UNIVERSITÉ DE LIMOGES

ÉCOLE DOCTORALE Sciences et Ingénierie pour l'Information
FACULTÉ des SCIENCES et TECHNIQUES
Département de Mathématiques et Informatique
Laboratoire XLIM - UMR CNRS n°7257

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LIMOGES

Discipline : Informatique et Applications

présentée et soutenue publiquement par

Julien SCHREK

le 27 novembre 2013

Signatures et authentifications pour les cryptosystèmes basés sur les codes correcteurs en métrique de Hamming et en métrique rang

JURY

Rapporteurs :

Nicolas Sendrier	Directeur de Recherche, INRIA Rocquencourt
Pierre Loidreau	Chercheur DGA-IM, titulaire d'une HDR

Examinateurs :

Ayoub Otmani	Professeur à l'Université de Rouen (27ème section)
Jean-Pierre Tillich	Directeur de Recherche, INRIA Rocquencourt (27ème section)
Thierry Berger	Professeur à l'Université de Limoges (25ème section)

Directeur :

Philippe GABORIT	professeur à l'université de Limoges (27ème section)

Table des matières

Remerciements

Je remercie en premier lieu Philippe Gaborit, qui m'a donné la possibilité de préparer cette thèse et faire ainsi mes premiers pas dans la recherche.
Je remercie ensuite l'Université de Limoges pour m'avoir donné les moyens de faire cette thèse sur quatre ans.
J'adresse ensuite mes remerciements à Pierre Loidreau et Nicolas Sendrier pour avoir accepté de rapporter ma thèse.
J'adresse aussi mes remerciements aux examinateurs Thierry Berger, Ayoub Otmani et Jean-Pierre Tillich pour avoir accepté d'évaluer mon travail.
Pour finir, je remercie mes coauteurs Carlos Aguilar-Melchor, Slim Bettayeb, Olivier Ruatta, Gilles Zémor sans qui cette thèse n'aurait pas été aussi riche.

Chapitre 1

Introduction

La cryptographie est souvent désignée comme la science du secret. On peut trouver ses origines dans la volonté de cacher de l'information dans les communications qu'entretenait Jules César avec ses généraux. Déjà en -40 avant J.C. il modifiait l'ordre des lettres de ses messages pour qu'ils ne soient compréhensibles que par ceux qui étaient dans le secret de cette manipulation. La cryptographie a continuée a être utilisée majoritairement par les militaires jusqu'à la deuxième guerre mondiale. C'est à cette époque qu'elle s'est grandement développée, notamment dans les recherches des cryptologues britanniques pour décrypter les communications de l'Allemagne nazie chiffrées par la machine Enigma.

De nos jours, la cryptographie est très présente dans le monde civil. Le monde est devenu extrêmement informatisé et la cryptographie sert à sécuriser les communications informatiques. On la retrouve dans les communications bancaires, dans les téléphones portables, dans les cartes à puces ou dans les communications internet sécurisées.

En mai 1973, le *National Bureau of Standard* demande la création d'un algorithme de chiffrement sûr et efficace. C'est l'algorithme Lucifer, créé par IBM qui fut retenu. On le connait aujourd'hui sous le nom de DES. Le DES devenant obsolète de nos jours, il ne devrait plus être utilisé. C'est maintenant l'AES qui a prit sa place.

En 1976, sort l'article "New direction in Cryptography" par W.Diffie et M.Hellman ([30]), qui va apporter une énorme ouverture d'esprit aux cryptographes de l'époque. Il y évoque la possibilité de pouvoir chiffrer sans pour autant pouvoir déchiffrer. Jusqu'alors, le chiffrement était symétrique, la même clé servait à chiffrer et à déchiffrer. Dans cette article, on propose d'utiliser une clé publique pour chiffrer et une clé privée pour déchiffrer. Cette nouvelle façon de voir le chiffrement, appelée chiffrement asymétrique apporte beaucoup à la cryptographie. Le chiffrement asymétrique est très souvent illustré par l'algorithme de chiffrement RSA, qui a été le premier à être trouvé en 1978 par A.Rivest, A.Shamir et L.Adelman dans [84]. Il repose sur l'arithmétique modulaire. Plusieurs notions sont apparues depuis qui ont enrichi le domaine d'expertise des cryptographes. La cryptographie ne

sert plus seulement au chiffrement mais aussi à la confidentialité en général, à l'authentifiction d'une personne ou d'un bien, à l'intégrité et à la non-répudiation de l'information ou à l'anonymat d'un acteur d'un protocole cryptographique. A l'image du chiffrement RSA, ces propriétés se sont souvent concrétisées par des cryptosystèmes reposant sur la théorie des nombres et se basant sur le problème de factorisation ou celui du logarithme discret. La cryptographie reposant sur les problèmes de théorie des nombres étant la plus étudiée, on peut supposer qu'elle est la plus sûre. Cependant il suffit que l'un de ces problèmes difficiles soit attaquable pour que tous ces systèmes deviennent obsolètes. L'exemple le plus marquant étant l'algorithme de Shor qui permettrait de résoudre le problème de factorisation en temps polynômial sur un ordinateur quantique.

Toujours dans le domaine des communications, en 1960, Hamming découvre un moyen efficace de détecter des erreurs pouvant s'être infiltrées entre une source et un destinataire. Il ajoute du contenu à son message sous forme de redondance afin qu'une faible perturbation de celui-ci n'altère pas la quantité d'information transmise. L'algorithme servant à coder les messages s'appelle un code correcteur. Depuis, la théorie des codes correcteurs a énormément évoluée et elle se révèle être indispensable lors d'un échange d'informations sur un canal bruité. De grandes familles de codes efficaces ont été trouvées et étudiées. On travaille aussi sur des modèles de l'erreur différents, comme avec l'utilisation de la métrique rang qui permet d'imaginer des codes corrigeant des erreurs intervenant sur l'ensemble des composantes du message au lieu d'un nombre faible de ces composantes comme c'est le cas le plus souvent. On retrouve des codes correcteurs dans toutes les télécommunications (téléphoniques, informatiques, ...), les disques compactes et DVD, les différentes mémoires d'un ordinateur (RAM, disque dur, ...), les codes barres et autres supports.

En 1978, McEliece présente le premier cryptosystème asymétrique dont la sécurité est basée sur un problème considéré comme difficile dans la théorie des codes correcteurs dans [69]. Il s'agit d'un algorithme de chiffrement asymétrique transformant un message de taille fixe en un mot de code bruité. Seul quelqu'un qui connait la structure exacte du code et son algorithme de décodage est sensé pouvoir déchiffrer le message. Cet algorithme se déduit naturellement de la structure du code qui elle est cachée par des masques, constituant la clé secrète. Grâce à ce système, on peut chiffrer un message en utilisant le code. Ce code correspond à la clé publique. De plus, quelqu'un qui connait la clé privée peut décoder un mot bruité et ainsi déchiffrer pour retrouver le message initial. L'avantage de ce système est qu'il n'utilise que des opérations simples pour fonctionner, il est donc rapide par rapport à RSA qui doit faire des exponentiations sur des grands nombres. Son inconvénient est d'utiliser des données de grandes tailles, les codes utilisés sont représentés par des matrices de plusieurs millions de coordonnées. On retrouve ce genre de propriétés sur les autres cryposystèmes basés sur la théorie des codes correcteurs proposés jusqu'ici. Citons le schéma d'authentification de Stern, la signature CFS [28] ou le schéma de signature KKS [60]. L'autre avantages de tous ces cryptosystèmes est qu'ils sont *a priori* résistants à l'ordinateur quantique. Si les ordinateurs quantiques devenaient plus performants, les cryptosystèmes basés sur la théorie des nombres deviendraient obsolètes alors que ceux basés sur la théorie des codes correcteurs resteraient aussi sûrs *a priori*.

Dans cette thèse on s'intéresse à l'étude des signatures et authentifications en métrique de Hamming et en métrique rang. Elle a donné lieu à 5 articles publiés [1], [51], [50], [8], [5] et 3 soumissions [49], [7] et [8]. Le dernier article, qui concerne l'implantation de la version améliorée du protocole de Stern décrite dans le chapitre 5, ne sera pas détailée dans cette thèse. Nous ne nous intéressons dans cette thèse qu'aux travaux concernant la métrique de Hamming et la métrique rang.

La première partie regroupe les travaux effectués sur la métrique de Hamming. Elle se compose de 7 chapitres. Les 3 premiers chapitres introduisent la cryptographie basée sur les codes correcteurs en rappelant la théorie des codes correcteurs et les différents cryptosystèmes se basant sur le problème de décodage par syndrome.
Le chapitre 5 décrit l'étude fait avec Carlos Aguilar Melchor et Philippe Gaborit dans [1] en introduisant une amélioration du protocole d'authentification zero-knowledge de Stern. Cette amélioration est possible grâce à l'utilisation de codes quasi-cycliques et donne une taille de signature de 93kb.
Dans chapitre 6, on présente un travail fait en collaboration avec Philippe Gaborit et qui a donné lieu à l'article suivant [50]. Il s'agit d'une signature à usage unique basé sur la théorie des codes correcteurs. Elle suit la même idée de construction que la signature de KKS mais repose sur une preuve et une idée de sécurité différente. Le problème de la signature à usage unique de KKS est quelle donne une grosse matrice de syndrome. Dans notre cas, la matrice est beaucoup plus petite car le protocole utilise des codes avec des gros groupes de permutation pour compenser cette taille. Cela permet aussi de faire une preuve de sécurité se basant sur l'entropie.
Le chapitre 7 présente une signature d'anneau à seuil basée sur les codes correcteurs. Il représente la version code de la signature présentée dans [8] avec Slim Bettaieb qui est basée sur les réseaux. Cette signature donne, dans la pratique, des tailles de signature plus petite que l'autre signature à seuils sur les codes présenté dans [2].
Le chapitre 8 est le dernier sur la métrique de Hamming et présente une signature indéniable sur les codes correcteurs. Ce travail a été réalisé avec Carlos Aguilar Melchor, Slim Bettaieb et Philippe Gaborit et a fait l'objet d'un papier dans [5]. Il y présente la première signature indéniable post-quantique.

La deuxième partie de cette thèse regroupe les travaux effectués sur la métrique rang. Les chapitres 9 et 10 rappellent quelques propriétés de la métrique rang, les cryptosystèmes basé sur les codes correcteurs en métrique rang ainsi que les meilleurs attaques sur le problème de décodage par syndrome en métrique rang.
Le chapitre 11 présente une nouvelle attaque sur le problème de décodage par syndrome. Cette attaque a été réalisée avec Philippe Gaborit et Olivier Ruatta et est en cours de soumission à IEEE Information Theory. Elle a permit de casser la plupart des paramètres connus de tous les cryptosystèmes se basant sur ce problème.
Dans le chapitre 12, on décrit une attaque sur le protocole d'authentification zero-knowledge de K.Chen réalisée avec Philippe Gaborit et Gilles Zémor dans [51]. Le protocole de K.Chen a comme avantage d'utiliser des fonction linéaires lors de ses mises en gage. Nous décrivons dans ce chapitre

une faille de sécurité sur ces mises en gage ainsi qu'une autre faille au niveau du masquage du secret. Il est aussi décrit dans ce chapitre une réparation de ce protocole.
Dans le chapitre 13, on présente une nouvelle façon de signer avec des codes décodables. Les codes utilisés sont en métrique rang et permettent d'obtenir des tailles de signatures de 1728 bits pour une clé publique de 5760 bits et une sécurité en 2^{90}. Ce travail, réalisé avec Philippe Gaborit, Olivier Ruatta et Gilles Zémor est soumis à Eurocrtpy 2013.

Métrique de Hamming

Chapitre 2

Rappel sur la cryptographie

La cryptographie est souvent décrite comme étant la science du secret. Elle se compose de la cryptographie et de la cryptanalyse. La cryptographie étant la science permettant l'élaboration d'algorithmes permettant la confidentialité, l'authentification, l'intégrité, la non-répudiation ou l'anonymat. Ces algorithmes devant être justifiés difficile à casser. La cryptanalyse consiste à étudier et attaquer les algorithmes que proposent les cryptographes.

C'est en 1883, par Auguste Kerckhoffs, que se formalise le principe qu'un cryptosystème doit être connu de la communauté pour pouvoir justifié de sa sécurité. La sécurité de l'algorithme doit reposer sur non connaissance de sa clé et non pas de lui même.

2.1 Primitives cryptographiques

Les trois principales primitives cryptographiques sont le chiffrement, l'authentification et la signature numérique.

2.1.1 Chiffrement

Le chiffrement consiste à permettre la transmission d'un message entre deux personnes à travers un canal sans que quelqu'un qui aurait accès aux transmissions de ce canal puisse avoir de l'information sur ce message.
Il existe deux types d'algorithmes de chiffrement :

Le chiffrement symétrique Ce sont les premiers cryptosystèmes utilisés d'un point de vue historique. Ils sont caractérisés par le fait que les deux protagonistes possèdent la même clé secrète et peuvent à la fois chiffrer et déchiffrer. Parmi les algorithmes modernes, le premier chiffrement symétrique à avoir été utilisé de façon standard a été le DES. Ils sont en général utilisés pour leur vitesse de chiffrement mais possèdent le gros inconvénient de requérir un nombre de clés secrètes linéaire en le nombre de personnes avec lesquelles on veut communiquer.

Le chiffrement asymétrique Il a été imaginé par W.Diffie et M.Hellman en 1976 dans [30] pour palier le défaut du chiffrement symétrique. Avec le chiffrement asymétrique, une seule clé secrète est nécessaire pour communiquer de manière sécurisée avec un nombre quelconque de personnes. Ces algorithmes sont en général plus lent que les algorithmes de chiffrement symétrique. Ils sont souvent basés sur des problèmes reconnus difficiles par la communauté mathématique.

2.1.2 Authentification

L'authentification est une notion incontournable dans un réseau informatique multi-utilisateur car elle implique de savoir prouver son identité. Pour accéder a des données personnelles ou à l'intérieur d'un protocole cryptographique, il peut être intéressant de prouver son identité. L'identité est associée à une donnée publique et l'authentification nécessite une donnée secrète. On dénote deux types d'authentification, selon le modèle de sécurité demandé. Le premier modèle demande que l'authentification soit suffisamment difficile à partir de la donnée publique seule, il est mis en application par les fonctions de hachage. Le deuxième demande en plus que si l'authentification se déroule à travers un canal non sécurisé, quelqu'un qui a accès aux transmissions sur ce canal ne doit pas être plus aidé pour s'authentifier sans connaitre la donnée secrète. Le deuxième modèle peut être mit en pratique à l'aide de protocoles à divulgation nulle de connaissance.

Fonctions de hachage

Les fonctions de hachages sont beaucoup utilisées en informatique. Elles servent à donner une empreinte d'un document, ou simplement d'un mot. Cela permet de traiter des données beaucoup plus petite. En cryptographie, les fonctions de hachage sont plus contraintes, pour plus de fonctionnalité.

Définition 2.1. On dit qu'une fonction de $\{0,1\}^*$ dans $\{0,1\}^n$ est une fonction de hachage lorsqu'elle vérifie les propriétés suivantes :

- L'antécédent d'une image donnée est difficile à obtenir. **(attaque sur la première préimage)**
- Il est difficile de trouver un mot qui est la même image qu'un autre mot donné. **(attaque sur la second préimage)**
- Il est difficile de trouver 2 mots qui ont les mêmes images **(résistance aux collisions)**

Les fonctions de hachage peuvent être vue comme des fonctions à sens unique. De cette façon ont peut les utiliser pour l'authentification. Par exemple, le mot de passe sur un ordinateur est stocké sous

forme de haché. Dans cette thèse nous les rencontrerons dans les algorithmes d'authentification pour créer des mises en gage et dans les algorithmes de signature pour créer une empreinte du message à signer.

Authentification avec divulgation nulle de connaissance :

Les preuves à divulgation nulle de connaissances, dites "zero-knowledge" ont été introduite par Goldwasser et al en 1989 dans [56]. L'idée étant de prouver à une autre personne la connaissance d'un secret sans révéler d'informations sur celui-ci. Ce procédé convient très bien pour l'authentification car le secret peut être vu comme une clé privée. Ainsi, la clé publique identifie la personne et la preuve à divulgation nulle de connaissances correspond à la preuve qu'une personne prétende être qui elle est.

Idée :

Ce processus fonctionne avec des challenges et des mises en gages. L'idée est de voir le secret comme une association de plusieurs secrets indépendants du secret initial mais très dépendants entre eux. La donnée de l'un de ces secrets ne donne pas d'information sur le secret initial. La connaissance de tous les nouveaux secrets correspond à la connaissance du secret initial. Ainsi, la méthode pour prouver la connaissance du secret est de demander à dévoiler l'un des nouveaux secrets pour essayer de piéger le prouveur. Ce procédé ne fonctionne qu'avec une phase de vérification qui doit associer plusieurs nouveaux secrets à une seule mise en gage.
Formellement, avec s la clé secrète et p la clé publique, la création de nouveaux secrets se fait à l'aide de masques, ici m. On peut ainsi voir la connaissance de s comme la connaissance de $m(s)$ et de m. Pour que la vérification fonctionne, le masque m doit être compatible avec la transformation de s à p. Ainsi le schéma suivant correspond à une preuve à divulgation nulle de connaissance :

- Mise en gage de m.
- Challenge de révélation de m ou de $m(s)$.
- Réponse en révélant m ou $m(s)$.
- Vérification

Le fait de procéder par challenge implique une probabilité que le challenge ait pu être anticipé et qu'une triche puisse être possible. Pour diminuer cette probabilité de triche il suffit de répéter le protocole.

exemples :

A.Fiat et A.Shamir, en 1987 dans [39]. Le secret est un nombre x dans $\mathcal{Z}_{pq}$ dont le carré y est la clé publique. Le masque est la multiplication par un nombre r de $\mathcal{Z}_{pq}$. Ainsi, la connaissance de x correspond à la connaissance de xr et r. La mise en gage se fait en donnant r^2 et la vérification se fait en notant l'égalité entre r^2 et $(xr)^2$ divisé par y. Le schéma devient :

- envoie de r^2
- demande d'envoi de xr ou r.
- Réponse en révélant xr ou r.
- Vérification avec $(xr)^2/y = r^2$ ou $r^2 = r^2$.

Un autre exemple, celui ci avec plusieurs masques est celui de A.Shamir en 1989 dans [88]. Ce protocole se base sur le problème difficile PKP, prouvé NP-Complet. Soit h une fonction de hachage. Ce protocole à pour clé privée une permutation π et pour clé publique une matrice A et un vecteur v. Le secret π est tel que $A\pi(v) = 0$. Le premier masque est une permutation σ et le deuxième est un mot aléatoire y.

- envoie de $h(\sigma|Ay) = c_1$ et $h(\pi \circ \sigma|\sigma(y)) = c_2$
- premier challenge α
- envoie de $w = \sigma(y + \alpha\pi(V))$
- deuxième challenge $b = 0$ ou $b = 1$
- Si $b = 0$, envoie de σ. Si $b = 1$, envoie de $\sigma \circ \pi$.
- Vérification :
  Si $b = 0$: vérifier si $h(\sigma|A\sigma^{-1}(w)$ est égale à c_1
  Si $b = 1$: vérifier si $h(\sigma \circ \pi|w - \alpha\sigma \circ \pi(v))$ est égale à c_2

2.1.3 Signature

La signature numérique à été introduite par Whitfield Diffie et Martin Hellman dans [30], en même temps que la cryptographie à clé publique. Le but est de reproduire la signature écrite en y améliorant les propriétés. En effet, il s'agit maintenant de prouver le lien entre un document signé et une personne. La définition usuelle de la signature numérique est due a Goldwasser, Micali et Rivest dans [57]. Cette définition permet d'obtenir des signatures résistantes aux attaques adaptatives par messages choisis.

La signature numérique permet d'associer un message et une identité. L'identité est représenté par une clé publique et une clé privée est nécessaire pour créer la signature. Elle doit pouvoir être vérifier par n'importe qui en utilisant un algorithme de vérification. A l'image de la signature classique, elle garantie l'intégrité d'un document et son approbation par une personne identifiable.

Heuristique de Fiat-Shamir :

A.Fiat et A.Shamir ont montré dans [39] que l'on peut transformer un algorithme d'authentification à divulgation nulle de connaissance en un algorithme de signature.
Cette méthode d'authentification s'opère à l'aide de challenges et de réponses entre un prouveur et un vérifieur. L'idée pour transformer un tel procédé en signature est de remplacer les challenges par des valeurs aléatoires de loi uniforme, dans l'ensemble des challenges, dépendant uniquement du

message à signer. Concrètement, un haché du message permettra de simuler le challenge du vérifieur dans la première étape du protocole.

Chapitre 3

Rappel sur la théorie des codes correcteurs

Les premiers codes correcteurs furent crées en 1940 par Richard Hamming. Ils avaient pour but de résoudre les erreurs physiques qui pouvaient intervenir sur les calculateurs à carte perforée sur lesquels il travaillait.
L'idée fut de rajouter de l'information redondante à la fin d'un mot pour pouvoir détecter les erreurs qui auraient pu s'y introduire. De nos jour, on trouve des codes dans différents tous les domaines liés à l'information :

- Les outils de télécommunication
- Mémoire RAM
- Les disques compactes, DVD
- codes barres

3.1 Définitions

Les codes correcteurs sont utilisés pour coder de l'information. Pour cela, l'information à besoin d'être décrite de façon rigoureuse, c'est pourquoi on introduit la notion de mot et d'alphabet avant de coder.

Définition 3.1. (Alphabet et mot)
Un alphabet A est un ensemble de symbole non vide. Un mot est un élément de A^*.

En général les alphabets utilisés sont des corps finis, il peut aussi s'agir d'anneaux.

Définition 3.2. (Code correcteur)
Un code correcteur $\mathcal{C}$ est un sous ensemble d'un espace de mot sur un alphabet.

Les codes les plus utilisés sont les codes linéaires, car leur structure est très naturelle et pratique pour représenter les mots.

Définition 3.3. (Code linéaire)
Un code correcteur linéaire est un code correcteur dont l'ensemble des mots forme un sous espace vectoriel.

Dans cette thèse, nous parlerons toujours de code correcteurs linéaire sans préciser d'avantage. Lorsque l'alphabet d'un code est $\{0, 1\}$, ce code est dit binaire.
On introduit une métrique sur ces codes linéaires.

Définition 3.4. Métrique de Hamming
La métrique de Hamming est une métrique sur les espace vectoriels vus en tant qu'espaces produits. La distance entre x et y est le nombre de coordonnées non nulles de $x - y$.

Avec cette métrique on peut donner une valeur à chaque mot du code.

Définition 3.5. Poids de Hamming
Le poids de Hamming de x est la distance de x à 0. Il sera noté wt, $wt(x) = d(x, 0)$.

De façon plus globale, on peut associer une valeur au code qui dépend de la valeur de l'ensemble de ces éléments. La distance minimale est très utilisée pour caractériser un code, elle donne une information globale sur le poids de ses éléments.

Définition 3.6. (Distance minimale)
La distance minimale d d'un code $\mathcal{C}$ est la plus petite distance entre deux mots du code. $d = Inf(d(x, y))$ avec $x \in \mathcal{C}$ et $y \in \mathcal{C}$.

Pour les codes linéaires, avec la métrique de Hamming, la distance minimale est le poids du mot de plus petit poids.
Les autres caractéristiques les plus utilisées en théorie des codes sont la longueur et la dimension d'un code.

Définition 3.7. Longueur et dimension
La longueur d'un code linéaire est le nombre des composantes des vecteurs du code. La dimension d'un code linéaire est sa dimension en tant qu'espace vectoriel.

On peut remarquer que la longueur est toujours plus grande que la dimension.

Définition 3.8. On dit qu'un code linéaire est $[n, k, d]$ si il est de longueur n, de dimension k et que sa distance minimale est d.

3.1.1 Représentation des codes par des matrices

Un code linéaire $[n, k, d]$ peut être représenté par une matrice G de taille $k \times n$. Une telle matrice est appelée matrice génératrice. De la même façon, un code possède un code dual, qui est le dual en tant qu'espace vectoriel. Ce dual peut aussi être représenté par une matrice.

Définition 3.9. Matrice génératrice et matrice de parité
Une matrice génératrice de $\mathcal{C}$ est une matrice génératrice du code en tant qu'espace vectoriel.
Une matrice de parité de $\mathcal{C}$ est une matrice génératrice du code dual de $\mathcal{C}$.

La matrice de parité est aussi appelée matrice de contrôle. La matrice génératrice et la matrice de parité permettent de définir un code de façon unique. Toutefois il existe beaucoup de matrices génératrices ou de parité pour représenter un même code. Dans la pratique, on choisit lorsqu'il existe, un représentant unique plus simple. Il s'agit de la matrice sous forme systématique. Elle est composée d'un bloc identité sur ses premières colonnes.

Définition 3.10. (Forme systématique)
Une matrice $k \times n$ est sous forme systématique lorsque les k premières colonnes forment une matrice identité.

Pour trouver une matrice équivalente systématique il suffit d'utiliser un pivot de gauss.
Si G est une matrice génératrice de $\mathcal{C}$ sous forme systématique, avec $G = (Id|A)$. La matrice $H = (A^T|Id)$ est une matrice duale de $\mathcal{C}$.

3.1.2 Encodage, décodage, détection et correction

Le but d'un code correcteur est de détecter ou de corriger des erreurs qui auraient pu survenir lors d'une transmission d'information. Concrètement, cette modification supposée inconnue intervient entre les phases d'encodage et de décodage.

Définition 3.11. (Encodage)
L'encodage correspond à faire correspondre un mot à un mot du code. Pour les codes linéaires, cela correspond à multiplier un mot par la matrice génératrice du code.

Pour un code $\mathcal{C}$ de paramètre $[n, k, d]$, on encode un mot à k coordonnée en un vecteurs du code à n coordonnée. Avec $x \in A^k$ et G la matrice génératrice de $\mathcal{C}$, l'encodage de x est xG.

Définition 3.12. (Décodage)
Le décodage correspond à faire correspondre un mot de la longueur du code en un mot encodable unique.

Trouver un algorithme de décodage efficace est un grand challenge de la théorie des codes. Les codes sont souvent structurés de façon à ce qu'il existe un algorithme de décodage efficace. Lorsque le décodage correspond à la fonction inverse du codage, on peut parler de détection d'erreurs.
Pour que la correction soit unique, on comprend facilement que les codes ne peuvent ni détecter ni corriger un nombre possible d'erreurs trop important. Pour caractériser les erreurs qui peuvent être détectée ou corrigée de celle qui ne le peuvent pas, on fixe une métrique sur l'ensemble des erreurs. Dans la première partie de cette thèse, on utilisera la métrique de Hamming. Elle permet le plus souvent de simuler les erreurs qui peuvent intervenir lors des communications. Ces erreurs sont très ponctuelles, donc n'interviennent que sur une seule coordonnée du code. Dans la deuxième partie de cette thèse on utilisera une autre métrique, la métrique rang. **Exemple de décodage intuitifs :**

- Recherche exhaustive du mot du code le plus proche du mot à décoder.
- Recherche exhaustive du mot le plus petit qui transforme le mot à décoder en un mot du code.

Capacité de correction : On appelle capacité de correction l'entier t égale à $\frac{d-1}{2}$. Un mot de distance supérieur à $\frac{d}{2}$ ne pourra pas être décodé dans tous les cas car il peut être à égale distance de plusieurs mots qui sont les plus proches de lui dans le code.
Décodage par syndrome : Lorsque l'on utilise les codes linéaires, avec G une matrice génératrice du code et m le mot à encoder, le mot encodé correspond à mG. Si on rajoute l'erreur e, le mot bruité est $mG + e$. Dans ce cas, on peut multiplier ce mot par une matrice duale H pour obtenir $H(mG + e)^T = He^T$. Cette valeur s'appelle le syndrome de e, elle ne dépend que de l'erreur contrairement au mot bruité qui dépend aussi du mot d'origine. La construction des syndromes permet d'avoir dans les codes linéaires des algorithmes de décodages qui ne peuvent pas exister dans les codes non linéaires.

3.2 Bornes sur les codes

Dans cette section on considère un code $\mathcal{C}$ de paramètre $[n, k, d]$. On appelle $\mathcal{A}_q(n, d)$ le nombre maximum de mots que peut contenir un code de paramètres $[n, k, d]$.

3.2.1 Borne de Hamming

Nous avons la borne suivante :

Théorème 3.13. (Borne de Hamming) Soit $t = \lfloor \frac{d-1}{2} \rfloor$, on a la borne suivante :

$$B_q(n, d) \leq A_q(n, d) \leq \frac{q^n}{\Sigma_{i=0}^{t} \binom{n}{i} (q-1)^i}.$$

Un code qui vérifie l'égalité de cette borne est appelé code parfait.

3.2.2 Borne de Singleton

La borne de singleton donne une majoration de la distance minimale d'un code linéaire.

Théorème 3.14. (Borne de Singleton) La distance minimale du code $\mathcal{C}$ vérifie :

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

Le nombre d'erreurs décodables est toujours inférieur à $\lfloor \frac{d-1}{2} \rfloor$ avec d la distance minimale du code. Un code qui atteint cette borne est dit MDS (distance maximum séparable).

3.2.3 Borne de Gilbert-Varshamov

Cette borne a été introduite par Gilbert et Varshamov de façon indépendante en 1952.

Théorème 3.15. (Borne de Gilbert-Varshamov) Les paramètres du code $\mathcal{C}$ vérifient :

$$\frac{q^n}{\Sigma_{i=0}^{d-1} \binom{n}{i} (q-1)^i} \leq A_q(n, d).$$

Il a été prouvé que les codes binaires définis par des matrices dont les coefficients sont aléatoires vérifient cette borne inférieur presque tout le temps.

3.3 Exemples de codes :

Le premier exemple est le code à répétition. Dans le cas d'une seule répétition, il permet de détecter un nombre impair d'erreurs, il peut donc détecter à coup sûr si le poids de l'erreur est au plus égale à 1.

3.3.1 Code à Répétition

Ce code utilise un codage par bloc, les mots à coder sont de taille fixe. Pour encoder un mot il suffit de le concaténer à lui même. Par exemple le mot *abcd* se code en *abcdabcd*.

3.3.2 Code de Hamming binaire

Les codes de Hamming binaires sont des codes 1-correcteurs à redondance minimale. Un mot $c_1 \dots c_n$ d'un code de Hamming binaire est tel que les bits c_i dont l'indice i est une puissance de 2 (2^l avec $l = 0, 1, \dots$) sont des bits de contrôle et les autres sont des bits de données. Le bit de contrôle d'indice $i = 2^l$ est la somme modulo 2 de tous les bits de données c_j dont l'indice j écrit en base 2 a le $(l+1)$ème bit (à partir de la gauche) à 1. Par exemple, dans un code $(7, 4)$ de Hamming, si le message de départ est $[s_0, s_1, s_2, s_3]$ alors le mot du code correspondant est $[c_1, \dots, c_7]$ tel que :

$$c_3 = s_1 c_5 = s_2 c_6 = s_3 c_7 = s_4 c_1 = s_3 + s_5 + s_7 c_2 = s_3 + s_6 + s_7 c_4 = s_5 + s_6 + s_7$$

Pour corriger un code de Hamming, il faut faire la somme des indices de ces bits de contrôle. Cette somme donne l'indice de l'erreur lorsqu'elle est unique. En effet, un code de Hamming ne peut corriger qu'une seule erreur car sa distance minimale est égale à 3. Les codes de Hamming sont de longueur $2^r - 1$ et de dimension $k = n - r$.

3.3.3 Codes de Goppa

Introduits par Goppa en 1970 dans [58], les codes de Goppa étaient appréciés en théorie de codes pour leurs algorithmes de décodage efficaces, [Mas69] et [Pat75]. Ils sont maintenant très utilisé en cryptographie, en particulier dans le chiffrement de McEliece ([69]) et la signature CFS ([28]).

Construction des codes de Goppa

Soit g un polynôme de degré t de $\mathbb{F}_{q^m}$ et $L = \alpha_1, \dots, \alpha_{n-1}$ un ensemble d'éléments de $\mathbb{F}_{q^m}$ qui ne sont pas des racines de g et $n \leq 2^m$. On construit la matrice suivante :

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

On construit ensuite la matrice du sous-code sur le sous-corps $\mathbb{F}_q$ en remplaçant chaque ligne de la précédente matrice par m lignes correspondant aux coordonnées des éléments de la ligne dans $\mathbb{F}_q$. Cela donne une matrice H de taille $tm \times n$. Cette matrice est une matrice de contrôle du code de Goppa $\Gamma(g, L)$. Le code de Goppa $\Gamma(g, L)$ est donc un code $[n, k, d]$ avec $k \geq n - mt$ et de distance minimale supérieure à $t + 1$ car H est de rang plein. Les paramètres de ces codes pour le cas binaire sont $[2^m, k \geq 2^m - mt, d \geq 2t + 1]$.

3.4 problème de décodage par syndrome

Dans cette thèse, la théorie des codes sera utilisée pour construire ou attaquer des cryptosystèmes cryptographiques. Ces cryptosystèmes seront basé sur cette même théorie. Cela signifie que leur sécurité fait référence à un problème de la théorie des codes correcteurs réputé difficile. Le difficile problème auquel se réfère le plus souvent les cryptosystème est le problème de décodage par syndrome.

Définition 3.16. (Problème de décodage par syndrome) Soit H une matrice $(n-k) \times k$, w un entier et s un syndrome de taille $n-k$. Le problème consiste à trouver un mot e de taille n tel que $He^T = s$.

Ce problème correspond à savoir décoder un code linéaire quelconque en temps polynômial. Il a été prouvé NP-difficile dans [6]. Nous avons vu avec les exemples de codes donnés précédemment que ce problème n'est pas difficile dans tous les cas. Pour certaines valeurs de matrices H et des valeurs de w pas trop grandes, on peut décoder en temps polynômial. Il est souvent utile de chercher une instance du problème qui ne comporte qu'une seule solution. Pour cela, il faut connaître la distance minimale du code.

3.4.1 Différentes utilisation du problème de décodage par syndrome

Pour utiliser ce problème difficile sans le dénaturer, il faut choisir une grande famille de code et connaître la distance minimale des codes de cette grande famille. Le but de choisir une grande famille est d'avoir une bonne chance qu'elle contienne des instances difficiles du problème de décodage par syndrome. L'idéal étant de choisir la famille de tous les codes. La connaissance de la distance minimale permet ensuite de pouvoir paramétrer le problème afin de contrôler son nombre de solutions au mieux. Lors de la génération des clés d'un cryptosystème, une instance de la famille de code choisie sera construite aléatoirement avec des paramètres concrets. On veut qu'il y ait une bonne chance que cette instance soit difficile à résoudre en temps polynômial.

- La première famille de code que l'on peut utiliser est simplement la famille de tous les codes dont les paramètres n et k sont fixés. La distance minimale de ces codes est approximée par la borne de Gilbert-Varshamov.
- On peut aussi utiliser une grande famille de codes que l'on sait décoder, à condition de les masquer pour cacher les paramètres du code et ainsi cacher l'algorithme de décodage. Par exemple, la famille des codes de Goppa pour le système de chiffrement de McEliece.
- On peut aussi utiliser une famille de code qui a une structure particulière. Une famille qui a un groupe de permutation bien identifié par exemple.

Ces méthodes permettent de justifier que les cryptosystèmes se basent sur un problème difficile.

3.4.2 Difficulté pratique du problème de décodage par syndrome

On considère le problème de décodage par syndrome avec une matrice H de taille $(n-k) \times n$.
La première attaque à laquelle on peut penser est l'attaque par force brute. Elle consiste à chercher le syndrome par la matrice H de chaque mot de poids w. Sa complexité est $\binom{w}{n}$ multiplié par le coût de la multiplication par H.
Mise à part l'attaque par force brute, les attaque sur ce problème résulte d'améliorations sur une première attaque appelée, attaque par ensemble d'information. Elle a été introduite par McEliece dans [69] ou par Lee et Brickell dans [61]. Cette attaque consiste à trouver les w colonnes de H qui permettent de former le syndrome s. Pour cela, l'idée est de permuter les colonnes de H puis de faire une élimination gaussienne sur les lignes de la nouvelle matrice. On obtient ainsi une matrice sous cette forme :

$$(Id|Q)$$

Les opérations de l'élimination gaussienne sont faites en parallèle sur le syndrome s afin d'obtenir un problème équivalent. On appelle s' ce nouveau syndrome. Ensuite un p est choisit tel que $p \leq w$, et on recherche toutes les combinaisons de p colonnes de Q qui soit à distance $w-p$ de s'. Si on trouve une telle combinaison, on peut reformer un mot de poids w qui a s' pour syndrome. Dans le cas contraire on recommence le procédé avec une nouvelle permutation des colonnes.
En 1989, Leon dans [62] et Stern dans [93] ont observé que l'on pouvait rajouter une modification de type "Meet In The Middle" à cette attaque pour la rendre plus performante. L'idée est de rajouter une variable $l \leq n-k$ destinée à réduire les recherches sur les colonnes de Q à des plus petites colonnes de taille l. De cette façon, il est possible de stocker les combinaisons de p de ces colonnes dans des listes. On divise Q en deux parties. On crée deux listes, L_1 et L_2, dans le but de stocker les combinaisons de p colonnes de Q_1 et de Q_2. On peut ensuite chercher un mot y_1 de L_1 et un mot y_2 de L_2 tel que $y_1 + y_2$ soit égale aux l dernières coordonnées de s'. Si tel est le cas, il suffit de vérifier les $n-k-l$ autres coordonnées. Dans le cas où les listes ne donnent aucun résultat, on recommence le procédé depuis le début.
Il y a eu ensuite une amélioration de Bernstein et al dans [7] qui permet de d'identifier le cas où les mots y_1 et y_2 sont très proches. M.Finiasz et N.Sendrier ont montré dans [41] que cette transformation est équivavlente à mettre, dans la matrice H, les dernière lignes de la matrice identité à zéro. En effet, le fait de mettre 0 sur les l dernière colonnes implique que l'on chercher un syndrome égal sur les dernière valeur et non plus approché.
En 2012, A.Becker, A.Joux, A.May et A.Meurer ont montré dans [3] que l'on peut prendre des ensembles de recherches dans Q_1 et Q_2 un peu plus grands et non disjoints a condition que l'intersection de ces supports s'annulent grâce à la formule "$1+1=0$" en langage binaire. On peut retrouver le résumé de ces améliorations dans le tableau suivant :

	décodage à $d/2$	décodage à d
Lee-Brickell	0.05752	0.1208
Stern	0.05564	0.1167
Bernstein et al	0.05559	0.1164
MMT	0.05364	0.1116
BJMM	0.04934	0.1019

Ici, le code étudié est un code de Goppa avec d tel que $d/n = 0,04$ et $k/n = 0,7577$.

Chapitre 4

Rappel sur la cryptographie basée sur les codes correcteurs

On présente dans ce chapitre les cryptosystèmes les plus connu en cryptographie basée sur les codes correcteurs. C'est à dire, le chiffrement de McEliece, la signature CFS, le schéma d'authentification de Stern, et la signature à usage unique de KKS.

4.1 Chiffrement de McEliece

Le cryptosystème de chiffrement de McEliece est le premier cryptosystème basé sur les codes correcteurs. Il a été publié par McEliece en 1978 dans un rapport [69]. Il s'agit d'un cryptosystème asymétrique qui a la particularité d'être très rapide au niveau des calculs mais qui possède une clé publique très grosse.

4.1.1 Idée

L'idée est de masquer un code décodable pour le faire passer pour un code aléatoire. En effet, décoder un code aléatoire est considérer comme un problème difficile alors que l'on connait des familles de code que l'on sait décoder rapidement. Le masquage ne doit ni altérer la structure du code ni empêcher l'algorithme de décodage. C'est pourquoi il s'agit de multiplier le code à gauche par une matrice inversible et à droite par une matrice de permutation.

1. **Génération des clés :**
   On utilise un code t-correcteur G de taille $k \times n$ dans une grande famille de code. On construit aléatoirement une matrice S, inversible, de taille $k \times k$ et une matrice P de permutation de taille $n \times n$. Le triplet (G, S, P) constitue la clé privée alors que $G' = SGP$ correspond à la clé publique.

2. **Chiffrement :**
   Soit m un message sous forme de vecteur de taille k. Le chiffré de m correspond à la valeur $mG' + e$, avec e une erreur de poids t, choisie aléatoirement.

3. **Déchiffrement :**
   Pour retrouver m à partir de $mG' + e$ et de la clé privée (G, S, P) il suffit de multiplier par P^{-1} cette valeur afin d'obtenir $mSG + eP^{-1}$ puis d'utiliser l'algorithme de décodage de G afin de retrouver mS. Ensuite on peut retrouver m à partir de mS et de S car S est inversible.

FIGURE 4.1 – Chiffrement de McEliece

4.1.2 Schéma

4.2 Authentification de Stern

Le protocole de Stern est un protocole interactif à divulgation nulle de connaissance entre un prouveur et un vérifieur. On explique dans le premier chapitre plus en détail ce qu'est un protocole à divulgation nulle de connaissance ainsi que comment fonctionne ce protocole avant d'y présenter une amélioration sur la taille de ses communications
Jacques Stern proposa en 1993 dans [91], un schéma d'authentification à divulgation nulle de connaissance basé sur les codes correcteurs d'erreurs. C'est un protocole de questions/réponses à 3 pass avec une probabilité de triche de 2/3 pour le prouveur. Il utilise un fonction de hachage et sa sécurité est basée sur le problème de décodage par syndrome dans le cas le plus général.

4.2.1 Schéma

Le schéma utilise comme valeur publique une $(n-k) \times n$ matrice H quelconque choisie de manière aléatoire. Chaque prouveur possède une clé secrète et une clé publique, la clé secrète s est un mot de taille n et de poids w et la clé publique est son syndrome $p = Hs^T$ par la matrice H.
Le prouveur s'authentifie en 3 étapes. Dans un premier temps, il envoie des mises en gage au vérifieur. Ensuite le vérifieur lui envoie un challenge, et enfin, le prouveur répond au challenge du vérifieur.

1. **étape de mise en gage :**
   P choisit aléatoirement $y \in \mathbb{F}_2^n$ et une permutation σ de $\{1, 2, \ldots, n\}$.
   P envoie à V les hachés c_1, c_2 et c_3 tels que :

$$c_1 = h(\sigma | Hy^T);\ c_2 = h(\sigma((y)); c_3 = h(\sigma(y + s));$$

2. **étape du premier challenge :**
   V envoie $b \in \{0, 1, 2\}$ à P.
3. **étape de réponse :**
   Il y a trois possibilités :
   – Si $b = 0$: P révèle y et σ.
   – Si $b = 1$: P révèle $y + s$ et σ.
   – Si $b = 2$: P révèle $\sigma(y)$ et $\sigma(s)$.

FIGURE 4.2 – Protocole de Stern

A la suite de ce schéma il y a une phase de vérification qui permet au vérifieur de constater la bonne construction des mises en gage.

Etape de vérification :
Il y a trois possibilités :
– Si $b = 0$: V doit vérifier que c_1, c_2 ont été honnêtement calculés.
  V reçoit x et π, il compare $h(\pi | Hx^T)$ et c_1 ainsi que $h(\pi(x))$ et c_2.
– Si $b = 1$: V doit vérifier que c_2, c_3 ont été honnêtement calculés.
  V reçoit x et π, il compare $h(\pi | Hx^T - p)$ et c_2 ainsi que $h(\pi(x))$ et c_3.
– Si $b = 2$: V doit vérifier que c_1, c_3 ont été honnêtement calculés et que $wt(\sigma(s)) = w$.
  V reçoit u et v, il compare $h(u)$ et c_2 ainsi que $h(v)$ et c_3. Il vérifie que $wt(v) = w$.

FIGURE 4.3 – Vérification du protocole de Stern

Ce schéma fonctionne avec une probabilité de triche. Le prouveur peut tricher avec une probabilité de 2/3. Pour faire baisser cette probabilité autant que l'on veut il suffit de répéter ce protocole. La triche consiste à choisir des mises en gage particulières afin de pouvoir répondre correctement au challenge du vérifieur. Cette triche est détaillée dans le schéma suivant :

T un tricheur qui prend la place du prouveur P sans connaitre la clé secrète s.
- **Pour tricher dans le cas "$b = 0$ ou $b = 1$" :**
  T choisit un y dans $\mathbb{F}_2^n$, une permutation σ de $\{1, \ldots, n\}$ et un mot m de $\mathbb{F}_2^n$ tel que $Hm^T = H(y + s)^T$.
  T construit les mises en gages en prenant un mot quelconque à la place de s.
  Dans le cas $b = 0$, il envoie y et σ. Dans le cas $b = 1$, il envoie m et σ.
- **Pour tricher dans le cas "$b = 0$ ou $b = 2$" :**
  T choisit un y dans $\mathbb{F}_2^n$, une permutation σ de $\{1, \ldots, n\}$ et un mot z de $\mathbb{F}_2^n$ de poids w.
  T construit les mises en gages avec z à la place de s.
  Dans le cas $b = 0$, il envoie y et σ. Dans le cas $b = 2$, il envoie $\sigma(y)$ et $\sigma(z)$.
- **Pour tricher dans le cas "$b = 1$ ou $b = 2$" :**
  T choisit une permutation σ de $\{1, \ldots, n\}$, un mot z de $\mathbb{F}_2^n$ de poids $w$et un mot m de $\mathbb{F}_2^n$ tel que $Hm^T = H(y + s)^T$.
  T construit les mises en gages avec z à la place de s et $m + z$ à la place de y.
  Dans le cas $b = 1$, il envoie m et σ. Dans le cas $b = 2$, il envoie $\sigma(m + z)$ et $\sigma(z)$.

FIGURE 4.4 – Schéma de triche dans 2 cas sur 3

4.3 Signature à usage unique de KKS

La signature de KKS est un algorithme basé sur la théorie des codes correcteurs. Il a été proposé par G. Kabastianskii, E. Krouk et B.J.M. Smeets en 1997 dans [60]. Comme toutes les signatures, il associe à un message donné une signature. Celle-ci est construite à l'aide de sa clé secrète et peut être vérifiée grâce à sa clé publique.
L'idée de KKS est de précalculer des syndromes avec leurs mots de petits poids associés. Une signature correspond à une certaine combinaison de ces mots de petit poids et la vérification à la même combinaison mais appliquée aux syndromes. On a donc comme clé secrète M une matrice de mots de petits poids et comme donnée publique H une matrice de parité d'un code aléatoire et $F = HM^T$ une matrice de syndromes.

4.3.1 Schéma

Dans leur article, KKS ont demandé à la communauté une étude de la sécurité de leur schéma. Malheureusement ce schéma et ces variantes ont pu être cassées avec au plus 20 signatures, par Cayrel et al dans [14]. On peut toutefois utiliser ce schéma en tant que signature few-time et l'étendre à une signature multi-time à l'aide d'arbre de Merkle.
Avec ce schéma on peut signer 19 fois avec des clés de taille 300000b et des signatures de taille 600b. Dans le chapitre 6 on présentera une nouvelle signature à utilisation unique fonctionnant sur

1. **Génération des clés :**
   Soit H une matrice $(n-k) \times n$ aléatoire dans $\mathbb{F}_2$ représentant la matrice de parité d'un code aléatoire. M une matrice $n \times t$ dans $\mathbb{F}_2$ avec des colonnes de poids w. F une matrice $(n-k) \times t$ telle que $F = HM$. La clé publique est constituée de F et de H alors que la clé privée est la matrice M.
2. **Signature :**
   Pour signer un message m, il faut construire un mot x de $\mathbb{F}_2^t$ à l'aide du haché de m. La signature correspond à $\sigma = Mx^T$.
3. **Vérification :**
   Pour vérifier la signature $\sigma = Mx^T$ il suffit de calculer $H\sigma^T$ et de le comparer à Fx^T. Si les deux valeurs sont égale, la signature est vérifiée.

FIGURE 4.5 – Signature KKS

ce principe.

4.4 Signature de CFS

Il s'agit d'une signature basée sur les codes correcteurs et proposée par Courtois, Finiasz et Sendrier dans [28]. Elle se base sur le principe de la signature RSA, tout en résolvant les problèmes liés à cette même signature sur les codes. La signature RSA part du chiffrement RSA, qui est un chiffrement asymétrique. La signature consiste en un déchiffrement du message. Ce procédé est possible pour la signature RSA car l'ensemble des chiffrés est égal à l'ensemble des messages. La vérification correspond donc en le chiffrement de la signature. Cela donne un résultat attendu car $D(C(m)) = C(D(m)) = m$. En théorie des codes, il existe un seul chiffrement asymétrique, le chiffrement de McEliece. Toutefois, deux problèmes se posent pour l'obtention d'une signature similaire à la signature RSA :

1. L'ensemble des messages est différent de l'ensemble des chiffrés.
2. Le chiffrement et le déchiffrement ne commutent pas.

La signature CFS répond à ces problèmes en y apportant des solutions. Pour des paramètres de codes de Goppa de type $[2^m, 2^m - kt, 2t+1]$, la densité de décodage est de $1/t!$, cela donne une probabilité de pouvoir décoder un syndrome avec CFS de $1/t!$. Si t n'est pas trop gros, la complexité de l'algorithme devient raisonnable. Par contre le taux du code doit être proche de 1 pour avoir un t petit. On utilise comme clé publique une matrice duale plate d'un code de Goppa masqué.

4.4.1 Schéma

1. **Génération des clés :**
   Soit $\mathcal{C}$ un code de paramètre $[n, k]$, D un algorithme de décodage du code, H une matrice $(n-k) \times n$ de parité du code $\mathcal{C}$ et h une fonction de hachage à valeur dans $\mathbb{F}_2^{n-k}$.
   On utilise une matrice U de taille $(n-k)\times(n-k)$ et une matrice de permutation P de taille $n \times n$.
   La clé privée est (U, P, H) et la clé publique est $H' = UHP$.
2. **Signature :**
   Pour signer un message m :
   (a) $i \leftarrow 0$
   (b) Calculer $\sigma = D(h(m|i))$.
   – Si D retourne un résultat, la signature est (σ, i).
   – Sinon faire $i \leftarrow i + 1$ et recommencer l'étape (b).
3. **Vérification :**
   La signature (σ, i) de m doit vérifier $H'\sigma^T = h(m|i)$.

FIGURE 4.6 – Signature de CFS

Chapitre 5

Un nouveau protocole d'authentification Zero-Knowledge

Dans ce chapitre nous présentons un nouveau schéma d'authentification 5-pass basé sur le problème de décodage par syndrome et qui a une probabilité de triche asymptotique de $\frac{1}{2}$. Cette solution est basée sur la génération de nouveaux challenges à partir de la clé secrète en utilisant la structure cyclique présente dans la clé publique. En effet, une rotation du syndrome correspond a une rotation sur les parties de son antécédent. Nous montrons aussi comment réduire la taille de la signature en utilisant différemment les fonctions de hachage. Cette approche fonctionne aussi sur les anciens schéma basé sur le problème de décodage par syndrome. A la fin du chapitre nous proposons des comparaisons avec d'autres schémas. Ce travail à été réalisé en collaboration avec Carlos Aguilar Melchor et philippe Gaborit et a été présenté à ITW 2011 dans [1].

5.1 Problématique

Les schémas d'authentification à divulgation nulle de connaissance sont intéressant en cryptographie car leur sécurité est directement basée sur un problème difficile. De plus, ils peuvent être transformés en signature grâce au paradigme de Fiat-Shamir. En cryptographie basée sur les codes correcteurs, on peut observer un inconvénient majeur à ce type de construction. La taille des communication est très importante. En effet, celle ci est directement liée à la probabilité de triche lors d'un tour. En pratique, pour une sécurité de 2^{80}, la taille des communications est supérieur à $200kb$ pour construire une signature.

Dans ce chapitre nous exposons une nouvelle façon de construire les schémas d'authentification à divulgation nulle de connaissance dans le but de réduire les communications de ceux ci. La première idée est d'utiliser des codes doublement circulant pour augmenter le nombre de challenge et ainsi diminuer la probabilité de triche. La deuxième est d'utiliser les mises en gage différemment, pour

limiter le nombre d'envois de celles ci. En pratique, il est maintenant possible de signer pour une sécurité de 2^{80} avec une signature de ce type faisant $93kb$ au lieu de $150kb$ et d'être authentifier pour un coût de $20kb$ au lieu de $31kb$.

5.2 Rappel sur les codes

Dans cette section nous présentons des codes structurés par un ensemble de permutations qui rend leur description plus facile mais qui n'améliore pas, *a priori*, la complexité de leurs décodage. Puis sera présenté le schéma d'authentification de Véron sur lequel se base le schéma présenté dans ce chapitre.

5.2.1 Codes quasi-cyclique

Dans cette sous-section nous présentons les codes cycliques, quasi-cycliques et le problème de décodage par syndrome correspondant. L'authentification présentée dans ce chapitre utilise les codes quasi-cycliques.

Définition 5.1. (Code cyclique)
Un code $\mathcal{C}$ est dit cyclique si, pour tout mot $c = (c_0, c_1, \ldots, c_{n-1})$ de $\mathcal{C}$, le mot $c = (c_{n-1}, c_0, c_1, \ldots, c_{n-2})$ appartient aussi au code $\mathcal{C}$.

Cette définition nous permet d'introduire les codes quasi-cycliques :

Définition 5.2. (Code quasi-cyclique)
Un code $\mathcal{C}$ est dit quasi-cyclique si il peut être représenté par une matrice cyclique par bloc, avec des bloc de même taille.

Les matrices $k \times 2k$, cycliques par bloc, composées de seulement deux blocs, sont appelées doublement circulante. i
Grâce à cette définition nous pouvons introduire le problème de décodage par syndrome dans le cas quasi-cyclique.

Définition 5.3. (Problème SD-QC)
Soient $r > 1$, kn matrices cycliques $A_{1,1}, \ldots, A_{k,n}$ de taille $r \times r$, un entier $w < rn$ et un mot s de taille rk. Avec A la matrice quasi-cyclique suivante :

$$A = \begin{pmatrix} A_{1,1} & \ldots & \ldots & A_{1,n} \\ A_{2,1} & \ldots & \ldots & A_{2,n} \\ \vdots & \ddots & \ddots & \vdots \\ A_{k,1} & \ldots & \ldots & A_{k,n} \end{pmatrix}$$

Trouver e de taille rn et de poids $wt(e)$ tel que $Ae^T = s$.

5.2.2 Schéma de Véron

Clé privée : (e, m) avec e de poids w et de taille n et m un élément aléatoire de $\mathbb{F}_2^k$.
Clé publique : $(G, x, w))$ avec G une matrice aléatoire de taille $k \times n$ et $x = e + mG$.

1. [étape de mise en gage] P choisit aléatoirement $u \in \mathbb{F}_2^k$ et une permutation σ de $\{1, 2, \ldots, n\}$. Ensuite P envoie à V les hachés c_1, c_2 et c_3 tels que :

$$c_1 = h(\sigma);\ c_2 = h(\sigma((u + m)G));$$

$$c_3 = h(\sigma(uG + x));$$

2. [étape du premier challenge] V envoie $b \in \{0, 1, 2\}$ à P.
3. [étape de réponse] Trois possibilités :
   - Si $b = 0$: P révèle $(u + m)$ et σ.
   - Si $b = 1$: P révèle $\sigma((u + m)G)$ et $\sigma(e)$.
   - Si $b = 2$: P révèle u et σ.
4. [étape de vérification] Trois possibilités :
   - Si $b = 0$: V vérifie que c_1, c_2 ont été honnêtement calculés.
   - Si $b = 1$: V vérifie que c_2, c_3 ont été honnêtement calculés et que $wt(\sigma(e)) = w$.
   - Si $b = 2$: V vérifie que c_1, c_3 ont été honnêtement calculés.

FIGURE 5.1 – Protocole de Véron

5.3 Le nouveau schéma

5.3.1 Augmentation du nombre de challenges

A la différence du Schéma de Fiat-Shamir, pour lequel la probabilité de triche est de $\frac{1}{2}$, la probabilité du protocole de Stern est de $\frac{2}{3}$. Cette probabilité provient du fait que dans le cas de Stern, le secret à prouver possède deux propriétés, contrairement au secret de Fiat-Shamir qui n'a que la propriété d'être une racine quadratique. Les deux propriétés sont : un poids fixé égale à w et un syndrome fixé égale à la clé publique. Dans le cas du protocole de Stern, un tricheur peut toujours tricher dans 2 cas parmi 3, ce qui implique une probabilité de triche de $\frac{2}{3}$.

Le petit poids du secret est prouvé en utilisant une permutation et l'opérateur bit à bit XOR correspondant à l'addition dans $\mathbb{F}_2$. Ces opérations permettent de retrouver le syndrome grâce à la structure linéaire des opérations. Dans tous les schémas basés sur le problème de décodage par syndrome, on retrouve la même façon pour masquer le secret :

$$\sigma(e) + v$$

Ici, e est le secret de petit poids, σ est une permutation et v un masque. Dans le schéma de Véron v est égale à $\sigma((u+m)G)$, qui est un bon masque pour $\sigma(e)$ avec u un mot aléatoire. L'idée que l'on peut retrouver dans le schéma de Stern 5-pass [91] et [15] est qu'une variation de e peut prévenir une dépendance sur v et σ. Il n'y a donc pas besoin de tester la construction de v et σ dorénavant. La probabilité de triche est maintenant proche de $\frac{1}{2}$, en effet, il y a maintenant plus que deux challenges possibles pour la deuxième étape de challenge.

La variation sur e peut être effectuée de différente façon. Stern a proposé dans [91] une solution avec des codes de Reed-Muller, Shamir dans [88] à utilisé une multiplication par un scalaire. Dans notre cas, nous utilisons une rotation sur les deux parties de e. Nous pouvons en déduire les différents syndromes induits par cette transformation. Soit $H = [I|A]$, avec A une matrice cyclique de taille k et soit le syndrome $s = H.y^t$ avec $y = (y_1, y_2)$. Pour r une rotation de n positions, nous obtenons :

$$s = H \cdot (y_1, y_2)^t \Leftrightarrow r(s) = H \cdot (r(y_1), r(y_2))^t.$$

Cela implique $2k$ challenges possibles car il y a k choix de rotations possibles et 2 challenges possibles dans l'étape de challenge suivante. Un attaquant peut anticiper k cas parmi les $2k$ possibles mais nous prouvons qu'il ne peut pas anticiper plus de $k + i$ challenges (avec i un paramètre de sécurité) sans connaitre la clé secrète.

Cette approche, basée sur les rotations, est très efficace pour réduire la probabilité de triche dans un schéma binaire sans augmenter les communications comme ce fut le cas avec la solution passant par les Reed-Muller ou celle impliquant des corps de base plus gros.

5.3.2 Compression à l'aide des fonctions de hachage

Dans le schéma de Stern (ou celui de Véron), le prouveur envoie d'abord trois hachés : c_1, c_2 et c_3. L'envoie de ces haché à un certain coût. C'est ce coût que nous allons diminuer dans un premier temps. Pour cela, remarquons que si le protocole ce passe bien, le vérifier retrouver deux antécédents dans hachés parmi les trois envoyés au début. L'optimisation consiste ici à envoyer un haché de c_1, c_2 et $c3$ au lieu d'envoyer les trois comme précédemment. Comme le vérifieur est toujours capable de reconstruire deux dès trois hachés, il suffit de lui envoyer le troisième haché avant la vérification pour

qu'il puisse finir son calcul et retrouver le nouveau haché. De cette façon, seulement deux hachés sont envoyés au lieux de trois, ce qui fait un gain assez important du fait que cette procédure soit répétée plusieurs fois pour l'obtention d'une signature.

L'idée peut être généralisée sur plusieurs tours consécutifs. Dans ce cas, pour chaque tour, le prouveur envoie seulement le haché manquant lorsque les deux autres sont révélés par le vérifieur. On peut ensuite envoyer une mise en gage généralisée de ces plusieurs tours pour gagner en communication. Cette approche est plus efficace car elle voit chuter le nombre de haché envoyés par tours de 3 à 1.

5.3.3 Description du protocole

Nous reprenons ici les mêmes notations et les mêmes clés que dans le schéma de Véron.

Clé privée : (e, m) avec e de poids w et de taille n et m un élément aléatoire de $\mathbb{F}_2^k$.

Clé publique : $(G, x, w))$ avec G une matrice aléatoire de taille $k \times n$ et $x = e + mG$.

Nous décrivons seulement dans la figure suivante la première optimisation, la deuxième étant générique.

1. [étape de mise en gage] P choisit aléatoirement $u \in \mathbb{F}^k$ et une permutation σ de $\{1, 2, \ldots, n\}$. Ensuite P envoie à V les hachés c_1 et c_2 tels que :
$$c_1 = h(\sigma);\ c_2 = h(\sigma(uG));$$
2. [première étape de challenge] V envoie r, avec $0 \leq r \leq k-1$ à P.
3. [première étape de réponse] P construit $e_r = Rot_r(e)$ et envoie la dernière partie du haché :
$$c_3 = h(\sigma(uG + e_r))$$
4. [deuxième étape de challenge] V envoie $b \in \{0, 1\}$ à P.
5. [deuxième étape de réponse] Deux possibilités :
   - if $b = 0$: P révèle $(u + m_r)$ et σ.
   - if $b = 1$: P révèle $\sigma(uG)$ et $\sigma(e_r)$ où $e_r = Rot_r(e)$.
6. [étape de vérification] Deux possibilités :
   - if $b = 0$: V vérifie que c_1, c_3 ont été honnêtement calculées.
   - if $b = 1$: V vérifie que c_2, c_3 ont été honnêtement calculées et que le poids de $\sigma(e_r)$ est w.

FIGURE 5.2 – Nouveau protocole doublement circulant

La vérification du protocole consiste en la reconstruction du haché donné lors de la mise en gage. Dans le cas $b = 0$, la première et le troisième valeur peuvent être reconstruite. Dans le cas $b = 1$ il s'agit de la deuxième et de la troisième et pour le cas $b = 2$ il s'agit des deux premières. La

reconstruction des hachés est en général assez évidente, à part pour le cas $b = 0$ pour lequel, avec u et σ on reconstruit $c_3 = \sigma(uG + x_r)$, avec x la clé publique après r rotations.

5.4 Sécurité

Dans cette section nous prouvons la sécurité du schéma en utilisant les arguments habituels pour la divulgation nulle de connaissance, à savoir, la completness, la soundness et le zero-knowledge. Il est ensuite question de sécurité en pratique et de paramètres concrets pour ce schéma.

5.4.1 Completness

La completnesss est décrite par le schéma d'authentification. Celui-ci montre que l'authentification est réalisable et donne le résultat escompté.

5.4.2 Soundness

On montrera que quelqu'un qui peut être authentifié par ce protocole peut récupérer la clé secrète en temps polynômial avec une certaine probabilité. Pour ce faire, nous introduisons un problème spécifique qui est plus facile à résoudre que le problème de décodage par syndrome, sauf quand il n'a qu'une seule solution. Dans ce cas, c'est exactement ce problème. La façon dont nous assurons le sécurité est que nous choisissons des paramètres qui permettront de réduire la taille des solutions du nouveau problème à 1 avec une probabilité exponentiellement proche de 1 (en pratique, la probabilité d'avoir plus d'une solution est 2^{-80}). Prenons H une matrice doublement circulante aléatoire, X un mot au hasard de poids w dans $\mathbb{F}_{2^n}$.

Problème de décodage par syndrome différentiel :
Soit ${HX_1}^t$ un syndrome. Le problème $P(i, \omega)$ consiste à trouver i mots z_i et une constante C tels que $Hrot^j(X_1)^t - {Hz_j}^t = C$ et z_i de poids ω pour tous j inférieurs à i.

Dans notre cas (doublement circulant), les z_i correspondent aux rotations de z_1, avec $z_1 = Hx^T$, x étant la clé secrète et z_1 la clé publique. Ce problème est plus facile que i problèmes de décodage de syndrome indépendants à cause de l'addition de l'inconnu C. Il semble toutefois difficile à résoudre. Notez que nous pouvons supposer qu'il existe une solution particulière $Z_1, \ldots, Z_i, C$ du problème $P(i, \omega)$ tel que C soit égale à 0. Dans ce cas, nous avons $Hrot^j(X)^t = {Hz_j}^t$ pour tout j inférieur à i.

Lemme 5.4. Les images des mots de poids constant sont uniformément répartis parmi les syndromes.

Démonstration. Soit $X_{i,j}$ une variable aléatoire qui décrit le coefficient de la matrice H à la ligne i et à la colonne j. Cette variable a une distribution de Bernoulli de paramètre $\frac{1}{2}$. Pour chaque mot x de poids constant ω, nous construisons un vecteur aléatoire de taille $n-k$ tel que sa i-ème coordonnée soit la variable aléatoire $\sum_{j \in C_x} X_{i,j}$ où C_x décrit les coordonnées de x. Nous pouvons prouver de manière récursive que ces nouvelles variables aléatoires sont indépendantes et ont une distribution de Bernoulli de paramètre $\frac{1}{2}$. Ainsi, ces vecteurs aléatoires sont uniformément répartis entre les syndromes. □

Lemme 5.5. Soit $Z_C = (Z_1, \ldots, Z_i, C)$ un vecteur aléatoire avec Z_j (j entre 1 et i) une variable aléatoire de distribution uniforme sur les mots de poids ω et C une constante. Soit S_i la variable aléatoire égale à l'ensemble des solutions du problème $P(i,n)$. Nous avons $Pr[Z_C \in S_i] = \frac{1}{(2^{n-k})^i}$.

Démonstration.

$$Pr[Z_C \in S_i] = Pr[HZ_1{}^t = C + HX^t \text{ et } HZ_2{}^t = C + Hrot(X)^t \text{ et } \ldots \text{ et } HZ_i{}^t = C + Hrot^i(X)^t]$$

qui est la même probabilité que le produit de ces deux probabilités :

$$Pr[HZ_1{}^t = C + HX^t | HZ_2{}^t = C + Hrot(X)^t \text{ et } \ldots \text{ et } HZ_i{}^t = C + Hrot^i(X)^t]$$

$$Pr[HZ_2{}^t = C + Hrot(X)^t \text{ et } \ldots \text{ et } HZ_i{}^t = C + Hrot^i(X)^t]$$

Avec une forme particulière de k choisie en paramètre, $rot^j(C) \neq C$ pour tous j non nuls. On peut en déduire que :

$rot^j(C) - HX^t \neq C - HX^t$. Dans le cas où les mots de poids w n'ont pas d'image communes par H, on a $Pr[Z_C \in S_i]$ égale au produit de ces deux probabilités :

$$Pr[HZ_1{}^t = C + HX^t | Z_1 \neq Z_2 \text{ et } \ldots \text{ et } Z_1 \neq Z_3]$$

$$Pr[HZ_2{}^t = C + Hrot(X)^t \text{ et } \ldots \text{ et } HZ_i{}^t = C + Hrot^i(X)^t]$$

Ces variables sont indépendantes, donc :

$$Pr[Z_C \in S_i] = Pr[HZ_1{}^t = C + HX^t] Pr[HZ_2{}^t = C + Hrot(X)^t \text{ et } \ldots \text{ et } HZ_i{}^t = C + Hrot^i(X)^t]$$

Avec un raisonnement par récurrence, on a

$$Pr[Z_C \in S_i] = Pr[HZ_1{}^t = C]^i$$

Avec le lemme 5.4 on en déduit :

$$Pr[Z_C \in S_i] = \frac{1}{(2^{n-k})^i}$$

□

Lemme 5.6. La distribution de N_i qui décrit la taille de S_i est la même que la variable $1 + Y$ avec Y une distribution binomiale de paramètre $N = (2^{n-k} - 1)\binom{n}{\omega}^i$ et $p = (\frac{1}{2^{n-k}})^i$. De plus, $\mathbb{E}[S_i] = Np + 1 = (\frac{\binom{n}{\omega}}{2^{n-k}})^i (2^{n-k} - 1) + 1$.

Démonstration. Soit $Z_C = (Z_1, \ldots, Z_i, c)$ le vecteur aléatoire définit dans 5.5 avec $C \neq 0$ et T_C la variable égale à 1 quand $Z_C \in S_i$ et 0 sinon. $N_i = \sum_{C \neq 0} T_C + Nb_0$ avec Nb_0 le nombre de solutions quand $C = 0$. La variable Nb_0 est égale à 1 car s_1, ..., s_i ont été choisi comme syndromes de mots ayant un poids w et nous supposons qu'ils sont uniques. Ainsi, nous avons $N_i = 1 + Y$ avec Y une loi binomiale de paramètres $N = (2^{nk} - 1)\binom{n}{\omega}^i$ et $p = \frac{1}{2^{nk}}^i$. □

Lemme 5.7. Avec Y' une variable de distribution de Poisson de paramètre Np, nous avons : $Pr[N_i = 1] \approx Pr[Y' = 0]$ et $Pr[Y' = 0] \approx 1 - \frac{\binom{n}{\omega}^i}{2^{(n-k)(i-1)}}$.

Démonstration. La distribution binomiale de Y' peut être évaluée avec une distribution de poisson de paramètre $\lambda = Np$. On en déduit que la probabilité $Pr[N_i = 1] \approx Pr[Y' = 0]$ et $Pr[N_i = 1] \approx e^{-\frac{\binom{n}{\omega}^i}{2^{(n-k)(i-1)}}}$. Quand x est proche de 0 on a $e^x \approx 1 - x$ donc $Pr[N_i = 1] \approx 1 - \frac{\binom{n}{\omega}^i}{2^{(n-k)(i-1)}}$. □

On appelle ϵ la valeur $1 - \frac{\binom{n}{\omega}^i}{2^{(n-k)(i-1)}}$.

Lemme 5.8. Si quelqu'un peut résoudre le problème $P(i, \omega)$ avec probabilité ϵ' , il peut retrouver la clé secrète du protocole à partir de la clé publique avec une probabilité à peu près égale à $\epsilon\epsilon'$.

Démonstration. Nous avons vu dans le lemme 5.7 que la probabilité que la solution de $P(i, \omega)$ soit unique est ϵ. □

Théorème 5.9. Si un prouveur B est capable d'être authentifié par un vérifieur A avec une probabilité supérieure à $\frac{k+i}{2k}$, alors B peut récupérer la clé secrète du protocole à partir de la clé publique avec une probabilité supérieure à $1 - \frac{\binom{n}{\omega}^i}{2^{(nk)(i-1)}}$ en temps polynômial ou trouver une collision sur la fonction de hachage en temps polynômial.

Démonstration. Le prouveur B est en mesure de répondre au défi de plus de $k + i$ requêtes. Dans ce cas, nous appelons t et b, respectivement, le premier et le second challenge du vérifieur. Nous nommons c_1, c_2 le premier envoi du prouveur, $c3$ le deuxième et (u_t, σ_t) ou (X_t, z_t) le dernier envoi, respectivement, lorsque $b = 0$ et lorsque $b = 1$.
D'après le principe des tiroirs de Dirichlet, on déduit que B est en mesure de répondre au défi $(t, b = 0)$ et $(t, b = 1)$ pour au moins i défis différents t , nous les appelons $t_1, \ldots, t_i$. Le vérifieur A doit vérifier que $hash(\sigma_{t_i}) = c_1$ et $hash(X_{t_i}) = c_2$, donc soit B trouve une collision sur la fonction de hachage ou $\sigma_{t_i} = \sigma$ et $X_{t_i} = X$ pour tous t_i. Le vérifieur A doit également vérifier que le poids de z_{t_i} est égal à ω et $hash(\sigma(u_{t_i}G + x_{t_i})) = hash(X + z_{t_i})$, donc soit B trouve une collision pour la fonction de hachage ou $\sigma(u_{t_i}G + x_{t_i}) = X + z_{t_i}$. On en déduit que $u_{t_i}G + x_{t_i} = \sigma^{-1}(X) + \sigma^{-1}(z_{t_i})$ puis ${Hx_{t_i}}^t - H(\sigma^{-1}(z_{t_i}))^t = H(\sigma^{-1}(X))^t$. Comme ${Hx_{t_i}}^t = {He_t}^t$ cette équation correspond au problème $P(i, \omega)$. On en déduit d'après le lemme 5.8 que B est en mesure de trouver la clé privée avec une probabilité supérieure à ϵ .

Théorème 5.10. Si un prouveur B est capable d'être authentifié par un vérifieur avec une probabilité supérieure à $(\frac{k+i}{2k})^n$, B peut récupérer la clé secrète du protocole à partir de la clé publique une avec une probabilité supérieure à $1 - \frac{\binom{n}{\omega}^i}{2^{(nk)(i-1)}}$ en temps polynômial ou trouver une collision sur la fonction de hachage en temps polynômial.

Démonstration. Le prouveur B peut construire $c_{1,1}, \ldots, c_{1,N}$ et $c_{2,1}, \ldots, c_{2,N}$ tels qu'il peut être authentifié avec un probabilité supérieure à $(\frac{k+i}{2k})^N$. D'après le principe des tiroirs de Dirichlet nous pouvons déduire l'existence d'un entier j tel que B peut être authentifié par le premier protocole avec une probabilité supérieure à $\frac{k+i}{2k}$. Le théorème précédent permet de conclure la démonstration. □

5.4.3 Zero-Knowledge

Cette partie de la preuve consiste à prouver qu'aucune information ne peut être déduite en temps polynômial à partir d'une exécution du protocole en plus des information sur les données publiques. L'idée est de prouver que l'on peut construire un simulateur du protocole en temps polynômial qui sera indistinguable du protocole d'origine.
Le protocole est construit par anticipation des challenges, à chaque tour il est possible de créer une instance valide en anticipant le challenge b. Cela implique que la construction prendra deux fois plus de temps que le protocole.
Le cas $b = 0$ peut être anticipé en choisissant une permutation aléatoire σ', un mot aléatoire v, $h_1 = hash(\sigma')$ et $h_3 = hash(\sigma'(vG + x_r))$. Notons que (v, σ') et $(u + m_r, \sigma)$ sont indistinguables. Le cas $b = 1$ peut être anticipé en choisissant v et z tel que z soit un mot de poids w, $v = \pi(uG)$ avec π une permutation aléatoire, u un mot aléatoire, $h_2 = hash(v)$ et $h_3 = hash(v + z)$. Notons que (v, z) et $(\sigma(uG), \sigma(e_r))$ sont indistinguables.
Le coût de construction de ce simulateur est négligeable et n'affecte pas les paramètres de sécurité du système. Quand nous utilisons la compression à l'aide des fonctions de hachage, la construction du simulateur diffère car le coût d'anticipation devient plus important. Le protocole doit ainsi hacher les challenges par blocs afin de pouvoir être anticipés en temps raisonnable par un simulateur.

5.4.4 Sécurité en pratique pour les codes doublement circulant

A la différence du schéma de Stern , notre protocole est basé sur un décodage de matrice aléatoire doublement circulante (problème SD-DC). Ce problème, à la différence du problème de décodage par syndrome, n'est pas prouvé NP-difficile (même si un résultat est connu sur la difficulté de décoder les codes quasi-cycliques en général). Dans notre cas le problème semble être difficile pour plusieurs raisons. Premièrement, il a été prouvé dans [52] que les codes aléatoires doublement circulantes satisfont sur la borne de Gilbert-Varshamov. De plus, on ne sait pas, même avec des codes très structurés, comment décoder un code sur la borne de Gilbert-Varshamov en temps polynômial. Dans

la pratique, il n'est pas connu d'algorithme spécialisé qui peut faire significativement mieux (plus d'un facteur linéaire n) pour résoudre le problème SD-DC. La situation est la même que pour les réseaux, entre réseaux idéaux et réseaux aléatoires. En pratique, les meilleurs algorithmes connus pour attaquer le problème SD-DC sont les mêmes que ceux pour le problème de décodage par syndrome.

5.5 Paramètres pour l'authentification et la signature

Selon les contraintes de sécurité pour les preuves de zero-knowledge, nous choisissons en tant que paramètres $n = 698$; $k = 349$; $i = 19$;$w = 70$ pour une sécurité de 2^{81} et une probabilité de triche de 2^{16}.

TABLE 5.1 – "Comparaison entre les schéma ZK pour une probabilité de triche de 2^{-16}"

	Stern 3	Stern 5
Rounds	28	16
Matrix size (bits)	122500	122500
Public Id (bits)	350	2450
Secret key (bits)	700	4900
Communication (bits)	42019	62272
Complexité du prouveur	$2^{22.7}$op. dans $\mathbb{F}_2$	$2^{21.92}$op. dans $\mathbb{F}_2$

Veron	CVE	Nouveau protocole
28	16	18
122500	32768	350
700	512	700
1050	1024	700
35486	31888	20080
$2^{22.7}$op. dans $\mathbb{F}_2$	2^{16}mult. dans $\mathbb{F}_{256}$	2^{21}op. dans $\mathbb{F}_2$

Pour une sécurité de 2^{100} nous choisissons $n = 838$; $k = 419$; $i = 20$;$w = 86$ et pour une sécurité de 2^{128} nous avons $n = 1094$; $k = 547$; $i = 14$;$w = 109$.

Pour la signature, et une probabilité de triche de 2^{80}, il est suffisant de multiplier par 5 les données précédentes. Cela permet d'obtenir une signature de longueur 93kb.

Remarque 5.11. il est possible de réduire encore plus le coût de la communication en utilisant un encodage de poids constant lors de l'envoi de e_r, le coût est alors de k bits plutôt que $2k$, dans l'ensemble, il fait diminuer à 17kb l'authentification et à 79kb la signature, mais l'encodage à un coût significatif en temps de calcul.

Chapitre 6

Signature à utilisation unique sur les codes de résidus quadratiques

Nous présentons dans ce chapitre une nouvelle signature à utilisation unique basée sur les codes correcteurs. Cette signature est une amélioration de la signature KKS introduite par Kabastianskii, Krouk et Smeets en 1997. La signature KKS est une signature à usage unique rapide mais qui a de trop grosses clés publiques. La signature présentée dans ce chapitre a une taille de clé de $17kb$, pour une taille de $7kb$. Elle utilise des codes de résidus quadratiques à la place des codes utilisés dans KKS. Cette signature à fait l'objet de l'article [50], résultant d'un travail avec P.Gaborit.

6.1 Problématique

Le problème de trouver une signature efficace est un problème récurrent en cryptographie. Dans la cas de la cryptographie basée sur les codes, aucune solution idéale n'a encore été trouvée. La signature présentée dans ce chapitre est un bon compromis des différentes techniques existantes en terme de signature basée sur les codes. Il y a trois façons différentes de signer avec les codes, avec le protocole zero-knowledge de Stern, avec la signature CFS ou avec la signature à usage unique KKS. Cette signature appartient à la famille des signatures basées sur KKS.

6.2 Idée du schéma de signature avec les matrices doublement circulantes

Le protocole reprend l'idée de KKS qui consiste à considérer une matrice de parité H et une matrice de syndrome F. La principale différence avec KKS se situe au niveau de la matrice F. L'idée

est de trouver une quantité importante de syndrome (plus de 2^{150}), décrit de façon minimaliste. On utilise ici 4 ou 13 syndromes au lieu de 160 dans le cas KKS.

Soit A une matrice $k \times k$ aléatoire circulante telle que chaque ligne soit une permutation de la ligne antérieure, la première ligne étant quelconque. Soit H la matrice $(I|A)$ avec I la matrice identité $k \times k$. Définissons par σ la rotation d'une position d'un mot de taille k et π_σ sa matrice $k \times k$ associée. Pour $x = (x_1, x_2)$ avec x_1 et x_2 dans $\mathbb{F}_2^k$, nous pouvons faire l'observation suivante :

$$H.(x_1.\pi_\sigma, x_2.\pi_\sigma)^t = \pi_\sigma^t(x_1)^t + A.\pi_\sigma^t.x_2^t$$

En utilisant le fait que A est cyclique et donc commute avec π_σ et que $\pi_\sigma^t = \pi_\sigma^{-1}$, nous avons :

$$\pi_\sigma^t(x_1)^t + A.\pi_\sigma^t.x_2^t = \pi_\sigma^{-1}(x_1)^t + \pi_\sigma^{-1}.A.x_2^t = \pi_\sigma^{-1}.(H.x^t).$$

On peut donc constater que l'on peut construire beaucoup de nouveaux syndromes décodables à l'aide de $H.x^t$ et de la rotation σ. L'action de π_σ^{-1} étant transmise à π_σ sur les deux parties x_1 et x_2 de x. Pour conclure, si l'on rend publique l différents syndromes $s_i = H.x_i^t$ que l'on sait décoder, on peut reconstruire k^l syndromes de l'on saura décoder en utilisant la rotation σ.
L'algorithme fonctionne en créant des challenges comme cela. Des syndromes publiques sont donnés, puis le signeur créé à partir du message un challenge lui demandant de prouver la connaissance de l'un de ces nombreux syndromes possibles. Le nombre de syndrome décodable potentiel doit être grand pour que deux messages n'aient pas le même syndrome demandé lors du challenge. On montrera dans la section 6.5 une autre famille de matrice permettant d'avoir un nombre de syndromes déduits en k^3 au lieu de k pour les rotations.

6.3 Approche formelle de la signature

Dans cette section, j'explique de façon formelle les mécanismes pour obtenir de nombreux challenges lors de la création de la signature.

Dans toute la suite nous notons G un groupe de permutation agissant sur k positions, g un élément de G et H une matrice $k \times n$ avec nrk. Nous notons π_g la matrice $k \times k$ représentant la permutation g et g_r la concaténation de l'action de g sur r blocs de k positions. La permutation g_r est donc la permutation sur kr positions tel que les kr positions sont considérées comme des suites de r blocs de k colonnes sur lesquelles g_r est agit comme g. La matrice $kr \times kr$ représentant g_r est la suivante :

$$\Pi_g = \begin{pmatrix} \pi_g & 0 & \cdots & 0 \\ 0 & \pi_g & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \pi_g \end{pmatrix},$$

Définition 6.1. (Compatibilité des syndromes) Soit G un groupe de permutations de k positions et $H = (I|H_1|H_2|\cdots|H_{r-1})$ une matrice de parité $k \times rk$. On dit que G est *compatible* avec les syndromes de H si pour tout g dans G il existe une matrice $k \times k$ noté L_g telle que pour tout $1 \leq i \leq r-1$ on a $H_i.\pi_g = L_g.H_i$. La matrice L_g est appelée la matrice compatible de g pour H_i.

Proposition 6.2. Si un groupe de permutations G est compatible avec les syndromes de H alors pour tout x dans F_2^n et tout $g \in G$:

$$H.(x.\Pi_g)^t = L_g.(H.x^t).$$

Démonstration. La proposition est claire à partir de la pseudo-commutativité induite par la définition de compatibilité. □

Remarque 6.3. La proposition précédente donne des conditions pour permettre d'associer des transformations sur $H.x^t$ à des transformations sur x.

exemple :

Un exemple de groupe compatible par syndrome est le groupe des doubles rotations. Ce groupe est compatible avec les matrices $k \times 2k$, $H = (I|H_1)$ avec H_1 une matrice cyclique. En effet, comme les doubles rotations commutent avec les matrices cycliques, on a $L_g = \pi_g^{-1}$. On verra dons la suite une application où le groupe compatible est $PSL_2(q)$ avec $L_g \neq \pi_g^{-1}$.

6.4 Schéma de signature à usage unique

Dans cette section on considère une fonction de hachage h, G un groupe à syndrome compatible avec H une matrice de parité $k \times rk$, et deux entiers w et t. On considère la bijection $\phi : [1 \ldots |G|] \to G$ qui associe à n'importe quel nombre entre 1 et l'ordre de G, un élément de G. On considère aussi l'application $L : G \to GL(k)$ qui associe à n'importe quel $g \in G$, sa matrice compatible avec H, L_g. On écrit $(m\|i)$ la concaténation de m et de i. On note t le paramètre de sécurité.

Algorithme de génération de clés

• **Public data** Un groupe de permutation G compatible avec H.

• **Clés générées**
Clé privée : $x_1, x_2, ..., x_l$ des mots aléatoires de poids de Hamming proche de t.
Clé publique : les syndromes associés $s_i = H.x_i^t$.

Algorithme de signature

Entrée : m un message, s un entier.

• **Signature**
1. Choisir aléatoirement j entre 1 et 2^s.
2. Générer l éléments $a_1, a_2, ..., a_l$ avec $1 \leq a_i \leq |G|$, à partir de $h(m\|j)$.
3. Calculer $sign = \sum_{i=1}^{l} x_i.\Pi_{\phi(a_i)}$.
4. Si $weight(sign) > w$ ou si le nombre de coordonnées communes entre $x_i.\Pi_{\phi(a_i)}$ et $sign$ est plus grand que t, retourner à l'étape 1.
5. Sortir la signature $(sign, j)$.

• **Vérification**
1. Générer les a_i à partir de m et de j
2. Vérifier que : $H.sign^t = \sum_{i=1}^{l} L_{\phi(a_i)} s_i$ et que $weight(sign) \leq w$.

Démonstration. La vérification fonctionne tant que pour tout i : $H.(x_i.\Pi_{\phi(a_i)})^t = L_{\phi(a_i)}(H.x_i^t) = L_{\phi(a_i)}.s_i$. □

6.5 Exemples de groupes de syndromes compatibles

On donne ici deux exemples de groupes compatibles et les signatures correspondantes.

6.5.1 Construction quasi-cyclique

Nous considérons une $[k, rk]$ matrice $H = (I|H_1|\dots|H_{r-1})$ pour H_i des matrices circulantes aléatoire de taille k. Comme nous l'avons vu dans les exemples précédents nous pouvons prendre pour G le groupe des décalages cycliques de taille k. Si l'on désigne par π_a le décalage de a positions pour $1 \leq a \leq k$ on obtient :

$$H.(x.\Pi_a)^t = \pi_a^{-1}.(H.x^t).$$

Dans ce cas, en partant de k de l'ordre de quelques milliers, on peut considérer comme clé publique un ensemble de 13 syndromes de taille k pour obtenir au moins 2^{144} signatures possibles.

6.5.2 Codes de résidus quadratique doublement circulant

Cette classe de codes regroupe des codes $[2(p+1), p+1]$ (p un nombre premier). Elle a été introduite à la fin des années 60 par Karlin et a depuis fait l'objet de nombreux articles. Cette classe de codes a la propriété de contenir en général de très bons codes, en terme de distance minimale, de plus elle a la propriété d'avoir un assez grand groupe d'automorphisme : le groupe $PSL(2,q)$ d'ordre $\frac{(p-1)p(p+1)}{2}$ agissant en même temps sur les deux parties de colonne.
Pour plus de précision sur les propriétés de ce groupe, se référer à [67](p.492).

Introduisons quelques définitions.
Soit p un nombre premier. Nous définissons la matrice de carrés Q_p de la façon suivante : la première rangée de Q_p a p positions numérotées de 0 à $p-1$. Si la position i pour $1 \leq i \leq p-1$ est un carré modulo p on met un '1', et si le numéro de position n'est pas un carré on met un '0' à cette position. Ensuite, les autres $p-1$ lignes sont obtenus sous forme de décalages cycliques successives de la première rangée.

Nous considérons aussi la matrice des non carrés N_p qui a '1' aux positions non carrés et '0'dans des positions carrés (avec toujours '0' en position zéro).

Nous définissons maintenant B_p la $(p+1) \times 2(p+1)$ matrice circulante *bordée* associée telle que :

$$B_p = (U_p|V_p) = \left(\begin{array}{c|c|c|c} 0 & 0\cdots 0 & 1 & 1\cdots 1 \\ \hline 1 & & 0 & \\ \vdots & I & \vdots & M_p \\ 1 & & 0 & \end{array}\right)$$

avec $M_p = I_p + N_p$ si $p = 4l+3$ et $M_p = Q_p$ si $p = 4l+1$ avec l un entier.

On note les colonnes de U_p et V_p : $(e_\infty, e_0, e_1, \cdots, e_{p-1})$. On présente ici les trois permutations sur e_i :

1. Décalage $S(b)$: Pour $0 \leq b \leq p-1$, $e_\infty.S(b) = e_\infty$ et $e_i.S(b) = e_{i+b}$ (où la somme $i+b$ est considérée modulo p.
2. La transformation carrée $T(s^2)$, $0 \leq s \leq p-1$ définie par : $e_\infty.T(s^2) = e_\infty$ et $e_i.S(b) = e_{s^2.i}$ avec le produit $s^2.i$ considéré modulo p.

3. La transformation inverse négative N : $e_\infty.N = e_0$, $e_0.N = e_\infty.N$ et $e_i.N = e_{-1/i}$, où l'inverse est pris modulo p.

Il est connu que ces transformations engendrent le groupe $PSL_2(p)$ d'ordre $\frac{(p-1)p(p+1)}{2}$ (voir par exemple [67](p.491)).

Proposition 6.4[67]. Pour tout nombre premier impair p :

1. Les permutations $S(b)$ et $T(s^2)$ commutent avec U_p et V_p.
2. De plus :
$$(U_p|V_q)N = L(U_p|V_p),$$
avec L la $(p+1)\times(p+1)$ matrice carrée dépendant de p, qui a pour lignes $(L_\infty, L_0, \cdots, L_{q-1})$ définies par :
• si $p = 4l-1$: $L_\infty = e_\infty, L_0 = e_0 + e_\infty$ et si i est un résidu quadratique : $L_i = e_\infty + e_0 + e_{-1/i}$ et si j n'est pas un résidu quadratique : $L_j = e_0 + e_{-1/j}$,
• si $p = 4l+1$: $L_\infty = e_\infty, L_0 = e_0$ et si i est un résidu quadratique : $L_i = e_0 + e_{-1/i}$ et si j n'est pas un résidu quadratique : $L_j = e_\infty + e_0 + e_{-1/j}$.

Dans la proposition suivante on considère que N agit simultanément sur les deux parties de B_p.

Proposition 6.5. Le groupe $PSL_2(p)$ est syndrome compatible avec la matrice B_p.

Démonstration. Nous avons vu que le groupe $PSL_2(p)$ a été généré par les trois transformations $S(b), T(s^2)$ et N, les deux premières commutent avec U_p et V_P, l'action de N se traduit par une multiplication par une matrice linéaire L, définie dans la proposition précédente. Cela prouve le résultat. Par ailleurs, une description du groupe $PSL_2(p)$ se déduit aisément les trois transformations. □

Si l'on décrit les éléments de $PSL_2(p)$ avec $y \to \frac{ay+b}{cy+d}$ et $ad-bc = 1$ il est facile de décrire $PSL_2(p)$ à partir de $S(b), T(s^2)$ et N (voir [67](p.492) pour plus de détails).

6.6 Sécurité de la signature à usage unique

6.6.1 Suppositions sur la difficulté du problème de décodage par syndrome dans le cas quasi-cyclique et dans le cas des codes quadratiques

Supposition 1 Le décodage d'un code quasi-cyclique aléatoire est un problème difficile.

Commentaire sur cette supposition

La classe des codes quasi-cycliques a été largement utilisée au cours des dernières années. Même si cette classe de codes peut sembler faible car ils sont plus structurés, dans la pratique, aucun algorithme pour résoudre ce problème ne fait nettement mieux que l'algorithme général. Récemment, ce problème a été étudié dans [19] et aucun gain réel a été observé, par ailleurs le gain supposé de l'ordre de la matrice (n) est loin d'être atteint.

Supposition 2 Dans les code quadratiques doublement circulant avec comme paramètres $[2p + 2, p + 1]$ pour p un nombre premier, il est difficile de décoder plus de $3log_2(p)$ erreurs.

Commentaire sur la supposition

Si on met de côté les deux colonnes d'extension, ces codes peuvent être considérées comme un cas particulier des codes quasi-cycliques . En fait, ils sont liés aux codes à résidus quadratiques sur $GF(4)$ (voir [67](p.490)), comme leurs images binaires. Les codes à résidus quadratiques ont été étudiés depuis plus de 50 ans et aucun algorithme n'a été trouvé pour de décoder les codes cycliques de manière plus efficace. Si l'on considère une distribution aléatoire de zéros du code cyclique de plus de $GF(4)$ on obtient facilement que la taille maximale moyenne d'une séquence de zéros du code cyclique est de l'ordre $2log_2(p)$. A peu près n'importe quelle position a une probabilité $1/2$ d'être un zéro (puisque le taux de code est $1/2$). Concrètement, nous ne savons pas comment décoder plus de $3log_2(p)$ erreurs pour ces codes. Au-delà de ce nombre d'erreurs (et probablement avant cette limite), nous ne savons pas comment décoder ces codes d'une autre manière que pour les codes aléatoires.

Notez que la recherche d'un nouvel algorithme pour décoder au-delà de cette limite serait clairement une percée dans la théorie des codes puisque ce problème a été connu (et non résolu) depuis plus de 50 ans.

6.6.2 Sécurité du schéma

Il y a deux principaux cas à prendre en considération, le premier cas est quand un attaquant ne connaît aucunes signatures et n'a que la clé publique du système, le second cas, plus difficiles à prendre en considération, est le cas d'un attaquant qui connaît une signature.

Réduction de sécurité dans le cas de l'inforgeabilité lorsque aucune signature n'est connue

Nous avons le théorème suivant.

Théorème 6.6. Considérons une famille de codes QC ou QDC à partir de laquelle un seul code C est utilisé dans le protocole. Supposons que nous choisissons comme paramètres dans le protocole w de poids sur la borne de Gilbert-Varshamov lié au code C et t une valeur donnée associée à w comme dans le protocole. Ensuite, si un attaquant est capable de forger une signature unique de la connaissance de la clé publique (les l syndromes) alors il est capable de décoder le code C à t erreurs.

Démonstration. Supposons que l'on voit attribuer un code C comme dans le théorème et un syndrome donné associé à un mot de code x_1 de poids t, alors il est possible de prendre $l-1$ autres mots codés aléatoires $x_2, \ldots, x_l$ avec un poids t. On obtient alors une famille de l syndromes comme dans le protocole. Si un attaquant est capable de forger une signature il est capable de construire une somme de permutés x_i, puisque w est en dessous de la borne de GV, nous savons qu'il existe une solution unique à un syndrome donné. Tous les x_i sont connus sauf x_1, donc il est possible de récupérer x_1 de la signature qui est leur somme permuté.

Or, puisque depuis, nous avons supposé que le décodage de ces codes, avec un poids supérieur à $4log_2(p)$, est dur, il est difficile de forger une signature lorsque seule la clé publique est connue.

□

Remarque 6.7. Ce que nous obtenons avec notre schéma est nettement meilleure que l'approche du système KKS (ou ses variantes) qui n'est pas liée à un problème de décodage classique. Notez que cette approche n'est pas possible dans le cas du système KKS car pour simuler les données exactes du protocole, il faudrait les supports exactes des différents secrets, ce qui n'est pas le cas pour notre protocole où tous les x_i sont indépendants.

Inforgeabilité avec une signature connue

Dans ce cas, il n'y a pas de bonne réduction comme dans le cas précédent. Cela n'est pas surprenant puisqu'un schéma de signature à usage unique ne peut pas être répétée sans donner trop d'informations.

Pour ce cas, l'idée est la suivante. L'attaquant connaît les permutations utilisés. Si un attaquant connaît une signature, alors il connaît une grande partie de positions de '1' de chaque x_i (en particulier x_1). D'un autre côté, il y a toujours une partie des positions des '1' de x_1 qui disparait toujours.

Calculons précisément la taille de cette partie : considérons $l-1$ variables de Bernoulli, chacune prenant la valeur '1' avec probabilité $\frac{t}{n}$. La probabilité que la somme modulo 2 de ces variables soit égale à 0 est égale à $q = \frac{1+(1-2\frac{t}{n})^{l-1}}{2}$. Ainsi, cela signifie que si tous les x_i ont un poids t, alors que le nombre de '1' de x_1 dans la somme est permuté est qt. Le nombre de disparition de positions à '1'

de x_1 dans la signature est donc $(1-q)t$.

L'attaquant veut retrouver x_1 sachant que qt positions à '1' de x_1 appartiennent au support de la signature de poids w. On sait aussi que $(1-q)t$ position à '1' sont inconnues. En considérant la matrice formée à partir des w colonnes de H associée aux positions non nulles de la signature, ces colonnes forment un code $C_1[n-k, w]$. Si on multiplie H par la matrice dual de C_1, on fait disparaitre les w colonnes de '1' et le problème revient à retrouver $(1-q)$ positions manquantes de x_1 dans un code $[n-w, k]$. Remarquons qu'un attaquant perdra toujours l'information sur les bits à '1' manquant dans les x_i. Cela permet d'avoir une borne inférieur sur la sécurité de la signature dans le cas où une signature est connue de l'attaquant.

6.7 Complexité et paramètres

La complexité du système est quadratique en k, la clé publique et la signature sont linéaires en la taille de k. Dans ce qui suit nous donnons les paramètres pour une sécurité de 2^{80} en fonction de la sécurité définie dans la section précédente. Pour évaluer le coût de l'attaque, nous avons utilisé le résultat récent de Finiasz et Sendrier ([41]) et l'analyse de la précédente section.

• **Codes quadratiques doublements circulants** : $G = PSL_2(p)$. On prend $p = 3457, l = 5$, le poids t de $x_i = 340$, poids w de la signature, 1650. nombre de bits communs : $t < 340 - 55 = 285$. Clé publique : 17kb,taille de signature : 7000b.

6.7.1 Signature à usage unique et arbres de Merkle

L'intérêt de signatures à usage unique est en soi pas très important, car très peu d'applications ne nécessite qu'une seule signature. L'intérêt principal de ces signatures ponctuels est la possibilité de les utiliser en relation avec un arbre de Merkle (également connu sous le nom d'arbres de hachage). Les arbres de Merkle permettent d'obtenir une signature qui peut être utilisée un nombre prédéfini de fois avec une petite clé publique et une signature constituée de la clé publique de la signature à usage unique acquise lors de sa signature et de quelques renseignements sur l'arbre de Merkle.

Avec les arbres de Merkles nous obtenons une signature de taille $\sim 28kb$.

Chapitre 7

Signature d'anneau

Nous présentons dans ce chapitre une amélioration d'une signature d'anneau à seuil présenté par Aguilar, Cayrel et Gaborit dans [2] en 2008. Cette signature avait pour inconvénient d'avoir une taille correspondant à peut près à N signature, N étant la taille de l'anneau. Comme la signature ressemble à une signature de Stern et qu'il est intéressant que N soit grand, cela donne au finale une signature de taille importante. La modification apportée et expliquée dans ce chapitre permet d'avoir une signature correspondant à peu près à t fois une signature qui ressemble à la signature de Stern. Cela est souvent beaucoup plus efficace en terme de communication. En particulier, dans le cas d'une signature d'anneau classique, $t = 1$, ce qui rend le gain obtenu très significatif. Ce chapitre est une adaptation d'un schéma basé sur les réseaux proposé par Bettaieb et moi même dans [8] à PQcrypto 2013.

7.1 Problématique

La notion de signature de groupe a été formalisée en premier par Chaume et van Heyst en 1991 dans [23]. Une signature de groupe est une signature qui autorise un utilisateur d'un groupe a signer de façon anonyme tout en prouvant son appartenance à ce groupe. Le groupe est administré par un manager, qui peut ajouter des utilisateurs au groupe et qui a la capacité de déterminer la vraie identité du signeur quand il le juge nécessaire.
Rivest, Shamir et Tauman introduisent le concept de signature d'anneau dans [83] et proposent un schéma basé sur le cryptosystème RSA. Contrairement à la signature de groupe, la signature d'anneau n'a pas besoin de manager. Les auteurs proposent comme utilisation, la situation où une personne publique hautement placée dans la hiérarchie gouvernementale voudrait signer une information pour un journaliste, sans révéler quel membre du gouvernement est à l'origine du message.

Ce concept a ensuite été étendu à la signature d'anneau à seuil par Bresson, Stern et Szydlo dans [11]. Dans ce cas, t utilisateurs parmi N interagissent pour produire une signature sans donner

d'autres informations sur l'ensemble de personne qui a signé que le nombre de personne en lui même. Depuis cette définition, plusieurs signatures d'anneau à seuil ont été proposées, [63, 29, 97]. Ces construction ont la particularité d'être en complexité $\mathcal{O}(Nt)$.

Aguilar, Cayrel et Gaborit [2] ont proposé une nouvelle signature d'anneau à seuil en 2008, basée sur les codes correcteurs, avec une complexité en $\mathcal{O}(N)$.

En 2010, Cayrel, Lindner, Rückert et Silva ont présentés dans [16] une version réseau de ce schéma.

7.2 Définitions

Dans cette partie nous présentons les définitions liées à la signature d'anneau.

Signature d'anneau

La définition est la même que dans [1]. Une signature d'anneau est une signature numérique où le signeur n'est pas connu. Toutefois, la vérification de celle-ci donne une preuve de l'appartenance du signeur à un ensemble donné.

Définition 7.1. (Signature d'anneau) une signature d'anneau est composée de trois algorithmes :

- R.KeyGen : Un algorithme probabiliste en temps polynômiale qui prend en entrée un paramètre de sécurité et donne en sortie un couple de clé, clé privée et clé publique.
- R.Sign : Un algorithme probabiliste en temps polynômiale qui prend en entrée un ensemble de clés publiques $PK_1, \ldots, PK_N$, un message μ, et une clé de signature. La sortie est une signature d'anneau du message μ.
- R.Verify : Un algorithme déterministe en temps polynômiale qui prend en entrée une signature d'anneau, un message μ et la liste de clé publique des utilisateurs de l'anneau. La sortie est accept si la signature est valide ou reject si elle n'est pas valide.

Signature d'anneau à seuil

En 2002, Bresson, Stern et Szydlo ont introduit le concept de signature d'anneau à seuil [11]. Dans un tel schéma, un ensemble de t utilisateurs peuvent collaborer afin de produire une signature qui garantie leur anonymat au seins de cet ensemble. Nous donnons une définition de t-parmi-N signatures d'anneau à seuil. Notons l'ensemble des utilisateurs $U' = \{1, \ldots, N\}$ et S' l'ensemble des signeurs, avec $S' \subset U'$. Chaque utilisateur i dans U' a un clé secrète et une clé publique (PK_i, SK_i).

Définition 7.2. (Signature d'anneau à seuil) Une signature d'anneau à seuil est composée de trois algorithmes :

- $\mathsf{T.KeyGen}$: renvoie une clé publique PK et une clé privée SK.
- $\mathsf{T.Sign}(\mu, U', S')$: Un protocole interactif entre t utilisateurs qui prend en entrée un ensemble de clé publiques correspondant aux utilisateurs dans U', un ensemble de t clés privées correspondant aux utilisateurs dans S' et un message μ. Sa sortie est une t-signature d'anneau à seuil σ de μ.
- $\mathsf{T.Verify}(\mu, \sigma, t, U')$: Un algorithme déterministe qui prend en entrée la valeur t, un ensemble de clés publiques correspondant aux utilisateurs dans U', un message μ et une signature σ. Il renvoie accept ou reject.

7.3 Modèles de sécurité pour les signatures d'anneau à seuil

Un schéma de signature d'anneau à seuil est dit sécurisé si il a les propréiétés d'anonymat de source hiding et d'inforgeabilité définies si dessous.

Définition 7.3. (Anomynat indistingable) Etant donné une t-parmi-N signature d'anneau à seuil et un algorithme probabiliste polynômial $\mathcal{A}$ décrivant un adversaire, on considère le jeu suivant :

1. Pour i de 1 à N, générer (PK_i, SK_i). Donner l'ensemble des clés publiques $P = \{PK_1, \ldots, PK_N\}$ à $\mathcal{A}$. L'adversaire $\mathcal{A}$ à aussi accès à un oracle de signature $\mathsf{OT.Sign}(\cdot, \cdot, \cdot)$, qui renvoie $\sigma = \mathsf{T.Sign}(\mu, P, S)$ en entrant (μ, P, S) avec S un ensembles de signeurs.
2. $\mathcal{A}$ renvoie un message μ, différents ensembles $\{i_{1,0}, \ldots, i_{t,0}\}$, $\{i_{1,1}, \ldots, i_{t,1}\}$ et un anneau P pour lequel $PK_{i_{l,j}} \in P$ pour $l \in \{0, 1\}$ et $j \in \{1, \ldots, t\}$. L'adversaire $\mathcal{A}$ reçoit $\{SK_1, \ldots, SK_N\} \setminus \{SK_{i_{1,0}}, \ldots, SK_{i_{t,0}}\}$.
   Ensuite, un bit aléatoire b est choisit, puis $\sigma \leftarrow \mathsf{T.Sign}(\mu, P, \{SK_{i_{1,b}}, \ldots, SK_{i_{t,b}}\})$ est envoyé à $\mathcal{A}$.
3. L'adversaire renvoie un bit b', et réussit le jeu si $b' = b$.

Inforgeabilité. Nous donnons ici une définition formelle de la définition d'inforgeabilité existentielle avec messages choisis dans le cas d'une t-parmi-N signature d'anneau à seuil. La définition est décrite en utilisant un jeu entre un forger $\mathcal{F}$ et un challenger $\mathcal{C}$. Cette définition est similaire à celle utilisée dans [1] et [16].

Définition 7.4. (Inforgeabilité existentielle)
Une signature d'anneau à seuil est dite existentiellement inforgeable par une attaque à messages choisis si pour un forger $\mathcal{F}$ représenté par un algorithme probabiliste en temps polynômial, la probabilité que $\mathcal{F}$ réussisse le jeu suivant est négligeable :

1. Le challenger $\mathcal{C}$ génère les clés privées $\{PK_i, SK_i\}_{i=1}^N$, et envoie l'ensemble de clés publiques $P = \{PK_i\}_{i=1}^N$ à $\mathcal{F}$.
2. $\mathcal{F}$ a accès à un oracle de signature donné par la définition 7.3.
3. $\mathcal{F}$ a aussi accès à un oracle de révélation de clé $\mathsf{OExp}(\cdot)$, il retourne la clé secrète SK_i avec i comme entrée.

4. $\mathcal{F}$ renvoie une t-parmi-N signature d'anneau à seuil $\sigma^\star$ pour un nouveau message $\mu^\star$.

L'adversaire $\mathcal{F}$ réussit si la vérification $\mathsf{T.Verify}(\mu^\star, \sigma^\star, t, P) = 1$, $\mu^\star$ n'a pas déjà été demandée par $\mathcal{F}$ lors d'une demande à l'oracle de signature de l'étape 2 et que le nombre de clés révélées est strictement inférieur à t.

7.3.1 Schéma d'authentification de base

Input : q, n, w
$\mathbf{x} \xleftarrow{\$} \{0,1\}^n$, avec $wt(\mathbf{x}) = w$
$\mathbf{H} \xleftarrow{\$} (\mathbb{F}_q)^{(n-k)\times n}$
$\mathbf{y} \leftarrow \mathbf{H}\mathbf{x}^T \bmod q$
h, une fonction de hachage.
Output : $(SK, PK) = (\mathbf{x}, (\mathbf{y}, \mathbf{H}, h))$

FIGURE 7.1 – **Algorithme de génération de clé**

P choisit $\sigma \xleftarrow{\$} S_n$, $\mathbf{u} \xleftarrow{\$} (\mathbb{F}_q)^n$, $\mathbf{r}_0 \xleftarrow{\$} \{0,1\}^n$ et $\mathbf{r}_1 \xleftarrow{\$} \{0,1\}^n$.
P calcule $c_0 \leftarrow h(\sigma || \mathbf{H}\mathbf{u}^\mathbf{T} || \mathbf{r}_0)$ et $c_1 \leftarrow h(\sigma(\mathbf{u}) || \sigma(\mathbf{x}) || \mathbf{r}_1)$.

1. [**première mise en gage**] P envoie c_0 et c_1 à V.
2. [**premier challenge**] V envoie $\alpha \xleftarrow{\$} \mathbb{F}_q$ à P.
3. [**seconde mise en gage**] P calcule $\beta = \sigma(\mathbf{u} + \alpha\mathbf{x})$ et envoie β à V.
4. [**second challenge**] V envoie $b \xleftarrow{\$} \{0,1\}$, à P.
5. [**réponse finale**]
   Si $b = 0$ alors
   P envoie σ et $\mathbf{r}_0$ à V.
   Si $b = 1$ alors
   P envoie $\sigma(\mathbf{x})$ te r_1 à V.

Vérification :
Si $b = 0$ alors V vérifie si

$$c_0 \stackrel{?}{=} h(\sigma \| \mathbf{H}\sigma^{-1}(\beta)^T - \alpha\mathbf{y} \| \mathbf{r}_0)$$

Si $b = 1$ alors V vérifie, $wt(\sigma(\mathbf{x})) \stackrel{?}{=} w$ et

$$c_1 \stackrel{?}{=} h(\beta - \alpha\sigma(\mathbf{x}) \| \sigma(\mathbf{x}) \| \mathbf{r}_1)$$

FIGURE 7.2 – **Protocole d'authentification de base**

7.4 Signature d'anneau

Une signature d'anneau est une signature numérique où le signeur n'est pas connu. Par contre, son appartenance a un ensemble donné peut être vérifié. Ici nous expliquons l'idée des schémas décris dans [2] et [16], ensuite nous montrons la différence avec le nouveau schéma. Nous considérons U un ensemble de N utilisateurs.

7.4.1 Description de la signature d'anneau de base

La signature d'anneau décrite dans [16] est obtenue en appliquant la transformation de Fiat-Shamir à partir d'un schéma d'authentification. L'idée est de construire, pour chaque utilisateur i, une clé secrète $\mathbf{x}_i$ avec la même propriété $\mathbf{H}\mathbf{x}_i^T = 0$. Par conséquent, on ne peut pas distinguer deux utilisateurs à partir de leurs clés publiques. Le problème est de construire plusieurs clés secrètes à syndrome nul. En effet, statistiquement, un seul vecteur $\mathbf{v}$ de petit poids w tel que $\mathbf{H}\mathbf{v}^T = 0$ est susceptible de vérifier cette propriété avec les paramètres de sécurité choisis. La construction consiste à changer la matrice $\mathbf{H}$ pour chaque utilisateur. Ils sont maintenant identifiés par leurs matrices publiques $\mathbf{H}_i$ au lieux de l'être par leurs syndromes $\mathbf{H}_i\mathbf{x}_i^T$. Pour générer une authentification d'anneau, N authentifications sont faites, une pour chaque utilisateur. Le vrai signeur utilise sa clé privée $\mathbf{x}_i$ pour l'une des authentifications et le vecteur nul pour les autres. La propriété d'anonymat est obtenue avec une permutation des mises en gage des utilisateurs.

7.4.2 Nouveau schéma de signature d'anneau

Ce schéma utilise la même idée de génération de clé que l'algorithme de base présenté ici 7.3.1. De plus, la matrice $\mathbf{M}$ est une nouvelle donnée publique. La matrice $\mathbf{M}$ est composée de tous les éléments publiques $\mathbf{y}_1, \ldots, \mathbf{y}_N$ avec $\mathbf{H}\mathbf{x_i}^T = \mathbf{y}_i$ pour $i \in \{1, \ldots, N\}$.

Cette signature d'anneau peut être obtenue à partir du schéma d'authentification d'anneau décrit dans la figure 7.3 en appliquant la transformation de Fiat-Shamir. Ce schéma ressemble beaucoup au schéma d'authentification de base décrit dans la section 7.3.1. Comme c'est détaillé dans la figue 7.3, une nouvelle mise en gage β' et deux nouveaux masques $\mathbf{u}'$ et Σ sont ajoutés ainsi qu'une valeur secrète δ_j. Ces nouveaux éléments ont été intégrés pour garantir l'anonymat du vrai signeur. L'idée est d'utiliser δ_j comme un secret qui permet d'identifier le vrai signeur. Pour le masquer on utilise la même méthode que pour la clé secrète $\mathbf{x}_i$.
Dans ce schéma, le syndrome $\mathbf{y}_j$, qui identifie le signeur i, et qui est utilisé pour calculer la première mise en gage, est masqué par β'. En effet, on a $\mathbf{y}_j = \mathbf{M}\delta_j^T$ avec δ_j masqué par Σ et $\mathbf{u}'$. La première mise en gage peut être calculée de la même façon pour tous les signeurs car $\mathbf{H}(\mathbf{x}_i + \mathbf{u})^T - \mathbf{M}(\delta_i + \mathbf{u}')^T$

est égal pour tous les signeurs i. On utilise la même construction pour masquer $\mathbf{x}_i$ et δ_i car ces deux valeurs sont identifiées par leurs poids et leur image par une matrice ($\mathbf{H}$ ou $\mathbf{M}$).
Le fait que $wt(\Sigma(\delta_j)) = 1$ avec Σ une permutation, garantie que l'un des utilisateur de l'anneau U' est le vrai signeur.

Soit $U' = \{\text{utilisateurs}\}$. Soit $\mathbf{M}$ la matrice de toutes les clés publiques $\mathbf{y}_1, \ldots, \mathbf{y}_N$ des utilisateurs dans U'.

$\mathbf{M} = \begin{pmatrix} | & | & \cdots & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_N \\ | & | & \cdots & | \end{pmatrix}$, avec $\mathbf{y}_i = \mathbf{H}\mathbf{x}_i^T$ pour tout $i \in \{1, \ldots, N\}$.

L'utilisateur S indexé par j dans $\{1, \ldots, N\}$, procède comme suit :
Il construit $\delta_j \in \{0,1\}^N$ avec 1 dans la j-ème position et 0 ailleurs. Il choisit une permutation aléatoire Σ de $\{1, \ldots, N\}$, $\mathbf{u}' \xleftarrow{\$} (\mathbb{F}_q)^N, \sigma \xleftarrow{\$} S_n$, $\mathbf{u} \xleftarrow{\$} (\mathbb{F}_q)^n$, $\mathbf{r}_0 \xleftarrow{\$} \{0,1\}^n$ et $\mathbf{r}_1 \xleftarrow{\$} \{0,1\}^n$.
Il calcule :

$$c_0 \leftarrow h(\sigma||\Sigma||\mathbf{H}\mathbf{u}^T - \mathbf{M}\mathbf{u}'^T||\mathbf{r}_0) \text{ et } c_1 \leftarrow h(\sigma(\mathbf{u})||\Sigma(\mathbf{u}')||\sigma(\mathbf{x}_j)||\Sigma(\delta_j)\|\mathbf{r}_1).$$

1. [**première mise en gage**] S envoie c_0 et c_1 au vérifieur V.

2. [**premier challenge**] V envoie $\alpha \xleftarrow{\$} \mathbb{F}_q$ à S.

3. [**seconde mise en gage**] S envoie β et β' à V, avec

$$\beta = \sigma(\mathbf{u} + \alpha\mathbf{x}_j) \text{ et } \beta' = \Sigma(\mathbf{u}' + \alpha\delta_j).$$

4. [**second challenge**] V envoie $b \xleftarrow{\$} \{0,1\}$ à S.

5. [**réponse finale**]
   Si $b = 0$ alors
   S envoie $\phi = \Sigma, \psi = \sigma$, et $\mathbf{a} = \mathbf{r}_0$ à V.
   Si $b = 1$ alors
   S envoie $\chi = \Sigma(\delta_j), \mathbf{d} = \sigma(\mathbf{x}_j)$ et $\mathbf{e} = \mathbf{r}_1$ à V.

Vérification :

Si $b = 0$ alors V vérifie si
$$c_0 \stackrel{?}{=} h(\psi\|\phi\|\mathbf{H}\psi^{-1}(\beta)^T - \mathbf{M}\phi^{-1}(\beta')^T\|\mathbf{a})$$

Si $b = 1$ alors V vérifie si
$$c_1 \stackrel{?}{=} h((\beta - \alpha\mathbf{d})\|(\beta' - \alpha\chi)\|\mathbf{d}\|\chi\|\mathbf{e})$$
$$\mathbf{d} \stackrel{?}{\in} \{0,1\}^n \text{ , } wt(\mathbf{d}) \stackrel{?}{=} w \text{ et } wt(\chi) \stackrel{?}{=} 1.$$

FIGURE 7.3 – **Schéma d'authentification d'anneau**

7.4.3 Propriétés du schéma

Ce schéma, sans β', u' et Σ, est le même que le schéma d'authentification de base. On a pas besoin d'envoyer N authentifications pour obtenir l'anonymat comme ce fut le cas dans les précédentes versions décrites précédemment. Ainsi, on obtient avec ce nouveau schéma une réduction significative de la taille de la signature pour des valeurs raisonnables de N et t.
Une autre propriété intéressante est que ce nouveau schéma n'utilise qu'une unique matrice $\mathbf{H}$ au lieu de N différentes matrices $\mathbf{H}_i$ comme dans [16]. Une conséquence est une réduction significative de la clé publique.

7.5 Signature d'anneau à seuil

La signature d'anneau à seuil est une généralisation de la signature d'anneau. Cette signature est créée par plusieurs signeurs, contrairement à la signature d'anneau qui est créée par un seul signeur.

Dans [11] et [2], les auteurs expliquent qu'une signature d'anneau à seuil n'est pas une répétition de plusieurs signatures d'anneau. La raison à cela est que dans la signature d'anneau à seuil, une preuve doit être donnée sur le nombre de signeurs. Dans le cas de plusieurs signatures d'anneau, un même signeur peut être à l'origine de plusieurs signature à lui tout seul, ce qui ne correspond à la définition de la signature d'anneau à seuil. L'idée de la signature présentée dans cette section est tout de même de faire autant de signatures d'anneau que de signeurs. La différence est que certains mécanismes sont ajoutés pour permettre de prouver le nombre de signeurs impliqués lors de la création.

7.5.1 Passer de la signature d'anneau à la signature d'anneau à seuil

La signature d'anneau à seuil est obtenue en appliquant une transformation de Fiat-Shamir au schéma d'authentification d'anneau à seuil donné en figure 7.4. L'idée est de faire une authentification d'anneau pour chaque δ_i, les δ_i étant contenus dans la matrice $\mathbf{M}$. On a plus alors qu'à prouver que les δ_i utilisés sont différents. On montre cela en vérifiant que les $\Sigma(\delta_i)$ sont différents. En effet, si $\Sigma(\delta_i) \neq \Sigma(\delta_j)$ alors $\delta_i \neq \delta_j$, avec Σ une permutation.
Au niveau de la sécurité, Σ est connu par tous les signeurs. Cela ne pose pas de problème pour la preuve d'inforgeabilité car Σ masque seulement δ_i, qui correspond aux identités des signeurs. De plus, aucunes information à propos de $\mathbf{x}_i$ n'est donnée. La preuve d'anonymat n'est pas non plus compromise car les signeurs sont supposés se connaitre lors de la création de la signature.
Lors de la vérification, lorsque $b = 0$, on a juste à vérifier que chaque $\Sigma(\delta_i)$ est différent des autres. Ceci prouve que t signeurs différents ont coopéré pour générer la signature d'anneau à seuil.

7.5.2 Signature d'anneau à seuil

Dans cette sous-section, nous présentons la signature d'anneau à seuil. Ce schéma est décrit dans la figure 7.4, l'algorithme de vérification est donné en figure 7.5.

Soit $\mathbf{M}$ la matrice de toutes les clés publiques $\mathbf{y}_1, \ldots, \mathbf{y}_N$ des utilisateurs dans U'.
$\mathbf{M} = \begin{pmatrix} | & | & \cdots & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_N \\ | & | & \cdots & | \end{pmatrix}$, avec $\mathbf{y}_i = \mathbf{H}\mathbf{x}_i^T$ pour tout $i \in \{1, \ldots, N\}$.
$\delta_i \in \{0,1\}^N$ le vecteur avec 1 sur la i-ème position et 0 ailleurs.
$U' = \{\text{utilisateurs}\}$ et $S' = \{\text{signeurs}\}$, avec $S' \subset U'$, $|S'| = t$ et $|U'| = N$.
L le leader, avec $L \in S'$ et V le vérifieur.

1. [**première mise en gage**]

   L construit une permutation Σ de façon aléatoire de $\{1, \ldots, N\}$.
   Pour i de 1 à t faire
   L choisit le i-ème signeur $\mathcal{S}_j$
   S_j reçoit Σ de la part de L et construit $\sigma_i \xleftarrow{\$} S_n$, $\mathbf{u}_i \xleftarrow{\$} (\mathbb{F}_q)^n$, $\mathbf{u}'_i \xleftarrow{\$} (\mathbb{F}_q)^N$, $\mathbf{r}_{0,i} \xleftarrow{\$} \{0,1\}^n$ et $\mathbf{r}_{1,i} \xleftarrow{\$} \{0,1\}^n$
   S_j calcule : $c_{0,i} \leftarrow h(\sigma_i || \Sigma \| \mathbf{H}\mathbf{u}_i^T - \mathbf{M}\mathbf{u}_i'^T || \mathbf{r}_{0,i})$
   $c_{1,i} \leftarrow h(\sigma_i(\mathbf{u}_i) || \Sigma(\mathbf{u}'_i) \| \sigma_i(\mathbf{x}_j) || \Sigma(\delta_j) \| \mathbf{r}_{1,i})$
   S_j envoie $c_{0,i}$ et $c_{1,i}$ à L.
   fin
   L calcule $\mathbf{r}_0 \xleftarrow{\$} \{0,1\}^n$ et $\mathbf{r}_1 \xleftarrow{\$} \{0,1\}^n$. L calcule la principale mise en gage :
   $C_0 = h(c_{0,1} || \ldots || c_{0,t} || \mathbf{r}_0)$ et $C_1 = h(c_{1,1} || \ldots || c_{1,t} \| \mathbf{r}_1)$

   L envoie C_0 et C_1 à V.

2. [**premier challenge**] V envoie α, tel que $\alpha \xleftarrow{\$} \mathbb{F}_q$.

3. [**seconde mise en gage**]

   On dénote $\bar{\mathbf{x}}_i$ (respectivement $\bar{\delta}_i$) le $\mathbf{x}_j$ (respectivement δ_j) correspondant au i-ème signeur $\mathcal{S}_j$.
   Pour i de 1 à t faire :
   L choisit le i-ème signeur $\mathcal{S}_j$
   S_j calcule $\beta_i \leftarrow \sigma_i(\mathbf{u}_i + \alpha \bar{\mathbf{x}}_i)$ et $\beta'_i \leftarrow \Sigma(\mathbf{u}'_i + \alpha \bar{\delta}_i)$
   S_j envoie β_i, β'_i à L.
   fin
   L envoie $\mathbf{v}_1 = \beta_1, \ldots, \mathbf{v}_t = \beta_t$ et $\mathbf{w}_1 = \beta'_1, \ldots, \mathbf{w}_t = \beta'_t$ à V.

4. [**second challenge**] V envoie $b = 0$ ou $b = 1$ à L.

5. [**réponse finale**]

   Si $b = 0$ alors
   Pour i de 1 à t faire
   L choisit le i-ème signeur $\mathcal{S}_j$
   S_j envoie $\sigma_i, \mathbf{r}_{0,i}$ à L
   fin
   Si $b = 1$ alors
   Pour i de 1 à t faire
   L choisit le i-ème signeur $\mathcal{S}_j$
   S_j envoie $\mathbf{u}_i, \mathbf{u}'_i$ et $\mathbf{r}_{1,i}$ à L
   fin
   Si $b = 0$ alors
   L envoie $\phi = \Sigma$, $\psi_1 = \sigma_1, \ldots, \psi_t = \sigma_t$, $\mathbf{a}_1 = \mathbf{r}_{0,1}, \ldots, \mathbf{a}_t = \mathbf{r}_{0,t}$ et $\rho = \mathbf{r}_0$ à V.
   Si $b = 1$ alors
   L envoie $\chi_1 = \Sigma(\bar{\delta}_1), \ldots \chi_t = \Sigma(\bar{\delta}_t)$, $\mathbf{d}_1 = \sigma_1(\bar{\mathbf{x}}_1), \ldots, \mathbf{d}_t = \sigma_t(\bar{\mathbf{x}}_t)$, $\mathbf{e}_1 = \mathbf{r}_{1,1}, \ldots, \mathbf{e}_t = \mathbf{r}_{1,t}$ et $\varrho = \mathbf{r}_1$ à V.

FIGURE 7.4 – **Authentification d'anneau à seuil**

Si $b = 0$ alors
  Pour i de 1 à t
    Construire $c_i = h(\psi_i \| \phi \| \mathbf{H}\psi_i^{-1}(\mathbf{v}_i)^T - \mathbf{M}\phi^{-1}(\mathbf{w}_i)^T \| \mathbf{a}_i)$
  fin
  V vérifie que $C_0 \stackrel{?}{=} h(c_1 \| \dots \| c_t \| \rho)$
Si $b = 1$ alors
  Pour i de 1 à t
    Construire $c_i = h(\mathbf{v}_i - \alpha\mathbf{d}_i \| \mathbf{w}_i - \alpha\chi_i \| \mathbf{d}_i \| \chi_i \| \mathbf{e}_i)$

    V vérifie que $wt(\mathbf{d}_i) = w$ et $\mathbf{d}_i \in \{0,1\}^n$

    V vérifie que $wt(\chi_i) = 1$
  fin
  V vérifie que $\sum_{i=1}^{t} wt(\chi_i) = t$ et $\chi_i \in \{0,1\}^N$
  V vérifie que $C_1 \stackrel{?}{=} h(c_1 \| \dots \| c_t \| \varrho)$

FIGURE 7.5 – **Algorithme de vérification**

7.6 Sécurité

Dans cette section nous prouvons la sécurité du schéma. Une signature d'anneau à seuil doit vérifier deux propriétés, la propriété de source hiding et l'inforgeabilité existentielle. La première assure l'anonymat des utilisateurs et la deuxième est une propriété classique des signatures numériques.

7.6.1 Source hiding

Lemme 7.5. Pour une transcription τ du schéma d'authentification à seuil en Figure 7.4 effectuée par un ensemble S de t signeurs, nous avons :
Pour tout ensemble S' de t signeurs il existe une transcription σ' effectuée par S' telle que la différence entre τ et τ' se trouve dans les mises en gage.

Démonstration. Nous prouvons ici le lemme pour un seul tour du schéma, tous les tours étant semblables et indépendants. Soit τ la transcription du schéma d'authentification à seuil, il est facile de voir que τ correspond à $((C_0, C_1), \alpha, (\beta_i, \beta'_i; i \in \{1, \dots, t\}), b, RSP)$ où RSP est la réponse envoyée par le leader L au vérifieur V.

Soit $(\mathbf{x}_i, \delta_i; i \in \{1, \dots, t\})$, les secrets utilisés par les signeurs de S et $(\mathbf{x}'_i, \delta'_i; i \in \{1, \dots, t\})$, les secrets utilisés par les signeurs de S'. Nous allons montrer comment choisir $(\mathbf{u}_i, \mathbf{u}'_i, \sigma_i, \Sigma)$ pour que la

transcription reste inchangée, sauf pour C_0 et C_1 lorsque les clés secrètes $(\mathbf{x}_i, \delta_i; i \in \{1, \ldots, t\})$ sont remplacées par $(\mathbf{x}'_i, \delta'_i; i \in \{1, \ldots, t\})$. Si $b = 0$, on construit $U_i = \mathbf{u}_i + \alpha\mathbf{x}_i - \alpha\mathbf{x}'_i$ et $U'_i = \mathbf{u}'_i + \alpha\delta_i - \alpha\delta'_i$. Quand on remplace $(\mathbf{x}_i, \mathbf{u}_i, \mathbf{u}'_i, \delta_i; i \in \{1, \ldots, t\})$ par $(\mathbf{x}'_i, U_i, U'_i, \delta'_i; i \in \{1, \ldots, t\})$, on obtient que β_i et β'_i conservent les mêmes valeurs. Comme $\mathbf{u}_i$ et $\mathbf{u}'_i$ n'apparaissent pas dans RSP, les seuls changements dans la transcription se trouvent dans C_0 et C_1. Si $b = 1$, on construit π_i et Π tels que $\pi_i(\mathbf{x}'_i) = \mathbf{x}_i$ et $\Pi(\delta'_i) = \delta_i$ et on note $U_i = \pi^{-1}(\mathbf{u}_i)$ et $U'_i = \Pi^{-1}(\mathbf{u}'_i)$. Quand on remplace $(\sigma_i, \mathbf{u}_i, \Sigma, \mathbf{u}'_i)$ par $(\sigma_i \circ \pi_i, U_i, \Sigma \circ \Pi, U'_i)$, on obtient que β_i et β'_i conservent les mêmes valeurs que dans RSP. Donc, les seuls changements dans la transcription se trouvent dans C_0 et C_1. Il ne reste plus qu'a étendre cette construction à tous les tours pour finir la preuve. □

Lemme 7.6. Soit n, q les paramètres du schéma, h la fonction de mise en gage et rd le nombre de tour du schéma d'authentification d'anneau à seuil. Pour chaque message μ et n'importe quelle signature d'anneau à seuil σ généré à partir de l'ensemble d'utilisateurs S par t signeurs, on a : Pour chaque ensemble S' de t signeurs, il existe une signature σ' effectuée par S' avec probabilité :

$$p = 1 - \left(1 - \frac{1}{(2q)^{rd}}\right)^{2^{n \times rd}}$$

telle que la différence entre σ et σ' ce trouve dans les mises en gage.

Démonstration. En utilisant la transformation de Fiat-Shamir, on peut voir la signature d'anneau à seuil comme le n-uplet $(R_1, CH_1, R_2, CH_2, RSP)$ avec R_1 la concaténation de C_0 et C_1 pour tous les tours, R_2 la concaténation de tous les β_i et β'_i de tous les tours et RSP la concaténation de la réponse finale de tous les tours. Les valeurs CH_1 et CH_2 sont calculées par deux oracles aléatoires $\mathcal{O}_1$ et $\mathcal{O}_2$ tels que $CH_1 = \mathcal{O}_1(\mu, R_1)$ avec μ le message et $CH_2 = \mathcal{O}_2(\mu, CH_1, R_2)$. Le lemme 7.5 établie l'existence d'une autre transcription égale à la signature, sauf pour les valeurs C_0 et C_1. Ces mises en gages sont construites en utilisant le schéma de mise en gage h avec les valeurs aléatoires $\mathbf{r}_0$ et $\mathbf{r}_1$. Pour chaque tour de la transcription, on peut prendre un autre $\mathbf{r}_0$ ou $\mathbf{r}_1$ et obtenir une autre transcription égale à la signature, sauf pour les valeurs C_0 et C_1. Pour finir la preuve, on calcule la probabilité que l'une de ces transcriptions corresponde à une signature de μ. La plupart du temps la transcription n'est pas une signature car les challenges sont différents de $\mathcal{O}_1(\mu, R_1)$ et $\mathcal{O}_2(\mu, CH_1, R_2)$. La probabilité que la transcription corresponde à une signature de μ peut être calculée par une distribution de Bernoulli de paramètre $p_B = \frac{1}{(2q)^{rd}}$ et $n_B = 2^{n \times rd}$. Le paramètre p_B correspond à la probabilité d'obtenir un challenge particulier et n_B correspond au numéro de la transcription. Donc, la probabilité d'obtenir la signature est

$$1 - \left(1 - \frac{1}{(2q)^{rd}}\right)^{2^{n \times rd}}.$$

□

Théorème 7.7. Aucuns attaquants $\mathcal{A}$ représenté par une machine probabiliste en temps polynômial ne peut gagner le jeu du source hiding dans le modèle de l'oracle aléatoire.

Démonstration. Soit S_0 et S_1 deux ensembles de t signeurs. Dans le jeu décrit en définition 7.3, on donne un challenge à l'attaquant sous forme de signature d'anneau à seuil générée par S_b (avec

$b \in \{0,1\}$) et il a à deviner b' tel que $b' = b$ avec une probabilité non négligeable. On considère que l'attaquant a accès à l'ensemble de toutes les clés secrètes et choisit deux ensembles de t signeurs S_0 et S_1. Ensuite, il choisit un message μ et demande un challenge. Après avoir reçut une signature d'anneau à seuil σ de μ signée par S_b, l'attaquant l'ensemble qui a permit de signer la signature. D'après le lemme 7.6, on obtient qu'il existe une signature σ' générée par $S_{\bar{b}}$ avec probabilité p telle que la différence entre σ et σ' est dans les mises en gage générées par h. Si l'attaquant $\mathcal{A}$ gagne le jeu du source hiding, alors $\mathcal{A}$ peut faire une distinction entre deux sorties de h avec une probabilité non négligeable, simulé par un oracle aléatoire. Cela est impossible.

□

7.6.2 Inforgeabilité

Dans cette sous-section nous prouvons l'inforgeabilité de la signature. Le raisonnement est que si un forger peut gagner le jeu de la définition 7.4, alors il pourrait résoudre un problème difficile en temps polynômial. La preuve d'inforgeabilité est donnée par le théorème 7.9.

Lemme 7.8. Si il existe un algorithme probabiliste en temps polynômial $\mathcal{A}$ qui est capable de produire une signature à seuil avec t personnes parmi N utilisateurs et une probabilité plus grande que $\left(\frac{q+2}{2q}\right)^{rd}$, alors il peut produire t valeurs qui peuvent être prises comme des clés secrètes soit il peut trouver une collision dans la fonction de mise en gage h.

Démonstration. Si $\mathcal{A}$ produit une signature d'anneau à seuil de t utilisateurs parmi N avec une probabilité plus grande que $\left(\frac{q+2}{2q}\right)^{rd}$, alors il peut faire de même, dans le schéma d'authentification correspondant, et dans au moins un tour, avec une probabilité supérieure à $\frac{q+2}{2q}$. Dans chaque tour, il y a $2q$ challenges possibles, q possibilités pour le premier challenge et 2 possibilités pour le second. Par le principe des tiroirs de Dirichlet on peut déduire que $\mathcal{A}$ peut répondre correctement pour une mise en gage fixée, à deux différents challenges α et β dans la première phase de challenge et n'importe quel challenge dans la seconde phase. De cette façon, il peut produire la transcription suivante :

$$((C_0, C_1), \alpha, (\mathbf{v}_i, \mathbf{w}_i), 0, (\phi, \psi_i, \mathbf{a}_i, \rho)), \quad i \in \{1, \ldots, t\}$$

$$((C_0, C_1), \alpha, (\mathbf{v}_i, \mathbf{w}_i), 1, (\chi_i, \mathbf{d}_i, \mathbf{e}_i, \varrho)), \quad i \in \{1, \ldots, t\}$$

$$((C_0, C_1), \beta, (\mathbf{v}'_i, \mathbf{w}'_i), 0, (\phi', \psi'_i, \mathbf{a}'_i, \rho')), \quad i \in \{1, \ldots, t\}$$

$$((C_0, C_1), \beta, (\mathbf{v}'_i, \mathbf{w}'_i), 1, (\chi_i, \mathbf{d}_i, \mathbf{e}_i, \varrho)), \quad i \in \{1, \ldots, t\}$$

telle que cette transcription passe l'algorithme de vérification pour ce tour. Ainsi, soit $\mathcal{A}$ peut trouver une collision sur h, soit on obtient les équations suivantes à partir de l'algorithme de vérification.

1. $\psi_i = \psi'_i$, $\phi = \phi'$, $\mathbf{H}\psi_i^{-1}(\mathbf{v}_i)^T - \mathbf{M}\phi^{-1}(\mathbf{w}_i)^T = \mathbf{H}\psi_i'^{-1}(\mathbf{v}'_i)^T - \mathbf{M}\phi'^{-1}(\mathbf{w}'_i)^T$, $\mathbf{a}_i = \mathbf{a}'_i$ pour $i \in \{1, \ldots, t\}$.

2. $\mathbf{v}_i - \alpha\mathbf{d}_i = \mathbf{v}'_i - \beta\mathbf{d}'_i$, $\mathbf{w}_i - \alpha\chi_i = \mathbf{w}'_i - \beta\chi'_i$, $\mathbf{d}_i = \mathbf{d}'_i$, $\chi_i = \chi'_i$, $\mathbf{e}_i = \mathbf{e}'_i$ pour $i \in \{1, \ldots, t\}$.
3. $wt(\mathbf{d}_i) = wt(\mathbf{d}'_i) = m/2$, $\mathbf{d}_i \in \{0,1\}^m$, $\mathbf{d}'_i \in \{0,1\}^m$ pour $i \in \{1, \ldots, t\}$.
4. $\sum_{i=1}^{t} wt(\chi_i) = t$, $\sum_{i=1}^{t} wt(\chi'_i) = t$, $\chi_i \in \{0,1\}^N$, $\chi'_i \in \{0,1\}^N$ pour $i \in \{1, \ldots, t\}$.

A partir des équations dans 2, on obtient $\mathbf{v}'_i = \mathbf{v}_i - \alpha\mathbf{d}_i + \beta\mathbf{d}'_i$ et $\mathbf{w}'_i = \mathbf{w}_i - \alpha\chi_i + \beta\chi'_i$. Lorsque l'on remplace $\mathbf{v}'_i$ et $\mathbf{w}'_i$ dans la troisième équation de 1, on obtient :

$$\mathbf{H}\psi_i^{-1}(\mathbf{v}_i)^T - \mathbf{M}\phi^{-1}(\mathbf{w}_i)^T = \mathbf{H}\psi_i^{-1}(\mathbf{v}_i - \alpha\mathbf{d}_i + \beta\mathbf{d}_i)^T - \mathbf{M}\phi^{-1}(\mathbf{w}_i - \alpha\chi_i + \beta\chi_i)^T$$

$$0 = (\beta - \alpha)(\mathbf{H}\psi_i^{-1}(\mathbf{d}_i)^T - \mathbf{M}\phi^{-1}(\chi_i)^T)$$

Puisque $\alpha \neq \beta$, on a l'égalité $\mathbf{H}\psi_i^{-1}(\mathbf{d}_i)^T = \mathbf{M}\phi^{-1}(\chi_i)^T$. A partir de l'équation dans 4, $\mathbf{M}\phi^{-1}(\chi_i)^T$ correspond à t différentes clés publiques et $\psi_i^{-1}(\mathbf{d}_i)$ peut être utilisé pour simuler t différentes clés secrètes. □ □

Dans le théorème suivant, nous prouvons l'inforgeabilité de la signature d'anneau à seuil.

Théorème 7.9. (Inforgeabilité) Si un forger peut gagner le jeu de la définition 7.4 avec probabilité p' en temps polynômial, alors celui-ci peut résoudre une instance du problème de décodage par syndrome en temps polynômial avec une probabilité supérieur à $p'p\frac{1}{N^2}$.

Démonstration. On donne au challenger $\mathcal{C}$ une instance du problème SD, $(\mathbf{H}, \mathbf{y})$. Ensuite, $\mathcal{C}$ choisit $k \in \{1 \ldots N\}$ et construit $\mathbf{x}_k := 0$, $\mathbf{y}_k := \mathbf{y}$. $\mathcal{C}$ génère $N-1$ clés $(\mathbf{x}_i, \mathbf{y}_i)$ avec $i \in \{1 \ldots N\}$ et $i \neq k$.

On commence le jeu de la définition 7.4 avec le forger $\mathcal{F}$ et les paires $(\mathbf{x}_i, \mathbf{y}_i)$ avec $i \in \{1 \ldots N\}$, en tant que paires de clés. On peut remarquer que seul la paire de clés $(\mathbf{x}_k, \mathbf{y}_k)$ n'est pas générée en tant que paire de clés valides et tant que $\mathbf{x}_k$ n'est pas révélé, l'ensemble des paires de clés $(\mathbf{x}_i, \mathbf{y}_i)$ est indistinguable d'un ensemble de paires de clés valides.

Le challenger $\mathcal{C}$ simule l'oracle de signature $\mathsf{OT.Sign}$. Lorsque $\mathcal{F}$ fait une requête à l'oracle de signature, il envoie une demande au challenger $\mathcal{C}$ en précisant les clés publiques $\mathbf{y}_i$, $i \in I$ et $I \subseteq N$. Si k n'est pas dans l'ensemble I, le challenger $\mathcal{C}$ est capable de produire une signature correcte. Sinon, le challenger produit une signature σ en remplaçant $\mathbf{x}_k$ par $\mathbf{x}_j$ avec $j \notin I$. D'après le lemme 2, il existe une signature σ' telle que σ' aurait pu être produite en utilisant la clé secrète correspondant à $\mathbf{y}_i$ avec probabilité p.

Le challenger $\mathcal{C}$ simule aussi l'oracle OExp. Si le forger $\mathcal{F}$ demande la valeur de $\mathbf{x}_k$, le jeu se termine sans qu'il n'y ait eu de forge de la signature. Tant que le forger ne demande pas toutes les clés secrètes, la probabilité qu'il demande pour $\mathbf{x}_k$ est moins que $\frac{N-1}{N}$. Sinon, le forger $\mathcal{F}$ gagne le jeu et sort une signature valide en temps polynômial avec probabilité p'.

D'après le lemme 8.6, $\mathcal{F}$ peut soit simuler t différentes clés secrètes soit trouver une collision sur la fonction de mise en gage h. Si il réussit à simuler t clés secrètes, au moins l'une d'entre elles, $\mathbf{x}_l$,

n'était pas une clé provenant de l'oracle. Le vecteur $\mathbf{x}_l$ est tel que $wt(\mathbf{x}_l) = w$ et $\mathbf{H}\mathbf{x}_l^T = \mathbf{y}_l$. Comme k a été choisit uniformément dans $\{1, \ldots, N\}$, la probabilité que k soit égale à l est $1/N$. Si l est égale à k, alors $\mathbf{x}_l$ est une solution de l'instance du problème SD, $(\mathbf{H}, \mathbf{y})$.

Avec l'interaction du challenger et du forger, on peut construire un algorithme probabiliste en temps polynômial qui résout une instance du problème SD avec une probabilité supérieur à $pp'\frac{1}{N^2}$. □ □

7.7 Paramètres

Dans cette section, nous comparons le nouveau schéma de signature d'anneau avec l'ancien schéma dans [2].

7.7.1 Paramètres

Dans le tableau suivant, nous comparons nos paramètres avec [2], le seul schéma de signature d'anneau basé sur les codes. Nous utilisons comme paramètres $n = 698$, $k = 349$ et $w = 70$. Nous considérons que les valeurs aléatoires peuvent être envoyés grâce à leur graine de taille 80 bits. La fonction de hachage est a valeur dans 2^{160}.

7.7.2 Signature d'anneau

Dans la table suivante, on observe une réduction significative de la taille de la signature dans le nouveau schéma par rapport à l'ancien. Le nombre de membres de l'anneau est donné par N.

N	100	1000	5000	10000	100000
ACG	$44,717$	$446,801$	$2233,841$	$4467,641$	$44676,041$
Schéme en Figure 7.3	$0,5$	$0,608$	$1,088$	$1,688$	$12,488$

TABLE 7.1 – Comparaison des schémas de signature d'anneau en Mbytes.

7.7.3 Signature d'anneau à seuil

Dans le tableau suivant on peut voir différentes tailles de signatures en fonction des paramètres t et N. Le paramètre t correspond au nombre de signeurs et N le nombre de membres de l'anneau.

N	100	100	100	100	100	100
t	2	10	30	50	70	100
ACG	44, 717	44, 717	44, 717	44, 717	44, 717	44.717
Schéma en Figure 7.4	0, 955	4, 593	13, 687	22, 781	31, 876	45, 517

TABLE 7.2 – Comparaison des schéma de signature d'anneau à seuil en Mbytes.

N	200	200	200	1000	1000	1000
t	2	10	50	2	10	50
ACG	89, 393	89, 393	89, 393	446, 801	446, 801	446, 801
Schéma en Figure 7.4	0, 975	4, 677	23, 185	1, 135	5, 349	26, 417

TABLE 7.3 – Comparaison des schéma de signature d'anneau à seuil en Mbytes.

On peut voir que le schéma de signature d'anneau à seuil à une taille proche de t signatures d'anneau et ne dépend pas énormément du paramètre N pour les paramètres donnés.

Chapitre 8

Signature indéniable basée sur les codes correcteurs

Dans ce chapitre, nous présentons la première signature indéniable basée sur la théorie des codes correcteurs. Plus généralement, il s'agit même de la première signature indéniable en cryptographie post-quantique. Le schéma est basé sur une variation du protocole de Stern [91]. Dans un premier temps nous présentons une signature à usage unique très simple, puis nous montrant comment la transformer en signature à usage illimité à l'aide d'une autre signature numérique standard. Le schéma utilise des opérations rapides et des clés de petites tailles. A la fin du chapitre, nous donnons les idées pour obtenir la propriété de non-transférabilité à partir de cette signature. Cette signature a fait l'objet d'un article accépté à IMAC 2013 sous le nom "A Code-Based Undeniable Signature Scheme", par C.Aguilar Melchor, S.Bettaieb, P.Gaborit et moi même.

8.1 Introduction

Le concept de signature indéniable a été introduit par Chaume et Antwerpen [22]. Le but était de limiter le nombre de vérification liée a un document secret signé. Le document est connu, mais la la preuve de son authenticité n'est pas accordé à tout le monde. Depuis leurs introduction en 1989, les signatures indéniables ont eu de nombreuses applications, le "software licensing" [20], le vote électronique [85] et aussi la monnaie électronique [21, 79]. La signature indéniable est composée de quatre algorithmes. Un algorithme de génération de clé et un algorithme de signature (comme dans la signature numérique) et deux algorithmes de vérification interactifs. Ces deux algorithmes remplacent la vérification universelle de la signature numérique et contiennent tout le sens de la signature indéniable. Le premier algorithme est une vérification interactive de la bonne formation d'une signature et le deuxième est une vérification interactive de la mauvaise formation d'une signature.
La signature indéniable est sécurisée si elle est inforgeable, invisible et résiste à l'usurpation d'identité.

Elle peut en plus avoir la propriété de non-transférabilité.

8.2 Préliminaires

8.2.1 Notations

Nous utilisons la syntaxe || pour la concaténation, h pour une fonction de hachage et h' pour la fonction de hachage dans $\mathcal{M}_{n,t}(\mathbb{F}_2)$. En ce qui concerne les protocoles, je note $\mathcal{P}$ le prouveur et $\mathcal{V}$ le vérifieur. La clé secrète du prouveur est $K_s^{\mathcal{P}}$ et sa clé publique est $K_p^{\mathcal{P}}$.
Pour cette construction nous avons besoin d'une signature générique Sign avec (K_s^S, K_p^S) le couple de clé privée et clé publique associée. Pour signer x avec Sign nous utilisons la notation $\mathsf{Sign}(x, K_s^S)$. Pour le choix de cette signature, plusieurs sont possibles ([26, 40]).
Nous utilisons les notations standards de la théorie des codes, avec $\mathcal{C}$ un code de paramètres $[n, k]$, H une matrice de contrôle du code $\mathcal{C}$ ($H \in \mathcal{M}_{n-k,n}(\mathbb{F}_2)$). On note w l'entier qui correspond au poids de Hamming d'un mot.
On note S_w^n l'ensemble de mots dans $\mathbb{F}_2$ de poids de Hamming w et wt la fonction qui donne ce poids de Hamming.
Dans ce chapitre, les références aux matrices et aux vecteurs sont en gras.

8.2.2 Gestion des clés

8.2.3 Définitions

Définition 8.1. Une signature indéniable est l'association de quatre algorithmes polynômiaux définis comme ceci :

- $\mathsf{KG}_{\mathcal{P}}(n, k)$: Ce protocole permet d'obtenir un couple de clé privée et clé publique $(K_s^{\mathcal{P}}, K_p^{\mathcal{P}})$ à partir des paramètres de sécurité (n, k).
- USign : Ce protocole permet d'obtenir une signature σ à partir d'une clé privée $K_s^{\mathcal{P}}$ et d'un message m.
- $\mathsf{Confirmation}_{\mathcal{P},\mathcal{V}}$: Ce protocole est un protocole interactif entre un prouveur $\mathcal{P}$ et un vérifieur $\mathcal{V}$. Le prouveur utilise comme entrée ses clés (privée et publique) alors que le vérifieur utilise comme entrée un message m et une signature σ. Le prouveur n'a pas de sortie et le vérifieur à soit une sortie égale à "$\mathsf{Success}$" si la signature σ est valide soit "Fail" sinon.
- $\mathsf{Desaveu}_{\mathcal{P},\mathcal{V}}$: Ce protocole est un protocole interactif entre un prouveur $\mathcal{P}$ et un vérifieur $\mathcal{V}$. Le prouveur utilise comme entrée ses clés (privée et publique) alors que le vérifieur utilise comme entrée un message m et une signature σ. Le prouveur n'a pas de sortie et le vérifieur à soit une sortie égale à "$\mathsf{Success}$" si la signature n'est pas valide soit "Fail" sinon.

8.2.4 Modèle de sécurité

Dans cette sous-section je donne les définitions et les notions de sécurité liées à la signature indéniable :

Définition 8.2. (Inforgeablité) L'inforgeabilité d'une signature indéniable est définie par le jeu suivant entre un challenger CH et un forgeur F. Le challenger CH commence le jeu en exécutant $\mathsf{KG}_{\mathcal{P}}(n,k)$ et obtient (KP_s, KP_p), ensuite il donne KP_p à F. Le forgeur est autorisé à faire des requêtes de signatures de confirmation et de désaveux. Il retourne ensuite un couple (m,σ). Le forgeur F gagne le jeu si (m,σ) est valide et si m n'a pas fait l'objet d'une requête de signature.

Définition 8.3. (Invisibilité) Soit D un distinguer en temps polynômial. On définit l'invisibilité d'une signature indéniable à l'aide du jeu suivant entre un challenger CH et un distinguer D. Le challenger CH exécute $\mathsf{KG}_{\mathcal{P}}(n,k)$ et obtient (KP_s, KP_p), puis il donne KP_p à D. Le distinguer D est autorisé à faire des requêtes à un oracle de signature et de vérification (confirmation et désaveux). Ensuite, D envoie un challenge sur le message m de son choix au challenger. Le challenger CH choisit b dans $\{0,1\}$ et envoie la signature de m si $b=0$ ou une signature aléatoire dans le cas contraire. Après ça, D est autorisé à faire des requêtes de signatures et de vérifications à condition que m ne soit pas sollicité. A la fin, D renvoie un bit b_0. On dit que D gagne le jeu si $b_0 = b$.

Définition 8.4. (Impersonation) Soit I un PTT attaquant (impersonator). On définit l'impersonation à l'aide du jeu suivant entre I et un vérifieur V. Pour commencer, I reçoit en entrée une clé publique KP_p. Ensuite I peut faire des requêtes de signatures et de vérifications. A la fin, I décide d'interagir avec le vérifieur V dans un protocole de confirmation ou de désaveux associé au couple (m,σ). On dit que I gagne le jeu si le vérifieur V est convaincu du protocole de vérification. Une signature indéniable est dite sécurisée contre les attaque d'impersonation à message choisit si aucun PPT I a un avantage non négligeable dans le jeu décrit précédemment.

Nous avons ici des définitions de completness et soundness propre aux signatures indéniables et qui concernent les protocoles de confirmation et de désaveux.

Définition 8.5. (Completeness) Si le prouveur $\mathcal{P}$ et le vérifieur $\mathcal{V}$ exécutent de façon honnête les protocoles de vérifications. Alors pour tout message m, nous avons que

$$Pr[\mathsf{Confirmation}_{\mathcal{P},\mathcal{V}}(m, \mathsf{USign}(K_s^{\mathcal{P}}, m)) \text{ retourne Accept}] = 1.$$

Et pour tout couple message-signature invalide (m,σ), nous avons :

$$Pr[\mathsf{Desaveu}_{\mathcal{P},\mathcal{V}}(m,\sigma) \text{ retourne Accept}] = 1.$$

Définition 8.6. (Soundness) Pour tout prouveur malhonnête $\mathcal{P}$ et tout couple message-signature invalide (m,σ), nous avons :

$$Pr[\mathsf{Confirmation}_{\mathcal{P},\mathcal{V}}(m,\sigma) \text{ retourne Accept}] = \text{ est négligeable.}$$

Pour tout couple message-signature (m,σ), nous avons :

$$Pr[\mathsf{Desaveu}_{\mathcal{P},\mathcal{V}}(m,\sigma) \text{ retourne Accept}] = \text{ est négligeable.}$$

8.3 Point de vue général

Dans cette section, nous décrivons l'idée principale pour obtenir la signature indéniable basée sur les codes correcteurs. Nous commençons par décrire cette signature avec les propriétés d'inforgeabilité et d'invisibilité avant d'expliquer dans une autre partie une autre version qui a la propriété de non-transférabilité. L'idée est de considérer une signature indéniable à utilisation unique et de l'étendre à une signature à utilisation illimité à l'aide d'une autre signature numérique quelconque. Nous commençons par expliquer l'idée de la signature à utilisation unique avant de décrire celle à utilisation illimité.

8.3.1 Signature à utilisation unique

Les signatures indéniables doivent être liées à un la clé secrète, mais pas trop. Si ce lien est trop fort, nous retombons sur une signature numérique classique.

L'idée de la signature à utilisation unique est que sa clé privée est un mot $\mathbf{x}$ avec un poids de Hamming très faible. Sa clé publique est son syndrome $\mathbf{s}$, calculé avec une matrice $(n-k)\times n$ aléatoire $\mathbf{H}$. La difficulté de retrouver $\mathbf{x}$ à partir de $\mathbf{s}$ et $\mathbf{H}$ correspond au problème de décodage par syndrome. L'idée est de prendre comme signature un nouveau syndrome $\mathbf{z}$ de cette même clé secrète $\mathbf{x}$ mais produit à l'aide d'une nouvelle matrice $\mathbf{M}$ reliée au message m. On peut supposer qu'il est difficile de faire un lien entre $\mathbf{s}$ et $\mathbf{z}$, ce qui constitue une propriété nécessaire des signatures indéniables. Toutefois, deux problèmes restent à régler. Le premier est que l'ajout d'un nouveau syndrome associé à une nouvelle matrice donne de l'information sur le secret $\mathbf{x}$. Ce problème peut être réglé avec une bonne paramétrisation des valeurs n et k, dans le cas d'une signature à utilisation unique. Le second problème est de faire intervenir le message m dans la signature. Ce problème est résolut en construisant la matrice aléatoire $\mathbf{M}$ à partir d'un haché de m. On obtient la signature en faisant l'opération $\mathbf{z} = \mathbf{M}\mathbf{x}^{\mathbf{T}}$, avec $\mathbf{z}$ une signature qui dépend de $\mathbf{x}$ et de m. Ces opérations sont décrites dans le schéma suivant.

$$
\begin{array}{r}
H \rightarrow \\
\\
\\
\\
M \rightarrow \\
\\
\\
\\
\end{array}
\left(\begin{array}{cccc}
H_{1,1} & H_{1,2} & \dots & H_{1,n} \\
\dots & \dots & \dots & \dots \\
H_{n-k,1} & H_{n-k,2} & \dots & H_{n-k,n} \\
\hline
M_{1,1} & M_{1,2} & \dots & M_{1,n} \\
\dots & \dots & \dots & \dots \\
M_{k',1} & M_{k',2} & \dots & M_{k',n}
\end{array}\right)
\left(\begin{array}{c}
x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n
\end{array}\right)
=
\left(\begin{array}{c}
s_1 \\ \vdots \\ s_{n-k} \\ \hline \\ z_1 \\ \vdots \\ z_{k'}
\end{array}\right)
\begin{array}{l}
\\ \\ \leftarrow s \\ \\ \\ \\ \leftarrow z
\end{array}
$$

Nous expliquons maintenant comment obtenir les propriétés de confirmation et de désaveu avec le précédant schéma.

8.3.2 Protocoles de confirmation et de désaveu

Le but du protocole de confirmation est de rendre possible, pour un signeur, de prouver la validité de sa signature de façon interactive. Dans le cas de la signature à utilisation unique, il s'agit de prouver le lien entre le syndrome $\mathbf{s}$ et le syndrome $\mathbf{z}$ (ils ont tous les deux été construis à partir de $\mathbf{x}$). Un outil naturel pour faire de la preuve de connaissance sur les syndromes est le protocole de Stern [91]. Ce protocole permet de prouver la connaissance d'un mot de petit poids associé à un syndrome.
On introduit une variation du protocole de Stern dans laquelle on considère les deux syndromes de la signature indéniable au lieu de n'en considérer qu'un dans le cas de l'authentification. L'idée est de faire une authentification de Stern en utilisant les deux matrices, $\mathbf{H}$ et $\mathbf{M}$ en même temps. Cela prouve que le signeur connait un même mot de petit poids associé à $\mathbf{s}$ et à $\mathbf{z}$.
Le but du protocole de désaveu est de prouver qu'une signature donnée n'est pas valide. Pour cela, on utilise le même protocole que précédemment. En effet, ce protocole a pour particularité que des valeurs doivent être reconstruites à l'aide d'une fonction de hachage pour pouvoir vérifier le résultat. L'idée est donc de constater pour le vérifieur que le prouveur connait un mot de petit poids associé à $\mathbf{s}$ mais qu'il ne correspond pas à un mot de petit poids associé à $\mathbf{z}$ car les hachés ne correspondent pas lors de la reconstruction de ceux-ci.

De cette façon nous obtenons les propriétés désirées, un signeur peut seulement prouver la validité d'une signature valide et ne peut seulement prouver l'invalidité des signature invalide.

8.3.3 Passer de la signature à simple usage à une signature à usage illimité

Dans cette sous-section nous montrons comment obtenir une signature indéniable à usage illimité à partir de la signature à usage limité et d'une signature numérique quelconque. A chaque fois que le signeur veut faire une nouvelle signature, il génère un nouveau couple de clés (clé secrète et clé publique) pour la signature à usage limité. La signature illimité correspond donc à un couple composé de la signature à usage simple faite avec la clé K et de cette même clé K signée avec un algorithme de signature numérique classique.

La propriété d'invisibilité est inchangée car la signature numérique n'est utilisé que pour signer la clé K et non pour signer le message m. La preuve d'inforgeabilité de la signature illimité est basé sur la preuve d'inforgeabilité de la signature à simple usage. Le seul problème restant est celui de conserver les clés intermédiaires comme K afin de faire les protocoles de confirmation et de désaveu. Ce problème peut être réglé en envoyant, en même temps que la signature, la graine servant à générer le couple de clés intermédiaires.
Pour conclure, on se sert de la clé privée e de la clé publique d'une signature numérique pour construire une signature indéniable à usage illimité.

8.4 Schéma de la signature indéniable

On décrit ici les 4 schémas qui composent la signature indéniable basée sur les codes : le générateur de clés, l'algorithme de signature, le protocole d confirmation et le protocole de désaveu.

8.4.1 Générateur de clés

Dans cette sous-section nous présentons le générateur de clés. Il dépend d'une maîtrise aléatoire publique $\mathbf{H}$, obtenue à l'aide d'un haché d'une donnée publique et t un paramètre de sécurité. Le générateur génère un mot aléatoire r et reprend la paire de clé de Sign (la signature standard) (K_S^S, K_p^S). Dans la suite nous considérons que $k = \frac{2n}{3}$.

$\mathsf{KG}_{\mathcal{P}}(n,k)$ **:**
Génère une paire de clés

1. Choisir un code aléatoire $\mathcal{C}$ de paramètre $[n,k]$ depuis la graine sd de sa matrice de parité $H = h(sd)$.
2. Exécuter Gen pour obtenir un couple de clé valide (K_s^S, K_p^S).
3. Choisir $\mathbf{r} \xleftarrow{\$} \mathbb{F}_2^t$.
4. Retourner $(K_s^{\mathcal{P}}, K_p^{\mathcal{P}}) = ((K_s^S, \mathbf{r}), (K_p^S, \mathbf{H}))$.

FIGURE 8.1 – **Génération des clés du signeur.**

On peut prendre par exemple comme signature numérique la signature CFS ou la signature basée sur le protocole d'authentification zero-knowledge de Stern.

8.4.2 L'algorithme de signature

Le signeur veut signer un message m. En premier lieu, il a besoin de générer une clé intermédiaire $(\mathcal{K}_s, \mathcal{K}_p)$ avec EKG, le générateur de clés intermédiaires, et une graine aléatoire α.

$\mathsf{EKG}(\alpha)$ **:**
Génération d'une paire de clés $(\mathcal{K}_s, \mathcal{K}_p)$, avec w fixé.

1. Choisir $\mathbf{e} \xleftarrow{\$} S_w^n$.
2. Choisir $\mathbf{s} = \mathbf{H}\mathbf{e}^T$.
3. Retourner $(\mathcal{K}_s, \mathcal{K}_p) = (\mathbf{e}, \mathbf{s})$.

FIGURE 8.2 – **Générateur de clé intermédiaire**

Un couple de clé intermédiaire est composé d'un mot de petit poids et de son syndrome. Pour construire une signature, le signeur a besoin du haché du message $\mathbf{M} = \mathbf{h}'(\mathbf{m})$ et de calculer :

$$((\mathbf{s}, \alpha \oplus \mathbf{r}), \mathsf{Sign}((\mathbf{s}, \alpha \oplus \mathbf{r})), \mathbf{M}\mathbf{e}^T)$$

.

$\mathsf{USign}(K_s^{\mathcal{P}}, m)$ **:**

Pour un message donné m, il génère une signature comme ceci :

1. Avec $\alpha \xleftarrow{\$} \mathbb{F}_2^t$
2. Le signeur utilise $\mathsf{EKG}(\alpha)$ et obtient un couple de clés $(\mathcal{K}_s, \mathcal{K}_p) = (\mathbf{e}, \mathbf{s})$.
3. Avec $\mathbf{M} = h'(m)$ où h' est une fonction de hachage telle que $h'(m) \in \mathbb{F}_2^{n \times t}$.
4. Avec $\mathbf{z} = \mathbf{M}\mathbf{e}^T$.
5. Avec $x = (\mathbf{s}, \alpha \oplus \mathbf{r})$.
6. Avec $y = \mathsf{Sign}(x, K_s^S)$.
7. Retour $\sigma = (x, y, \mathbf{z})$.

FIGURE 8.3 – **Algorithme de signature**

Espace des signatures :

Pour la propriété d'invisibilité , nous devons introduire l'espace de signature. Même si il cet espace est évident pour la signature à usage unique, ce n'est pas le cas pour la signature usage illimité. L'espace de signature associée à la paire de clés $(KP_s; KP_p)$ est la suivante : (x, y, z) où $x \in (\mathbb{F}_2)^3$, $y = \mathsf{Sign}(x, K_s^S)$ et $z \in \mathbb{F}_2^{n-k}$. Il est clair que la signature obtenue avec l'algorithme de signature appartient à cet espace . D'autre part, avec les paramètres dans la zone des paramètres, cet espace est beaucoup plus grand que les résultats possibles de l'algorithme de signature.

Remarque 8.7. Cet espace de signature est composé de signatures valides et non valides . Toutes les signatures sont composées avec un couple message signature valide, pour la signature standard. Cela pourrait paraître dérangeant, mais il n'y a pas de contradictions avec la définition. En fait,

n'importe qui peut construire une signature non valide à partir d'une signature valide ou invalide, juste en changeant la troisième valeur z. La validité de cette signature standard signifie seulement que son propriétaire a participé à un processus de signature standard donné, mais cela ne veut pas dire ce qu'il a fait : il pourrait s'agir d'une signature précédente, pour laquelle la dernière partie z de la triplette (X, y, z) peut être valide ou non .

8.4.3 La vérification

En accord avec la définition, la signature indéniable à besoin de deux protocoles de vérification. Un protocole de confirmation et un protocole de désaveu. Les deux sont des protocoles interactifs faisant intervenir un prouveur $\mathcal{P}$ et un vérifieur $\mathcal{V}$. Le protocole de confirmation doit pouvoir prouver la validité d'une signature alors que le protocole de désaveu doit pouvoir prouver l'invalidité d'une signature. Nous décrivons dans cette sous-section les deux protocoles avec le même schéma car ces deux protocoles sont très proches. Nous désignons par "vérification" ces deux protocoles, la seule différence résidant dans l'étape de vérification $c = 1$.
Pour faire cette vérification, le signeur à besoin de la clé intermédiaire utilisée lors la signature. Pour la récupérer il utilise son mot secret r pour retrouver α et reconstruire cette clé $(\mathcal{K}_s, \mathcal{K}_p)$.

KE(σ) :

1. $\sigma = (x, y, \mathbf{z})$
2. $x = (x_1, x_2)$
3. Avec $(\mathcal{K}_s, \mathcal{K}_p) \leftarrow \mathsf{EKG}(x_2 \oplus \mathbf{r})$
4. Retour $(\mathcal{K}_s, \mathcal{K}_p, \mathbf{z}) = (\mathbf{e}, \mathbf{s}, \mathbf{z})$

FIGURE 8.4 – **Algorithme d'extraction de clé**

Le protocole de vérification est une variation du protocole de Stern ([92]), avec les clé intermédiaires $(\mathcal{K}_s, \mathcal{K}_p)$. Ici le protocole utilise quatre mises en gage, la dernière permet de vérifier la validité ou l'invalidité de la signature.

$\mathsf{Confirmation}_{\mathcal{P},\mathcal{V}}(m,\mathbf{z}) - \mathsf{Desaveu}_{\mathcal{P},\mathcal{V}}(m,\mathbf{z})$ **:**
Le prouveur possède un couple de clés $(\mathcal{K}_s, \mathcal{K}_p) = (\mathbf{e}, \mathbf{s})$. Le vérifieur veut vérifier la validité de la signature $\mathbf{z}$ Avec $\mathbf{H} = h(sd)$ et $\mathbf{M} = h'(m)$ où h' est une fonction de hachage telle que $h'(m) \in \mathbb{F}_2^{n\times t}$. Le protocole se décrit comme suit :

1. Le prouveur choisit une permutation aléatoirement π de $\{1,\ldots,n\}$ et un vecteur aléatoire $\mathbf{u} \in \mathbb{F}_2^n$, il envoie ensuite les mises en gages c_1, c_2, c_3, et c_4 * calculées comme suit :

$$C_1 = \mathsf{h}(\pi||\mathbf{H}\mathbf{u}^T), C_2 = \mathsf{h}(\pi(\mathbf{u})), C_3 = \mathsf{h}(\pi(s+\mathbf{u})) \text{ et } C_4 = \mathsf{h}(\pi||\mathbf{M}\mathbf{u}^T).$$

2. Le vérifieur envoie un challenge $c \in \{0,1,2\}$ aléatoirement au prouveur.
3. Il y a trois possibilités :
   - Si $c = 0$: alors il envoie π, $\mathbf{u}$.
   - Si $c = 1$: alors il envoie π, $\mathbf{e}+\mathbf{u}$.
   - Si $c = 2$: alors il envoie $\pi(\mathbf{e})$ et $\pi(\mathbf{u})$.
4. Trois possibilités
   - Si $c = 0$:, le vérifieur vérifie que C_1, C_2 et C_4 ont été construit de façon honnête.
   - Si $c = 1$: ($\mathsf{Confirmation}_{\mathcal{P},\mathcal{V}}(m,\mathbf{z})$), le vérifieur vérifie que C_1, C_3 ont été construit de façon honnête et que C_4 est valide.
   - Si $c = 1$: ($\mathsf{Desaveu}_{\mathcal{P},\mathcal{V}}(m,\mathbf{z})$), le vérifieur vérifie que C_1, C_3 ont été construit de façon honnête et $C_4 \neq \mathsf{h}(\pi||\mathbf{M}(\mathbf{e}+\mathbf{u})^T)$
   - Si $c = 2$:, le vérifieur vérifie que C_2, C_3 ont été construit de façon honnête et $\mathsf{wt}(\pi(\mathbf{e})) = w$.

Renvoie "Accepté" si toutes les vérifications sont correctes, sinon renvoie $\perp$.

FIGURE 8.5 – **Protocole de vérification**

8.5 Sécurité

Dans cette section, nous donnons les preuves de sécurité des propriétés suivantes : completness, soundness, zero-knowledge, inforgeabilité, invisbilité et impersonation. Dans cette partie, on prend comme paramètres pour le décodage par syndrome associé à un code $\mathcal{C}$ $[n, k = \frac{2n}{3}]$ avec un mot de petit poids w, la valeur de la borne de Gilbert-Varshamov pour le code $\mathcal{C}$. On introduit avant tout le lemme suivant :

Lemme 8.8. (Unicité de l'antécédent) Avec $\mathbf{H}$ une matrice aléatoire de taille $(n-k)\times n$, $\mathbf{x}$ un mot de poids w et $\mathbf{s} = \mathbf{H}\mathbf{x}^T$. La probabilité qu'il existe un autre mot $\mathbf{y} \neq \mathbf{x}$ de poids w tel que $\mathbf{H}\mathbf{y}^T = \mathbf{s}$ est majoré par $\frac{\Sigma_{i=0}^{2w}\binom{n}{i}}{2^{n-k}}$.

Démonstration. Supposons qu'il existe $\mathbf{y}$ de poids w qui ait le même syndrome que $\mathbf{x}$. Alors $(\mathbf{y}-\mathbf{x})$ est de syndrome nul et de poids au plus $2w$. Comme la probabilité pour un mot aléatoire d'avoir un

syndrome nul est de 2^{n-k} et que le nombre de mots de poids $2w$ est borné par $\Sigma_{i=0}^{2w}\binom{n}{i}$, nous avons bien la propriété recherchée. □

8.5.1 Completness

Théorème 8.9. Si le prouveur $\mathcal{P}$ et le vérifieur $\mathcal{V}$ exécutent de façon honnête les protocoles de vérifications. Alors pour tout message m, nous avons que

$$Pr[\mathsf{Confirmation}_{\mathcal{P},\mathcal{V}}(m, \mathsf{USign}(K_s^{\mathcal{P}}, m)) \text{ retourne Accept}] = 1.$$

Et pour tout couple message-signature invalide (m, σ), nous avons :

$$Pr[\mathsf{Desaveu}_{\mathcal{P},\mathcal{V}}(m, \sigma) \text{ retourne Accept}] = 1.$$

Démonstration. Pour la première partie, prenons (m, σ) un couple message-signature valide. Nous avons que $\sigma = (x, y, z)$ est valide, donc nous pouvons extraire la clé additionnelle $(\mathbf{e}, \mathbf{s})$ telle que $\mathbf{He^T} = \mathbf{s}$ et $\mathbf{Me^T} = \mathbf{z}$ où $h'(m) = \underline{\mathrm{M}}.$ avec $(\mathbf{e}, \mathbf{s})$, le prouveur peut effectuer le protocole de confirmation. La dernière partie à vérifier est que la vérification des valeurs des hachés est possible. Cette vérification est évidente en général et n'utilise que des propriétés comme $\mathbf{H}(\mathbf{e}+\mathbf{u})^\mathbf{T}$, $\mathbf{s} = \mathbf{Hu^T}$, $\mathbf{M}(\mathbf{e}+\mathbf{u})^\mathbf{T}$, $\mathbf{z} = \mathbf{Mu^T}$ ou $\sigma(\mathbf{e}) + \sigma(\mathbf{u}) = \sigma(\mathbf{e}+\mathbf{u})$. Le protocole de confirmation retourne toujours vrai dans ces conditions.
Pour la deuxième partie du théorème, prenons (m, σ) un couple messsage-signature invalide. On a vu que même pour un couple invalide, $\sigma = (x, y, z)$ est composé d'un couple valide $(x, y) = (x, \mathsf{Sign}(x, K_s^S))$. On peut donc utiliser l'extracteur de clé pour obtenir $(\mathbf{e}, \mathbf{s})$. Comme σ est invalide, on a $\mathbf{z} \neq \mathbf{Me^T}$, avec $\mathbf{M} = \mathbf{h'(m)}$. Cela implique que $\mathbf{M}(\mathbf{e}+\mathbf{u})^\mathbf{T}\ \mathbf{z} \neq \mathbf{Mu^T}$ et que la valeur de hachage C_4 ne peut pas être vérifiée. Les autres valeurs de hachages sont par contre, elles, vérifiées. Le protocole de désaveux est donc vérifié dans ces conditions. □

8.5.2 Soundness

Théoreme 8.5.1. Pour tout prouveur malhonnête $\mathcal{P}$ et tout couple message-signature invalide (m, σ), nous avons :

$$Pr[\mathsf{Confirmation}_{\mathcal{P},\mathcal{V}}(m, \sigma) \text{ retourne Accept}] \text{ est négligeable.}$$

Et pour tout couple message-signature (m, σ), nous avons :

$$Pr[\mathsf{Desaveu}_{\mathcal{P},\mathcal{V}}(m, \sigma) \text{ retourne Accept}] \text{ est négligeable.}$$

Démonstration. Prouvons le premier point du théorème avec (m, σ) un couple message-signature invalide. La signature σ est égale à $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Comme nous avons déjà vu, même si σ n'est pas valide, le couple $(\mathbf{x}, \mathbf{y})$ reste valide pour la signature Sign. Nous pouvons effectuer une extraction de la clé $(\mathbf{e}, \mathbf{s})$ où $\mathbf{He}^T = \mathbf{s}$ et $wt(\mathbf{e}) = w$. L'invalidité de σ et la validité de $(\mathbf{x}, \mathbf{y})$ implique que $\mathbf{z} \neq \mathbf{Me}^T$, avec $\mathbf{M} = h'(m)$.
Pour accepter le protocole de confirmation, le vérifieur à besoin de vérifier toutes les valeurs de hachage à la fin de l'interaction. On considère la situation où le prouveur peut répondre correctement aux 3 challenges possibles. Plus précisément, supposons que pour $c = 0$ le prouveur a envoyé une permutation ϕ (avec $\phi(x)$ la permutation du vecteur x) et α, la permutation ψ et le mot β pour $c = 1$ ou le mot γ et le mot δ pour $c = 3$. Ces valeurs vérifient la construction des valeurs de hachage, Cela implique que :

$$C_1 = h(\phi||\mathbf{H}\alpha^T) = h(\psi||\mathbf{H}\beta^T - \mathbf{s})$$

$$C_2 = h(\phi(\alpha)) = h(\delta)$$

$$C_3 = h(\psi(\beta)) = h(\gamma + \delta)$$

$$C_4 = h(\phi||\mathbf{M}\alpha^T) = h(\psi||\mathbf{M}\beta^T - \mathbf{z})$$

$$wt(\gamma) = w$$

Soit le prouveur peut faire une collision sur h soit nous avons les égalités :

$$\phi = \psi$$

$$\mathbf{H}\alpha^T = \mathbf{H}\beta^T - \mathbf{s}$$

$$\phi(\alpha) = \delta$$

$$\psi(\beta) = \gamma + \delta$$

$$\mathbf{M}\alpha^T = \mathbf{M}\beta^T - \mathbf{z}$$

Ces équations impliquent que $\mathbf{s} = \mathbf{H}\phi^{-1}(\gamma)^T$ et $\mathbf{z} = \mathbf{M}\phi^{-1}(\gamma)^T$ et $wt(\gamma) = w$. A partir du lemme 8.8 nous avons que $\phi^{-1}(\gamma) = \mathbf{e}$. L'équation suivante donne $\mathbf{z} = \mathbf{Me}^T$ et laisse apparaitre une contradiction avec l'invalidité de $\mathbf{z} \neq \mathbf{Me}^T$. Cela signifie que le prouveur n'est pas capable de répondre correctement aux 3 différents challenges en un tour. Le mieux qu'il puisse faire est de tricher avec une probabilité de 2/3. Avec un protocole à t-tours, cela donne une probabilité de triche pour la confirmation égale à $(2/3)^t$, qui devient négligeable lorsque t augmente.
La preuve du second point est proche du premier. Considérons (m, σ) un couple message-signature valide. Nous pouvons utiliser l'extracteur de clé pour obtenir $(\mathbf{e}, \mathbf{s})$ tel que $\mathbf{He}^T = \mathbf{s}$, $wt(\mathbf{e}) = w$ et $\mathbf{Me}^T = \mathbf{z}$ où $\sigma = (\mathbf{x}, \mathbf{y}, \mathbf{z})$. La dernière équation est due à la validité de σ. Pour le protocole de confirmation, le fait que le prouveur puisse anticiper les 3 différents challenges implique que $\mathbf{z} = \mathbf{Me}^T$. Pour le protocole de désaveux, il est facile de voir l'équation devient $\mathbf{z} \neq \mathbf{Me}^T$. C'est aussi une contradiction dans ce cas là. Cela veut dire que le prouveur ne peut pas anticiper les 3 challenges pendant le protocole et que la probabilité que l'algorithme retourne *Accept* est $(2/3)^t$, qui est négligeable en t. □

8.5.3 Zero-Knowledge

La preuve de zero-knowledge consiste à construire un simulateur qui peut recréer une interaction entre un prouveur et un vérifieur en temps polynômial. L'interaction obtenue doit être indistinguable d'une vraie interaction entre le prouveur et le vérifieur. Nous pouvons construire un simulateur pour ce protocole en utilisant celui de la preuve de zero-knowledge de l'authentification de Stern. L'interaction étant exactement la même, nous renvoyons à la preuve de l'article de Stern pour prouver cette propriété, dans [91].

8.5.4 Inforgeabilité

Dans cette partie, je prouve la propriété d'inforgeabilité de la signature. La première partie prouve la sécurité de la signature à usage unique et la deuxième partie prouve l'inforgeabilité de la signature complète.

Théoreme 8.5.2. Considérons les paramètres précédents associés à la signature à usage unique $[n, k = \frac{2n}{3}]$ pour le code $\mathcal{C}$ et w la valeur de la borne de Gilbert-Varshamov lié à un code aléatoire $[n, k = \frac{2n}{3}]$. Dans ce contexte, si un forgeur peut forger une signature unique en O opérations avec une probabilité λ alors il peut résoudre le problème de décodage par syndrome d'un code aléatoire $[n, \frac{n}{3}]$ avec un petit poids w en $n^3 + O$ opérations avec une probabilité $\frac{1}{2q_{\mathcal{RO}}}\lambda$ dans le modèle de l'oracle aléatoire, où $q_{\mathcal{RO}}$ correspond au nombre de requêtes à l'oracle aléatoire.

Démonstration. On considère une instance $\mathbf{Hx^T} = \mathbf{y}$ du problème de décodage par syndrome tel que $\mathbf{H}$ est une matrice $2/3n \times N$ et $\mathbf{x}$ est un petit mot de poids de poids w, avec w la valeur de la borne de Gilbert-Varshamov lié à un code aléatoire $[n, k = \frac{2n}{3}]$. Je prouve qu'un forgeur $\mathcal{F}$ qui peut forger une signature à usage unique (avec les paramètres décrits précédemment peut résoudre cette instance du problème de décodage par syndrome avec un facteur polynômial et une certaine probabilité. On décomposa l'instance $(\mathbf{H}, \mathbf{y})$ en $(\mathbf{H_1}, \mathbf{Y_1})$ et $(\mathbf{H_2}, \mathbf{y_2})$ avec $\mathbf{H} = \begin{pmatrix} \mathbf{H_1} & \mathbf{H_2} \end{pmatrix}$ et $\mathbf{y} = \begin{pmatrix} \mathbf{y_1} \\ \mathbf{y_2} \end{pmatrix}$. Le problème maintenant est de chercher $\mathbf{x}$ tel que $\mathbf{H_1 x^T} = \mathbf{Y_1}$ et $\mathbf{H_2 x^T} = \mathbf{y_2}$ avec $\mathbf{H_1}$ et $\mathbf{H_2}$ de taille $\frac{n}{3} \times n$. Dans ces problèmes, x doit être de poids w. La clé publique de la signature à usage unique correspond à $(\mathbf{H_1}, \mathbf{Y_1})$. Le forgeur reçoit la clé publique $(\mathbf{H_1}, \mathbf{Y_1})$. Ensuite, le forgeur a besoin d'interagir avec un oracle de confirmation ou de désaveu et un oracle de signature pour produire un résultat. Il a également accès à un oracle aléatoire $\mathcal{RO}$.
Lorsque $\mathcal{F}$ demande une signature, on lui donne $(m, \mathbf{y_2})$ (pour m un message aléatoire). Dans ce cas, comme la signature est à usage unique, le forgeur a seulement l'accès à au plus une signature. Le forgeur aurait pu demander la valeur de hachage d'un message m auparavant. C'est pourquoi, parmi les $q_{\mathcal{RO}}$ requêtes à l'oracle aléatoire, l'une est fixée pour avoir la valeur $\mathbf{H_2}$. La probabilité que la valeur de hachage $\mathbf{H_2}$ corresponde au message m est $\frac{1}{q_{\mathcal{RO}}}$, dans le cas où cette valeur ne correspond

pas, l'algorithme s'arrête.
Il est également possible pour $\mathcal{F}$ de demander une interaction avec le protocole de confirmation ou de désaveu. Il y a une difficulté ici car seul quelqu'un qui connaît la clé secrète peut simuler ces deux protocoles de façon interactive. Pour surmonter ce problème on modifie le forgeur $\mathcal{F}$: lorsque le forgeur exécute le protocole de confirmation, il reçoit à la fin une valeur de résultat égale à vrai ou faux. Nous construisons deux nouvelles machines de Turing PPT $\mathcal{F}_1$ et $\mathcal{F}_2$ telles que les deux machines répondent vrai quand $\mathcal{F}$ demande pour le couple (m, y_2), mais de telle sorte que pour toutes les autres requêtes $\mathcal{F}_1$ répond toujours vrai et $\mathcal{F}_2$ répond toujours faux. Notez que dans notre système, le protocole de confirmation et celui de désaveu sont presque les mêmes. Si la confirmation réussit cela signifie que le protocole de désaveu échoue et vice-versa. Ce n'est pas vrai en général, mais le fait que la construction de ces deux protocoles (confirmation et désaveu) soient presque identiques dans notre schéma, donne cette propriété. Cela signifie que si le protocole de confirmation réussit avec une probabilité négligeable, le protocole de désaveu réussit avec une probabilité non négligeable. Par conséquent, soit $\mathcal{F}_1$ soit $\mathcal{F}_2$ renvoie le même résultat que $\mathcal{F}$ avec bonne probabilité. On renomme maintenant $\mathcal{F}$ la PPTM qui est $\mathcal{F}_1$ avec probabilité $1/2$ et $\mathcal{F}_2$ dans l'autre cas. Ce nouveau forgeur est une machine de Turing polynômiale qui réussit avec une probabilité non négligeable et ne demande rien aux protocoles de confirmation et de désaveu.
Avec ces considérations, la machine $\mathcal{F}$ retourne un couple valable message signature (M, σ). Si nous désignons par $\mathbf{H_3}$ la valeur de hachage de M, nous avons une nouvelle instance du problème de décodage syndrome. Cette instance est de $(\mathbf{H_1}|\mathbf{H_2}|\mathbf{H_3}, \mathbf{Y_1}|\mathbf{y_2}|\sigma)$, ce cas a une probabilité non négligeable d'être résolu dans polynôme temps parce que la matrice $(\mathbf{H_1}|\mathbf{H_2}|\mathbf{H_3})$ a une probabilité non négligeable d'être inversible. La solution de ce problème est évidemment la solution au premier problème difficile $(\mathbf{H}, \mathbf{y})$, le coût de l'inversion de matrice i N^3. Voilà qui conclut la preuve.

□

Théoreme 8.5.3. Considérons une signature complète construite à partir d'une signature à usage unique avec des paramètres choisis comme dans le théorème précédent et d'une signature générique Sign. Dans ce contexte, si un forgeur peut forger la signature complète en O opérations et N accès à l'oracle de signature avec probabilité λ, soit il peut forger la signature Sign soit il peut résoudre une instance du problème de décodage par syndrome (comme dans le théorème précédent : pour un code $[n, n/3]$) avec w la valeur de la borne de GV d'un code aléatoire $[n, 2n/3]$ en $N(n^3 + O)$ opérations avec une probabilité $\frac{1}{2q_{\mathcal{RO}}^2}\lambda$ dans le modèle de l'oracle aléatoire , où $q_{\mathcal{RO}}$ correspond au nombre de requêtes à l'oracle aléatoire.

Démonstration. Cette preuve est très similaire à celle de la signature à usage unique. Ici, on n'utilise pas seulement une instance du problème de décodage par syndrome, on considère les N instances de ce problème, mais avec différentes matrices $\mathbf{H_i}$. Ces instances correspondent à différents couples $(\mathbf{H_1}, \mathbf{Y_1}), \ldots, (\mathbf{H_N}, \mathbf{y_n})$. Chaque matrice $\mathbf{H_i}$ est divisé en deux matrices $\mathbf{H_{i,1}}$ et $\mathbf{H_{i,2}}$ de tailles égales, ainsi que chaque $\mathbf{y_i}$ est divisé en $\mathbf{y_{i,1}}$ et $\mathbf{y_{i,2}}$.
Nous allons prouver qu'un forgeur $\mathcal{F}$ qui peut forger une signature complète en temps polynômial et avec une probabilité non négligeable de succès, peut résoudre une instance du problème de décodage par syndrome en temps polynômial et avec une probabilité non négligeable.

On commence la preuve de la même façon que pour le théorème précédent. On calcule une paire de clés (skpk) et on donne pk au forgeur. Le forgeur a besoin d'accéder à un oracle de signature , un oracle de confirmation, un oracle de désaveu et un oracle aléatoire. Cette fois, le forgeur peut demander un nombre polynômial de signatures.
L'oracle de signature est simulé avec le protocole de signature en utilisant comme clé supplémentaire une instance $\mathbf{H_{i,1}}, \mathbf{y_{i,1}}$ du problème difficile. Le forgeur aurait pu demander la valeur de hachage du message utilisé avant. C'est pourquoi, parmi les $q_{\mathcal{RO}}$ requêtes à l'oracle aléatoire, on en fixe une à $\mathbf{H_{j,2}}$.
Le problème avec le protocole de confirmation et celui de désaveu est résolu avec la même approche que dans la preuve de la signature à usage unique.
Le forgeur peut renvoyer un couple message-signature (m, σ) valide en temps polynômial et avec une probabilité non négligeable. La signature σ est un triplet $(x, \mathsf{Sign}(x, K_S^S), z)$. Ainsi, soit le forgeur peut forger la signature Sign soit il utilise un couple déjà signé (x, K_S^S) qui a donc déjà été utilisé. Si il utilise un couple déjà signé (x, K_S^S), qui a déjà été utilisé, cela signifie que le forgeur se trouve dans le cas de forgeage d'une signature à usage unique. Nous pouvons donc réduire la suite de la preuve en reprenant la preuve de la signature à usage unique. La différence étant que la probabilité que la valeur de hachage $\mathbf{H_{i,2}}$ corresponde au message m est $\frac{1}{q_{\mathcal{RO}}^2}$ dans le cas.
Finalement, nous avons construit une machine PPT qui peut résoudre une instance parmi les N du problème de décodage par syndrome. Le fait que ces instances soient indépendantes fait que le gain maximum que nous pouvons obtenir par rapport à la recherche d'une seule instance est presque N (voir aussi [86]). Avec N polynômiale en n, ce problème est ce réduit de façon polynômiale au problème de décodage par syndrome avec le même paramètre de sécurité que pour la signature à usage unique.

□

8.5.5 Invisibilité

La preuve suit la même approche que dans la preuve d'inforgeabilité. On prouve que le distingueur $\mathcal{D}$ peut résoudre le problème de décodage par syndrome décisionnelle. Le problème est donnée par une matrice $\mathbf{H}$ et un syndrome y. Nous voulons savoir s'il existe un mot de poids w dont le syndrome associé est $\mathbf{y}$ par la matrice $\mathbf{H}$. On divise $\mathbf{H}$ en $\begin{pmatrix}\mathbf{H_1}\\ \mathbf{H_2}\end{pmatrix}$ et $\mathbf{y}$ dans $\begin{pmatrix}\mathbf{y_1}\\ \mathbf{y_2}\end{pmatrix}$. On fait la même construction que pour les théorèmes précédents afin d'interagir avec le distingueur $\mathcal{D}$, en prenant $(\mathbf{H_2}, \mathbf{y_2})$ pour calculer la nouvelle signature. Le distingueur retourne vrai ou faux, ce qui correspond au fait que $(\mathbf{H}, \mathbf{y})$ a une solution ou non.

8.5.6 Impersonation

Théorème 8.10. La signature est sécurisée contre les attaques d'impersonation à message choisis ou nous pouvons attaquer le problème de décodage par syndrome en temps polynômial.

Démonstration. Considérons une instance du problème de décodage par syndrome $(\mathbf{H}, \mathbf{s})$ avec $\mathbf{H}$ une matrice $((n-k) \times n)$ et $\mathbf{s}$ son syndrome sur $\mathbb{F}_2$. Soit I une PPT qui gagne le jeu d'usurpation d'identité avec une probabilité non négligeable et avec comme entrée $\mathbf{H}$. On note (m, σ) le couple message-signature utilisé pour convaincre le vérifieur V et $(\mathbf{e}, \mathbf{He^T})$ le couple de secret à usage unique associée à la signature à usage unique dans σ. Comme on peut choisir les requêtes de signatures, on peut supposer que $\mathbf{He^T} = \mathbf{s}$. Les clés à usage unique dans les signatures sont indépendantes des clés à usage illimité. Cela implique que I n'a tiré aucunes informations sur $(\mathbf{e}, \mathbf{He^T})$ depuis les requêtes de signatures. De plus, les algorithmes de vérifications sont zero-knowledge, donc il n'a pas tiré d'informations sur $(\mathbf{e}, \mathbf{He^T})$ grâce à ceux-ci non plus. Si nous enlevons C_4 des protocoles de vérification, nous obtenons le même protocole, qui n'utilise plus la valeur de m. La machine I est toujours capable de passer ce protocole, cela veut dire que l'on peut extraire de I un algorithme polynômial qui peut faire fonctionner le protocole de vérification sans C_4 et sans information sur $(\mathbf{e}, \mathbf{He^T})$. Comme ce nouveau protocole de vérification correspond exactement au protocole de Stern [91], nous en déduisons de la preuve de soundness du protocole de Stern que I connait $\mathbf{e}$ avec une probabilité non-négligeable. La machine PPT I peut donc résoudre l'instance $(\mathbf{e}, \mathbf{He^T})$ en temps polynômial et avec une probabilité non-négligeable. □

8.6 Paramètres

Dans cette section, nous donnons quelques paramètres à utiliser avec la signature de différents niveaux de sécurité. pour plus de simplicité nous considérons le cas où $nk = n/3 = k'$ (la longueur de la signature). Le paramètre w - le poids de $\mathbf{x}$ - doit être choisi de sorte que x soit inférieure à la borne de Gilbert-Varshamov liée à un code aléatoire $[n, \frac{2n}{3}]$, et la longueur n du code doit être choisie de sorte qu'il résiste aux attaques du problème de décodage par syndrome pour un code $[n, \frac{n}{3}]$. Pour les meilleures attaques pratiques, nous suivons les bornes inférieures de [42]. Notez que les matrices $\mathbf{H}$ et $\mathbf{M}$ peuvent également être choisies quasi-cyclique, ce qui réduit la taille de la clé publique.

n	k	w	bits de sécurité
4500	3000	130	80
5100	3400	157	90
6000	4000	175	110
7500	5000	220	128

En utilisant la signature zero-knowledge de Stern en temps que signature numérique, on obtient une signature indéniable avec une taille d'environ 40kBytes.

8.7 Non-transférabilité

La non-transférabilité est une propriété que peuvent obtenir les signeurs indéniables. Elle correspond à ne pas pouvoir transmettre la preuve de vérification à un troisième partit.

Définition 8.11. (Non-Transférabilité) Soit (m, σ) un couple message-signature. Si quelqu'un obtient de l'information sur la validité ou l'invalidité de la signature avec bonne probabilité, alors il obtient de l'information sur l'une des clé privée qui a été utilisée pendant la vérification. La clé privée peut être celle du prouveur ou celle du vérifieur.

Cette propriété nécessite l'usage de clés privée et publique pour le vérifieur, pour le différencié d'une personne n'ayant pas accès à cette vérification. On peut noter de la même façon que la propriété de zero-knowledge n'implique pas la non-transférabilité car en conservant les graines servant à générer les challenges lors de la vérification interactive, on peut transformer la vérification interactive en vérification non-interactive, et ainsi transmettre la preuve de vérification.
Un travail futur sera de rendre la signature non-transférable en utilisant la signature d'anneau entre le vérifieur et le prouveur. De cette façon, l'interaction ne sera intelligible que par ces deux protagonistes, une tierce personne ne sachant pas si l'ensemble des communications n'est pas en fait réalisée par une seule personne.

Métrique Rang

Chapitre 9

Rappel sur la métrique rang

La métrique rang est une métrique sur les espaces vectoriels finis qui donne à chaque vecteur une valeur en fonction de ces coordonnées et des coordonnées de celle-ci dans un sous-espace vectoriel.

Dans la théorie des codes correcteurs, on utilise le plus souvent la métrique de Hamming pour simuler les erreurs que l'on souhaite corriger. En effet, on observe que si un bruit intervient lors d'une communication, il aura beaucoup de chance de ne perturber que très peu de coordonnées de l'information (codées par des bits). Dans ce cas là, la métrique de Hamming convient très bien.
La métrique rang, introduite en 1985 par Gabidulin, permet d'obtenir des codes correcteurs avec des propriétés différentes. Elle peut être utilisée lorsque les erreurs ont des coordonnées dans un même sous espace vectoriel (comme un mot constant), par exemple pour le stockage de données sur bande magnétique. En cryptographie, le problème de décodage par syndrome en métrique rang est plus difficile à résoudre avec les méthodes connues que ce même problème en métrique de Hamming pour des paramètres de même taille. Cela implique des tailles de clé plus petites pour des paramètres de sécurité du même ordre. Toutefois, les cryptosystèmes de théorie des codes ne se traduisent pas directement en métrique rang. Les codes en métrique rang connues ont une structure trop riche, ce qui fait apparaître des nouveaux problèmes de masquage et de nouvelles attaques. Le premier cryptosystème fut introduit par Gabidulin, Paramonov et Tretjakov en 1991 [47]. Il s'agit d'une version métrique rang du chiffrement de McEliece. Il y a eu ensuite différentes attaques et variantes sur ce cryptosystème. Ensuite est apparu le protocole d'authentification de Chen. Celui-ci est un protocole à divulgation nulle de connaissance qui n'utilise pas de fonction de hachage. Le chapitre 11 propose une nouvelle attaque générale sur le problème de décodage en métrique rang. Le chapitre 12 propose une attaque et une réparation sur le schéma d'authentification de Chen. Celle-ci permet de casser la plupart des paramètres des cryptosystèmes en métrique rang mais elle reste toutefois exponentielle. Le chapitre 13 présente une nouvelle signature en métrique rang.

9.1 Définition

Soit β une base $\beta_1, \ldots, \beta_m$ de $\mathbb{F}_{q^m}$ sur $\mathbb{F}_q$.

Définition 9.1. ([44]) Soit $x = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$. On appelle rang de x sur $\mathbb{F}_q$, le rang de la matrice $X = (x_{i,j})$, où $x_j = \sum_{i=1}^{m} x_{i,j}\beta_i$. On le note $Rg(x|\mathbb{F}_q)$, où plus simplement $Rg(x)$ lorsqu'il n'y a pas d'ambiguïté.

Cette définition ne dépend pas de la base β choisie précédemment. La métrique rang étant définie, la théorie des codes en métrique rang conserve les notation habituelles pour les codes. On notera donc les paramètres d'un code $\mathcal{C}$, $[n, k, d]$ avec n sa longueur, k sa dimension et d sa distance minimale. On définit la distance minimale comme pour les codes de Hamming.

Définition 9.2. La distance minimale d'un code est la plus petite distance entre deux mots du code.

Remarque 9.3. Pour un code linéaire, la distance minimale est égale à :

$$d = \min_{c \neq 0 \text{ et } c \in \mathcal{C}} (Rg(c))$$

Soient $\mathcal{S}(c, t)$ la sphère de centre c et de rayon t dans $(\mathbb{F}_q^m)^n$ et $\mathcal{B}(c, t)$ la boule de même centre c et de même rayon t. On a :

$$\mathcal{S}(c, t) = (\Pi_{j=0}^{t-1}(q^n - q^j))(\frac{\Pi_{j=0}^{t-1} q^m - q^j}{\Pi_{j=0}^{t-1} q^t - q^j})$$

$$\mathcal{B}(c, t) = \Sigma_{i=0}^{t} |\mathcal{S}(c, i)|$$

Remarque 9.4. On peut montrer que le volume de la sphère de rayon t est borné par :

$$q^{(m+n-2)t-t^2} \leq |\mathcal{S}(0, t)| \leq q^{(m+n+1)t-t^2}$$

On en déduit que :

$$|\mathcal{S}(0, t)| \leq |\mathcal{B}(0, t)| = \Sigma_{i=0}^{t} |\mathcal{S}(0, i)| \leq q^{(m+n+1)t-t^2+1}$$

9.2 Bornes sur la métrique rang

En métrique rang on retrouve les bornes de Singleton et Gilbert-Varshamov. Les formules sont toutefois différentes.

9.2.1 Borne de Singleton

Soit $\mathcal{C}$ un code de longueur n, de taille M et de distance minimale d sur $\mathbb{F}_q^m$. La borne de Singleton de $\mathcal{C}$ est :

$$M \leq q^{min(m(n-d+1),n(m-d+1))}$$

On peut trouver une démonstration dans [44].

Théorème 9.5. En métrique rang, il n'existe pas de code parfait. Aucun code ne vérifie :

$$m \times |\mathcal{B}(0, \lfloor (d-1)/2 \rfloor)| = q^{mn}$$

On peut trouver une preuve dans l'HDR de Pierre Loidreau [65].

9.2.2 Borne de Gilvert-Varshamov

En métrique rang il existe aussi une borne d'existence de code.

Théorème 9.6. Si $M \times |\mathcal{B}(0, d-1)| < q^{mn}$, alors il existe un code de longueur n, de taille $M+1$ et de distance minimale d sur $\mathbb{F}_q^m$.

On reprend la preuve de Pierre Loidreau dans [65].

Démonstration. On fait une preuve par récurrence sur M. Supposons que l'on construise un code de longueur n, de taille M et de distance minimale d appelé $\mathcal{C}$. Considérons alors l'ensemble formé par la réunion des boules de rayon $d-1$ entrées sur les mots de code, soit :

$$\mathcal{V} := \cup_{c \in \mathcal{C}} \mathcal{B}(c, d-1)$$

Alors $|\mathcal{V}| \leq M \times |\mathcal{B}(0, d-1)|$. Donc si $M \times |\mathcal{B}(0, d-1)| < q^{mn}$, il existe un vecteur$a \in (\mathbb{F}_q^m)^n$
$\mathcal{V}$ qui est à distance au moins d de tous led autres mots. On constate alors que $\mathcal{C} \cup \{a\}$ est un code de distance rang minimale au moins d et de cardinal $M+1$. □

9.2.3 Code Aléatoires

En métrique de Hamming, la distance minimale d'un code aléatoire vérifie la borne de Gilbert-Varshamov. En métrique rang, la répartition des mots selon leur rang est donnée par la formule ci-dessous :

$$\mathcal{A}(i) = \frac{M \times \Sigma_{j=0}^{i}(q^n - q^i)(\frac{\Sigma_{j=0}^{i-1} q^m - q^j}{\Sigma_{j=0}^{i-1} q^i - q^j})}{q^{mn}}$$

avec $\mathcal{A}(i)$ le nombre de mot de rang i dans $\mathcal{C}$.
On reprend la démonstration de l'HDR de Pierre Loidreau [65].

Démonstration. Pour prouver la proposition, on doit évaluer a probabilité qu'une matrice de taille $m \times n$ sur $\mathcal{F}_q$ soit de rang i, que l'on multiplie par le nombre de mots du code. Or le nombre de telles matrices est exactement $|\mathcal{S}(0, i)|$. □

9.3 q-polynômes

Les q-polynômes interviennent énormément dans la théorie des codes correcteurs en métrique rang. La raison est que les solutions d'un q-polynôme forment un espace vectoriel et qu'en métrique rang, le support d'un mot forme aussi un espace vectoriel. Ils serviront à donner une nouvelle formaulation du problème de décodage par syndrome dans le chapitre 11 qui sera plus rapide à résoudre que les précédentes formulations à l'aide de bases de Gröbner.

Définitions :

Définition 9.7. Un q-polynôme P de degré t sur $\mathbb{F}_q$ est un polynôme tel que :

$$P(z) = \Sigma_{i=0}^{t} p_i z^{q^i}$$

avec p_i dans $\mathbb{F}_q$ pour tout i entre 0 et t.

On peut remarquer qu'un q-polynôme est une combinaison linéaire des différents Fröbenius du corps de base. On en déduit la proposition suivante :

Proposition 9.8. Un q-polynôme sur $\mathbb{F}_{p^m}$, avec $q = p^m$, est une application linéaire sur $\mathbb{F}_p$.

On en déduit donc que l'ensemble des solutions d'un q-polynôme forme un espace vectoriel sur le corps de base.

Anneau des q-polynômes

Muni de la loi de composition des applications, l'ensemble des q-polynômes forme un anneau non commutatif. En effet, il est assez évident que pour P et Q des q-polynômes alors $P(z) + Q(z) = (P + Q)(z)$. La loi multiplicative provient de la composition :

Proposition 9.9. Soient P et Q des q-polynômes, $P \circ Q$ est un q-polynôme.

Démonstration. Soit P tel que $P(z) = \Sigma_{i=0}^{t} p_i z^{q^i}$ et Q tel que $Q(z) = \Sigma_{i=0}^{t} q_i z^{q^i}$. On a :

$$Q \circ P(z) = \Sigma_{j=0}^{t} (q_j (\Sigma_{i=0}^{t} p_i z^{q^i})^{q^j}$$

$$Q \circ P(z) = \Sigma_{j=0}^{t} \Sigma_{i=0}^{t} q_j {p_i}^{q^j} z^{q^{i+j}}$$

□

Dans cet anneau on peut effectuer une division euclidienne à droite ou à gauche, comme l'a prouvé Ore en 1933 dans [72].

Théoreme 9.3.1. Soient :

$$P(z) = p_k z^{q^k} + \Sigma_{i=0}^{k-1} p_i z^{q^i}$$
$$G(z) = g_k z^{q^k} + \Sigma_{i=0}^{s-1} g_i z^{q^i}$$

avec $p_k \neq 0$ et $g_s \neq 0$, alors il existe deux couples de q-polunômes (q_1, r_1) et (q_2, r_2) tels que :

$$P = q_1 \circ G + r_1 \text{ et } deg_q(r_1) < deg_q(G)$$

$$P = G \circ q_2 + r_2 \text{ et } deg_q(r_2) < deg_q(G)$$

Les couples (q_1, r_1) et (q_2, r_2) peuvent être déterminés en $s(k-s)$ multiplications dans $\mathcal{F}_q^m$.

Dans la suite nous utiliserons la proposition suivant :

Proposition 9.10. Soit $x = (x_1, \ldots, x_n)$ un vecteur de rang r sur $\mathcal{F}_q^m$, alors :
Le plus petit polynôme P tel que $P(x_i) = 0$, pour tout i inférieur à n, est de q-degré r.

Démonstration. Soit $P' = x^{q^m} - x$. Le polynôme P' est un q-polynôme qui annule tous les éléments de $\mathcal{F}_q^m$. Le polynôme P' annule entre autre tous les x_i. Le noyau de P' est de taille q^m, donc soit $r = m$ (et la démonstration est terminée) soit il existe un élément a qui qui n'appartient pas à $< x_1, \ldots, x_n >$ mais qui appartient au noyau de P'. On considère le polynôme $P'' = x^q - ax$. Les polynômes P'' et P' annulent tous les deux l'espace engendré par a, comme P'' annule seumement cet espace, P'' divise P' et P'/P'' est un polynôme de q-degré $m-1$ qui annule $< x_1, \ldots, x_n >$. En continuant le raisonnement on obtient un polynôme $P^(m-r)$ dont seul $< x_1, \ldots, x_n >$ est contenu dans le noyau. □

9.4 Code de Gabidulin

Les codes de Gabidulin ont été proposés dans [44] par E.M.Gabidulin en 1985. Ils ressemblent fortement aux codes de Reed-Solomons et forment l'une des seule famille de code que l'on sait décoder efficacement en métrique rang.

Construction :

Soit g un n-uplet $(g_1, \ldots, g_n)$ dans $\mathbb{F}_q^n$.

Définition 9.11. Le code de Gabidulin $Gab_k(g)$ est le code généré par la matrice :

$$\begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^q & g_2^q & \cdots & g_n^q \\ g_1^{q^2} & g_2^{q^2} & \cdots & g_n^{q^2} \\ \vdots & & \ddots & \vdots \\ g_1^{q^{k-1}} & g_2^{q^{k-1}} & \cdots & g_n^{q^{k-1}} \end{pmatrix}$$

Les codes de Gabidulin ont une distance $d = n - k + 1$, ce qui en fait des codes MDR, et sont capables de corriger $\lfloor \frac{n-k}{2} \rfloor$ erreurs.

On peut voir le code de Gabidulin $Gab_k(g)$ comme l'ensemble des évaluations de g par des q-polynômes de degré $k-1$.

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

Forme Systématique :

Proposition 9.12. Soit $Gab_k(g)$ un code de Gabidulin. Il existe P_1, ..., P_k des q-polynômes de q-degré inférieur à $k-1$ tels que $P_i(g_j^{q^i}) = \delta_{i,j}$ pour i et j entre 1 et k. La matrice génératrice de $Gab_k(g)$ sous forme systématique est la matrice $(P_i(g_j^{q^i}))_{(i,j)i}$ avec i entre 1 et k et j entre 1 et n.

Démonstration. Soit Q_i le q-polynôme unitaire de q-degré $k-1$ qui s'annule sur $\{g_1, \ldots, g_k\} \setminus \{g_i\}$. Le polynôme P_i tel que $P_i(z) = Q(g_i)^q - Q(z)(Q(g_i)^{q-1} - \frac{1}{Q(g_i)})$ est un q-polynôme qui vaut 0 sur $\{g_1, \ldots, g_k\} \setminus \{g_i\}$ et 1 sur g_i. Le fait que le code $Gab_k(g)$ puisse être vu comme un code d'évaluation de q-polynômes de q-degré inférieur à $k-1$ termine la démonstration. □

Matrice de Parité :

Le code dual d'un code de Gabidulin est encore un code de Gabidulin.

Décodage des Gabidulin :

On peut décoder les codes de Gabidulin à l'aide de l'algorithme d'Euclide à droite en $O(d^3 + dn)$.

Chapitre 10

Cryptographie en métrique rang

Dans ce chapitre nous présentons les particularités de la cryptographie en métrique rang, par rapport à la métrique de Hamming. Nous commençons par introduire le problème de décodage par syndrome en métrique rang. A l'image de la métrique de Hamming, c'est le problème sur lequel se réfèrent la plupart des preuves de sécurité. Ensuite nous présentons les meilleurs attaques sur ce problèmes. Puis nous parlons des bases de Gröbner qui nous servent, dans le chapitre 11, à résoudre des systèmes d'équations afin de décoder le plus efficacement possible un code aléatoire. Ensuite nous présentons les deux cryptosystèmes en métrique rang, le schéma de chiffrement GPT et le protocole d'authentification de Chen.

10.1 Problème de décodage par syndrome en métrique rang

Le problème de décodage par syndrome en métrique rang se traduit de la même façon qu'en métrique de Hamming :

Définition 10.1. (Problème de décodage par syndrome en métrique rang) Soit H une matrice $(n-k) \times k$, w un entier et s un syndrome de taille $n-k$. Le problème consiste à trouver un mot e de taille n tel que $He^T = s$.

Contrairement au problème de décodage en métrique de Hamming, le problème de décodage en métrique rang n'a pas été prouvé NP-difficile. Par contre, le problème très similair MinRank a lui été prouvé NP-difficile.

Les meilleurs attaques sur ce problème ont été introduites par A. Ourivski et T. Johannson dans [74] en 2002. Ces attaques sont des améliorations de l'attaque de Chabaud-Stern proposée dans [17] en 1996. Avec G une matrice génératrice d'un code de taille $k \times n$, x un mot de taille k sur GF(q^m), e un mot de taille n et de rang r sur GF(q^m), le problème consiste à retrouver e et x qui vérifient

$y = xG + e$.
Avec H la matrice $(n-k) \times n$ de parité du code, le problème peut se réécrire $Hy^T = He^T$. On peut passer d'une forme à l'autre, soit en multipliant par H, soit en inversant un ensemble d'information de H.

Mise en équation :

On considère $\beta = (\beta_1, \ldots, \beta_m)$ une base de $\mathrm{GF}(q^m)$. On peut écrire le mot e comme le produit entre une matrice P de taille $r \times n$ dans $\mathrm{GF}(q)$ et un vecteur u de taille et de rang r dans $\mathrm{GF}(q^m)$, $e = uP$. L'équation du décodage par syndrome devient :

$$Hy^T = H(uP)^T$$

Ce système est quadratique car les inconnues sont u et P. L'idée des attaques de Ourivski et Johannson correspond à faire une recherche exhaustive sur l'une des deux inconnue afin de rendre le système linéaire pour l'autre inconnue.

Recherche de la base u :
Cette attaque consiste à faire une recherche exhaustive sur les sous espaces vectoriels u de rang $r-1$ de $\mathrm{GF}(q^m)$ en essayant de résoudre le système linéaire associé $Hy^T = h(uP)^T$. On recherche des sous espaces de dimension $r-1$ au lieu de r car on recherche une erreur sous la forme $(1, e_2, \ldots, e_r)$. Cette forme est suffisante car le système linéaire à résoudre à la fin de chaque recherche est linéaire en α avec α dans $\mathrm{GF}(q)$. En multipliant α par $(1, e_2, \ldots, e_r)$ on obtient un r-uplet qui recouvre tous les sous espace vectoriel de dimension r possible. La formule qui donne le nombre de sous espace de taille $r-1$ dans $\mathrm{GF}(q^m)$ est :

$$\frac{\Sigma_{j=0}^{r-2} q^m - q^j}{\Sigma_{j=0}^{r-2} q^{r-1} - q^j}$$

On donne une borne supérieure de cette expression en majorant le numérateur par $q^{(r-1)m}$ et en minorant le dénominateur par $q^{(r-1)(r-1)}$, cela donne $q^{(r-1)(m-r)}$. Ourivski et Johannson donnent une complexité de $(k+r)^3 q^{(r-1)(m-r)+2}$.

Recherche de la matrice coordonnée P :
Cette attaque consiste à faire une recherche exhaustive sur la matrice coordonnée de l'erreur e en essayant de résoudre le système linéaire associé $Hy^T = h(uP)^T$. On sait que $e = y + xG$. L'erreur e appartient donc au code généré par G et y. Il s'agit d'un code de longueur n et de dimension $k+1$. Sa matrice génératrice sous forme systématique s'écrit $(Id|R)$ avec Id la matrice $(k+1) \times (k+1)$ identité et R une matrice calculée à partir de G et y. Comme e appartient à ce code, e est de la forme $(e', e'R)$ avec e' de taille $(k+1)$ et de rang r. On a vu précédemment que e était de la forme uP, on en déduit que e' est de la forme uP', avec P' une matrice $r \times (k+1)$ de rang r dans $\mathrm{GF}(q)$. Dans ce cas la recherche exhaustive porte sur la matrice P'. Il y a donc $q^{r(k+1)}$ éléments à chercher et un système linéaire à résoudre. Ourivski et Johannson donnent une complexité de $(k+r)^3 r^3 q^{(r-1)(k+1)}$.

Dans le chapitre 11, nous présenterons deux nouvelles attaques qui généralisent et améliorent les attaques de A. Ourivski et T. Johannson.

10.2 Bases de Gröbner

Les bases de Gröbner sont un outil très utile pour résoudre les systèmes polynômiaux à plusiuers indéterminées. Elles permettent entre autre de généraliser la division euclidienne aux anneaux $\mathcal{F}_q[X_1, \ldots, X_n]$. Nous les utilisons dans le chapitre 11 pour résoudre des systèmes à plusieurs indéterminées correspondant au problème de décodage par syndrome.
Comme pour le degré de la division euclidienne classique, les bases de Gröbner ont besoin d'un degré sur $\mathcal{F}_q[X_1, \ldots, X_n]$. Il existe différentes possibilités pour celui-ci, nous citerons les plus connues. Avec α et β des multi-degrés :

1. **Ordre lexicographique :**
   $X^\alpha <_{lex} X^\beta$ si le premier coefficient non nul de $\alpha - \beta$ en partant de la gauche est négatif.
2. **Ordre lexicographique gradué :**
   $X^\alpha <_{glex} X^\beta$ si $|\alpha| < |\beta|$ ou si ($|\alpha| = |\beta|$ et $X^\alpha <_{lex} X^\beta$)
3. **Ordre lexicographique gradué renversé :**
   $X^\alpha <_{grevlex} X^\beta$ si $|\alpha| < |\beta|$ ou si ($|\alpha| = |\beta|$ et le premier coefficient non nul en partant de la droite dans $\alpha - \beta$ est strictement positif).

Avec ces ordres, qui permet de définir un degré, on peut maintenant établir un algorithme de division de deux polynômes multivariés A et B. Il suffit de diviser le monôme de plus haut degré de A par le monôme de plus haut degré de B, lorsque c'est possible, pour obtenir le premier coefficient du quotient.
Cet algorithme qui marche bien pour la division d'un polynôme par un autre montre toutefois ses limites lorsque la division doit s'effectuer sur un famille de polynômes. Il peut arriver, en effet, que cette division ne renvoie pas un reste nul alors que le polynôme à diviser appartient à la famille en question.
Pour palier ce problème on définit des bonnes bases de division qui identifierons à coup sûr l'appartenance d'un polynôme à cette la famille engendrée par la base en renvoyant un reste nul à chaque fois que ce polynôme appartiendra à la famille engendrée par la base. On appelle cette base une base de Gröbner.

Algorithmes classiques de résolution des bases de Gröbner :

Ces bases ont été introduites par Buchberger en 1965 dans [12]. Il donne dans cet article le premier algorithme pour trouver de telles bases. Dans cette thèse, nous utiliserons l'algorithme F4, introduit par Faugère en 1999 dans [32] et implanté dans le logiciel magma dans [10].
L'algorithme F4 étant une amélioration de l'algorithme de Buchberger, nous présentons pour l'exemple le premier algorithme dans sa version la plus simple.

Algorithme de Buchberger :

Définition 10.2. Soient f un polynôme de $\mathcal{F}_q^m[X_1, \ldots, X_n]$ de coefficient dominant c_f et de terme

dominant t_f et g un polynôme de $\mathcal{F}_q^m[X_1, \ldots, X_n]$ de coefficient dominant c_g et de terme dominant t_g. Un S-polynôme est le polynôme $S(f, g)$ définit par :

$$S(f, g) = uf - vg$$

avec $u = \frac{t_g}{c_g}$ et $v = \frac{t_f}{c_f}$.

Entrées : $(f_1, \ldots, f_n) \subset \mathcal{F}_q^m[X_1, \ldots, X_n]$
Sortie : une base de Gröbner de $< f_1, \ldots, f_n >$
$G \leftarrow (f_1, \ldots, f_n)$
$CP \leftarrow \{S(f_i, f_j), 1 \leq i \leq j \leq k\}$
Tant que $CP \neq$ faire :
    Choisir $s \in CP$ et l'extraire de CP
    r prend la valeur du reste de la division de s par G
    Si $r \neq 0$ alors
        $CP \leftarrow CP \cup \{S(g, r), g \in G\}$
        $G \leftarrow G \cup \{r\}$
Retourner G

FIGURE 10.1 – Algorithme de Buchberger

10.3 GPT

Le cryptosystème GPT à été introduit par Gabidulin et al en 1991 dans [47]. Il s'agit d'un cryptosystème de McEliece qui utilise des codes en métrique rang. Ce cryptosystème donné plusieurs variantes et à subit plusieurs attaques structurelles. Dans cette partie nous montrons le cryptosystème original et ses variantes, nous expliquons aussi les différentes attaques que ces cryptosystèmes ont subi. L'attaque sur le problème de décodage par syndrome en métrique rang présentée dans le chapitre 11 permet de reparamétrer ces différentes versions de GPT.

10.3.1 Schéma d'origine :

Le schéma utilise les codes de Gabidulin. Soit $\mathcal{C}$ un code de Gabidulin de paramètre $[n, k, d]$ sur $\mathrm{GF}(q^m)$ et t un entier. Les clés sont construites sur le même principe que pour le schéma de McEliece en métrique de Hamming. La principale différence entre ces deux métrique est l'apparition d'une matrice S en métrique rang pour ajouter du masquage à la clé publique. En métrique rang, les codes de Gabidulin sont trop structurés pour n'être masqués que par des matrices inversibles. La clé privée est une matrice d'un code décodable et la clé publique correspond à cette même matrice masquée.

Génération des clés : Pour masquer la clé privée, nous avons besoin de trois matrices X, S et T. Avec X une matrice $k \times t$ de rang s sur GF(q^m) et de rang t sur GF(q), S une matrice $k \times k$ inversible sur GF(q^m) et T une matrice $n \times n$ inversible sur GF(q).

- Clé privée : (G, S, T). Une matrice G, de taille $k \times n$, génératrice de $\mathcal{C}$ ainsi que les matrices S et T servant à masquer G.
- Clé publique : (G', r), Avec $G' = S([X|0 + G)T$ et $r = \frac{n-k-t}{2}$.

Chiffrement : Pour chiffrer un mot x de taille k sur GF(q^m) il suffit de choisir un mot z de taille n et de rang r sur GF(q^m) puis de calculer le chiffré suivant :

$$y = xG' + z$$

Déchiffrement : Pour déchiffrer un chiffré y, il faut commencer par décoder le mot yT^{-1} à l'aide de l'algorithme de décodage du code de Gabidulin G. On obtient un mot x de taille n qui a été en partie masqué par X. Pour retrouver le message original, il faut multiplier à gauche par S les coordonnées de x entre t et n.

Paramètres :

On reprend le tableau donné dans la thèse de R.Overbeck. La fin du tableau est prit dans l'HDR de P.Loidreau.

m	k	t	s	Taille de la clé publique en bit	Sécurité
48	10	16	3	2 880	2^{134}
48	16	18	4	1 608	2^{124}
64	8	40	1	3 584	2^{87}

10.3.2 Versions améliorées :

Plusieurs versions ont été proposés pour améliorer la version d'origine de GPT. Elles consistent en générale à utiliser un autre code qu'un code de Gabidulin. Une premièer version utilise les codes RRC sur des Gabidulin et une autre version utilise des subfield subcodes de code de Gabidulin.

10.3.3 Attaques :

Une première attaque proposée par K.Gibson dans [53] permet d'attaquer le schéma d'origine de GPT. Il vient ensuite l'attaque de A.V.Ourivski dans [75] qui permet d'attaquer la version Niederreiter de GPT. Ensuite, R.Overbeck à montré des attaques dans [76]. On donne le tableau de paramètres

prit dans l'HDR de P.Loidreau.

m	k	t	s	Taille de la clé publique en bit	Sécurité
24	12	40	3	18 432	2^{83}
24	12	52	3	21 888	2^{83}

10.4 Chen

Le protocole de Chen est un protocole d'authentification à divulgation nulle de connaissance dont la sécurité est basée sur le problème de décodage par syndrome en métrique rang. Il a été introduit dans [25] en 1996 par K.Chen. Il fonctionne sur le même principe que le protocole de A.Shamir de 1989 dans [88]. Ces deux protocoles sont des protocoles à 5 tours dont les masques sont identiques et dont les vérifications consistent à faire les mêmes opérations. La nouveauté du protocole de Chen est le fait de ne plus utiliser de fonction de hachage lors des mises en gage. Cette amélioration est très utile puisque les fonctions de hachages constituent les opérations les plus coûteuses en temps (ou en nombre de calculs) pour les protocoles d'authentification basés sur des problèmes linéaires sous contraintes. Dans le protocole de Chen, les fonctions de hachage sont remplacées par une multiplication matricielle. Le fait de retrouver un antécédent n'est donc plus un problème difficile mais retrouver le bon antécédent est, *a priori*, infaisable sans plus d'informations.

Notations :

Soit V l'espace vectoriel à n dimensions sur GF(q^m). La sécurité du protocole se base sur le problème du décodage par syndrome en métrique rang. Avec H une matrice $(n-k) \times n$ et s, dans V, un mot de rang r. Il est difficile de retrouver s à partir de H, Hs^T et r. La clé privée du protocole est s et sa clé publique est Hs^T.

Schéma :

1. **Etape de mise en gage :**
   Le prouveur choisit $x \in V$ et $P \in \mathrm{GL}_n(\mathrm{GF}(q))$.
   Il envoie $c = HP^t x^t$ et $c' = Hx^t$.
2. **Première étape de challenge :**
   Le vérifieur envoie $\lambda \in \mathrm{GF}(q^m)$ choisit aléatoirement.
3. **Première étape de réponse :**
   Le prouveur calcule et envoie $w = x + \lambda s P^{-1}$.
4. **Seconde étape de challenge :**
   Le vérifieur envoie $b \in \{0, 1\}$ choisit aléatoirement.
5. **Seconde étape de réponse :**
   Le prouveur envoie P si $b = 0$ ou x si $b = 1$.
6. **Etape de vérification :**
   Si $b = 1$ et $\lambda \neq 0$, le vérifieur vérifie c' et si $\mathrm{rank}(w - x) = r$.
   Si $b = 1$ et $\lambda = 0$, le vérifieur vérifie c' et si $\mathrm{rank}(w - x) = 0$.
   Si $b = 0$, le vérifier vérifie que $HP^t w^t = c + \lambda i$.

FIGURE 10.2 – Protocole de Chen

Sécurité :

Ce protocole à fait l'objet de deux attaques. La première à été présenté par Chabaud et Stern en 1996 dans [17] et correspond à une attaque sur le problème de décodage par syndrome en métrique rang. Cette attaque est exponentielle et sont application au protocole de Chen a eu pour effet un recalibrage de ses paramètres. La deuxième attaque à été introduite dans le cadre de cette thèse dans et est décrite en détail dans le chapitre 11. Cette deuxième attaque est spécifique au protocole de Chen et profite du fait que le protocole n'utilise pas de fonction de hachage. Elle fonctionne en temps polynômial et a pour effet de rendre totalement inutilisable ce protocole. Dans le même chapitre, nous verrons une réparation de ce protocole utilisant des fonctions de hachage.

Chapitre 11

Sur la complexité du problème de décodage en métrique rang

Dans ce chapitre, nous proposons deux nouvelles attaques génériques sur le problème de décodage par syndrome en métrique rang. Soit C un code aléatoire $[n,k]$ sur $GF(q^m)$ et $y = x + e$ un mot reçu tel que $x \in C$ et $Rg(e) = r$. La première attaque, l'attaque de soutien, est combinatoire et permet de récupérer une erreur e de poids rang r en $min(O((n-k)^3m^3q^{r\lfloor\frac{km}{n}\rfloor}, O((n-k)^3m^3q^{(r-1)\lfloor\frac{(k+1)m}{n}\rfloor}))$ opérations sur $GF(q)$. Cette attaque améliore considérablement l'attaque précédente en introduisant la longueur n du code en l'exposant dans la complexité, ce qui n'était pas le cas dans les attaques génériques précédentes. La deuxième attaque est une attaque algébrique basée sur la théorie des q-polynômes introduites par Ore, nous proposons un nouveau cadre algébrique pour le problème SD en métrique rang qui permet d'envisager les équations et les inconnues dans l'extension $GF(q^m)$ plutôt que $GF(q)$ comme cela peut être le cas habituellement. Nous considérons deux approches pour résoudre le problème dans ce nouveau contexte . La première est une linéarisation et montre que si $n \geq (k+1)(r+1)-1$, le problème SD en métrique rang peut être résolu en temps polynômial. Plus généralement, nous prouvons que si $\lceil\frac{(r+1)(k+1)-(n+1)}{r}\rceil \leq k$, le problème peut être résolu avec une complexité moyenne en $O(r^3k^3q^{r\lceil\frac{(r+1)(k+1)-(n+1)}{r}\rceil})$. Nous considérons également la résolution avec des bases de Gröbner et nous donnons des argumentons sur sa complexité théorique, nous considérons également la résolution avec des bases de Gröbner sur des paramètres pratiques hybride. Comme exemple d'application que nous utilisons nos nouvelles attaques sur tous les récents paramètres des cryptosystèmes qui réparent le cryptosystème GPT, nous brisons tous les exemples de paramètres proposés, certains paramètres sont attaqués en moins de 1s.
Ce chapitre correspond à un article en cours de soumission à IEEE Information Theory, produit avec la collaboration de P.Gaborit et O.Ruatta.

11.1 Introduction

There exist several alternative problems to classical cryptography based on number theory : besides lattice based cryptography and multivariate cryptography, code-based cryptography has been recently the object of papers [42, 68, 3, 7] considering in details the practical complexity of the syndrome decoding problem for random codes for the Hamming metric. The rank metric for coding theory was introduced by Gabidulin in 1985 in [44] and he proposed a family of codes, the Gabidulin codes, analogous to Reed-Solomon codes in Hamming metric, which can be decoded in polynomial time. The Rank Syndrome Decoding (RSD) problem is the analagous for rank metric of the Syndrome Decoding problem for Hamming distance. Concerning cryptography, Gabidulin and *al.* proposed a few years later in [47] a cryptosystem (GPT) analogous to the McEliece cryptosystem but with Gabidulin codes for rank metric. One of the advantage of rank metric is that the complexity of the best known attacks for solving the RSD problem have an exponential complexity which is quadratic in the parameters of the system. For C a $[n, k]$ code over $GF(q^m)$ that one wants to decode for an error of rank r, the 1996 attack by Chabaud and Stern [17] has an exponential term in $q^{(m-r)(r-1)}$ and the 2003 attack by Ourivski and Johansson [73] has an exponential term in $q^{(k+1)(r-1)}$. It means that in practice very high security in 2^{80} can potentially be obtained with a public key of only a few thousands bits for the generic RSD problem, when for Hamming distance for instance, relying on the generic Syndrome Decoding (SD) problem means considering matrices of at least several hundred thousands bits. Because of the strong structure of Gabidulin codes, the GPT cryptosystem has been the object of several structural attacks over the years and several variations [46] for hiding the structure of the Gabidulin codes, like the Rank Reducible codes, have been proposed, with always public keys size of order 10.000 bits. Besides the GPT system, Faure and Loidreau [38] proposed a cryptosystem also relying on the Gabidulin codes but different from the GPT approach. At last public key zero-knowledge authentication schemes relying directly on random instances of RSD and with very small public keys have been proposed like [24] or very recently [51].

In 2005 Overbeck proposed a new structural attack [77] (see also the long version in J. of Crypto [78]), which permits to recover the structure of Gabidulin codes when hidden in different forms. His attack broke indeed all proposed parameters (at that time) of cryptosystems based on hiding the Gabidulin codes. A few years later, new parameters have been proposed [66, 82] which resist the attack by Overbeck.

Meanwhile besides the Overbeck attack which is a structural attack only related to Gabidulin codes, the complexity of the generic RSD problem has not evolved for more than 10 years. In particular when looking at the exponential complexity of [17] and [73], it is striking that the exponential term does not depend on the length the code. Besides these combinatorial attacks, an algebraic approach was also proposed in [31] but with limited results as soon as r was greater than 2 or 3, eventually the case $n = m$ is indirectly considered in [36]. Overall the RSD problem appears to be a cryptographic problem with a strong potential which seems to be under exploited.

11.2 background on rank metric, rank codes and algebraic systems

11.2.1 Notations and definition of rank distance codes

In this subsection, we introduce notions use hereafter and we define code for rank distance.

Let q be a power of a prime p, m and $n \in \mathbb{N}$ some integers and let V_n be a n dimensional vector space over $\mathrm{GF}(q^m)$. Set $\beta = \{\beta_1, \ldots, \beta_m\}$ be a basis of $\mathrm{GF}(q^m)$ over $\mathrm{GF}(q)$ in a way that for all $x \in \mathrm{GF}(q^m)$ we have $x = \sum_{i=1}^{m} x_i \beta_i$. If $\bar{v} = (v_1, \ldots, v_n) \in V_n$ then, for each $i \in \{1, \ldots, n\}$, we have $v_i = \sum_{j=1}^{m} v_{i,j}\beta_j$ and $\bar{v}$ can be interpreted as a matrix $(v_{i,j})_{i,j} \in M_{m\times n}(\mathrm{GF}(q))$. We define the rank weight of a element $\bar{v} \in V_n$ has the rank of the associated matrix $(v_{i,j})_{i,j}$ denoted $\mathrm{rank}(\bar{v})$. If $\bar{v}$ and $\bar{w} \in V_n$ we define $\mathrm{rd}(\bar{v}, \bar{w}) = \mathrm{rank}(\bar{v} - \bar{w})$. The function rd is a distance over V_n and is called the rank distance.

Définition 11.1. A rank code $\mathcal{C}$ of length n and dimension k over $\mathrm{GF}(q^m)$ is a subspace of dimension k of $\mathrm{GF}(q^m)^n$ embedded with the rank metric.

Définition 11.2. The minimum rank distance of the code $\mathcal{C}$ is the minimum rank (rank distance to zero) of a non-zero element of $\mathcal{C}$.

11.2.2 Rank distance and cryptography

In this section, we introduce the classical syndrome decoding problem and rank syndrome decoding problem together with some uses of these problems in cryptography.

Syndrome decoding problem (SD) Let $\mathcal{C}$ be a $[n, k]$ linear code over a finite field $\mathrm{GF}(q)$ of generator matrix G and parity check matrix H, then for each $y \in \mathrm{GF}(q)^n$, the set $y+\mathcal{C} = \{y+x \mid x \in \mathcal{C}\}$ is such that $H(y+\mathcal{C}) = \{Hy^t\}$. The value $H^t y$ is called the syndrome associated to y. The set $y + \mathcal{C}$ has at least a minimal weight element. We denote $\bar{y}^{\mathcal{C}}$ one of this minimal weight elements of $y + \mathcal{C}$. Clearly, $y - \bar{y}^{\mathcal{C}} \in \mathcal{C}$ and has minimal distance to y among the elements of $\mathcal{C}$. When there is only one minimal weight element in $y + \mathcal{C}$, we find the closest point to y in $\mathcal{C}$ and this gives us a method to decode y. This is the motivation to the following problem :

Problem Given $y \in \mathrm{GF}(q)^n$ and its syndrome Hy^t, find a minimal weight element in $y + \mathcal{C}$.

Decoding using this approach is a maximum likelihood decoding method. This approach is used

to attack the McElliece type crypto systems. The McEliece crypto system is a public key crypto system. Consider is a a $[n, k]$ linear code $\mathcal{C}_0$ for which we know to decode at distance t together with a generator matrix G_0 of this code. We generate an invertible $k \times k$ matrix S and an $n \times n$ permutation matrix to compute the matrix $G_{\text{pub}} = SG_0P$. The public key is the pair (G_{pub}, t) and the private key is the tuple (S, G_0, P, t). The encryption algorithm takes a word $\boldsymbol{m} \in \text{GF}(q)^k$ (the message text) and generates a word $\boldsymbol{e}$ in $\text{GF}(q)^n$ of weight t to compte the encrypted text $\boldsymbol{c} = \boldsymbol{m}G_{\text{pub}} + \boldsymbol{e}$. To decrypt, one compute cP^{-1} which is a word at distance at most t of the code $\mathcal{C}_0$. We use the error correction algorithm to recover $\boldsymbol{m}SG_0$. Then we recover $\boldsymbol{m}$ decoding and inverting S.

Cryptographers often prefer the Neiderreiter variant of this system. This variant consists in giving a parity check matrix H_0 of the code $\mathcal{C}_0$, to generate an invertible $(n-k) \times (n-k)$ matrix S and a $n \times n$ permutation matrix P. Then we compute the matrix $H_{\text{pub}} = SH_0P$ and the public key of the system is the pair (H_{pub}, t) and the public key is the tuple (H_0, S, P, t). The set of messages is set of all the points in $\text{GF}(q)^n$ of weight less or equal to t. We encode a message $\boldsymbol{m}$ computing $\boldsymbol{c} = H_{\text{pub}}\boldsymbol{m}^t$. To encode, we compute x such that $H_{\text{pub}}x^t$ and solve the associated syndrome problem to compute a word y. The message is $\boldsymbol{m} = x - y$.

Rank syndrome decoding (RSD) Here we describe a generalization of the syndrome decoding problem for a code embedded with rank metric. Let $\mathcal{C}$ be a rank metric code in $\text{GF}(q^m)^n$ of dimension k (over $\text{GF}(q^m)$) that can be effectively decoded at rank distance r and let H be a $((n-k) \times n)$ matrix with entries lying in $\text{GF}(q^m)$ such that $Hx^t = 0$ if and only if $x \in \mathcal{C}$.

Problem Given $y \in \text{GF}(q^m)^k$ and an integer r, find $s \in \text{GF}(q^m)^n$ such that $\text{rank}(s) = r$ and $Hs^t = y$.

This problem is not proven to be NP-hard, but its relation with the classical one with Hamming distance together with the fact that the best known algorithms to solve this problem have exponential complexity makes this problem difficult in practice. Furthermore, this problem is generally thought to be hard by the community.

The two best approaches to this problem (rank syndrome decoding) are :

- the approach by Chabaud and Stern (basis enumeration) which solves the problem with a $\mathcal{O}\left((nr+m)^3 q^{(m-r)(r-1)}\right)$ complexity (see [17]).
- the approach of Ourivski and Johannson which consists in two algorithms : one improving bases enumeration of [17] has a $\mathcal{O}\left((k+r)^3 q^{(m-r)(r-1)+2}\right)$ complexity and the second using a coordinates enumeration has a $\mathcal{O}\left((k+r)^3 r^3 q^{(m-1)(r-1)+2}\right)$ complexity (see [73]).

11.2.3 Polynomial solving

Some attacks proposed here consist to reduce the RSD problem to solving a polynomial system. Let us, now, introduce the problem of solving polynomial systems :

Problem : polynomial system solving (PoSSo) :
Input : let $\mathbb{K}$ be a field and $f_1(x_1, \cdots, x_u), \cdots, f_t(x_1, \cdots, x_u)$ polynomial over $\mathbb{K}[x_1, \cdots, x_u]$ where $\mathbb{K}$ is a field.
Goal : Find all $\mathbf{z} = (z_1, \cdots, z_u) \in \mathbb{K}^u$ such that $f_1(\mathbf{z}) = \cdots = f_n(\mathbf{z}) = 0$.

We will use two main methods to solve this problem : linearization (when we have enough equations) and Gröbner bases (this is a general approach). It is well known that PoSSo problem is NP-hard even if all the f_i have degree 2 (in this case the problem is called $\mathcal{MQ}$ for multivariate quadratic). Gröbner basis is a systematic tool to solve the PoSSo problem. When such a system has a finite number of solutions, it is said to be zero-dimensional. We will only consider zero-dimensional systems here since the roots coordinates are in a finite field and the field equations on each variable form a zero-dimensional system by itself.

11.3 Support attack

11.3.1 Background on information set decoding for Hamming distance

The best algorithms for decoding general random codes for Hamming distance is the information set decoding approach [42]. This method can be considered in two different ways. Consider for instance G a generator matrix of a $[n, k]$ (binary) code and let H a parity check matrix of G.

The first original approach starts from the received word $y = xG + e$ and consists in guessing a set of k coordinates of y with no error (an information set), once such a set is found with a probability $\frac{\binom{n-t}{k}}{\binom{n}{k}}$, a linear inversion of a $k \times k$ matrix permits to recover x. This is what is done in some sens for rank codes by Ourivski and Johansson in [73].

Another approach for Hamming distance consists in starting from the syndrome $H.y^t$ of length $n-k$ of the received vector. The basic idea of the decoding algorithm consists in guessing a set of $n-k$ coordinates which contains the support of the error e, it can be obtained with probability $\frac{\binom{n-t}{n-k-t}}{\binom{n}{n-k}}$. Then since one gets $n-k$ equations from the syndrome equations and a set of $n-k$ coordinates containing the error support, it is possible to recover the error e by a $(n-k) \times (n-k)$ matrix inversion

from the syndrome of the message.

It turns out that because of the properties of binomial coefficients, the two previous probabilities are equal and hence lead to the same exponential complexity for these two approaches (only in their simple form though - see recent improvements [42, 68, 3, 7]). Meanwhile one can remark that, although these attack are both considered as 'information set decoding', the second approach is not really connected with the notion of information set, but rather with the notion of error support.

We want to generalize the latter error support approach in the case of rank codes. We will see that at the difference of Hamming distance, for rank distance these two approaches lead to different exponential complexities and that the error support approach leads in general to a better complexity than the information set approach (corresponding to the Ourivski-Johansson approach).

11.3.2 General idea

Let C be a $[n, k]$ random code over $GF(q^m)$ with generator matrix G of size $k \times n$ and suppose one receives $y = c + e$ for $c \in C$ and $rank(e) = r$, in particular for $e = (e_1, \cdots, e_n)$ there exists a subspace E of dimension r which contains all the errors coordinates e_i. If one denotes by $(E_1, \cdots, E_r)$ a basis of E, one gets that : $\forall i, 1 \leq i \leq n$, there exist $e_{ij} \in GF(q)(1 \leq i \leq n, 1 \leq j \leq r)$ such that $e_i = \sum_{j=1}^{r} e_{ij} E_j$.

Let now H be a matrix of the dual code of C, then one gets

$$H.e^t = H.y^t. \qquad (1)$$

In a context of rank distance the notion of support corresponds to the notion of error space E since E contains all possible coordinates errors. Notice that for rank distance the support is a notion related to value of the coordinate errors e_i, when for Hamming distance the notion concerns a set of coordinates.

Now we want to guess a support E' which contains the support E ; an important point is the fact that for such a support E' it has to be possible to recover the error e by solving a linear system (as for Hamming distance). In the case of rank distance, we have the rank syndrome equations. There are $n-k$ equations over the extension field $GF(q^m)$ given by the rank syndrome, when writing these equations over the small field $GF(q)$ we get $(n-k)m$ equations on the small field. Now suppose we know E' of dimension r' which contains E, then each error coordinate e_i can be written as an element of E'. If we denote by $(E'_1, \cdots, E'_r)$ a basis of E' in $GF(q^m)$ over $GF(q)$, then there exist $e'_{ij} \in GF(q)$ such that :

$$\forall i, 1 \leq i \leq n, \qquad e_i = \sum_{j=1}^{r} e'_{ij} E'_j.$$

Since E' is fixed (and hence the E'_i), it gives $r'.n$ unknowns (the e'_{ij}) in $GF(q)$ and hence it is possible to recover the errors coordinates e_i by solving a linear system as long as $r'n \leq (n-k)m$.

11.3.3 Error support attack

If one also uses the fact that there is a rank code structure, the previous idea permits to prove the following proposition :

Proposition 11.3. Let C be a $[n,k]$ random code over $GF(q^m)$ with generator matrix G of size $k \times n$ and suppose one receives $y = c + e$ for $c \in C$ and $rank(e) = r$. Then one can recover c with an average complexity : $min(O((n-k)^3 m^3 q^{r\lfloor \frac{km}{n} \rfloor}, O((n-k)^3 m^3 q^{(r-1)\lfloor \frac{(k+1)m}{n} \rfloor}))$.

Proof.

Let C be a $[n,k]$ random code over $GF(q^m)$ with generator matrix G of size $k \times n$ and suppose one receives $y = c+e$ for $c \in C$ and $rank(e) = r$, in particular there exists a subspace E of dimension r which contains all the errors coordinates e_i. Let now H be a matrix of the dual code of C, then one gets $H.e^t = H.y^t$. Suppose now one knows a subspace E' of dimension r' which contains E, then for all $e_i (1 \leq i \leq n)$, if we denote by $E'_1, \cdots, E_{r'}$ a basis of E', there exist $e'_{ij} \in GF(q)$ such that :

$$e_i = \sum_{j=1}^{r} e'_{ij} E'_j.$$

Equation (1) gives $(n-k)m$ equations over the small field, the number of unknowns derived from the e'_{ij} is $r'n$. Hence it is possible to recover the e'_{ij} (and therefore the e_i) by solving a linear system, as long as

$$r'n \leq (n-k)m,$$

and hence :

$$r' \leq \lfloor \frac{(n-k)m}{n} \rfloor$$

(for $\lfloor a \rfloor$: floor of a (the integer part of a)).

Let now E' be a subspace of dimension r' over $GF(q)$ of $GF(q^m)$, supposing that everything is random the probability that E of dimension r is included in E' of dimension r' (for $r' \geq r$) is $q^{-(m-r')r}$. Indeed consider a basis of E, one gets $E \subset E'$ if and only any element of a given basis of E is included in E'. Since any vector of a basis of E has a probability $\frac{q^r}{q^m} = q^{-(m-r')}$ to be in E' (the

number of element of E' divided by the number element in $GF(q^m)$), the probability that $E \subset E'$ is therefore $q^{-(m-r')r}$.

Hence if one takes $r' = \lfloor \frac{(n-k)m}{n} \rfloor = \lfloor m - \frac{km}{n} \rfloor$ one gets a probability that E is included in a random space E' of dimension r', which is $q^{-(m-r')r} = q^{-r\lfloor \frac{km}{n} \rfloor}$. Hence if one also consider the complexity of the matrix inversion one gets the first proposed complexity of the proposition.

Now it is also possible to use the code structure and decrease the value of r by one, when increasing the value of k by one in the exponential coefficient, it is an interesting point since in practice, r is in general small and k is bigger than r.

The idea works as follows : one starts again, from the equation $y = xG + e$, we introduce a new $(k+1) \times n$ matrix G' obtained from G by adding a last row y. Now e belongs to the code C' generated by G', but more generally since C' is a code over $GF(q^m)$, for any $\alpha \in GF(q^m)$, the vector αe is also in G'. The idea now is to fix a special value of α which will fix an element of the searched error space, it will decrease by 1 the number of basis element which are to be included in E'. Then once the space αE is recovered, one recovers the α and the original E.

To go in more detail on this idea : we suppose without loss of generality that $e_1 \neq 0$, if one considers the subspace $e_1^{-1}E$ it has still dimension r but contains the vector 1. One can apply the same method that previously but this time the code has dimension $k+1$ and one knows an element of E. The number of syndrome equations over $GF(q)$ is $(n-k-1)m$. And hence the dimension r' of E' must satisfy : $r' \leq \lfloor \frac{(n-k-1)m}{n} \rfloor$. Since one knows that $1 \in E$, one just need that the remaining $r-1$ elements of a basis of E are also in E', which gives a probability $q^{-(r-1)\lfloor \frac{(k+1)m}{n} \rfloor}$. Once we recover $e_1^{-1}E$, taking e_1^{-1} as unknown in syndrome equations permits to recover it easily at almost no cost. Overall if one adds the polynomial complexity one gets the second complexity of the proposition.

□

Remark Comparison with previous attacks : In term of support, the basis enumeration attack corresponds to enumerate all possible supports of the error, it is the equivalent in Hamming distance, to enumerate all combination of error but with exact weight : the weight of the error. Such an approach does not take in account the fact that one knows $(n-k)$ linear equations in the extension field. Hence our attack can be seen as a combinatorial generalization of this point a view, in particular our attack is always better in term of exponential complexity. Our attack is also better in term of exponent complexity than [73] as soon as $n \geq m$, which is often the case in proposed parameters. Overall our attack can be seen as a generalization of the previous attacks [17].

Remark False solutions : There is the theoretical possibility that false solutions appear in the solving of the linear system, now since we consider the system as random, this case does not happen on the average. In practice with a strong probability we find only one solution to the system : the searched solution.

11.4 The q-polynomials and annulator polynomial setting

In this section, we introduce the algebraic material needed to present our new setting for algebraic cryptanalysis of rank syndrome decoding problems. We give a way to reduce a rank syndrome decoding to a polynomial system. In order to compare our approach to analogous formulation, we introduce the MinRank problem. This problem is very similar to rank syndrome decoding. We describe shortly already known cryptanalysis cryptanalysis of MinRank problem and how to transpose these attacks for rank syndrome decoding. This allows us to explain that our approach gives a better analysis of the rank syndrome decoding problem taking advantage of the underlying algebraic structure even if it does not give improvement for the MinRank problem.

11.4.1 Background on q-polynomials and annulator polynomials

The q-polynomials were first studied by Öre in [72] because of their deep links with finite fields structures and their surprising algebraic properties.

Définition 11.4. A q-polynomial of q-degree r in $GF(q^m)$ is a polynomial of the form :

$$P(x) = \sum_{i=0}^{r} p_i x^{q^i}, \text{ with } p_r \neq 0 \text{ and } p_i \in GF(q^m) \, \forall i \in \{0, \ldots, r\}.$$

One can remark that for all x and $y \in GF(q^m)$ and all α and $\beta \in GF(q)$, we have :

$$P(\alpha x + \beta y) = \alpha P(x) + \beta P(y)$$

since $x \to x^q$ is the Frobenius automorphism of $GF(q^m)/GF(q)$. This allows us to consider every q-polynomial over $GF(q^m)$ as a $GF(q)$-linear map taking $a \in GF(q^m) \mapsto P(a) \in GF(q^m)$.

We denote by $GF(q^m)[x^q]$ the set of q-polynomials. This set has a non-commutative ring structure with the following internal laws :

- Addition : $(P+Q)(x) = P(x) + Q(x)$,
- Composition : $(P \circ Q)(x) = P(Q(x))$.

In [72], Öre gives the main properties of this ring. This a non-commutative Euclidian ring (this implies that this is a principal ring).

Proposition 11.5. Let E be a subspace of $GF(q^m)$, then, $\text{Ann}(E) = \{P \in GF(q^m)[x^q] \mid P(a) = 0, \forall a \in E\}$ is an (both right and left) ideal of $GF(q^m)[x^q]$.

The above proposition show that $\text{Ann}(E)$ is a $GF(q)$-vector space. The following proposition give the main property of the of value where a q-polynomial vanishes.

Proposition 11.6. The roots of a q-polynomial of q-degree r over $GF(q^m)$ form a $GF(q)$-subvector space of $GF(q^m)$ of dimension at most r.

The two last proposition allow to easily establish the following proposition.

Proposition 11.7. [Öre] For any subspace E of $GF(q^m)$ of dimension r there exists a unique monic q-polynomial of q-degree r, such that :

$$\forall z \in E,\ P(z) = 0.$$

It is the polynomial $P(x)$ such that $\mathrm{Ann}(E) = (P)$ as a principal ideal.

This last proposition is the starting point of our cryptanalysis of the rank syndrome decoding problem. The existence of such a polynomial for the syndrome will give rise to a new kind of algebraic systems.

11.4.2 Annulator based setting for rank decoding

In this subsection, we give a new algebraic formulation of the syndrome rank decoding. Let C be a $[n,k]$ be a random rank metric code over $GF(q^m)$ with generator matrix $G = (g_{i,j})$ of size $k \times n$. We denote by G_i the i^{th} rows of G. Assume that one receives $y = m + e$ for $m \in C$ and $\mathrm{rank}(e) = r$. We denote E the vector space generated in $GF(q^m)$ by the coordinates of e. Usual algebraic approach may consider $(n+1)r + k$ unknowns to describe algebraically this problem : r unknowns in $GF(q^m)$ for to describe a basis of E, k unknowns in $GF(q^m)$ for the coordinates c_i of c and nk unknowns in $GF(q)$ for the coordinates of the error in the basis of E. In the next paragraph, we introduce a new approach that consider only $k + r$ unknowns, all in $GF(q^m)$.

Since $\dim(E) = r$, by proposition 11.7, there exists a unique monic q-polynomial P generating the annulator space of E of q-degree r. We have $c = \sum_{i=1}^{k} c_i G_i$, $y = (y_1, \ldots, y_n)$ and $e = (e_1, \ldots, e_n) \in E$, in a way that $P\left(y_j - \sum_{i=1}^{k} c_i g_{i,j}\right) = P(e_j) = 0$ for all $j \in \{1, \ldots, n\}$. This gives us a system of $k + r$ unknowns, the k coefficients c_i of c and the k coefficients of P, and n equations :

$$\left\{ \left(y_j - \sum_{i=1}^{k} c_i g_{i,j}\right)^{q^r} + \sum_{l=0}^{k-1} p_l \left(y_j - \sum_{i=1}^{k} c_i g_{i,j}\right)^{q^l}, \right.$$
$$\left. j \in \{1, \ldots, n\} \right. \tag{11.1}$$

This setting has less unknowns than the previous settings since it take into account that the variable are in $GF(q^m)$. The system is very structured and is linear on the coefficients of the q-polynomial P. Unfortunately, the monomials has shape $p_l c_i^{q^l}$ and so they have very high degree (less or equal to $q^r + 1$) but the system is very very sparse.

We are now interested in solving this algebraic system. In what follows, we will consider two ways to do so : linearization and solving using Gröbner basis. But, before to go further, we want to compare our setting to others arising in very closed problem.

11.4.3 MinRank and induced attacks

Buss et al. introduced MinRank problem in [13] and show that this is an NP-hard problem. This problem can be described as follows :

Problem Let $\mathbb{K}$ be a field, consider a positive integers N, n, k, r and matrices $M_0, M_1, \ldots, M_k \in \mathcal{M}_{N\times n}(\mathbb{K})$. Find $\lambda_1, \ldots, \lambda_k$ such that

$$\operatorname{rank}\left(\sum_{i=1}^{k}\lambda_i M_i - M_0\right) \leqslant r.$$

In [36], Levy-dit-Vehel and al. reduced the rank decoding problem to the MinRank problem :

Proposition 11.8. A rank decoding problem of parameters (m, n, k, r) can be reduced to MinRank problem of parameter (m, n, mk, r).

To do so, we consider an rank decoding problem associated to a $[n, k]$ rank metric code over $GF(q^m)$, G a generator matrix of C, $y \in GF(q^m)^n$ and $B = \{\beta_1, \ldots, \beta_m\}$ a basis of $GF(q^m)$ over $GF(q)$. We start with the problem of finding $m \in GF(q^m)^k$ such that $y - mG$ has rank $\leqslant r$ over $GF(q)$. We denote M_0 the $m \times n$ matrix over $GF(q)$ compute from y expressing in columns the coordinates of y in the basis B. Let G_i denote the i^{th} row of G, and $M_l = b_j G_i \in \mathcal{M}_{k\times n}(GF(q))$ where $l = (i-1)m + j$, for $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, m\}$. We have $m = (m_1, \ldots, m_k)$ and $m_i = m_{i,1}\beta_1 + \cdots + m_{i,m}\beta_m$, so we have $y - mG = y - \sum_{i=1}^{k} m_i G_i = y - \sum_{i=1}^{k}\sum_{j=1}^{m} m_{i,j} b_j G_i = y - \sum_{i=1}^{k}\sum_{j=1}^{m} m_{i,j} M_{(i-1)m+j}$. Finally, one get that if $y - mG$ has rank less or equal to r if and only if :

$$\operatorname{rank}\left(M_0 - \sum_{i=1}^{k}\sum_{j=1}^{m} m_{i,j} M_{(i-1)m+j}\right) \leqslant r. \tag{11.2}$$

Equation 11.2 proves that rank decoding of parameters (m, n, k, r) can be transformed in a MinRank problem of parameter (m, n, km, r) in polynomial time.

In [36], authors proposed a analysis of the Kipnis-Shamir attack for MinRank problem. Since the matrix $\sum_{i=1}^{k}\lambda_i M_i - M_0$ has rank at most r, its kernel E_λ has dimension at least $n - r$. It is to say that

their exist at least $n-r$ independent vector in E_λ. To describe $n-r$ independent vectors $v_1, \ldots, v_{n-r}$, fixing arbitrarily the first $(n-r)$ coordinates of each v_i one can have independent vectors in E_λ taking $v_i = (u_1, \ldots, u_{n-r}, v_{i,1}, \ldots, v_{i,r})$. Writing that :

$$\left(M_0 - \sum_{i=1}^{k} \lambda_i M_i\right) v_j = 0, j \in \{1, \ldots, n-r\},$$

we obtain a system of $n(n-r)$ equations in $r(n-r)+k$ unknowns. We denote by $\mathcal{I}_{\text{KS}}$ the ideal generated by these equations. Using the structure of this algebraic systems, the authors were able to give some bounds on the complexity of the attack in some *very* particular cases. First the authors always reduce the problem to the "square" case where $m = n$. The author consider the case where $s = n-r$ is fixed and parameters of the MinRank problem are $(n, n, s^2, n-s)$. With these parameters, they have ns equations with $rs + s^2$ equations and they estimate the complexity of solving MinRank using fast Gröbner basis to $O\left(\ln(q)\, n^{3s^2}\right)$.

Previously to [36] and still for the case $n = m$, Courtois and Goubin [27] gave an attack called kernel attack with complexity $O\left(q^{\lceil \frac{k}{n} \rceil r} k^3\right)$ with no constrain on $n-r$. Finally, in [31], the authors proposed an other approach also based on the Ourivski-Johannsson approach to solve directly the rank decoding problem. They rewrite the system in order to have only coefficients and unknowns lying in $GF(q)$. Their approach lead to a system of $m(2(n-k)-1)$ equations with $nr + m(r-1)$ unknowns. All these algebraic setting must be compared to the n equations with $k+r$ unknowns of our approach, despite the highest degree of the equations.

11.5 Solving by linearization

11.5.1 Basic approach

A basic approach consists in counting the number of different monomials in the c_i and the p_j and independent unknowns and the number of equations, in our setting, although the degree of equation is very high it turns out that the equations are also very sparse so that there a not so many different monomials, it is possible to obtain the following result :

Proposition 11.9. Let C be a $[n, k]$ random code over $GF(q^m)$ with generator matrix G of size $k \times n$ and suppose one receives $y = c + e$ for $c \in C$ and $rank(e) = r$. If $n \geq (r+1)(k+1) - 1$, the complexity of solving the rank decoding problem is polynomial in $((r+1)(k+1)-1)^3$ operations in $GF(q^m)$.

Proof.

We saw in previous section how the new setting could be described : Let $c = \sum_{i=1}^{k} c_i G_i$, $e = (e_1, \cdots, e_n)$ and $y = (y_1, \cdots, y_n)$. Since e has rank r, the subspace E generated by the e_i has

dimension r. By Proposition 7 there exists a unique monic annulator q-polynomial $P(x) = \sum_{i=0}^{r} p_i x^{q^i}$ with $p_r = 1$ such that $\forall z \in E, \qquad P(z) = 0$. Hence we obtain :

$$\forall j, 1 \leq j \leq n, \qquad P(y_j - \sum_{i=1}^{k} c_i g_{ij}) = P(e_j) = 0, \tag{2}$$

which gives n equations in the $k + r$ unknowns : $c_i (1 \leq i \leq k)$ and $p_j (0 \leq j \leq r - 1)$. Now the system we obtain is quadratic in the unknowns c_i and p_i. Such a non linear system can be solved through Gröbner basis, but it is also possible to solve by linearization, indeed in this case by linearization we obtain $(r + 1)(k + 1) - 1$ terms :

- $k.r$ terms of the form : $p_j c_i^{q^j}$ for $1 \leq i \leq k$ and $0 \leq j \leq r - 1$

- k terms of the form : $c_i^{q^r}$ for $1 \leq i \leq k$ (corresponding to the term $p_r = 1$).

- r terms of the form : p_j for $0 \leq j \leq p_{r-1}$ (corresponding to the scalar coordinates of y)

Hence overall $(r + 1)(k + 1) - 1$ linearized terms. In the case where the number of equations n satisfy $n \geq (r + 1)(k + 1) - 1$, the problem can hence be solved on the average by solving a linear system over $GF(q^m)$ with $(r + 1)(k + 1) - 1$ unknowns.

□

11.5.2 An hybrid advanced approach

We saw how it was possible depending on conditions on n, k and r to solve directly the problem, now what happens if such a condition is not fulfilled ? We saw that in the basic linearization of the previous section, that the number of unknowns was quadratic in r and k. It is possible to decrease this number by guessing an error. Suppose indeed that an error e_j is zero, recall that :

$$\forall j, 1 \leq j \leq n, \quad y_j = \sum_{i=1}^{k} c_i g_{ij} + e_j,$$

then if $e_j = 0$ one obtains a linear equation in the c_i, which permits to substitute one of the c_i by a linear combination of the others in all rows equations of the code.

In particular it means that if one can find an error $e_j = 0$, then one can decrease the number of c_i by one and hence decrease the number of unknowns in the linearization by $(r + 1)$ terms. Now since

the error $e_i \in E$ of dimension r, for random e_i the probability that $e_i = 0$ is q^{-r}.

This idea is precised in :

Proposition 11.10. Let C be a $[n, k]$ random code over $GF(q^m)$ with generator matrix G of size $k \times n$ and suppose one receives $y = c + e$ for $c \in C$ and $rank(e) = r$. If there exists an integer $t \leq k$ such that $n - t \geq (r + 1)(k + 1 - t) - 1$ then complexity of solving the rank decoding problem has an average complexity bounded above by $O((nkt + r^3k^3)q^{rt})$ operations in $GF(q^m)$.

Proof.

Our algebraic setting gives n equations in c_i and p_j. Suppose that we know that for a given equation, the error e_j is zero, then we obtain a linear equation $(\sum_{i=1}^{k} c_i g_{ij})$ with only unknowns the c_i. Suppose that c_1 (for instance) is written in terms of the others $c_j (c_j \neq 1)$, then substituting c_1 by a linear equation in the $c_j (c_j \neq 1)$, in all the n equations given by the relation $y = cG + e$ gives a new system of equations, with $n - 1$ linear equations without c_1 and one equation that is kept aside with c_1. Since the error rank is still the same, one knows that the annulator polynomial is still an annulator polynomial since the remaining errors e_j are the same. Hence one can still use equation (2) of the previous section, but this time the number of unknowns c_i has decreased by one. We hence obtain a new linearized system of equations with only $(r + 1)(k + 1) - 1 - (r + 1)$ terms. Now since we have used an equation to describe c_1 from the c_i we have one equation less (which contain terms with c_1) and hence $n - 1$ independent equations without c_1.

We hence saw how to it was possible to decrease the number of linearized terms when a zero error e_j was known. Now all rows of the code permits to derive linear equations :

$$\forall i, 1 \leq i \leq n, \qquad y_i = \sum_{j=1}^{k} c_j g_{ij} + e_i.$$

If one considers these n equations and consider new equations obtained by additions of multiplication of these equations by a random non-zero element of $GF(q)$, the obtained equations are still linear in c_j and since multiplication by an element of $GF(q)$ does not change the error support, the error obtained in the new equation can be considered as a random element of E. Therefore we can deduce that the probability to obtain a zero error in the linear combination of these equations is $\frac{1}{q^r}$ since there are only q^r possible errors.

Repeating the process t times permits each time to decrease the number of linearized terms by $r + 1$ and reduces the number of equations to be used (ie : without the substituted c_i) by 1, with a probability of success of $\frac{1}{q^{rt}}$. The complexity of the attack has a probability part in q^{rt} and a polynomial part. The polynomial part consists in searching new equations with error zero, this part is negligible since one can use a method where one modifies very few equations for each new trial. Once a potential zero is found, after finding i zeros, one has to write the c_j in terms

of c_l for $1 \leq l \leq j-1$, and then modify the terms of each c_l with the terms coming from c_k. After t trials the cost is hence $\sum_{i=1}^{t}(k-i)(n-i)$. Then the last part is the solving of a linear system in $GF(q^m)$ with $(r+1)(k+1-t)-1$ unknowns. The overall polynomial cost is hence $\sum_{i=1}^{t}(k-i)(n-i)+((r+1)(k+1-t)-1)^3$ operations in $GF(q^m)$. The first term can be bounded above by nkt and the second by $(r+1)^3(k+1)^3$, which gives the result.

□

Corollary (1)**.** Let C be a $[n,k]$ random code over $GF(q^m)$, suppose one receives $y = c + e$ for $c \in C$ and $rank(e) = r$. Then if $\lceil \frac{(r+1)(k+1)-(n+1)}{r} \rceil \leq k$, the error e can be recovered with complexity $O(r^3k^3q^{r\lceil \frac{(r+1)(k+1)-(n+1)}{r} \rceil})$.

Proof.

We apply the previous proposition : the condition $t \leq k$ such that $n - t \geq (r+1)(k+1-t)-1$ gives $t = \lceil \frac{(r+1)(k+1)-(n+1)}{r} \rceil$. In the complexity since in general for practical parameters $t << n$ we neglect the part in nkt. □

11.6 Solving with Gröbner basis

11.6.1 Solving polynomial systems : Gröbner basis approach

The notion of Gröbner basis is linked to the one of monomial term ordering. A monomial order is an order on monomials which is compatible with the product in order to have a pseudo-division with respect to such an order. Roughly speaking, a Gröbner basis $\mathcal{G} = \{g_1, \cdots, g_s\}$ of the ideal generated by a set of polynomials $f_1, \cdots, f_n$ is a family such that, for each $h \in \mathbb{K}[x_1, \cdots, x_u]$, then remainder of the pseudo-division of h with respect to $\mathcal{G}$ is 0 if and only if h lies in the ideal $(f_1, \cdots, f_n)$.

The lexicographical orders are particularly interesting since the shape of the Gröbner basis for such an order is the following :

$$\begin{aligned} g_u(x_u) \\ g_{u-1,1}(x_{u-1}, x_u), \cdots, g_{u-1,l_{u-1}}(x_{u-1}, x_u) \\ \vdots \\ g_{1,1}(x_1, \cdots, x_u), \cdots, g_{1,l_1}(x_1, \cdots, x_u) \end{aligned}$$

This structure allows to solve the original system (it is a type of triangular system). It was the initial motivation to the research of more efficient algorithms to compute Gröbner bases. Generally,

computing a Gröbner basis for a lexicographical order is harder than computing one for graded ordering. But one you know a Gröbner basis for a graded order you can use the FGLM algorithm (see [35]) to have one for lexicographical order or use solver that use directly the structure of the pseudo-division by Gröbner basis with respect to graded order.

The more efficient algorithm to compute Gröbner basis is the F_5 algorithm of Faugère [32], but experiments realized in this work were made using the F_4 algorithm in MAGMA [10]. Here, we give complexity results using the F_5 algorithm even if we used the F_4 algorithm since the use of F_5 algorithm has been carefully studied for cryptography. An important quantity for Gröbner basis computation of a ideal is the regularity of the generating system, denoted d_{reg}, defining the ideal. The number d_{reg} is the biggest degree reach in the Gröbner basis computation by the F_5 algorithm. In [33], the authors give a way to bound the complexity of the algorithm with respect to the regularity of the system :

Proposition 11.11. The complexity of computing Gröbner basis of a zero-dimensional system of t equations in u variables using the F_5 algorithm is :

$$\mathcal{O}\left(n * \binom{u + d_{reg} - 1}{d_{reg}}^{\omega}\right)$$

where d_{reg} is the degree of regularity of the system and $2 \leq \omega \leq 3$ is the linear algebra constant.

11.6.2 Gröbner bases for RSD

We will now use this technical background to study the original system, denoting $\mathbf{p} = (p_0, \cdots, p_{r-1})$ and $\mathbf{c} = (c_1, .., c_k)$:

$$\forall i \in \{1, \cdots, n\}, l_i(\mathbf{p}, \mathbf{c}) = \sum_{a=0}^{r} \left[\left(p_a y^{q^a}\right) - \left(\sum_{j=0}^{k} p_a c_j^{q^a} g_{i,j}^{q^a} \right) \right]. \tag{11.3}$$

Complexity issues of our method :

The use of Gröbner bases is very important when $n < (r+1)(k+1) - 1$, possibly combined with guessing of some variables for an hybrid approach (see below for a presentation of the hybrid approach). Here, instead of $(r+1)(k+1)$ variables of the linear attack, we have $r(k+1)$ variables since we can assume that the polynomial is unitary (p_r=1).

The system has a suitable structure (sparse or semi-sparse and algebraically structured). We will bound the complexity of the regularity for our system by a semi-regular one with a same degree sequence. If we denote by $M_d(u)$ the set of monomial of degree d in u variables, we have $\#M_d(u) =$

$\binom{u+d-1}{d}$. Following [32], the complexity to compute a Gröbner basis of an ideal of degree if regularity d_{reg} in a ring of polynomial of u variables with the F_5 algorithm is $\mathcal{O}\left(\left(\#M_{d_{reg}}(u)\right)^{\omega}\right)$.

Remark that all the equations have degree q^r+1 in a way that d_{reg} is the first non positive coefficient of $\frac{\left(1-z^{q^r+1}\right)^{rk}}{(1-z)^n}$. Since :

$$\frac{\left(1-z^{q^r+1}\right)}{(1-z)} = \sum_{i=0}^{q^r} z^i,$$

we have that d_{reg} is the first non positive coefficient of :

$$\left(\sum_{i=0}^{q^r} z^i\right)(1-z)^{kr-n}.$$

We obtain a complexity in $\mathcal{O}\left(n\binom{(k+1)r+d_{reg}}{d_{reg}}\right)$. We used the package described in [9] to compute the regularity of some generic problems with the same degree (and one can deduce a close formula for d_{reg} from the above computation). The sparseness of the system makes the theoretical complexity evaluation very pessimistic for practical achievement. For instance, for the case where $q = 2^{24}$, $k = 12$, $r = 6$ (i.e. equations have degree 2^6+1) and $n = 64$, we have a regularity of 200 and a complexity bounded above by 2^{152} ! But the running time to solve using F_4 in MAGMA is only few hours (and we can take advantage of the hybrid approach in order to improve the approach). It appears, experimentally, that the equations appearing in the computations are very very sparse and remain sparse. When the number of equations decreases, the algorithm destroys quickly the sparse structure. So, the theoretical bound has generally no meaning by itself, but it reveals some structural properties of the formulation. Experimentally, the running time of the algorithm behaves as if we replace the degrees of the equations by their q-degree (here the degree is q^r and the q-degree is r). In the previous example, instead of a complexity of 2^{152} with the degree, the complexity with q-degree is 2^{55} and the algorithm effectively runs within few hours. This remark is always valid in example as long as $n > r(k+1)$. This gives a range of parameters for which the Gröbner bases approach improves the linearization. Furthermore, the hybrid approach extends naturally this approach as we will see below.

Comparison with other approaches :

Other approaches, introduced in the context of cryptanalysis of systems based on MinRank problem can be extended to the RSD problem and reduce the considered problems to PoSSo problem just as we did in the previous paragraphs. We will show that our approach is of particular interest compare to those ones. The two methods has in common to get back to the linear algebra formulation and so, they work on the field $GF(q)$. We do not introduce here MinRank problem, we only adapt the attack to RSD. To do this, we use the reduction introduced in [36] to transform in poly-time a rank decoding problem to a MinRank problem. We also use the bounds given in [36] since, those

authors give finer result in [33] and [34], but algorithm in [33] works only for square matrices (which is generally not true in our cases) and in [34] the algorithm is probabilistic and the complexity is not improved drastically. Using the reduction of [36], we reduce a RSD problem on $GF(q^m)$ with parameters k for the dimension of the code, n for it size and r for the error rank to a MinRang problem of parameters m (number of rows of the matrices), n (number of column of the matrices), r (rank) and km (number of matrices). The method was develop only for square matrices, but it is possible to extend it to rectangular matrices. Using the theoretical bound of [33], the Kipnis-Shamir approach apply to a RSD problem of parameters n, k, r leads, when the generated MinRank problem is square, to an algorithm with complexity $\mathcal{O}\left(\binom{k*m+r(n-r)+d_{reg}}{d_{reg}}^{\omega}\right)$ (ω still denotes the linear algebra constant) and with $d_{reg} \leq 1 + min\{k * m, (n-r)r\}$). Here, the number of variables depend of n and m in contrary to our approach and is the bound seem also very pessimistic. Finally, the minors approach apply to a RSD problem of parameters n, k, r needs $\mathcal{O}\left(\binom{k*m+r(n-r)+1}{r(n-r)+1}\right)$ operations over $GF(q)$ with some more restrictive conditions. So, even if it is possible to give bounds for this approach, the number of variable highly depend on the dimension m of the field $GF(q^m)$ over $GF(q)$ and of the number of equations n in the exponent. Our approach, staying in $GF(q^m)$, avoid the parameter m in the combinatorial factor (it is on the constant for the complexity of the basic operations on $GF(q^m)$). Furthermore, for our approach, the number of equation n does not appear in the combinatorial factor, but only on the regularity making regularity decrease when n rises.

11.6.3 Hybrid approach

Just as for the advance linearization attack, it is possible to make an hybrid approach making some guess on the values of some variables c_i. Since the number of variables for the Gröbner bases approach is $(k+1) * r$ each time we find a c_i, we reduce the number of variables of r. Furthermore, we reduce the number of variable without decreasing the number of equation a lot. It is to say that making guesses on several variables improves the ratio of the number of equations over the number of variables. It is known that it is easier to compute an Gröbner basis of a very over-constrained non-coherent systems. We use this in order to define an heuristic : try to guess sufficiently many c_i to be able to check, in a fast way, that the generated system is not coherent. There is tradeoff between the number t of c_i we try to guess (it gives a q^t factor to the complexity and decrease the probability of success) and the speed of checking if the system is not coherent.

11.7 Cryptanalysis of some cryptosystems

In this section we apply our results in a cryptanalysis context. We first recall basic facts about the attack on the GPT cryptosysm then we show how our new attack can break all proposed parameters for the reparation of the scheme.

11.7.1 The GPT rank-based cryptosystem

The GPT cryptosystem is similar to the McELiece cryptosystem but works for rank distance. The Gabidulin codes are the equivalent of the Reed-Solomon codes for rank metric. The main problem in the cryptosystem consists in finding a way to hide the decoding matrix. For Hamming distance it is done through a permutation matrix. In the case of Gabidulin codes, several approaches have been proposed by adding words of small rank, by adding a scrambling matrix, introducing a new class of codes : the Rank reducible codes etc... All these systems lead to interesting parameters. There are two ways to attack such systems : a first way is structural and the attacker tries to recover the mask (or the hiding procedure) from the public key, based on the structural properties of the Gabidulin codes. In 2005 Overbeck [77] proposed a structural attack which broke many proposed parameters. After this attack some new parameters have been proposed which resist to this attack. We show in the following that these parameters are not secure either, meanwhile at the difference of Overbeck's attack, our attack is not structural but completely generic and depends only on code parameters.

11.7.2 Cryptanalysis of some proposed parameters in rank metric

In the following we apply our method on different reparation of GPT cryptosystem. Since the Basis enumeration and the Ourivski-Johansson attack were well known people proposed new variations which focused on resisting to Overbeck attack, since in general it was rather easy to resist to the Basis and Coordinate enumeration attacks.

Several approaches have been proposed to resist Overbeck's attack [45, 66, 82, 80, 81], but only two papers propose published parameters : an approach by Loidreau in [66] and an 'advanced standard approach' by Gabidulin, Rashwan and Honory ([82, 81]). In the following we show that all the proposed parameters in these papers are completely broken and can be practically recovered, even in less than 1s sometimes.

In the following we attack the RSD problem for $[n, k]$ codes for an error of rank r, $q = 2$ and an extension of size 2^m. In the following tables we give the different complexity of the different attack regarding the code used. Notice that our attacks are not structural attacks since we do not use any particular structure of the code. In the tables : 'OJ1' stands for the improved basis enumeration by Ouriski and Joahsson, 'OJ2' stands for coordinates enumeration, 'Over' stands for the complexity of the Overbeck attack, 'ES' stands for the complexity of the Error Support attack of Section 3, 'L' stands for the attack by linearization of section 5 (usually it does not work and hence we put ∞r, 'LH' stands for the complexity of the attack by hybrid linearization when guessing zero coordinates errors and 'HGb' is the complexity (usually in time) when one attacks with hybrid solving with Gröbner basis in our new setting. We did not put the complexity with simple Gröbner basis since it usually does not finish. All our computations were done with the F4 version of Magma on a double core of a 2GHz INTEL with 8 Go RAM.

We now consider the different type of reparations.

• **Loidreau reparation [66]**

The idea of the reparation is to add sufficiently many columns so that the Overbeck attack does not work. The author focus on the complexity of the Overbeck attack since there is no difficulty to resist other attacks since in previous complexity, the length n of the code did not appeared in the exponential complexity of the attack. The author starts from a $[24, 12, 12]$ Gabidulin code which can correct 6 errors and proposes two sets of parameters for which he adds 40 random columns or 52 random columns. The following table gives the different complexities for our attacks.

(n, k, r, m)	OJ1	OJ2	Over	ES	L	LH	HGb
$(64, 12, 6, 24)$	2^{104}	2^{85}	2^{80}	2^{50}	∞	2^{48}	2 hours
$(76, 12, 6, 24)$	2^{104}	2^{85}	2^{80}	2^{49}	∞	2^{36}	< 1 s

For hybrid Gröbner basis attack we added mixed multiplied columns (by a non zero element of $GF(q)$) and fixed 3 coordinates, hoping to have a zero error in this three coordinates. We then built our algebraic setting that we solved with Gröbner bases and the F4 algorithm. If the guessing was wrong a failure was obtained with F4 in an average of 0.13 s, repeating the process on a 4 processors computer permitted us to retrieve the solution in 2h. We also run in practice the LH attack which had a complexity in 2^{44} field operations, we run the attack in Magma and overall the HGb attack was far more efficient and faster than the attack with Gröbner bases. Our attacks show that all parameters sets proposed in [66] are completely broken, the second set of parameters which was supposed to be stronger than the first one, could in fact by attacked in less than a second with hybrid Gröbner bases attack.

• **Cryptanalysis of Gabidulin et al. reparations [82, 81]**

In [82] and [81], Gabidulin et al. propose an approach and parameters (claimed with security 2^{80}) to resist Overbeck's attack, the approach called 'advanced approach for standard variant' proceeds by hiding as usual the generator matrix G with a matrix M with a special form, overall the proposed parameters can be attacked directly by decoding an error e of rank r in a $[n, k]$ code over $GF(2^m)$. We give in the following table the different code parameters proposed and the complexity , the two first set of parameters are from [82] and the two last ones are from [81] corresponding to a public key of size 4000b.

(n, k, r, m)	Over	ES	L	LH	HGb
$(28, 14, 3, 28)$	2^{80}	2^{55}	∞	2^{49}	2 days
$(28, 14, 4, 28)$	2^{80}	2^{70}	∞	2^{65}	not finished
$(20, 10, 4, 20)$	2^{80}	2^{56}	∞	2^{51}	5 days
$(20, 12, 4, 20)$	2^{80}	2^{60}	∞	2^{60}	not finished

Experimental results show that it was possible to recover the message in 2 and 5 days with an hybrid Gröbner bases attack for the first and third set of parameters. In particular it shows that parameters proposed in [81] with a public key of 4000b are clearly unsafe. For the second and fourth case the computation could not finish with hybrid Gröbner bases meanwhile the hybrid linearization attack (without Gröbner bases) gives attack complexities of order 2^{60} which implies that these parameters can also be considered broken. Practical computation were done which shows that in practice the time estimation of the complexity followed these complexities.

11.8 Conclusion

In this chapter we give two new generic attacks for the RSD problem, the first one, the Support attack generalizes in a natural way to the rank metric, the classical information set decoding approach for Hamming distance. In particular this attacks permits to take account of the length of the code in the exponential term of the complexity, which was not the case before. The second attack, the polynomial annulator attack, is an algebraic attack which uses results on q-polynomials by Ore. The attack provides a new algebraic setting for attacking the RSD problem. This attack is specially efficient when the rank metric weight of the error is not too high, which is often the case in a rank metric context (compared to Hamming distance for instance). Overall our two new attacks break all published sets of parameters which repair the GPT cryptosystem after the Overbeck attack.

Overall the results we give here can be seen as a way to better understand the practical hardness of the RSD problem. This metric remains a very interesting tool for cryptography, in the way that all known attack have a quadratic complexity in the exponential term, which means that at the difference of Hamming distance, it is possible to obtain very low size public keys for a whole generator matrix, for suitable security parameters. In some sense there is an inherent complexity in the rank metric compared to the Hamming distance, indeed for evaluating the practical complexity of the syndrome decoding problem for Hamming distance, the basic tool is the Newton binomial coefficient which corresponds to counting the number of words of weight t for a length n vector, (value which bounded above by 2^n), when for rank metric the equivalent tool is the Gaussian binomial coefficient, which counts the number of subspace of dimension r in a vector space of dimension n, but this coefficient grows much faster than the Newton binomial coefficients, with an exponential term depending on the product of r and n.

In practice our results show that it should be doable to consider new parameters for the reparations of the GPT cryptosystems but with larger public keys. Another interesting recent development is the introduction of LRPC codes [48] for rank metric, these codes are an equivalent of LDPC codes for rank metric, and seems provide very low size of public with a rather low structure, not based on Gabidulin codes.

Chapitre 12

Cryptanalyse du protocole d'authentification de Chen

En 1995, K. Chen a proposé un protocole d'authentification zero-knowledge basé sur la métrique rang. Ce protocole est un protocole 5-passe avec une probabilité de triche de $\frac{1}{2}$. Il est construit dans l'esprit du protocole de Shamir et du protocole de Stern, mais il a la propriété supplémentaire de ne pas utiliser de fonction de hachage. Cette dernière caractéristique est très intéressante pour une perspective de cryptographie à faible coût. Dans ce chapitre, nous montrons tout d'abord que la preuve de connaissance zero-knowledge n'est pas correctement écrite et nous décrivons comment casser complètement le protocole en temps polynomial dans la taille des paramètres de deux manières différentes. Deuxièmement, nous proposons un moyen de le réparer et une preuve rigoureuse de zero-knowledge. La nouvelle preuve nécessite cependant l'utilisation d'une fonction de hachage. En particulier, nous améliorons considérablement les paramètres de la version réparée à l'aide une preuve plus raffiné.

Ce chapitre a donné lieu à un article [51] après un travail avec P.Gaborit et G.Zémor.

12.1 Introduction

Today, identification protocols are essential for the security of computer networks and smart cards. Zero-Knowledge is a useful tool to prove that a protocol can be reused without loss of security. Most zero-knowledge protocols are based on number theory. It is interesting to search for alternative zero-knowledge protocols, not based on number theory, for at least two reasons : first, number theory based protocols are often costly in term of computational complexity and second it is always interesting to have alternative protocols not based on number theory in the case a putative quantum computer may come to exist.

Over the years several protocols have been proposed relying on $\mathcal{NP}$-Complete problem and based on linear functions [87], [94], [95]. Although recent advances appeared like [54] to improve some of these protocols most of these protocols are still unsatisfactory because either of their key size and/or their communication cost, moreover they intrinsically rely on a hash function which can be seen as a drawback in the context of low cost cryptography. In this context K.Chen proposed in 1995 [24] an efficient zero-knowledge authentication protocol *without using a hash function* and based on the syndrome decoding problem for the rank metric.

The Chen protocol is in the spirit of Shamir PKP protocol, but the hard underlying problem is, as in the case of Stern's SD protocol, a syndrome decoding problem, but for the rank distance rather than Hamming distance.

The syndrome decoding problem for rank metric is less known and studied than for Hamming distance but it is believed to be hard by the community. In particular the complexity of the best known attacks is more than exponential in the size of the parameters, which permits to consider small parameters and hence obtain rather small size of keys. This latter aspect and the fact that Chen protocol does not use any hash function makes the protocol a very interesting alternative protocol to consider.

Over the years parameters proposed for the Chen protocol have already been attacked twice. First by Chabaud and Stern at AsiaCrypt'96 in [18] and later by Ourivski and Johansson in [73], meanwhile the attacks proposed in these two papers were not specific to the Chen protocol. In fact these attacks improved the general attack on the syndrome decoding problem for rank distance and as a by-product broke some of proposed parameters for the Chen protocol. In practice since the complexity of these new general attacks for the syndrome decoding problem for rank distance remain largely exponential, a small increase of parameters for the Chen protocol permits to resist these attack while preserving relatively small sizes of keys. Overall the Chen protocol was still unbroken in itself and was still attractive for applications.

12.2 Basic facts on rank distance

The rank distance was introduced in coding theory by Gabidulin in 1985 in [43]. Since the decoding problem is seemingly more difficult for the rank distance than for the Hamming distance, codes for the rank metric have been variously proposed as alternatives to ordinary error-correcting codes when they play a role in a cryptographic protocol. In the following we recall basic facts on this metric. We refer the reader to P. Loidreau's paper [64] for more details on the rank distance and its use in cryptography.

12.2.1 Definitions and notation

Notation :
Let q be a power of a prime p, m an integer and let V be a n dimensional vector space over the finite field $\mathrm{GF}(q^m)$. Let $\beta = (\beta_1, \ldots, \beta_m)$ be a basis of $\mathrm{GF}(q^m)$ over $\mathrm{GF}(q)$.
Let $\mathcal{F}_i$ be the map from $\mathrm{GF}(q^m)$ to $\mathrm{GF}(q)$ where $\mathcal{F}_i(x)$ is the i-th coordinate of x in the basis β.
To any $v = (v_1, \ldots, v_n)$ in V we associate the matrix $\overline{v} \in \mathcal{M}_{m,n}(\mathrm{GF}(q))$ in which $\overline{v}_{i,j} = \mathcal{F}_i(v_j)$.
The rank weight of a vector v can be defined as the rank of the associated matrix $\overline{v}$. If we name this value $\mathrm{rank}(v)$ we can have a distance between two vectors x, y using the formula $\mathrm{rd}(x, y) = \mathrm{rank}(x-y)$.

We now introduce an equivalence relation $\sim$ between two vectors x and y of V :

Definition (1)**.** For $x, y \in V$ we say that x and y are equivalent (denoted by $x \sim y$) if and only if their coordinates generate the same vector space.

We remark that this definition is independent of the basis and that it is possible to have $\mathrm{rank}(x) = \mathrm{rank}(y)$ without $x \sim y$.

We can see that $x \sim y$ is equivalent to the existence of a $n \times n$ invertible matrix P over $\mathrm{GF}(q)$ which satisfy $xP = y$.

12.2.2 Properties

The isometry group for the rank distance is computed in [4] and is composed of two families : right multiplication by an invertible matrix over $\mathrm{GF}(q)$ and multiplication of columns by a non-zero element of $\mathrm{GF}(q^m)$. With these two families it is not possible to transform a word with fixed rank to any word with same rank using also isometry. The fact that isometries may transform a word of given (rank or Hamming) weight to any other word with the same weight is very useful, it is for instance the case for the Hamming weight with permutation. We now introduce the product "$*$" in order to obtain a similar propriety with rank metric.

For a given basis β, we denote Φ_β the inverse of the function $V \to \mathcal{M}_{m,n}(\mathrm{GF}(q)) : x \to \overline{x}$ computed with the basis β.

Definition (2)**product.** Let Q be in $\mathcal{M}_{m,m}(\mathrm{GF}(q))$, $v \in V$ and β a basis. We define the product $Q * v$ such that $Q * v = \Phi_\beta(Q\overline{v})$, where $\overline{v}$ is constructed from the basis β.

The product has the important following property :

Proposition 12.1. For any $x \in V$ and $Q \in GL_m(q)$, $\mathrm{rank}(Q * x) = \mathrm{rank}(x)$.

This enables one to introduce a new class of functions f such that $\text{rank}(f(x)) = \text{rank}(x)$, supposing a basis β has been fixed. Notice that this type of function is not linear over $\text{GF}(q)$. Indeed, with $Q \in GL_m(q)$, $x \in V$ and $\lambda \in \text{GF}(q^m)$, we have $Q * (\lambda v) = Q\lambda Q^{-1} Q * v$ and any $\lambda \in \text{GF}(q^m)$ do not commute with all matrix in $GL_m(q)$.

The following properties of the product are useful.

Proposition 12.2. For any $x \in V$, $P \in \mathcal{M}_{n,n}(\text{GF}(q))$ and $Q \in \mathcal{M}_{m,m}(\text{GF}(q))$, we have $(Q * x)P = Q * (xP)$.

Proof (1)**.** It is clear that $(Q\overline{x})P = Q(\overline{x}P)$. To conclude we just have to notice that $\overline{x}P = \overline{xP}$.

Proposition 12.3. For any $x, y \in V$ and $\text{rank}(x) = \text{rank}(y)$, it is possible to find $P \in \mathcal{M}_{n,n}(\text{GF}(q))$ and $Q \in \mathcal{M}_{m,m}(\text{GF}(q))$ such that $x = Q * yP$.

Proof (2)**.** Let us denote by r the rank of x and y. Notice that r is then less or equal to m and n. To prove the property we just have to prove that there is $Q \in \mathcal{M}_{m,m}(\text{GF}(q))$ which satisfies $x \sim Q * y$. As we said previously, this is equivalent to the existence of a matrix P which satisfies $Q * yP = x$. Let $y_1, \ldots, y_n$ be the coordinates of y. We reorder them to obtain $y_{\sigma(1)}, \ldots, y_{\sigma(r)}, \ldots, y_{\sigma(n)}$ with $y_{\sigma(1)}, \ldots, y_{\sigma(r)}$ linearly independent. We extend the family $y_{\sigma(1)}, \ldots, y_{\sigma(r)}$ to make a basis γ. We do the same with $x_1, \ldots, x_n$ to make γ', an extended basis of the family (x_i). Let Q be the matrix which transforms γ into γ' ; $Q * y$ is now a vector with all its coordinates in γ' and moreover, all its coordinates are in $x_{\sigma'(1)}, \ldots, x_{\sigma'(r)}$.

12.2.3 Codes for rank distance

We refer to [64] for more details on codes for rank distance.

A code C of length n and dimension k over $\text{GF}(q^m)$ is a subspace of dimension k of $\text{GF}(q^m)$. The minimum rank distance of the code C is the minimum rank of non null vectors of the code.
It is known from [64] that these codes satisfy a Gilbert-Varshamov-like bound and that random codes attain this bound with a very high probability. In the following we will use this result to set up our parameters.

12.2.4 Rank distance and cryptography

The security of the Chen protocol is based on the rank version of a $\mathcal{NP}$-hard problem in Hamming distance, the syndrome decoding problem [5].

Syndrome decoding problem

Let H be a $((n-k) \times n)$ matrix over $\mathrm{GF}(q^m)$ with $k \leq n$, $i \in \mathrm{GF}(q^m)^k$ and ω an integer. The problem is to find s such that $wt(s) = \omega$ and $Hs^t = i$ where wt denotes the Hamming weight.

The problem is considered hard in general, especially when the matrix H is chosen at random. The best known algorithms for solving this problem are all exponential in ω, a recent survey on this complexity can be found in [42].

The previous problem can be naturally extended to the rank distance :

Syndrome decoding problem for the rank distance Let H be a $((n-k) \times n)$ matrix over $\mathrm{GF}(q^m)$ with $k \leq n$, $i \in \mathrm{GF}(q^m)^k$ and r an integer. The problem is to find s such that $\mathrm{rank}(s) = r$ and $Hs^t = i$.

In that case it is not proven that the problem is $\mathcal{NP}$-hard, but the relation with the Hamming case and the fact that the best known algorithms are all exponential makes this problem difficult in practice and the problem is generally believed to be hard.

There are two main approaches to this problem with rank matrix :

Chabaud and Stern proposed an algorithm to solve the problem in $O((nr+m)^3 q^{(m-r)(r-1)}$(see [18]) Ourivski and Johannson proposed two algorithms, the first uses a basis enumeration and is in $O((k+r)^3 q^{(m-r)(r-1)+2}$, the second uses a coordinate enumeration and is in $O((k+r)^3 q^{(m-r)(k+1)}$(see [73]).

12.3 The Chen protocol

The Chen identification protocol is a 5-pass zero-knowledge protocol with a cheating probability of 1/2. This protocol has the nice feature that it does not use any hash function, as in the original Fiat-Shamir protocol, but at the difference of other zero-knowledge protocols based on linear functions. Indeed, the PKP protocol and Stern's SD protocol both need a hash function. Chen's protocol works as follows. Let H be a random matrix over $\mathrm{GF}(q^m)$, s a word of low rank weight and $i = Hs^t$. If Alice knows the secret s then i is called her identity. The aim of the protocol is for Alice to prove that she knows s and for Bob to know whether Alice knows s. The protocol is split into six steps and uses H and the syndrome i as public key and the word s of rank r as private key.

The difficulty of the protocol relies on the difficulty of solving the syndrome decoding problem for the rank distance. In the original paper, Chen proved that to satisfy the soundness property one needs to choose the rank weight of the secret not more than $\frac{d}{3}$ where d is the minimum rank distance of the code whose parity-check matrix is H (d correspond to the Gilbert-Varshamov-like bound).

1. [First commitment step]
   The prover chooses $x \in V$ and $P \in \mathrm{GL}_n(\mathrm{GF}(q))$.
   He sends $c = HP^t x^t$ and $c' = Hx^t$.
2. [First challenge step]
   The verifier sends a random $\lambda \in \mathrm{GF}(q^m)$.
3. [Second commitment step]
   The prover computes $w = x + \lambda s P^{-1}$ and sends it.
4. [Second challenge step]
   The verifier sends at random $b \in \{0, 1\}$.
5. [Answer step]
   The prover sends P if $b = 0$ or x if $b = 1$.
6. [Verification step]
   If $b = 1$ and $\lambda \neq 0$, the verifier checks c' and if $\mathrm{rank}(w-x) = r$.
   If $b = 1$ and $\lambda = 0$, the verifier checks c' and if $\mathrm{rank}(w-x) = 0$.
   If $b = 0$, the verifier checks if $HP^t w^t = c + \lambda i$.

FIGURE 12.1 – Chen's protocol

Notice that the rank weight of the secret was increased to $\frac{d}{2}$ by Chabaud-Stern in [18]. Eventually we will see how it is possible to take it to d in section 12.6 with a new protocol.

12.4 Cryptanalysis

In this section we first explain why there are flaws in the zero-knowledge proof of the Chen protocol and then we propose two full cryptanalysis by passive attacks on the protocol which uses these flaws. For the attacks an attacker has just to have access to the public data exchanged during the protocol to retrieve the secret key of the protocol. The first attack relies on the fact that in the protocol, the masking of the secret at Step 3 by a right multiplication by an invertible matrix is not enough. This attack can be easily countered by modifying the masking. The second attack involves a recovery of the secret by linear algebra from leaking information when $b = 0$ and does not seem to be reparable without the introduction of hash functions.

12.4.1 Flaws in Chen's zero-knowledge proof

The two attacks rely on a flaw in Chen's proof of zero-knowledge. Proofs of zero-knowledge are generally made with a simulator of the scheme. The simulator proves that someone can create a scheme indistinguishable from a real execution in reasonable time without knowing the private key.

With this assumption, it is clear that the view of the execution of the scheme is not needed to attack it. The simulator given by Chen in his paper [24] is in fact false since the zero-knowledge proof omits the last sending of the scheme. Hence the incompletness of the simulator may imply attacks using a potential leak of information of the protocol. We present here two attacks based on using that leak of information. Indeed, we prove that Hx^t gives information about x and sP gives information about s with the two following attacks.

12.4.2 The Support Attack

In the protocol P is used to mask the secret s into a potential secret sP like it is done in PKP [87] or in SD [94], P taking the place of the permutation in these protocols. The fact that $s \sim sP^{-1}$ implies that the transformed secret is more dependent of s than lots of words with same rank. That makes this attack is more efficient than the general attack by Chabaud and Stern of [18]. Indeed the equivalence of s and sP^{-1} gives the basis researched since we read it directly from sP^{-1} and saves an enumeration of any basis. Let us see how to retrieve s from sP^{-1} and sH with the type of parameters proposed for the protocol : s a word with low rank, P an invertible matrix of $GL_n(\mathrm{GF}(q))$ and H a parity check matrix of a code $[n, \frac{n}{2}]$.

Description

Each time a prover wants to be identified with the protocol and his key s, he has to pass a sequence of challenges. The attack presented in this paragraph works when the challenge is the one represented by the value $b = 1$ in the description of the Chen protocol. This challenge occurs so many times that the attack can be executed in a passive way instead of an active way. We consider an execution of the protocol when the prover has to send the value x (case $b = 1$). In this case the value sP^{-1} can be computed with the formula $\lambda^{-1}(w - x)$. The attack uses the fact that there is a way to retrieve s from sP^{-1} with a very low complexity.

In the Basic Facts section we saw that $s \sim sP^{-1}$, so their coordinates generate the same vector space E over GF(q). The specific rank r of s implies that E is a vector space of dimension r. The coordinates of sP^{-1} generate the vector space E so we can construct a basis of E over GF(q) with them.

Let us denote $\gamma = (\gamma_1, \ldots, \gamma_r)$ this basis, with $\gamma \in \mathrm{GF}(q)^r$. We can express each coordinate s_j, j from 1 to n, of s into this basis and obtain $s_j = \sum_{k=1}^{r} a_{k,j}\gamma_k)$ with $a_{k,j} \in \mathrm{GF(q)}$ for k between 1 and r and j between 1 and n. We construct a basis of $\mathrm{GF}(q^m)$ over $\mathrm{GF}(q)$ such as its first r vectors are equal to $\gamma_1, \ldots, \gamma_r$. We call it $\gamma' = (\gamma_1, \ldots, \gamma_r, \gamma_{r+1}, \ldots, \gamma_m)$ with $\gamma_l \in \mathrm{GF}(q)$ for l from 1 to m. Writing the equation $Hs^t = i$ into $\mathrm{GF}(q)$ with the basis γ', permits to obtain $(n-k) \times m$ equations on $\mathrm{GF}(q)$ and $n \times r$ unknowns(the $a_{k,j}$). In the parameters proposed for the Chen protocol we always

have $n \times r \leq (n-k) \times m$, so the system is directly solvable by a Gaussian elimination and one can recover the secret s.

Attack's complexity

For this algorithm we have to generate a basis of cardinality r from n vectors, complete this basis and invert an $n \times r$ matrix over GF(q). The only cost we have to focus on is the matrix inversion, a O(N^3) algorithm, with $N = n \times r$ equal to 128 as proposed in [18], this attack cost about 2^{21} operations in GF(q). Our approach , which is specific to this protocol, permits to avoid the (exponential) search for a basis in which one can express the secret s. As soon as the previous inequality is satisfied, which is the case in all proposed parameters (the original Chen parameters and Chabaud and Stern parameters) the protocol can be broken in polynomial time at the cost of a Gaussian elimination for a matrix of size $n \times r$. We checked on examples that the attack worked in practice with this complexity.

We now describe our second attack.

12.4.3 Linear attack

Suppose a passive attacker can observe an exchange between a prover and a verifier with $b = 0$ in the protocol. In this case, the values $HiP^t x^t$ and Hx^t each correspond to a system of k equations in the n coordinates of x. In the Chen parameters, where n equal to $2k$, it is possible to retrieve x in only one iteration of the protocol. With the knowledge of P (in the $b = 0$ case) and x, the secret s can be deduced from $w = x + \lambda s P^{-1}$. In this paragraph we show how to deduce the secret s for any type of possible parameters. Indeed, each occurrence of the protocol for the $b = 0$ case has a chance to give new linear equations on the variable which composes the secret key s. More specifically, each time the prover sends a new P, (say P_j), there are as many new equations as the number of rows of HP_j^t independent from the rows of HP_k^t for $k \leq j$ and H.

Description

In order to prove his identity to a verifier, a prover runs the protocol several times. For each execution of the protocol, the prover sends two possible data (see step 5 of the description of the Chen protocol). The linear attack works when the prover has to send the variable named P (see description of the protocol). Since the protocol is based on a choice of two challenges, we can consider that it happens often. We call this challenge the challenge "$b = 0$". Let us denote by x, P, w, λ, c', the variables used in a challenge "$b = 0$" encountered, corresponding to the variables with same names

as in the description of the Chen protocol. Remind that for the challenge "$b = 0$" the variable x is unknown and the variables P, w, λ and c' are known. We will prove that other occurrences of the challenge "$b = 0$" gives linear equations in the coordinates of s.

The definition of $Hs^t = i$ gives numbers of equations in the coordinates of s but not enough to resolve a system. To add more equations to the system we use those given by the challenge "$b = 0$" :

$$w = x + \lambda s P^{-1}$$

$$c' = xH^t$$

We obtain :

$$Hw^t = c' + \lambda H(sP^{-1})^t$$

In the challenge "$b = 0$" the only unknown here is s, this gives new equations in the coordinates of s as far as these equations are linearly independent. Supposing that we have $n - 1$ equations in our system, witch is the worst case, the probability for a uniformly generated equation to be dependent of the system is equal to $\frac{1}{q^m}$. The fact that P is uniformly generated implies that HP^{-t} form uniformly generated equations. The probability to obtain linearly independent equation is hence very strong.

The attack is finished when the number of equations is enough to solve the system.

example[Active attack] The previous passive attack can easily be turned into an active attack. An attacker can adopt the following strategy : always anticipating 1 for challenges. Hence if the challenge, the attacker knows how to answer it correctly. Now if the challenge is '0' the previous attack permits to obtain information leaking which eventually gives the previous attack.

The fact that the Zero-Knowledge proof is false implies that this attack is possible.

Complexity of the attack and implementation

A square matrix in $\mathrm{GF}(q^m)$ has a great chance to be invertible, expecting that the cardinal of $\mathrm{GF}(q^m)$ is far from 2. The only cost we have to focus on is the matrix inversion which is in $\mathrm{O}(N^3)$ with a simple algorithm and can be decrease with a fast multiplication. With n about equal to 32 as proposed in [24], the complexity of the attack is about $W \times 2^5$ with W the cost of a multiplication in $\mathrm{GF}(q^m)$. An implementation of this attack using the mathematics software sage [90] finds 2^{21} secret s in 38013.98 seconds and is not able to directly inverse the matrix one time during those 2^{21} tests. It makes the attack very fast and coherent with the expected complexity.

12.5 Countermeasures

12.5.1 Defense against the support attack

The support attack uses the fact that $s \sim sP$. A simple repair is to multiply s by a matrix Q in $GL_m(q)$ using the product $*$ and a matrix P in $GL_n(\mathrm{GF}(q))$ instead of just using the matrix P. The fact that each vector can be turned into any vector with same rank implies that a support attack is not possible. The change affects also c' which has to be $c' = H(Q * xP)^t$, w becomes $x + Q^{-1}sP^{-1}$ and if $(b = 0)$ the prover has to send Q and P instead of P. We notice that the verification has no problem because we have $(Q * x)P = Q * (xP)$. Meanwhile this repair does not affect the linear attack.

12.5.2 Defense against the linear attack

The problem of the Chen protocol exploited in the linear attack is that Hx^t is a linear equation which simplifies other linear equations. A simple way to repair the linear attack is to replace $c' = Hx^t$ by a non linear equation. We use here $c = hash(x)$ where $hash$ is a hash function to be sure that no information can be obtained from c.

12.6 Repaired protocol

The new protocol is a 3-pass protocol with cheating probability equal to 1/2 and using hash functions. We still use H and i as public keys and s as the secret key.

12.6.1 Description of the protocol

The reparation proposed here is a rank metric version of the Stern protocol [94].
The notation $(a|b)$ correspond to the concatenation of a and b.
The notation $hash(a)$ is the hash value of a.

Verification step

Here we explain the verification step. There are three cases in the verification, $b = 0, 1, 2$.
In the case $b = 0$, V receives x and $(Q|P)$, he can compute $H(Q^{-1} * xP^{-1})^t$ and checks the hash

1. [Commitment step] The prover P chooses $x \in V$, $P \in GL_n(\mathrm{GF}(q))$ and $Q \in GL_m(q)$. He sends c_1, c_2, c_3 such that :

$$c_1 = hash(Q|P|H(Q^{-1} * xP^{-1})^t), c_2 = hash(x), c_3 = hash(x + Q * sP)$$

2. [Challenge step] V sends $b \in \{0, 1, 2\}$ to P.
3. [Answer step] there are three possibilities :
   - if $b = 0$, P reveals x and $(Q|P)$
   - if $b = 1$, P reveals $x + Q * sP$ and $(Q|P)$
   - if $b = 2$, P reveals x and $Q * sP$
4. [Verification step] there are three possibilities :
   - if $b = 0$, V checks c_1 and c_2.
   - if $b = 1$, V checks c_1 and c_3.
   - if $b = 2$, V checks c_2 and c_3 and that $rank(Q * sP) = r$.

FIGURE 12.2 – Repaired protocol

value $c_1 = hash(Q|P|H(Q^{-1} * xP^{-1})^t)$ and $c_2 = hash(x)$.
In the case $b = 1$, V receives $x + Q * sP$ and $(Q|P)$, he can compute $Q^{-1} * xP^{-1} + s$ and $H(Q^{-1} * xP^{-1})^t + i$ and then checks the hash value $c_1 = hash(Q|P|H(Q^{-1} * xP^{-1})^t)$. He can also compute the hash value $c_3 = hash(x + Q * sP)$.
In the case $b = 2$, V receives x and $(Q * sP)$, he can compute the hash values $c_2 = hash(x)$ and $c_3 = hash(x + Q * sP)$. He also checks that $rank(Q * sP) = r$.

12.6.2 Zero-knowledge property

Completeness

An honest prover who knows s will be able to construct c_1, c_2 and c_3. He will always be identified by a verifier because the verifications match with the data c_1, c_2 and c_3.

Soundness

The sketch of proof exposed here is not the same as in [24], because of the use of hash functions. In the Chen protocol the secret key s is taken of weight under $\frac{d}{3}$ ($\frac{d}{2}$ in Chabaud-Stern). The use of a hash function permit us to reach d (it dramatically improves the parameters which can be taken). If someone can answer in the three cases $b = 0, 1, 2$ for a chosen triple (c_1, c_2, c_3), then he knows :

– x_1 and (Q_1, P_1) such that :

$$c_1 = hash(Q_1|P_1|H(Q_1^{-1} * x_1 P_1^{-1})^t), c_2 = hash(x_1)$$

– w and $(Q_2|P_2)$ such that :

$$c_1 = hash(Q_2|P_2|H(Q_2^{-1} * wP_2^{-1})^t - i), c_3 = hash(w)$$

– x_2 and z such that :

$$c_2 = hash(x_2), c_3 = hash(x_2 + z), rank(z) = r$$

So either the attacker find a collision in the hash function either we have :

$$Q_1 = Q_2, P_1 = P_2, x_1 = x_2, w = x_2 + z$$

and

$$H(Q_1^{-1} * x_1 P_1^{-1})^t = H(Q_2^{-1} * wP_2^{-1})^t - i$$

We can deduce that

$$H(Q_1^{-1} * x_1 P_1^{-1})^t = H(Q_1^{-1} * (x_1 + z)P_1^{-1})^t - i$$

and then

$$H(Q_1^{-1} * zP_1^{-1})^t = i$$

with $rank(z) = r$.
So the attacker knows a solution of the difficult problem who describes the secret key s. He is able to build the secret s in polynomial time. The probability of success when the secret key s is not known is hencer, less or equal to $\frac{2}{3}$. The simulator built in the zero-knowledge proof shows that this probability is exactly $\frac{2}{3}$.

Security

The security of the secret key is based on the Syndrome decoding problem for the rank distance in the same way than for the Chen protocol.

Zero-Knowledge

The aim of this proof is to construct a simulator of the scheme. If we can simulate the scheme, we can use this construction to attack the secret key without any view of the scheme. This implies that the view of the scheme give no usable information to attack it more efficiently.
Let be B a verifier and S the simulator that we construct. The protocol can be simulated as follow :

1. S chooses $x \in V$ and $P \in \mathrm{GL}_n(\mathrm{GF}(q))$, $Q \in GL_m(q)$, z with $rank(z) = r$ and w a random value in V
   He also chooses $j \in \{0, 1, 2\}$.
   It sends c_1, c_2 and c_3 such that if $j = 0$:

$$c_1 = hash(Q|P|H(Q^{-1} * xP^{-1})^t), c_2 = hash(x), c_3 = hash(w)$$

   It sends c_1, c_2 and c_3 such that if $j = 1$:

$$c_1 = hash(Q|P|H(Q^{-1} * wP^{-1})^t - i), c_2 = hash(x), c_3 = hash(w)$$

   It sends c_1, c_2 and c_3 such that if $j = 2$:

$$c_1 = hash(w), c_2 = hash(x), c_3 = hash(x + z)$$

2. B chooses $b \in \{0, 1, 2\}$.
3. If $b = 0$, B sends x, Q and P.
   If $b = 1$, B sends w, Q and P.
   If $b = 2$, B sends x and z.
   If $b = j$ the execution is saved, otherwise the simulator restart this execution.

The simulation succeeds if it can save l executions of this scheme with $b = j$. This can be done in about $3l/2$ executions because the probability that $b = j$ is $\frac{2}{3}$. The simulator create a view indistinguishable from a real execution because of the use of the hash function, it was not the case in the Chen protocol. The value x, Q and P are clearly constructed with a uniform distribution. The value w is also constructed with a random distribution, witch is the same as the distribution given by $x + Q * sP$ in the real scheme. The last value z has the same distribution as the value $Q * sP$ because of the use of Q and P. Indeed, if Q and P are randomly chosen, $Q * sP$ can be any vector of rank r with a uniform probability. The value j permits to have the right distribution among the challenges values. All distributions are equal to those in the real scheme, it makes this simulator indistinguishable from a real execution. With this simulator anyone can simulate an execution of the scheme without knowing the secret key.

12.7 Parameters, improvements and comparison

12.7.1 Parameters

The soundness proof we proposed in the previous section enables us to take for the weight of the secret s a value just below the minimum distance of the code. If one considers random codes, it is known that with a good probability they lie on the Gilbert-Varshamov bound (cf [64] for the details on the exact formulae for these parameters). If we consider, $q = 2, n = 20, m = 20$ and $k = 11$ one

obtains a minimal distance of 7, hence we can take $r = 6$ for the rank weight of the secret. In that case the known attacks lead to a complexity of at least 2^{83} operations.

The fact that one can take a rank weight r close to d enables one to greatly decrease the size of the parameters.

Parameters of the repaired Chen protocol with parameters $q = 2, n = 20, m = 20, k = 11, r = 6$

Public matrix H : $(n-k) \times k \times m = 1980$bits

Public key i : $(n-k) \times m = 180$ bits

Secret key s : $n \times m = 400$ bits

Average number of bits exchanged in one round : 2 hash + one word of $\mathrm{GF}(q^m) \sim 800$bits.

12.7.2 Possible improvements

The protocol described in the previous section follows Stern's SD scheme, the only difference being that the notion of permutation for Hamming distance, which gives the indistinguishability is replaced by an equivalent notion for rank distance, the product $P * sQ$. Following this analogy other protocols like Veron's protocol ([96]) can be adapted in this case.

12.8 Conclusion

In this chapter we presented two polynomial attacks on the Chen identification protocol, the first attack takes advantage of the structure of the masking of the secret in the protocol and the second attack takes advantage of the fact that no hash function is used in the protocol. Overall our attacks completely break the system when previous cryptanalysis remained of exponential complexities. We propose a repaired protocol but with a hash function with a rigorous zero-knowledge proof. Besides the theoretical security of our repaired protocol, it also enables us to have better parameters in the soundness proof. Overall it permits a substantial decrease of the size of the parameters of the repaired protocol. The newly obtained protocol has a rigorous zero-knowledge proof and presents interesting features which can make it a good candidate for low-cost cryptography.

Chapitre 13

Une nouvelle signature efficace sur la métrique rang

Dans ce chapitre, nous proposons une nouvelle approche pour signer avec des codes. L'approche classique consiste à trouver l'unique préimage d'un syndrome à travers un algorithme de décodage. Ici, nous proposons d'introduire la notion de décodage mixte en ajoutant des effacements pour décoder les syndromes. Dans ce cas le problème devient difficile, comme pour la cryptographie basée sur les réseaux. Le document décrit un nouvel algorithme de signature pour la métrique rang basé sur un nouvel algorithme de décodage mixte pour les codes LRPC récemment introduits. Nous expliquons comment il est possible (en fonction des choix des paramètres) d'obtenir un algorithme de décodage complet qui est capable de trouver un antécédent de rang raisonnable pour tout syndrome aléatoire avec une très forte probabilité. Nous étudions la sécurité sémantique de notre algorithme de signature et nous montrons comment il est possible de réduire les attaques liées à l'inforgeabilité à des attaques directes sur la matrice publique. De cette façon, il n'y a pas de fuites d'information avec les signatures . Finalement, nous donnons plusieurs exemples de paramètres pour notre schéma, dont certains avec une clé publique de $5,760$ bits et la signature de $1,728$ bits. En outre, le système peut être très rapide pour le champ de base de petite taille.
Ce travail est en cours de soumission à Eurocrypt 2013 et à été réalisé en collaboration avec Philippe Gaborit, Olivier Ruatta et Gilles Zémor, [7].

13.1 Introduction

The problem of finding an efficient signature algorithm has been a major challenge for code-based cryptography since its introduction in 1978 by McEliece. Signing with error-correcting codes can be achieved in different ways : the CFS algorithm [28] cconsiders extreme parameters of Goppa codes to obtain a class of codes in which a non-negligeable part of random syndromes are invertible. This scheme has a very small signature size, however it is rather slow and the public key is very

large. Another possibility is to use the Fiat-Shamir heuristic to turn a zero-knowledge authentication scheme (like the Stern authentication scheme [91] into a signature scheme. This approach leads to very small public keys of a few hundred bits and is rather fast, but the signature size in itself is large (about 100,000b), so that overall no wholly satisfying scheme is known.

Besides classical code-based cryptography that relies on the Hamming distance, it is also possible to use another metric : the rank metric. This metric introduced in 1985 by Gabidulin [44] is very different from the Hamming distance. The rank metric has received in recent years very strong attention from the coding community because of its relevance to network coding. Moreover, this metric can also be used for cryptography. Indeed it is possible to construct rank-analogues of Reed-Solomon codes : the Gabidulin codes. Gabidulin codes inspired early cryptosystems, like the GPT cryposystem ([47]), but they turned out to be inherently vulnerable because of the very strong structure of the underlying codes. More recently, by considering an approach similar to NTRU [59](and also MDPC codes [71]) constructing a very efficient cryptosystem based on weakly structured rank codes was shown to be possible [48]. However, in terms of signatures based on the rank metric, only systems that use Fiat-Shamir are presently known [51].

An interesting point in code-based cryptography is that in general the security of the protocols relies on finding small weight vectors *below* the Gilbert-Varshamov bound (the typical minimum weight of a random code). This is noticeably different from lattice based cryptography for which it is very common for the security of a signature algorithm [55, 70] to rely on the capacity to approximate a random vector far beyond its closest lattice vector element (the Gap-CVP problem).

Traditionally, this approach was not developed for code-based cryptography since no decoding algorithm is known that decodes beyond the Gilbert-Varshamov bound : in fact this problem is somewhat marginal for the coding community since it implies many possibilities for decoding, while the emphasis is almost always to find the most probable codeword or a short list of most likely codewords.

13.2 Background on rank metric codes and cryptography

13.2.1 Definitions and notation

Notation : Let q be a power of a prime p, m an integer and let V_n be a n dimensional vector space over the finite field GF(q^m). Let $\beta = (\beta_1, \ldots, \beta_m)$ be a basis of $GF(q^m)$ over $GF(q)$.
Let $\mathcal{F}_i$ be the map from $GF(q^m)$ to $GF(q)$ where $\mathcal{F}_i(x)$ is the i-th coordinate of x in the basis β.
To any $v = (v_1, \ldots, v_n)$ in V_n we associate the matrix $\overline{v} \in \mathcal{M}_{m,n}(F_q)$ in which $\overline{v}_{i,j} = \mathcal{F}_i(v_j)$.
The rank weight of a vector v can be defined as the rank of the associated matrix $\overline{v}$. If we name this value rank(v) we can define a distance between two vectors x, y through the formula $\mathrm{d_r}(x, y) =$

rank$(x-y)$. We refer to [64] for more details on codes for the rank distance.

A rank code C of length n and dimension k over $GF(q^m)$ is a subspace of dimension k of $GF(q^m)$ viewed as a (rank) metric space. The minimum rank distance of the code C is the minimum rank of non-zero vectors of the code. In the following, C is a rank metric code of length n and dimension k over $GF(q^m)$. The matrix G denotes a $k \times n$ generator matrix of $\mathcal{C}$ and H one of its parity check matrix.

Definition (3)**.** Let $x = (x_1, x_2, \cdots, x_n) \in GF(q^m)^n$ be a vector of rank r. We denote E the $GF(q)$-sub vector space of $GF(q^m)$ generated by $x_1, x_2, \cdots, x_n$. The vector space E is called the **support** of x.

Remark (1)**.** The notion of support of a code word for the Hamming distance and for the the one introduced in definition 3 are different but they share a common principle : in both cases, suppose one is given a syndrome s and that there exists a low weight vector x such that $H.x^t = s$, then, if the support of x is known, it is possible to recover all the coordinates values of x by solving a linear system.

Definition (4)**.** Let e be an error vector of rank r and error support space E. We call **generalized erasure of dimension** t of the error e, a subspace T of dimension t of its error support E.

The notion of erasure for Hamming distance corresponds to knowing a particular position of the error vector (hence some partial information on the support), in the rank distance case, the support of the error being a subspace E, the equivalent notion of erasure (also denoted generalized erasure) is therefore the knowledge of a subspace T of the error support E.

13.2.2 Bounds for rank metric codes

The classical bounds for the Hamming metric have straightforward rank metric analogues, since two of them are of interest for the paper we recall them below.

Rank Gilbert-Varshamov bound

[GVR] The number of elements $S(m, q, t)$ of a sphere of radius t in $GF(q^m)^n$, is equal to the number of $m \times n$ q-ary matrices of rank t. For $t = 0$ $S_0 = 1$, for $t \geq 1$ we have (see [64]) :

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

From this we deduce the volume of a ball $B(n, m, q, t)$ of radius t in $GF(q^m)$ to be :

$$B(n, m, q, t) = \sum_{i=0}^{t} S(n, m, q, i)$$

In the linear case the Rank Gilbert-Varshamov bound $GVR(n, k, m, q)$ for a $[n, k]$ linear code over $GF(q^m)$ is then defined as the smallest integer t such that $B(n, m, q, t) \geq q^{m(n-k)}$.

The Gilbert-Varshamov bound for a rank code C with dual matrix H, corresponds to the smallest rank weight r for which, for any syndrome s, there exists on the average a word x of rank weight r such that $H.x^t = s$. To give an idea of the behaviour of this bound, it can be shown that, asymptotically in the case $m = n$ ([64]) : $\frac{GVR(n,k,m,q)}{n} \sim 1 - \sqrt{\frac{k}{n}}$.

Singleton bound

The classical Singleton bound for a linear $[n, k]$ rank code of minimum rank r over $GF(q^m)$ works in the same way as for linear codes (by finding an information set) and reads $r \leq 1 + n - k$: in the case when $n > m$ this bound can be rewritten as $r \leq 1 + \lfloor \frac{(n-k)m}{n} \rfloor$ [64]. Codes achieving this bound are called Maximum Rank Distance codes (MRD).

13.2.3 Cryptography and rank codes

The main use of rank codes in the cryptographic context is through the rank analogue of the classical syndrome decoding problem.

Maximum Likelihood - Rank Syndrome Decoding problem (ML-RSD) Let H be a $(n - k) \times n$ matrix over $GF(q^m)$ with $k \leq n$, $s \in GF(q^m)^{n-k}$. The problem is to find the smallest weight r such that $\text{rank}(x) = r$ and $Hx^t = s$.

In that case it is not proven that the problem is NP-hard, but this problem is very close to the syndrome decoding problem which is NP-hard, moreover the problem can be seen as a structured version of the MinRank problem which is also NP-hard (the RSD problem can be attacked as a MinRank problem but in practice the attack does not work since there are too many unknowns [36]). Moreover the problem has been studied for more than 20 years and the best algorithms are exponential, so that the problem is generally believed to be hard.

There exist several 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 (typically 30 and more), when q increases, the combinatorial aspect makes them less efficient. The first non-trivial attack on the problem was proposed by Chabaud and Stern [17] in 1996, then in 2002 Ourivski and Johannson [74] improved the previous attack and proposed a new attack, meanwhile these two attacks did not take account of the value of n in the exponent. Very recently the two previous attacks were generalized in [49] by Gaborit et *al.* in $(n-k)^3 m^3 q^{(r-1)\lfloor \frac{(k+1)m}{n} \rfloor})$ and take the value of n into account and were used to break some repaired versions of the GPT cryposystem.

- **algebraic attacks** : the particular nature of the rank metric makes it a natural field for algebraic attacks and solving by Groebner basis, since these attack 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 when q increases and when the parameters are not too high (say less than 30). There exist different types of algebraic equations settings : the first one by Levy and Perret [31] in 2006 considers a quadratic setting by taking as unknowns the support E of the error and the error coordinates regarding E, there is also the Kernel attack by [36] and the minor approach which consists in considering multivariate equations of degree $r+1$ obtained from minors of matrices [33], and more recently the annulator setting by Gaborit et *al.* in [49] (which is valid on certain type of parameters but may not be independent on m). In our context for some of the parameters considered in the end of the paper, the Levy-Perret attack is the most efficient one to consider. All the complexities for Grobner basis attacks are estimated through the very nice program of L. Bettale [9].

The case of more than one solution : approximating beyond the GVR bound

In code based cryptography there is usually only one solution to the syndrome problem (for instance for the McEliece scheme), now in this situation we are interested in the case when there are a large number of solutions. This case is reminiscent of lattice-based cryptography when one tries to approximate as much as possible a given syndrome by a word of weight as low as possible.

This motivates us to introduce a new problem which corresponds to finding a solution to the general decoding problem for the case when the weight of the word associated to the syndrome is greater than the GVR bound, in that case there may be several solutions, and hence the term decoding does not seem well chosen. Notice that in a lattice cryptography context, it corresponds to the case of Gap-CVP, which does not make sense here, since it implies a multiplicative gap.

Approximate - Rank Syndrome Decoding problem (App-RSD) Let H be a $(n-k) \times n$ matrix over $GF(q^m)$ with $k \leq n$, $s \in GF(q^m)^{n-k}$ and let r be an integer. The problem is to find a solution of rank r such that $\text{rank}(x) = r$ and $Hx^t = s$.

It is help to first consider the situation of a binary linear $[n,k]$ Hamming metric code. Given a random element of length $n-k$ of the syndrome space, we know that with high probability there exists word that has this particular syndrome and whose weight is on the GV bound. This word is usually hard to find, however. Now what is the lowest minimum weight for which it is easy to find

such a word ? A simple approach consists in taking $n-k$ random column of the parity-check matrix (a potential support of the solution word) and inverting the associated matrix, multiplying by the syndrome gives us a solution of weight $(n-k)/2$ on average. In fact it is difficult to do better than this without a super-polynomial increase in complexity.

Now for the rank metric, one can apply the same approach : suppose one starts from a random $[n,k]$ code over $GF(q^m)$ and that one searches for a word of small rank weight r with a given syndrome. One fixes (as in the Hamming case) a potential support for the word - here a subspace of dimension r of $GF(q^m)$- and one tries to find a solution. Let $x=(x_1,\cdots,x_n)$ be a solution vector, so that $H.x^t=s$. If we consider the syndrome equations induced in the small field $GF(q)$, there are $n.r$ unknowns and $m.(n-k)$ equations. Hence it is possible (with a good probability) to solve the system whenever $nr \geq m(n-k)$, therefore it is possible to find in probabilistic polynomial time a solution to a typical instance of the RSD problem whenever $r \geq \frac{m(n-k)}{n} = 1 + \lfloor \frac{m(n-k)}{n} \rfloor$, which corresponds to the Singleton bound (except if $\frac{m(n-k)}{n}$ is an integer).

This proves the following proposition :

Proposition 13.1. There is a probabilistic polynomial time algorithm that solves random instances of the App-RSD problem in polynomial time when $r \geq 1 + \lfloor \frac{m(n-k)}{n} \rfloor$ or for $r \geq \frac{m(n-k)}{n}$ if $\frac{m(n-k)}{n}$ is an integer.

For a rank weight r below this bound, the best known attacks are, as in the Hamming distance case, obtained by considering the cost of finding a word of rank r divided by the number of potential solutions : $\frac{B(n,k,m,q)}{q^{m(n-k)}}$. In practice the complexity we find is coherent with this.

13.3 Approximating a random syndrome beyond the GVR bound with LRPC codes

13.3.1 Decoding algorithm in rank metric

The rank metric has received a lot of attention in the context of network coding [89]. There exist very few algorithms, however, for decoding codes in the rank metric. The most well known $[n,k]$ codes which are decodable are the Gabidulin codes [44]. These codes can correct up to $\frac{n-k}{2}$ errors, and have been proposed for encryption : but since they cannot decode up to the GVR bound, they do not seem suitable for full decoding in the spirit of [28] for signature algorithms. Another more recent family of decodable codes are the LRPC codes [48], these codes are defined through a low rank matrix.

Definition (5). A Low Rank Parity Check (LRPC) code of rank d, length n and dimension k over $GF(q^m)$ is a code defined by an $(n-k)\times n$ parity check matrix $H=(h_{ij})$, such that all its coordinates

h_{ij} belong to the same $GF(q)$-subspace F of dimension d of $GF(q^m)$. We denote by $\{F_1, F_2, \cdots, F_d\}$ a basis of F.

These codes can decode with a good probability up to $\frac{n-k}{d}$ errors, they can be used for encryption [48], but since they can decode only up to $\frac{n-k}{2}$ errors at best, they also seems unsuitable for signature algorithms.

13.3.2 Using LRPC codes to approximate a random syndrome beyond the GVR bound

In this section we explain how LRPC codes can be used to approximate a random syndrome beyond the GVR bound.

High level overview

The traditional approach for decoding random syndromes, that is used by the CFS scheme for instance, consists in taking advantage of the decoding properties of a code (e.g. a Goppa code) and considering parameters for which the proportion of decodable vectors – the decodable density – is not too low. For the Hamming metric, this approach leads to very flat dual matrices, ie, codes with high rate and very low Hamming distance. In the rank metric case, this approach leads to very small decodable densities and does not work in practice. However, it is possible to proceed otherwise. It turns out that the decoding algorithm of LRPC codes can be adapted so that it is possible to decode not only errors but also (generalized) erasures. This new decoding algorithm allows us to decode more rank errors since the support is then partially known. In that case since the size of the balls depends directly on the dimension of the support, it leads to a dramatic increase of the size of the decodable balls. Semantically, what happens is that the signer can fix an erasure space, which relaxes the condition for finding a preimage. This approach works because in the particular case of our algorithm, it is possible to consider the erasure space at no cost in terms of error correction : to put it differently, the situation for LRPC is different from traditional Hamming metric codes for which "an error equals two erasures".

In practice it is possible to find parameters (not flat at all) for which it is possible to decode a random syndrome with the constraint that its support contains a fixed random subspace. Fixing part of the rank-support of the error, (the generalized erasure) allows us more rank-errors. For suitable parameters, the approach works then as follows : for a given random syndrome-space element s, one chooses a random subspace T of fixed dimension t (a generalized erasure of Definition 2), and the algorithm returns a small rank-weight word, whose rank-support E contains T, and whose syndrome is the given element s. Of course, there is no unicity of the error e since different choices of T lead

to different errors e, which implies that the rank of the returned error is above the GVR bound : it is however only just above the GVR bound for the right choice of parameters.

LRPC decoding with errors and generalized erasures

Setting : Let an $[n, k]$ LRPC code be defined by an $(n-k) \times n$ parity-check matrix H whose entries lie in a space $F \subset GF(q^m)$ of small dimension d. Let t and r' be two parameters such that

$$r' \leq \frac{n-k}{d}.$$

Set $r = t + r'$. Given an element of the syndrome space s, we will be looking for a rank r vector e of $GF(q^m)^n$ with syndrome s. We first look for an acceptable subspace E of dimension r of $GF(q^m)$ and then solve the linear system $H.e^t = s$ where $e \in E^n$. To this end we choose a random subspace T of dimension t of $GF(q^m)$ and impose the condition $T \subset E$.

The subspace T being fixed, we now describe the set of decodable elements of the syndrome space. We will then see how to decode them.

Definition (6)**.** Let F_1 and F_2 be two fixed linearly independent elements of the space F. We shall say that an element $s \in GF(q^m)^{n-k}$ of the syndrome space is *T-decodable* if there exists a rank r subspace E of $GF(q^m)$ satisfying the following conditions.

(i) $\dim\langle FE\rangle = \dim F \dim E$,

(ii) $\dim(F_1^{-1}\langle FE\rangle \cap F_2^{-1}\langle FE\rangle) = \dim E$,

(iii) the coordinates of s all belong to the space $\langle FE\rangle$ and together with the elements of the space $\langle FT\rangle$ they generate the whole of $\langle FE\rangle$.

Decoding algorithm. We now argue that if a syndrome s is T-decodable, we can effectively find e of rank r such that $H.e^t = s$. We first determine the required support space E. Since the decoder knows the subspaces F and T, he has access to the product space $\langle FT\rangle$. He can then construct the subspace S generated by $\langle FT\rangle$ and the coordinates of s. Condition (iii) of T-decodability ensures that the subspace S is equal to $\langle FE\rangle$ for some E, and since

$$F_1^{-1}\langle FE\rangle \cap F_2^{-1}\langle FE\rangle \supset E,$$

condition (ii) implies that E is uniquely determined and that the decoder recovers E by computing the intersection of subspaces $F_1^{-1}S \cap F_2^{-1}S$.

It remains to justify that once the subspace E is found, we can always find e of support E such that $H.e^t = s$. This will be the case if the mapping

$$\begin{array}{rcl} E^n & \to & \langle FE\rangle^{n-k} \\ e & \mapsto & H.e^t \end{array} \tag{13.1}$$

can be shown to be surjective. Extend $\{F_1, F_2\}$ to a basis $\{F_1, \cdots, F_d\}$ of F and let $\{E_1, \cdots, E_r\}$ be a basis of E. Notice that the system $H.e^t = s$ can be rewritten formally as a linear system in the small field $GF(q)$ where the coordinates of e and the elements of H are written in the basis $\{E_1, \cdots, E_r\}$ and $\{F_1, \cdots, F_d\}$ respectively, and where the syndrome coordinates are written in the product basis $\{E_1.F_1, \cdots, E_r.F_d\}$. We therefore have a linear system with $n.r$ unknowns and $(n-k).rd$ equations over $GF(q)$ that is defined by an $(n.r) \times (n-k)rd$ formal matrix H_f (*say*) whose coordinates are functions only of H (see [48] for more details on how to obtain H_f from H).

We now see that the matrix H can be easily chosen so that the matrix H_f is of maximal rank $n.r$, which makes the mapping (13.1) surjective, *for any subspace* E of dimension d satisfying condition (i) of T-decodability.

Remarks :

1. For applications, we will consider only the case where $nr = (n-k)rd$, meaning that the mapping (13.1) is always one-to-one.
2. The system $H.e^t = s$ can be formally inverted and stored in a pre-processing phase, so that the decoding complexity is only that of multiplication by a square matrix of length nr, rather than a cubic inversion.
3. In principle, the decoder could derive the support E by computing

$$E = F_1^{-1}S \cap \cdots \cap F_d^{-1}S \tag{13.2}$$

   rather than simply $E = F_1^{-1}S \cap F_2^{-1}S$, and the procedure would work in the same way in cases when (13.2) holds but not the simpler condition (ii). This potentially increases the set of decodable syndromes, but the gain is somewhat marginal and condition (ii) makes the forthcoming analysis simpler. For similar reasons, when conditions (i)–(iii) are not all satisfied, we do not attempt to decode even if there are cases when it stays feasible.

Figure 13.1 summarizes the decoding algorithm. Note that the decoder can easily check conditions (i)–(iii), and that a decoding failure is declared when they are not satisfied.

13.3.3 Proportion of decodable syndromes for unique decoding of LRPC codes

Signature algorithms based on codes all inject the message space in some way into the syndrome space and then decode them to form a signature. We should therefore estimate the proportion of decodable syndromes. The classical decoding approach tells us to look for a preimage by H that sits on the Gilbert-Varshamov bound : for typical random codes, a preimage typically exists and is (almost) unique. Computing such a preimage is a challenge, however. In our case, we are looking for a preimage above the Gilbert-Varshamov bound, for which many preimages exist, but for a fixed (erasure) subspace T, decoding becomes unique again. In the following, we count the number of

Input : $T = \langle T_1, \cdots, T_t \rangle$ a subspace of $GF(q^m)$ of dimension t, H an $(n-k) \times n$ matrix with elements in a subspace $F = \langle F_1, \cdots, F_d \rangle$ of dimension d, and $s \in GF(q^m)^{n-k}$.
Output : a vector $e = (e_1, \dots e_n)$ such that $s = H.e^t$, with $e_i \in E$, E a subspace of dimension $\dim E = r = t + \frac{n-k}{d}$ satisfying $T \subset E$.

1. **Syndrome computations**
   a) Compute a basis $B = \{F_1 T_1, \cdots, F_d T_t\}$ of the product space $\langle F.T \rangle$.
   b) Compute the subspace $S = \langle B \cup \{s_1, \cdots, s_{n-k}\} \rangle$.

2. **Recovering the support E of the error**
   Compute the support of the error $E = F_1^{-1} S \cap F_2^{-1} S$, and compute a basis $\{E_1, E_2, \cdots, E_r\}$ of E.

3. **Recovering the error vector** $e = (e_1, \dots, e_n)$
   For $1 \leq i \leq n$, write $e_i = \sum_{i=1}^{n} e_{ij} E_j$, solve the system $H.e^t = s$, where the equations $H.e^t$ and the syndrome coordinates s_i are written as elements of the product space $P = \langle E.F \rangle$ in the basis $\{F_1 E_1, \cdots, F_1 E_r, \cdots, F_d E_1, \cdots, F_d E_r\}$. The system has nr unknowns (the e_{ij}) in $GF(q)$ and $(n-k).rd$ equations from the syndrome.

FIGURE 13.1 – Algorithm 1 : a general errors/erasures decoding algorithm for LRPC codes

T-decodable syndromes and show that for some adequate parameter choices, their proportion can be made to be close to 1.

It will be convenient to use the following notation.

Definition (7)**.** For a subspace T of $GF(q^m)$ of dimension t, denote by $\mathcal{E}(T)$ the number of subspaces of dimension $r = r' + t$ that contain T.

Lemma (8)**.** We have

$$\mathcal{E}(T) = \prod_{i=0}^{r'-1} \left(\frac{q^{m-t-i} - 1}{q^{i+1} - 1} \right)$$

Proof (3)**.** Consider the case where $r = t+1$, we need to construct distinct subspaces of dimension $t+1$ containing T. This can be done by adjoining an element of $GF(q^m)$ modulo the subspace T, which gives $(q^m - q^t)/(q^{t+1} - q^t) = (q^{m-t} - 1)/(q-1)$ possibilities. Now any subspace of dimension $t+1$ contains $q^{t+1} - 1$ subspaces of dimension t containing T. A repetition of this approach $r' - 1$ times gives the formula. (see also [67] p.630).

Theorem (9)**.** The number $\mathcal{T}(t, r, d, m)$ of T-decodable syndromes satisfies the upper bound :

$$\mathcal{T}(t, r, d, m) \leq \mathcal{E}(T) q^{rd(n-k)}.$$

Furthermore, under the conditions $r(2d-1) \leq m$ and

$$\dim\langle FT\rangle = \dim F \dim T, \tag{13.3}$$
$$\dim(F_1^{-1}F + F_2^{-1}F) = 2\dim F - 1 = 2d-1, \tag{13.4}$$

we also have the lower bound :

$$\left(1 - \frac{1}{q-1}\right)^2 \mathcal{E}(T) q^{rd(n-k)} \leq \mathcal{T}(t,r,d,m).$$

Note that condition (13.4) depends only on the subspace F and can be ensured quite easily when designing the matrix H. Random spaces F with random elements F_1 and F_2 will typically have this property. Condition (13.3) depends on the choice of the subspace T : as will be apparent from Lemma below, for a random subspace T condition (13.3) holds with probability very close to 1.

To prove Theorem 9 we will rely on the following lemma.

Lemma (10). Let A be a fixed subspace of $\mathbb{F}_q^m$ of dimension α and let T be a subspace of dimension t (with possibly $t=0$) such that $\dim\langle AT\rangle = \alpha t$. Let B be a subspace generated by T together with β random independent uniform vectors, with β satisfying $\alpha(t+\beta) \leq m$. Then

$$\mathbb{P}\left(\dim\langle AB\rangle < \alpha(t+\beta)\right) \leq \frac{q^{\alpha(t+\beta)}}{(q-1)q^m}.$$

Proof of lemma 10.

Proof (4). Suppose first that $B = B' + \langle b\rangle$ where b is a uniformly chosen random element of $\mathbb{F}_q^m$ and where $B' \supset T$ is a fixed space such that $\dim\langle AB'\rangle = \alpha(t+\beta-1)$. Let A^P be a projective version of A, meaning that for every $a \neq 0$ in A, we have exactly one element of the set

$$\{\lambda a, \lambda \in \mathbb{F}_q^*\}$$

in A^P.

We have $\dim\langle AB\rangle < \alpha(t+\beta-1)+\alpha$ if and only if the subspace bA has a non-zero intersection with $\langle AB'\rangle$, and also if and only if the set bA^P has a non-zero intersection with $\langle AB'\rangle$. Now,

$$\begin{aligned}\mathbb{P}\left(\dim\langle AB'\rangle \cap Ab \neq \{0\}\right) &\leq \sum_{a\in A^P,\, a\neq 0} \mathbb{P}\left(ab \in \langle AB'\rangle\right) \\ &= \frac{|A|-1}{q-1}\frac{q^{\alpha(t+\beta-1)}}{q^m} \\ &= \frac{q^{\alpha(t+\beta)}}{(q-1)q^m} - \frac{q^{\alpha(t+\beta-1)}}{(q-1)q^m}.\end{aligned} \tag{13.5}$$

since for any fixed $a \neq 0$, we have that ab is uniformly distributed in $\mathbb{F}_q^m$, and since the number of elements in bA^P equals $(|A|-1)/(q-1)$.

Now write

$$B_0 = T \subset B_1 = T + \langle b_1 \rangle \subset B_2 = T + \langle b_1, b_2 \rangle \subset \cdots, \subset B_i = T + \langle b_1, \ldots, b_i \rangle \subset \cdots \subset B = B_\beta$$

where $b_1 \ldots, b_\beta$ are independent uniform vectors in $\mathbb{F}_q^m$. We have that the probability

$$\mathbb{P}\left(\dim\langle AB \rangle < \dim A \dim B\right)$$

that AB is not full-rank is not more than

$$\sum_{i=1}^{\beta} \mathbb{P}\left(\dim\langle AB_i \rangle < \dim A \dim B_i \mid \dim\langle AB_{i-1} \rangle = \dim A \dim B_{i-1}\right)$$

so that (13.5) gives :

$$\begin{aligned}\mathbb{P}\left(\dim\langle AB \rangle < \dim A \dim B\right) &\leq \frac{1}{q-1} \sum_{i=0}^{\beta-1} \left(\frac{1}{q^{m-(t+i+1)\alpha}} - \frac{1}{q^{m-(t+i)\alpha}} \right) \\ &\leq \frac{1}{q-1} \left(\frac{1}{q^{m-\alpha(t+\beta)}} - \frac{1}{q^{m-t\alpha}} \right) \leq \frac{1}{(q-1)q^{m-\alpha(t+\beta)}}.\end{aligned}$$

Proof of Theorem 9.

Proof (5)**.** To obtain a T-decodable syndrome, we must choose $n-k$ elements in a space $\langle FE \rangle$ for a given space E that contains T. There are $\mathcal{E}(T)$ ways of choosing E, and for any given E there are at most $q^{\dim F \dim E} = q^{dr}$ ways of choosing a syndrome coordinate in $\langle FE \rangle$. This gives the upper bound on $\mathcal{T}(t, r, d, m)$.

We proceed to prove the lower bound. First consider that Lemma 10 proves that, when we randomly and uniformly choose a subspace E that contains T, then with probability at least $1 - 1/(q-1)$, we have :

$$\dim\langle (F_1^{-1}F + F_2^{-1}F)E \rangle = \dim(F_1^{-1}F + F_2^{-1}F) \dim E = (2d-1)r$$

by property (13.4). This last fact implies, that

$$\dim(F_1^{-1}\langle FE \rangle + F_2^{-1}\langle FE \rangle) = 2dr - r \tag{13.6}$$

since clearly

$$F_1^{-1}\langle FE \rangle + F_2^{-1}\langle FE \rangle = \langle (F_1^{-1}F + F_2^{-1}F)E \rangle.$$

Now, since we have $E \subset F_1^{-1}F \cap F_2^{-1}F$, applying the formula $\dim(A+B) = \dim A + \dim B - \dim A \cap B$ to (13.6) gives us simultaneously that :

$$\begin{aligned}\dim\langle FE \rangle &= dr \\ F_1^{-1}F \cap F_2^{-1}F &= E.\end{aligned}$$

In other words, both conditions (i) and (ii) of T-decodability are satisfied. We have therefore proved that the proportion of subspaces E containing T that satisfy conditions (i) and (ii) is at least $(1 - 1/(q-1))$.

Now let E be a fixed subspace satisfying conditions (i) and (ii). Among all $(n-k)$-tuples of elements of $\langle FE \rangle$, the proportion of those $(n-k)$-tuples that together with $\langle FT \rangle$ generate the whole of $\langle FE \rangle$ is at least

$$\left(1 - \frac{1}{q}\right)\left(1 - \frac{1}{q^2}\right) \ldots \left(1 - \frac{1}{q^i}\right) \ldots \geq 1 - \frac{1}{q-1}. \tag{13.7}$$

We have therefore just proved that given a subspace E satisfying conditions (i) and (ii), there are at least $(1 - 1/q)(q^{rd})^{n-k}$ $(n-k)$-tuples of $\langle FE \rangle^{n-k}$ satisfying condition (iii).

To conclude, notice that since a T-decodable syndrome entirely determines the associated subspace E, the set of T-decodable syndromes can be partitioned into sets of $(n-k)$-tuples of $\langle FE \rangle^{n-k}$ satisfying condition (iii) for all E satisfying conditions (i) and (ii). The two lower bounds on the number of such E and the number of T-decodable syndromes inside a given $\langle FE \rangle^{n-k}$ give the global lower bound of the Theorem.

Remarks :

1. It can be shown with a finer analysis that the term $(1 - 1(q-1))^2$ in the lower bound can be improved to a quantity close to $1 - 1(q-1)$.
2. For large q, Theorem 9 shows that, for most choices of T, the density of T-decodable syndromes essentially equals

$$\frac{\mathcal{E}(T) q^{rd(n-k)}}{q^{m(n-k)}} \approx q^{(r-t)(m-r)+(n-k)(rd-m)}. \tag{13.8}$$

Remarkably, it is possible to choose sets of parameters (m, t, r, d), with $(n-k) = d(r-t)$, such that the exponent in (13.8) equals zero, which gives a density very close to 1.

Example of parameters with density almost 1 : For $q = 2, m = 45, n = 40, k = 20, t = 5, r = 10$, the algorithm decodes up to $r = t + r' = 15$ for a fixed random partial support T of dimension 5. The GVR bound for a random $[40, 20]$ code with $m = 45$ is 13, the Singleton bound is 20, we see that the decoding radius 15 is therefore just above the GVR bound at 13 and rather far from the Singleton bound at 20. Moreover one can notice that if parameters (m, t, r, d) satisfy the two equations $(r-t)(m-r) + (n-k)(rd-m) = 0$ and $(n-k) = d(r-t)$ (the case for which the density is almost 1), then for any integer α greater than 1, the parameter set $(\alpha m, \alpha t, \alpha r, d)$ satisfies the same equations, and hence for a given d one obtains an infinite family of parameters with density almost 1.

Decoding in practice. In practice it easy enough to find sets of parameters for which the density of decodable syndromes is very close to 1, i.e. such that $(r-t)(m-r) + (n-k)(rd-m) = 0$. When $q = 2$, the formula (13.8) is only an approximation, and simulations give a density close to $1/3$ rather than 1.

13.4 RankSign : a rank-based signature scheme

We saw in the previous section how to construct a matrix H of an LRPC code, with a unique support decoding, which opens the way for a signature algorithm. In practice the best decoding results are obtained for $d = 2$: the natural strategy is to define for the public key a matrix $H' = AHP$, where A is a random $(n-k) \times (n-k)$ invertible matrix in the extension field and P is an invertible $n \times n$ matrix in the small field. However, it is easily possible for a cryptanalyst to recover the words of small weight $d = 2$ in H' and it is therefore necessary to hide the matrix H in another way. In what follows we present a simple type of masking : $RankSign^{+}$ which consists in adding a few random columns to H, and other more complex types of masking : $RankSign^{\times}$ (RankSign-multiply) and $RankSign^{+\times}$.

13.4.1 The $RankSign^{+}$ scheme : augmented LRPC codes

Suppose one has a fixed support T of dimension t. We consider the public matrix $H' = A(R|H)P$ with R a random $(n-k) \times t'$ matrix in $GF(q^m)$. We will typically take $t' = t$ but one could envisage other values of t'. We denote by *augmented LRPC codes* such codes with parity-check matrices $H' = A(R|H)P$.

Starting from a partial support T that has been randomly chosen and is then fixed, the signature consists in decoding not a random s but the syndrome $s' = s - R.(e_1, \cdots, e_t)^t$ for e_i random independent elements of T.

The overall rank of the solution vector e is still $r = t + r'$. the masking gives us that the minimum rank-weight of the code generated by the rows of H' is $t+d$ rather than purely d : therefore recovering the hidden structure involves finding relatively large minimum weight vectors in a code. In practice we consider $d = 2$ and H is a $n/2 \times n$ matrix with all coordinates in a space F of dimension 2. Moreover for $\{F_1, F_2\}$ a basis of F, we choose the matrix H such that when H is written in the basis $\{F_1, F_2\}$, one obtains a $n \times n$ invertible matrix (of maximal rank) over $GF(q)$. It can be done easily. Figure 13.2 describes the scheme, where —— denotes concatenation.

Parameters : *Public key size :* $(k+t)(n-k)mLog_2(q)$ *Signature size :* $(m+n+t)rLog_2(q)$.

The cost of the decoding algorithm is quadratic because of preprocessing of H_f^{-1}, hence the major cost comes from the linear algebra over the large field $GF(q^m)$.

Signature complexity : $(n-k) \times (n+t)$ operations in $GF(q^m)$. *Verification complexity :* $(n-k) \times (n+t)$ operations in $GF(q^m)$.

1. **Secret key :** an augmented LRPC code over $GF(q^m)$ with parity-check matrix $(R|H)$ of size $(n-k) \times (n+t)$ which can decode r' errors and t generalized erasures : a randomly chosen $(n-k) \times (n-k)$ matrix A that is invertible in $GF(q^m)$ a randomly chosen $(n+t) \times (n+t)$ matrix P invertible in $GF(q)$.
2. **Public key :** the matrix $H' = A(R|H)P$, a small integer value l, a hash function $hash$.
3. **Signature of a message M :**

   a) *initialization* : seed $\leftarrow \{0,1\}^l$, pick t random independent elements $(e_1, \cdots, e_t)$ of $GF(q^m)$

   b) *syndrome* : $s \leftarrow hash(M||seed) \in GF(q^m)^{n-k}$

   c) decode by the LRPC matrix H, the syndrome $s' = A^{-1}.s^T - R.(e_1, \cdots, e_t)^T$ with erasure space $T = \langle e_1, \cdots, e_t \rangle$ and r' errors by Algorithm 1.

   d) if the decoding algorithm works and returns a word $(e_{t+1}, \cdots, e_{n+t})$ of weight $r = t + r'$, signature=$((e_1, \cdots, e_{n+t}).(P^T)^{-1}, seed)$, else return to a).
4. **Verification** : Verify that $Rank(e) = r = t + r'$ and $H'.e^T = s = hash(M||seed)$.

FIGURE 13.2 – The $RankSign^+$ signature algorithm

The length l of the seed can be taken as $\frac{80}{Log_2(q)}$ for instance.

13.4.2 Optimized parameters : cyclic-RankSign

In order to decrease the size of the public key it is possible to use a quasi-cyclic structure. Whenever all parameters n, k, t or/and l are divisible by a certain cyclicity order σ it is possible to consider that all matrices consist of small cyclic $\sigma \times \sigma$ random matrices. This particular type of structure divides the size of the public by a factor σ.

13.4.3 The $RankSign^{\times}$ scheme : multiplied LRPC codes

Let H be a parity-check matrix of an LRPC coder, for l a small fixed integer, we consider now M a random invertible $n \times n$ matrix of the form :

$$M = (M_1|M_2)$$

with M_1 a random $n \times l$ matrix over $GF(q^m)$ and M_2 a random $n \times (n-l)$ matrix in the base field $GF(q)$. Now consider the matrix

$$H'' = H.(M^{-1})^t$$

. We denote by *multiplied LRPC codes*, the codes with parity-check matrices of the form H''. The matrix H'' enables one to approximate a syndrome to a rank $r+t+l$ rather than $r+t$ with uniquely H. Hence the masking increases the rank of the small vector (remember that this rank has to be lower than the Singleton bound) but it considerably masks H. Indeed if M has a regular form, the inverse matrix M^{-1} is rather tricky and considerably masks H.

In that case one takes as public key : $H'' = AH.((MP)^{-1})^t$, suppose that for a given syndrome s we want to solve $H''.e'^t = s$, one proceeds as follows :

$$H''.e'^t = s \Leftrightarrow H.(e'(PM)^{-1})^T = A^{-1}s = s'$$

Rewriting the right handside as $H.e^t = s'$, we can decode s' in $e = e'(PM)^{-1}$ with $rank(e) = t+r$, and hence

$$e' = eMP$$

now since M has the particular form previously described, the rank of e' is $rank(e)+l = t+r+l$, the invertible matrix M does not change the weight.

Unlike the previous masking, this masking increases the obtained rank of the signature, and therefore makes the difference with the Singleton bound already big enough.

13.4.4 The $RankSign^{+\times}$ scheme : augmented-multiplied LRPC codes

This last masking combines the two previous masking scheme, we consider codes with parity-check matrices of the form :

$$H'.(M^{-1})^t$$

for H' an augmented matrix and M a matrix defined for multiplied LRPC, we denote these codes by *augmented-multiplied LRPC codes*. In the $RankSign^{\times}$ one starts from an augmented matrix H'

as in $RankSign^+$ rather than an LRPC parity-check matrix H. The decoding algorithm mixes in a straightforward way the two previous approaches.

13.5 Security

13.5.1 A difficult problem

We now introduce a problem to which the security of our scheme can be reduced, we then discuss the difficulty of the problem.

Approximate Syndrome Decoding Problem for augmented LRPC codes (App-RSD-LRPC-Aug)

Given a masked parity-check matrix $H' = A(R|H)P$ of an augmented LRPC codes as defined in the previous section, and a random syndrome s, find a vector x of not too small rank d (typically of rank (GVR + Singleton)/2) such that $H'.x^t = s$.

Assumption : *The problem App-RSD-LRPC-Aug is difficult .*

Discussion on the assumption : The family of augmented LRPC codes is not of course a family of random codes, but they are weakly structured codes : the main point being that they have a parity-check matrix one part of which consists only in low rank coordinates the other part consisting in random entries. The attacker never has direct access to the LRPC matrix H, which is hidden by the augmented part.

There are two different types of attacks on the problem : structural attacks in which the attacker tries to recover the structure of the code, or the attacker can try to forge directly a signature and try to find codewords of weight $r = t + r'$. We consider in the following these two approaches and explain why it is difficult to attack this problem.

Remark There is a long history of broken cryptosystems related to hiding structured matrices. One can though remark the following : 1) there are strongly structured families of hidden codes like the Goppa-McEliece scheme which have not been attacked, 2) the families of hidden structures which have been attacked rely usually on ***very*** strongly structured codes like the Reed-Solomon codes or the Gabidulin codes [37], which are very difficult to hide. Moreover there exist hidden families of codes or lattices with a low structure which have never been really attacked like for instance the NTRU lattices or the double circulant codes. It seems that the weak structure of the LRPC codes place them in this latter category.

Structural attacks on the public key : trying to recover the secret key

Previously known structural attacks The main structural attack for the rank metric is the Overbeck attack on the GPT cryptosystem, the attack consists in considering concatenated public matrices $G^q, G^{q^2}, ..., G^{q^{n-k-}}$, in that case the particular structure of Gabidulin codes enables one to find a concatenated matrix with a rank default ; this is due to the particular structure of the Gabidulin codes and the fact that for Gabidulin codes G^{q^i} is very close to $G^{q^{i+1}}$. In the case of LRPC codes, since the rows are taken randomly in a small space, this attack makes no sense, and cannot be generalized.

Attack on the parity-check matrix H' : another approach consists in finding directly words of small weight induced by the structure of the code, from which one can hope to recover the global structure. For augmented LRPC codes, the rank of the minimum weight words is $d+t : d$ for LRPC and t for the masking. This attack becomes very hard when t increases, even for low t. For instance for $t = 2$ and $d = 2$ it gives a minimum weight of 4, which for most parameters n and k is already out of reach of the best known attacks on the rank syndrome decoding (see Section 2).

Attack on the isometry matrix P : The attacker can also try to guess the action of P on H, since d is usually small negating this action may permit to attack directly a code of rank d. Since d is small it is enough to guess the resulting action of P on $n - k + 3$ columns by considering only the action of P coming from the first t columns of the matrix R - the only columns which may increase the rank-, it means guessing $(n - k + 3) \times t$ elements of $GF(q)$ (since coordinates of P are in $GF(q)$, hence a complexity of $q^{(n-k+3)t}$. In general this attack is not efficient, except in the case of quasi-cyclic matrices and small q, in which case the complexity becomes $q^{\frac{(n-k+3)t}{\sigma}}$, for σ the index of quasi-cyclicity.

Direct attack on the syndrome

In that case one has to find a word of weight between the GVR bound and the Singleton bound, as discussed in Section 2 the best attack consists in computing the complexity of finding a word of given weight divided by the number of such words (the traditionnal approach for Hamming metric, which is also applicable here). Parameters are chosen in order to resist this attack.

Information leakage.

We considered in previous attacks the case where no additional information was known besides the public parameters. Often the most efficient attacks on signatures is to recover the hidden structure of the public key by using information leaking from real signatures. This for instance is what happened in the case of NTRUSign : the secret key is not directly attacked, but the information leaked from real signatures enables one to recover successfully the hidden structure. We show in the next section

that with our masking scheme no such phenomenon can occur, since we prove that, if an attacker can break the signature scheme for public augmented matrices with the help of information leaking from a number of (approximately) q real signatures, then he can also break the scheme just as efficiently **without** any authentic signatures.

13.5.2 Unforgeability

Theorem 11 below states the unforgeability of signatures. It essentially states that valid signatures leak no information on the secret key. More precisely, under the random oracle model, there exists a polynomial time algorithm that takes as input the public matrix H' and, produces couples (m, σ), where m is a message and σ a valid signature for m when one's only access to the hashing oracle is through the simulator, and this with the same probability distribution as those output by the authentic signature algorithm. Therefore whatever forgery can be achieved from the knowledge of H' and a list of valid signed messages, can be simulated and reproduced with the public matrix H' as only input.

Theorem (11)**.** : For any algorithm $\mathcal{A}$ that leads to a forged signature using $N \leq q/2$ authentic signatures, there is an algorithm $\mathcal{A}'$ with the same complexity that leads to a forgery using only the public key as input and without any authentic signatures.

Proof (6)**.** Recall that a signature of a message M is a pair (x', y') where y' is a hashed value of the message M and $y' = H'x'^T$ and $\text{rank}(x') = r$. If $\mathcal{A}$ is an algorithm that leads to a forgery with the use of N authentic signatures, then the algorithm $\mathcal{A}'$ consists of a simulated version of $\mathcal{A}$ where authentic signatures (x', y') are replaced by couples (x'', y'') where x'' is randomly and uniformly chosen among vectors of rank-weight e, and $y'' = H'x''^T$ is claimed to be the hashed value of the message M output by a random oracle. In the random oracle model, the algorithm $\mathcal{A}'$ must behave exactly as algorithm $\mathcal{A}$ and give the same output whenever (x'', y'') is statistically indistinguishable from (x', y').

We now compare the statistics of (x', y') and (x'', y''). We have $H' = A(R|H)P$ and since the transformation :

$$(x', y') \mapsto (x^a = Px', y^a = A^{-1}y') \tag{13.9}$$

$$(x'', y'') \mapsto (x^s = Px'', y^s = A^{-1}y'') \tag{13.10}$$

is one-to-one, comparing the statistics of (x', y') and (x'', y'') amounts to comparing the distributions of (x^a, y^a) (authentic) and (x^s, y^s) (simulated).

Now (x^a, y^a) is obtained in the following way : the signer chooses a subspace T of $\mathbb{F}_{q^m}$ together with a random vector $\tau \in T^t$ of rank t and is given a vector u which is uniformly distributed in the syndrome space $\mathbb{F}_{q^m}^{n-k}$ and is equal to $A^{-1}h(M) - R\tau^T$. Precisely, the signer chooses a random vector τ of rank t, and sets T to be the subspace generated by its coordinates. The signer then proceeds to try to decode u, meaning it looks for a subspace E of $\mathbb{F}_{q^m}$ that contains T and such that all

coordinates of u fall into $\langle FE \rangle$, where F is the space generated by the elements of the LRPC matrix H. He succeeds exactly when the syndrome vector u is T-decodable in the sense of Definition 6 : when this doesn't occur, the decoder aborts.

When the syndrome u is T-decodable, the decoder proceeds to solve the equation

$$Hx_H^T = u \tag{13.11}$$

and then sets

$$x^a = (\tau, x_H)$$

to create the couple $(x^a, y^a = (R|H)(x^a)^T)$ in (13.9). Recall from Remark 1 in Section 13.3.2 that the matrix H has been chosen so that equation (13.11) (equivalently equation (13.1)) always has a unique solution for every T-decodable u.

We may therefore speak about T-decodable *couples* (x_H, u), where u uniquely determines x_H and x_H uniquely determines u. Now, to re-cap, the authentic signer starts with uniformely random u, and whenever u turns out to be non T-decodable, then we declare a decoding failure and start the whole process again generating another τ, another random space T and another u by another call to the random oracle h (meaning a counter appended to the message M is incremented before applying the random hash again). This keeps happening until we hit a T-decodable u. We see therefore that when it does hit a T-decodable u, the couple

$$(x_H, u)$$

is uniformly distributed among all T-decodable couples.

We now turn to the action of the simulator : what the simulator does is he tries to generate a uniform T-decodable couple (x_H, u) through x_H rather than through u like the signer.

Specifically, the simulator starts with a random subspace E of $\mathbb{F}_{q^m}$ of dimension r and an x'' with coordinates independently and uniformly drawn from E. Since the transformation (13.10) $x'' \mapsto x^s = Px''$ is rank-preserving, the simulator is implicitely creating a uniform vector x^s of E^n. Write

$$x^s = (\tau, x_H).$$

With overwhelming probability (at least $1 - 1/q^{r-t}$), the vector $\tau \in E^t$ is of maximum rank-weight t, since its coordinates are independently and uniformly chosen in E. The vector τ generates the required random space T. Let $u = Hx_H^T$: note that by construction, all its coordinates must be in $\langle FE \rangle$. Consider the conditions (i),(ii),(iii) of Definition 6 for u to be T-decodable.

Remember that the first two conditions (i) and (ii) are properties *only of the subspace* E. When they are not satisfied, no choice of x_H can yield a T-decodable couple (x_H, u). When conditions (i) and (ii) are satisfied we have that, since the mapping (13.11) $x_H \mapsto u$ is invertible, the vector u is a uniform random vector in $\langle FE \rangle^{n-k}$. Since $n - k + \dim\langle FT \rangle = \dim\langle FE \rangle$, the probability that

condition (iii) is not satisfied is governed by the probability than a random vector falls into a given subspace of $\langle FE \rangle$ of co-dimension 1 and is of the order of $1/q$: it is also at most $1/(q-1)$ according to computation (13.7). We also see that the number of x_H that satisfies condition (iii) is independent of the space E and is always the same for all E that satisfy conditions (i) and (ii). This last fact implies that when

- A random uniform subspace E is chosen among all possible subspaces E of dimension r,
- a random x_H is chosen in E^n,

then either (x_H, u) is not T-decodable, or (x_H, u) is T-decodable and is uniformly distributed among all T-decodable couples.

The simulator has no oracle to tell him when he has produced a non T-decodable couple, he can only hope this doesn't occur. As long as he produces T-decodable couples (x_H, u), then they are distributed (uniformly) exactly as those that are produced by the authentic signer and are undistinguishable from them. If we call π the probability that he produces a non-decodable u, then he can reasonably expect to produce a list of approximately $N = 1/\pi$ signatures (x'', y'') that are undistiguishable from genuine signatures in the random oracle model.

Consider now the probability π that the simulator produces a non-decodable u. It is at most the sum of the probabilities that E does not satisfy (i) and (ii) and the probability $\leq 1/(q-1)$ that (iii) is not satisfied. The probability that E does not satisfy (i) and (ii) is at most the probability that the product space $\langle (F_1^{-1}F + F_2^{-1})E \rangle$ does not have maximal dimension $(2d-1)r$, as argued in the proof of Theorem 9, and is at most $1/(q-1)$. We obtain therefore $\pi \leq 1/(q-1) + 1/(q-1) = 2/(q-1) \approx 2/q$ which concludes the proof.

Corollary (2)**.** Let us take $q > 2^{t_{sec}}$, with t_{sec} a security parameter polynomially equivalent to n. If a polynomial time forger succeeds in forging the signature with some probability, he can solve the difficult problem App-RSD-LRPC-Aug with the same complexity and the same probability.

Proof (7)**.** Theorem 11 tells us that either the forger used $N > q/2$ signatures, or he did not use any. In the case when the forger did not use signatures, the problem of forging a new signature is exactly the same as the problem App-RSD-LRPC-Aug. In the other case, the forger used $N > q/2$ signatures and the forger could not run in polynomial time in the parameter n.

13.6 Practical security and parameters

13.6.1 Parameters

In the following we give some examples of parameters. The parameters are adjusted to resist all previously known attacks. The security reduction holds for up to $q/2$ signatures, hence if one considers $q = 2^{40}$ it means we are protected against leakage for up to 2^{40} obtained authentic signatures. such an

amount of signatures is very difficult to obtain in real life, moreover if one multiplies by the amount of time necessary to obtain a signature (about 2^{30} for $q = 2^{40}$) we clearly see that obtaining such a number of authentic signatures is out of reach, and it justifies our security reduction.

We also give parameters for q lower than 2^{40} : in that case the reduction is weaker in the sense that it does not exclude a leaking attack for sufficiently many signatures. However, such a leaking attack seems difficult to obtain anyway, and these parameters can be seen as challenges for our system.

In the case of large q, the best attacks are structural attacks for recovering a vector of weight $t+d$ orthogonal to the matrix H'. The best attacks are algebraic attacks from Levy-Perret : for example the case $n = 16$ gives 270 quadratic equations for 126 unknowns, with a theoretical complexity of 2^{120} from [9]. For smaller q, the best attacks are combinatorial (Gaborit *et al.*).

- $RankSign^{+}$

In that case the matrix H' is a $(n-k) \times (n+t)$ matrix.

n	n-k	m	q	t	r'	r	GVR	Singleton	public key (bits)	sign. size (bits)	security
16	8	18	2^{40}	2	4	6	5	8	57600	8640	130
16	8	18	2^{8}	2	4	6	5	8	11520	1728	120
16	8	18	2^{16}	2	4	6	5	8	23040	3456	120
32	16	39	2	5	8	13	10	16	13104	988	80
40	20	45	2	5	10	15	12	20	22500	1350	110

- $Cyclic - RankSign^{+}$

In that case the matrix H' is a $(n-k) \times (n+t)$ quasi-cyclic matrix, and all matrices A, P, R are formed with concatenation of $\sigma \times \sigma$ cyclic blocks. We consider solely the case $\sigma = 2$. The complexity may be a little smaller than in the previous case since in that case attacks that try to guess a part of P are more efficient because of the cyclicity.

n	n-k	m	q	t	r'	r	σ	Singleton	public key (bits)	sign. size (bits)	security
16	8	18	2^{40}	2	4	6	2	8	28300	8640	130
16	8	18	2^{8}	2	4	6	2	8	5760	1728	90
16	8	18	2^{16}	2	4	6	2	8	11520	3456	120

13.6.2 Implementation

We implemented our scheme in a non optimized way, the results we obtained showed that for small q the scheme was very fast, when q increases, one has to consider the cost of multiplication in $GF(q)$, however for $q = 2^8$ or $q = 2^{16}$ some optimized implementation may reduce this cost.

13.7 Conclusion

In this paper we introduced a new approach to devising signatures with coding theory and in particular in the rank metric, by proposing to decode both erasures and errors rather than simply errors. This approach enables one to return a small weight word beyond the Gilbert-Varshamov bound rather than below. We proposed a new efficient algorithm for decoding LRPC codes which makes this approach a realizable. We then proposed a signature scheme based on this algorithm and the full decoding of a random syndrome beyond the Gilbert-Varshamov bound. We also showed that it was possible to protect our system against leakage from authentic signatures. Overall we propose different types of parameters, some of which are rather small.

Publication Personnelles

[1] Bettaieb, Slim and Schrek, Julien : Improved Lattice-Based Threshold Ring Signature Scheme. *Post-Quantum Cryptography 2013* :Springer 34–51

[2] Gaborit, Philippe and Schrek, Julien : Efficient code-based one-time signature from automorphism groups with syndrome compatibility. *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on* : IEEE 1982–1986

[3] Gaborit, Philippe and Schrek, Julien and Zémor, Gilles.Full cryptanalysis of the chen identification protocol. *Post-Quantum Cryptographya 2011* : Springer 35–50

[4] Aguilar, Carlos and Gaborit, Philippe and Schrek, Julien : A new zero-knowledge code based identification scheme with reduced communication. *Information Theory Workshop (ITW), 2011 IEEE* : 648–652

[5] Aguilar, Carlos and Slim Bettaieb and Gaborit, Philippe and Schrek, Julien : A Code-Based Undeniable Signature Scheme. *IMACC 2013* : Springer

En cours de soumission :

[6] Gaborit, Philippe and Ruatta, Olivier and Schrek, Julien : On the complexity of the Rank Syndrome Decoding problem. *arXiv preprint arXiv 2013* :1301.1026 2013 (IEEE Information Theory)

[7] Gaborit, Philippe and Ruatta, Olivier and Schrek, Julien and Gilles Zémor : An efficient signature algorithm based on rank metric. (EUROCRYPT) 2013

[8] Arnaud Dambra, Philippe Gaborit, Mylène Roussellet, Julien Schrek and Nicolas Tafforeau : Improved Implementation of Code-Based Signature Schemes on Embedded Devices. 2013 (CARDIS)

[9] Carlos Aguilar Melchor, Slim Bettaieb, Philippe Gaborit, Khoa Nguyen Ta Toan and Julien Schrek ; Lattice Based Undeniable Signature Scheme. (PKC) 2013

Bibliographie

[1] Carlos Aguilar, Philippe Gaborit, and Julien Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *Information Theory Workshop (ITW), 2011 IEEE*, pages 648–652. IEEE, 2011.

[2] C. Aguilar Melchor, P.L. Cayrel, and P. Gaborit. A new efficient threshold ring signature scheme based on coding theory. *Post-Quantum Cryptography*, pages 1–16, 2008.

[3] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in 2 n/20 : how 1+ 1= 0 improves information set decoding. In *Advances in Cryptology–EUROCRYPT 2012*, pages 520–536. Springer, 2012.

[4] Thierry P. Berger. Isometries for rank distance and permutation group of gabidulin codes. *IEEE Transactions on Information Theory*, 49(11) :3016–3019, 2003.

[5] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (Corresp.). *IEEE Transactions on Information Theory*, 24(3) :384–386, May 1978.

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

[7] Daniel J Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents : ball-collision decoding. In *Advances in Cryptology–CRYPTO 2011*, pages 743–760. Springer, 2011.

[8] Slim Bettaieb and Julien Schrek. Improved lattice-based threshold ring signature scheme. In *Post-Quantum Cryptography*, pages 34–51. Springer, 2013.

[9] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, 3(3) :177–197, 2009.

[10] Wieb Bosma, John Cannon, and Catherine Playoust. The magma algebra system i : The user language. *Journal of Symbolic Computation*, 24(3) :235–265, 1997.

[11] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. *Advances in Cryptology-Crypto 2002*, pages 75–99, 2002.

[12] Bruno Buchberger. Ein algorithmus zum auffinden der basiselemente des restklassenrings nach einem nulldimensionalen polynomideal. *Universitat Innsbruck, Austria, Ph. D. Thesis*, 1965.

[13] Jonathan F Buss, Gudmund S Frandsen, and Jeffrey O Shallit. The computational complexity of some problems of linear algebra. *Journal of Computer and System Sciences*, 58(3) :572–596, 1999.

[14] Pierre-Louis Cayrel, Ayoub Otmani, and Damien Vergnaud. On kabatianskii-krouk-smeets signatures. In *Arithmetic of Finite Fields*, pages 237–251. Springer, 2007.

[15] Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In *Selected Areas in Cryptography*, pages 171–186. Springer, 2011.

[16] P.L. Cayrel, R. Lindner, M. Rückert, and R. Silva. A lattice-based threshold ring signature scheme. *Progress in Cryptology–LATINCRYPT 2010*, pages 255–272, 2010.

[17] Florent Chabaud and Jacques Stern. The cryptographic security of the syndrome decoding problem for rank distance codes. In *Asiacrypt*, volume 1163, pages 368–381. Springer, 1996.

[18] Florent Chabaud and Jacques Stern. The cryptographic security of the syndrome decoding problem for rank distance codes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT*, volume 1163 of *Lecture Notes in Computer Science*, pages 368–381. Springer, 1996.

[19] Christophe Chabot and Matthieu Legeay. Using permutation group for decoding. In *Proceedings of the Twelfth International Workshop on Algebraic and Combinatorial Coding Theory, ACCT-12*, pages 86–92, 2010.

[20] D. Chaum. Zero-knowledge undeniable signatures. In *Advances in Cryptology-EUROCRYPT'90*, pages 458–464. Springer, 2006.

[21] D. Chaum and T. Pedersen. Wallet databases with observers. In *Advances in Cryptology-CRYPTO'92*, pages 89–105. Springer, 1993.

[22] D. Chaum and H. Van Antwerpen. Undeniable signatures. In *Advances in Cryptology-CRYPTO'89 Proceedings*, pages 212–216. Springer, 1990.

[23] D. Chaum and E. Van Heyst. Group signatures. In *Advances in Cryptology-EUROCRYPT'91*, pages 257–265. Springer, 1991.

[24] Kefei Chen. A new identification algorithm. In Ed Dawson and Jovan Dj. Golic, editors, *Cryptography : Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 244–249. Springer, 1995.

[25] Kefei Chen. A new identification algorithm. In *Cryptography : Policy and Algorithms*, pages 244–249. Springer, 1996.

[26] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a mceliece-based digital signature scheme. In *ASIACRYPT*, pages 157–174, 2001.

[27] Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier. Solving underdefined systems of multivariate quadratic equations. In *Public Key Cryptography*, pages 211–227. Springer, 2002.

[28] Nicolas T Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a mceliece-based digital signature scheme. In *Advances in Cryptology-ASIACRYPT 2001*, pages 157–174. Springer, 2001.

[29] L. Dallot and D. Vergnaud. Provably secure code-based threshold ring signatures. *Cryptography and Coding*, pages 222–235, 2009.

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

[31] F. Levy dit Vehel and L. Perret. Algebraic decoding of rank metric codes. In *proceedings of YACC06*. 2006.

[32] Jean-Charles Faugere. A new efficient algorithm for computing gröbner bases (¡ i¿ f¡/i¿¡ sub¿ 4¡/sub¿). *Journal of pure and applied algebra*, 139(1) :61–88, 1999.

[33] 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 of the 2010 International Symposium on Symbolic and Algebraic Computation*, pages 257–264. ACM, 2010.

[34] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. On the complexity of the generalized minrank problem. *Journal of Symbolic Computation*, 2013.

[35] Jean-Charles Faugere, Patrizia Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *Journal of Symbolic Computation*, 16(4) :329–344, 1993.

[36] Jean-Charles Faugère, Françoise Levy-Dit-Vehel, and Ludovic Perret. Cryptanalysis of minrank. In *Advances in Cryptology–CRYPTO 2008*, pages 280–296. Springer, 2008.

[37] Jean-Charles Faugere, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of mceliece variants with compact keys. In *Advances in Cryptology–EUROCRYPT 2010*, pages 279–298. Springer, 2010.

[38] Cédric Faure and Pierre Loidreau. A new public-key cryptosystem based on the problem of reconstructing p–polynomials. In *Coding and Cryptography*, pages 304–315. Springer, 2006.

[39] Amos Fiat and Adi Shamir. How to prove yourself : practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO'86*, pages 186–194. Springer, 1987.

[40] Matthieu Finiasz. Parallel-cfs - strengthening the cfs mceliece-based signature scheme. In *Selected Areas in Cryptography*, pages 159–170, 2010.

[41] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *Advances in Cryptology–ASIACRYPT 2009*, pages 88–105. Springer, 2009.

[42] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *ASIACRYPT*, pages 88–105, 2009.

[43] E. M. Gabidulin. Theory of Codes with Maximum Rank Distance. *Probl. Peredachi Inf.*, 21(1) :3–16, 1985.

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

[45] Ernst M Gabidulin. Attacks and counter-attacks on the gpt public key cryptosystem. *Designs, Codes and Cryptography*, 48(2) :171–177, 2008.

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

[47] Ernst M Gabidulin, AV Paramonov, and OV Tretjakov. Ideals over a non-commutative ring and their application in cryptology. In *Advances in Cryptology—EUROCRYPT'91*, pages 482–489. Springer, 1991.

[48] Philippe Gaborit, Gaétan Murat, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes and their application to cryptography.

[49] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. *arXiv preprint arXiv :1301.1026*, 2013.

[50] Philippe Gaborit and Julien Schrek. Efficient code-based one-time signature from automorphism groups with syndrome compatibility. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, pages 1982–1986. IEEE, 2012.

[51] Philippe Gaborit, Julien Schrek, and Gilles Zémor. Full cryptanalysis of the chen identification protocol. In *Post-Quantum Cryptography*, pages 35–50. Springer, 2011.

[52] Philippe Gaborit and Gilles Zemor. Asymptotic improvement of the gilbert–varshamov bound for linear codes. *Information Theory, IEEE Transactions on*, 54(9) :3865–3872, 2008.

[53] Keith Gibson. The security of the gabidulin public key cryptosystem. In *Advances in Cryptology—EUROCRYPT'96*, pages 212–223. Springer, 1996.

[54] M. Girault and P. Gaborit. Lightweight code-based authentification and signature. *In IEEE International Symposium on Information Theory – ISIT'2007*, pages 191–195, 2007.

[55] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology—CRYPTO'97*, pages 112–131. Springer, 1997.

[56] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1) :186–208, 1989.

[57] 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.

[58] Valerii Denisovich Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3) :24–30, 1970.

[59] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru : A ring-based public key cryptosystem. In *Algorithmic number theory*, pages 267–288. Springer, 1998.

[60] Gregory Kabatianskii, E Krouk, and Ben Smeets. A digital signature scheme based on random error-correcting codes. In *Crytography and Coding*, pages 161–167. Springer, 1997.

[61] Pil Joong Lee and Ernest F Brickell. An observation on the security of mceliece's public-key cryptosystem. In *Advances in Cryptology—EUROCRYPT'88*, pages 275–280. Springer, 1988.

[62] Jeffrey S Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *Information Theory, IEEE Transactions on*, 34(5) :1354–1359, 1988.

[63] J.K. Liu, V.K. Wei, and D.S. Wong. A separable threshold ring signature scheme. *Information Security and Cryptology-ICISC 2003*, pages 12–26, 2004.

[64] Pierre Loidreau. Properties of codes in rank metric. *CoRR*, abs/cs/0610057, 2006.

[65] Pierre Loidreau. *Metrique rang et cryptographie*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2007.

[66] Pierre Loidreau. Designing a rank metric based mceliece cryptosystem. In *Post-Quantum Cryptography*, pages 142–152. Springer, 2010.

[67] F Florence Jessie MacWilliams and NJ Neil James Alexander Sloane. *The Theory of Error-correcting Codes : Part 2*, volume 16. Elsevier, 1977.

[68] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $o(2^{0.0054n})$. In *Advances in Cryptology–ASIACRYPT 2011*, pages 107–124. Springer, 2011.

[69] Robert J McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report*, 42(44) :114–116, 1978.

[70] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-Quantum Cryptography*, pages 147–191. Springer, 2009.

[71] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo SLM Barreto. Mdpc-mceliece : New mceliece variants from moderate density parity-check codes. *IACR Cryptology ePrint Archive*, 2012 :409, 2012.

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

[73] A. V. Ourivski and T. Johansson. New technique for decoding codes in the rank metric and its cryptography applications. *Probl. Inf. Transm.*, 38 :237–246, July 2002.

[74] 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.

[75] AV Ourivski. Recovering a parent code for subcodes of maximal rank distance codes. In *Proc. of WCC*, volume 3, pages 357–363, 2003.

[76] Raphael Overbeck. A new structural attack for gpt and variants. In *Progress in Cryptology–Mycrypt 2005*, pages 50–63. Springer, 2005.

[77] Raphael Overbeck. Extending gibson's attacks on the gpt cryptosystem. In *Coding and Cryptography*, pages 178–188. Springer, 2006.

[78] Raphael Overbeck. Structural attacks for public key cryptosystems based on gabidulin codes. *Journal of cryptology*, 21(2) :280–301, 2008.

[79] D. Pointcheval. Self-scrambling anonymizers. In *Financial Cryptography*, pages 259–275. Springer, 2001.

[80] Haitham Rashwan, Ernst M Gabidulin, and Bahram Honary. A smart approach for gpt cryptosystem based on rank codes. In *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pages 2463–2467. IEEE, 2010.

[81] Haitham Rashwan, Ernst M Gabidulin, and Bahram Honary. Security of the gpt cryptosystem and its applications to cryptography. *Security and Communication Networks*, 4(8) :937–946, 2011.

[82] Haitham Rashwan, Bahram Honary, and Ernst M Gabidulin. On improving security of gpt cryptosystems. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 1110–1114. IEEE, 2009.

[83] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. *Advances in Cryptology-ASIACRYPT 2001*, pages 552–565, 2001.

[84] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[85] K. Sakurai and S. Miyazaki. An anonymous electronic bidding protocol based on a new convertible group signature scheme. In *Information Security and Privacy*, pages 385–399. Springer, 2000.

[86] Nicolas Sendrier. Decoding one out of many. In *PQCrypto*, pages 51–67, 2011.

[87] Adi Shamir. An efficient identification scheme based on permuted kernels (extended abstract). In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 606–609. Springer, 1989.

[88] Adi Shamir. An efficient identification scheme based on permuted kernels. In *Advances in Cryptology—CRYPTO'89 Proceedings*, pages 606–609. Springer, 1990.

[89] Danilo Silva, Frank R Kschischang, and R Kotter. Communication over finite-field matrix channels. *Information Theory, IEEE Transactions on*, 56(3) :1296–1305, 2010.

[90] W. A. Stein et al. *Sage Mathematics Software (Version 3.3)*. The Sage Group, 2009. `http://www.sagemath.org`.

[91] J. Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology-CRYPTO-593*, pages 13–21. Springer, 1994.

[92] J. Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology-CRYPTO-593*, pages 13–21. Springer, 1994.

[93] Jacques Stern. A method for finding codewords of small weight. In *Coding theory and applications*, pages 106–113. Springer, 1989.

[94] Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1993.

[95] Jacques Stern. Designing identification schemes with keys of short size. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 164–173. Springer, 1994.

[96] Pascal Véron. Improved identification schemes based on error-correcting codes. *Appl. Algebra Eng. Commun. Comput.*, 8(1) :57–69, 1996.

[97] D. Zheng, X. Li, and K. Chen. Code-based ring signature scheme. *IJ Network Security*, 5(2) :154–157, 2007.