for $d = 2$, which is not the case of f_{prob} studied in the next subsection.

4.3.3 Reducing the decryption failure rate

Description of the algorithm

In this subsection we will study f_{prob} in a more detailed way (the algorithm is described figure 4.4.

$$f_{prob} : S \mapsto S + FS_{ij}.$$

This function fits the parameters used for cryptography and is used to reduce the decoding failure rate. It uses the fact that $S_{ij} = S_i \cap S_j$ contains at least $r-2c$ independent elements of E (see Proposition 4.3.5) and is a subspace of E with a very high probability when m is large enough. Thus, the subspace FS_{ij} has a chance to contain a vector $x \in EF$ and $x \notin S$.

```
1: **Input:** The syndrome space $S$ and the vector space $F$
2: **Output:** The expanded syndrome space $S'$, which may be $EF$, or failure
3: while $true$ do
4:     $tmp \leftarrow \dim S$
5:     for Every $(i,j)$ such that $i \neq j$ do
6:         $S \leftarrow S + FS_{ij}$
7:     if $\dim S = rd$ then return $S$
8:     if $\dim S = tmp$ then return failure
```

Figure 4.4 – Syndrome space expansion using f_{prob}

Proposition 4.3.5. *$\dim(S_{ij} \cap E) \geqslant r - 2c$ for all $i, j \in [1..d]$, with $c = \dim EF - \dim S$.*

Proof. Using Proposition 4.3.1, a simple computation gives us the result:

$$\begin{aligned}
\dim(S_{ij}\cap E) &= \dim(S_i\cap S_j\cap E)\\
&= \dim(S_i\cap E)+\dim(S_j\cap E)\\
&\quad -\dim\big((S_i\cap E)+(S_j\cap E)\big)\\
&\geqslant 2(r-c)-r\\
&\geqslant r-2c.
\end{aligned}$$

□

Theorem 4.3.6. *Let* $\mathrm{Prob}(c = l)$ *be the probability that* $\dim EF - \dim S = l$ *and let* $\mathrm{Prob}(failure|c = l)$ *be the probability that the syndrome space expansion algorithm does not recover* EF *when* $c = l$*. Using the syndrome space expansion algorithm with* f_{prob}*, the probability of failure is :*

$$\mathrm{Prob}(c \geqslant \frac{r}{2}) + \sum_{l=1}^{\lfloor \frac{r}{2} \rfloor -1} \mathrm{Prob}(c = l) \times \mathrm{Prob}(failure|c = l).$$

Proof. The main source of decoding failures for the basic decoding algorithm is the fact that the syndrome coordinates do not generate the whole product space EF, so we need to compute the global probability of not recovering EF using f_{prob}.

When $c \geqslant \frac{r}{2}$, in general S_{ij} is empty so the syndrome space expansion algorithm will not recover EF, which leads to a failure.

When $c < \frac{r}{2}$, the probability that the syndrome space expansion algorithm recovers EF is $\mathrm{Prob}(failure|c = l)$. We multiply this probability by the probability of c being equal to l to get the overall probability of not recovering EF. □

Proposition 4.3.7. $\mathrm{Prob}(c = l) \simeq q^{-l(n-k-rd+l)}$

Proof. The probability that the $n-k$ syndrome coordinates generate a subspace of codimension l of EF is the probability that a random $(n-k) \times rd$ matrix in $\mathbb{F}_q$ is of rank $rd - l$. From [54], we have that S_t, the number of $m \times n$ matrix of rank t, is :

$$S_t = \prod_{j=0}^{t-1} \frac{(q^n - q^j)(q^m - q^j)}{(q^t - q^j)}.$$

We can approximate this formula by ignoring q^j, which leads to :

$$S_t \simeq \prod_{j=0}^{t-1} \frac{(q^n)(q^m)}{(q^t)} = \prod_{j=0}^{t-1} q^{n+m-t} = q^{t(n+m-t)}.$$

From that, we deduce the number of $(n-k) \times rd$ matrices of rank $rd - l$:

$$q^{(rd-l)(n-k+rd-rd+l)} = q^{(rd-l)(n-k+l)}.$$

By dividing this quantity by the total number of $(n-k) \times rd$ matrices, we get the probability that a random matrix has rank $rd - l$:

$$\begin{aligned}\frac{q^{(rd-l)(n-k+l)}}{q^{(n-k)rd}} &= q^{(rd-l)(n-k+l)-(n-k)rd} \\ &= q^{lrd-l(n-k)-l^2} \\ &= q^{-l(n-k-rd+l)}\end{aligned}$$

Hence the result. □

Case of codimension 1: suppose dim S = dim EF - 1

Before giving the probability of success of Algorithm 4.4, we need the following lemma:

Lemma 4.3.8. *The dimension of $S_i \cap E$ is :*

— r *with probability* $p_r = \dfrac{\begin{bmatrix} rd-1 \\ r \end{bmatrix}}{\begin{bmatrix} rd \\ r \end{bmatrix}} \approx q^{-r}$.

— $r-1$ *with probability* $1 - p_r \approx 1 - q^{-r}$.

Proof. We have: $\dim S_i \cap E = r \iff E \subset S_i \iff f_i E \subset S$.
The vector space $f_i E$ is a subspace of dimension r of EF and S is a random subspace of dimension $rd-1$ of EF, hence

$$\text{Prob}(\dim(S_i \cap E) = r) = \frac{\begin{bmatrix} rd-1 \\ r \end{bmatrix}}{\begin{bmatrix} rd \\ r \end{bmatrix}} \approx q^{-r}$$

which proves the first point. According to Proposition 4.3.1, $\dim(S_i \cap E) \geqslant r-1$ which concludes the proof of this lemma. □

We can now give the probability of success of the decoding algorithm in the case $\dim S = \dim EF - 1$.

Theorem 4.3.9. *Under the assumption that* $S_{ij} \subset E \ \forall \ (i,j)$ *and* $\dim S = \dim EF - 1$, *the syndrome space expansion algorithm using* f_{prob} *recovers* EF *with probability at least* $1 - q^{(2-r)(d-2)}$.

Proof. To find an upper bound on the probability of failure, we are going to focus on the case where $\dim(S_{ij} \cap E) = r-2$. Indeed, Lemma 4.3.8 shows that the typical dimension of $S_i \cap E$ is $r-1$. Furthermore, when there exists i such that $\dim(S_i \cap E) = r$, the algorithm has access to more elements of E and so this leads to a smaller probability of failure. The same thing goes for the case where there exists $\dim(S_{ij}) = r-1$ instead of $r-2$ which is the expected size, so we can consider that $\dim(S_{ij}) = r-2$ since this will lead to an upper bound of the probability of failure.

We are now going to study the dimension of $S_{ij} + S_{jk} + S_{ik}$. There are two possibilities:

— If $S_{ij} = S_{jk}$, then necessarily we have $S_{ik} = S_{ij} = S_{jk}$, because every elements of $S_{ij} + S_{jk}$ are both in S_i and S_k. This happens with probability q^{2-r} : the probability that two subspaces of dimension $r-2$ (S_{ij} and S_{jk}) of a vector space of dimension $r-1$ ($S_j \cap E$) are equal.

— If $S_{ij} \neq S_{jk}$, then we have $S_{ik} + S_{ij} + S_{jk} = E$: we know that $S_{ik} + S_{ij} = S_i \cap E$, and that $S_{jk} \not\subset S_i$ (otherwise we would be in the first case). Hence $S_{ik} + S_{ij} + S_{jk} = E$.

If there exists a set $\{S_{i_1 j_1}, \dots, S_{i_n j_n}\}$ such that their sum equals E, then the algorithm will succeed since it will have added exactly EF to S at one point. If such a set does not exist, then it means that every S_{ij} are equal to each other. In particular, every vector of E

the algorithm as access to is included in every S_i, which means that when we multiply this vector by F, the resulting elements will already be in S and we will obtain no information, causing the algorithm to fail.

The probability that all the S_{ij} are equal is the probability of $S_{12} = S_{23} = \cdots = S_{d-1,d}$. Since $d-2$ equalities are needed, the probability of this happening is $q^{(2-r)(d-2)}$, which gives the probability of failure. □

Impact of m on decoding. If the value of m is not high enough, then S_{ij} will contain not only elements of E but also random elements. When this happens, the S_{ij} cannot be used by the expansion algorithm because it would add random elements to S.

Proposition 4.3.10. *Let E be a random subspace of $\mathbb{F}_{q^m}$ of dimension r. Let F_1 and F_2 be two random subspaces of $\mathbb{F}_{q^m}$ such that*

— $\dim F_i = d_i$

— $\dim(E \cap F_i) = r - c_i$

Then $\dim(F_1 \cap F_2) \geqslant r - c_1 - c_2$ *and we have*

$$\mathrm{Prob}[(F_1 \cap F_2) \subset E] \geqslant \frac{\begin{bmatrix} m - d_1 - c_1 \\ d_2 - r + c_2 \end{bmatrix}}{\begin{bmatrix} m - r \\ d_2 - r + c_2 \end{bmatrix}} q^{(d_1 - r + c_1)(d_2 - r + c_2)}.$$

If $m \geqslant d_1 + d_2$, *we have* $\mathrm{Prob}[(F_1 \cap F_2) \subset E] \geqslant 1 - \frac{q^{-m+r}(q^{d_1-r+c_1}-1)(q^{d_2-r+c_2}-1)}{q-1} + \mathcal{O}\left(q^{-2m}\right)$ *when* $m \to \infty$.

Proof. We have:

$$\begin{aligned}
&\dim(F_1 \cap F_2) \geqslant \dim(E \cap F_1 \cap F_2) \\
=& \dim(E \cap F_1) + \dim(E \cap F_2) \\
&\qquad - \dim\big((E \cap F_1) + (E \cap F_2)\big) \\
=& 2r - c_1 - c_2 - \dim(E \cap F_1 + E \cap F_2) \\
\geqslant& 2r - c_1 - c_2 - \dim E = r - c_1 - c_2.
\end{aligned}$$

For the second point, we need to define the projection over the quotient vector space $\mathbb{F}_{q^m}/E$.

$$\begin{aligned}
\text{Let } \varphi : &\mathbb{F}_{q^m} \to \mathbb{F}_{q^m}/E \\
&\boldsymbol{x} \mapsto \boldsymbol{x} + E
\end{aligned}$$

φ is a linear map of $\mathbb{F}_q$-vector spaces. For all $U \subset \mathbb{F}_{q^m}, U \subset E \iff \varphi(U) = \{0\}$. Moreover we have the equality $\dim \varphi(U) = \dim U - \dim(E \cap U)$. Therefore,

$$\left\{ \begin{array}{l} \dim \varphi(F_i) = d_i - r + c_i \\ F_1 \cap F_2 \subset E \iff \varphi(F_1 \cap F_2) = \{0\} \end{array} \right. .$$

Since $\varphi(F_1 \cap F_2) \subset \varphi(F_1) \cap \varphi(F_2)$, $\varphi(F_1) \cap \varphi(F_2) = \{0\} \implies \varphi(F_1 \cap F_2) = \{0\}$. So

$$\mathrm{Prob}(F_1 \cap F_2 \subset E) \geqslant \mathrm{Prob}(\dim(\varphi(F_1) \cap \varphi(F_2)) = 0).$$

Supposing that $a+b \leqslant m$, the probability that two subspaces of dimension a and b of a vector space of dimension n have a null intersection is given by the formula

$$\frac{\begin{bmatrix} n-b \\ a \end{bmatrix} q^{ab}}{\begin{bmatrix} n \\ a \end{bmatrix}}$$

(see [48] Proposition 3.2). By taking $n = m-r, a = d_2 - r + c_2$ and $b = d_1 - r + c_1$, we obtain

$$\text{Prob}\,(F_1 \cap F_2) \subset E \geqslant \frac{\begin{bmatrix} m-d_1-c_1 \\ d_2-r+c_2 \end{bmatrix}}{\begin{bmatrix} m-r \\ d_2-r+c_2 \end{bmatrix}} q^{(d_1-r+c_1)(d_2-r+c_2)}.$$

To compute an approximation, let us use the formula $\begin{bmatrix} n \\ a \end{bmatrix} = \prod_{i=0}^{a-1} \frac{q^n-q^i}{q^a-q^i}$.

$$\begin{aligned} P_{a,b}(n) = \frac{\begin{bmatrix} n-b \\ a \end{bmatrix} q^{ab}}{\begin{bmatrix} n \\ a \end{bmatrix}} &= \frac{\prod_{j=0}^{a-1} \frac{q^{n-b}-q^j}{q^a-q^j}}{\prod_{i=0}^{a-1} \frac{q^n-q^i}{q^a-q^i}} q^{ab} \\ &= \left(\prod_{i=0}^{a-1} \frac{q^{n-b}-q^i}{q^n-q^i} \right) q^{ab} = \prod_{i=0}^{a-1} \frac{q^n-q^{i+b}}{q^n-q^i} \\ &= \prod_{i=0}^{a-1} \frac{1-q^{i+b-n}}{1-q^{i-n}}. \end{aligned}$$

Therefore,

$$\ln P_{a,b}(n) = \sum_{i=0}^{a-1} [\ln(1-q^{i+b-n}) - \ln(1-q^{i-n})].$$

$\ln(1-x) = -x + \mathcal{O}(x^2)$ when $x \to 0$ hence

$$\begin{aligned} \ln P_{a,b}(n) &= \sum_{i=0}^{a-1} \left(q^{i-n} - q^{i+b-n} \right) + \mathcal{O}\left(q^{-2n}\right) \\ &= q^{-n}(1-q^b) \sum_{i=0}^{a-1} q^i + \mathcal{O}\left(q^{-2n}\right) \\ &= -q^{-n}(1-q^b)\frac{q^a-1}{q-1} + \mathcal{O}\left(q^{-2n}\right) \end{aligned}$$

when $n \to +\infty$.
$e^x = 1 + x + \mathcal{O}(x^2)$ when $x \to 0 \implies P_{a,b}(n) = 1 - \frac{q^{-n}(q^a-1)(q^b-1)}{q-1} + \mathcal{O}\left(q^{-2n}\right)$ By replacing the variables, we obtain

$$\begin{aligned} \text{Prob}(F_1 \cap F_2 \subset E) \geqslant 1 - &\frac{q^{-m+r}(q^{d_1-r+c_1}-1)(q^{d_2-r+c_2}-1)}{q-1} \\ &+ \mathcal{O}\left(q^{-2m}\right). \end{aligned}$$

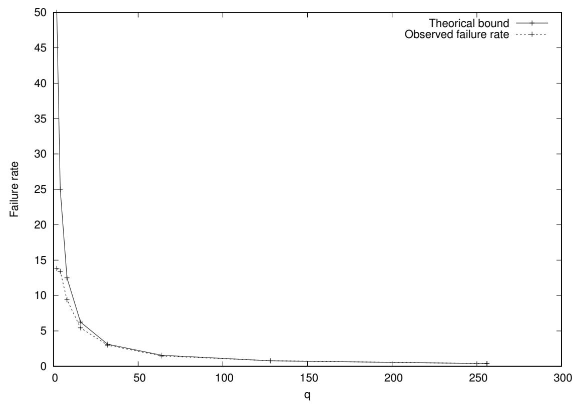

Figure 4.5 – Simulation results for $c = 1$, $d = 3, r = 3$

$\square$

Corollary 4.3.11. *Applying Proposition 4.3.10 with $F_i = S_i$, $\dim S_i = rd - 1$ and $c_i = 1$, we obtain* $\text{Prob}\,(S_i \cap S_j) \subset E \approx 1 - \frac{q^{-m+2rd-r}}{q-1}$.

From corollary 4.3.11, we can see that adding 1 to the value of m leads to a factor q^{-1} in the probability of finding random elements in S_{ij}, thus this probability can be made arbitrarily small.

To detect that S_{ij} contains random elements, we can keep track of the dimension of S during the expansion. Indeed, if S_{ij} contains random elements, adding FS_{ij} to S will lead to a dimension greater than rd. In this case we just need to discard the elements we just added and continue with the next S_{ij}.

Comments on the small values of q When using small values of q, the cases where $\dim S_i \cap E$ or $\dim S_{ij} > r - 2$ are not so rare. Simulation for $d = 3, r = 3$ seems to show that the probability $q^{(2-r)(d-2)}$ is accurate when q is high enough, but is a pessimistic bound when q is small, especially when $q = 2$. Simulation results are shown in figure 4.5.

Proposition 4.3.12. *For $q = 2$, the syndrome space expansion algorithm using f_{prob} recovers EF with probability at least* $1 - q^{(1-r)(d-2)}$.

Proof. When $q = 2$, the probability that $S_{ij} = S_{jk}$ used in the proof above is not q^{2-r} but q^{1-r}. Indeed, from Proposition 4.3.10, we get that the probability of S_{ij} and S_{jk} being equal is approximately $(q-1)q^{1-r}$ which is equal to q^{1-r} when $q = 2$. $\square$

This leads to a failure probability of $q^{(1-r)(d-2)}$. This is still not as good as what happens in practice, since it does not take into account the fact that the algorithm often has more than $r - 2$ elements of E in each S_{ij} to work with, but is a closer estimate in this special case.

Case of codimension $\geqslant 2$

In the case $c \geqslant 2$, we have no theoretical analysis, but we can do simulations for particular parameters to estimate the probability of failure. In practice we obtain good estimations of the probability of failure. For example, for $q = 2$, $c = 2$, $r = 5$, $d = 6$ and $m = 80$, we observe a decoding failure rate of 2^{-14}.

Upper bound on the decryption failure rate

To summarize : the probability of the codimension of S in EF being 1 is $q^{(n-k)-rd}$ (proposition), and the algorithm fails with probability $q^{(2-r)(d-2)}$ (theorem 4.3.9) when this is the case. The algorithm can also fail because of the codimension being greater than 1, which happens with probability $\simeq q^{-2(n-k-rd+2)}$ (proposition 4.3.7), and we consider that the algorithm fails in that case. That gives us the following upper bound of the decryption failure rate :

$$q^{(2-r)(d-2)} \times q^{-(n-k-rd+1} + q^{-2(n-k-rd+2)}.$$

4.3.4 Tradeoff between expansion functions

In the previous subsections we studied two expansion functions in detail:

— f_{decode} can decode for higher values of r but needs a higher m ($m \geq 3rd - r$)

— f_{prob} fits values of m ($m \geq 2rd - r$) better suited for cryptography, but is less effective since it decodes up to $c < \frac{r}{2}$ instead of $c < r$

Depending on the parameters, we can build new expansion functions: in general, the higher m is, the better the expansion function will be. For example, if m is really high, we can use this function: $S \leftarrow (S + f_iS_j + f_kS_l) \cap (S + f_aS_b + f_cS_d)$ which is basically an improvement of f_{decode} since it has a higher chance of recovering elements of EF by adding two f_iS_j at a time.

If m is lower than the one needed for f_{prob}, one can use the function $S \leftarrow S + F(S_{ij} \cap S_k)$ which requires a lower m but can only decode with $c < \frac{r}{3}$.

4.3.5 Rank Support Recovery algorithm

Algorithm 2.1 computes the error vector from the syndrome equations $\boldsymbol{He}^T = \boldsymbol{s}^T$. However, for cryptographic applications, we may just recover the support of the error and use it as the shared secret since the $I - RSR$ and $I - RSD$ problems are equivalent. We use the Rank Support Recovery (RSR) algorithm 4.6 which is a shortened version of the decoding algorithm 2.1 which does not compute the coordinates of the error, and uses f_{prob} to reduce the decoding failure rate.

Algorithm 4.4 allows to reduce the decryption failure rate by a factor $q^{(2-r)(d-2)}$. However, this algorithm takes a variable number of iterations to terminate, which may lead to big variations in the global execution time. In particular, if one can determine whether the syndrome space expansion algorithm was needed or not, the execution time may leak some information. To prevent that, we present a variation of the syndrome space recovery algorithm in [4.7] that performs a fixed number of intersections and still fits the proof of Theorem 4.3.9.

```
1: Input: A parity-check matrix $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ of support $F$ of dimension $d$ of an LRPC
   code, a syndrome $\boldsymbol{s} \in \mathbb{F}_{q^m}^{n-k}$, an integer $r$.
2: Output: A subspace $E$ of dimension $\leqslant r$ such that it exists $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$ of support $E$
   with $\boldsymbol{He}^T = \boldsymbol{s}^T$.
3: Compute the syndrome space $S = \langle s_1, \cdots, s_{n-k} \rangle$
4: Use the syndrome space expansion algorithm described in Algorithm 4.4
5: Define $S_i = f_i^{-1}EF$ then compute the support of the error $E = S_1 \cap S_2 \cap \cdots \cap S_d$
   return $E$
```

Figure 4.6 – the Rank Support Recovery algorithm

```
1: Input: The syndrome space $S$ and the vector space $F$
2: Output: The expanded syndrome space $S'$, which may be $EF$
3: for $i$ from 1 to $d-1$ do
4:     Precompute $S_{i,i+1}$
5: for $i$ from 1 to $d-2$ do
6:     $tmp \leftarrow S + F(S_{i,i+1} + S_{i+1,i+2} + S_{i,i+2})$
7:     if $\dim tmp \leqslant rd$ then
8:         $S \leftarrow tmp$
   return $S$
```

Figure 4.7 – Syndrome space expansion algorithm used in cryptography

4.3.6 Complexity of the decoding algorithm

Algorithms 4.2 and 4.4 are iterative algorithms, hence their exact complexity is hard to study precisely. We can however give an upper bound of their complexity, by noticing that the maximum number of iterations before the algorithms finishes is at most c: indeed, c must decrease at each iteration, otherwise the algorithms will stop returning $failure$.

Proposition 4.3.13. *The number of operations in $\mathbb{F}_q$ for the syndrome space expansion algorithm using f_{decode} in algorithm 4.2 is bounded above by:*

$$c \times d(d-1) \times (d(d-1)-1) \times 16r^2d^2m$$

Proof. As said before, the complexity of this algorithm can be bounded by c times the cost of an iteration. When using f_{decode}, the most costly operation is the computation of the intersection. Intersecting two vector spaces of dimension $2rd$ costs $16r^2d^2m$ operations in $\mathbb{F}_q$, and $d(d-1) \times (d(d-1)-1)$ different values are possible for (i,j,k,l) at each iteration, hence the result. □

Proposition 4.3.14. *The number of operations in $\mathbb{F}_q$ for the syndrome space expansion algorithm using f_{prob} in algorithm 4.4 is bounded above by :*

$$c \times d(d-1) \times 4r^2d^2m$$

Proof. The proof is the same as for proposition 4.3.13, except that the most costly step of each iteration is the computation of the S_{ij}. Intersecting two vector spaces of dimension

rd costs $4r^2d^2m$ operations in $\mathbb{F}_q$, and there are $d(d-1)$ S_{ij} to compute at each iteration, hence the result. □

In order to allow constant time implementations, the syndrome space expansion algorithm used in the cryptographic context [4.7] computes a fixed number of intersections, so its complexity can be derived easily:

Proposition 4.3.15. *The number of operations in $\mathbb{F}_q$ for the syndrome space expansion algorithm used in the cryptographic context in algorithm 4.7 is:*

$$((d-1)+(d-2)) \times 4r^2d^2m$$

Proof. As for Proposition 4.3.14, the most costly step of this algorithm is the computation of every S_{ij}. Each intersection costs $4r^2d^2m$, and the structure of the algorithm allows to reduce the number of intersections to $(d-1)+(d-2)$, hence the result. □

In practice the algorithm for cryptographic applications is very efficient, and for the iterative case, the upper bound is scarcely met since the algorithm usually stops before doing the maximum number of possible iterations.

4.4 Applications to cryptography

LRPC codes are good candidates for use in the McEliece cryptosystem [58]. As a reminder, the McEliece cryptosystem is a public key encryption scheme based on coding theory. Basically, the secret key is the description of a code $\mathcal{C}$ that we can efficiently decode and the public key is a scrambled description of this code. To encrypt a message M, one computes a codeword $c_M \in \mathcal{C}$ and one adds to it an error e. The ciphertext is $C = c_M + e$. If we know the "good" description, we can decode C and recover the message whereas an attacker has to either apply generic decoding algorithms for arbitrary codes, which amounts to trying to solve an NP-hard problem, or recover the hidden structure of $\mathcal{C}$.

Ideal LRPC codes are easy to hide since we only need to reveal their systematic parity-check matrix. Due to their weak algebraic structure, it is hard to recover the structure of an LRPC code from its systematic form. We can now introduce a new problem on which the security of our cryptosystem is based:

Problem 4.4.1 (Ideal LRPC codes indistinguishability)**.** *Given a polynomial $P \in \mathbb{F}_q[X]$ of degree n and a vector $\boldsymbol{h} \in \mathbb{F}_{q^m}^n$, it is hard to distinguish whether the ideal code $\mathcal{C}$ with the parity-check matrix generated by $\boldsymbol{h}$ and P is a random ideal code or whether it is an ideal LRPC code of weight d.*

In other words, it is hard to distinguish whether $\boldsymbol{h}$ was sampled uniformly at random or as $\boldsymbol{x}^{-1}\boldsymbol{y} \mod P$ where the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same support of small dimension d.

In [49], a structural attack against DC-LRPC codes is made by using the divisors of $X^n - 1$ in $\mathbb{F}_q[X]$. That is why the ideal LRPC codes are particularly interesting if we choose an irreducible polynomial for P. In this case we utterly counter this structural attack.

In the current state of the art, there is no structural attack against generic LRPC codes : the attack [25] on RankSign only works on very specific parameters: it is needed that the weight d of the secret parity-check matrix is less or equal to $\frac{n}{n-k}$, which is not the case for the parameters described in 4.4.5.

Thus, the most efficient known attack against this problem consists of searching for a codeword of weight d in the dual $\mathcal{C}^\perp$ of the LRPC code. Indeed, $\boldsymbol{h} = \boldsymbol{x}^{-1}\boldsymbol{y} \mod P \iff \boldsymbol{y} + \boldsymbol{x}\boldsymbol{h} = 0 \mod P$ so $(\boldsymbol{x}|\boldsymbol{y}) \in \mathcal{C}^\perp$. The small weight codeword research algorithms are the same as for the algorithm solving the RSD problem, so we consider the algorithm of [14]. However this algorithm is more efficient for LRPC codes than for random codes. Indeed, since an LRPC code admits a parity-check matrix $\boldsymbol{H}$ of small weight d, any $\mathbb{F}_q$-linear combinations of the rows of $\boldsymbol{H}$ is a codeword of weight at most d of $\mathcal{C}^\perp$. In the case of Ideal-LRPC codes, the complexity of the algorithm is divided by q^n. Thus the complexity of the attack is $\mathcal{O}\left(n^3m^3q^{d\lceil\frac{m}{2}\rceil-m-n}\right)$.

We can also consider algebraic attacks using Groebner bases [52]. The advantage of these attacks is that they are independent of the size of q. They mainly depend on the number of unknowns with respect to the number of equations. However, in the case $q = 2$ the number of unknowns is generally too high for the algorithms by Groebner bases to be more efficient than combinatorial attacks described in 4.4.4. We have chosen our parameters in such a way so that the best attacks are combinatorial: the expected complexity of the algorithms by Groebner basis is based on the article [23].

4.4.1 An IND-CPA KEM based on the rank metric

Definition of a KEM

A Key-Encapsulation scheme KEM = $(\mathsf{KeyGen}, \mathsf{Encap}, \mathsf{Decap})$ is a triple of probabilistic algorithms together with a key space $\mathcal{K}$. The key generation algorithm KeyGen generates a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$. The encapsulation algorithm Encap uses the public key pk to produce an encapsulation c, and a key $K \in \mathcal{K}$. Finally Decap using the secret key sk and an encapsulation c, recovers the key $K \in \mathcal{K}$ or fails and return $\perp$.

We define IND-CPA-security of KEM formally via the following experiment, where Encap_0 returns a valid key pair c^*, K^*, while Encap_1 returns a valid c^* and a random K^*.

Indistinguishability under Chosen Plaintext Attack: This notion states that an adversary should not be able to efficiently guess which key is encapsulated.

```
$\mathbf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\lambda)$
    1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
    2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
    3. $(c^*, K^*) \leftarrow \mathsf{Encap}_b(\mathsf{pk})$
    4. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : c^*, K^*)$
    5. RETURN $b'$
```

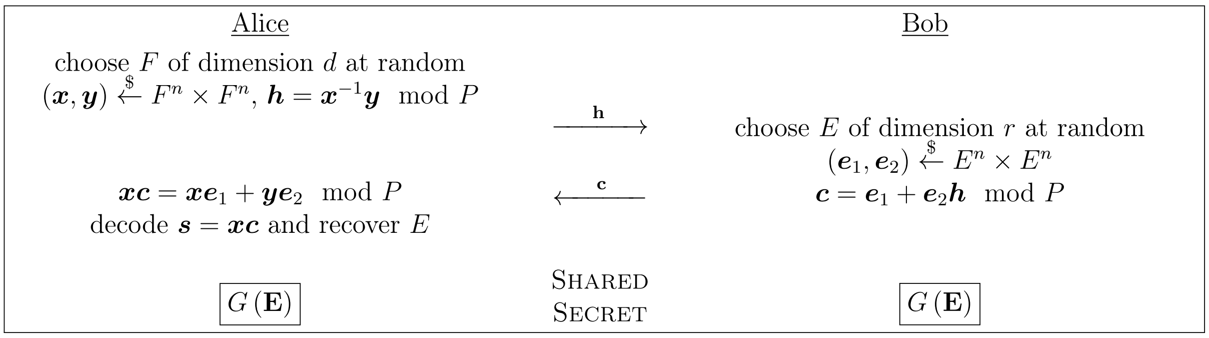
Alice
Bob
choose F of dimension d at random
$(\boldsymbol{x}, \boldsymbol{y}) \overset{\$}{\leftarrow} F^n \times F^n$, $\boldsymbol{h} = \boldsymbol{x}^{-1}\boldsymbol{y} \mod P$
$\xrightarrow{\mathbf{h}}$
choose E of dimension r at random
$(\boldsymbol{e}_1, \boldsymbol{e}_2) \overset{\$}{\leftarrow} E^n \times E^n$
$\boldsymbol{c} = \boldsymbol{e}_1 + \boldsymbol{e}_2\boldsymbol{h} \mod P$
$\xleftarrow{\mathbf{c}}$
$\boldsymbol{xc} = \boldsymbol{xe}_1 + \boldsymbol{ye}_2 \mod P$
decode $\boldsymbol{s} = \boldsymbol{xc}$ and recover E
$\boxed{G(\mathbf{E})}$
Shared Secret
$\boxed{G(\mathbf{E})}$

Figure 4.8 – Informal description of our new Key Encapsulation Mechanism. **h** constitutes the public key. The product is defined as a product of polynomials.

Definition 4.4.2 (IND-CPA Security)**.** *A key encapsulation scheme KEM is* IND-CPA-*secure if for every PPT (probabilistic polynomial time) adversary $\mathcal{A}$, we have that*

$$\mathsf{Adv}_{\mathsf{KEM}}^{indcpa}(\mathcal{A}) := |\Pr[\mathsf{IND\text{-}CPA}_{\mathsf{real}}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{IND\text{-}CPA}_{\mathsf{rand}}^{\mathcal{A}} \Rightarrow 1]|$$

is negligible.

Description of the scheme

Our scheme contains a hash function G modeled as a Random Oracle Model (ROM).

— $\mathsf{KeyGen}(1^\lambda)$:
  - — choose an irreducible polynomial $P \in \mathbb{F}_q[X]$ of degree n.
  - — choose uniformly at random a subspace F of $\mathbb{F}_{q^m}$ of dimension d and sample a couple of vectors $(\boldsymbol{x}, \boldsymbol{y}) \overset{\$}{\leftarrow} F^n \times F^n$ such that $\mathrm{Supp}(\boldsymbol{x}) = \mathrm{Supp}(\boldsymbol{y}) = F$.
  - — compute $\boldsymbol{h} = \boldsymbol{x}^{-1}\boldsymbol{y} \mod P$.
  - — define $\mathsf{pk} = (\boldsymbol{h}, P)$ and $\mathsf{sk} = (\boldsymbol{x}, \boldsymbol{y})$.

— $\mathsf{Encap}(\mathsf{pk})$:
  - — choose uniformly at random a subspace E of $\mathbb{F}_{q^m}$ of dimension r and sample a couple of vectors $(\boldsymbol{e}_1, \boldsymbol{e}_2) \overset{\$}{\leftarrow} E^n \times E^n$ such that $\mathrm{Supp}(\boldsymbol{e}_1) = \mathrm{Supp}(\boldsymbol{e}_2) = E$.
  - — compute $\boldsymbol{c} = \boldsymbol{e}_1 + \boldsymbol{e}_2\boldsymbol{h} \mod P$.
  - — define $K = G(E)$ and return $\boldsymbol{c}$.

— $\mathsf{Decap}(\mathsf{sk})$:
  - — compute $\boldsymbol{xc} = \boldsymbol{xe}_1 + \boldsymbol{ye}_2 \mod P$ and recover E with the RSR algorithm 4.6.
  - — recover $K = G(E)$.

We need to have a common representation of a subspace of dimension r of $\mathbb{F}_{q^m}$. The natural way is to choose the unique matrix $\boldsymbol{M} \in \mathbb{F}_q^{r \times m}$ of size $r \times m$ in its row echelon form such that the rows of $\boldsymbol{M}$ are a basis of E.

We deal with the semantic security of the KEM in Section 4.4.3.

4.4.2 An IND-CCA-2 PKE based on the rank metric

Our KEM can be slightly modified to become a PKE cryptosystem.

A Public Key Encryption (PKE) scheme is defined by three algorithms: the key generation algorithm KeyGen which takes on input the security parameter λ and outputs a pair of public and private keys (pk, sk); the encryption algorithm $\text{Enc}(pk, M)$ which outputs the ciphertext C corresponding to the message M and the decryption algorithm $\text{Dec}(sk, C)$ which outputs the plaintext M.

Our PKE scheme contains a hash function G modeled as a ROM.

— $\text{KeyGen}(1^\lambda)$:
  — choose an irreducible polynomial $P \in \mathbb{F}_q[X]$ of degree n.
  — choose uniformly at random a subspace F of $\mathbb{F}_{q^m}$ of dimension d and sample a couple of vectors $(\boldsymbol{x}, \boldsymbol{y}) \overset{\$}{\leftarrow} F^n \times F^n$ such that $\text{Supp}(\boldsymbol{x}) = \text{Supp}(\boldsymbol{y}) = F$.
  — compute $\boldsymbol{h} = \boldsymbol{x}^{-1}\boldsymbol{y} \mod P$.
  — define $pk = (\boldsymbol{h}, P)$ and $sk = (\boldsymbol{x}, \boldsymbol{y})$.

— $\text{Enc}(pk, M)$:
  — choose uniformly at random a subspace E of $\mathbb{F}_{q^m}$ of dimension r and sample a couple of vectors $(\boldsymbol{e}_1, \boldsymbol{e}_2) \overset{\$}{\leftarrow} E^n \times E^n$ such that $\text{Supp}(\boldsymbol{e}_1) = \text{Supp}(\boldsymbol{e}_2) = E$.
  — compute $\boldsymbol{c} = \boldsymbol{e}_1 + \boldsymbol{e}_2\boldsymbol{h} \mod P$.
  — output the ciphertext $C = (\boldsymbol{c}, M \oplus G(E))$.

— $\text{Dec}(sk, C)$:
  — compute $\boldsymbol{x}\boldsymbol{c} = \boldsymbol{x}\boldsymbol{e}_1 + \boldsymbol{y}\boldsymbol{e}_2 \mod P$ and recover E with the ideal RSR algorithm 4.6.
  — output $M = C \oplus G(E)$.

4.4.3 Security of our schemes

Theorem 4.4.3. *Under the* Ideal LRPC indistinguishability *4.4.1 and the* Ideal-Rank Support Recovery *2.3.11 Problems, the KEM presented above in section 4.4.1 is indistinguishable against Chosen Plaintext Attack in the Random Oracle Model.*

Proof. We are going to proceed in a sequence of games. The simulator first starts from the real scheme. First we replace the public key matrix by a random element, and then we use the ROM to solve the Ideal-Rank Support Recovery.

We start from the normal game G_0: We generate the public key $\boldsymbol{h}$ honestly, and $E, \boldsymbol{c}$ also

— In game G_1, we now replace $\boldsymbol{h}$ by a random vector, the rest is identical to the previous game. From an adversary point of view, the only difference is the distribution of $\boldsymbol{h}$, which is either generated at random, or as a product of low weight vectors. This is exactly the *Ideal LRPC indistinguishability* problem, hence

$$\mathsf{Adv}_{\mathcal{A}}^{G_0} \leq \mathsf{Adv}_{\mathcal{A}}^{G_1} + \mathsf{Adv}_{\mathcal{A}}^{\text{I-LRPC}}.$$

— In game G_2, we now proceed as earlier except we receive $\boldsymbol{h}, \boldsymbol{c}$ from a Support Recovery challenger. After sending $\boldsymbol{c}$ to the adversary, we monitor the adversary queries to the Random Oracle, and pick a random one that we forward as our simulator answer to the Ideal-Rank Support Recovery problem. Either the adversary was able to predict the random oracle output, or with probably $1/q_G$, we picked the query associated with the support E (by q_G we denote the number of queries to the random oracle G), hence

$$\mathsf{Adv}_{\mathcal{A}}^{G_1} \leq 2^{-\lambda} + 1/q_G \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathrm{I-RSR}}$$

which leads to the conclusion.

□

Theorem 4.4.4. *Under the* Ideal LRPC indistinguishability *4.4.1 and the* Ideal-Rank Support Recovery *2.3.11 Problems, the encryption scheme presented in section 4.4.2 is indistinguishable against Chosen Plaintext Attack in the Random Oracle Model.*

Proof. We are going to proceed in a sequence of games. The simulator first starts from the real scheme. First we replace the public key matrix by a random element, and then we use the ROM to solve the Ideal-Rank Support Recovery.

We start from the normal game G_0: We generate the public key $\boldsymbol{h}$ honestly, and $E, \boldsymbol{c}$ also

— In game G_1, we now replace $\boldsymbol{h}$ by a random vector, the rest is identical to the previous game. From an adversary point of view, the only difference is the distribution of $\boldsymbol{h}$, which is either generated at random, or as a product of low weight vectors. This is exactly the *Ideal LRPC indistinguishability* problem, hence

$$\mathsf{Adv}_{\mathcal{A}}^{G_0} \leq \mathsf{Adv}_{\mathcal{A}}^{G_1} + \mathsf{Adv}_{\mathcal{A}}^{\mathrm{I-LRPC}}.$$

— In game G_2, we now proceed as earlier except we replace $G(E)$ by random. It can be shown, that by monitoring the call to the ROM, the difference between this game and the previous one can be reduced to the Ideal-Rank Support Recovery problem, so that:

$$\mathsf{Adv}_{\mathcal{A}}^{G_1} \leq 2^{-\lambda} + 1/q_G \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathrm{I-RSR}}.$$

— In a final game G_3 we replace $C = M \oplus \mathsf{Rand}$ by just $C = \mathsf{Rand}$, which leads to the conclusion.

□

CCA-2 security proof

When applying the HHK [51] framework for the Fujisaki-Okamoto transformation, one can show that the final transformation is CCA-2 secure such that:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CCA-2}} \leq q_G \cdot \delta + q_V \cdot 2^{-\gamma} + \frac{2q_G + 1}{|\mathcal{M}|} + 3\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CPA}}$$

where q_G is the number of queries to the random oracle G and q_V is the number of verification queries.

As our scheme is CPA secure, the last term is negligible, we can handle exponentially large message space for a polynomial number of queries, so the previous is too.

As shown before, our scheme is gamma-spread so again for a polynomial number of verification queries, the term in q_V is negligible.

The tricky term remaining is $q_G \cdot \delta$, this is the product of the number of queries to the random oracle, by the probability of generating an decipherable ciphertext in an honest execution. For real applications, we want schemes to be correct enough so that the probability of such an occurrence is very small. This often leads, in application in running with a probability of a magnitude of 2^{-64}. This may seem not low enough for pure cryptographic security, however it should be noted that this number corresponds to the number of requests adversarially generated where the simulator gives an honest answer to a decryption query, which would mean that a single user would be able to do as many queries as expected by the whole targeted users in a live application, so a little trade-off at this level seems more than fair.

4.4.4 Best known attacks on RSD

The complexity of practical attacks grows very fast with the size of parameters, there is a structural reason for this: for Hamming distance a key notion for the attacks consists of counting the number of words of length n and support size t, which corresponds to the Newton binomial coefficient $\binom{n}{t}$, whose value is exponential and upper bounded by 2^n. In the rank metric case, counting the number of possible supports of size r for a rank code of length n over $\mathbb{F}_{q^m}$ corresponds to counting the number of subspaces of dimension r in $\mathbb{F}_{q^m}$: **the Gaussian binomial coefficient** of size roughly q^{rm}, whose value is also exponential but with a quadratic term in the exponent.

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 first non-trivial attack on the problem was proposed by Chabaud and Stern [24] in 1996. It was improved in 2002 by Ourivski and Johannson [62] who proposed a new attack in $\mathcal{O}\left((n-k)^3 m^3 q^{(w-1)(k+1)}\right)$. However, these two attacks did not take account of the value of m in the exponent. Very recently the two previous attacks were generalized in [41] (and used to break some repairs of the GPT cryposystems) moreover an algebraic new setting was also proposed, which gives an attack in $\mathcal{O}\left(w^3 k^3 q^{w\left\lceil \frac{(w+1)(k+1)-n-1}{w} \right\rceil}\right)$. Finally, a last improvement of the combinatorial attack of [41] was proposed this year in [14], for a complexity of $\mathcal{O}\left((n-k)^3 m^3 q^{w\left\lceil \frac{(k+1)m}{n} \right\rceil - m}\right)$.

Remark 4.4.5. *Since the linear system is not random, it is reasonable to take $\omega = 2$ for the choice of the parameters of ROLLO-I, ROLLO-II and ROLLO-III, even if the attack described in [15] takes $\omega = 3$.*

Let us remark that the choice of our parameter is flexible. We could take $\omega = 0$ and increase the parameters, which corresponds to only keeping the exponential complexity of the attack, for instance by slightly increasing m.

- **algebraic attacks**: the particular nature of the rank metric makes it a natural

Security	128	192	256
n	47	53	67
m	71	89	113
d	6	7	8
r	5	6	7
P	$X^{47}+X^5+1$	$X^{53}+X^6+X^2+X+1$	$X^{67}+X^5+X^2+X+1$
Structural Attack	130	207	312
Generic Attack	146	221	329
p_f	2^{-30}	2^{-32}	2^{-36}
Entropy	311	499	743
Public key (bits)	3337	4717	7571

Table 4.1 – Examples of parameters of our KEM 4.4.1

field for algebraic attacks and solving by Groebner bases, since these attacks are largely independent of the value of q and in some cases may also be largely independent of m. These attacks are usually the most efficient ones when q increases. There exist different types of algebraic equations settings to try to solve a multivariate system with Groebner bases. The algebraic context proposed by Levy and Perret [52] in 2006 considers a quadratic setting over $\mathbb{F}_q$ by taking as unknowns the support E of the error and the error coordinates regarding E. It is also possible to consider the Kernel attack by [31] and the minor approach [32] which give multivariate equations of degree $r+1$ over $\mathbb{F}_q$ obtained from minors of matrices Finally, the recent annulator setting by Gaborit et *al.* in [41] (which is valid on certain type of parameters but may not be independent on m) gives multivariate sparse equations of degree q^{r+1} but on the large field $\mathbb{F}_{q^m}$ rather than on the base field $\mathbb{F}_q$. The latter attack is based on the notion of q-polynomial [61] and is particularly efficient when r is small. Moreover, all these attacks can be declined in an hybrid approach where some unknowns are guessed. In practice (and for the case of this paper) for the known state of the art these attacks seem less efficient than combinatorial attacks. Moreover, with today's knowledge, these attacks cannot benefit from a quantum speed up, hence even if such attacks do a little better for classical security than 256 bits, it won't necessarily imply a difference for a quantum security of 128 bits.

Impact of quantum algorithms on the complexity of the attacks

In post-quantum cryptography, it is important to study the impact of quantum computers on the security of the schemes. In rank metric cryptography, the problems on which we are based are still difficult for a quantum computer, however there is a quantum speedup based on a generalization of Grover's quantum search algorithm for combinatorial attacks: the exponent of the complexity is divided by 2 [38], which lead to a complexity of $\mathcal{O}\left((n-k)^3m^3q^{\frac{w}{2}\lceil\frac{(k+1)m}{n}\rceil-m}\right)$.

Concerning the algebraic attacks, there is currently no quantum speedup as far as we know.

Security	128	192	256
n	83	83	89
m	71	101	107
d	7	7	8
r	5	5	6
P	$X^{83}+X^7+X^4+X^2+1$	$X^{83}+X^7+X^4+X^2+1$	$X^{89}+X^{38}+1$
Structural Attack	133	209	273
Generic Attack	144	195	260
p_f	2^{-64}	2^{-64}	2^{-64}
Entropy	331	481	607
Public key Size (bits)	5893	8383	9523

Table 4.2 – Example of parameters of our PKE 4.4.2, $p_f \leqslant 2^{-64}$

Security	128	192	256
n	101	103	103
m	79	97	107
d	7	8	8
r	5	6	6
P	$X^{101}+X^7+X^6+X+1$	$X^{103}+X^9+1$	$X^{103}+X^9+1$
Structural Attack	136	229	259
Generic Attack	157	234	260
p_f	2^{-80}	2^{-80}	2^{-80}
Entropy	371	547	607
Public key Size (bits)	7979	9991	11021

Table 4.3 – Example of parameters of our PKE 4.4.2, $p_f \leqslant 2^{-80}$

4.4.5 Parameters

In this section, we give some sets of parameters for a quantum security level of 64, 96 and 128, corresponding to classical level of security with respect to combinatorial attacks of 128, 192 and 256. In all cases, we have chosen $q = 2$. The choice of these parameters depends on the probability of decryption failure and the complexity of the attacks against our cryptosystem:

1. Message attacks: they consist in solving an instance of the I − RSR problem of weight r 2.3.11.
2. Attacks on the secret key: they consist to recover the structure of the Ideal-LRPC code by computing a codeword of weight d in the dual code 4.4.1.
3. Spurious key attack: as in the NTRU case (see [50]) this attack corresponds to finding small word vectors in the row space of $\boldsymbol{H}$ with rank slightly greater than d, and to use them for decoding. Although theoretically possible, this attack is not doable in practice since the fact that $\boldsymbol{H}$ contains small weight vectors implies that many words of weight $2d$ exist. We do not go into details in this article but as for MDPC codes [59], when the weight increases the complexity of the attacks grows faster than the number of small weight vectors, so that this attacks - as for NTRU and MDPC- does not work in practice.

The different parameters are :

— n is the length the vectors $\boldsymbol{h}$ and $\boldsymbol{c}$ sent on the public channel.

— m is the degree of the extension $\mathbb{F}_{2^m}$.

— d is the weight of the ideal LRPC code used in the protocol.

— r is the weight of the error.

— P is the irreducible polynomial of degree n of $\mathbb{F}[X]$ which defines the ideal LRPC code. We have chosen sparse polynomials in order to diminish the computation costs.

— the structural attack parameter is the complexity of the best attack to recover the structure of the ideal LRPC code. It consists in looking for a codeword of weight d in an ideal LRPC of type $[2n, n]_{2^m}$ defined by the parity-check matrix $(\boldsymbol{I}_n|\boldsymbol{H})$ where

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{h}(X) \mod P \\ X\boldsymbol{h}(X) \mod P \\ \vdots \\ X^{n-1}\boldsymbol{h}(X) \mod P \end{pmatrix}.$$

Because of the structure of the ideal LRPC, the complexity of looking for a codeword of weight d is easier than in a random code (see [7]). The complexity of the attack, for these parameters, is $\mathcal{O}\left((nm)^\omega 2^{d\lceil \frac{m}{2} \rceil - m - n}\right)$ where $\omega = \log_2 7 \approx 2.8$ is the cost of linear algebra.

— the generic attack parameter is the complexity of the best attack to recover the support E. It consists to solve the I-RSD problem for a random ideal code of type $[2n, n]_{2^m}$ and an error of weight r. For our parameters, the complexity of this attack is $\mathcal{O}\left((nm)^\omega 2^{r\lceil \frac{m(n+1)}{2n} \rceil - m}\right)$ where $\omega = \log_2 7 \approx 2.8$ is the cost of linear algebra.

— p_f is the probability of failure of the decoding algorithm. We have chosen the parameters such that the theoretic upper bound is below 2^{-26} for the KEM. For the PKE, it is necessary to have a lower probability of failure since the public key can be used to encrypt a high number of messages. We give two sets of parameters: the first one for a probability below 2^{-64} and the second one for a probability below 2^{-80}.

— the entropy parameter is the entropy of the subspace E. It is equal to $\log_2\left(\begin{bmatrix} r \\ m \end{bmatrix}\right)$ and has to be greater than the security parameter. We represent E by a matrix of size $r \times m$ in its row echelon form.

— the public key size is the number of bits needed to represent the public key $\boldsymbol{h} \in \mathbb{F}_{q^m}^n$. It is equal to mn.

Comparison with other cryptosystems: In terms of size of public key, the system compares very well with other proposals to the NIST competition. For 128 bits security key exchange (in which case one is allowed to a not too small decryption failure rate) the BIKE proposal (MDPC cryptosystem) has a public key of 10 kilo bits, hence the LRPC cryptosystem has public keys roughly three times smaller. When considering the PKE variant, the system is comparable to the RQC cryptosystem in terms of bandwidth, and far smaller than the McEliece cryptosystem. Globally among all candidates to the NIST competition, the LRPC cryptosystem has the second smaller size of public keys for 128 bits security, better than candidates based on lattices, but a little larger than systems based on isogenies. Moreover the system is also efficient in terms of speed compared to the other NIST proposals.

We also provide concrete timings of our implementations (given tables 4.4 and 4.5 for the KEM and tables 4.6 and 4.7 for the PKE). The benchmarks were performed on an Intel®Core™i7-4700HQ CPU running @ up to 3.40GHz and the software was compiled using GCC (version 6.3.0) with the following command : `g++ -O2 -pedantic -Wall -Wextra -Wno-vla`.

Security	KeyGen	Encap	Decap
128	0.65	0.13	0.53
192	0.73	0.13	0.88
256	0.77	0.15	1.24

Table 4.4 – Timings (in ms) of our KEM 4.4.1

Security	KeyGen	Encap	Decap
128	1.58	0.30	1.27
192	1.74	0.31	2.09
256	1.79	0.35	2.89

Table 4.5 – Timings (in millions of cycles) of our KEM 4.4.1

Security	p_f	KeyGen	Encap	Decap
128	64	1.09	0.22	1.04
192	64	1.33	0.23	1.08
256	64	1.51	0.25	1.58
128	80	1.51	0.29	1.16
192	80	1.82	0.36	1.80
256	80	1.94	0.31	1.68

Table 4.6 – Timings (in ms) of our PKE 4.4.2

4.5 Conclusion

In this chapter, as in the recent MDPC paper [59], we generalize the NTRU [50] approach in a coding context but with the rank metric. To do so we have introduced a new family of codes, LRPC codes, for which we propose an efficient decoding algorithm and we have carefully analyzed its failure probability. In order to reduce the public key size, we have chosen to use the special family of Ideal-LRPC codes.

Security	p_f	KeyGen	Encap	Decap
128	64	2.71	0.55	2.57
192	64	3.19	0.54	1.08
256	64	3.58	0.60	3.77
128	80	3.72	0.71	2.86
192	80	4.36	0.86	4.32
256	80	4.68	0.75	4.06

Table 4.7 – Timings (in millions of cycles) of our PKE 4.4.2

Overall, as it is often the case for rank metric codes, the obtained results compare well to Hamming distance cryptosystems since the difficulty of known attacks increases. Moreover, while rank metric cryptosystems have a strong history of broken schemes because of structural attacks based on recovering the Gabidulin code structure, the cryptosystems we propose are the first rank-metric based cryptosystems that is not based on based on Gabidulin codes but on codes with a weak structure and a strong random component.

Part III

Nouveaux schémas

Chapter 5

Durandal : un nouveau schéma de signature

This chapter introduces Durandal, a rank-metric based signature scheme. It is based upon the Schnorr-Lyubashevsky approach adapted to the rank metric, with an idea that allows to efficiently randomize and avoid information leakage in the signatures. We give a security proof in the EUF-CMA security model, reducing the security of the scheme to the Rank Support Learning (RSL) problem, the (ideal) Rank Syndrome Decoding (RSD) problem and a newly introduced problem, the Product Spaces Subspaces Indistinguishability (PSSI) problem for which we give a detailed analysis of a distinguisher. This approach is specifically tailored for the rank metric and there is no obvious way to adapt it to the Hamming metric, since it extensively uses the notion of support in the rank metric. We provide parameters with key sizes of order 20kB. This work is a joint work with Olivier Blazy, Philippe Gaborit, Adrien Hauteville and Gilles Zémor and was published to Eurocrypt 2019 [10].

Contents

5.1 Presentation of rank metric codes

Some rank metric definitions were introduced chapter 2. We also need to introduce the problems on which the security of this signature scheme relies.

The following problem was introduced in [36]. It is similar to the RSD problem, the difference is that instead of having one syndrome, we are given several syndromes of errors of same support and the goal is to find this support.

Problem 5.1.1. ***Rank Support Learning (RSL)*** *[37] Let $\boldsymbol{H}$ be a random full-rank $(n-k) \times n$ matrix over $\mathbb{F}_{q^m}$.*

Let $\mathcal{O}$ be an oracle which, given $\boldsymbol{H}$, gives samples of the form $\boldsymbol{H}\boldsymbol{s}_1^T, \boldsymbol{H}\boldsymbol{s}_2^T, \dots, \boldsymbol{H}\boldsymbol{s}_N^T$, with the vectors $\boldsymbol{s}_i$ randomly chosen from a space E^n, where E is a random subspace of $\mathbb{F}_{q^m}$ of dimension r. The $RSL_{q,m,n,k,r}$ problem is to recover E given only access to the oracle.

We denote $RSL_{q,m,n,k,r,N}$ the $RSL_{q,m,n,k,r}$ problem where we are allowed to make exactly N calls to the oracle, meaning we are given exactly N syndrome values $\boldsymbol{H}\boldsymbol{s}_i^T$. By an instance of the RSL problem, we shall mean a sequence

$$(\boldsymbol{H}, \boldsymbol{H}\boldsymbol{s}_1^T, \boldsymbol{H}\boldsymbol{s}_2^T, \dots, \boldsymbol{H}\boldsymbol{s}_N^T)$$

that we can also view as a pair of matrices $(\boldsymbol{H}, \boldsymbol{T})$, where $\boldsymbol{T}$ is the matrix whose columns are the $\boldsymbol{H}\boldsymbol{s}_i^T$.

The last problem we need before introducing our scheme is a variant of the RSD problem. Instead of searching for the error associated to a syndrome, this problem consists in finding an error associated to a syndrome which belongs to a given $\mathbb{F}_q$-affine subspace of $\mathbb{F}_{q^m}^{n-k}$. Formally:

Problem 5.1.2. ***Affine Rank Syndrome Decoding (*ARSD*)*** *Let $\boldsymbol{H}$ be an $(n-k) \times n$ parity-check matrix of an $[n,k]$ $\mathbb{F}_{q^m}$-linear code, $\boldsymbol{H}'$ an $(n-k) \times n'$ random matrix over $\mathbb{F}_{q^m}$, F an $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ of dimension r', $\boldsymbol{s} \in \mathbb{F}_{q^m}^{n-k}$ and r an integer. The $\mathrm{ARSD}_{q,m,n,k,r,n',F}$ problem is to find $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$ and $\boldsymbol{e}' \in \mathbb{F}_{q^m}^{n'}$ such that*

$$\begin{cases} \boldsymbol{H}\boldsymbol{e}^T + \boldsymbol{H}'\boldsymbol{e}'^T = \boldsymbol{s} \\ |\boldsymbol{e}|_r = r \\ \mathrm{Supp}(\boldsymbol{e}') \subseteq F \end{cases}$$

Remark: This problem can seen as that of finding a vector $\boldsymbol{x}$ of weight r such that $\boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{s}'$ with $\boldsymbol{s}' \in \{\boldsymbol{s} - \boldsymbol{H}'\boldsymbol{x}'^T : \mathrm{Supp}(\boldsymbol{x}') \subseteq F\}$. This set is an $\mathbb{F}_q$-affine subspace of $\mathbb{F}_{q^m}^{n-k}$, which explains the name of the problem.

The following proposition shows that the ARSD problem is as hard as the RSD problem for large values of m.

Proposition 5.1.3. *Let $\mathcal{A}$ be an algorithm which can solve the* $\mathrm{ARSD}_{q,m,n,k,r,n',F}$ *problem with* $m \geqslant \frac{r(n-r)+n' \dim F}{n-k-r}$*. Then $\mathcal{A}$ can be used to solve the* $\mathrm{RSD}_{q,m,n,k,r}$ *problem with non negligible probability.*

Proof. Let $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}, \boldsymbol{s} \in \mathbb{F}_{q^m}^{n-k}$ such that $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{e}^T$ with $|\boldsymbol{e}|_r = r$ be an instance of the $\mathrm{RSD}_{q,m,n,k,r}$ problem. First we need to transform this instance into an instance of the ARSD problem. Let $\boldsymbol{H}' \in \mathbb{F}_{q^m}^{(n-k)\times n'}$ and let F be a subspace of $\mathbb{F}_{q^m}$ of dimension r' such that $m \geqslant \frac{r(n-r)+n' \dim F}{n-k-r}$. Let $\boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{H}'\boldsymbol{e}'^T$ with $\mathrm{Supp}(\boldsymbol{e}') = F$.

$(\boldsymbol{H}, \boldsymbol{s}', r, \boldsymbol{H}', F)$ is an instance of the $\mathrm{ARSD}_{q,m,n,k,r,n',F}$ problem. Let $(\boldsymbol{x}, \boldsymbol{x}')$ be a solution of this instance given by algorithm $\mathcal{A}$. Now we will prove that this solution is unique with a non negligible probability.

Let us consider the application f defined by

$$\begin{aligned} f :& \mathcal{S}_{\mathbb{F}_{q^m}^n}(r) \times F^{n'} \to \mathbb{F}_{q^m}^{n-k} \\ & (\boldsymbol{x}, \boldsymbol{x}') \mapsto \boldsymbol{H}\boldsymbol{x}^T + \boldsymbol{H}'\boldsymbol{x}'^T \end{aligned}$$

where $\mathcal{S}_{\mathbb{F}_{q^m}^n}(r)$ is the set of words of $\mathbb{F}_{q^m}^n$ of rank r. By definition of the ARSD problem, we have $(\boldsymbol{x}, \boldsymbol{x}') \in f^{-1}(\{\boldsymbol{s}'\})$.

Let $S(\mathbb{F}_{q^m}^n, r)$ denote the cardinality of $\mathcal{S}_{\mathbb{F}_{q^m}^n}(r)$. By definition of the rank metric, $S(\mathbb{F}_{q^m}^n, r)$ is equal to the number of matrices of $\mathbb{F}_q^{m\times n}$ of rank r and we have

$$S(\mathbb{F}_{q^m}^n, r) = \prod_{i=0}^{r-1} \frac{(q^m - q^i)(q^n - q^i)}{q^r - q^i} = \Theta\left(q^{r(m+n-r)}\right).$$

Thus the cardinality of the domain of f is in $\Theta(q^{r(n+n-r)+n'r'})$ and the cardinality of its codomain is equal to $q^{m(n-k)}$. We have $m \geqslant \frac{r(n-r)+n'r'}{n-k-r}$ which implies $m(n-k) \geqslant r(m+n-r) + n'r'$, hence $\boldsymbol{s}'$ has only one preimage with a non negligible probability. Thus $\boldsymbol{H}\boldsymbol{x}^T + \boldsymbol{H}'\boldsymbol{x}'^T = \boldsymbol{s}' = \boldsymbol{H}\boldsymbol{e}^T + \boldsymbol{H}\boldsymbol{e}'^T$ implies $(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{e}, \boldsymbol{e}')$ so $\boldsymbol{x}$ is a solution of the instance of the $\mathrm{RSD}_{q,m,n,k,r}$ problem. □

Remark: All these problems are defined for random codes but can straightforwardly be specialized to the families of double circulant codes or of ideal random codes. In this case, these problems are denoted $\mathrm{I}-\mathrm{RSD}$, $\mathrm{I}-\mathrm{ARSD}$ and $\mathrm{I}-\mathrm{RSL}$ respectively. The reductions are unchanged, the only difference being that the $\mathrm{I}-\mathrm{RSD}$ problem is reduced to the Syndrome Decoding problem for ideal codes, which has not been proven NP-complete. However this problem is considered hard by the community since the best attacks stay exponential.

5.2 A new signature scheme based on the RSL problem

5.2.1 General overview

Our scheme consists of adapting to the rank metric the idea proposed in [56]. This idea can be viewed as a framework for an authentication scheme and can be loosely described as follows. Two matrices, over some fixed finite field, $\boldsymbol{H}$ and $\boldsymbol{T}$, are public, and a Prover

Prover		Verifier
$\boldsymbol{y} \overset{\$}{\leftarrow} \mathcal{D}_y$	$\xrightarrow{\boldsymbol{x}=\boldsymbol{H}\boldsymbol{y}^T}$	
	$\xleftarrow{\boldsymbol{c}}$	$\boldsymbol{c} \overset{\$}{\leftarrow} \mathcal{D}_c$
$\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}$	$\xrightarrow{\boldsymbol{z}}$	$\lvert\boldsymbol{z}\rvert_r \leqslant \lvert\boldsymbol{y}\rvert_r + \lvert\boldsymbol{c}\rvert_r \lvert\boldsymbol{S}\rvert_r$ *Check* $\boldsymbol{H}\boldsymbol{z}^T - \boldsymbol{T}\boldsymbol{c}^T = \boldsymbol{x}$

Figure 5.1 – Overview of the authentication framework from [56]

wishes to prove that she is in possession of secret matrix $\boldsymbol{S}$ with "small" entries such that $\boldsymbol{T} = \boldsymbol{H}\boldsymbol{S}^T$. She chooses a random vector $\boldsymbol{y}$ of small norm (to be defined appropriately) according to some distribution $\mathcal{D}_y$. She computes the syndrome $\boldsymbol{x} = \boldsymbol{H}\boldsymbol{y}^T$ of $\boldsymbol{y}$ and sends it to the verifier. The verifier chooses a random vector $\boldsymbol{c}$ of the appropriate length and of small norm according to some distribution $\mathcal{D}_c$ and sends it as a challenge to the Prover. The Prover computes $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}$ and sends it to the Verifier. The verifier checks that $\boldsymbol{z}$ is of small norm, and that

$$\boldsymbol{H}\boldsymbol{z}^T - \boldsymbol{T}\boldsymbol{c}^T = \boldsymbol{x}.$$

This scheme is described on Figure 5.1.

The general idea is that cheating is difficult for the Prover because it requires finding a vector $\boldsymbol{z}$ of small norm such that $\boldsymbol{H}\boldsymbol{z}^T$ equals a prescribed value, and this is an instance of the decoding problem for a random code. Also, the vector $\boldsymbol{z}$ sent by the legitimate Prover should yield no useful information on the secret $\boldsymbol{S}$, because the noisy random vector $\boldsymbol{y}$ drowns out the sensitive quantity $\boldsymbol{c}\boldsymbol{S}$.

If we instantiate this scheme in the rank metric, $\boldsymbol{H}$ would be a random matrix over $\mathbb{F}_{q^m}$, and for $\boldsymbol{S}$ to be a matrix of small norm would mean it to be homogeneous matrix of some small rank r (i.e all of the coodinates of $\boldsymbol{S}$ are included in a subspace of $\mathbb{F}_{q^m}$ of dimension r). Requiring that the vectors $\boldsymbol{y}$ and $\boldsymbol{c}$ are also small will mean that their entries are taken in random subspaces of $\mathbb{F}_{q^m}$ of fixed dimensions respectively w and d.

The problem with this approach in the rank metric is that adding $\boldsymbol{y}$ to $\boldsymbol{c}\boldsymbol{S}$ does not hide $\boldsymbol{c}\boldsymbol{S}$ properly. Indeed, the verifier, or any witness to the protocol of Figure 5.1, can recover the support of the secret matrix $\boldsymbol{S}$ even after a single instance of the protocol, using techniques from the decoding of LRPC codes [39]: since the verifier has $\boldsymbol{c}$ he can choose a basis $f_1, \dots, f_d$ of $\mathrm{Supp}(\boldsymbol{c})$ and then with high probability it will occur that:

$$\bigcap_{i=1}^{d} f_i^{-1} \,\mathrm{Supp}(\boldsymbol{z}) = \mathrm{Supp}(\boldsymbol{S})$$

and with the support of $\boldsymbol{S}$ the verifier can compute $\boldsymbol{S}$ explicitely from the linear equations $\boldsymbol{H}\boldsymbol{S}^T = T$.

To tackle this problem, we will modify the protocol of Figure 5.1 by adding an other term to $\boldsymbol{z}$.

Prover		Verifier
$W \xleftarrow{\$} \mathbf{Gr}(w, \mathbb{F}_{q^m}),\ F \xleftarrow{\$} \mathbf{Gr}(d, \mathbb{F}_{q^m})$		
$\boldsymbol{y} \xleftarrow{\$} (W + EF)^n$	$\xrightarrow{\boldsymbol{x}=\boldsymbol{H}\boldsymbol{y}^T,\ F}$	
	$\xleftarrow{\boldsymbol{c}}$	$\boldsymbol{c} \xleftarrow{\$} F^{l'k}$
$\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}' + \boldsymbol{p}\boldsymbol{S}$	$\xrightarrow{\boldsymbol{z},\boldsymbol{p}}$	$\lvert\boldsymbol{z}\rvert_r \leqslant w + rd - \lambda$
		Check $\boldsymbol{H}\boldsymbol{z}^T - \boldsymbol{T}'\boldsymbol{c}^T + \boldsymbol{T}\boldsymbol{p}^T = \boldsymbol{x}$

Figure 5.2 – Overview as an authentication scheme

5.2.2 An authentication scheme

We will first describe our scheme as an authentication scheme. It calls upon the notion of product of $\mathbb{F}_q$-linear subspaces of $\mathbb{F}_{q^m}$. The Grassmannian $\mathbf{Gr}(k, \mathbb{F}_{q^m})$ represents the set of all subspaces of $\mathbb{F}_{q^m}$ of dimension k.

The public key consists of a random $(n-k) \times n$ matrix $\boldsymbol{H}$ over $\mathbb{F}_{q^m}$ and two matrices $\boldsymbol{T}$ and $\boldsymbol{T}'$, of size $(n-k) \times lk$ and $(n-k) \times l'k$ respectively, and such that $(\boldsymbol{H}, \boldsymbol{T}|\boldsymbol{T}')$ is an instance of the RSL problem, where $|$ denotes matrix concatenation. The private key consist of two homogeneous matrices $\boldsymbol{S}$ and $\boldsymbol{S}'$ of weight r such that $\boldsymbol{H}\boldsymbol{S}^T = \boldsymbol{T}$ and $\boldsymbol{H}\boldsymbol{S}'^T = \boldsymbol{T}'$. Accordingly, $\boldsymbol{S}$ and $\boldsymbol{S}'$ are $lk \times n$ and $l'k \times n$ matrices respectively. We denote by E the vector space spanned by the coordinates of $\boldsymbol{S}$ and $\boldsymbol{S}'$.

In the commitment step, we sample uniformly at random two vector spaces : $W \in \mathbf{Gr}(w, \mathbb{F}_{q^m})$ and $F \in \mathbf{Gr}(d, \mathbb{F}_{q^m})$. We then randomly choose $\boldsymbol{y} \in (W + EF)^n$. This vector will be used to mask the secret information in answer to the challenge. The commitment consists of $\boldsymbol{x} = \boldsymbol{H}\boldsymbol{y}^T$ together with the subspace F.

The verifier then chooses a challenge $\boldsymbol{c} \in F^{l'k}$.

To answer the challenge, the prover first computes $\boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}'$. Since the entries of the vector $\boldsymbol{c}$ are in F and the entries of the matrix $\boldsymbol{S}'$ are in E, we have that $\boldsymbol{c}\boldsymbol{S}'$ has its entries in the product space EF, and the vector $\boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}'$ has its entries in the space $W + EF$, like the vector $\boldsymbol{y}$. The linear span of the coordinates of $\boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}'$ is typically equal, or very close to $W + EF$, and this yields too much information on the secret space E to be given to the verifier. To counter this, we add a vector $\boldsymbol{p}\boldsymbol{S}$. Coordinates of $\boldsymbol{p}$ are chosen in F, so that the coordinates of $\boldsymbol{p}\boldsymbol{S}$ fall in the product space EF, and through linear algebra the prover chooses $\boldsymbol{p}$ such that the the linear span of the coordinates of the sum $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}' + \boldsymbol{p}\boldsymbol{S}$ is restricted to a smaller subspace: namely a subspace $W + U$ for U some subspace of EF of codimension λ inside EF. In other words, $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}' + \boldsymbol{p}\boldsymbol{S}$ is computed so as to be of rank at most $w + rd - \lambda$. The vector $\boldsymbol{z}$ is then sent to the verifier, together with the vector $\boldsymbol{p}$. This operation is at the heart of the present protocol and the derived signature scheme. More details are given about this in the following section and in Section 5.3.1.

The verifier accepts if $\lvert\boldsymbol{z}\rvert_r \leqslant rd + w - \lambda$ and $\boldsymbol{H}\boldsymbol{z}^T - \boldsymbol{T}'\boldsymbol{c}^T + \boldsymbol{T}\boldsymbol{p}^T = \boldsymbol{x}$. An overview of this protocol is given in Figure 5.2.

Using the Fiat-Shamir heuristic, we turn this authentication scheme into a signature scheme.

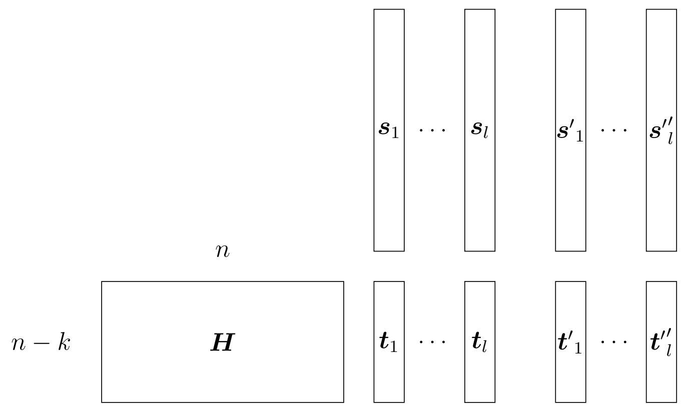

Figure 5.3 – Overview of public and secret key. $\boldsymbol{s}_i$ is a line of $\boldsymbol{S}$.

5.2.3 Signature scheme

Key generation

— Randomly choose an $(n-k) \times n$ ideal double circulant matrix $\boldsymbol{H}$ as in definition 2.3.5 for an irreducible polynomial P, in practice we consider $k = \frac{n}{2}$

— Choose a random subspace E of dimension r of $\mathbb{F}_{q^m}$ and sample l vectors $\boldsymbol{s}_i$ and l' vectors $\boldsymbol{s}'_i$ of length n from the same support E of dimension r

— Set $\boldsymbol{t}_i = \boldsymbol{H}\boldsymbol{s}_i^T$ and $\boldsymbol{t'}_i = \boldsymbol{H}\boldsymbol{s'}_i^T$

— Output $(\boldsymbol{H}, \boldsymbol{t}_1, \dots, \boldsymbol{t}_l, \boldsymbol{t'}_1, \dots, \boldsymbol{t'}_{l'})$ as public key, and $(\boldsymbol{s}_1, \dots, \boldsymbol{s}_l, \boldsymbol{s'}_1, \dots, \boldsymbol{s'}_{l'})$ as secret key

Note that, since $\boldsymbol{H}$ has an ideal structure, each relation of the form $\boldsymbol{H}\boldsymbol{s}_i^T = \boldsymbol{t}_i$ can be shifted $\mod P$ to generate k syndrome relations. We denote $\boldsymbol{S}$ (respectively $\boldsymbol{S}'$) the matrix consisting of all $\boldsymbol{s}_i$ (respectively $\boldsymbol{s'}_i$) and their ideal shifts. Let $\boldsymbol{T} = \boldsymbol{H}\boldsymbol{S}^T$ and $\boldsymbol{T}' = \boldsymbol{H}\boldsymbol{S}'^T$: the public key consists of $(\boldsymbol{H}, \boldsymbol{T}, \boldsymbol{T}')$. $\boldsymbol{T}$ and $\boldsymbol{T}'$ are $\frac{n}{2} \times lk$ and $\frac{n}{2} \times l'k$ matrices respectively, but can be described using only the vectors $(\boldsymbol{t}_1, \dots, \boldsymbol{t}_l)$ and $(\boldsymbol{t'}_1, \dots, \boldsymbol{t'}_{l'})$. The secret key consists of the homogeneous matrices $\boldsymbol{S}$ and $\boldsymbol{S}'$ of rank r such that $\boldsymbol{H}\boldsymbol{S}^T = \boldsymbol{T}$ and $\boldsymbol{H}\boldsymbol{S}'^T = \boldsymbol{T}'$.

Figure 5.3 describes the key pair.

Signature of a message μ

— Randomly choose W, a subspace of $\mathbb{F}_{q^m}$ of dimension w.

— Randomly choose F, a subspace of $\mathbb{F}_{q^m}$ of dimension d.

— Sample $\boldsymbol{y} \in (W + EF)^n$ and set $\boldsymbol{x} = \boldsymbol{H}\boldsymbol{y}^T$.

— For some hash function $\mathcal{H}$, set $\boldsymbol{c} = \mathcal{H}(\boldsymbol{x}, F, \mu)$, $\boldsymbol{c} \in F^{l'k}$. This is done by using the output of $\mathcal{H}$ as the coordinates of $\boldsymbol{c}$ in a basis of F.

— Choose U, a subspace of the product space EF, of dimension $rd - \lambda$, and such that U contains no non-zero elements of the form ef, for $e \in E$ and $f \in F$. More details on this process are given in subsection 5.2.4.

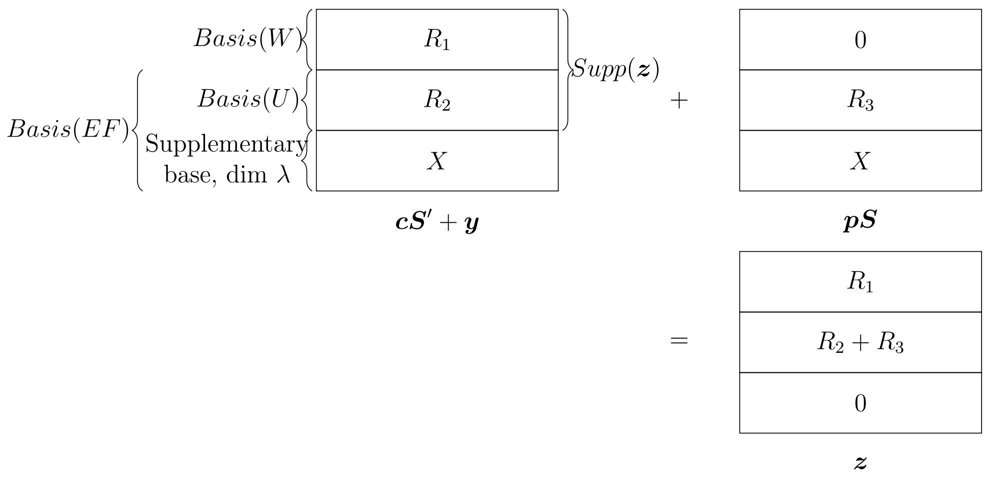

Figure 5.4 – Construction of $\boldsymbol{z}$. R_1, R_2, R_3 are random.

— Solve $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{cS}' + \boldsymbol{pS}$ with $\boldsymbol{p} \in F^{lk}$ as unknown, such that $\mathrm{Supp}(\boldsymbol{z}) \subset W + U$: as mentioned in the previous section, $\boldsymbol{p}$ is computed through linear algebra. Specifically, we write $\boldsymbol{p} = (p_1, \dots, p_{lk})$, and each coordinate $p_i \in F$ of $\boldsymbol{p}$ is decomposed as

$$p_i = \sum_{\ell=1}^{d} p_{i\ell} f_\ell$$

where $f_1, \dots, f_d$ is a basis of F that will be used to describe the space F. The j-th coordinate of the vector $\boldsymbol{cS}$ is therefore equal to

$$(\boldsymbol{pS})_j = \sum_{i=1}^{lk} \sum_{\ell=1}^{d} p_{i\ell} f_\ell \boldsymbol{S}_{ij}. \tag{5.1}$$

Recall that $f_\ell \boldsymbol{S}_{ij}$ is in FE because $\boldsymbol{S}$ has support E. Choose a basis of EF of the form $u_1, \dots, u_{rd-\lambda}, v_1, \dots, v_\lambda$, where $u_1, \dots, u_{rd-\lambda}$ is a basis of U (the typical dimension of EF is rd). Let $\pi_1, \dots, \pi_\lambda$ be the projections of elements of EF on the last λ coordinates of the above basis. For $h = 1 \dots \lambda$, applying π_h to all n coordinates of the vector $\boldsymbol{y} + \boldsymbol{cS}' + \boldsymbol{pS}$ and declaring the result equal to 0, yields a linear system of λn equations in the variables p_{ij} by using linearity in (5.1) to express $\pi_h[(\boldsymbol{pS})_j]$ as

$$\sum_{i=1}^{lk} \sum_{\ell=1}^{d} p_{i\ell} \pi_h(f_\ell \boldsymbol{S}_{ij}). \tag{5.2}$$

Parameters are chosen so that this system has more variables than equations and typically has a solution. If it doesn't, another space U is sampled.

Figure 5.4 gives a schematic view of how $\boldsymbol{z} \in W + U$ is obtained.

— Output $(\boldsymbol{z}, F, \boldsymbol{c}, \boldsymbol{p})$ as signature.

The signature consists therefore of the challenge $\boldsymbol{c}$, computed through a hash function, together with the answer to this challenge.

Verification of a signature $(\mu, \boldsymbol{z}, F, \boldsymbol{c}, \boldsymbol{p})$

Key generation: $E \overset{\$}{\leftarrow} \mathbf{Gr}(r, \mathbb{F}_{q^m})$
Signing key: $\boldsymbol{S} \overset{\$}{\leftarrow} E^{n \times lk}$, $\boldsymbol{S}' \overset{\$}{\leftarrow} E^{n \times l'k}$
Verification key: $\boldsymbol{H} \overset{\$}{\leftarrow}$ ideal $\frac{n}{2} \times n$ matrix, $\boldsymbol{T} = \boldsymbol{H}\boldsymbol{S}^T, \boldsymbol{T}' = \boldsymbol{H}\boldsymbol{S}'^T$

Sign($\mu, \boldsymbol{S}, \boldsymbol{S}'$):

```
1. $W \overset{\$}{\leftarrow} \mathbf{Gr}(w, \mathbb{F}_{q^m})$,
   $F \overset{\$}{\leftarrow} \mathbf{Gr}(d, \mathbb{F}_{q^m})$
2. $\boldsymbol{y} \overset{\$}{\leftarrow} (W + EF)^n$, $\boldsymbol{x} = \boldsymbol{H}\boldsymbol{y}^T$
3. $\boldsymbol{c} = \mathcal{H}(\boldsymbol{x}, F, \mu)$, $\boldsymbol{c} \in F^{l'k}$
4. $U \overset{\$}{\leftarrow}$ filtered subspace of $EF$ of dimension $rd - \lambda$
5. $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}' + \boldsymbol{p}\boldsymbol{S}$, $\boldsymbol{z} \in W + U$
6. Output $(\boldsymbol{z}, F, \boldsymbol{c}, \boldsymbol{p})$
```

Verify($\mu, \boldsymbol{z}, F, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{H}, \boldsymbol{T}, \boldsymbol{T}'$):

```
1. Accept if and only if :
   $|\boldsymbol{z}|_r \leqslant w + rd - \lambda$ and
   $\mathcal{H}(\boldsymbol{H}\boldsymbol{z}^T - \boldsymbol{T}'\boldsymbol{c}^T + \boldsymbol{T}\boldsymbol{p}^T, F, \mu) = \boldsymbol{c}$
```

Figure 5.5 – The Durandal Signature scheme

— Check that $|\boldsymbol{z}|_r \leqslant rd + w - \lambda$,
— Verify that $\mathcal{H}(\boldsymbol{H}\boldsymbol{z}^T - \boldsymbol{T}'\boldsymbol{c}^T + \boldsymbol{T}\boldsymbol{p}^T, F, \mu) = \boldsymbol{c}$.

To verify the signature, we have to check the rank weight of $\boldsymbol{z}$ and that $\mathcal{H}(\boldsymbol{x}, F, \mu) = \boldsymbol{c}$. The vector $\boldsymbol{x}$ is recomputed using the answer to the challenge. Our signature scheme is summarized on Figure 5.5.

5.2.4 Filtering vector spaces

The goal of filtering U during the signature step is to ensure to there is no non-zero element of the form ef in the support of $\boldsymbol{z}$, for $e \in E$ and $f \in F$. This is to prevent an attack that would recover E through techniques for decoding LRPC codes [39]. Indeed, if there is an element of the form ef in Supp($\boldsymbol{z}$), then $e \in E \cap f^{-1}\,\text{Supp}(\boldsymbol{z})$ which allows an attack against the secret key (moreover elements of this form can be used to distinguish between signatures and randomly generated vectors, as explained in 5.3.1). To achieve that, we need to find a pair (U, F) such that:

— U is a subspace of EF of dimension $rd - \lambda$
— For every non-zero $x = ef$ with $e \in E$ and $f \in F$, we have that $x \notin U$.

We argue that, for a given F, finding the required space U is quite manageable. We use the following obvious proposition to check the second condition:

Proposition 5.2.1. *Let U be a subspace of EF of dimension $rd - \lambda$. Let $E/\mathbb{F}_q$ be a set of representatives of the equivalence relation $x \equiv y \iff \exists \alpha \in \mathbb{F}_q^*$ such that $x = \alpha y$. We* *have the following equivalence:*

$$\{ef : e \in E, f \in F\} \cap U = \{0\} \iff \forall e \in E/\mathbb{F}_q, eF \cap U = \{0\}$$

Hence, the cost of this verification is $(q^r-1)/(q-1)$ intersections of subspaces of dimension d and $rd-\lambda$, that is to say $\frac{q^r-1}{q-1} \times (d+rd-\lambda)^2 m$ operations in $\mathbb{F}_q$ ($(d+rd-\lambda)^2 m$ is the cost of performing a Gaussian elimination on a $(d+rd-\lambda)\times m$ matrix with coefficients in $\mathbb{F}_q$).

We now briefly estimate the probability that a random U contains no element $x = ef$. For simplicity, we only deal with a typical practical case, namely $q = 2$ and $d = r$.

The subspace U is chosen randomly and uniformly of codimension λ inside the vector space EF: we study the probability that U contains no non-zero product ef. Let $x = ef$ be such a non-zero product. Let $\mathcal{U}_x$ be the event $\{x \in U\}$. We are interested in $1-\text{Prob}\,\mathcal{U}$ where

$$\mathcal{U} = \bigcup_{x=ef,\, x\neq 0} \mathcal{U}_x.$$

Clearly,

$$\text{Prob}\,\mathcal{U}_x = 2^{-\lambda}.$$

Our goal is to evaluate $\text{Prob}\,\mathcal{U}$ through inclusion-exclusion, i.e.

$$\text{Prob}\,\mathcal{U} = \sum_x \text{Prob}\,\mathcal{U}_x - \sum_{x,y} \text{Prob}\,\mathcal{U}_x \cap \mathcal{U}_y + \cdots + (-1)^i \sum_{X\in\Pi, |X|=i} \text{Prob} \bigcap_{x\in X} \mathcal{U}_x + \cdots \quad (5.3)$$

where Π denotes the set of non-zero elements of EF of the form $x = ef$. We have $|\Pi| = (2^r-1)^2$. Note that whenever X is made up of linearly independent elements, then the events $\mathcal{U}_x, x \in X$ are independent in the sense of probability, so that

$$\text{Prob} \bigcap_{x\in X} \mathcal{U}_x = 2^{-\lambda|X|}.$$

More generally, since any linear combination of vectors that are in U is also in U, we have

$$\text{Prob} \bigcap_{x\in X} \mathcal{U}_x = 2^{-\lambda \text{rk}(X)}$$

where $\text{rk}(X)$ denotes the rank of X.

For $\lambda = 2r-1$, tedious computations show that the contribution of the non full-rank subsets X for a growing (with r) set of first terms of (5.3) is negligible, so that we have

$$\text{Prob}\,\mathcal{U} \approx 2 - \frac{4}{2!} + \frac{8}{3!} + \cdots \approx 1 - e^{-2}$$

Giving $1 - \text{Prob}\,\mathcal{U} \approx e^{-2}$.

5.2.5 Value of λ

In order to find U that contains no element $x = ef$, we need to take the highest possible value for λ. We denote $\boldsymbol{z}_1 = \boldsymbol{y} + \boldsymbol{c}\boldsymbol{S}'$. When $\boldsymbol{z}_1$ is written as a $rd \times n$ matrix over $\mathbb{F}_q$ by rewriting each of its coordinates in a basis of EF of the form $\{u_1, \dots, u_{rd-\lambda}, v_1, \dots v_\lambda\}$ such that $U = \{u_1, \dots, u_{rd-\lambda}\}$, we want $\boldsymbol{pS}$ to be equal to z_1 on the last λ lines, corresponding to $\{v_1, \dots v_\lambda\}$. This gives λn equations in the base field, and the system has dlk unknowns (the coordinates of $\boldsymbol{p}$ in a basis of F). This gives the following condition on λ :

$$\lambda n < dlk \Leftrightarrow \lambda < \frac{dlk}{n}$$

Since we want to maximize the value of λ, we take $\lambda = \lfloor \frac{dlk}{n} \rfloor$.

5.2.6 Computational cost

Key generation

The most costly operation of the key generation step is the multiplication of $\boldsymbol{H}$ and the syndromes $\boldsymbol{s}_i$. Each matrix-vector multiplication costs n^2 multiplications in $\mathbb{F}_{q^m}$, hence a total cost of $(l + l')n^2$ multiplications.

Signature of a message μ

The signature step splits naturally in two phases: an offline phase during which the signature support is computed (this is the most costly part) and an online phase to compute the actual signature. The two phases are as follows:

1. Offline phase
   — Choose the vector spaces W and F.
   — Sample $\boldsymbol{y} \in (W + EF)^n$ and set $\boldsymbol{x} = \boldsymbol{H}\boldsymbol{y}^T$.
   — Choose U, a random subspace of EF of dimension $rd - \lambda$. If U contains non-zero elements of the form ef, $e \in E$ and $f \in F$, choose another U.
   — Write the $\mathbb{F}_{q^m}$-coordinates of the vector $\boldsymbol{pS}$ in a basis of EF of the form $\{u_1, \dots, u_{rd-\lambda}, v_1, \dots, v_\lambda\}$ where $U = \langle u_1, \dots, u_{rd-\lambda}\rangle$ to obtain linear expressions in the variables p_{ij} of the form (5.2). Compute a $\lambda n \times \lambda n$ matrix $\boldsymbol{D}$ that inverts this linear mapping of the p_{ij}. This will allow to compute $\boldsymbol{p}$ such that $\boldsymbol{z} \in U$ in the online phase with a matrix multiplication instead of an inversion. If the linear map cannot be inverted to produce the matrix $\boldsymbol{D}$, choose another random subspace U of EF.
2. Online phase
   — Set $\boldsymbol{c} = \mathcal{H}(\boldsymbol{x}, F, \mu)$, $\boldsymbol{c} \in F^{l'k}$
   — Solve $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{cS}' + \boldsymbol{pS}$ with $\boldsymbol{p} \in F^{lk}$, using the matrix $\boldsymbol{D}$ computed during the offline phase
   — Output $(\boldsymbol{z}, F, \boldsymbol{c}, \boldsymbol{p})$ as signature.

The most costly step in the offline phase is the computation of the matrix $\boldsymbol{D}$, which requires inverting a linear system over $\mathbb{F}_q$ with λn equations, hence the cost is $(\lambda n)^3$ multiplications in $\mathbb{F}_q$.

The online phase consists in the computation of $\boldsymbol{p}$ which costs $(\lambda n)^2$ multiplications in $\mathbb{F}_q$ as well as the computation of $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{cS}' + \boldsymbol{pS}$ which costs $(l'k)^2 + (lk)^2$ multiplications in $\mathbb{F}_{q^m}$ for computing the matrix/vector products.

Verification of a signature

The most costly step during the verification phase is the computation of $\boldsymbol{H}\boldsymbol{z}^T - \boldsymbol{T}'\boldsymbol{c}^T + \boldsymbol{T}\boldsymbol{p}^T$, which costs $n^2 + (l'k)^2 + (lk)^2$ multiplications in $\mathbb{F}_{q^m}$.

5.3 Security of the scheme

5.3.1 Product Spaces Subspaces Indistinguishability (PSSI)

The PSSI problem is a new problem which appears naturally when we try to prove the indistinguishability of the signatures.

Problem 5.3.1. *Product Spaces Subspaces Indistinguishability. Let E be a fixed $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ of dimension r. Let F_i, U_i and W_i be subspaces defined as follow:*

— $F_i \overset{\$}{\leftarrow} \mathbf{Gr}(d, \mathbb{F}_{q^m})$

— $U_i \overset{\$}{\leftarrow} \mathbf{Gr}(rd - \lambda, EF_i)$ *such that* $\{ef, e \in E, f \in F\} \cap U_i = \{0\}$

— $W_i \overset{\$}{\leftarrow} \mathbf{Gr}(w, \mathbb{F}_{q^m})$

The $\mathrm{PSSI}_{r,d,\lambda,w,\mathbb{F}_{q^m}}$ *problem consists in distinguishing samples of the form $(\boldsymbol{z}_i, F_i)$ where $\boldsymbol{z}_i$ is a random vector of $\mathbb{F}_{q^m}^n$ of support $W_i + U_i$ from samples of the form $(\boldsymbol{z}'_i, F_i)$ where $\boldsymbol{z}'_i$ is a random vector of $\mathbb{F}_{q^m}^n$ of weight $w + rd - \lambda$.*

In order to study the complexity of this problem, we first reduce it to the case where the samples are of the form (Z_i, F_i) with $Z_i = \mathrm{Supp}(\boldsymbol{z}_i)$. Let us suppose we have a distinguisher D for this last case. Then given N samples of the PSSI problem, it is easy to compute the supports Z_i of the vectors $\boldsymbol{z}_i$ and to use D to distinguish if Z_i is a random subspace of dimension $w + rd - \lambda$ or if it is of the form $W_i + U_i$ with U_i a subspace of the product space EF_i.

Conversely, let us suppose we have a distinguisher D' for the PSSI problem. We are given N samples of the form (Z_i, F_i). For each sample, we can compute a random vector $\boldsymbol{z}_i$ of support Z_i and use D' to distinguish whether $\boldsymbol{z}_i$ is a random vector of weight $w+rd-\lambda$ or if its support is of the form $W_i + U_i$.

Thus we can consider the case when the samples are only couples of subspaces of $\mathbb{F}_{q^m}$.

This problem is related to the decoding of LRPC codes [39]. Indeed we can consider a subspace $Z = U + W$ as the noisy support of a syndrome for an LRPC code, the noise corresponding to W. Consequently, it is natural to try and apply techniques used for decoding LRPC codes in order to solve the PSSI problem. The first idea is to use the basic decoding algorithm (see [39]). It consists in computing intersections I of the form $f^{-1}Z \cap f'^{-1}Z$ with $(f, f') \in F^2$. If Z is of the form $U + W$ then the probability that $\dim I \neq 0$ is much higher than if Z were truly random. However, this technique cannot be used because we filter the subspace U.

The decoding algorithm for LRPC codes has been improved in [12]. The idea is to consider product spaces of the form ZF_i where F_i is a subspace of F of dimension 2. The probability that $\dim ZF_i = 2(w + rd - \lambda)$ depends on whether Z is random or not. We study in detail the advantage of this distinguisher in the following paragraphs.

Consider the product subspace EF inside $\mathbb{F}_{q^m}$ with $\dim F = \dim E = r$. Suppose E is unknown, the typical dimension of the product EF is then r^2, if we assume $r^2 \ll m$.

We now suppose we are given a subspace Q of $\mathbb{F}_{q^m}$ of dimension r^2 that is either a product space EF or a randomly chosen one, and we wish to distinguish between the two events. One easy way to do so, if the dimension m of the ambient space $\mathbb{F}_{q^m}$ is large enough, is to multiply Q by F. If Q is random, we will get the typical product dimension $\dim FQ = r \dim Q = r^3$. While if $Q = EF$, we will get $\dim FQ \leq \binom{r+1}{2} r < r^3$. In fact, to distinguish the two cases it is enough to multiply Q by any subspace A of F of dimension 2, since we will have $\dim AQ \leq 2r^2 - r$ when $Q = EF$ and $\dim AQ = 2r^2$ in the typical random Q case.

To make our two cases difficult to distinguish, our query space Q is actually chosen to be constructed in one of two ways, making up a *distinguishing problem*:

Distinguishing problem. Distinguish whether Q is of the form (i) or (ii) below:

(i) $Q = U + W$ where U is a subspace of EF of codimension λ. The space E is chosen randomly of dimension r as before. The subspace U is chosen in such a way that, for any subspace A of F of dimension 2, we have $\dim AU = 2 \dim U$. The space W is chosen randomly of dimension w, so that $\dim Q = r^2 - \lambda + w$.

(ii) Q is a random subspace of dimension $r^2 - \lambda + w$. Equivalently we may think of Q of the form $Q = V + W$ where both V and W are random (and independent) of dimensions $r^2 - \lambda$ and w respectively.

The purpose of choosing such a subspace U of EF is to make the dimension of AU equal to that of AV for a random V. Adding the random space W to U should keep the probability distributions of $\dim AQ$ equal for both ways of construction of Q. The purpose of W is to make the dimension of Q sufficiently large with respect to the dimension m of the ambient space, so that multiplying Q by a space of dimension more than 2 will typically fill up the whole space $\mathbb{F}_{q^m}$ anyway. In this manner, the two ways of constructing Q will be indistinguishable by measuring dimensions of the product of Q by a subspace.

First, we give a criterion for a subspace U of EF to have the property that $\dim AU = 2 \dim U$ for any subspace A of dimension 2 of F.

Let E, F be two subspaces of $\mathbb{F}_{q^m}$, both of dimension r over $\mathbb{F}_q$. Let us make the remark that the maximum possible dimension of $F^{\langle 2 \rangle}$ is $\binom{r+1}{2}$, and the maximum possible dimension of $F^{\langle 2 \rangle}E$ is therefore $r\binom{r+1}{2}$.

Let $f_1, f_2, \ldots, f_r$ be a basis of F. Denote by F_2 the subspace of F generated by f_1, f_2, by F_3 the subspace of F generated by f_1, f_2, f_3, and so on.

Lemma 5.3.2. *Suppose* $\dim F^{\langle 2 \rangle}E = r\binom{r+1}{2}$. *Then* $f_1FE \cap f_2FE = f_1f_2E$.

Proof. We have $F_2FE = f_1FE + f_2FE$, and $f_1FE \cap f_2FE \supset f_1f_2E$, therefore

$$\dim F_2FE \geq 2 \dim EF - \dim E \tag{5.4}$$

by using the formula $\dim(A + B) = \dim A + \dim B - \dim A \cap B$. Similarly, $F_{i+1}FE = F_iFE + f_{i+1}FE$ and $F_iFE \cap f_{i+1}FE \supset f_jf_{i+1}E$ for all $j = 1, 2, \ldots i$. From which we have

$$\dim F_{i+1}FE \geq \dim F_iFE + \dim FE - i \dim E.$$

Now $\dim F^{\langle 2 \rangle}E = r\binom{r+1}{2}$ only occurs when we have equality in all the above inequalities, in particular we have equality in (5.4) which implies that the inclusion $f_1f_2E \subset f_1FE \cap f_2FE$ is also an equality. □

Lemma 5.3.3. *Let U be a subspace of EF. Under the hypothesis $\dim F^{\langle 2\rangle}E = r\binom{r+1}{2}$, we have that there exists a subspace $A \subset F$ of dimension 2 such that,*

$$\dim AU < 2 \dim U$$

if and only if U contains two non-zero elements of the form fe and $f'e$ $f, f' \in F$, $e \in E$ where f and f' are two linearly independent elements of F.

Proof. Let A be a subspace of F of dimension 2 generated by f_1, f_2. We have $AU = f_1U + f_2U$ so that $\dim AU < 2\dim U$ if and only if $f_1U \cap f_2U \neq \{0\}$. But we have

$$f_1U \cap f_2U \subset f_1FE \cap f_2FE$$

and under the hypothesis $\dim F^{\langle 2\rangle}E = r\binom{r+1}{2}$ we have that $\dim AFE = 2r^2 - r$ and $f_1FE \cap f_2FE = f_1f_2E$. Therefore $f_1U \cap f_2U$ contains a non-zero element if and only if U contains an element of the form f_2e and an element of the form f_1e, for $e \in E$, $e \neq 0$. □

Corollary 5.3.4. *Suppose $\dim F^{\langle 2\rangle}E = r\binom{r+1}{2}$, and that U is a subspace of FE such that for any two non-zero elements $f \in F$ and $e \in E$, $ef \notin U$. Then we have, for any subspace $A \subset F$ of dimension 2,*

$$\dim AU = 2\dim U.$$

Next, we study the probability distribution of the dimension of the product space $A(U+W)$, where W is random of dimension w, and U is either constructed as above or uniform random. We only focus on the binary extension field case $q = 2$, and from the previous discussion we only keep the property that $\dim AU$ is maximal. In other words, for the purpose of the following analysis, U is a fixed subspace of $\mathbb{F}_{2^m}$ with $\dim U = u$, A is a fixed subspace of $\mathbb{F}_{2^m}$ of dimension $\dim A = 2$ and we suppose that we have $\dim AU = 2u$. Let W be a *random* subspace of dimension $\dim W = w$ of $\mathbb{F}_{2^m}$. The space W is chosen by choosing $x_1, x_2, \dots, x_w$ random independent (in the sense of probability) elements of $\mathbb{F}_{2^m}$ and W is taken to be the subspace generated by the x_i. Strictly speaking, $x_1, \dots, x_w$ may turn out not to be linearly independent and not generate a space of dimension w. However, w will be taken to be much smaller than m, so that this event happens with negligible probability.

Our goal is to study the probability that $A(U+W)$ does not have dimension $2(u+w)$ and see how it may vary for two different spaces U_1 and U_2.

Consider the mapping

$$\begin{array}{rcl} A^w & \xrightarrow{\Phi} & \mathbb{F}_{2^m} \\ (a_1, a_2, \dots, a_w) & \mapsto & a_1x_1 + a_2x_2 + \cdots + a_wx_w \end{array}$$

The product space $A(U+W)$ does not have maximal dimension, namely $2(u+w)$, if and only if there is a non-zero

$$\mathbf{a} = (a_1, a_2, \dots, a_w)$$

in A^w such that $\Phi(\mathbf{a}) \in AU$. This event $\mathcal{E}$, over all choices of $\mathbf{x} = (x_1, \dots, x_w)$, can therefore be written as:

$$\mathcal{E} = \bigcup_{\substack{\mathbf{a}\in A^w \\ \mathbf{a}\neq 0}} \mathcal{E}_{\mathbf{a}}$$

where $\mathcal{E}_{\mathbf{a}}$ denotes the event $\Phi(\mathbf{a}) \in AU$. Since

$$\operatorname{Prob} \mathcal{E}_{\mathbf{a}} = \frac{4^u}{2^m}$$

the union bound gives us

$$\operatorname{Prob} \mathcal{E} \leq (4^w - 1)\frac{4^u}{2^m}. \tag{5.5}$$

We now study the lower bound

$$\operatorname{Prob} \mathcal{E} \geq \sum_{\substack{\mathbf{a} \in A^w \\ \mathbf{a} \neq 0}} \operatorname{Prob} \mathcal{E}_{\mathbf{a}} - \sum_{\mathbf{a},\mathbf{b}} \operatorname{Prob} \mathcal{E}_{\mathbf{a}} \cap \mathcal{E}_{\mathbf{b}} \tag{5.6}$$

where the second sum runs over all unordered pairs of distinct w-tuples $\mathbf{a}$ and $\mathbf{b}$. To evaluate this second sum we split the pairs $\mathbf{a}, \mathbf{b}$ into two disjoint sets:

1. linearly independent pairs $\mathbf{a}, \mathbf{b}$. In which case the two random variables $\mathbf{ax}$ and $\mathbf{bx}$ are independent, and we have

$$\operatorname{Prob} \mathcal{E}_{\mathbf{a}} \cap \mathcal{E}_{\mathbf{b}} = \operatorname{Prob} \mathcal{E}_{\mathbf{a}} \operatorname{Prob} \mathcal{E}_{\mathbf{b}} = \left(\frac{4^u}{2^m}\right)^2.$$

2. linearly dependent pairs $\mathbf{a}, \lambda\mathbf{a}$, for some $\lambda \in \mathbb{F}_{2^m}$, $\lambda \neq 1$, such that $\lambda\mathbf{a} \in A^w$. In this case, we have

$$\operatorname{Prob} \mathcal{E}_{\mathbf{a}} \cap \mathcal{E}_{\mathbf{b}} = |AU \cap \lambda AU| \frac{1}{2^m} \leq \frac{4^u}{2^m}.$$

   We now estimate the number of such pairs $\mathbf{a}, \lambda\mathbf{a}$.

   Denote the non-zero elements of A by $a_1, a_2, a_3 = a_1 + a_2$ (recall that A is a vector space). Suppose we have $\lambda a_1 = a_2$ and $\lambda a_2 = a_3 = a_1 + a_2$ ($\lambda a_2 = a_2$ would imply $\lambda = 1$ and $\lambda a_2 = a_1$ would imply $\lambda^2 = 1$ hence $\lambda = 1$ in a field of characteristic 2). Then $a_2 a_1^{-1} = \lambda$ satisfies $\lambda^2 + \lambda + 1$ which is not possible for odd m and happens with negligible probability for even m. Assuming this does not happen we have therefore that any $\mathbf{a} \in A^w$ such that $\lambda\mathbf{a} \in A^w$ must have all non-zero coefficients equal. Hence the number of such pairs $\mathbf{a}, \lambda\mathbf{a}$ is at most 3.2^w. Inequality (5.6) gives us therefore:

$$\operatorname{Prob} \mathcal{E} \geq (4^w - 1)\frac{4^u}{2^m} - \binom{4^w - 1}{2}\frac{4^{2u}}{2^{2m}} - 3.2^w \frac{4^u}{2^m}. \tag{5.7}$$

From which we get:

Proposition 5.3.5. *If U and V are two spaces of dimension u such that* $\dim AU = \dim AV = 2u$ *then*

$$|\operatorname{Prob}[\dim A(U+W) < 2(u+w)] - \operatorname{Prob}[\dim A(V+W) < 2(u+w)]| \leq \binom{4^w - 1}{2}\frac{4^{2u}}{2^{2m}} + 3.2^w \frac{4^u}{2^m}.$$

Product space distinguisher. We go back to our distinguishing problem defined above. As mentioned in the discussion leading up to the problem, it is natural to try and distinguish between (i) and (ii) by computing the dimension of some AQ for many instances of Q and basing the decision on the number of times an abnormal (less than $2\dim Q$) turns up. The consequence of Proposition 5.3.5 is that to distinguish confidently with this method requires a very large number of queries. Specifically, if the two probabilities of producing an abnormal dimension are p and $p(1+\varepsilon)$, then the number of products AQ that one must produce is of the order $1/p\varepsilon^2$. Proposition 5.3.5 gives $p \approx 2^{2u+2w-m}$ and $\varepsilon = 2^{\log_2 3-w}$.

Proposition 5.3.6. *By applying Proposition 5.3.5 to the* $\mathrm{PSSI}_{r,d,\lambda,w,\mathbb{F}_{q^m}}$ *problem, the advantage with which one may distinguish the two distributions is of the order of* $2^{m-2(rd-\lambda)}$.

Remark: One might also consider computing product spaces of the form ZE' where E' is a subspace of E of dimension larger than 2. However, we have chosen our parameters such that $3(w+rd-\lambda) > m$ so this idea cannot work.

New Problem: Advanced Product Subspaces Indistinguishability (PSSI^+) The PSSI^+ problem is a generalization of the previous problem, with some extra side information. We need to consider this problem for our security proof.

Problem 5.3.7. *Advanced Product Spaces Subspaces Indistinguishability. Let E be a fixed $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ of dimension r. Let F_i, U_i and W_i be subspaces defined as before:*

— $F_i \xleftarrow{\$} \mathbf{Gr}(d, \mathbb{F}_{q^m})$

— $U_i \xleftarrow{\$} \mathbf{Gr}(rd-\lambda, EF_i)$ *such that* $\{ef, e \in E, f \in F\} \cap U_i = \{0\}$

— $W_i \xleftarrow{\$} \mathbf{Gr}(w, \mathbb{F}_{q^m})$

Let $\boldsymbol{H}$ be a Randomly chosen $(n-k)\times n$ ideal double circulant matrix as in definition 2.3.5 for an irreducible polynomial P.

— *Sample l vectors $\boldsymbol{s}_i$ and l' vectors $\boldsymbol{s}'_i$ of length n from the same support E of dimension r*

— *Set $\boldsymbol{S}$ (respectively $\boldsymbol{S}'$) the matrix consisting of all $\boldsymbol{s}_i$ (respectively $\boldsymbol{s}'_i$) and their ideal shifts. Let $\boldsymbol{T} = \boldsymbol{H}\boldsymbol{S}^T$ and $\boldsymbol{T}' = \boldsymbol{H}\boldsymbol{S}'^T$.*

The $\mathrm{PSSI}^+(N)_{r,d,\lambda,w,\mathbb{F}_{q^m}}$ *problem consists in distinguishing N samples of the form $(\boldsymbol{z}_i, F_i)$ where $\boldsymbol{z}_i$ is a random vector of $\mathbb{F}_{q^m}^n$ of support $W_i + U_i$ from samples of the form $(\boldsymbol{z}'_i, F_i)$ where $\boldsymbol{z}'_i$ is a random vector of $\mathbb{F}_{q^m}^n$ of weight $w+rd-\lambda$ when additionally given $\boldsymbol{H}, \boldsymbol{T}, \boldsymbol{T}'$.*

The PSSI^+ consists of an instance of the PSSI problem and an instance of the RSL problem that share the same secret support E. The question is to determine whether or not the instance of RSL can be used in order to reduce the difficulty of PSSI.

In general, combining two difficult problem together does not necessarily lead to another hard problem. For example, combining two difficult instances of the factorization of large integers n, n' with $n = pq$ and $n' = pq'$ where p, q and q' are prime is a an easy problem.

In our case, the knowledge of an instance of RSL could be useful if it gives us some information on the support E. But, for our parameters, the best attacks on the RSL problem are based on the GRS$^+$ algorithm [15, 36, 26] and this algorithm recovers the whole support or nothing. Moreover, the main idea behind the GRS$^+$ algorithm (which consists of looking for a subspace E' which contains E) cannot be applied to the PSSI$^+$ problem since E is "multiplied" by an F_i at each sample. Thus it appears that the knowledge of an instance of RSL that shares the same secret support E does not help to solve the PSSI problem and we will consider that the PSSI$^+$ problem is as hard to attack as the PSSI problem.

5.3.2 Security model

One of the security models for signature schemes is existential unforgeability under an adaptive chosen message attack (EUF-CMA). Basically, it means that even if an adversary has access to a signature oracle, it cannot produce a valid signature for a new message with a non negligible probability.

Existential Unforgeability under Chosen Message Attacks [45] ($\mathsf{EUF-CMA}$). Even after querying N valid signatures on chosen messages (μ_i), an adversary should not be able to output a valid signature on a fresh message μ. To formalize this notion, we define a signing oracle OSign:

— $\mathsf{OSign}(\mathsf{vk}, \mu)$: This oracle outputs a signature on μ valid under the verification key vk. The requested message is added to the signed messages set $\mathcal{SM}$.

```
$\mathbf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\mu^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \mathsf{OSign}(\mathsf{vk}, \cdot))$
4. $b \leftarrow \mathsf{Verify}(\mathsf{vk}, \mu^*, \sigma^*)$
5. IF $\mu^* \in \mathcal{SM}$ RETURN 0
6. ELSE RETURN $b$
```

The probability of success against this game is denoted by

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\lambda) = 1], \quad \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(\lambda, t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\lambda).$$

5.3.3 $\mathsf{EUF-CMA}$ proof

To prove the $\mathsf{EUF-CMA}$ security of our scheme, we proceed in two steps. In the first step, we show that an adversary with access to N valid signatures has a negligible advantage on the same adversary with only access to the public keys. In other words, we prove that signatures do not leak information of the secret keys. In the second step, we show that if we only have access to the public keys, a valid signature allows us to solve an instance of the I − ARSD problem.

A Technical Lemma

Lemma 5.3.8. *Let $\mathcal{F}$ be a family of functions defined by*

$$\mathcal{F} = \left\{ \begin{array}{cccc} f_{\boldsymbol{H}} : & (W + EF)^n & \rightarrow & \mathbb{F}_{q^m}^{n-k} \\ & \boldsymbol{y} & \mapsto & \boldsymbol{x} = \boldsymbol{y}\boldsymbol{H}^T \end{array} \right\}$$

Since $\boldsymbol{H}$ is chosen uniformly at random amongst the $(n-k)\times n$ ideal double circulant matrices, $\mathcal{F}$ is a pairwise independent family of function.
The number of choices for $\boldsymbol{y}$ depends on W and F and on the choice of the coordinates of $\boldsymbol{y}$. Overall, the entropy of $\boldsymbol{y}$ is equal to

$$\Theta\left(\begin{bmatrix} w \\ m \end{bmatrix}\begin{bmatrix} d \\ m \end{bmatrix} q^{(w+rd)n}\right) = 2^{(w(m-w)+d(m-d)+(w+rd)n)\log q+O(1)}.$$

Since $|\boldsymbol{y}|_r > d_{RGV}$, any vector of $\mathbb{F}_{q^m}^{n-k}$ can be reached, thus the entropy of $\boldsymbol{x}$ is equal to $2^{(n-k)m\log q}$. According to the Leftover Hash Lemma [47], we have

$$\Delta(\mathcal{D}_{\mathbf{G_0}},\mathcal{U}) < \frac{\varepsilon}{2}$$

where $\Delta(X,Y)$ denotes the statistical distance between X and Y, $\mathcal{D}_{\mathbf{G_0}}$ denotes the distribution of $\boldsymbol{x}$ in game $\mathbf{G_0}$, $\mathcal{U}$ denotes the uniform distribution over $\mathbb{F}_{q^m}^{n-k}$ (so the distribution of $\boldsymbol{x}'$ in game $\mathbf{G_1}$) and

$$\varepsilon = 2^{\frac{((n-k)m-w(m-w)+d(m-d)+(w+rd)n)\log q}{2}+O(1)}.$$

Proofs For the first step, we proceed in a sequence of games. We denote $\mathbb{P}_{\mathbf{G_i}}$ the probability that the adversary returns 1 in the end of the game $\mathbf{G_i}$ and $\mathsf{Adv}(\mathbf{G_i}) = |\mathbb{P}_{\mathbf{G_i}} - \frac{1}{2}|$ the advantage of the adversary for the game $\mathbf{G_i}$.

— $\mathbf{G_0}$: this is the real $\mathsf{EUF-CMA}$ game for $\mathcal{S}$. The adversary has access to the signature oracle OSign to obtain valid signatures.

$$\mathbb{P}_{\mathbf{G_0}} = \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\lambda)$$

— $\mathbf{G_1}$: we replace $\boldsymbol{z}$ by a vector $\boldsymbol{z}'$ of the same weight chosen uniformly at random in the correct subspace U of $W+EF$, and sample $\boldsymbol{c}', \boldsymbol{p}'$ uniformly with support F. Now set $\boldsymbol{x}' = \boldsymbol{H}\boldsymbol{z}' - \boldsymbol{c}'\boldsymbol{T}' - \boldsymbol{p}'\boldsymbol{T}$ and use the Random Oracle to set $\boldsymbol{c} = \mathcal{H}(\boldsymbol{x}', F, \mu)$. In $\mathbf{G_0}$, $\boldsymbol{x}$ is the syndrome of the vector $\boldsymbol{y}$ of support of the form $EF+W$, while here $\boldsymbol{x}'$ is not necessarily. Under Lemma 5.3.8 we conclude.

$$\mathsf{Adv}(\mathbf{G_1}) \leq \mathsf{Adv}(\mathbf{G_0}) + \varepsilon$$

The parameters of the signature are chosen such that ε is lower than the security parameter.

— $\mathbf{G_2}$: We now sample $\boldsymbol{z}$ at random in $\mathbb{F}_{q^m}^n$ with the same weight, and proceeds as in $\mathbf{G_2}$.
This corresponds to an instance of the $\mathrm{PSSI}^+(N)$ problem 5.3.7. Since the adversary can have access to at most N signatures, we have

$$|\mathsf{Adv}(\mathbf{G_2}) - \mathsf{Adv}(\mathbf{G_1})| \leqslant \mathsf{Adv}(\mathrm{PSSI}^+(N))$$

— $\mathbf{G_3}$: We now pick $\boldsymbol{T}, \boldsymbol{T}'$ at random and proceed as before. The difference between $\mathbf{G_3}$ and $\mathbf{G_2}$ resides in the public key, on whether it was sampled using vectors in a given subspace or not.

$$|\mathsf{Adv}(\mathbf{G_3}) - \mathsf{Adv}(\mathbf{G_2})| \leqslant \mathsf{Adv}(\mathrm{DRSL})$$

At this step, everything we send to the adversary is random, and independent from any secret keys. Hence the security of our scheme is reduced to the case where no signature is given to the attacker.

If he can compute a valid signature after the game $\mathbf{G_3}$, then the challenger can compute a solution of the $\mathrm{I-ARSD}$ problem. Indeed, the couple $(\boldsymbol{z}, \boldsymbol{p})$ is a solution of the instance $(\boldsymbol{H}, -\boldsymbol{T}, F, \boldsymbol{x}+\boldsymbol{T}'\boldsymbol{c}^T, \omega+rd-\lambda)$ of the $\mathrm{I-ARSD}$ problem. According to Proposition 5.1.3, the $\mathrm{I-ARSD}$ problem is reduced to the $\mathrm{I-RSD}$ problem.

Finally, we can give the main theorem of our chapter:

Theorem 5.3.9 (EUF-CMA security)**.** *Under the hypothesis of the hardness of the* PSSI^+ *problem 5.3.1 and of the* $\mathrm{DRSL}, \mathrm{I-RSD}$ *problems 2.3.6, our signature scheme is secure under the EUF-CMA model in the Random Oracle Model.*

5.4 Attacks

5.4.1 Attacks on the RSL problem

In this section we will study the hardness of recovering the secret matrices $\boldsymbol{S}$ and $\boldsymbol{S}'$ from $\boldsymbol{H}, \boldsymbol{T}, \boldsymbol{T}'$. This is exactly an instance of the $\mathrm{RSL}_{q,m,n,k,w,N}$ problem.

We will use the setting proposed in [36]. First, we recall how the problem is reduced to searching for a codeword of weight w in a code containing q^N codewords of this form. We introduce the following $\mathbb{F}_q$-linear code :

$$C = \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \boldsymbol{H}\boldsymbol{x} \in W_T\}$$

Where W_T is the $\mathbb{F}_q$-linear subspace of $\mathbb{F}_{q^m}^{n-k}$ generated by the linear combinations of the columns of the public matrices $\boldsymbol{T}$ and $\boldsymbol{T}'$. As in the lemma 1 in [36], we define :

$$C' = \{\sum_i \alpha_i \boldsymbol{s}_i, \alpha_i \in \mathbb{F}_q\}.$$

We have :

— $dim_{\mathbb{F}_q} C \leqslant km + N$

— $C' \subset C$

— the elements of C' are of weight $\leqslant w$.

Combinatorial attack

In [36], the authors search for codewords of rank w in C by using information-set decoding techniques, using the fact that C' contains q^N words of weight w. As such a codeword will very likely be a linear combinations of the vectors $\boldsymbol{s}_i$, it will reveal the secret support E with high probability. Theorem 2 in [36] gives a complexity of $q^{min(e_-,e_+)}$, where :

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

Where $K = km + N$. See [36] for more details about this.

Algebraic attack

We will now study how algebraic attacks can be used to find codewords of weight w in C.

We are looking for $X \in C$ such that $X \in E^n$. We can write X as :

$$X = \begin{bmatrix} \sum_{i=1}^{w} x_1^{(i)} y_1^{(i)} & \dots & \sum_{i=1}^{w} x_1^{(i)} y_n^{(i)} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{w} x_m^{(i)} y_1^{(i)} & \dots & \sum_{i=1}^{w} x_m^{(i)} y_n^{(i)} \end{bmatrix}.$$

Where $(x^{(1)}, \dots, x^{(w)})$ represent a basis of E, and the $(y_i^j), 1 \leqslant i \leqslant w, 1 \leqslant j \leqslant n$ represent the coordinates of X written in this basis.

C has length nm and dimension $N + km$ in $\mathbb{F}_q$, which gives $(n-k)m - N$ parity check equations, and $(n+m)w$ unknowns (the $x_i^{(j)}$ and the $(y_i^{(j)})$).

To decrease the number of unknowns, we will first write the basis of E in an echeloned form, which removes w^2 unknowns :

$$\forall (i,j) \in [1,w]^2, i \neq j, x_i^{(j)} = 0$$

$$x_i^{(i)} = 1$$

Then we will use the fact that for a fixed basis of E, the solution space has dimension N, which allows us to set N of the $(y_i^{(j)})$ to specialize one solution, as in [25] : for a random subset $I \subseteq [1,n] \times [1,w]$ of size $N-1$:

$$\forall (i,j) \in I, y_i^{(j)} = 0$$

$$y_{i_0}^{(j_0)} = 1, (i_0, j_0) \notin I$$

Which removes N unknowns.

Proposition 5.4.1. *Using this setting we obtain :*

— $(n-k)m - N$ *equations*

— $(n+m)w - w^2 - N$ *unknowns.*

We implemented this approach in Magma to try it on small examples, and the combinatorial attacks become much more efficient than the algebraic approach when the number of samples is around kr, while this attack is faster when the number of samples is higher. Another drawback of this attack is the high memory cost, making parameters as small as $n = m = 30, k = 15, r = 3$ with kr samples too big for a computer using $16GB$ of RAM.

For concrete parameters (section 5.5), we chose N, the number of samples for the RSL problem, equal to either $k(r-1)$ or $k(r-2)$. Our experiments on smaller parameters showed that combinatorial attacks should be way faster for this number of samples. This also defeats the setting proposed in [25] since it needs at least kr samples.

The parameter set I gives 2117 unknowns for 23836 equations and the parameter set II gives 2809 unknowns for 29154 equations. Based on our experiments on smaller parameters this seems really hard to reach.

5.4.2 Attack on the ARSD problem

As explained in the security proof in Section 5.3.3, a forgery attack consists in solving an instance of the ARSD problem 5.1.2. In order to choose the parameters of our signature, we need to deal with the complexity of the attacks on this problem. Since it is very similar to the RSD problem, these attacks are adaptations of the attacks against the RSD problem [41, 15].

The following proposition gives a bound from which the problem becomes polynomial.

Proposition 5.4.2. *Let $(\boldsymbol{H}, \boldsymbol{H}', \boldsymbol{s}, F)$ be an instance of the $ARSD_{q,m,n,k,r,n',F}$ problem. If $\max(m,n)r + n'r' \geqslant m(n-k)$ then the* ARSD *problem can be solved in polynomial time with a probabilistic algorithm.*

Proof. To prove this proposition, we will use the method used to compute the Singleton bound.

Let us begin with the case $n \geqslant m$. Let E be a subspace of $\mathbb{F}_{q^m}$ of dimension r and suppose that there exists a solution $(\boldsymbol{x}, \boldsymbol{x}')$ of the ARSD problem such that $\text{Supp}(\boldsymbol{x}) = E$. Then, we can express the coordinate x_i of $\boldsymbol{x}$ in a basis $\{E_1, \ldots, E_r\}$ of E:

$$\forall i \in \{1 \ldots n\}, x_j = \sum_{i=1}^{r} \lambda_{ij} E_i$$

Likewise, we can express the coordinates of $\boldsymbol{x}'$ in a basis of F:

$$\forall i \in \{1 \ldots n'\}, x'_t = \sum_{s=1}^{r'} \lambda'_{st} F_s$$

Let us write the linear system satisfied by the unknown $(\lambda_{ij}, \lambda'_{st})$:

$$\begin{aligned}
& \boldsymbol{H}\boldsymbol{x}^T && +\boldsymbol{H}'\boldsymbol{x}'^T && =\boldsymbol{s} \\
\iff \; & \forall i \in \{1..n-k\}, \sum_{j=1}^{n} H_{ij} x_j && +\sum_{t=1}^{n'} H'_{it} x'_t && =s_i \\
\iff \; & \forall i \in \{1..n-k\}, \sum_{j=1}^{n} H_{ij} \sum_{i=1}^{r} \lambda_{ij} E_i && +\sum_{t=1}^{n'} H'_{it} \sum_{s=1}^{r'} \lambda'_{st} F_s && =s_i
\end{aligned} \qquad (5.8)$$

By fixing E as a random subspace of $\mathbb{F}_{q^m}$ of dimension r, we obtain $(n-k)$ linear equations (5.8) over $\mathbb{F}_{q^m}$ that can be projected on $\mathbb{F}_q$ to obtain $m(n-k)$ linear equations over $\mathbb{F}_q$. Since we have $nr + n'r \geqslant m(n-k)$, there are more unknowns than equations so the system admits at least a solution with a non negligible probability.

In the case $m > n$, we need to consider the matrix $\boldsymbol{M}(\boldsymbol{x})$ associated to $\boldsymbol{x}$ (cf definition 2.1.5) and express its rows in a basis of a subspace E of dimension r of $\mathbb{F}_q^n$. Since the support of $\boldsymbol{x}'$ is fixed, its coordinates still give us $n'r'$ unknowns over $\mathbb{F}_q$. This gives us $mr + n'r'$ unknowns over $\mathbb{F}_q$ in total. Then we transform the equation $\boldsymbol{H}\boldsymbol{x}^T + \boldsymbol{H}'\boldsymbol{x}'^T = \boldsymbol{s}$ into a linear system over $\mathbb{F}_q$ as previously. This operation is not difficult but technically complicated and we do not give the details of the equations. Finally we obtain a linear system of $m(n-k)$ equations and $mr+n'r'$ unknowns over $\mathbb{F}_q$. This system has a solution with a non negligible probability since $mr + n'r' \geqslant m(n-k)$.

In both case, the solution of the system gives us a solution to the instance of the ARSD problem. □

In the case where $\max(m,n)r + n'r' < m(n-k)$, we need to adapt the best attacks against the RSD problem in the case of the ARSD problem. The attack is detailed in [41]. The general idea is to find a subspace E of dimension δ such that $\mathrm{Supp}(\boldsymbol{x}) \subset E$ (in the case $n \geqslant m$). Then we can express the coordinates of $\boldsymbol{x}$ if $n \geqslant m$ or the rows of the matrix $\boldsymbol{M}(\boldsymbol{x})$ if $m > n$ in a basis of E exactly as in the previous proposition. We want δ as large as possible to increase the probability that $\mathrm{Supp}(\boldsymbol{x}) \subset E$ but we have to take δ such that $\max(m,n)\delta + n'r' < m(n-k)$ in order to obtain an over-constrained linear system. Hence $\delta = \left\lfloor \frac{m(n-k)-n'r'}{\max(m,n)} \right\rfloor$. The probability that $E \supset \mathrm{Supp}(\boldsymbol{x})$ depends on m and n:

— If $n \geqslant m$, then E is a subspace of $\mathbb{F}_{q^m}$: $\mathbb{P}(E \supset \mathrm{Supp}(\boldsymbol{x})) = \dfrac{\begin{bmatrix} r \\ \delta \end{bmatrix}}{\begin{bmatrix} r \\ m \end{bmatrix}} = \Theta(q^{-r(m-\delta)})$.

— If $n < m$ then E is a subspace of $\mathbb{F}_q^n$: $\mathbb{P}(E \supset \mathrm{Supp}(\boldsymbol{x})) = \dfrac{\begin{bmatrix} r \\ \delta \end{bmatrix}}{\begin{bmatrix} r \\ n \end{bmatrix}} = \Theta(q^{-r(n-\delta)})$.

In order to respect the constraints of our signature, we have to take parameters such that the instance of the ARSD problem has several solutions. Thus, the average complexity of this attack is equal to the inverse of the probability $\mathbb{P}(E \supset \mathrm{Supp}(\boldsymbol{x}))$ divided by the number of solutions times the cost of the linear algebra. The number of solutions is in $\Theta\left(q^{r(m+n-r)+n'r'-m(n-k)}\right)$ (see Proposition 5.1.3 for the details).

Proposition 5.4.3. *In the case* $\max(m,n)r + n'r' < m(n-k)$*, the complexity of the best attack against the* $\mathrm{ARSD}_{q,m,n,k,r,n',F}$ *problem is in*

$$\mathcal{O}\left(m^3(n-k)^3 q^{r\left\lceil \frac{km+n'r'}{\max(m,n)} \right\rceil - r(m+n-r)-n'r'+m(n-k)}\right).$$

Remark: We did not consider the improvement of the attack of the RSD problem in [15] because this attack does not fit the case where there are several solutions to the RSD problem.

5.5 Parameters

5.5.1 Constraints

In this section we recap the different constraints on our parameters.

Choice of l, l', r and d

First we need to choose l' such that the entropy of $\boldsymbol{c}$ is high enough. For our parameters, $l' = 1$ is always enough since $\boldsymbol{c} \in F^{l'dk}$ and $dk > 512$. In practice using less than dk coordinates for $\boldsymbol{c}$ is a possibility to make the parameters a little smaller.

We then need to choose r high enough such that both the attack on both the RSD and RSL problems are hard. d and l must be chosen such that $\lambda \geqslant r + d$: $d = r$ and $l = 4$ is a way to meet this condition. In the sets of parameters given below, this value of l leads to $N = k(r-1)$ and $N = k(r-2)$ respectively, which allows us to be pretty conservative with respect to the attacks on the RSL problem.

Choice of m

In order to avoid the distinguisher attack for a security parameter of 128, the relation $m - 2u \geqslant 128 + 64$ (we use Proposition 5.3.6, where $u = rd - \lambda$, must be verified to fit the security proof : we consider that the adversary has access to 2^{64} signatures, so the probability of distinguishing signatures and random vectors must be lower than 2^{-192}. We choose a prime m (so there is no intermediate field between $\mathbb{F}_q$ and $\mathbb{F}_{q^m}$) such that $m \geqslant 192 + 2u$.

Choice of n, k and w

n, k and w must be chosen such that $3(u + w) > m$ to avoid the distinguisher attack using subspaces of dimension 3, and $(u + w) < (n - k) - \lambda$ in order to keep the weight of the signature below the Singleton bound $-\lambda$ (due to ARSD). k is taken prime for having access to really sparse polynomials to define the ideal codes.

5.5.2 Example of parameters

The public key consists of :

— $\boldsymbol{H}$ which can be recovered from a seed (256 bits)

— $l(n-k)m\log(q)$ bits to describe the syndromes

The signature consists of :

— $(rd + \omega - \lambda)(n + m - rd - \omega + \lambda)\log(q)$ bits to describe $\boldsymbol{z}$. We give $\text{Supp}(\boldsymbol{z})$ in echelon form as well as the coordinates in this basis

— A seed to describe F (256 bits)

— 512 bits to describe $\boldsymbol{c}$

— $dlk\log(q)$ bits to describe $\boldsymbol{p}$

The complexity of the key recovery attack is computed using the complexity of the combinatorial attack given in Section 5.4.1.

For our parameters, the complexity of the forgery attack using the algorithm against ARSD described in Section 5.4.2 is disproportionately large compared to the key recovery attack.

As stated in 5.5.1, to target a security level of 2^{128}, we choose our parameters such that the probability of distinguishing signatures from random vectors is smaller than 2^{-192}, considering the maximum number of calls to the signature oracle an attacker can make is 2^{64}.

	m	n	k	l	l'	d	r	ω	λ	q	Public key size	Signature size	Key recovery attack	Distinguisher	Security
I	241	202	101	4	1	6	6	57	12	2	121 961	32 514	461	193	128
II	263	226	113	4	1	7	7	56	14	2	148 851	40 150	660	193	128

The implementation of our scheme on an Intel(R) Core(TM) i5-7440HQ CPU running at 2.80GHz gives the following computation times :

Parameter	Keygen	Online signature phase	Verification
I	4ms	4ms	5ms
II	5ms	5ms	6ms

For the offline phase, the most costly step, the computation of the matrix $\boldsymbol{D}$ (see 5.2.6 from precisions), takes 350ms for parameter I and 700ms for parameter II.

5.6 Conclusion

We have described a variation on the Schnorr-Lyubashevsky approach for rank based cryptography. This new approach enables us to obtain a randomization of the signature, which seemed difficult to derive before this work for code-based cryptography. We provide a detailed analysis of attacks and an EUF-CMA proof for our scheme. Overall the parameters we propose are efficient and comparable in terms of signature size to the Dilithium lattice-based scheme [29].

Chapter 6

HQC-RMRS

In this chapter we present a twofold contribution to the HQC cryptosystem. Firstly, we provide a better analysis of the distribution of the weight of the error, which allows for a better Decryption Failure Rate (DFR) analysis. Secondly we propose the use of concatenated Reed-Muller and Reed-Solomon codes as the public code: both of these improvements allow to reach low DFR (for example $< 2^{-128}$) with shorter codes, hence leading to shorter public keys and ciphertexts. This work is a joint work with Philippe Gaborit and Gilles Zémor.

Contents

6.1 Preliminaries

The preliminaries were introduced chapter 2. In particular, the HQC cryptosystem was introduced in 2.4.1.

6.2 Error vector distribution

To study the decryption failure rate (DFR) of the HQC scheme we first need to study the distribution of the error vector $\boldsymbol{e}' = \boldsymbol{x} \cdot \boldsymbol{r}_2 + \boldsymbol{y} \cdot \boldsymbol{r}_1 + \boldsymbol{e}$ where the vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{r}_1, \boldsymbol{r}_2$ and $\boldsymbol{e}$ have been taken randomly and uniformly among the vectors of weight exactly w for

$\boldsymbol{x}, \boldsymbol{y}$, weight $w_{\mathbf{r}}$ for $\boldsymbol{r}_1, \boldsymbol{r}_2$, and weight $w_{\mathbf{e}}$ for $\boldsymbol{e}$. The precise distribution of $\boldsymbol{e}'$ is difficult to derive but we can compute exactly the distribution of its individual coordinates.

We first focus on the distribution of the products $\boldsymbol{x} \cdot \boldsymbol{r}_2$ and $\boldsymbol{y} \cdot \boldsymbol{r}_1$.

6.2.1 Analysis of the product of two vectors

Proposition 6.2.1. *Let $\boldsymbol{x} = (x_0, \dots x_{n-1})$ be a random vector chosen uniformly among all binary vectors of weight w and let $\boldsymbol{r} = (r_0, \dots, r_{n-1})$ be a random vector chosen uniformly among all vectors of weight w_r and independently of $\boldsymbol{x}$. Then, denoting $\boldsymbol{z} = \boldsymbol{x} \cdot \boldsymbol{r}$, we have that for every $k \in \{0, \dots n-1\}$, the k-th coordinate z_k of $\boldsymbol{z}$ is Bernoulli distributed with parameter $\tilde{p} = P(z_k = 1)$ equal to:*

$$\tilde{p} = \frac{1}{\binom{n}{w}\binom{n}{w_{\mathbf{r}}}} \sum_{\substack{1 \leqslant \ell \leqslant \min(w, w_{\mathbf{r}}) \\ \ell \text{ odd}}} C_\ell$$

where $C_\ell = \binom{n}{\ell}\binom{n-\ell}{w-\ell}\binom{n-w}{w_{\mathbf{r}}-\ell}$.

Proof. The total number of ordered pairs $(\boldsymbol{x}, \boldsymbol{r})$ is $\binom{n}{w}\binom{n}{w_{\mathbf{r}}}$. Among those, we need to count how many are such that $z_k = 1$. We note that

$$z_k = \sum_{\substack{i+j=k \bmod n \\ 0 \le i,j \le n-1}} x_i r_j.$$

We need therefore to count the number of couples $(\boldsymbol{x}, \boldsymbol{r})$ such that we have $x_i r_{k-i} = 1$ an odd number of times when i ranges over $\{0, \dots, n-1\}$ (and $k-i$ is understood modulo n). Let us count the number C_ℓ of couples $(\boldsymbol{x}, \boldsymbol{r})$ such that $x_i r_{k-i} = 1$ exactly ℓ times. For $\ell > \min(w, w_{\mathbf{r}})$ we clearly have $C_\ell = 0$. For $\ell \le \min(w, w_{\mathbf{r}})$ we have $\binom{n}{\ell}$ choices for the set of coordinates i such that $x_i = r_{k-i} = 1$, then $\binom{n-\ell}{w}$ remaining choices for the set of coordinates i such that $x_i = 1$ and $r_{k-i} = 0$, and finally $\binom{n-w}{w_{\mathbf{r}}-\ell}$ remaining choices for the set of coordinates i such that $x_i = 0$ and $r_{k-i} = 1$. Hence $C_\ell = \binom{n}{\ell}\binom{n-\ell}{w-\ell}\binom{n-w}{w_{\mathbf{r}}-\ell}$. The formula for $\tilde{p}$ follows. □

6.2.2 Analysis of $\boldsymbol{e}'$

Proposition 6.2.2. *Let $\boldsymbol{x}, \boldsymbol{y}$ be uniformly chosen among vectors of weight w, let $\boldsymbol{r}_1, \boldsymbol{r}_2$ be uniformly chosen among vectors of weight $w_{\mathbf{r}}$, and let $\mathbf{e}$ be uniformly chosen among vectors of weight $w_{\mathbf{e}}$. We suppose furthermore that the random vectors $\mathbf{x}_v, \boldsymbol{y}, \boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{e}$ are independent. Let $\mathbf{e}' = \mathbf{x} \cdot \mathbf{r}_2 - \mathbf{r}_1 \cdot \mathbf{y} + \mathbf{e} = (e'_0, \dots, e'_{n-1})$. Then, for every $k \in \{0, \dots, n-1\}$ we have:*

$$\begin{cases} \Pr[e'_k = 1] = 2\tilde{p}(1-\tilde{p})(1 - \frac{w_{\mathbf{e}}}{n}) + ((1-\tilde{p})^2 + \tilde{p}^2)\,\frac{w_{\mathbf{e}}}{n}, \\ \Pr[e'_k = 0] = ((1-\tilde{p})^2 + \tilde{p}^2)\,(1 - \frac{w_{\mathbf{e}}}{n}) + 2\tilde{p}(1-\tilde{p})\frac{w_{\mathbf{e}}}{n}. \end{cases} \tag{6.1}$$

Proof. The vectors $\mathbf{x} \cdot \mathbf{r}_2$, $\mathbf{r}_1 \cdot \mathbf{y}$ and $\boldsymbol{e}$ are clearly independent. The k-th coordinate of $\boldsymbol{e}$ is Bernoulli distributed with parameter $w_{\mathbf{e}}/n$. The random Bernoulli variable e'_k is therefore the sum modulo 2 of three independent Bernoulli variables of parameters $\tilde{p}$ for

Parameter	w	$w_{\mathbf{e}} = w_{\mathbf{r}}$	n	$n_1 n_2$	$p^\star$
I	67	77	23869	23746	0.2918
II	67	77	20533	20480	0.3196

Figure 6.1 – HQC parameters used to test the binomial approximation

the first two and of parameter $w_{\mathbf{e}}/n$ for the third one. The formula therefore follows standardly. $\square$

In the following we denote $\Pr[e'_k = 1]$ by $p^\star$.

6.2.3 Simulation results

In this section we are going to compare the weight distribution of vectors following the binomial distribution $\mathcal{B}(n, p^\star)$ and actual vectors $\boldsymbol{e}'$ computed as $\boldsymbol{x} \cdot \boldsymbol{r}_2 + \boldsymbol{y} \cdot \boldsymbol{r}_1 + \boldsymbol{e}$.

Figure 6.1 shows the parameters we performed the simulations on.

Simulations for Parameter set I

Simulation results are shown figure 6.2. We computed the weights such that 0.1%, 0.01% and 0.001% of the vectors are of weight greater than this value, to study how often extreme weight values occur. Results are presented table 6.1.

To match the security definitions of the HQC scheme, we generated vectors of length n and then truncated the last $n - n_1 n_2$ coordinates to obtain vectors of length $n_1 n_2$.

	0.1%	0.01%	0.001%	0.0001%
Error vectors	7101	7134	7163	7190
Binomial approximation	7147	7191	7228	7267

Table 6.1 – Simulated probabilities of large weights for Parameter I for the distributions of the error vector and the binomial approximation

Simulations for Parameter set II

Simulation results are shown on figure 6.3. We perform the same analysis as for the parameter set I about extreme weight values. Results are presented table 6.2.

	0.1%	0.01%	0.001%	0.0001%
Error vectors	6714	6748	6780	6810
Binomial approximation	6754	6796	6832	6866

Table 6.2 – Simulated probabilities of large weights for Parameter II for the distributions of the error vector and the binomial approximation

These results indicate that the extreme values are more frequent when considering the binomial distribution, which means that approximating the HQC vectors by Bernoulli variables for studying the DFR will lead to an upper bound.

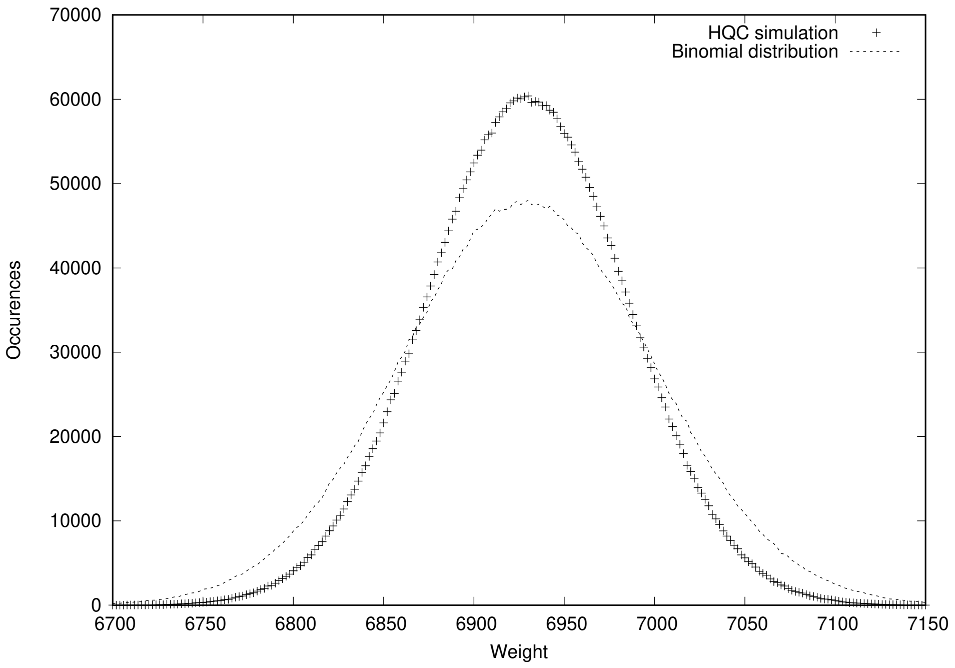

Figure 6.2 – Comparison between error $\mathbf{e}'$ generated using parameter set I and its binomial approximation.

6.3 Proposition of codes

In this section we study the impact of using a new family of codes instead of the tensor product codes used in the HQC cryptosystem: concatenated Reed-Muller and Reed-Solomon codes. We denote this variation of the cryptosystem HQC-RMRS.

6.3.1 Construction

Definition 6.3.1. *Concatenated codes*

A concatenated code consists of an external code $[n_e, k_e, d_e]$ over $\mathbb{F}_q$ and an internal code $[n_i, k_i, d_i]$ over $\mathbb{F}_2$, with $q = 2^{k_i}$. We use a bijection between elements of $\mathbb{F}_q$ and the words of the internal code, this way we obtain a transformation:

$$\mathbb{F}_q^{n_e} \to \mathbb{F}_2^N$$

where $N = n_e n_i$. The external code is thus transformed into a binary code of parameters $[N = n_e n_i, K = k_e k_i, D \geqslant d_e d_i]$.

For the external code, we chose a Reed-Solomon code of dimension 32 over $\mathbb{F}_{256}$ and, for the internal code, we chose the Reed-Muller code $[128, 8, 64]$ that we are going to duplicate between 2 and 6 times (i.e duplicating each bit to obtain codes of parameters $[256, 8, 128], [512, 8, 256], [786, 8, 384]$).

Decoding:

We perform maximum likelihood decoding on the internal code. Doing that we obtain a vector of $\mathbb{F}_q^{n_e}$ that we then decode using an algebraic decoder for the Reed-Solomon code.

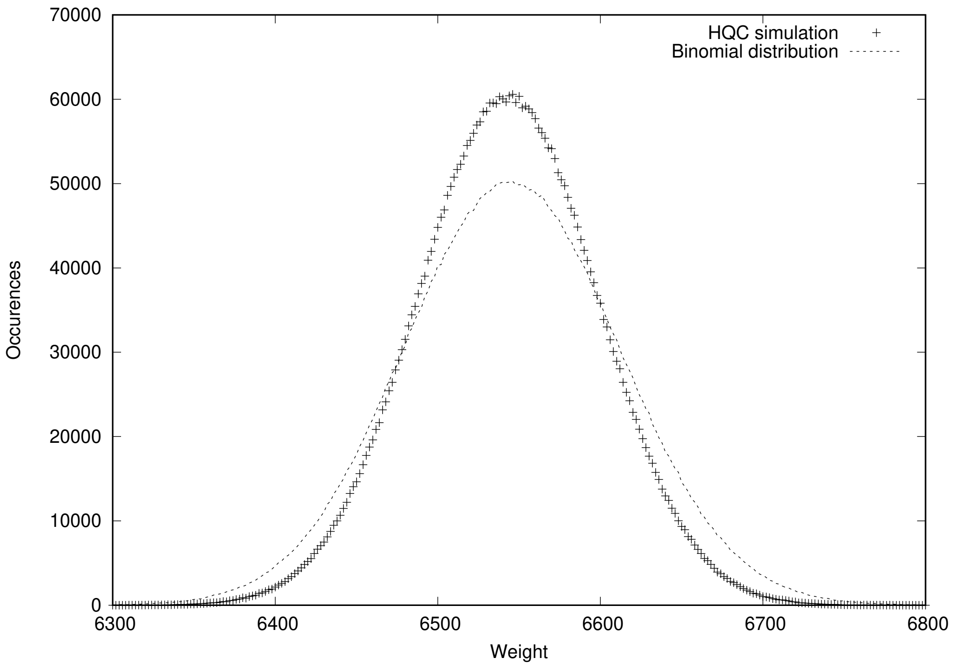

Figure 6.3 – Comparison between error $\mathbf{e}'$ generated using parameter set II and its binomial approximation.

Decoding the internal Reed-Muller code:

The Reed-Muller code of order 1 can be decoded using a fast Hadamard transform (see chapter 14 of [57]). The algorithm needs to be slightly adapted when decoding duplicated codes. For example, if the Reed-Muller is duplicated three times, we create the function $F : \mathbb{F}_2^7 \rightarrow 3, 1, -1, -3^7$ where we started with transforming each block of three bits $x_1x_2x_3$ of the received vector in

$$(-1)^{x_1} + (-1)^{x_2} + (-1)^{x_3}$$

We then apply the Hadamard transform to the function F. We take the maximum value in $\hat{F}$ and $x \in \mathbb{F}_2^7$ that maximizes the value of $|\hat{F}|$. If $\hat{F}(x)$ is positive, then the closest codeword is xG where G is the generator matrix of the Hadamard code (without the all-one-vector). If $\hat{F}(x)$ is negative, then we need to add the all-one-vector to it.

6.3.2 Decryption failure rate analysis

Proposition 6.3.2. *Decryption Failure Rate of the internal code*

Let p be the transition probability of the binary symetric channel. Then the DFR of a binary linear code of dimension 8 and minimal distance d_i can be upper bounded by:

$$p_i = 255 \sum_{j=d_i/2}^{d_i} \binom{d_i}{j} p^j (1-p)^{d_i-j}$$

Proof. For any linear code C of length n, when transmitting a codeword $\boldsymbol{c}$, the probability that the channel makes the received word $\boldsymbol{y}$ at least as close to a word $\boldsymbol{c}' = \boldsymbol{c} + \boldsymbol{x}$ as $\boldsymbol{c}$ is:

$$\sum_{j \geqslant |\boldsymbol{x}|/2} \binom{n}{j} p^j (1-p)^{n-j}$$

The probability of a decryption failure can thus be upper bounded by:

$$\sum_{\boldsymbol{x} \in C, \boldsymbol{x} \neq 0} \sum_{j \geqslant |\boldsymbol{x}|/2} \binom{n}{j} p^j (1-p)^{n-j}$$

The number of codewords of a binary code of dimension 8 is 256, hence the result. □

Better approximation of the DFR of the internal code

Proposition 6.3.3. *Decryption Failure Rate of the internal code*

Let p be the transition probability of the binary symetric channel. Then the DFR of a binary linear code of dimension 8 and minimal distance d_i can be upper bounded by:

$$\begin{aligned} p_i = {} & \frac{1}{2} 255 \binom{d_i}{d_i/2} p^{d_i/2} (1-p)^{d_i/2} \\ & + 255 \sum_{i=d_i/2+1}^{d_i} \binom{d_i}{i} p^i (1-p)^{d-i} \\ & + \frac{1}{2} \binom{255}{2} \sum_{i=0}^{d_i/2} \binom{d_i/2}{i}^3 p^{d_i - i} (1-p)^{d_i/2+i} \end{aligned}$$

Proof. Let E be the decoding error event. Let $\boldsymbol{e}$ be the error vector.

- Let A be the event where the closest non-zero codeword $\boldsymbol{c}$ to the error is such that $d(\boldsymbol{e}, \boldsymbol{c}) = d(\boldsymbol{e}, \boldsymbol{0}) = |\boldsymbol{e}|$
- Let B be the event where the closest non-zero codeword $\boldsymbol{c}$ to the error vector is such that $d(\boldsymbol{e}, \boldsymbol{c}) < |\boldsymbol{e}|$
- Let $A' \subset A$ be the event where the closest non-zero codeword $\boldsymbol{c}$ to the error vector is such that $d(\boldsymbol{e}, \boldsymbol{c}) = |\boldsymbol{e}|$ and such a vector is unique, meaning that for every $\boldsymbol{c}' \in C, \boldsymbol{c}' \neq \boldsymbol{c}, \boldsymbol{c}' \neq \boldsymbol{0}$, we have $d(\boldsymbol{e}, \boldsymbol{c}') > |\boldsymbol{e}|$
- Finally, let A'' be the event that is the complement of A' in A, meaning the event where the closest nonzero codeword $\boldsymbol{c}$ to the error is at distance $|\boldsymbol{e}|$ from $\boldsymbol{e}$, and there exists at least one codeword $\boldsymbol{c}', \boldsymbol{c}' \neq \boldsymbol{c}, \boldsymbol{c}' \neq \boldsymbol{0}$, such that $d(\boldsymbol{e}, \boldsymbol{c}') = d(\boldsymbol{e}, \boldsymbol{c}) = |\boldsymbol{e}|$

The probability space is partitioned as $\Omega = A \cup B \cup C = A' \cup A'' \cup B \cup C$, where C is the complement of $A \cup B$. When C occurs, the decoder always decodes correctly, i.e. $\text{Prob}(E|C) = 0$. We therefore write:

$$\text{Prob}(E) = \text{Prob}(E|A')\,\text{Prob}(A') + \text{Prob}(E|A'')\,\text{Prob}(A'') + \text{Prob}(E|B)\,\text{Prob}(B)$$

When the event A' occurs, the decoder chooses at random between the two closest codewords and is correct with probability 1/2, i.e. $\text{Prob}(E|A') = 1/2$. We have $\text{Prob}(E|B) = 1$ and writing $\text{Prob}(E|A'') \leqslant 1$, we have:

$$\begin{aligned}
\operatorname{Prob}(E) &\leqslant \frac{1}{2}\operatorname{Prob}(A') + \operatorname{Prob}(A'') + \operatorname{Prob}(B) \\
&= \frac{1}{2}(\operatorname{Prob}(A') + \operatorname{Prob}(A'')) + \frac{1}{2}\operatorname{Prob}(A'') + \operatorname{Prob}(B) \\
\operatorname{Prob}(E) &\leqslant \frac{1}{2}\operatorname{Prob}(A) + \frac{1}{2}\operatorname{Prob}(A'') + \operatorname{Prob}(B) \qquad (6.2)
\end{aligned}$$

Now we have the straightforward union bounds:

$$\operatorname{Prob}(B) \leqslant 255 \sum_{i=d_i/2+1}^{d_i} \binom{d_i}{i} p^i (1-p)^{d-i} \qquad (6.3)$$

$$\operatorname{Prob}(A) \leqslant 255 \binom{d_i}{d_i/2} p^{d_i/2}(1-p)^{d_i/2} \qquad (6.4)$$

and it remains to find an upper bound on $\operatorname{Prob}(A'')$.
We have:

$$\operatorname{Prob}(A'') \leqslant \sum_{\boldsymbol{c},\boldsymbol{c}'} \operatorname{Prob}(A_{\boldsymbol{c},\boldsymbol{c}'})$$

where the sum is over pairs of distinct nonzero codewords and where:

$$A_{\boldsymbol{c},\boldsymbol{c}'} = \{d(\boldsymbol{e},\boldsymbol{c}) = d(\boldsymbol{e},\boldsymbol{c}') = |\boldsymbol{e}|\}$$

This event is equivalent to the error meeting the supports of $\boldsymbol{c}$ and $\boldsymbol{c}'$ on exactly half their coordinates. All codewords except the all-one vector have weight d_i, and any two codewords of weight d_i either have non-intersecting supports or intersect in exactly $d_i/2$ positions. $\operatorname{Prob}(A_{\boldsymbol{c},\boldsymbol{c}'})$ is largest when $\boldsymbol{c}$ and $\boldsymbol{c}'$ have weight d and non-zero intersection. In this case we have:

$$\operatorname{Prob}(A_{\boldsymbol{c},\boldsymbol{c}'}) = \sum_{i=0}^{d_i/2} \binom{d_i/2}{i}^3 p^{d_i-i}(1-p)^{d_i/2+i}$$

Hence

$$\operatorname{Prob}(A'') \leqslant \sum_{\boldsymbol{c},\boldsymbol{c}'} \operatorname{Prob}(A_{\boldsymbol{c},\boldsymbol{c}'}) \leqslant \binom{255}{2} \sum_{i=0}^{d_i/2} \binom{d_i/2}{i}^3 p^{d_i-i}(1-p)^{d_i/2+i} \qquad (6.5)$$

Plugging 6.4, 6.3 and 6.5 into 6.2 we obtain the result.

□

Remark 6.3.4. *The formulas 6.3.2 and 6.3.3 give upper bounds of the Decryption Failure Rate for the internal code. When the DFR gets smaller, the bounds become closer to the real value. We give a comparaison the bounds from 6.3.2 and 6.3.3 and the actual DFR for* $[256, 8, 128]$*,* $[512, 8, 256]$ *and* $[768, 8, 384]$ *duplicated Reed-Muller using* $p^\star$ *values from actual parameters. Simulation results are presented table 6.3.*

Security level	$p^\star$	Reed-Muller code	DFR from 6.3.2	DFR from 6.3.3	Observed DFR
128	0.3196	$[256, 8, 128]$	-7.84	-8.03	-8.72
192	0.3535	$[512, 8, 256]$	-11.81	-12.12	-12.22
256	0.3728	$[768, 8, 384]$	-13.90	-14.20	-14.18

Table 6.3 – Comparison between the observed Decryption Failure Rate and the formula from proposition 6.3.2. Results are presented as $log_2(DFR)$.

Theorem 6.3.5. *Decryption Failure Rate of the concatenated code*
Using a Reed-Solomon code $[n_e, k_e, d_e]_{\mathbb{F}_{256}}$ as the external code, the DFR of the concatenated code can be upper bounded by:

$$\sum_{l=\delta_e+1}^{n_e} \binom{n_e}{l} p_i^l (1-p_i)^{n_e-l}$$

Where $d_e = 2\delta_e + 1$ and p_i is defined as in proposition6.3.2.

6.3.3 Simulation results

We tested the Decryption Failure rate of the concatenated codes against both symetric binary channels and HQC vectors. For Reed-Muller codes, rather than considering the lower bound approximation we effectively decoded the Reed-Muller, which means than in practice the lower bound that we use for our theoretical DFR, is greater than what is obtained in the simulations. Simulation results are presented figure 6.4.

6.3.4 Proposed parameters

From the DFR analysis we derive new parameters for the HQC-RMRS cryptosystem. These are described figure 6.5.

6.4 Conclusion

In section 6.2 we presented a better analysis of the error weight distribution for HQC, which leads to a better DFR estimation. This can be used to reduce the size of the parameters, no matter what family of codes is used for decoding. In section 6.3 we propose to use concatenated Reed-Muller and Reed-Solomon codes and we provide an upper bound of the DFR in this setting. This family allows to reduce the public key and ciphertext sizes by about 15% when compared to the tensor product of BCH and repetition (when considering the same error weight distribution).

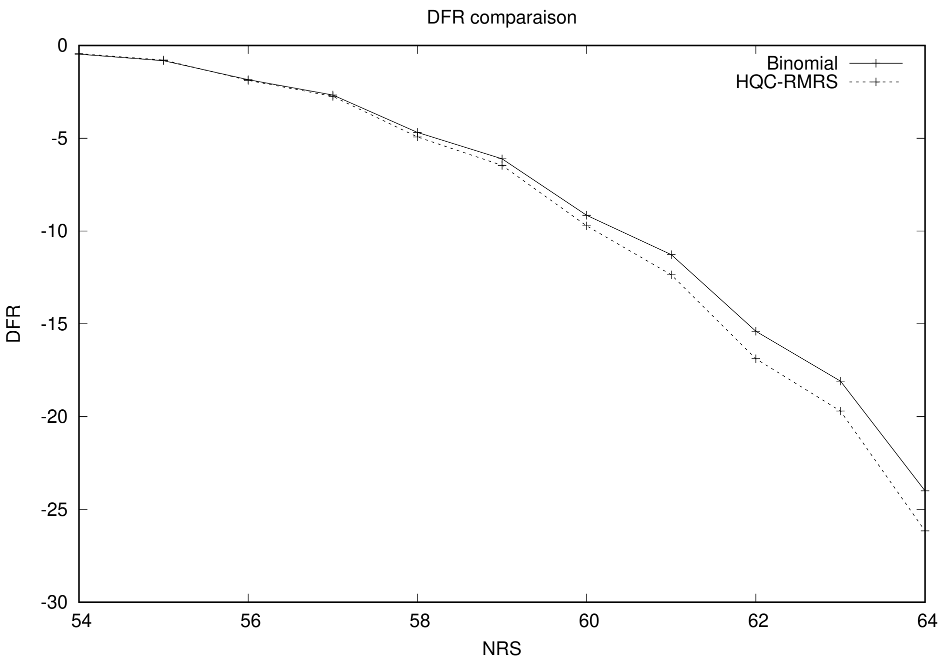

Figure 6.4 – Comparison of the Decryption Failure Rate of concatenated codes against approximation by a binary symmetric channel and against HQC error vectors. Parameters simulated are derived from those of HQC-RMRS for 128 security bits: $w = 67, w_r = w_e = 77$, a $[256, 8, 128]$ duplicated Reed-Muller code for internal code and a $[NRS, 32]$ Reed-Solomon code for external code.

w	$w_{\mathbf{e}} = w_{\mathbf{r}}$	Reed-Muller	Reed Solomon	N	DFR	Improvment over HQC
67	77	$[256, 8, 128]$	$[80, 32, 49]$	20,533	$< 2^{-128}$	14%
101	117	$[512, 8, 256]$	$[76, 32, 45]$	38,923	$< 2^{-128}$	13.9%
133	153	$[768, 8, 384]$	$[78, 32, 47]$	59,957	$< 2^{-128}$	13.4%

Figure 6.5 – New proposed parameters for the HQC-RMRS cryptosystem

Part IV

Cryptanalyse

Chapter 7

Cryptanalyse de deux adaptations du schéma de signature de Lyubashevsky

In this chapter we are going to study two adaptations of the Lyubashevsky signature scheme that was presented section 5.2 : the first one is a straightforward adaptation of the framework to the rank metric (without the weight reduction technique used in Durandal), we will refer to this scheme as the SHMW signature scheme. The second one is an adaptation to the Hamming metric, we will refer to this scheme as the SHMWW signature scheme. We present both schemes as well as techniques used to exploit information leakage from the signatures in order to recover the secret keys. The cryptanalysis of the SHMW signature scheme is a joint work with Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Terry Shue Chien Lau, Chik How Tan and Keita Xagawa and was published to DCC [9]. The cryptanalysis of the SHMWW signature scheme is a joint work with Jean-Christophe Deneuville and Philippe Gaborit.

Contents

7.1 SHMW signature scheme

We recall the scheme by Song *et al.* in figure 7.1. To simplify the description of the scheme and without loss of generality, we assume that the collision-resistant hash function (CRHF) $\mathcal{H} : \{0,1\}^* \to \mathcal{S}^n_{w_{\boldsymbol{g}}}(\mathbb{F}_{q^m})$ used in SHMW [68] takes as input random binary strings, and outputs words in $\mathbb{F}^n_{q^m}$ of rank weight $w_{\boldsymbol{g}}$.

7.1.1 Cryptanalysis of SHMW signature scheme

In this section, we present a way to turn a valid signature into a decoding problem. We show that this problem can be efficiently solved using the LRPC decoding algorithm.

Algorithm 1 SHMW.KeyGen$(n, w, w_{\boldsymbol{r}}, w_{\boldsymbol{g}})$

```
Require: Public parameters $(n, w, w_{\boldsymbol{r}}, w_{\boldsymbol{g}})$ depending on the security parameter $1^\lambda$
Ensure: $(\mathsf{pk}, \mathsf{sk})$ with $\mathsf{pk} = (\boldsymbol{h}, \boldsymbol{s})$ and $\mathsf{sk} = (\boldsymbol{x}, \boldsymbol{y})$
1: $\boldsymbol{h} \xleftarrow{\$} \mathbb{F}_{q^m}^n$
2: $\boldsymbol{x}, \boldsymbol{y} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ such $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = w$ and $\operatorname{Supp} \boldsymbol{x} = \operatorname{Supp} \boldsymbol{y}$
3: $\boldsymbol{s} \leftarrow \boldsymbol{x} + \boldsymbol{h}\boldsymbol{y}$
4: return $(\mathsf{pk} = (\boldsymbol{h}, \boldsymbol{s}), \mathsf{sk} = (\boldsymbol{x}, \boldsymbol{y}))$
```

Algorithm 2 SHMW.Sign$(\mathsf{pk}, \mathsf{sk}, \boldsymbol{m})$

```
Require: Public and private keys, message $\boldsymbol{m} \in \{0,1\}^*$ to be signed
Ensure: Signature $(\boldsymbol{g}, \boldsymbol{u})$ of message $\boldsymbol{m}$
1: $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2) \xleftarrow{\$} \mathbb{F}_{q^m}^{2n}$ such that $\|\boldsymbol{r}_1\| = \|\boldsymbol{r}_2\| = w_{\boldsymbol{r}}$ and $\operatorname{Supp} \boldsymbol{r}_1 = \operatorname{Supp} \boldsymbol{r}_2$
2: $\boldsymbol{t} \leftarrow \boldsymbol{r}_1 + \boldsymbol{h}\boldsymbol{r}_2$
3: $\boldsymbol{g} \leftarrow \mathcal{H}(\boldsymbol{t}, \boldsymbol{m})$
4: $\boldsymbol{u}_1 \leftarrow \boldsymbol{x} \cdot \boldsymbol{g} + \boldsymbol{r}_1$ and $\boldsymbol{u}_2 \leftarrow \boldsymbol{y} \cdot \boldsymbol{g} + \boldsymbol{r}_2$, $\boldsymbol{u} = (\boldsymbol{u}_1, \boldsymbol{u}_2)$
5: return $(\boldsymbol{g}, \boldsymbol{u})$
```

Algorithm 3 SHMW.Verify$(\mathsf{pk}, (\boldsymbol{g}, \boldsymbol{u}), \boldsymbol{m})$

```
Require: Public key, message $\boldsymbol{m}$, and the signature $(\boldsymbol{g}, \boldsymbol{u})$ to verify
Ensure: Accept if $(\boldsymbol{g}, \boldsymbol{u})$ is a valid signature on $\boldsymbol{m}$, Reject otherwise
1: if $\mathcal{H}(\boldsymbol{u}_1 + \boldsymbol{h}\boldsymbol{u}_2 - \boldsymbol{s}\boldsymbol{g}, \boldsymbol{m}) = \boldsymbol{g}$ and $\|\boldsymbol{u}_i\| \leq w w_{\boldsymbol{g}} + w_{\boldsymbol{r}}$ then
2:     return Accept
3: else
4:     return Reject
```

Figure 7.1 – Description of the SHMW signature scheme.

This leads to a full cryptanalysis of Song *et al.* signature scheme, from a single signature. We also provide an implementation of both the SHMW signature scheme and our cryptanalysis. Timings for both softwares are reported in Tab. 7.1.

Signature as a decoding problem

We now focus on a single signature of the SHMW scheme. Let $\sigma = (\boldsymbol{g}, \boldsymbol{u})$ be a valid signature on the message $\boldsymbol{m}$. Therefore, we have:

— $\boldsymbol{g} = \mathcal{H}(\boldsymbol{r}_1 + \boldsymbol{h}\boldsymbol{r}_2, \boldsymbol{m})$, for some $\boldsymbol{r}_1, \boldsymbol{r}_2$ unknown of small rank,

— $\boldsymbol{u}_1 = \boldsymbol{x}\boldsymbol{g} + \boldsymbol{r}_1$ for some unknown (secret key part) $\boldsymbol{x}$ of small weight,

— $\boldsymbol{u}_2 = \boldsymbol{y}\boldsymbol{g} + \boldsymbol{r}_2$ for some unknown (secret key part) $\boldsymbol{y}$ of small weight,

— $\|\boldsymbol{g}\| \leq w_{\boldsymbol{g}}$ with $\frac{1}{17}n \leq w_{\boldsymbol{g}} \leq \frac{1}{13}n$ depending on the SHMW parameters.

The relatively low weight of $\boldsymbol{g}$ is a necessary condition for the security of the signature scheme (see [68]) that stems from the RGV bound. However, since $\boldsymbol{g}$ is public (given in the signature), it is possible to use techniques coming from the decoding of LRPC codes in order to recover the support of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Let $\mathcal{C}$ denote the code whose parity-check matrix is $\boldsymbol{H} = \boldsymbol{M}(\boldsymbol{g})$. Then the equations of $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ correspond to two (related) instances of a noisy version of the Syndrome Decoding problem in rank metric.

Decoding LRPC codes

To recover the secret key in SHMW scheme, we have the following system of equations:

$$\begin{cases} \boldsymbol{u}_1^\top = \boldsymbol{H}\boldsymbol{x}^\top + \boldsymbol{r}_1^\top \\ \boldsymbol{u}_2^\top = \boldsymbol{H}\boldsymbol{y}^\top + \boldsymbol{r}_2^\top \end{cases}$$

We are first going to recover the support of the secret key $(\boldsymbol{x}, \boldsymbol{y})$, thus recovering the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ from the above equations reduces to linear algebra.

Notation 7.1.1. *In the following, we denote by :*

— *F the support of $\boldsymbol{H}$ of weight $w_{\boldsymbol{g}}$*

— *E the support of $(\boldsymbol{x}, \boldsymbol{y})$ of weight w*

— *R the support of $(\boldsymbol{r}_1, \boldsymbol{r}_2)$ of weight $w_{\boldsymbol{r}}$*

Let S denote the vector space generated by the coordinates of the vector $\boldsymbol{u}$:

$$S = \langle \boldsymbol{u}_{11}, ..., \boldsymbol{u}_{1n}, \boldsymbol{u}_{21}, ..., \boldsymbol{u}_{2n} \rangle.$$

Its dimension is at most $ww_{\boldsymbol{g}} + w_{\boldsymbol{r}}$, and it is a subspace of $EF + R$.

Cryptanalysis Algorithm

We use the decoding from [39] to recover E from the coordinates of $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. The algorithm is depicted Fig. 4.

This algorithm relies on the fact that $E \subset S_i$, where $S_i = F_i^{-1}.S$, in order to recover the support of the error. Since adding $(\boldsymbol{r}_1, \boldsymbol{r}_2)$ to the signature does not remove this inclusion, the algorithm still works in the same way.

Algorithm 4 SupportRecoverer(F, $\boldsymbol{s}$, r)

```
Require: $F$, $(\boldsymbol{u}_{11}, ..., \boldsymbol{u}_{1n}, \boldsymbol{u}_{21}, ..., \boldsymbol{u}_{2n})$ (a vector), $w$ (the dimension of $E$)
Ensure: A candidate for the vector space $E$
1: Compute $S = \langle \boldsymbol{u}_{11}, ..., \boldsymbol{u}_{1n}, \boldsymbol{u}_{21}, ..., \boldsymbol{u}_{2n} \rangle$    $\rightarrow$ Part 1 : Compute the vector space $E.F + R$
2: Compute every $S_i = F_i^{-1}.S$ with $F_i$ an element of a basis of $F$, for $i = 1$ to $w_{\boldsymbol{g}}$
3: $E \leftarrow S_1 \cap ... \cap S_{w_{\boldsymbol{g}}}$    $\rightarrow$ Part 2 : Recover the vector space $E$
4: return $E$
```

This algorithm is the same as the one described in [39], except that S is a subspace of $E.F + R$ instead of $E.F$.

Proposition 7.1.2. *If $2n \geqslant ww_{\boldsymbol{g}} + w_{\boldsymbol{r}}$, then Alg. 4 recovers E with a probability $1 - q^{-(2n-(ww_{\boldsymbol{g}}+w_{\boldsymbol{r}})+1)}$.*

Proof. In order for the algorithm to succeed, parts 1 and 2 both need to succeed. We treat each part separately.

Part 1. First we need $S = \langle \boldsymbol{u}_{11}, ..., \boldsymbol{u}_{1n}, \boldsymbol{u}_{21}, ..., \boldsymbol{u}_{2n} \rangle$ to be equal to $E.F + R$, that is to say the $2n$ coordinates from $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ must span the whole vector space of dimension $ww_{\boldsymbol{g}} + w_{\boldsymbol{r}}$. This is possible as long as :

$$2n \geqslant ww_{\boldsymbol{g}} + w_{\boldsymbol{r}}$$

The probability that this step fails is the probability that the $2n \times (ww_{\boldsymbol{g}} + w_{\boldsymbol{r}})$ matrix formed by unfolding the coordinates of $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ in a basis of $E.F + R$ is not full rank, which is equal to (see [39] for more details) :

$$q^{-(2n-(ww_{\boldsymbol{g}}+w_{\boldsymbol{r}})+1)}$$

Part 2. As in [39], we know that each of the $E \subset S_i$ for $i = 1$ to $w_{\boldsymbol{g}}$. For the considered parameters, the probability that $\dim(\bigcap_{i=1}^{w_{\boldsymbol{g}}} S_i) > \dim(E)$ is negligible compared to the probability that part one fails, hence the result. □

Once the support E of $\boldsymbol{x}$ and $\boldsymbol{y}$ is recovered, we can compute the coordinates of the secret key using linear algebra. From the equations $\boldsymbol{s} = \boldsymbol{x} + \boldsymbol{hy}$ we can build a linear system consisting of nm equations (from $\boldsymbol{s}$) and $2nw$ unknowns (the coordinates of $\boldsymbol{x}$ and $\boldsymbol{y}$) in the base field.

Putting the pieces together

Implementation. In order to gauge the cryptanalysis efficiency, we implemented both Song *et al.* signature scheme and the proposed attack. Both implementations are available online at `github.com/deneuville/cryptanalysisSHMW`. The code was compiled using `gcc` v5.4.0 with flags `-O3`, and ran on an Intel® Core™ i7-6920HQ CPU @ 2.90GHz with TurboBoost disabled. The timings are reported in Tab. 7.1.

Instance	Claimed security	t_{sign} (ms)	t_{break} (ms)
RQCS-I	128	4	45
RQCS-II	192	8	165
RQCS-III	256	11	200

Table 7.1 – Performance of SHMW signature generation versus cryptanalysis for all the proposed sets of parameters.

Complexity analysis. As claimed by the authors, a signature generation requires $\mathcal{O}\left(n^2 m \log\left(m\right) \log\left(\log\left(m\right)\right)\right)$ operations in the base field $\mathbb{F}_q$. The decoding algorithm of LRPC codes requires $(w_{\boldsymbol{g}} - 1)(ww_{\boldsymbol{g}} + w_{\boldsymbol{r}})^2 m$ operations in $\mathbb{F}_q$ to recover the support E (the cost of $w_{\boldsymbol{g}} - 1$ intersections of vector spaces of dimension $ww_{\boldsymbol{g}} + w_{\boldsymbol{r}}$), plus $(nw)^3$ operations in order to solve the linear allowing to recover $sk = (\boldsymbol{x}, \boldsymbol{y})$.

7.2 SHMWW's variation on Schnorr-Lyubashevsky

Recently, Song *et al.* proposed a code-based variation on Schnorr-Lyubashevsky's framework. Their approach essentially differs from previous adaptations in the construction of the secret key. In the rest of the paper, we use the notations of [69]: k' and n' denote the dimension and length of the inner generator matrices $\boldsymbol{E}_i = [\boldsymbol{I}_{k'} \mid \boldsymbol{R}_i]$ under systematic form ($\boldsymbol{I}_{k'}$ denotes the identity matrix of dimension k'), l denotes the number of such matrices, and k and $n = ln'$ are the dimension and length of a random code over $\mathbb{F}$ defined by its parity-check matrix $\boldsymbol{H}$. Let $\boldsymbol{P}_1$ (resp. $\boldsymbol{P}_2$) be a random permutation matrix of k' (resp. n) elements. The secret key is the matrix $\boldsymbol{E} = \boldsymbol{P}_1[\boldsymbol{E}_1 \mid \cdots \mid \boldsymbol{E}_l]\boldsymbol{P}_2 \in \mathbb{F}^{k' \times n}$, and the public key consists of $\boldsymbol{H} \in \mathbb{F}^{(n-k)\times n}$ and $\boldsymbol{S} = \boldsymbol{H}\boldsymbol{E}^\top \in \mathbb{F}^{(n-k)\times k'}$.

By constructing the secret key this way, row elements of the secret key $\boldsymbol{E}$ have an average weight of $l \times (1 + \frac{n'-k'}{2})$, which is close to 1/3 or 1/4 of the Gilbert-Varshamov bound for random linear codes of rate k'/n (depending on the parameters). Notice that this is different from previous proposals such as [64] or the matrix version of [27] where the row weight is closer to $\sqrt{n}$. Also notice that while the permutation $\boldsymbol{P}_2$ permutes the columns of $\boldsymbol{E}' = [\boldsymbol{E}_1 \mid \cdots \mid \boldsymbol{E}_l]$, the permutation $\boldsymbol{P}_1$ only permutes the rows of $\boldsymbol{E}'$, therefore $\boldsymbol{E}'$ and $\boldsymbol{P}_1\boldsymbol{E}'$ generate the same code. In other words, $\boldsymbol{P}_1$ has no positive impact on the security of the SHMWW scheme.

Finally, Song *et al.*also use a Weight Restricted Hash (WRH) function $\mathcal{H}$ in their construction, that on input an arbitrary bit string returns a word of length $\ell = k'$ and Hamming weight $w = w_1$. The authors describe a method for constructing such a hash function. Since our cryptanalysis is independent from that function, we simply denote it $\mathcal{H}_{w_1}^{k'}$ or $\mathcal{H}$ if the context is clear.

To sign a message $\boldsymbol{m}$, a mask $\boldsymbol{e}$ of small weight w_2 is sampled uniformly at random, then committed by its syndrome, together with the message, to get the challenge $\boldsymbol{c} = \mathcal{H}\left(\boldsymbol{m}, \boldsymbol{H}\boldsymbol{e}^\top\right)$. The response to this challenge is the product of the secret key and the challenge, hidden by the committed mask: $\boldsymbol{z} = \boldsymbol{c}\boldsymbol{E} + \boldsymbol{e}$. The signature consists of the challenge and the response: $\sigma = (\boldsymbol{z}, \boldsymbol{c})$. The algorithms for SHMWW signature scheme are depicted in details in Fig. 7.2. Proposed parameters are recalled in Table 7.2.

Criteria for parameters selection. In [69], the authors study the impact of applying Prange Information Set Decoding (ISD) algorithm for both "direct and indirect" key recovery attacks. This essentially provides parameters n, k, d_{GV} and w_2, the other parameters follow by the Gilbert-Varshamov bound and by choosing a value for l:

$$l\left(w_1 + n' - k'\right) + w_2 \leq d_{GV}. \tag{7.1}$$

One of the most technical aspects in the design of a signature scheme is to make the signature distribution statistically independent from the secret key. This allows (by programming the random in the security reduction) the forger to produce valid signatures without knowing the secret key, which can then be used to solve the underlying hard problem. This technicality provides guidance for the choice of the parameters, especially for the Hamming weight (or ℓ_1 norm for Lyubashevsky) of the challenge. Indeed, in order for the SD problem to admit a unique solution, the weight of the signature must be below the Gilbert-Varshamov bound. In the meantime, the weight of the secret key should be big enough in order not to be exhibited easily. In the case of SHMWW, this results in a

Algorithm 5 KeyGen($\mathsf{params} = (n, k, n', k', l, w_1, w_2, d_{GV})$)

```
Require: Public parameters $\mathsf{params} = (n, k, w_1, w_2, \delta)$ chosen from Tab 7.2.
Ensure: $(\mathsf{pk}, \mathsf{sk})$ with $\mathsf{pk} = (\boldsymbol{H}, \boldsymbol{S}) \in \mathbb{F}^{(n-k)\times n} \times \mathbb{F}^{(n-k)\times k'}$ and $\mathsf{sk} = \boldsymbol{E} \in \mathbb{F}^{k'\times n}$
1: $\boldsymbol{H} \xleftarrow{\$} \mathbb{F}^{(n-k)\times n}$
2: $\boldsymbol{R}_i \xleftarrow{\$} \mathbb{F}^{k'\times(n'-k')}$ for $i \in [\![1, l]\!]$
3: Let $\boldsymbol{P}_1$ (resp. $\boldsymbol{P}_2$) be a random $k' \times k'$ (resp. $n \times n$) permutation matrix
4: $\boldsymbol{E} \leftarrow \boldsymbol{P}_1 [\boldsymbol{I}_{k'} | \boldsymbol{R}_1 | \cdots | \boldsymbol{I}_{k'} | \boldsymbol{R}_l] \boldsymbol{P}_2$
5: return $(\mathsf{pk} = (\boldsymbol{H}, \boldsymbol{S} = \boldsymbol{H}\boldsymbol{E}^\top), \mathsf{sk} = \boldsymbol{E})$
```

Algorithm 6 Sign($\mathsf{pk}, \mathsf{sk}, \boldsymbol{m}$)

```
Require: Public and private keys, message $\boldsymbol{m} \in \{0,1\}^*$ to be signed
Ensure: Signature $(\boldsymbol{z}, \boldsymbol{c}) \in \mathbb{F}^n \times \mathbb{F}^{k'}$ of message $\boldsymbol{m}$
1: $\boldsymbol{e} \xleftarrow{\$} \mathcal{S}^n_{w_2}$
2: $\boldsymbol{c} \leftarrow \mathcal{H}(\boldsymbol{m}, \boldsymbol{H}\boldsymbol{e}^\top)$   ($\boldsymbol{c}$ has length $k'$ and weight $w_1$)
3: $\boldsymbol{z} \leftarrow \boldsymbol{c}\boldsymbol{E} + \boldsymbol{e}$
4: return $(\boldsymbol{z}, \boldsymbol{c})$        $\rightarrow$ no rejection sampling
```

Algorithm 7 Verify($\mathsf{pk}, (\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{m}$)

```
Require: Public key, message $\boldsymbol{m}$, and the signature $(\boldsymbol{z}, \boldsymbol{c})$ to verify
Ensure: Accept if $(\boldsymbol{z}, \boldsymbol{c})$ is a valid signature of $\boldsymbol{m}$, Reject otherwise
1: if $\mathcal{H}(\boldsymbol{m}, \boldsymbol{H}\boldsymbol{z}^\top - \boldsymbol{S}\boldsymbol{c}^\top) = \boldsymbol{c}$ and $\|\boldsymbol{z}\| \le l(w_1 + n' - k') + w_2$ then
2:     return Accept
3: else
4:     return Reject
```

Figure 7.2 – Song *et al.*code based proposal [69].

secret key $\boldsymbol{E}$ that is sparse: Only $\frac{l(k'+k'(n'-k')/2)}{k'n} \simeq 7\%$ of non-zero coordinates for both parameter sets. This sparsity will be useful for the cryptanalysis.

7.2.1 Cryptanalysis of the SHMWW scheme

In this section we describe how we can exploit the structure of the matrix $\boldsymbol{E}$ and N valid signatures to recover the secret key of the SHMWW scheme.

ISD complexity with the knowledge of random columns

First we are going to show that knowing which columns of $\boldsymbol{E}$ are random and which come from an identity matrix (i.e only have one non-zero coordinate) can be used to efficiently recover $\boldsymbol{E}$.

Instance	n	k	$n-k$	l	n'	k'	$n'-k'$	w_1	w_2	$d=d_{GV}$	λ
Para-1	4096	539	3557	4	1024	890	134	31	531	1191	80
Para-2	8192	1065	7127	8	1024	880	144	53	807	2383	128

Table 7.2 – Original SHMWW parameters [69] for λ bits of security.

Let $\mathcal{I}_R$ be the set of random columns of $\boldsymbol{E}$. Then we can recover $\boldsymbol{E}$ row by row by applying any Information Set Decoding (ISD) algorithm, such as Prange algorithm, and choose an information set $\mathcal{I}$ such that $\mathcal{I}_R \subset \mathcal{I}$. This way we maximize the probability that every non-zero coordinate of the row we are trying to recover are included in the information set.

More precisely, in the SHMWW scheme, we have:

— $|\mathcal{I}_R| = (n'-k') \times l$

— $|\mathcal{I}| = n-k$

Proposition 7.2.1. *The probability p that the l non-zero coordinates of $\boldsymbol{E}$ (the ones from the non-random columns) are included in $\mathcal{I}$ is:*

$$p = \frac{\binom{n-k-(n'-k')\times l}{l}}{\binom{n-(n'-k')\times l}{l}}. \tag{7.2}$$

Proof. By choosing an information set $\mathcal{I}$ such that $\mathcal{I}_R \subset \mathcal{I}$, we have to choose $|\mathcal{I}|-|\mathcal{I}_R| = n-k-(n'-k')\times l$ columns at random and hope that the l remaining non-null coordinates (from the identity matrices) are included in this set.

From this we deduce that the probability of success is the probability that the l non-null coordinates that are distributed in $n-(n'-k')\times l$ positions are included in an information set of size $n-k-(n'-k')\times l$, hence the result.

□

From Proposition 7.2.1 we deduce that the probability of success is approximately of 50% for the parameter set Para-1 and 30% for Para-2.

We are now going to estimate the complexity of recovering the secret key $\boldsymbol{E}$ given the knowledge of the set $\mathcal{I}_R$.

Proposition 7.2.2. *Given the knowledge of $\mathcal{I}_R$, recovering the secret key $\boldsymbol{E}$ costs:*

— 2^{48} *operations for Para-1*

— 2^{52} *operations for Para-2*

Proof. The complexity of solving a linear system to recover a row of $\boldsymbol{E}$ is $(n-k)^3$.

Since the SHMWW scheme only uses binary matrices, the probability that the matrix defining said linear system is invertible can be approximated by 0.288, and the probability p that the system gives the correct solution is given by proposition 7.2.1.

This has to be repeated for each of the k' rows of $\boldsymbol{E}$, which gives the following complexity:

$$\frac{k'((n-k)^3)}{0.288p}$$

Hence the result.

□

Remark: Even if we only have an approximate knowledge of $\mathcal{I}_R$ (i.e if the set is missing some random columns, or non-random columns are included), the ISD will still recover the secret row of $\boldsymbol{E}$ but the probability of success will be lower, hence leading to a higher complexity. Experimental results about how this affects the complexity are given section 7.2.1.

Next we are going to show how we can recover $\mathcal{I}_R$ using leakage from the signatures.

Leakage from the signatures

We are going to exploit the following bias in the weight of the signatures in order to recover $\mathcal{I}_R$:

Proposition 7.2.3. *Let $\mathcal{I}_R$ be the set of random columns of $\boldsymbol{E}$ and let $\boldsymbol{z} = (z_1, \ldots, z_n) = \boldsymbol{cE} + \boldsymbol{e}$. Then we have:*

— $P(z_i = 1) = \frac{1}{2}$ *if* $i \in \mathcal{I}_R$

— $P(z_i = 1) = \frac{w_1}{k'} + \frac{w_2}{n}(1 - 2\frac{w_1}{k'})$ *otherwise.*

Proof. We know that:

$$\boldsymbol{z} = \boldsymbol{cE} + \boldsymbol{e}$$

Where $\boldsymbol{c}$ is a vector of length k' and weight w_1 and $\boldsymbol{e}$ is a vector of length n and weight w_2.

Since $w_1 \ll \frac{k'}{2}$, $\boldsymbol{c}$ has a much lower weight than a random vector of the same length. We are now going to study the weight of each coordinate of the vector $\boldsymbol{z}' = \boldsymbol{cE}$.

Let z'_i be the i-th coordinate of $\boldsymbol{z}'$. Then there are two possibilities:

— If the i-th column of $\boldsymbol{E}$ is random, then the weight of z'_i is 1 with probability $\frac{1}{2}$

— If the i-th column of $\boldsymbol{E}$ is a column of weight 1, then the weight of z'_i is 1 with probability $\frac{w_1}{k'}$

Now we want to compute the probability $P(z_i = 1)$ that the i-th coordinate of $\boldsymbol{z}$ is of weight 1. Since $\boldsymbol{z}'$ and $\boldsymbol{e}$ are independent we have:

$$\begin{aligned} P(z_i = 1) &= P(z'_i = 1) + P(e_i = 1) - 2P(z_i = 1 \wedge e_i = 1) \\ &= P(z'_i = 1) + P(e_i = 1)(1 - 2P(z'_i = 1)) \end{aligned}$$

Which gives the result by replacing $P(z'_i = 1)$ by either $\frac{1}{2}$ or $\frac{w_1}{k'}$ depending on i and $P(e_i = 1)$ by $\frac{w_2}{n}$.

□

Table 7.3 shows the values of $P(z_i = 1)$ for the two SHMWW parameter sets.

	Para-1	Para-2
$P(z_i = 1 \mid i \in \mathcal{I}_R)$	0.5	0.5
$P(z_i = 1 \mid i \notin \mathcal{I}_R)$	0.155	0.147

Table 7.3 – Values of $P(z_i = 1)$ for the SHMWW parameter sets

```
Input: $\boldsymbol{H}, \boldsymbol{S}$, a threshold value $t$,
a set of signatures $(\sigma_1, \ldots, \sigma_N) = ((\boldsymbol{z}_1, \boldsymbol{c}_1), \ldots, (\boldsymbol{z}_N, \boldsymbol{c}_N))$
Output: the secret matrix $\boldsymbol{E}$
    1. $\mathcal{I}_R = \emptyset$
    2. For each $i$ from 1 to $n$:
        — Compute $w_i = \sum_{j=1}^{N} (\boldsymbol{z}_j)_i$
        — If $w_i > N \times t$ then $\mathcal{I}_R = \mathcal{I}_R \cup \{i\}$
    3. For each $i$ from 1 to $k'$:
        — Recover the $i$-th row of $\boldsymbol{E}$ by using an ISD algorithm and the knowledge
          of $\mathcal{I}_R$
    4. Return $\boldsymbol{E}$
```

Figure 7.3 – Secret key recovery of the SHMWW scheme

Using proposition 7.2.3 we can distinguish between random columns and columns from an identity matrix: when acquiring multiple signatures, the coordinates z_i for which, on average, their weight is lower than $\frac{1}{2}$ are most likely to be the coordinates corresponding to columns of weight 1.

From this we can now build an algorithm to recover the secret key of the SHMWW scheme. This algorithm is presented figure 7.3 and uses a threshold value, computed as the mean of $P(z_i = 1 \mid i \in \mathcal{I}_R)$ and $P(z_i = 1 \mid i \notin \mathcal{I}_R)$.

Experimental results

Recovery of $\mathcal{I}_R$. We performed experiments to measure the effectiveness of the recovery of the set of random columns $\mathcal{I}_R$. For each parameter set, we ran our script with an increasing number of signatures and checked the percentage of correct guesses (i.e the number of coordinates in the set $\mathcal{I}_R$ that was computed by the cryptanalysis algorithm that corresponds to actual random columns of the secret key). Results are presented in Figure 7.4. This result shows that the recovery of $\mathcal{I}_R$ quickly becomes very precise when the number of available signatures increases.

Execution time. To demonstrate the effectiveness of our cryptanalysis for each parameter set, we generated 10^3 key pairs. For each of them, we generated N signatures, and ran our cryptanalysis algorithm. The average timings are reported in Table 7.4. The experiments were led on an Intel® Core™ i9-9980HK CPU @ 2.40GHz with Sagemath version 7.5.1.

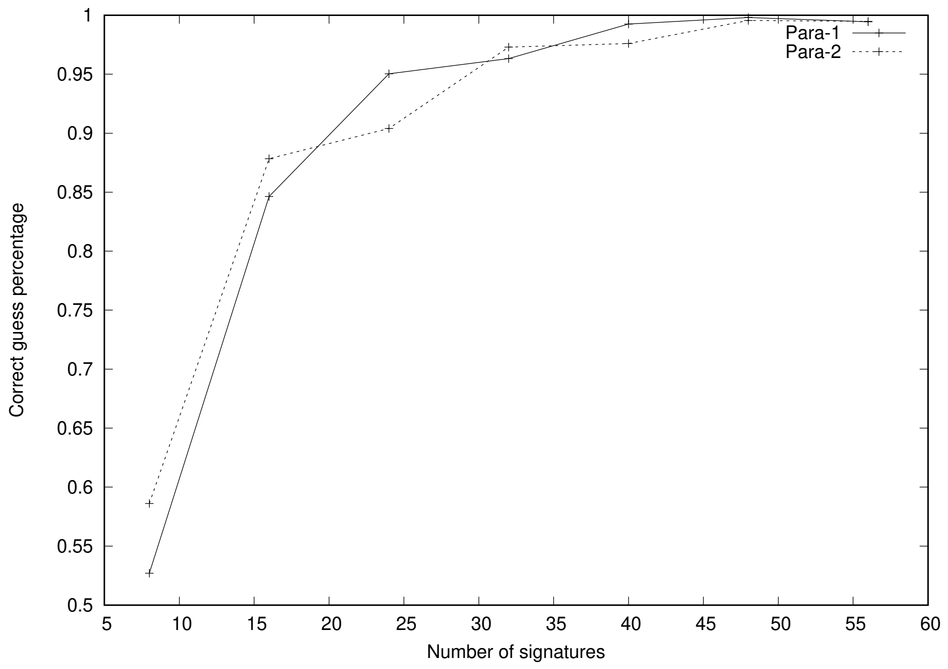

Figure 7.4 – Percentage of correct guesses with respect to the number of signatures available for the cryptanalysis.

Instance	Claimed Security	Number N of collected signatures	
		$N = 32$	$N = 64$
Para-1	80	2h07m17s	44m37s
Para-2	128	16h21m04s	5h21m40s

Table 7.4 – Experimental results for the cryptanalysis of the SHMWW signature scheme.

Cinquième partie

Implémentation de la métrique rang

Chapitre 8

Bibliothèque Rank-Based Cryptography (RBC)

Dans ce chapitre nous présentons la bibliothèque RBC (Rank-Based Cryptography), dont le but est de fournir une implémentation des opérations mathématiques fondamentales de la métrique rang ainsi que des cryptosystèmes basés sur la métrique rang. La version actuelle de la bibliothèque, disponible sur `https://rbc-lib.org`, fournit une implémentation de ROLLO [8] et RQC [4].

La bibliothèque est divisée en plusieurs parties :

— Une couche *coeur* fournissant les opérations permettant de manipuler les éléments, vecteurs, polynômes et espaces vectoriels sur $\mathbb{F}_{q^m}$.

— Une couche *code* fournissant les opérations nécessaires à l'utilisation des codes de Gabidulin et des codes LRPC en cryptographie.

— Une couche *schéma* fournissant les implémentations de ROLLO et RQC.

La bibliothèque s'adresse à deux différents types de public : d'une part, les utilisateurs qui souhaitent utiliser les schémas fournis par la bibliothèque, soit pour réaliser des tests (sur la performance, la sécurité...), soit pour intégrer les schémas dans d'autres logiciels, et d'autre part, des développeurs qui souhaitent utiliser les fonctions des couches *coeur* et *code* afin d'implémenter de nouvelles primitives en métrique rang.

Actuellement, la bibliothèque ne supporte que les corps de la forme $\mathbb{F}_{q^m}$ avec $q = 2$, ce qui couvre l'ensemble des paramètres de ROLLO et RQC. Les valeurs de m disponibles dans la bibliothèque sont actuellement limitées à celles utilisées dans ces deux schémas.

Afin de faciliter l'intégration de nouveaux algorithmes, la bibliothèque inclut un ensemble de scripts écrits en Python permettant d'écrire le code pour toutes les valeurs de m demandées par l'utilisateur au moment de la compilation. Cela permet de générer du code optimisé pour chaque valeur de m en évitant la duplication de code. Ce système permet aussi à l'utilisateur de spécifier l'architecture machine : par exemple, activer ou non l'utilisation d'instructions AVX.

Contents

8.1 Architecture de la bibliothèque

La bibliothèque implémente les types de données suivants :

— `rbc_elt` qui représente un élément de $\mathbb{F}_{q^m}$.

— `rbc_vec` qui représente un vecteur d'éléments de $\mathbb{F}_{q^m}$.

— `rbc_vspace` qui représente un sous-espace vectoriel de $\mathbb{F}_{q^m}$ sur $\mathbb{F}_q$.

— `rbc_poly` qui représente un polynôme à coefficients dans $\mathbb{F}_{q^m}$.

— `rbc_qre` qui représente un élément de $\mathbb{F}_{q^m}[X]/\langle P\rangle$.

Le type `rbc_elt` est représenté par un ensemble d'entiers non signés. Le nombre d'entiers nécessaire pour représenter un élément pour une valeur de m donnée dépend de l'architecture de la machine, suivant si on utilise des entiers de 32 ou 64 bits.

Les types `rbc_vec` et `rbc_vspace` sont représentés par des vecteurs de `rbc_elt`. L'espace vectoriel représenté par un ensemble d'éléments $\{x_1, \ldots, x_l\}$ est $\langle x_1, \ldots, x_l\rangle$: les espaces vectoriels sont donc représentés par des familles génératrices, et non pas nécessairement par des bases.

Le type `rbc_poly` contient un `rbc_vec` pour stocker les coefficients du polynôme, ainsi que le degré du polynôme. Le type `rbc_qre` est simplement un `rbc_poly` sur lequel l'ensemble des opérations sont réalisées modulo P.

8.2 Algorithmes

8.2.1 Algorithmes sur les `rbc_elt`

Les éléments de $\mathbb{F}_{2^m}$ sont représentés en base polynomiale dans la bibliothèque. $\mathbb{F}_{2^m}$ est défini comme $\mathbb{F}_2[X]/\langle\Pi\rangle$, où Π est un polynôme irréductible de degré m à coefficients dans $\mathbb{F}_2$, choisi de manière à avoir le moins de coefficients non nuls possible pour accélérer certains calculs.

Multiplication : L'opération de multiplication dans $\mathbb{F}_{2^m}$ encapsule une multiplication de polynômes ainsi qu'une réduction modulaire modulo le polynôme Π utilisé pour définir le corps fini $\mathbb{F}_{2^m}$. Pour réaliser la multiplication, deux techniques sont implémentées suivant les jeux d'instructions disponibles sur la plateforme. Si les jeux d'instructions CLMUL et AVX2 sont disponibles, la bibliothèque utilise un algorithme de multiplication simple, accéléré par l'instruction $\_mm\_clmulepi64()$. Autrement, la bibliothèque implémente l'algorithme 2.36 de [46] : un algorithme tirant partie du pré-calcul des différents produits $u(x) \times v(x)$ pour tous les polynômes $u(x)$ et $v(x)$ de degrés inférieurs à une constante w.

Inversion : l'inversion d'éléments de $\mathbb{F}_{2^m}$ est implémentée en utilisant une version de l'algorithme d'Euclide étendu.

Mise au carré : La mise au carré d'un élément $e = (e_0, \ldots, e_{m-1})$ est réalisée en insérant un certain nombre de zéros entre les coordonnées e_i afin d'obtenir un élément $e' = (e_0, 0, e_1, \ldots, e_{m-1})$ qui correspond au carré de e dans $\mathbb{F}_{2^m}$ après réduction modulaire. Si elle est disponible, l'instruction $\_mm\_unpacklo\_epi8()$ est utilisée pour accélérer le calcul : voir l'algorithme 1 de [17].

Réduction modulaire : La réduction modulaire exploite le fait que la bibliothèque utilise des polynômes Π ayant le moins de coefficients non nuls possible afin d'accélérer les calculs. L'algorithme de réduction est réécrit pour chaque valeur de m, étant donné que le polynôme Π est différent pour chaque m.

8.2.2 Algorithmes sur les `rbc_vec`

Le type `rbc_vec` est un type utilitaire principalement utilisé pour construire les types `rbc_poly` et `rbc_vspace`. Il est implémenté comme un vecteur de `rbc_elt` dont la taille est fixée à l'initialisation.

Calcul du rang : afin de calculer le rang d'un vecteur $\{x_1, \ldots, x_l\}$ on voit ce vecteur comme une matrice $M_x \in \mathbb{F}_2^{m \times l}$ en écrivant chacune des coordonnées dans la base polynomiale et on calcule le rang de M_x en utilisant l'algorithme de Gauss. Une implémentation de l'algorithme de Gauss en temps constant, inspirée de [22] mais adaptée pour fonctionner sur n'importe quelle matrice (y compris si la partie carrée à gauche de la matrice n'est pas inversible) est implémentée et décrite figure 8.1. La description contient des branchements conditionnels qui permettraient à un attaquant d'obtenir des informations en observant les accès à la mémoire. Afin de ne pas faire fuir d'information, ces branchements sont réalisés avec des masques. On décrit ce procédé pour l'exemple suivant :

— Si $\boldsymbol{M}[ligne][i] \mathrel{!}= \boldsymbol{M}[j][i]$ alors $\boldsymbol{M}[ligne] \mathrel{+}= \boldsymbol{M}[j]$

Afin de réaliser cette opérations sans branchement conditionnel, on commence par calculer un masque :

— $somme = \boldsymbol{M}[ligne] - \boldsymbol{M}[j]$

— $masque = somme[i]$

— $tmp = masque \times \boldsymbol{M}[j]$

Il ne reste ainsi plus qu'à réaliser l'addition avec tmp qui, si la condition n'était pas vérifiée, vaudra 0. De cette manière, que la condition soit vérifiée ou non, on accède aux mêmes coordonnées de $\boldsymbol{M}$ et on réalise le même nombre d'opérations :

— $\boldsymbol{M}[ligne] \mathrel{+}= tmp$

Génération de vecteurs aléatoires : Trois manières de générer des vecteurs aléatoires à coefficients dans $\mathbb{F}_{q^m}$ sont implémentés dans la bibliothèque :

1. La fonction `rbc_vec_set_random()` génère simplement chacune des coordonnées aléatoirement.

```
1: Entrée : Une matrice $\boldsymbol{M}$ de taille $l \times c$
2: Output : La matrice $\boldsymbol{M}$ échelonnée
3: $dimension = 0$
4: for $i$ de 0 à $c - 1$ do
5:     $ligne = min(dimension, l - 1)$
6:     for $j$ de 0 à $l - 1$ do
7:         if $j > ligne$ then
8:             if $\boldsymbol{M}[ligne][i] \mathrel{!=} \boldsymbol{M}[j][i]$ then
9:                 $\boldsymbol{M}[ligne] \mathrel{+=} \boldsymbol{M}[j]$
10:        else$\boldsymbol{D}[ligne] \mathrel{+=} \boldsymbol{D}[j]$
11:    for $j$ de 0 à $l - 1$ do
12:        if $j = ligne$ then
13:            $continue$
14:        if $dimension < l$ then
15:            if $\boldsymbol{M}[j](i]] = 1$ then
16:                $\boldsymbol{M}[j] \mathrel{+=} \boldsymbol{M}[ligne]$
17:        else
18:            $\boldsymbol{D}[j] \mathrel{+=} \boldsymbol{D}[ligne]$
19:    dimension $\mathrel{+=} \boldsymbol{M}[ligne][i]$
```

FIGURE 8.1 – Algorithme de Gauss implémenté en temps constant. Les branchements conditionnels sont réalisés avec des masques pour ne pas faire fuir d'information.

2. La fonction `rbc_vec_set_random_full_rank()` génère un vecteur aléatoire de rang plein. Pour ça, chacune des coordonnées est tirée aléatoirement et le rang du vecteur est ensuite vérifié. Le processus est répété jusqu'à ce qu'on obtienne un vecteur de rang plein.
3. La fonction `rbc_vec_set_random_from_support()` permet de générer un vecteur pour lequel chacune des coordonnées appartient à un espace vectoriel F de dimension d (le support F est un paramètre de la fonction). La fonction commence par copier les coordonnées de F à des coordonnées aléatoires du vecteur, afin de s'assurer qu'il soit de dimension d, puis remplit les coordonnées restantes d'éléments aléatoires de F.

8.2.3 Algorithmes sur les `rbc_poly`

Le type `rbc_poly` est implémenté comme une structure contenant un `rbc_vec` utilisé pour stocker les coefficients du polynôme, un entier qui garde en mémoire le degré actuel du polynôme, et une variable qui permet de garder en mémoire la taille du `rbc_vec` qui a été alloué. Si une opération nécessite un `rbc_vec` plus grand, il est automatiquement alloué.

Multiplication : la multiplication de deux polynômes est implémentée en utilisant l'algorithme de Karatsuba. Chaque niveau de récursion sépare les polynômes en deux polynômes de plus petits degrés, et on réalise une multiplication simple lorsque le degré des polynômes est au plus 1. L'implémentation est inspirée de celle réalisée dans la

```
1: **Entrée :** deux espaces vectoriels $A = \langle \boldsymbol{a}_1, \ldots, \boldsymbol{a}_d \rangle$ et $B = \langle \boldsymbol{b}_1, \ldots, \boldsymbol{b}_r \rangle$
2: **Sortie :** une base $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_l)$ de $A \cap B$
3: On construit la matrice $\begin{pmatrix} A & A \\ B & 0 \end{pmatrix}$
4: $\begin{pmatrix} U & * \\ 0 & V \\ 0 & 0 \end{pmatrix} = gauss \begin{pmatrix} A & A \\ B & 0 \end{pmatrix}$
5: **Retourner** $V$
```

FIGURE 8.2 – Description de l'algorithme de Zassenhaus

bibliothèque NTL [67].

Inversion : l'inversion de polynômes est implémentée en utilisant l'algorithme d'Euclide étendu.

8.2.4 Algorithmes sur les `rbc_vspace`

Un espace vectoriel est représenté par une famille génératrice, on utilise donc simplement un `rbc_vec` afin de stocker les éléments de $\mathbb{F}_{q^m}$ constituant cette famille génératrice.

Somme directe : pour calculer une famille génératrice de la somme directe de deux espaces vectoriels A et B, on concatène simplement leurs familles génératrices.

Produit : soient A et B deux espaces vectoriels générés respectivement par les éléments $\{a_1, \ldots, a_l\}$ et $\{b_1, \ldots, b_m\}$ respectivement. On calcule une famille génératrice $\{c_{1,1}, \ldots, c_{l,m}\}$ de l'espace vectoriel produit $C = \langle AB \rangle$ de la manière suivante :

$$c_{i,j} = a_i \times b_j \text{ } pour \text{ } i \in [1, l] \text{ } et \text{ } j \in [1, m]$$

Intersection : l'intersection d'espaces vectoriels est implémentée en utilisant l'algorithme de Zassenhaus, décrit figure 8.2. Afin d'implémenter cet algorithme en temps constant, il suffit d'utiliser une implémentation de l'algorithme de Gauss elle aussi en temps constant.

Base canonique : en cryptographie, les espaces vectoriels sont parfois utilisés comme des entrées pour des fonctions de hachage afin de dériver un secret partagé à partir de la connaissance d'un espace vectoriel : cela nécessite d'avoir une représentation canonique des espaces vectoriels. A un espace vectoriel F de famille génératrice $\{f_1, \ldots, f_d\}$ on associe une matrice $M_F \in \mathbb{F}_2^{d \times m}$ en dépliant chacun des éléments f_i dans la base polynomiale. La représentation canonique de l'espace vectoriel F est la forme échelonnée de la matrice M_F, calculée en utilisant l'algorithme de Gauss.

8.2.5 Algorithmes sur les codes de Gabidulin et codes LRPC

Gabidulin : le décodage des codes de Gabidulin implémenté dans la bibliothèque est celui proposé par Loidreau dans [53] et amélioré dans [19]. Cet algorithme utilise beaucoup d'opérations sur les q-polynômes, qui sont représentés par la structure `rbc_qpoly`. Cet algorithme est implémenté en temps constant.

LRPC : la bibliothèque n'implémente pas de structure particulière pour les codes LRPC, étant donné que les opérations d'encodage pour les codes LRPC idéaux peuvent être réalisées en utilisant les opérations sur les `rbc_qre`. L'algorithme de décodage implémenté est un algorithme qui permet seulement de récupérer le support de l'erreur, qui est utilisé en cryptographie pour dériver des secrets partagés. Deux versions de l'algorithme sont proposées, une optimisée pour les performances et une en temps constant, dans laquelle le nombre d'intersections et la taille des entrées fournies à l'algorithme de Zassenhaus sont fixés.

8.3 Performances

8.3.1 Comparaison avec les bibliothèques NTL, MPFQ et RELIC

Dans cette section nous allons comparer les performances de la bibliothèque RBC avec les bibliothèques $\texttt{mp}\mathbb{F}_q$ [44] et NTL [67] (qui étaient utilisées dans les premières versions des implémentations de ROLLO et RQC), ainsi qu'avec la bibliothèque RELIC [16], une bibliothèque proposant des briques de base pour l'implémentation de primitives cryptographiques, et notamment des implémentations d'opérations sur les corps finis.

Les opérations les plus critiques pour la performance des cryptosystèmes en métrique rang sont l'ensemble des calculs (multiplications et inversions) dans $\mathbb{F}_{2^m}^n$. Les tables 8.1, 8.2 et 8.3 présentent les performances des différentes bibliothèques sur les corps $\mathbb{F}_{2^{67}}^{83}$, $\mathbb{F}_{2^{79}}^{97}$ et $\mathbb{F}_{2^{97}}^{113}$ qui sont les corps utilisés pour les paramètres du cryptosystème ROLLO-I.

Ces résultats montrent que la bibliothèque est plus performante que NTL et RELIC sur l'ensemble des opérations considérées, mais aussi qu'elle est parfois moins performante que $\texttt{mp}\mathbb{F}_q$ pour les opérations dans le corps de base $\mathbb{F}_{2^m}$, en particulier la bibliothèque RBC est plus lente d'environ 20% pour l'inversion dans $\mathbb{F}_{2^m}$. Cela donne des pistes d'améliorations de performances pour la bibliothèque RBC.

Opération	RBC	$\texttt{mp}\mathbb{F}_q$	NTL	RELIC
Multiplication sur $\mathbb{F}_{2^{67}}$	60	32	448	1 175
Inversion sur $\mathbb{F}_{2^{67}}$	2 670	2 327	4 099	4 347
Mise au carré sur $\mathbb{F}_{2^{67}}$	60	32	406	208
Multiplication sur $\mathbb{F}_{2^{67}}^{83}$	73 821	3 220 639	316 721	-
Inversion sur $\mathbb{F}_{2^{67}}^{83}$	771 595	-	6 554 298	-

TABLE 8.1 – Performances des bibliothèques en cycles CPU sur $\mathbb{F}_{2^{67}}^{83}$

Opération	RBC	$\mathtt{mp}\mathbb{F}_q$	NTL	RELIC
Multiplication sur $\mathbb{F}_{2^{79}}$	31	32	218	1 161
Inversion sur $\mathbb{F}_{2^{79}}$	3 147	2 529	5 010	5 019
Mise au carré sur $\mathbb{F}_{2^{79}}$	31	32	159	167
Multiplication sur $\mathbb{F}_{2^{79}}^{97}$	79 367	4 801 501	370 442	-
Inversion sur $\mathbb{F}_{2^{79}}^{97}$	989 418	-	4 889 575	-

Table 8.2 – Performances des bibliothèques en cycles CPU sur $\mathbb{F}_{2^{79}}^{97}$

Opération	RBC	$\mathtt{mp}\mathbb{F}_q$	NTL	RELIC
Multiplication sur $\mathbb{F}_{2^{97}}$	32	32	225	1 094
Inversion sur $\mathbb{F}_{2^{97}}$	4 036	3 081	5 891	6 004
Mise au carré sur $\mathbb{F}_{2^{97}}$	32	32	187	166
Multiplication sur $\mathbb{F}_{2^{97}}^{113}$	87 471	7 607 949	353 243	-
Inversion sur $\mathbb{F}_{2^{97}}^{113}$	1 403 285	-	6 232 926	-

Table 8.3 – Performances des bibliothèques en cycles CPU sur $\mathbb{F}_{2^{97}}^{113}$

Conclusion et perspectives

Dans cette thèse nous avons commencé par présenter deux améliorations d'algorithmes de décodage en métrique rang. L'analyse de la complexité de l'amélioration de l'algorithme GRS a permis de mieux estimer la complexité des attaques combinatoires contre les cryptosystèmes basés sur les codes en métrique rang. L'amélioration de l'algorithme de décodage des codes LRPC permet quand à elle de trouver des paramètres proposant des tailles de clés plus faibles en gardant la même probabilité d'échec de décodage.

Nous avons ensuite présenté une nouvelle signature en métrique rang, qui permet de combler l'absence de schéma de signature suite à l'attaque sur RankSign.

L'amélioration de l'analyse de la distribution du poids des erreurs, ainsi que la proposition d'une nouvelle famille de codes (les Reed-Muller et Reed-Solomon concaténés), ont permis de proposer des tailles de clés plus compétitives pour ce cryptosystème, en affinant l'analyse de la probabilité d'échec de décodage.

Nous avons ensuite cryptanalysé deux adaptations du schéma de signature de Lyubashevsky : une en métrique rang, pour laquelle on peut retrouver la clé secrète à partir d'une seule signature en utilisant des propriétés des codes LRPC, et une en métrique de Hamming, pour laquelle nous avons exploité une fuite d'informations dans les signatures afin de retrouver la clé secrète grâce à un algorithme d'Information Set Decoding (ISD). Ces deux cryptanalyses montrent l'importance des preuves de sécurité réduisant la sécurité des cryptosystèmes à des problèmes clairement identifiés.

Enfin nous avons décrit les choix effectués lors de la conception de la bibliothèque Rank Based Cryptography (RBC). Cette bibliothèque a été utilisée dans le cadre des soumissions ROLLO et RQC à l'appel à projets du NIST, et nous avons réussi à obtenir de meilleures performances qu'en utilisant d'autres bibliothèques permettant de réaliser des calculs sur corps finis, grâce à des représentations de données et des choix d'algorithmes spécifiquement adaptés à la métrique rang.

Perspectives.

Métrique de Hamming. En métrique de Hamming, HQC a été accepté en temps que schéma alternatif pour le troisième tour de l'appel à projets du NIST. Les travaux présentés dans le chapitre HQC-RMRS ont été utilisés pour proposer de nouveaux jeux de paramètres et pour affiner l'analyse de la probabilité d'échec de décodage dans les nouvelles spécifications du troisième tour. La suite des travaux à réaliser concerne les implémentations sur des environnements contraints et l'étude des contremesures contre les attaques matérielles.

Métrique rang. Suite aux nouvelles attaques algébriques [20], les deux propositions basées sur la métrique rang encore en lice lors du second tour de l'appel à projets du NIST (ROLLO et RQC) n'ont pas été retenues pour le troisième tour. Le développement de ces cryptosystèmes reste toutefois un sujet de recherche intéressant, notamment parce que la métrique rang permet d'obtenir des tailles de clés très compétitives, y compris comparée à la cryptographie basée sur les réseaux euclidiens.

Dans ce contexte, plusieurs travaux doivent être effectués. Pour commencer, les paramètres des primitives existantes doivent être mises à jour : la mise à jour de Durandal notamment, permettra de proposer un ensemble schéma de chiffrement / schéma de signature entièrement basé sur la métrique rang. Du côté des implémentations, un travail rigoureux doit être effectué pour que l'ensemble des calculs soient réalisés à la fois en temps constant et sans branchement sur des données secrètes. Les prochaines versions de la bibliothèque RBC devraient également inclure de nouvelles primitives, notamment Durandal, ainsi que de nouvelles structures de données permettant par exemple la manipulation de matrices à coefficients dans $\mathbb{F}_2$ et $\mathbb{F}_{2^m}$. L'inclusion de nouvelles valeurs de q, différentes de 2, est également prévue.

Plusieurs primitives devraient également être développées pour compléter les possibilités offertes par la métrique rang : il serait par exemple intéressant de proposer une nouvelle signature de type "hash-and-sign" ou de réparer Ranksign, qui pourrait être utilisée pour réparer l'IBE présenté dans [36]. La métrique rang ne propose pas non plus de fonction de hachage, et l'approche utilisée en métrique de Hamming [18] ne semble pas pouvoir être facilement adaptée.

Une piste de recherche intéressante serait d'intégrer la métrique rang dans des applications réelles, par exemple en réalisant une implémentation dans SSL. Cela permettrait de mesurer de nouvelles métriques, comme l'impact des paramètres sur la latence du protocole, à la fois à cause des calculs réalisés par les deux parties (client et serveur) mais aussi à cause de la latence induite par le réseau. Cela permettrait également de mesurer l'impact du passage à des cryptosystèmes basés sur la métrique rang par rapport aux solutions actuellement utilisées.

Bibliographie

[1] Carlos Aguilar Melchor, Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Güneysu, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor. BIKE. First round submission to the NIST post-quantum cryptography call, November 2017.

[2] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. HQC, November 2017. NIST Round 1 submission for Post-Quantum Cryptography.

[3] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. Rank quasi cyclic (RQC). First round submission to the NIST post-quantum cryptography call, November 2017.

[4] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Gilles Zémor, Alain Couvreur, and Adrien Hauteville. Rank quasi cyclic (RQC). Second round submission to the NIST post-quantum cryptography call, April 2019.

[5] 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.

[6] Nicolas Aragon, Loic Bidoux, et al. Rbc-lib : Rank-based cryptography library, 2020. `http://rbc-lib.org/index.html`.

[7] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, and Gilles Zémor. LOCKER – LOw rank parity ChecK codes EncRyption. First round submission to the NIST post-quantum cryptography call, November 2017.

[8] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zémor, Carlos Aguilar Melchor, Slim Bettaieb, Loïc Bidoux, Bardet Magali, and Ayoub Otmani. ROLLO (merger of Rank-Ouroboros, LAKE and LOCKER). Second round submission to the NIST post-quantum cryptography call, March 2019.

[9] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Terry Shue Chien Lau, Chik How Tan, and Keita Xagawa. Cryptanalysis of a rank-based signature with short public keys. *Designs, Codes and Cryptography*, pages 1–11, 2019.

[10] Nicolas Aragon, Olivier Blazy, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. Durandal : a rank metric based signature scheme. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, volume 11478 of *LNCS*, pages 728–758. Springer, 2019.

[11] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, Oliver Ruatta, and Gilles Zémor. Ranksign – a signature proposal for the NIST's call. First round submission to the NIST post-quantum cryptography call, November 2017.

[12] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes : New decoding algorithms and application to cryptography. Available on http ://arxiv.org/abs/1111.4301, 2018.

[13] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes : New decoding algorithms and applications to cryptography. *IEEE Transactions on Information Theory*, 65(12) :7697–7717, 2019.

[14] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. Improvement of Generic Attacks on the Rank Syndrome Decoding Problem. working paper or preprint, October 2017.

[15] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. A new algorithm for solving the rank syndrome decoding problem. In *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*, pages 2421–2425. IEEE, 2018.

[16] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

[17] Diego F Aranha, Julio López, and Darrel Hankerson. Efficient software implementation of binary field arithmetic using vector instruction sets. In *International Conference on Cryptology and Information Security in Latin America*, pages 144–161. Springer, 2010.

[18] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A fast provably secure cryptographic hash function. IACR Cryptology ePrint Archive, Report2003/230, 2003. `http://eprint.iacr.org/`.

[19] Daniel Augot, Pierre Loidreau, and Gwezheneg Robert. Generalized Gabidulin codes over fields of any characteristic. *Designs, Codes and Cryptography*, 86(8) :1807–1848, 2018.

[20] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. An algebraic attack on rank metric code-based cryptosystems. In *Advances in Cryptology - EUROCRYPT 2020 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020. Proceedings*, 2020. to appear, preprint available on http ://arxiv.org/abs/1910.0081.

[21] 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.

[22] Daniel J. Bernstein, Tung Chou, and Peter Schwabe. McBits : fast constant-time code-based cryptography. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 250–272. Springer, 2013.

[23] Luk Bettale, Jean-Charles Faugere, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, 3(3) :177–197, 2009.

[24] Florent Chabaud and Jacques Stern. The cryptographic security of the syndrome decoding problem for rank distance codes. In *Advances in Cryptology - ASIACRYPT 1996*, volume 1163 of *LNCS*, pages 368–381, Kyongju, Korea, November 1996. Springer.

[25] Thomas Debris-Alazard and Jean-Pierre Tillich. A polynomial attack on a NIST proposal : Ranksign, a code-based signature in rank metric. preprint, April 2018. IACR Cryptology ePrint Archive.

[26] Thomas Debris-Alazard and Jean-Pierre Tillich. Two attacks on rank metric code-based schemes : Ranksign and an identity-based-encryption scheme. In *Advances in Cryptology - ASIACRYPT 2018*, volume 11272 of *LNCS*, pages 62–92, Brisbane, Australia, December 2018. Springer.

[27] Jean-Christophe Deneuville and Philippe Gaborit. Cryptanalysis of a code-based one-time signature. *Designs, Codes and Cryptography*, pages 1–10, 2020. `https://doi.org/10.1007/s10623-020-00737-8`.

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

[29] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium : A lattice-based digital signature scheme. *IACR TCHES*, 2018(1) :238–268, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/839`.

[30] 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, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

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

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

[34] 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.

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

[36] Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. Identity-based encryption from rank metric. IACR Cryptology ePrint Archive, Report2017/623, May 2016. `http://eprint.iacr.org/`.

[37] 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, Santa Barbara, CA, USA, August 2017. Springer.

[38] Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. Ranksynd a PRNG based on rank metric. In *Post-Quantum Cryptography 2016*, pages 18–28, Fukuoka, Japan, February 2016.

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] Pierrick Gaudry and Emmanuel Thomé. MPFQ, a finite field library, 2008. `https://mpfq.gitlabpages.inria.fr/`.

[45] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2) :281–308, 1988.

[46] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography.* Springer Science & Business Media, 2006.

[47] 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.

[48] Adrien Hauteville. Décodage en métrique rang et attaques sur un système de chiffrement à base de codes LRPC. Master's thesis, University of Limoges, September 2014.

[49] 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.

[50] 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.

[51] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *Theory of Cryptography Conference*, pages 341–371. Springer, 2017.

[52] Françoise Lévy-dit Vehel and Ludovic Perret. Algebraic decoding of codes in rank metric. communication at YACC06, Porquerolles, France, June 2006.

[53] Pierre Loidreau. A Welch–Berlekamp like algorithm for decoding Gabidulin codes. In *International Workshop on Coding and Cryptography*, pages 36–45. Springer, 2005.

[54] Pierre Loidreau. Properties of codes in rank metric, 2006.

[55] Pierre Loidreau. A new rank metric codes based encryption scheme. In *Post-Quantum Cryptography 2017*, volume 10346 of *LNCS*, pages 3–17. Springer, 2017.

[56] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–755. Springer, 2012.

[57] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error-correcting codes*, volume 16. Elsevier, 1977.

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

[59] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece : New McEliece variants from moderate density parity-check codes, 2012.

[60] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece : New McEliece variants from moderate density parity-check codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2069–2073, 2013.

[61] Oystein Ore. On a special class of polynomials. *Trans. Amer. Math. Soc.*, 35(3) :559–584, 1933.

[62] 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.

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

[64] Edoardo Persichetti. Efficient one-time signatures from quasi-cyclic codes : A full treatment. *Cryptography*, 2(4) :30, 2018.

[65] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2) :120–126, 1978.

[66] Peter W. Shor. Algorithms for quantum computation : Discrete logarithms and factoring. In *35th FOCS*, pages 124–134, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.

[67] Victor Shoup et al. NTL : A library for doing number theory, 2001. `https://www.shoup.net/ntl/`.

[68] Yongcheng Song, Xinyi Huang, Yi Mu, and Wei Wu. A new code-based signature scheme with shorter public key. Cryptology ePrint Archive, Report 2019/053, 2019.

[69] Yongcheng Song, Xinyi Huang, Yi Mu, Wei Wu, and Huaxiong Wang. A code-based signature scheme from the lyubashevsky framework. *Theoretical Computer Science*, 2020.

Cryptographie à base de codes correcteurs d'erreurs en métrique rang et applications

Résumé : La cryptographie basée sur les codes correcteurs d'erreurs est un des domaines permettant de construire des cryptosystèmes post-quantiques, c'est à dire résistants à l'ordinateur quantique. Contrairement à la factorisation et au logarithme discret, qui sont les deux problèmes les plus utilisés à l'heure actuelle en cryptographie, aucun algorithme n'est connu pour résoudre le problème de décodage de codes correcteurs aléatoires en temps polynomial avec un ordinateur quantique.

Dans cette thèse, on se concentre plus particulièrement sur la cryptographie basée sur la métrique rang, dans laquelle on étudie des codes correcteurs munis de la métrique rang plutôt que la métrique de Hamming. Cette métrique présente l'avantage de pouvoir construire des cryptosystèmes avec de plus petites tailles de clés, mais est moins mature que la métrique de Hamming.

Nous présentons dans un premier temps deux nouveaux algorithmes de décodage en métrique rang : le premier est un algorithme combinatoire permettant de résoudre le problème de décodage dans le cas de codes aléatoires, et permet donc de mieux estimer la complexité des attaques. Le second est une amélioration de l'algorithme de décodage pour les codes Low Rank Parity Check (LRPC). Nous présentons ensuite deux cryptosystèmes basés sur les codes : un schéma de signature en métrique rang, Durandal, qui est une adaptation de l'approche de Schnorr-Lyubashevsky en métrique euclidienne, et une amélioration du schéma de chiffrement Hamming Quasi-Cyclic (HQC) en métrique de Hamming, pour lequel on propose une nouvelle analyse du taux d'échec de déchiffrement et l'utilisation d'une autre famille de codes correcteurs d'erreurs.

Nous nous intéressons ensuite à deux adaptations de l'approche de Schnorr-Lyubashevsky, une en métrique de Hamming et l'autre en métrique rang, pour lesquelles on donne des cryptanalyses permettant de retrouver les clés publiques des schémas en utilisant la fuite d'information dans les signatures. Enfin nous présentons les choix effectués pour implémenter les cryptosystèmes en métrique rang dans la bibliothèque Rank-Based Cryptography (RBC).

Mots clés : Cryptographie, post-quantique, codes correcteurs d'erreurs, métrique rang, décodage, cryptanalyse.

Cryptography based on rank metric error correcting codes and applications

Abstract : Code-based cryptography is one of the fields allowing to build post-quantum cryptosystems, i.e secure against a quantum computer. Contrary to factorization and discrete logarithm, which are the two most used problems in cryptography right now, no algorithm is known to solve the decoding problem for random codes in polynomial time using a quantum computer.

In this thesis, we focus on rank-based cryptography, in which we study codes based on the rank metric instead of the Hamming metric. This metric has the advantage of allowing to build cryptosystems with lower key sizes, but is less mature than the Hamming metric.

Firstly, we present two new decoding algorithms in the rank metric : the first one is a combinatorial algorithm solving the decoding problem for random codes, hence allowing to better estimate the complexity of the attacks. The second one is an improvement of the decoding algorithm for Low Rank Parity Check (LRPC). We then present two code-based cryptosystems : a rank-based signature scheme which is an adaptation of the Schnorr-Lyubashevsky approach in the Euclidean metric, and an improvement of the Hamming Quasi-Cyclic (HQC) encryption scheme, for which we propose a new analysis of the decryption failure rate and the use of another family of error correcting codes.

We then study two adaptations of the Schnorr-Lyubashevsky approach : one in the Hamming metric and the other one in the rank metric, for which we propose cryptanalysis allowing to recover secret keys using information leakage from the signatures. Finally we present the choices used to implement rank-based cryptosystems in the Rank-Based Cryptography (RBC) library.

Keywords : Cryptography, post-quantum, error correcting codes, rank metric, decoding, cryptanalysis.