Etudes de systèmes cryptographiques construits à l'aide de codes correcteurs, en métrique de Hamming et en métrique rang.

Cédric Faure

Thèse de Doctorat

présentée à
L'ÉCOLE POLYTECHNIQUE

pour obtenir le titre de
DOCTEUR EN SCIENCES

Spécialité **Informatique**

soutenue le 17 mars 2009 par

Cédric FAURE

Études de systèmes cryptographiques construits à l'aide de codes correcteurs, en métrique de Hamming et en métrique rang

Jury

Rapporteurs

Pierrick GAUDRY	LORIA
Philippe GABORIT	Université de Limoges

Directeurs de Thèse

Nicolas SENDRIER	INRIA-Rocquencourt projet SECRET
Pierre LOIDREAU	CELAR

Examinateurs

Félix ULMER	Université de Rennes 1
Andreas ENGE	École Polytechnique
Daniel AUGOT	École Polytechnique

Résumé

Ce manuscrit rassemble et résume les résultats des travaux que j'ai effectués durant mes quatre années de thèse à l'ENSTA et à l'INRIA Rocquencourt, sous la direction conjointe de Pierre Loidreau et de Nicolas Sendrier. Le thème général de ma thèse est l'étude des codes correcteurs d'erreurs en vue de leur utilisation en cryptographie. L'intérêt de ce document réside surtout dans la présentation de résultats assez précis (dont certains n'ont pas été publiés), et devrait intéresser particulièrement les spécialistes du domaine. Cependant, tout au long du texte, je me suis efforcé de définir les notions utilisées, de rappeler et d'expliquer les résultats connus, et de détailler mes raisonnements. Ainsi, toute personne disposant d'une formation basique en mathématiques devrait être capable, en lisant ce manuscrit, de saisir l'intérêt des problématiques que je soulève, et de juger de la validité des solutions que j'apporte.

Pour les personnes qui découvrent le domaine, je recommande instamment la lecture des deux chapitres suivants. Dans le premier chapitre, j'introduis de façon très générale les bases de la cryptographie moderne, en particulier la cryptographie à clé publique. Dans le second chapitre, plus mathématique, je définis les objets appelés codes correcteurs, et j'énonce des résultats connus concernant leur utilisation en cryptographie. Ces deux chapitres ne visent en aucun cas à être exhaustifs sur le sujet de l'utilisation des codes correcteurs en cryptographie, mais permettent à un non-spécialiste de visualiser les enjeux de ce domaine de recherche.

Le cryptosystème classique à base de codes correcteurs est le système de McEliece utilisant les codes de Goppa [31]. Comparativement aux systèmes issus d'autres primitives cryptographiques, le chiffrement et le déchiffrement par McEliece sont rapides. De plus, ce système semble immunisé aux attaques qui seraient possibles si l'on disposait d'ordinateurs quantiques. Cependant, ce cryptosystème présente un inconvénient majeur, la taille très importante de sa clé publique. Il semble donc intéressant de trouver des méthodes pour

réduire la taille des clés publiques des cryptosystèmes construits à l'aide de codes correcteurs. Je me suis intéressé à deux axes de recherche existants dans cette direction. Je décris l'utilisation de codes géométriques dans le chapitre trois, et l'utilisation de la métrique rang dans le chapitre quatre.

Reconnaissance des codes géométriques

Une idée pour réduire la taille de la clé publique est d'utiliser des codes correcteurs à taux de correction élevé. On peut alors espérer que pour une taille de clé publique relativement faible, la distance de correction associée soit suffisamment élevée pour décourager les attaques par décodage. On aimerait donc trouver des familles de codes linéaires optimaux qui soient à la fois rapidement décodables et difficilement reconnaissables.

Les codes de Reed-Solomon sont des codes linéaires optimaux (au sens de la borne de Singleton) que l'on sait rapidement décoder, Niederreiter avait donc proposé leur utilisation dans [33]. Mais, en 1992, Sidelnikov et Shestakov ont montré que ces codes sont faciles à reconnaître [43]. Leur attaque rend donc les codes de Reed-Solomon inutilisables comme base pour le système de McEliece.

Les codes géométriques sont des codes d'évaluation de fonctions rationnelles sur des courbes algébriques de genre g. Ils sont quasi-optimaux, car leur distance minimale $d = n - k + 1 - g$ est très proche de la borne de Singleton. De plus, il est possible de les décoder au delà de leur capacité de correction [26], ce qui les rend particulièrement intéressants pour le problème qui nous intéresse. Ces familles de codes paraissaient donc très prometteuses pour la cryptographie. En particulier, l'utilisation des codes géométriques construits sur des courbes de genre 2 a été proposée en 1996 par Janwa et Moreno dans [27]. Dans cet article, il est proposé l'emploi de codes [171,109,61] sur $\mathbb{F}_{2^7}$, pour atteindre une complexité d'attaque de 2^{80}.

Cependant, en 2007, Lorenz Minder a mis au point un algorithme permettant avec une grande probabilité de reconnaître en temps polynomial les codes géométriques de genre 1 [32]. Ensemble, nous avons pu généraliser son approche aux courbes de genre 2, brisant ainsi les paramètres de Janwa et Moreno, puis aux courbes de genre $g \geq 3$. Notre résultat sur le genre 2 a été présenté dans [16], et la généralisation aux genres supérieurs est complètement inédite.

Nos algorithmes sont eux aussiégalement probabilistes, avec grande probabilité de succès. Leur temps d'exécution en genre g est en $O(n^3 g!(\frac{q}{n-k-g/2})^g + q^{g/2})$ opérations sur le corps de base. Comme on peut le voir, la complexité est exponentielle en g, mais polynomiale en les autres paramètres à g fixé. Par conséquent, notre attaque est tout à fait efficace sur des systèmes de genre relativement faible, et entaille sérieusement la sécurité des systèmes de McEliece construits à l'aide de codes géométriques. Notre approche est détaillée et expliquée dans le troisième chapitre.

Le principe est le suivant : Soit $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, (k+g-1)\Delta_0, \mathbf{P})$ un code géométrique $[n,k]$ de genre g. Alors il existe un mot de code non nul $\mathbf{x} \in \mathcal{C}$ tel que $x_1 = \cdots = x_{k+g-1} = 0$ si et seulement si on a, dans la Jacobienne de $\mathcal{X}$:

$$\langle P_1 \rangle + \cdots + \langle P_{k+g-1} \rangle - (k+g-1)\Delta_0 = 0.$$

Autrement dit, chaque mot de code de poids faible correspond à une équation dans la Jacobienne de la courbe. Si l'on connaît suffisamment de mots de poids faible, on peut alors déterminer la structure de la Jacobienne. Dans un second temps, la connaissance de cette structure nous permet d'appliquer l'idée de l'attaque de Sidelnikov et Shestakov : on est capable de générer un couple de mots de code de poids faible dont les supports sont presque identiques. Ensuite, si on suppose connues les coordonnées de quelques points du support, on peut, en comparant nos deux mots de codes, calculer les coordonnées des autres points du support.

Par une procédure de supposition/évaluation/vérification, on peut donc calculer les coordonnées des points du support de notre code, ainsi que l'équation de la courbe $\mathcal{X}$. Il reste quelques subtilités techniques à régler, ce que je détaille dans le chapitre consacré au sujet. Toujours est-il que notre algorithme est capable, à partir d'une clé publique du système, de générer une clé privée permettant de déchiffrer. Il s'agit donc bien d'une attaque cryptographique. L'efficacité de cette attaque semble interdire l'usage des codes géométriques (d'un genre raisonnable) comme alternative aux codes de Goppa dans des systèmes de type McEliece.

Cryptographie en métrique rang

Une autre idée pour réduire la taille des clés publiques des cryptosystèmes à base de codes correcteurs est l'utilisation de la métrique rang. Le

principe est d'adopter une nouvelle définition de la distance, correspondant au rang d'une certaine matrice. Avec cette définition, une erreur de faible rang peut être "étalée" sur toutes les coordonnées d'un mot (le mot $(1 \dots 1)$ est de rang 1, par exemple). Les algorithmes classiques de décodage général en métrique de Hamming utilisent la méthode du type Ensemble d'Information, qui consiste à déplacer une fenêtre sur le mot reçu, et à espérer que l'erreur se situe en dehors (voir par exemple [5] pour plus de précisions). En métrique rang, cette méthode est inapplicable. Les meilleurs algorithmes de décodage général en métrique rang sont ceux d'Ourivski et de Johannson [36], et leur complexité est bien plus élevée que ce que l'on sait faire en métrique de Hamming. En conséquence, on peut espérer construire des codes rang dont la matrice génératrice est petite, mais dont le décodage n'est pas envisageable sans connaissance de la structure algébrique de ces codes.

Néanmoins, la métrique rang a été peu étudiée jusqu'à présent. Quasiment tous les codes efficaces sont des dérivations des codes de Gabidulin [22]. Ces codes sont des codes d'évaluation de polynômes linéaires, optimaux pour la métrique rang, décodables rapidement jusqu'à leur capacité de correction, et présentent de nombreux autres points communs avec les codes de Reed-Solomon. Cependant, on ne sait toujours pas si les codes de Gabidulin sont décodables en liste au-delà de la capacité de correction, par une approche du type Sudan. Un tel résultat changerait tous les paramètres pratiques de cryptographie en métrique rang.

Je me suis donc attaqué au problème, avec peu de succès malheureusement. Cependant, sans arriver à montrer l'existence ou l'inexistence d'algorithmes de décodage en liste en temps polynomial des codes de Gabidulin, j'ai pu établir une borne supérieure du rayon de décodage de tels algorithmes. Mon idée est simple : Supposons qu'un tel algorithme existe, et qu'on lui demande de décoder en liste le mot $\mathbf{y}$ à distance t. Alors l'algorithme devra renvoyer tous les mots de code situés dans une boule de centre $\mathbf{y}$ et de rayon t. Si la taille de cette liste est exponentielle, l'algorithme ne peut pas s'exécuter en temps polynomial. Ainsi, en calculant la taille moyenne des boules de décodage, je montre l'existence d'une borne t_0 au delà de laquelle le décodage en liste ne peut pas être polynomial en moyenne. Il est intéressant de remarquer que, dans le cas de codes de longueur maximale, j'obtiens $t_0 = n - \sqrt{nk}$, soit la même expression que la borne de Johnson concernant les codes de Reed-Solomon.

Au passage, ces calculs de volume montrent également qu'il n'est pas envisageable d'utiliser la métrique rang dans des schémas de signature par une

approche du type Courtois-Finiasz-Sendrier [11]. Tous ces résultats sur les volumes des boules de décodage sont présentés dans la seconde section du quatrième chapitre de ce document.

Le cryptosystème le plus connu en métrique rang est le système GPT, proposé par Gabidulin, Paramonov et Tretjakov dans [24]. Ce système, ainsi que ses variantes, est considéré avec méfiance par la communauté cryptographique, car, par deux fois, une attaque en temps polynomial a été publiée contre ce système ([25] et [38]), imposant à chaque fois une réparation du cryptosystème. Cependant, comme l'a montré Loidreau dans [29], il est tout à fait possible de neutraliser l'attaque d'Overbeck par un choix astucieux des paramètres de conception du système. Le système GPT réparé, que je détaille dans la troisième section du quatrième chapitre, atteint une sécurité cryptographique face à toutes les attaques connues, alors que la taille de sa clé publique reste tout à fait compétitive.

Il existe diverses variantes cryptographiques utilisant la métrique rang. Avec Pierre Loidreau, nous avons mis au point un cryptosystème basé sur le problème de reconstruction de polynômes linéaires. L'idée, qui se base sur des travaux d'Augot et Finiasz en métrique de Hamming [1], est d'inclure de façon visible dans la clé publique la structure du code de Gabidulin utilisé par le système. La matrice génératrice publique est de la forme

$$G_{\text{pub}} = \begin{pmatrix} \mathbf{g} \\ \vdots \\ \mathbf{g}^{[k-1]} \\ \mathbf{K}_1 \\ \vdots \\ \mathbf{K}_u \end{pmatrix}.$$

Puisque le haut de la matrice est généré de façon systématique, on réduit énormément la taille de la clé publique, car celle-ci ne doit contenir que le support $\mathbf{g}$ et les vecteurs $\mathbf{K}_1 \dots \mathbf{K}_u$ (l'entier u vaut typiquement 3 ou 4).

Les vecteurs $\mathbf{K}_i$ ont eux-même une structure particulière, puisqu'ils sont de la forme $\mathbf{K}_i = \text{Tr}(\gamma_i P)(\mathbf{g}) + \text{Tr}(\gamma_i \mathbf{E})$. C'est la connaissance de cette structure qu'utilise le destinataire pour déchiffrer le message. L'utilisation de la métrique rang, en "étalant" les erreurs, empêche de retrouver la structure des $\mathbf{K}_i$ en effectuant des décodages parallèles. Une implémentation que j'ai réalisée en MAGMA montre que les performances du cryptosystème (en temps d'exécution du chiffrement et du déchiffrement) sont tout à fait raisonnables.

Nous avons présenté ce système dans [17] mais les paramètres que nous avions donné à l'époque induisent une vulnérabilité aux attaques d'Overbeck publiées depuis. Dans la quatrième section du quatrième chapitre, je détaille la présentation du cryptosystème, j'analyse sa résistance aux différentes attaques existantes, et je donne des paramètres de sécurité tenant compte de l'attaque Overbeck. La taille de la clé publique ainsi adaptée reste extrêmement faible pour un système basé sur des codes, plus faible encore que GPT et ses variantes.

Bien sûr, la sécurité de ces cryptosystèmes en métrique rang reste sujette à caution, on en sait encore trop peu sur les mécanismes sous-jacents pour pouvoir donner de vrais arguments de sécurité. Néanmoins, il a là une alternative intéressante pour réduire les tailles de clé des systèmes à base de codes correcteurs. Il convient donc d'étudier plus avant ces problèmes, ce dont je me suis efforcé durant ma thèse.

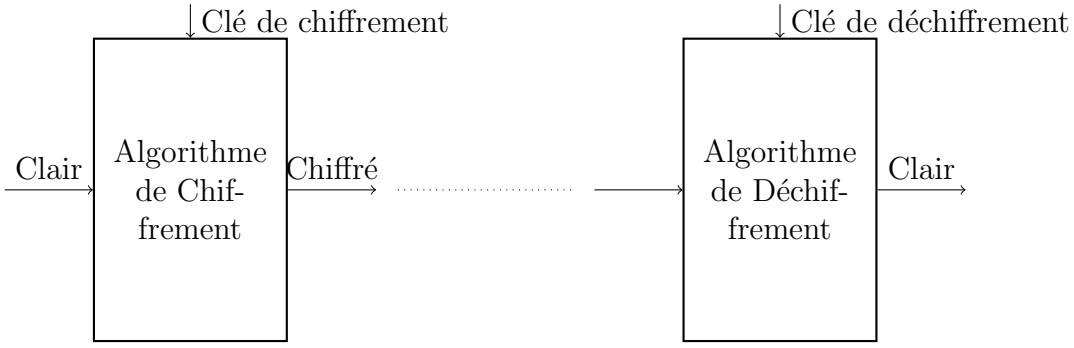
Table des matières

Chapitre 1

Introduction à la cryptographie

La cryptographie est l'ensemble des techniques permettant d'écrire un message de façon brouillée afin que seul son destinataire légitime soit capable de comprendre la teneur du message. En dépit de l'évolution des moyens de communication depuis l'antiquité, il a toujours été difficile de garantir la sécurité du canal par lequel transite un message. Un cavalier transportant un rapport sur la position d'une armée peut être intercepté par des éclaireurs ennemis ; un employé du télégraphe transmettant des consignes d'investissements boursiers peut être à la solde d'un actionnaire concurrent ; un ordre militaire transmis par les ondes radios peut être capté par n'importe quel récepteur réglé sur la bonne fréquence ; un courriel qui transite sur Internet peut être lu par n'importe quel ordinateur sur la chaîne reliant l'expéditeur au destinataire. Aussi, la problématique d'établir des communications sécurisées en utilisant un médium non sécurisé a toujours été d'importance en vue d'applications militaires, financières, ou simplement du respect de la vie privée.

Un cryptosystème fonctionne de la façon suivante :



Traditionnellement, la sécurité du cryptosystème reposait sur la non-

divulgation des algorithmes de chiffrement et de déchiffrement. Mais en 1883, Auguste Kerckhoff énonça comme principe que la sécurité d'un cryptosystème doit reposer uniquement sur le secret des clés, et que les algorithmes doivent être supposés publiquement connus. Cette contrainte semble rendre plus ardue l'invention d'un cryptosystème fiable, mais présente deux grands avantages. Premièrement, en cas de divulgation du secret, la réparation est simple. Si le cryptosystème est intrinsèquement sûr, il suffit de générer de nouvelles clés, et l'on peut reprendre les communications. Deuxièmement, l'inventeur d'un cryptosystème peut, sans trahir de secret, révéler ses algorithmes à ses collègues cryptographes. Ceux-ci soumettront le cryptosystème à de multiples examens, à la recherche de la moindre faille, et proposeront parfois des raffinements ou des améliorations, avant que l'on aboutisse à un cryptosystème considéré comme sûr. Auparavant un art, la cryptographie est ainsi devenue une science.

Le *chiffrement* est la fabrication du message chiffré à partir du clair et de la clé de chiffrement. Le *déchiffrement* est l'extraction du message clair à partir du chiffré en utilisant la clé de déchiffrement. Le *décryptage* ou l'*attaque* est l'extraction du message clair à partir du chiffré sans connaître la clé de déchiffrement. Le mot *cryptage* n'a aucun sens en français. La *cryptanalyse* est l'étude théorique d'un système de chiffrement en vue de mettre au point des algorithmes de décryptage.

Un bon cryptosystème doit permettre un chiffrement et un déchiffrement rapide, tout en interdisant toute possibilité de décryptage. Pour alléger les charges en mémoire informatique, on préfère aussi que les clés soient de petite taille.

Mathématiquement, la sécurité du cryptosystème semble impossible à atteindre. En effet, l'algorithme de déchiffrement est connu de tous, et seule la clé de déchiffrement est secrète. Or, cette clé est de petite taille, un attaquant peut donc en théorie essayer toutes les clés possibles. S'il est capable de les essayer suffisamment vite, il finira par trouver la bonne clé de déchiffrement et aura alors décrypté le message chiffré. On va donc supposer que la puissance de calcul disponible pour l'attaquant n'est pas infinie (cf. tableau suivant).

On définit donc un paramètre pratique de sécurité cryptographique, qui correspond à une limite maximale de la puissance de calcul d'un attaquant raisonnable. Actuellement, on considère que ce paramètre vaut 2^{80}. Cela veut

Nombre d'opérations	Temps d'exécution
2^{30}	$<$ 1 seconde
2^{40}	2 minutes
2^{50}	36 heures
2^{60}	4 ans
2^{70}	4400 ans
2^{80}	4,4 millions d'années

TAB. 1.1 – Temps théorique d'exécution sur un ordinateur récent, avec processeur quadri-core à 2Ghz

dire que si une attaque effectue en moyenne plus de 2^{80} opérations pour décrypter un message, cette attaque ne menace pas la sécurité du cryptosystème car elle n'est à la portée d'aucune organisation humaine. Bien sûr, le nombre 2^{80} est valable à l'heure actuelle, il faudra le modifier dans le futur pour tenir compte de l'amélioration des performances informatiques.

En utilisant cette notion de sécurité cryptographique, l'attaque par recherche exhaustive de la clé de déchiffrement est très vite vouée à l'échec. Avec une clé aussi courte que 80 bits (une suite de 80 chiffres zéro ou un), il existe 2^{80} clés différentes, et il est impossible en pratique de toutes les essayer. Par contre, il est possible qu'un autre algorithme de décryptage attaque plus efficacement le sytème.

Ainsi, pour juger de la validité d'un mécanisme de chiffrement, on va considérer la complexité (i.e. le nombre d'opérations) de l'attaque la plus rapide contre le système. Généralement, on estime que le mécanisme peut être utilisé en cryptographie si la complexité du décryptage croît exponentiellement avec les paramètres du système (taille de la clé, ou du message, par exemple), tandis que la complexité du chiffrement et du déchiffrement croît polynomialement avec ces mêmes paramètres. Comme l'exponentielle croît beaucoup plus vite que tout polynôme, on peut alors espérer que, pour certains paramètres, les attaques deviendront infaisables en pratique, alors que le chiffrement et le déchiffrement resteront rapides.

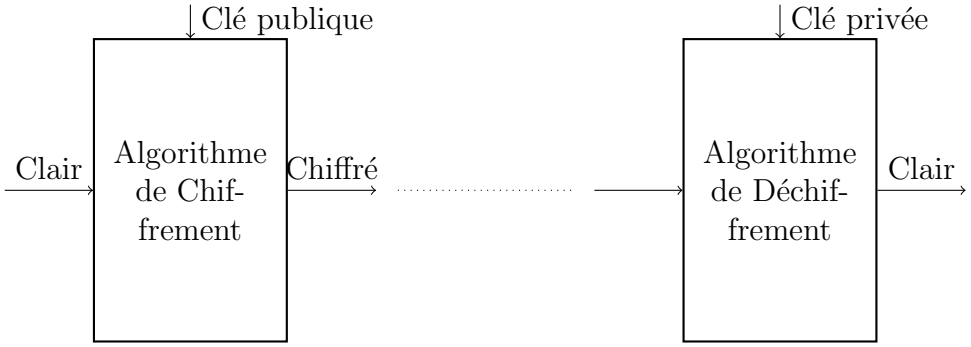
Montrer qu'un cryptosystème n'est pas sûr est relativement facile : il suffit de trouver un algorithme d'attaque suffisamment rapide. Par contre, prouver la sécurité du système est extrêmement difficile. Même en considérant toutes les méthodes d'attaque connues, rien ne prouve qu'il n'existe pas un algorithme inédit, plus rapide que tout ce qui existe. Les arguments de sécurité

reposent toujours, au final, sur des arguments d'autorité : "Cela fait trente ans que toute la communauté essaye d'attaquer ce système, sans succès. Il n'y a donc sans doute pas d'algorithme de décryptage rapide."

On distingue deux types de cryptosystèmes :

– Les cryptosystèmes à clé secrète, ou symétriques : Les clés de chiffrement et de déchiffrement sont identiques. L'expéditeur du message doit donc au préalable partager un secret avec le destinataire. L'avantage est que l'on peut atteindre la sécurité cryptographique en utilisant des clés très courtes (une centaine de bits), et en conservant un chiffrement et un déchiffrement extrêmement rapide.

– Les cryptosystèmes à clé publique, ou asymétriques : Les clés de chiffrement et de déchiffrement sont différentes, et la connaissance de la clé de chiffrement (ou clé publique) ne permet pas de retrouver la clé de déchiffrement (ou clé privée) associée. On considère donc que la clé publique est connue de tous (d'où son nom), et que le sécurité du système repose uniquement sur le secret de la clé privée. L'expéditeur et le destinataire n'ont pas besoin d'avoir échangé de secret pour communiquer de façon chiffrée. Ces cryptosystèmes sont donc très utiles pour initialiser une communication. Malheureusement, ces systèmes sont par ailleurs moins efficaces (clés plus longues, algorithmes plus lents) que les systèmes symétriques.



Les cryptosystèmes à clé publique sont difficile à concevoir. La fonction de chiffrement (qui au clair associe le chiffré) doit être une fonction injective à sens unique (son inversion est difficile) à trappe (la connaissance d'un secret rend l'inversion facile). On connait peu d'objets de cette sorte, c'est pourquoi la plupart des cryptosystèmes asymétriques sont issus de l'un des quatre domaines suivants :

- Arithmétique : L'inversion de la fonction à sens unique se rapporte au problème de factorisation des nombres entiers. Le système RSA utilise ce type de primitive.

- Courbes elliptiques : L'inversion de la fonction à sens unique se rapporte au problème du logarithme discret sur le groupe des points d'une courbe. Par exemple, le protocole de signature électronique ECDSA utilise ce genre de constructions.

- Théorie des réseaux (lattices) : L'inversion de la fonction à sens unique se rapporte au problème du calcul d'un vecteur de longueur minimale dans un réseau. Les algorithmes de chiffrement et de signature NTRU reposent sur la théorie des réseaux.

- Théorie des codes correcteurs : L'inversion de la fonction à sens unique se rapporte au problème du décodage d'un code aléatoire. Plusieurs exemples seront étudiés par la suite, dont le schéma de chiffrement de McEliece.

Tous les cryptosystèmes décrits dans ce document sont des systèmes asymétriques issus de la théorie des codes.

Chapitre 2

Notions de théorie des codes

2.1 Introduction

Les codes correcteurs d'erreurs ont été développés dans la deuxième moitié du vingtième siècle, suite aux travaux de Shannon [41]. L'objectif est alors de pouvoir établir des communications claires (sans parasites, sans brouillage). Plutôt que d'essayer sans cesse d'améliorer physiquement les systèmes de transmission, Shannon a l'idée d'une autre approche : Faire subir au signal un traitement informatique après sa réception afin de détecter et de corriger les erreurs de transmission. C'est le début de la digitalisation de l'information.

Le principe est simple, il s'agit d'ajouter de la redondance dans le message transmis. On obtient ainsi un message plus long, mais dont l'information excédentaire peut être exploitée pour détecter voire corriger des erreurs de transmission. Les codes correcteurs sont utilisés ainsi dans toutes les télécommunications, mais aussi dans le stockage de données (gravure de CD-rom, par exemple) ou encore dans les identifiants personnels (numéro de sécurité social, de comptes bancaires, etc.).

Le domaine de la théorie des codes a donc connu un très fort développement ces soixante dernières années. Ce développement a donné naissance à des analyses mathématiques et algorithmiques poussées des opérations possibles sur les codes correcteurs. Notamment, il semble clair que le décodage d'un code aléatoire est un problème algorithmique difficile.

Pour mettre au point un système cryptographique, il faut un problème algorithmique difficile. McEliece a présenté dans [31] l'idée d'utiliser le décodage comme problème difficile sous-jacent à un système cryptographique.

Ainsi, les codes correcteurs d'erreurs, destinés à l'origine à clarifier des messages transmis, peuvent être utilisés au contraire pour les chiffrer.

On ne s'intéressera dans ce chapitre qu'aux codes linéaires corrigeant des erreurs en métrique de Hamming. Les codes corrigeant des erreurs en métrique rang seront étudiés dans un chapitre ultérieur.

2.2 Définitions

2.2.1 Codes linéaires et métrique de Hamming

Soit $\mathbb{F}_q$ le corps fini à q éléments. Soit n un entier (la longueur du code), et k un autre entier (la dimension du code), avec $k \leq n$. On se place sur $\mathbb{F}_q^n$, l'ensemble des mots de la forme $\mathbf{x} = (x_1, \ldots, x_n)$.

Si f est une fonction de $\mathbb{F}_q$ dans $\mathbb{F}_q$, on notera $f(\mathbf{x})$ le mot $(f(x_1), \ldots, f(x_n))$.

Définition 1 (Distance de Hamming)**.** *La* distance de Hamming *entre deux mots* $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ *est le nombre de positions sur lesquelles les deux mots diffèrent :*

$$d(\mathbf{x}, \mathbf{y}) = \#\{i \in [1, n] | x_i \neq y_i\}.$$

Il s'agit bien d'une distance, l'inégalité triangulaire en particulier est vérifiée :

$$d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}).$$

Définition 2 (Poids de Hamming)**.** *Le* poids de Hamming *d'un mot* $\mathbf{x} \in \mathbb{F}_q^n$ *est simplement sa distance au mot nul :*

$$|\mathbf{x}| = d(\mathbf{x}, \mathbf{0}) = \#\{i \in [1, n] | x_i \neq 0\}.$$

Définition 3 (Code linéaire)**.** *Un* code linéaire $[n, k]$ *sur l'alphabet* $\mathbb{F}_q$ *est un espace vectoriel* $\mathcal{C} \in \mathbb{F}_q^n$ *de dimension* k *sur* $\mathbb{F}_q$*. Son* taux de transmission *est* $\frac{k}{n}$.

Définition 4 (Distance minimale)**.** *La* distance minimale *du code* $\mathcal{C}$ *est simplement le minimum de la distance entre deux mots de code différents. Sa* capacité de correction théorique *est définie à partir de cette distance minimale :*

$$d = \min_{\mathbf{x},\mathbf{y}\in\mathcal{C},\mathbf{x}\neq\mathbf{y}} d(\mathbf{x}, \mathbf{y}).$$

$$t_{cor} = \left\lfloor \frac{d-1}{2} \right\rfloor.$$

Si $\mathcal{C}$ est un code linéaire $[n, k]$ de distance minimale d, on note que $\mathcal{C}$ est un code linéaire $[n, k, d]$.

Proposition 1. *Dans le cas d'un code linéaire, si* $\mathbf{x}$ *et* $\mathbf{y}$ *sont deux mots du code* $\mathcal{C}$*, alors le mot* $\mathbf{x} - \mathbf{y}$ *également. Par conséquent, la distance minimale du code linéaire* $\mathcal{C}$ *est également son poids minimum :*

$$d = \min_{\mathbf{x}\in\mathcal{C},\mathbf{x}\neq\mathbf{0}} d(\mathbf{x}, \mathbf{0}) = \min_{\mathbf{x}\in\mathcal{C},\mathbf{x}\neq\mathbf{0}} |\mathbf{x}|.$$

Proposition 2 (Unicité du décodage)**.** *Soit* $\mathcal{C}$ *un code linéaire de capacité de correction théorique* t_{cor}*. Soit* $\mathbf{x} \in \mathbb{F}_q^n$ *un mot quelconque. Alors il existe au plus un mot de code* $\mathbf{c} \in \mathcal{C}$ *tel que* $d(\mathbf{x}, \mathbf{c}) \leq t_{cor}$.

Quand on utilise les codes pour la correction d'erreurs, la capacité de correction théorique est donc le nombre d'erreurs que le code permet de corriger à coup sûr. On essaie alors de maximiser cette capacité, à taux de transmission fixé. Le théorème suivant nous renseigne sur ce qu'il est possible de faire :

Théorème 1 (Singleton)**.** *Soit* $\mathcal{C}$ *un code linéaire* $[n, k, d]$*. Alors* $\mathcal{C}$ *vérifie l'inégalité de Singleton :*

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

Définition 5 (Maximum Distance Separable)**.** *Soit $\mathcal{C}$ un code linéaire $[n, k, d]$. Le code $\mathcal{C}$ est dit* maximum distance separable *(MDS) s'il atteint la borne de Singleton, i.e. si :*

$$k + d = n + 1.$$

L'optimalité du code est un paramètre important de son efficacité en terme de décodage, mais ce n'est pas le seul. Une distance minimale élevée garantit que le code permet théoriquement de corriger beaucoup d'erreurs (par la proposition 2), mais pas que cette correction est algorithmiquement possible. La vitesse des algorithmes de décodage est une autre qualité fondamentale du code.

2.3 Matrice génératrice, matrice de parité

Soit $\mathcal{C}$ un code linéaire $[n, k]$ sur $\mathbb{F}_q$.

Définition 6 (Matrice génératrice)**.** *Soient $\mathbf{y}_1, \ldots, \mathbf{y}_k$ une base de $\mathcal{C}$. En considérant les mots $\mathbf{y}_1, \ldots, \mathbf{y}_k$ comme les lignes d'une matrice $k \times n$ sur $\mathbb{F}_q$, on obtient une* matrice génératrice *G du code $\mathcal{C}$. Puisqu'un espace vectoriel admet plusieurs bases, la matrice génératrice d'un code linéaire n'est pas unique, loin de là. Par contre, la matrice génératrice d'un code $[n, k]$ est toujours une matrice $k \times n$ de rang k.*

Proposition 3. *Une matrice génératrice est suffisante pour définir le code linéaire associé. En effet, tous les mots de code peuvent être générés grâce à cette matrice :*

$$\mathcal{C} = \{\mathbf{x}G | \mathbf{x} \in \mathbb{F}_q^k\}.$$

Un code linéaire peut aussi être caractérisé par une autre matrice, la matrice de parité :

Définition 7 (Matrice de parité)**.** *Si $\mathcal{C}$ est un espace linéaire de dimension k sur $\mathbb{F}_q^n$, alors il existe une matrice H à $n-k$ lignes et n colonnes telle que $\ker(H) = \mathcal{C}$. Autrement dit, H est une matrice $(n-k) \times n$ sur $\mathbb{F}_q$, de rang $n-k$, telle que*

$$\forall \mathbf{y} \in \mathbb{F}_q^n, H\mathbf{y}^T \iff \mathbf{y} \in \mathcal{C}.$$

Une autre façon encore de l'exprimer, est de dire que chacune des $n-k$ lignes de la matrice de parité correspond à une des équations linéaires caractérisant l'espace vectoriel $\mathcal{C}$.

Comme précédemment, un code linéaire admet de nombreuses matrices de parité, mais une seule d'entre elles suffit à parfaitement définir le code associé.

Proposition 4. *Si G et H sont respectivement une matrice génératrice et une matrice de parité d'un même code linéaire, alors on a :*

$$GH^T = 0.$$

La réciproque est vraie si les deux matrices sont de rang maximal.

Corollaire 1. *Étant donné une matrice génératrice G d'un code linéaire $[n, k]$, on peut calculer une matrice de parité de ce code par un simple pivot de Gauss, en $O(n^3)$ opérations dans $\mathbb{F}_q$ (mais cette complexité peut être améliorée par l'utilisation d'algorithmes d'algèbre linéaire plus évolués). Le problème inverse (calculer une matrice génératrice à partir d'une matrice de parité) est strictement équivalent.*

Définition 8 (Code dual)**.** *Soit H une matrice de parité du code linéaire $[n, k]$ $\mathcal{C}$. Alors, comme H est de rang maximal, on peut la considérer comme la matrice génératrice d'un code linéaire. Le code ainsi généré, de longueur n et de dimension $n - k$, est indépendant du choix de H. On l'appelle* code dual *de $\mathcal{C}$, et on le note $\mathcal{C}^\perp$.*

Proposition 5. *Le code dual vérifie les propriétés suivantes :*

- $\forall G \in \mathcal{M}_{k\times n}(\mathbb{F}_q)$, *$G$ matrice génératrice de $\mathcal{C} \Leftrightarrow$ G matrice de parité de $\mathcal{C}^\perp$.*
- $\forall H \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_q)$, *$H$ matrice de parité de $\mathcal{C} \Leftrightarrow$ H matrice génératrice de $\mathcal{C}^\perp$.*
- $\mathcal{C}^{\perp\perp} = \mathcal{C}$.

2.4 Résolution algorithmique du décodage

Ce qui nous intéresse maintenant, c'est la possibilité ou non de mettre en place des algorithmes de décodage d'un code linéaire, et la complexité de tels algorithmes. Nous allons étudier ceci dans cette partie. On considère ici $\mathcal{C}$, un code linéaire $[n, k, d]$ sur $\mathbb{F}_q$ (connu par sa matrice génératrice), un mot $\mathbf{y} \in \mathbb{F}_q^n$, et un entier t.

2.4.1 Formalisation

Voici les formalisations classiques des problèmes de décodage les plus courants :

Décodage au maximum de vraisemblance$(\mathcal{C}, \mathbf{y})$
Trouver un mot $\mathbf{c} \in \mathcal{C}$ *vérifiant* $d(\mathbf{c}, \mathbf{y}) = d(\mathcal{C}, \mathbf{y})$.

Décodage borné$(\mathcal{C}, \mathbf{y}, t)$
Trouver, s'il existe, un mot $\mathbf{c} \in \mathcal{C}$ *vérifiant* $d(\mathbf{x}, \mathbf{y}) \leq t$.

Décodage en liste$(\mathcal{C}, \mathbf{y}, t)$
Trouver l'ensemble des mots $\mathbf{c} \in \mathcal{C}$ *vérifiant* $d(\mathbf{c}, \mathbf{y}) \leq t$.

Résoudre le décodage au maximum de vraisemblance revient, à partir d'un mot reçu $\mathbf{y}$ pouvant contenir des erreurs, à retrouver le mot de code $\mathbf{x} \in \mathcal{C}$ le plus proche au sens de Hamming du mot $\mathbf{y}$, c'est-à-dire le mot qui a été le plus probablement envoyé par l'expéditeur (dans un modèle de canal symétrique). Le décodage en liste revient à établir un ensemble de candidats réalistes pour le mot envoyé.

Il est clair que le problème de décodage borné est plus simple que chacun des deux autres. Cependant, si $t < d/2$, alors Décodage borné$(\mathcal{C}, \mathbf{y}, t)$ et Décodage en liste$(\mathcal{C}, \mathbf{y}, t)$ sont équivalents. On ne s'intéresse donc au décodage en liste que pour des distances excédant la capacité de correction du code.

Nous allons maintenant étudier deux algorithmes permettant de résoudre de façon générale le problème de décodage d'un code linéaire.

2.4.2 Décodage par tableau standard

Définition 9 (Syndrome)**.** *Soit* H *une matrice de parité du code* $\mathcal{C}$*, et* $\mathbf{y} \in \mathbb{F}_q^n$ *un mot quelconque. Le syndrome de* $\mathbf{y}$ *associé à* H *est le vecteur* $H\mathbf{y}^T$*, de longueur* $(n - k)$.

Soit H une matrice de parité du code $\mathcal{C}$.
Si on a $\mathbf{y} = \mathbf{c} + \mathbf{e}$ avec $\mathbf{c} \in \mathcal{C}$, alors

$$H\mathbf{y}^T = H\mathbf{c}^T + H\mathbf{e}^T,$$
$$H\mathbf{y}^T = H\mathbf{e}^T.$$

Par conséquent, si $\mathbf{c}$ est une solution du problème de décodage borné, on peut calculer $\mathbf{c}$ à l'aide d'un mot de poids faible de même syndrome que $\mathbf{y}$.

Algorithme 1 : Décodage par tableau standard

Input : $G, \mathbf{y} = (y_1, \ldots, y_n), t$
Output : $\mathbf{x} \in \mathcal{C}$ tel que $d(\mathbf{x}, \mathbf{y}) \leq t$

Précalculs : *Calculer une matrice de parité H associée à la matrice génératrice G.*
Pour chaque mot $\mathbf{e} \in \mathbb{F}_q^n$ de poids inférieur ou égal à t, calculer $H\mathbf{e}^T$ et l'ajouter à la liste des syndromes L.

Décodage : *Calculer $H\mathbf{y}^T$, et rechercher dans la liste L un mot $\mathbf{e}$ admettant le même syndrome.*
Si $\mathbf{e}$ existe, renvoyer $\mathbf{c} = \mathbf{y} - \mathbf{e}$
Sinon, renvoyer "pas de solution"

Cet algorithme doit cependant calculer et manipuler une liste L contenant les syndromes de tous les mots de poids de Hamming inférieur à t. Il existe au moins $C_n^t(q-1)^t$ mots d'un tel poids, la complexité de l'algorithme est donc exponentielle en t.

2.4.3 Décodage par Ensemble d'Information

On peut améliorer la complexité du décodage, en utilisant un algorithme de Décodage par Ensemble d'Information.

Définition 10 (Ensemble d'Information)**.** *Soit $\mathcal{C}$ un code linéaire $[n, k]$ sur $\mathbb{F}_q$, et G une matrice génératrice de ce code. Un ensemble d'indices $I \subset [1, n]$ est appelé ensemble d'information pour le code $\mathcal{C}$ si I est de cardinal k, et que la restriction de G aux colonnes d'indices $i \in I$ forme une matrice inversible. Cette définition est indépendante du choix de G.*

Proposition 6. *Si $[1, k]$ est un ensemble d'information du code linéaire $\mathcal{C}$, alors il existe une unique matrice génératrice de $\mathcal{C}$ de la forme $(I_k|R)$, et une unique matrice de parité de $\mathcal{C}$ de la forme $(Q|I_{n-k})$. De plus, ces matrices vérifient $Q = -R^T$. On peut calculer ces deux matrices à partir de G en*

temps $O(n^3)$ par un pivot de Gauss.

Proposition 7. *Si $I = \{i_1, \ldots, i_k\}$ est un ensemble d'information du code linéaire $\mathcal{C}$, et $(x_1, \ldots, x_k)$ un vecteur quelconque de $\mathbb{F}_q^k$, alors il existe un unique mot de code $\mathbf{c} \in \mathcal{C}$ tel que pour tout $j \leq k$, $c_{i_j} = x_j$. De plus, on peut facilement calculer $\mathbf{c}$ à partir d'une matrice génératrice de $\mathcal{C}$, par une simple inversion matricielle.*

Algorithme 2 : Décodage par Ensemble d'Information

Input : $G, \mathbf{y} = (y_1, \ldots, y_n), t$
Output : $\mathbf{x} \in \mathcal{C}$ tel que $d(\mathbf{x}, \mathbf{y}) \leq t$

Interpolation : *Choisir un sous-ensemble $I \subset [1, n]$ de cardinal k.*
Construire l'unique mot de code $\mathbf{c} \in \mathcal{C}$ tel que pour tout $i \in I$, $c_i = y_i$.
Si $d(\mathbf{c}, \mathbf{y}) \leq t$, renvoyer $\mathbf{c}$.
Sinon, recommencer avec un nouveau I.

Si l'on a $\mathbf{y} = \mathbf{c} + \mathbf{e}$, avec $\mathbf{c} \in \mathcal{C}$ et $|\mathbf{e}| = t$, alors l'interpolation de $\mathbf{y}$ sur I produira le mot de code $\mathbf{c}$ si et seulement si $\mathbf{e}$ est nul sur tous les éléments de I. Cela se produit avec une probabilité $\frac{C_{n-k}^t}{C_n^t} = \frac{C_{n-t}^k}{C_n^k}$.

Chaque étape d'interpolation coûte grossièrement $O(n^3)$ opérations. Donc, si les ensembles d'informations sont choisis au hasard, la complexité moyenne de l'agorithme vaut $O(n^3 \frac{C_n^t}{C_{n-k}^t})$. Cet algorithme est donc exponentiel, mais bien plus efficace que le Décodage par tableau standard.

Il y a moyen d'améliorer l'algorithme par un choix astucieux des ensembles d'informations, en utilisant des algorithmes plus performants d'algèbre linéaire, ou en autorisant un nombre faible d'erreurs à se trouver sur l'ensemble d'information. Pour plus de détails, on pourra se référer à [46], et [5]. Cependant, pour des tailles cryptographiques classiques, ces améliorations changent peu la complexité de l'algorithme.

2.4.4 Algorithmes de décodage plus rapides

Si le décodage d'un code linéaire donné avait toujours une complexité exponentielle, on ne pourrait pas utiliser efficacement les codes pour faire de la correction d'erreurs. Heureusement, pour certaines familles de codes, on peut

écrire la matrice génératrice de façon à ce qu'elle vérifie certaines propriétés, et le décodage peut alors être accéléré.

Par exemple les codes de Reed-Solomon (cf. chapitre suivant) sont tels que leur matrice génératrice peut s'écrire sous forme d'une matrice de Vandermonde. En utilisant cette structure, on est alors en mesure de les décoder en $O(n^2)$ opérations [30]. De plus, les codes de Reed-Solomon sont MDS (cf. définition 5). Ces codes sont donc très utiles pour implémenter la correction d'un signal.

En pratique, on utilise aussi d'autres codes (LDPC, Turbocodes, etc.) qui sont encore plus rapides à décoder que les Reed-Solomon, au prix d'une légère perte d'optimalité (et donc de capacité de correction).

On sait également construire d'autres familles de codes dont le décodage peut être effectué en temps polynomial (souvent cubique), à condition toujours que la matrice génératrice soit écrite de façon à mettre en avant la structure du code. Ces codes, comme les codes de Goppa, ou bien les codes géométriques (cf. chapitre suivant), ont des performances trop faibles pour être utilisés en correction du signal, mais ont un intérêt en cryptographie.

Cependant, pour un code qui ne présente pas de structure particulière (ou dont on ne sait pas identifier la structure), on ne connaît pas d'algorithme plus rapide que le Décodage par Ensemble d'Information. Puisque la complexité de cet algorithme est exponentielle, cela motive l'utilisation des problèmes de décodage comme primitive cryptographique.

2.5 Utilisation des codes correcteurs en cryptographie

2.5.1 Cryptosystème de McEliece

L'idée du cryptosystème de McEliece (présenté dans [31]) est d'utiliser la difficulté du décodage d'un code aléatoire comme primitive de cryptographie à clé publique. Pour déchiffrer le message envoyé, il faut résoudre un problème de décodage sur un code dont la structure est masquée. Cependant, le destinataire légitime connaît cette structure, et utilise cette connaissance pour déchiffrer.

On se place sur le corps $\mathbb{F}_q$.

- **Génération des clés :** On se donne $G_0 \in \mathcal{M}_{k\times n}(\mathbb{F}_q)$, matrice génératrice d'un code $\mathcal{C}_0$ dont on sait algorithmiquement décoder t erreurs. On génère S, une matrice inversible $k \times k$ sur $\mathbb{F}_q$, et P, une matrice de permutation $n \times n$. On calcule $G_{\text{pub}} = SG_0P$, matrice $k \times n$ à coefficients dans $\mathbb{F}_q$.
  La clé publique est le couple (G_{pub}, t).
  La clé secrète est sa décomposition (S, G_0, P, t).

- **Chiffrement :** Pour chiffrer le message $\mathbf{m} \in \mathbb{F}_q^k$, on génère aléatoirement un vecteur $\mathbf{e} \in \mathbb{F}_q^n$ de poids de Hamming t. Le chiffré est alors le vecteur $\mathbf{c} = \mathbf{m}G_{\text{pub}} + \mathbf{e}$.

- **Déchiffrement :** Le vecteur $\mathbf{c}P^{-1}$ est à distance au plus t du code $\mathcal{C}_0$. Notre algorithme de décodage permet donc de retrouver le vecteur $\mathbf{m}SG_0 \in \mathcal{C}_0$. On en déduit $\mathbf{m}$ par inversion matricielle.

2.5.2 Variante de Niederreiter

En 1986, Niederreiter proposa dans [33], apparemment sans connaître les travaux de McEliece, un autre cryptosystème à base de codes :

- **Génération des clés :** On se donne $H_0 \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_q)$, matrice de parité d'un code $\mathcal{C}_0$ dont on sait algorithmiquement décoder t erreurs. On génère S, une matrice inversible $(n-k) \times (n-k)$ sur $\mathbb{F}_q$, et P, une matrice de permutation $n \times n$. On calcule $H_{\text{pub}} = SH_0P$, matrice $(n-k) \times n$ à coefficients dans $\mathbb{F}_q$.
  La clé publique est le couple (H_{pub}, t).
  La clé secrète est sa décomposition (S, H_0, P, t).

- **Chiffrement :** L'ensemble des messages est ici l'ensemble des mots de longueur n et de poids de Hamming au plus t. Pour un message $\mathbf{m}$ dans cet ensemble, le chiffré est le vecteur $\mathbf{c} = H_{\text{pub}}\mathbf{m}^T$.

- **Déchiffrement :** On choisit une solution arbitraire $\mathbf{x}$ du système $H_{\text{pub}}\mathbf{x}^T = \mathbf{c}$. On sait alors que $\mathbf{x}$ est à distance au plus t du code $\mathcal{C}$. On utilise donc la structure connue du code pour décoder $\mathbf{x}$ en $\mathbf{y} \in \mathcal{C}$. On retourne ensuite $\mathbf{x} - \mathbf{y}$.

Comme on peut le voir, les systèmes de McEliece et de Niederreiter sont très proches, l'un utilisant les matrices génératrice, et l'autre, les matrices de parité. En fait, il a été montré dans [28] que la sécurité des deux systèmes est équivalente. Toute attaque structurelle sur l'un se traduit par une attaque structurelle sur l'autre. La variante de Niederreiter présente cependant deux avantages :

- Le système de Niederreiter provoque asymptotiquement une expansion moins importante du message. L'expansion du message peut cependant également être évitée dans le système de McEliece.

- Le système de McEliece est vulnérable à une attaque par réémission de message : Si le même message est chiffré deux fois, les deux chiffrés seront distants d'au plus $2t$. On peut alors trouver facilement les positions d'erreurs en comparant ces deux chiffrés. La variante de Niederreiter n'est pas affectée par ce problème.

Malgré ces légères différences, puisque les sécurités des deux systèmes sont équivalentes, nous allons nous limiter par la suite à l'étude du système de McEliece.

2.5.3 Remarques générales

Par rapport à des cryptosystèmes à clé publique issus d'autres primitives, le système de McEliece présente un avantage et deux inconvénients :

Le chiffrement et le déchiffrement correspondent à des opérations classiques de codage et de décodage. Or on connaît des algorithmes très rapides pour effectuer ces deux types d'opérations. Le cryptosytème de McEliece peut donc être implémenté efficacement, pour un système asymétrique.

Cependant, le message chiffré subit une *expansion*. Pour un clair de k caractères, on envoie un chiffré de $n > k$ caractères. Néanmoins, cette expansion n'est pas très gênante en pratique. De plus, comme cela a été montré dans [4], il est possible d'y remédier en incluant une partie du message dans le vecteur d'erreur $\mathbf{e}$.

La clé publique est une matrice, elle occupe donc $nk\log_2(q)$ bits de mémoire. Pour des valeurs raisonnable de n et k, cette clé publique risque de devenir très grosse, comparativement à ce que l'on sait faire avec d'autres

systèmes asymétriques. C'est probablement, en bonne partie, à cause de la taille de sa clé publique que le cryptosystème de McEliece n'est pas utilisé en pratique.

2.5.4 Sécurité

Un attaquant connaît G_{pub} sans connaitre sa décomposition, et connait le chiffré $\mathbf{c}$. Son but est de trouver une décomposition

$$\mathbf{c} = \mathbf{m}G_{\text{pub}} + \mathbf{e},$$

c'est-à-dire de résoudre le problème Décodage borné$(\mathcal{C}, \mathbf{c}, t)$ où $\mathcal{C}$ est le code engendré par G_{pub}.

La sécurité du système repose sur deux suppositions :

Premièrement, que le code $\mathcal{C}$ est difficile à décoder au rayon t, ce qui est le cas par exemple si ce code ne peut pas être algorithmiquement distingué d'un code aléatoire.

Deuxièmement, que la structure du code a bien été masquée, c'est-à-dire qu'il est difficile de retrouver les matrices S, G_0, P à partir de G_{pub}.

L'attaquant peut donc tenter d'attaquer le système par deux méthodes : Ou bien il tente directement de décoder le chiffré $\mathbf{c}$ dans le code $\mathcal{C}$ (attaque sur le chiffré), ou bien il essaye de retrouver une décomposition valide de la clé $G_{\text{pub}} = S'G'_0P'$, qui lui permet de mettre au point son propre algorithme de décodage (attaque sur la clé).

La résistance du système à ces deux méthodes d'attaque dépend intrinsèquement de la famille de codes dans laquelle est choisie G_0. Le choix de cette famille est le point délicat dans la conception du cryptosystème. En effet, on veut utiliser une famille de codes que l'on sait décoder (et si possible rapidement), ce qui impose que ces codes soient structurés. D'un autre coté, des codes trop structurés prêtent le flanc à des attaques tirant parti de cette structure. Il faut donc atteindre un compromis difficile.

La famille des codes de Goppa est actuellement la meilleure candidate pour satisfaire ces deux contraintes. La famille des codes géométriques semblait prometteuse, mais nous verrons dans le chapitre suivant que ces codes trop structurés rendent le cryptosystème vulnérable à une attaque sur la clé.

2.5.5 Paramètres pratiques

Il est supposé dans [11] que les codes de Goppa sont indistinguables de codes aléatoires, et cette hypothèse n'a jamais été infirmée depuis. Ce sont donc de bons candidats pour implémenter le système de McEliece, car ils résistent à toutes les attaques structurelles. La meilleure attaque est encore de décoder le chiffré par un algorithme du type Ensemble d'Information. La complexité d'une telle attaque est approximativement $O(n^3 \frac{C_n^t}{C_{n-k}^t})$.

Malheureusement, les codes de Goppa sont loin d'être optimaux, on a $t \ll \frac{n-k}{2}$. Aussi, pour obtenir une distance de décodage suffisante pour neutraliser l'attaque par Ensemble d'Information, il faut une clé publique de taille importante.

Des paramètres du cryptosystème correspondant à une sécurité de 2^{80} sont donnés dans [6] ou [20]. On peut par exemple choisir $n = 2048$, $k = 1685$, $t = 33$. Cela donne des blocs message de 1685 bits, un taux de transmission de 0.82, et une clé publique d'environ 3Mbits (que l'on peut éventuellement réduire à 600 kbits). Pour comparaison, une clé RSA de 1024 bits atteint le même paramètre de sécurité.

Les codes de Reed-Solomon sont optimaux, et permettraient donc de résister aussi bien à un décodage par Ensemble d'Information, pour une clé de taille beaucoup plus faible. Cependant, ils ne sont pas utilisables, car ils sont soumis à une attaque structurelle en temps polynomial (cf. [43] ou le chapitre suivant).

Les codes géométriques auraient également permis de réduire la taille de la clé par leur quasi-optimalité (Janwa et Moreno proposent l'utilisation d'une clé de 13 kbits dans [27]), mais ils sont aussi victimes d'une attaque structurelle rapide, comme nous le montrerons dans le chapitre suivant.

2.6 Conclusion

La sécurité des cryptosystèmes à base de codes est intrinsèquement liée à la famille de codes utilisée dans la conception du système. La version de McEliece utilisant des codes de Goppa est étudiée depuis 30 ans, et elle semble parfaitement sûre d'un point de vue cryptographique. Cependant, elle n'est pas utilisée en pratique, principalement car la taille de sa clé publique est

beaucoup plus importante que ce que l'on sait faire avec des systèmes issus d'autres domaines.

Si l'on veut diversifier notre panel de primitives cryptographiques, il faut rendre les systèmes asymétriques basés sur des codes compétitifs vis-à-vis d'autres cryptosystèmes. Pour celà, il est important de trouver un moyen de réduire la taille de leur clé publique.

Une méthode pour atteindre ce but est de trouver une famille de codes décodables et presque optimaux pour implémenter McEliece. Les codes géométriques satisfont ces conditions et je les présenterai dans le chapitre suivant. Cependant, comme on le verra, leur trop forte structure les rend trop facilement reconnaissables.

Une autre méthode est de neutraliser la possibilité d'attaquer le système par des ensembles d'information. Comme cette attaque est générale à la métrique utilisée, il faut pour cela sortir de la métrique de Hamming, et considérer une autre définition de la distance. La métrique rang paraît prometteuse en ce sens, c'est pourquoi je la présenterai dans le quatrième chapitre.

Chapitre 3

Codes géométriques et applications

3.1 Introduction

Les codes géométriques sont des codes d'évaluation de fonctions rationnelles sur des courbes algébriques. Cette famille de codes hautement structurés, inclue et généralise les codes de Reed-Solomon, en étendant certaines de leurs propriétés. Notamment, les codes géométriques sont des codes linéaires presque MDS que l'on sait décoder relativement rapidement jusqu'à de bons paramètres de correction. Depuis les travaux de Guruswami et Sudan ([26]), on est même capable de décoder les codes géométriques en temps polynomial au delà de leur capacité de correction.

Dans le domaine de la correction d'erreurs, les codes géométriques sont plutôt moins intéressants que les codes de Reed-Solomon. Leurs performances (en terme de taux de correction et taux de transmission) sont moins bonnes, et l'utilisation de courbes algébriques rend leur programmation compliquée. L'avantage des codes géométriques est la possibilité de construire des codes plus long que la taille de l'alphabet, en choisissant une courbe algébrique contenant beaucoup de points.

Cependant, en cryptographie, les codes géométriques présentent un grand intérêt. En effet, pour réduire la taille de la clé publique d'un système de type McEliece, une solution est d'utiliser une famille de codes la plus optimale possible. Or, depuis l'attaque de Sidelnikov et Shestakov sur les codes de Reed-Solomon d'une part [43], et l'attaque de Sendrier sur les codes concaténés d'autre part [40], toutes les familles de codes optimaux que l'on sait décoder

rapidement sont soumises à des attaques structurelles. Il est donc logique de se tourner vers des codes quasi-optimaux, comme les codes géométriques. L'usage des codes géométriques sur des courbes de genre deux a été proposée pour l'implémentation de McEliece dans [27], où les auteurs annoncent une complexité d'attaque de 2^{80} en utilisant des codes [171,109,61] sur $\mathbb{F}_{2^7}$.

Nous sommes néanmoins en mesure de montrer que de tels codes sont aussi soumis à des attaques structurelles efficaces. En 2007, L. Minder a montré dans [32] que l'on pouvait retrouver en temps polynomial la structure des codes géométriques construits sur des courbes de genre 1, et a réalisé une implémentation efficace de son attaque. Nous avons ensuite étendu ce résultat aux courbes de genre 2, attaquant ainsi les paramètres de Janwa et Moreno [27]. Je peux aussi montrer que notre méthode d'attaque reste utilisable tant que le genre g est faible, la complexité de l'attaque devenant alors $O(n^3 g!(\frac{q}{n-k-g/2})^g + q^{g/2})$ opérations sur le corps de base. Ces résultats n'ont pas encore été publiés, l'algorithme en genre 2 a simplement été présenté dans [16].

Nos algorithmes sont probabilistes, avec une très forte probabilité de succès, pour peu que le code utilisé vérifie quelques propriétés. Notamment, nous utilisons le fait que la longueur n du code est proche du cardinal de la courbe. Cette hypothèse ne nous semble pas audacieuse, car un code raccourci diminuerait fortement les performances du système. Nos autres hypothèses paraissent encore plus raisonnables.

L'idée de base de nos algorithmes est, comme dans l'approche de L. Minder pour le genre 1, de déduire des équations dans la Jacobienne de la courbe, en examinant le support des mots de poids faible du code. Si l'on dispose de suffisamment de mots de poids faible, on génère assez d'équations pour connaitre complètement la structure de la Jacobienne, et pour calculer l'image des points du support d'évaluation dans cette Jacobienne.

La deuxième étape de notre algorithme reprend l'idée de l'attaque de Sidelnikov et Shestakov. En utilisant notre connaissance de la Jacobienne, nous sommes capables de générer deux mots de code, de poids faible et de supports quasiment identiques. La comparaison de ces deux mots nous fournit des équations sur les abscisses des points d'évaluation du code. Ces équations nous permettent, en fixant arbitrairement certaines abscisses, de toutes les déduire. Les ordonnées des points d'évaluation sont calculées à partir des abscisses, en réalisant des interpolations de faible degré. L'utilisation d'isomorphismes de courbe nous épargne un examen exhaustif de tous les choix

possibles pour les coordonnées arbitrairement fixées, on peut se contenter pour cette étape de $O(n^2)$ essais différents.

Une fois ces deux étapes menées à bien, la reconstruction de la courbe et du code attaqué ne présentent absolument aucune difficulté. Nous disposons alors d'un algorithme rapide de décodage du code engendré par la clé publique de McEliece, nous avons donc structurellement attaqué avec succès la clé publique du système.

Dans ce chapitre, je présente dans un premier temps les outils mathématiques liés à la géométrie algébrique, j'expose ensuite le problème qui nous intéresse en remarquant quelques propriétés utiles à sa résolution. Je rappelle ensuite brièvement les algorithmes résolvant ce problème en genres 0 et 1, puis je présente notre solution, d'abord concernant le genre 2, et enfin dans le cas général.

3.2 Introduction à la géométrie algébrique

3.2.1 Courbes algébriques

Soit $\mathbb{F}_q$ un corps fini, et $\overline{\mathbb{F}_q}$ sa cloture algébrique. Sur $(\overline{\mathbb{F}_q}^3)^* = \overline{\mathbb{F}_q}^3 - \{(0,0,0)\}$, on définit la relation d'équivalence $\sim$ par :

$$(X,Y,Z) \sim (X',Y',Z') \Leftrightarrow \exists \alpha \neq 0, (X',Y',Z') = (\alpha X, \alpha Y, \alpha Z).$$

On peut alors définir le plan projectif $\mathbb{P}_2(\overline{\mathbb{F}_q})$ comme l'espace quotient $(\overline{\mathbb{F}_q}^3)^*/\sim$. Le plan projectif $\mathbb{P}_2(\overline{\mathbb{F}_q})$ peut aussi être vu comme l'ensemble des droites vectorielles de l'espace $\overline{\mathbb{F}_q}^3$.

Lorsque l'on travaille sur le plan projectif, il est souvent pratique de l'assimiler au plan affine $\overline{\mathbb{F}_q}^2$ complété de la droite à l'infini. À une classe d'équivalence du plan projectif, on associe le point (x,y) du plan affine tel que $(x,y,1)$ soit un représentant de la classe. Cependant, un tel point n'existe pas toujours. Les classes qui n'admettent pas un tel représentant sont appelés points à l'infini. Ainsi, le plan projectif est assimilable à la réunion du plan affine et de l'ensemble des points à l'infini.

Définition 11 (Polynômes homogènes)**.** *Un polynôme homogène de degré d est un polynôme dont chaque monôme est de la forme $\alpha X^k Y^l Z^{d-k-l}$. Tous*

ses monômes sont de degrés d (en sommant les degrés en X, Y et Z).

Puisque l'on assimile souvent le plan projectif au plan affine, il est pratique d'associer à un polynôme homogène en (X, Y, Z) un polynôme non homogène en (x, y). Au monôme $\alpha X^k Y^l Z^{d-k-l}$ est ainsi associé le monôme $\alpha x^k y^l$. Ce processus simple (remplacer X par x, Y par y et Z par 1) est appelé deshomogénéisation, le procédé inverse est appelé homogénéisation.

Un polynôme en X, Y, Z ne peut pas être évalué sur $\mathbb{P}_2(\overline{\mathbb{F}_q})$. Cependant, si le polynôme est homogène, l'ensemble de ses zéros est parfaitement défini sur $\mathbb{P}_2(\overline{\mathbb{F}_q})$.

Définition 12 (Courbe algébrique)**.** *Une courbe algébrique $\mathcal{X} \subset \mathbb{P}_2(\overline{\mathbb{F}_q})$ est l'ensemble des zéros d'un polynôme homogène que l'on notera $Eq(\mathcal{X})$.*

Définition 13 (Points rationnels d'une courbe)**.** *Si $\mathcal{X}$ est une courbe algébrique, on note $\mathcal{X}(\mathbb{F}_q)$ l'ensemble des points rationnels de la courbe, c'est à dire l'ensemble des points de la courbe qui admettent un représentant (X, Y, Z) dont les trois coordonnées sont dans $\mathbb{F}_q$ (et non dans une extension).*

$$\mathcal{X}(\mathbb{F}_q) = \{(X, Y, Z) \in \mathcal{X} | \exists \alpha \in \overline{\mathbb{F}_q}^*, (\alpha X, \alpha Y, \alpha Z) \in (\mathbb{F}_q^3)^*\}.$$

Par la suite, on supposera que l'équation déshomogénéisée de la courbe $\mathcal{X}$ est de la forme :

$$y^2 + G(x)y = F(x), \tag{3.1}$$

avec F et G polynômes à coefficients dans $\mathbb{F}_q$, F unitaire de degré impair $2g + 1$, G de degré au plus g, et $g \geq 1$. On suppose de plus que le polynôme $4F(x) + G^2(x)$ n'admet pas de racine double.

La courbe $\mathcal{X}$ est alors régulière dans sa partie affine. Elle admet un unique point à l'infini, de coordonnées projectives $(0, 1, 0)$, que l'on note $\mathcal{O}$. Pour tout $x_0 \in \overline{\mathbb{F}_q}$, la courbe $\mathcal{X}$ passe par exactement deux points affines d'abscisse x_0. Ces deux points, appelés points opposés, ont pour coordonnées (x_0, y_0) et $(x_0, -y_0 - G(x_0))$, et peuvent éventuellement être confondus. Si un point est noté P, son opposé est généralement noté $-P$. Si un point $P \in \mathcal{X}$ est rationnel, alors son abscisse est sur le corps de base, et $-P$ est également rationnel.

Cependant, les points de la courbe $\mathcal{X}$ d'abscisse $x \in \mathbb{F}_q$ peuvent être tous deux non rationnels.

L'entier g est appelé le genre de la courbe. Le genre de la courbe a une définition plus générale que celle-ci, en fonction des propriétés géométriques de cette courbe. Notre définition du genre, bien que plus restrictive, sera suffisante pour ce travail, à une exception près : Les courbes de genre 0 existent, ce sont les droites (lorsque, comme ici, nous nous plaçons sur un corps fini). Par la suite, si l'on suppose que $\mathcal{X}$ est de genre 0, alors elle vérifie l'équation d'une droite. Sinon, on supposera qu'elle vérifie l'équation ci-dessus. Les courbes de genre 1 sont appelées courbes elliptiques, les courbes de genre 2 et plus vérifiant cette équation sont appelées courbes hyperelliptiques.

3.2.2 Fonctions rationnelles

On souhaite maintenant définir sur $\mathcal{X}$ des fonctions à valeurs dans $\overline{\mathbb{F}_q}$. Comme on s'en est aperçu auparavant, si P est un polynôme, P n'induit pas une fonction sur $\mathbb{P}_2(\overline{\mathbb{F}_q})$, et pas non plus sur $\mathcal{X}$. Cependant, si P et Q sont deux polynômes homogènes de même degré d, on constate que la valeur $\frac{P}{Q}(X, Y, Z)$ est indépendante du choix du représentant d'une classe donnée de $\mathbb{P}_2(\overline{\mathbb{F}_q})$. C'est cette idée qui nous conduira à la définition des fonctions rationnelles. On veut cependant s'assurer que deux fonctions rationnelles dont l'évaluation est identique sur la courbe $\mathcal{X}$ sont algébriquement égales. On va donc tout d'abord quotienter les polynômes homogènes par l'équation de la courbe $\mathcal{X}$.

Définition 14. *On note $\Gamma_h(\mathcal{X})$ l'ensemble des polynômes homogènes quotienté par l'idéal engendré par $Eq(\mathcal{X})$. Autrement dit, deux polynômes homogènes sont égaux dans $\Gamma_h(\mathcal{X})$ si leur différence est divisible par le polynôme homogène $Eq(\mathcal{X})$.*

Définition 15 (Fonction rationnelle)**.** *L'ensemble des fonctions rationnelles sur la courbe $\mathcal{X}$ est défini comme un corps de fractions à partir de $\Gamma_h(\mathcal{X})$.*

$$\overline{\mathbb{F}_q}(\mathcal{X}) = \left\{ \frac{Q}{R} \;\middle|\; \begin{array}{c} Q, R \in \Gamma_h(\mathcal{X}), R \neq 0, \\ \deg(Q) = \deg(R), \end{array} \right\} \bigcup \{0\}.$$

Si $f \in \overline{\mathbb{F}_q}(\mathcal{X})$ peut être exprimée à partir de polynômes homogènes dont tous les coefficients sont dans le corps de base $\mathbb{F}_q$, on note $f \in \mathbb{F}_q(\mathcal{X})$.

Proposition 8. *Les ensembles $\mathbb{F}_q(\mathcal{X})$ et $\overline{\mathbb{F}_q}(\mathcal{X})$ sont des corps, et on a $\mathbb{F}_q \subset \mathbb{F}_q(\mathcal{X}) \subset \overline{\mathbb{F}_q}(\mathcal{X})$.*

Définition 16 (Ordre d'une fonction rationnelle)**.** *On peut définir l'ordre d'une fonction rationnelle $f \in \overline{\mathbb{F}_q}(\mathcal{X})$ en un point de la courbe $P \in \mathcal{X}(\mathbb{F}_q)$ de la façon dont on le définirait sur les fractions rationnelles classiques : Si $f = \frac{Q}{R}$, alors*

$$ord_P\left(\frac{Q}{R}\right) = mult_P(Q) - mult_P(R),$$

où $mult_P(Q)$ et $mult_P(R)$ sont les multiplicités (éventuellement nulles) du point P comme racine des polynômes Q et R.

Proposition 9.

$$\text{Basiquement, on a :} \quad \begin{array}{ll} ord_P(f) > 0 & si\ f(P) = 0, \\ ord_P(f) < 0 & si\ f(P) = \infty, \\ ord_P(f) = 0 & sinon. \end{array}$$

L'ordre d'une fonction rationnelle en un point vérifie des propriétés similaires à celles des multiplicités, en particulier :

$$\begin{aligned} ord_P(f.g) &= ord_P(f) + ord_P(g), \\ ord_P(f+g) &\geq \min(ord_P(f), ord_P(g)). \end{aligned}$$

Enfin, si $f \neq 0$, l'ensemble des points $P \in \mathcal{X}$ tels que $ord_P(f) \neq 0$ est fini.

Théorème 2. *Si $f \in \overline{\mathbb{F}_q}(\mathcal{X})$ est une fonction rationnelle non nulle, alors la somme de ses ordres sur tous les points de la courbe est nulle, en tenant compte des points à l'infini :*

$$\sum_{P \in \mathcal{X}} ord_P(f) = 0. \tag{3.2}$$

Cependant, cette dernière propriété n'est en général pas vraie si l'on restreint la somme aux points rationnels de la courbe, et ce même si $f \in \mathbb{F}_q(\mathcal{X})$.

3.2.3 Groupe des diviseurs

Sauf mention contraire, les preuves des résultats concernant les groupes des diviseurs et le théorème de Riemann-Roch peuvent être trouvés par exemple dans [49].

Définition 17 (Diviseur)**.** *Un diviseur Δ sur une courbe $\mathcal{X}$ est une somme formelle finie de points (rationnels ou non), avec des coefficients entiers, positifs ou négatifs :*

$$\Delta = \sum_{P \in \mathcal{X}} n_P \langle P \rangle, n_P \in \mathbb{Z},$$

où $n_P = 0$ pour tout $P \in \mathcal{X}$, à l'exception d'un ensemble fini que l'on appelle le support :

$$supp(\Delta) = \{P \in \mathcal{X} | n_P \neq 0\}.$$

Le diviseur Δ est dit rationnel s'il est invariant par l'action du groupe de Galois $Gal(\overline{\mathbb{F}_q}/\mathbb{F}_q)$. C'est le cas en particulier si tous les points de son support sont rationnels.

Par la suite, on notera $\langle P \rangle$ le diviseur composé d'une fois le point P, pour le différencier du point lui-même.

Proposition 10. *L'ensemble des diviseurs sur la courbe $\mathcal{X}$, que l'on notera $Div(\mathcal{X})$, forme un groupe abélien pour la loi additive.*

Définition 18 (Degré d'un diviseur)**.** *On définit le degré d'un diviseur comme la somme des coefficients associés à chacun de ses points :*

$$\deg(\Delta) = \sum_{P \in \mathcal{X}} n_P.$$

Définition 19 (Diviseur d'une fonction rationnelle)**.** *À une fonction rationnelle $f \in \overline{\mathbb{F}_q}(\mathcal{X}) - \{0\}$, on associe un diviseur :*

$$div(f) \stackrel{def}{=} \sum_{P \in \mathcal{X}} ord_P(f) \langle P \rangle.$$

Proposition 11. *On peut aisément vérifier, pour toutes fonctions rationnelles f, g non nulles :*

$$\begin{aligned} div(fg) &= div(f) + div(g), \\ div(1/f) &= -div(f). \end{aligned}$$

Définition 20 (Diviseur principal)**.** *Un diviseur est dit principal s'il correspond à une fonction rationnelle. L'ensemble des diviseurs principaux forme un sous-groupe de $Div(\mathcal{X})$. Deux diviseurs sont dits équivalents si leur différence est un diviseur principal.*

D'après le théorème 2, tout diviseur principal est de degré 0. La réciproque n'est vraie qu'en genre 0, cependant.

Définition 21. *On définit le groupe des classes de diviseurs de degré 0 de la courbe $\mathcal{X}$:*

$$Pic^0(\mathcal{X}) = Diviseurs\ de\ degré\ 0 / Diviseurs\ principaux$$

On s'intéresse plus particulièrement à l'ensemble des classes admettant un diviseur rationnel pour représentant. On appellera cet ensemble la Jacobienne de la courbe :

Définition 22 (Jacobienne de la courbe)**.**

$$\mathrm{Jac}(\mathcal{X}(\mathbb{F}_q)) = \{Classes\ de\ Pic^0(\mathcal{X})\ dont\ un\ des\ représentants\ est\ rationnel\}$$

Théorème 3 (Hasse-Weil généralisé)**.** *Pour une courbe de genre $g \geq 1$ sur $\mathbb{F}_q$, la Jacobienne est un groupe fini dont on peut encadrer le cardinal :*

$$(\sqrt{q} - 1)^{2g} \leq \#\mathrm{Jac}(\mathcal{X}(\mathbb{F}_q)) \leq (\sqrt{q} + 1)^{2g} \tag{3.3}$$

Corollaire 2 (Nombre de points rationnels)**.** *Si $\mathcal{X}$ est une courbe de genre g sur $\mathbb{F}_q$, on dispose d'un encadrement du nombre de points rationnels de cette courbe :*

$$q + 1 - 2g\sqrt{q} \leq |\mathcal{X}(\mathbb{F}_q)| \leq q + 1 + 2g\sqrt{q} \tag{3.4}$$

Théorème 4 (Structure de groupe de la Jacobienne)**.** *Si $\mathcal{X}$ est une courbe de genre $g \geq 1$ sur $\mathbb{F}_q$, il existe un isomorphisme de groupes :*

$$\mathrm{Jac}(\mathcal{X}(\mathbb{F}_q)) \simeq \frac{\mathbb{Z}}{d_1\mathbb{Z}} \times \cdots \times \frac{\mathbb{Z}}{d_{2g}\mathbb{Z}},$$

avec $d_1 | \ldots | d_{2g}$ et $d_1 | (q-1)$.

Remarque 1. *Par le théorème de Hasse-Weil généralisé, on obtient un encadrement des d_i du théorème précédent, soit :*

$$(\sqrt{q}-1)^{2g} \leq \prod_{i=1}^{2g} d_i \leq (\sqrt{q}+1)^{2g} \tag{3.5}$$

Proposition 12 (Complexité de l'addition dans la Jacobienne)**.** *Connaissant l'équation de $\mathcal{X}$, l'addition de deux éléments de $\mathrm{Jac}(\mathcal{X}(\mathbb{F}_q))$ peut être effectuée en $O(g^2)$ opérations dans $\mathbb{F}_q$.*

Pour des algorithmes effectuant ces additions, on pourra, par exemple, se rapporter à [7] ou bien encore à [9].

3.2.4 Diviseurs de fonctions usuelles

Soit $\mathcal{X}$ une courbe algébrique de genre $g \geq 1$, d'équation $y^2 + G(x)y = F(x)$. On s'intéresse aux diviseurs de fonctions rationnelles usuelles, écrites sous forme deshomogénéisée :

- $\mathrm{div}(cte) = 0$ si $cte \neq 0$. La réciproque est vraie, une fonction rationnelle de diviseur nul est une fonction constante non nulle.

- $\mathrm{div}(x-a) = \langle P \rangle + \langle -P \rangle - 2\langle \mathcal{O} \rangle$, avec $P \in \mathcal{X}$ un point d'abscisse a. Notons que, même avec $a \in \mathbb{F}_q$, le point P n'est pas forcément rationnel.

- $\mathrm{div}((x-a)(x-b)) = \langle P \rangle + \langle -P \rangle + \langle Q \rangle + \langle -Q \rangle - 4\langle \mathcal{O} \rangle$, avec $P \in \mathcal{X}$ un point d'abscisse a, et $Q \in \mathcal{X}$ un point d'abscisse b. Il est intéressant de constater que cette formule peut être utilisée avec $a, b \in \overline{\mathbb{F}_q}$ pour calculer le diviseur d'un polynôme irréductible sur $\mathbb{F}_q$.

- $\mathrm{div}(\frac{1}{x-a}) = 2\langle\mathcal{O}\rangle - \langle P\rangle - \langle -P\rangle$.
- $\mathrm{div}(y-a) = \langle P_1\rangle + \cdots + \langle P_{2g+1}\rangle - (2g+1)\mathcal{O}$, où P_i est le point de $\mathcal{X}$ de coordonnées (x_i, a). Les abscisses $x_1, \ldots, x_{2g+1} \in \overline{\mathbb{F}_q}$ sont les racines du polynôme $F(x) - aG(x) - a^2$.
- $\mathrm{div}(y^2) = \mathrm{div}(F(x) - yG(x))$.
- $\mathrm{ord}_P(X^\alpha Y^\beta) \geq 0$ si $P \neq \mathcal{O}$.
- $\mathrm{ord}_\mathcal{O}(X^\alpha Y^\beta) = -2\alpha - (2g+1)\beta$.

3.2.5 Théorème de Riemann-Roch

Définition 23 (Relation d'ordre sur les diviseurs)**.** *On définit une relation d'ordre partielle sur Div($\mathcal{X}$) :*

$$\sum_{P\in\mathcal{X}} n_P\langle P\rangle \leq \sum_{P\in\mathcal{X}} m_P\langle P\rangle \Longleftrightarrow \forall P \in \mathcal{X}, n_P \leq m_P.$$

Définition 24 (Espace associé à un diviseur)**.** *On peut ensuite définir l'espace linéaire associé à un diviseur $\Delta \in Div(\mathcal{X})$:*

$$\mathcal{L}(\Delta) = \{f \in \mathbb{F}_q(\mathcal{X}) | div(f) + \Delta \geq 0\} \cup \{0\}.$$

Proposition 13. *Si Δ est un diviseur de $\mathcal{X}$, alors $\mathcal{L}(\Delta)$ est un $\mathbb{F}_q$ espace vectoriel de dimension finie.*

Le théorème de Riemann-Roch nous renseigne sur la dimension de l'espace linéaire associé à un diviseur :

Théorème 5 (Riemann-Roch)**.** *Soit $\mathcal{X}$ une courbe algébrique de genre g, et $\Delta \in Div(\mathcal{X})$ un diviseur sur cette courbe.*
Si $\deg(\Delta) < 0$*, alors dim($\mathcal{L}(\Delta)$) = 0. Sinon,*

$$dim(\mathcal{L}(\Delta)) \geq \deg(\Delta) - g + 1 \tag{3.6}$$

De plus, il y a égalité si $\deg(\Delta) \geq 2g - 1$.

Une preuve du théorème de Riemann-Roch pourra être trouvée, par exemple, dans [49].

Proposition 14. *À partir de l'équation de la courbe $\mathcal{X}$, et de l'expression du diviseur Δ, on peut générer une base de $\mathcal{L}(\Delta)$ en $O((\deg(\Delta)+g)^3)$ opérations.*

Un algorithme effectif pour construire l'espace $\mathcal{L}(\Delta)$ peut être trouvé, par exemple, dans [3].

3.2.6 Définition des codes géométriques

Soit $\mathcal{X}$ une courbe hyperelliptique de genre g définie sur $\mathbb{F}_q$. Soit $\Delta \in \mathrm{Div}(\mathcal{X})$ tel que $\deg(\Delta) \geq 2g-1$. Soit $\mathbf{P} = (P_1, \ldots, P_n) \in \mathcal{X} - \mathrm{supp}(\Delta)$ un n-uplet de points distincts[1]. Alors on définit le code géométrique de diviseur Δ évalué en $(P_1, \ldots, P_n)$ comme un code d'évaluation de $\mathcal{L}(\Delta)$:

$$\mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}) = \{(f(P_1), \ldots, f(P_n)) | f \in \mathcal{L}(\Delta)\} \tag{3.7}$$

Théorème 6. *L'ensemble* $\mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P})$ *est un code linéaire* $[n, \deg(\Delta) - g + 1]$ *de distance minimale* $d \geq n - \deg(\Delta)$.

Preuve . *Cet ensemble est clairement un code linéaire de longueur n. Sa dimension est la même que $\mathcal{L}(\Delta)$, c'est à dire exactement* $\deg(\Delta) - g + 1$ *d'après le théorème de Riemann-Roch (théorème 5). Quelle est sa distance minimale ?*

Soit $\mathbf{y}$ *un mot de code admettant $t < n$ positions nulles. On suppose qu'il s'agit des positions $y_1, \ldots, y_t$. Par définition du code, il existe $f \in \mathcal{L}(\Delta)$ telle que $y_i = f(P_i)$. On a donc* $\mathrm{div}(f) \geq -\Delta$, *et $f(P_1) = 0$, d'où* $ord_{P_1}(f) \geq 1$. *Comme P_1 n'apparait pas dans l'expression de Δ, on en déduit* $\mathrm{div}(f) \geq -\Delta + \langle P_1 \rangle$. *Par un raisonnement similaire, on obtient* $\mathrm{div}(f) \geq -\Delta + \sum_{i \leq t} \langle P_i \rangle$, *d'où l'on déduit* $\deg(\mathrm{div}(f)) \geq -\deg(\Delta) + t$. *Or f est une fonction rationnelle non nulle, donc* $\deg(\mathrm{div}(f)) = 0$, *d'où* $t \leq \deg(\Delta)$. *Un mot de code non nul contient au plus* $\deg(\Delta)$ *zéros, donc la distance minimale d du code vérifie* $d \geq n - \deg(\Delta)$, *d'où* $d \geq n - k + 1 - g$.$\diamondsuit$

[1] Par abus de notation ici aussi, $\mathbf{P}$ désignera toujours la suite de points $(P_1, \ldots, P_n)$; si f est une fonction, $f(\mathbf{P})$ désignera la suite $(f(P_1), \ldots, f(P_n))$, etc.

Remarque 2. *Les codes géométriques construits sur une courbe de petit genre sont des codes linéaires presque MDS. En effet, un code de longueur n et de dimension k vérifie $n + 1 - g \leq k + d$. Ainsi, le défaut du code par rapport à la borne de Singleton est au plus g.*

Corollaire 3. *Les codes géométriques de genre 0 (ou codes de Reed-Solomon généralisés) sont des codes linéaires MDS.*

3.2.7 Multiplication directionnelle

Définition 25. *Soit $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P})$ un code géométrique. Soit $\mathbf{c} = (c_1, \ldots, c_n) \in (\mathbb{F}_q^*)^n$ un n-uplet de coefficients non nuls. On définit le code $\mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}, \mathbf{c})$ comme une multiplication directionnelle du code $\mathcal{C}$:*

$$(c_1 y_1, \ldots, c_n y_n) \in \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}, \mathbf{c}) \overset{def}{\Longleftrightarrow} (y_1, \ldots, y_n) \in \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}).$$

On serait tenté d'appeler ces objets des codes géométriques généralisés, mais un examen attentif révèle que la multiplication directionnelle d'un code géométrique reste un code géométrique (simplement avec un diviseur différent) :

Proposition 15. *Soit $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P})$ un code géométrique, et soit $\mathcal{C}' = \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}, \mathbf{c})$ une multiplication directionnelle de ce code. Alors il existe un diviseur Δ', équivalent à Δ, tel que $\mathcal{C}' = \mathrm{AGC}(\mathcal{X}, \Delta', \mathbf{P})$.*

Preuve . *Soit $P \in \mathbb{F}_q(\mathcal{X})$ un point rationnel de la courbe, extérieur aux P_i. On construit une fonction rationnelle φ sur $\mathcal{X}$ telle que $\varphi(P_i) = c_i$ pour tout i, et telle que P soit le seul pôle (éventuellement multiple) de φ.*
Pour cela, si $P = \mathcal{O}$, on interpole les (P_i, c_i) par un polynôme. Si P est un point affine de coordonnées (x_0, y_0), on appelle P^- son opposé, de coordonnées (x_0, y_0^-). Pour $m \in \mathbb{N}$, on pose $L_k(X)$ le polynôme unitaire de degré m tel que $(X - x_0)^m | (L(X) + y_0^-)$. Alors la fonction rationnelle $f_m(x, y) = \frac{L(x)+y}{(x-x_0)^m}$ admet P pour unique pôle. On construit φ comme combinaison linéaire des f_m.
Maintenant que φ est construite, on pose $\Delta' = \Delta + \mathrm{div}(\varphi)$. Alors, comme les c_i sont non nuls, aucun des P_i n'apparait dans l'expression de Δ', et on vérifie aisément que $\mathcal{C}' = \mathrm{AGC}(\mathcal{X}, \Delta', (P_1, \ldots, P_n))$. $\diamondsuit$

La réciproque est vraie également, les codes géométriques définis sur un même support d'évaluation par deux diviseurs équivalents sont identiques à une multiplication directionnelle près :

Proposition 16. *Soient* $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P})$ *et* $\mathcal{C}' = \mathrm{AGC}(\mathcal{X}, \Delta', \mathbf{P})$ *deux codes géométriques tels que* Δ *et* Δ' *sont équivalents (i.e. leur différence est un diviseur principal). Alors il existe* $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{F}_q^*$ *tels que* $\mathcal{C}' = \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}, \mathbf{c})$.

Preuve . *Par définition de l'équivalence, il existe une fonction rationnelle* φ *telle que* $\Delta' = \Delta + \mathrm{div}(\varphi)$. *Du fait de l'existence des codes* $\mathcal{C}$ *et* $\mathcal{C}'$, *la fonction* φ *n'admet ni pole ni zéro sur l'ensemble* $(P_1, \ldots, P_n)$. *On peut donc poser* $c_i = \varphi(P_i)$, *et vérifier que* $\mathcal{C}' = \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}, \mathbf{c})$. $\diamondsuit$

Ainsi, la multiplication directionnelle correspond simplement à un choix différent du diviseur au sein de la même classe d'équivalence, mais cette notation sera utile par la suite dans nos attaques.

3.2.8 Décodage classique

Ce qui rend les codes géométriques utilisables en pratique, c'est qu'il existe des algorithmes permettant de les décoder jusqu'à la capacité de correction. Par exemple, des méthodes du type Berlekamp-Welch (valides sur les codes d'évaluation) peuvent être appliquées :

Soit $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \ldots, P_n))$ un code géométrique de dimension k sur une courbe de genre g, soit $\mathbf{y} = (y_1, \ldots, y_n)$ un mot reçu à distance au plus τ du code $\mathcal{C}$. On cherche une fonction $f \in \mathcal{L}(\Delta)$, et un mot $\mathbf{e} = (e_1, \ldots, e_n)$ de poids au plus τ tels que :

$$\forall i \leq n,\ y_i = f(P_i) + e_i. \tag{3.8}$$

La première idée est de rechercher une fonction associée aux positions d'erreurs, plutôt qu'un vecteur d'erreur. Soit $V \in \overline{\mathbb{F}}(\mathcal{X})$ une fonction rationnelle s'annulant sur tous les P_i tels que $e_i \neq 0$. Alors, on aura :

$$\forall i \leq n,\ y_i V(P_i) = f(P_i)V(P_i). \tag{3.9}$$

La seconde idée est de linéariser cette équation en posant $N = f \times V$, on doit alors résoudre

$$\forall i \leq n,\ y_i V(P_i) = N(P_i), \tag{3.10}$$

dont les inconnues sont les fonctions V et N. Si on impose l'appartenance de V et N à des espaces linéaires de dimension finie (souvent des ensembles de polynômes de degré majoré), alors on obtient un système linéaire à n équations. Ce système est plus général que notre équation de base, il faut donc extraire de l'ensemble des solutions (V, N) celle qui correspond à une solution $(f, \mathbf{e})$ valide.

Lemme 1. *Soit $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \ldots, P_n))$ un code géométrique, soit $\mathbf{y} = (y_1, \ldots, y_n)$ un mot reçu, soit $\tau \leq n$ un entier naturel, et soit Δ' un diviseur de $\mathcal{X}$ de degré $n - \tau - 1$ ne contenant aucun des P_i. Alors pour tout $(N, V) \in \mathcal{L}(\Delta') \times \mathcal{L}(\Delta' - \Delta)$ vérifiant $\mathbf{y}V(\mathbf{P}) = N(\mathbf{P})$, et pour tout $f \in \mathcal{L}(\Delta)$ vérifiant $\|\mathbf{y} - f(\mathbf{P})\| \leq \tau$, on a $N = V \times f$.*

Preuve . *Soit N, V, f vérifiant les conditions ci-dessus. Alors $N - Vf \in \mathcal{L}(\Delta')$. Or Δ' est de degré $n - \tau - 1$, donc si $N - Vf$ est non nulle, $N - Vf$ admet au plus $n - \tau - 1$ zéros en dehors du support de Δ'. Or, pour tout i tel que $y_i = f(P_i)$, on a $(N - Vf)(P_i) = 0$. La fonction $N - Vf$ admet donc au moins $n - \tau$ zéros sur l'ensemble des (P_i), par conséquent elle est nulle.* $\diamondsuit$

Algorithme 3 : Berlekamp-Welch (codes géométriques)

Input : $\mathbf{y} = (y_1, \ldots, y_n), \mathcal{X}, \Delta, \mathbf{P}, \tau$
Output : $f \in \mathcal{L}(\Delta)$ vérifiant $\|\mathbf{y} - f(\mathbf{P})\| \leq \tau$

Précalculs : *Choisir $\Delta' \in Div(\mathcal{X})$ de degré $n - \tau - 1$.*
Calculer une base de $\mathcal{L}(\Delta')$ et de $\mathcal{L}(\Delta' - \Delta)$.

Linéarisation : *Trouver une solution non triviale*
$(N, V) \in \mathcal{L}(\Delta') \times \mathcal{L}(\Delta' - \Delta)$ au système linéaire $\mathbf{y}V(\mathbf{P}) = N(\mathbf{P})$.
Si pas de solution non nulle, échec de l'algorithme.

Division : *Calculer $f = N/V$, et vérifier si $\|\mathbf{y} - f(\mathbf{P})\| \leq \tau$.*
Si oui, renvoyer f. Si non, renvoyer "pas de solution".

Proposition 17. *Si on choisit $\tau < \frac{n - \deg(\Delta)}{2} - g$ (ie., par rapport à la distance minimale supposée $\tau < \frac{d}{2} - g$), alors l'algorithme précédent renvoie en $O(n^3)$ opérations l'unique mot de code (s'il existe) à distance au plus τ du mot $\mathbf{y}$ reçu.*

Preuve . *D'après le théorème 5, les espaces vectoriels* $\mathcal{L}(\Delta')$ *et* $\mathcal{L}(\Delta' - \Delta)$ *sont de dimensions respectives au moins* $n - \tau - g$ *et* $n - \tau - g - \deg(\Delta)$. *Le système linéaire 3.10 admet donc* n *équations, et* $2n - 2\tau - 2g - \deg(\Delta) > n$ *inconnues. Par conséquent, ce système admet une solution* (N, V) *non triviale, et l'algorithme réussit.* $\diamondsuit$

Cet algorithme n'est pas idéal, car il ne permet pas de décoder à coup sûr jusqu'à la capacité de correction du code. Il existe des algorithmes de décodage par syndrome qui atteignent la capacité de correction ([19] ou [39]), tout en améliorant la vitesse, mais je ne les détaillerai pas ici.

3.2.9 Décodage en liste

Il est possible de corriger au delà de la capacité de correction. Évidemment, dans ce cas, on ne peut pas garantir l'unicité de la solution, l'algorithme renvoie donc une liste de tous les mots de code suffisamment proche du mot reçu, ce type de décodage s'appelle donc le décodage en liste. Les codes géométriques ont suffisamment de structure pour que le décodage en liste soit possible en temps polynomial, par un algorithme de type Sudan.

Lors du décodage classique, on commençait par résoudre une équation de la forme $YV(P) = N(P)$ sur l'ensemble des couples (P_i, y_i). L'idée de l'algorithme de Sudan est d'augmenter le nombre de degrés de liberté de ce système en ajoutant des termes de degrés 2, 3 et plus en Y . Cet algorithme est présenté juste après, et les lemmes suivants justifient sa correction :

Lemme 2. *Si on fixe* $\tau < n - \sqrt{2kn} - g$, *alors il est certain que la phase d'interpolation de l'algorithme 4 fournit un polynôme* Q *valide.*

Lemme 3. *Si* $f \in \mathcal{L}(\Delta)$ *vérifie* $\|\mathbf{y} - f(\mathbf{P})\| \leq \tau$, *alors on a forcément* $Q(P, f(P)) = 0$.

Preuve . *Soit* f *une telle fonction, et supposons* $Q(P, f(P)) \neq 0$. *Alors, on a* $Q(P, f(P)) \in \mathcal{L}(\Delta')$, *et* $Q(P_i, f(P_i)) = 0$ *en au moins* $n - \tau$ *points. Comme* Δ' *est de degré* $n - \tau - 1$, *et de support disjoint des* P_i, *on en déduit* $Q(P, f(P)) = 0$. $\diamondsuit$

Algorithme 4 : Décodage en liste des codes géométriques

Input : $\mathbf{y} = (y_1, \ldots, y_n), \mathcal{X}, \Delta, \mathbf{P}, \tau$.
Output : Liste des $f \in \mathcal{L}(\Delta)$ vérifiant $\|\mathbf{y} - f(\mathbf{P})\| \leq \tau$.

Précalculs : *Choisir* $\Delta' \in Div(\mathcal{X})$ *de degré* $n - \tau - 1$.
Calculer des bases des $\mathcal{L}(\Delta' - i\Delta)$.

Interpolation : *Construire* $Q(P, Y) = \sum_{i=0}^{l} Q_i(P)Y^i \in (\overline{\mathbb{F}_q}(\mathcal{X}))[Y]$
vérifiant :
- $Q \neq 0$,
- $Q_i \in \mathcal{L}(\Delta' - i\Delta)$,
- $Q(P_i)(y_i) = 0$ *pour tout* $i \leq n$.

Calcul des racines : *Trouver les* $f \in \mathcal{L}(\Delta)$ *vérifiant* $Q(P, f(P)) = 0$, *en extraire ceux vérifiant* $\|\mathbf{y} - f(\mathbf{P})\| \leq \tau$.

Lemme 4. *Si* $Q(Y)$ *est un polynôme à coefficients dans* $\mathcal{L}(\Delta')$ *(les coefficients de* $Q(Y)$ *sont des fonctions rationnelles en* X*), alors on sait trouver en temps polynomial toutes les racines de* Q, *ie. toutes les fonctions* $f \in \mathcal{L}(\Delta')$ *telles que* $Q(f)$ *est identiquement nulle sur* $\mathcal{X}$.

Preuve . *Cf. [26] et [42].* $\diamondsuit$

Proposition 18. *L'algorithme renvoie tous les mots de codes à distance au plus* τ *du mot reçu, en temps polynomial, pour* $\tau < n - \sqrt{2\deg(\Delta)n} - g$.

L'algorithme de Sudan peut être amélioré en introduisant des multiplicités dans l'équation d'annulation du polynôme Q, on obtient ainsi l'algorithme de Guruswami-Sudan, qui corrige en temps polynomial jusqu'à $n - \sqrt{nk}$ erreurs.

Définition 26 (Multiplicité)**.** *Soit* $Q(Y)$ *un polynôme à coefficients dans* $\mathcal{L}(\Delta')$. *Soit* $P \in \mathcal{X}$, *et* $y \in \mathbb{F}_q$. *On dit que* Q *admet un zéro de multiplicité au moins* r *en* (P, y) *si* $Q(Y - y)$ *peut s'écrire sous la forme*

$$Q(Y - y) = \sum_{i=0}^{l} f_i Y^i,$$

avec $\mathrm{div}(f_i) \geq (r - i)\langle P_i \rangle$.

Algorithme 5 : Décodage en liste des codes géométriques (Guruswami-Sudan)

Input : $\mathbf{y} = (y_1, \ldots, y_n), \mathcal{X}, \Delta, \mathbf{P}, \tau, r$.
Output : Liste des $f \in \mathcal{L}(\Delta)$ vérifiant $\|\mathbf{y} - f(\mathbf{P})\| \leq \tau$.

Précalculs : *Choisir* $\Delta' \in Div(\mathcal{X})$ *de degré* $r(n-\tau)$.
Calculer des bases des $\mathcal{L}(\Delta' - i\Delta)$.

Interpolation : *Construire* $Q(P,Y) = \sum_{i=0}^{l} Q_i(P)Y^i \in (\overline{\mathbb{F}}(\mathcal{X}))[Y]$
vérifiant :
- $Q \neq 0$,
- $Q_i \in \mathcal{L}(\Delta' - i\Delta)$,
- Q *admet un zéro de multiplicité au moins* r *en* (P_i, y_i) *pour tout* $i \leq n$.

Calcul des racines : *Trouver les* $f \in \mathcal{L}(\Delta)$ *vérifiant* $Q(P, f(P)) = 0$, *en extraire ceux vérifiant* $\|\mathbf{y} - f(\mathbf{P})\| \leq \tau$.

Théorème 7. *L'algorithme 5 renvoie tous les mots à distance au plus* τ *du mot reçu, en temps polynomial, pour* $\tau \leq n - \sqrt{n\deg(\Delta)(1+\frac{1}{r})} - \frac{g-1}{r}$, *soit en prenant* r *suffisamment grand,* $\tau < n - \sqrt{n\deg(\Delta)}$.

3.3 Cryptanalyse des systèmes de McEliece utilisant des codes géométriques

3.3.1 Reconnaissance des codes géométriques

Pour générer une clé McEliece, le concepteur choisit secrètement une courbe hyperelliptique $\mathcal{X}'$ de genre g, un diviseur Δ' de degré $k+g-1$, un support d'évaluation $(P'_1, \ldots, P'_n)$. Il calcule alors G' la matrice $k \times n$ génératrice du code $\mathcal{C}' = \mathrm{AGC}(\mathcal{X}', \Delta', (P'_1, \ldots, P'_n))$, écrite sous forme systématique. Il sélectionne ensuite une matrice inversible S, une matrice de permutation P, et calcule la clé publique $G_{\mathrm{pub}} = SG'P$. Cette clé publique est traitée, pour le chiffrement, comme la matrice génératrice d'un code linéaire $[n,k]$.

Par construction, le code $\mathcal{C}$ généré par la matrice G_{pub} est également un

code géométrique, mais la définition précise du code est, au premier abord, difficile à retrouver. Mettre au point une attaque structurelle sur la clé publique revient à résoudre le problème suivant :

Reconnaissance des codes géométriques(G, g)
En notant $\mathcal{C}$ le code de matrice génératrice G, dans le cas où $\mathcal{C}$ est un code géométrique $[n, k]$ défini sur une courbe de genre g, trouver $\mathcal{X}, \Delta, \mathbf{P}, \mathbf{c}$ tels que $\mathcal{C} := \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}, \mathbf{c})$.

Théorème 8. *Si l'on sait résoudre en temps polynomial le problème Reconnaissance des codes géométriques(G_{pub}, g), alors on sait attaquer en temps polynomial la clé publique G_{pub} d'un cryptosystème de McEliece utilisant des codes géométriques de genre g.*

Preuve . *On dispose d'algorithmes permettant de décoder le code géométrique* $\mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P})$. *Ces algorithmes peuvent sans problème être adaptés pour tenir compte de la multiplication directionnelle de vecteur* $\mathbf{c}$. $\diamondsuit$

Par conséquent, nous allons nous intéresser à la résolution du problème de Reconnaissance des codes géométriques.

Remarque 3. *La solution au problème de Reconnaissance des codes géométriques(G, g) n'est pas unique.*

Les attaques présentées dans ce chapitre exploitent cette remarque en fixant certains éléments de la solution au problème de Reconnaissance des codes géométriques. Nous allons voir sous quelles condition cette solution peut ainsi être contrainte.

3.3.2 Isomorphisme de courbes

Les définitions données ici sont tirées de [21], et adaptées au cas particulier qui nous intéresse.

Définition 27. *Une application $\mathbb{F}_q$-rationnelle de la courbe hyperelliptique $\mathcal{X}$ vers $\mathcal{X}'$ est une application de la forme*

$$\begin{array}{rcl} \Phi : \mathcal{X} & \to & \mathcal{X}' \\ (X, Y, Z) & \mapsto & (f_1(X, Y, Z), f_2(X, Y, Z), f_3(X, Y, Z)) \end{array}$$

avec $f_1, f_2, f_3 \in \mathbb{F}_q(\mathcal{X})$.

L'application Φ n'est a priori pas définie en $P \in \mathcal{X}$ si P est un pôle de l'un des f_i, ou bien un zéro de tous à la fois. Dans ce cas, si cela est possible, on étend Φ en P par "continuité" : S'il existe une fonction $g \in \mathbb{F}_q(\mathcal{X})$ tel que $gf_i(P)$ est défini pour $i = 1, 2, 3$ et que l'un au moins est non nul, on pose $\Phi(P) = ((gf_1)(P), (gf_2)(P), (gf_3)(P))$. On constate que cette définition est indépendante de la fonction g choisie.

L'application Φ est dite régulière en $P \in \mathcal{X}$, si $\Phi(P)$ est défini, au moins par continuité. Si Φ est régulière en tout point de $\mathcal{X}$, alors Φ est un $\mathbb{F}_q$-morphisme de courbes. Si, de plus, Φ est bijective et que Φ^{-1} est un $\mathbb{F}_q$-morphisme de $\mathcal{X}'$ vers $\mathcal{X}$, alors Φ est un $\mathbb{F}_q$-isomorphisme de courbes.

Notons que l'image d'un point rationnel par un $\mathbb{F}_q$-morphisme de courbe reste rationnelle.

Théorème 9. *Si $\Phi : \mathcal{X} \to \mathcal{X}'$ est un $\mathbb{F}_q$-isomorphisme de courbe, alors Φ induit un isomorphisme additif (que l'on note également Φ) entre les groupes de diviseurs $Div(\mathcal{X})$ et $Div(\mathcal{X}')$.*
De plus, $\Phi : Div(\mathcal{X}) \to Div(\mathcal{X}')$ conserve la rationnalité, le degré, ainsi que le caractère principal d'un diviseur. Par conséquent, Phi induit également un isomorphisme entre $\mathrm{Jac}(\mathcal{X})$ *et* $\mathrm{Jac}(\mathcal{X}')$.

Preuve . *Si $\Delta = \sum_{P \in \mathcal{X}} n_P \langle P \rangle \in Div(\mathcal{X})$, on pose $\Phi(\Delta) = \sum_{P \in \mathcal{X}} n_{\Phi(P)} \langle \Phi(P) \rangle$. Ceci définit clairement un isomorphisme de groupes conservant le degré. La conservation de la rationnalité vient du fait que les composantes de Φ sont à coefficients rationnels, donc Galois-invariantes. Pour le caractère principal, notons que si $\Delta = \mathrm{div}(f)$ est un diviseur principal sur $\mathcal{X}$, alors $\Phi(\Delta) = \mathrm{div}(f \circ \Phi^{-1})$ est également rationnel.* $\diamondsuit$

Théorème 10 (Invariance des codes géométriques)**.** *Soit $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}, \mathbf{c})$ un code géométrique, et soit Φ un $\mathbb{F}_q$-isomorphisme de courbes de $\mathcal{X}$ vers $\mathcal{X}'$. Alors, on a $\mathcal{C} = \mathrm{AGC}(\mathcal{X}', \Phi(\Delta), \Phi(\mathbf{P}), \mathbf{c})$.*

Preuve . *Il suffit de reprendre la construction des codes géométriques, toutes les propriétés sont conservées par l'isomorphisme Φ.* $\diamondsuit$

Théorème 11. *Soit $\mathcal{X}$ une courbe hyperelliptique de genre $g \geq 1$. Soit $T \in \mathbb{F}_q[X]$ un polynôme rationnel de degré inférieur ou égal à g, et $\tilde{T}(X,Z)$ le polynôme homogène de degré d correspondant. Soient $\alpha \in \mathbb{F}_q^*$, $\beta \in \mathbb{F}_q$. Alors la fonction*

$$\begin{aligned} \Phi : \mathcal{X} &\rightarrow \Phi(\mathcal{X}) \\ (X,Y,Z) &\mapsto \left(\frac{\alpha^2 X + \beta}{Z}, \frac{\alpha^{2g+1} Y Z^{g-1} + \tilde{T}(X,Z)}{Z^g}, 1 \right) \end{aligned}$$

est un isomorphisme de courbes, qui s'écrit, sous forme déshomogénéisée :

$$\Phi(x,y) = (\alpha^2 x + \beta, \alpha^{2g+1} y + T(x))$$

$$\Phi(\mathcal{O}) = \mathcal{O}$$

Preuve . *Une preuve complète peut être trouvée dans [45]. Ici, sans entrer dans les détails de calcul, nous nous bornerons à quelques constatations : Φ est une application rationnelle définie sur tous les points affines de $\mathcal{X}$. Une multiplication par $g = \frac{Z^{\deg(T)}}{Y^{\deg(T)}}$ permet de définir $\Phi(\mathcal{O}) = \Phi(0,1,0) = \mathcal{O}$. L'application Φ est alors régulière en tout point de $\mathcal{X}$.*
De plus, si $\mathcal{X}$ est la courbe hyperelliptique d'équation $y^2 + G(x)y = F(x)$, on constate que $Im(\Phi)$ est l'ensemble des points (x,y) vérifiant $y^2 + G'(x)y' = F'(x)$, avec

$$G'(x) = -2T\left(\frac{x-\beta}{\alpha^2}\right) + \alpha^{2g+1} G\left(\frac{x-\beta}{\alpha^2}\right),$$

$$F'(x) = \alpha^{4g+2} F\left(\frac{x-\beta}{\alpha^2}\right) - T^2\left(\frac{x-\beta}{\alpha^2}\right) + \alpha^{2g+1} G\left(\frac{x-\beta}{\alpha^2}\right) T\left(\frac{x-\beta}{\alpha^2}\right).$$

On constate que G' et F' vérifient les mêmes conditions que G et F, à savoir $\deg(G') \leq g$, et F' unitaire de degré $2g+1$. Donc $Im(\Phi)$ est une courbe hyperelliptique de genre g.
L'injectivité de Φ est triviale, sa surjectivité découle de sa définition. Il ne reste plus qu'à inverser l'expression de Φ pour constater que Φ^{-1} s'exprime à partir de fonctions rationnelles à coefficients dans $\mathbb{F}_q$. Φ est donc un $\mathbb{F}_q$-isomorphisme de courbes hyperelliptiques. $\diamondsuit$

Corollaire 4. *Soit $\mathcal{C} = \mathrm{AGC}(\mathcal{X}', \Delta', \mathbf{P}', \mathbf{c})$ un code géométrique construit sur une courbe de genre g. Soit $k_1, \ldots, k_{g+1} \in [1,n]$ $g+1$ entiers distincts. Soit $x_{k_1}, x_{k_2}, y_{k_1}, \ldots, y_{k_{g+1}} \in \mathbb{F}_q$ $g+3$ valeurs choisies aléatoirement. Alors, avec probabilité 1/2, on peut compléter ces $g+3$ valeurs en $x_1, \ldots, x_n, y_1, \ldots, y_n$ tels que l'on puisse écrire $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, \mathbf{P}, \mathbf{c})$, avec P_i de coordonnées (x_i, y_i).*

Preuve . *On construit un isomorphisme de courbes en utilisant le théorème précédent. À condition que $x_{k_1} \neq x_{k_2}$, on détermine α, β, T à partir des valeurs de $x_{k_1}, x_{k_2}, y_{k_1}, \ldots, y_{k_{g+1}}$ choisies. L'isomorphisme Φ existe si et seulement si la solution trouvée pour α^2 est un carré dans $\mathbb{F}_q$, d'où la probabilité 1/2. Si cela est le cas, on pose $\mathcal{X} = \Phi(\mathcal{X}')$, $\Delta = \Phi(\Delta')$, et $\mathbf{P} = \Phi(\mathbf{P}')$.* $\diamondsuit$

Ce corollaire nous permet de gagner du temps dans la résolution du problème de Reconnaissance des codes géométriques. En effet, on peut fixer arbitrairement 2 abscisses et $g + 1$ ordonnées des points du support. Avec probabilité 1/2, on pourra compléter nos paramètres arbitrairement fixés pour obtenir une courbe et un support correspondant à notre code.

De plus, une fois la courbe et le support du code fixé, on conserve encore de la liberté dans le choix du diviseur. D'après la propriété 16, on peut choisir n'importe quel diviseur de la même classe d'équivalence (tant que son support est disjoint du support d'évaluation), tant que les constantes $\mathbf{c}$ ne sont pas encore fixées.

3.3.3 Genre 0 : Codes de Reed-Solomon généralisés et attaque de Sidelnakov-Shestakov

Les codes de Reed-Solomon ont été proposés à l'origine par Niederreiter dans [33] comme une famille de codes envisageable pour son cryptosystème. Cependant, en 1992, Sidelnikov et Shestakov ont montré dans [43] qu'il était facile de reconnaitre un tel code. On peut voir les codes de Reed-Solomon généralisés comme des codes géométriques définis sur une courbe de genre 0, et c'est en utilisant ce formalisme, inspiré du travail de Minder dans [32], que nous allons présenter l'attaque de Sidelnikov et Shestakov, dont certaines idées seront reprises dans les attaques contre les genres supérieurs.

Soit $\mathbb{F}_q$ un corps fini, et $k \leq n \leq q$ deux entiers positifs. Soit $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ deux à deux distincts. On appelle $\mathcal{X}$ la ligne projective d'équation $Y = 0$ dans $\mathbb{P}_2(\overline{\mathbb{F}_q})$. $\mathcal{X}$ est une courbe de genre 0 contenant $q + 1$ points rationnels. Pour $1 \leq i \leq n$, on appelle P_i le point de $\mathcal{X}$ de coordonnées $(\alpha_i, 0, 1)$. On note $\mathcal{O}$ le point à l'infini de $\mathcal{X}$: $\mathcal{O} = (1, 0, 0)$. Alors on définit le code de Reed-Solomon de dimension k évalué en $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ de la façon suivante :

$$\mathrm{RS}_k(\boldsymbol{\alpha}) = \mathrm{AGC}(\mathcal{X}, (k-1)\langle\mathcal{O}\rangle, \mathbf{P}),$$

et pour $\mathbf{c} = (\mathbf{c_1}, \ldots, \mathbf{c_n}) \in \mathbb{F_q}*$ le code de Reed-Solomon généralisé correspondant :

$$\mathrm{GRS}_k(\boldsymbol{\alpha}, \boldsymbol{c}) = \mathrm{AGC}(\mathcal{X}, (k-1)\langle\mathcal{O}\rangle, \boldsymbol{P}, \boldsymbol{c}).$$

Les mots d'un code de Reed-Solomon correspondent à l'évaluation sur $\mathbb{F}_q$ de polynômes de degré au plus $k-1$. Un code de Reed-Solomon généralisé est une multiplication directionnelle d'un code de Reed-Solomon.

Les codes de Reed-Solomon généralisés sont des codes géométriques de genre 0, et la réciproque est vraie également : tout code géométrique de genre 0 peut être vu comme un code de Reed-Solomon généralisé. En effet, toutes les courbes de genre 0 sont isomorphes à la ligne $Y = 0$. De plus, la Jacobienne est triviale, donc tous les diviseurs sont équivalents, et donc on peut considérer (par multiplication directionnelle) que le diviseur d'un code géométrique de genre 0 est de la forme $\deg(\Delta)\langle\mathcal{O}\rangle$.

Les codes de Reed-Solomon généralisés sont optimaux, au sens où ils vérifient l'égalité de Singleton $k+d = n+1$, c'est à dire que, pour une longueur et une dimension fixée, sa distance minimale est maximale. De plus, les zéros d'un mot de poids faible peuvent être librement choisis :

Proposition 19. *Soit $I \subset [1, n]$ un ensemble de $k-1$ entiers, alors il existe un mot $\mathbf{x}$ du code $\mathcal{C}$ de poids minimal (n-k+1) tel que $x_i = 0 \Leftrightarrow i \in I$. De plus, un tel mot se construit facilement (en $O(k^2 n)$ opérations) en appliquant un procédé de pivot à une matrice génératrice de $\mathcal{C}$.*

C'est de cette facilité de construction des mots de poids faible que vient la fragilité cryptographique des codes de Reed-Solomon généralisés, et l'attaque de Sidelnikov-Shestakov tire parti de cette faiblesse.

Théorème 12 (Sidelnikov-Shestakov)**.** *Soit $\mathcal{C}$ un code de Reed-Solomon généralisé de longueur n et de dimension k sur $\mathbb{F}_q$, connu uniquement par une matrice génératrice G. Alors, on sait construire en $O(n^3)$ opérations $\boldsymbol{\alpha} \in \mathbb{F}_q^n$ et $\boldsymbol{c} \in \mathbb{F}_q^n$ tels que $\mathcal{C} = GRS_k(\boldsymbol{\alpha}, \boldsymbol{c})$.*

Pour ce faire, on construit $\mathbf{v}$ et $\mathbf{w}$, deux mots de poids faible du code $\mathcal{C}$, dont les zéros sont presque en même position :

$$v_4 = \cdots = v_{k+1} = v_{k+2} = 0,$$

$$w_4 = \cdots = w_{k+1} = w_{k+3} = 0.$$

On appelle respectivement f et g les fonctions rationnelles inconnuues associées à ces deux mots dans $\mathcal{L}((k-1)\langle\mathcal{O}\rangle)$. On peut cependant dire des choses sur les diviseurs de ces fonctions :

On a $f \in \mathcal{L}((k-1)\langle\mathcal{O}\rangle)$, soit

$$\mathrm{div}(f) \geq -(k-1)\langle\mathcal{O}\rangle.$$

Mais $f(P_4) = v_4 = 0$, donc $\mathrm{ord}_{P_4}(f) \geq 1$, d'où

$$\mathrm{div}(f) \geq \langle P_4\rangle - (k-1)\langle\mathcal{O}\rangle.$$

Par un raisonnement analogue, on a

$$\mathrm{div}(f) \geq \langle P_4\rangle + \cdots + \langle P_{k+2}\rangle - (k-1)\langle\mathcal{O}\rangle.$$

Or, dans cette inégalité, les deux termes sont de degré 0. Par conséquent, il s'agit d'une égalité, et on a

$$\mathrm{div}(f) = \langle P_4\rangle + \cdots + \langle P_{k+2}\rangle - (k-1)\langle\mathcal{O}\rangle. \tag{3.11}$$

Par un raisonnement similaire, on montre

$$\mathrm{div}(g) = \langle P_4\rangle + \cdots + \langle P_{k+1}\rangle + \langle P_{k+3}\rangle - (k-1)\langle\mathcal{O}\rangle. \tag{3.12}$$

On en déduit

$$\mathrm{div}\left(\frac{f}{g}\right) = \langle P_{k+2}\rangle - \langle P_{k+3}\rangle.$$

Par conséquent, on peut écrire

$$\frac{f}{g} = \frac{aX + bZ}{cX + dZ}, \tag{3.13}$$

avec a, b, c, d quatre constantes homogènes inconnues.

De plus, si $w_i \neq 0$, on a aussi

$$\frac{f}{g}(P_i) = \frac{c_i v_i}{c_i w_i} = \frac{v_i}{w_i}, \tag{3.14}$$

avec v_i et w_i connus.

On en déduit donc, pour tout $i \leq n$,

$$\frac{v_i}{w_i} = \frac{a\alpha_i + b}{c\alpha_i + d}. \tag{3.15}$$

En utilisant des isomorphismes de courbes spécifiques au genre 0, on peut arbitrairement choisir trois valeurs distinctes dans $\mathbb{F}_q$ pour $\alpha_1, \alpha_2, \alpha_3$. Une fois ces valeurs fixées, on peut donc utiliser l'équation 3.15 avec $i = 1, 2, 3$ pour calculer les valeurs de a, b, c et d.

Ensuite, pour chaque $i \geq k + 4$, on utilise la même équation 3.15, mais cette fois-ci, l'inconnue est α_i. On retrouve ainsi les coordonnées de $n-k-3$ points d'évaluation du code. Les points d'évaluation restants sont calculés facilement, par exemple en réitérant le procédé à partir de deux mots de code différents.

Il reste à calculer les constantes de multiplication directionnelle $c_1, \ldots, c_n$. Cela peut se faire facilement, par exemple à partir de la matrice de parité H du code $\mathcal{C}$. On construit

$$\mathbf{x} = (x_1, \ldots, x_n) \in \mathrm{AGC}(\mathcal{X}, (k-1)\langle\mathcal{O}\rangle, (P_1, \ldots, P_n)).$$

Alors

$$\mathbf{cx} \in \mathcal{C}, \text{ donc } H\mathbf{cx}^T = 0,$$

soit pour tout $i \leq n - k$,

$$\sum_{j=1}^{n} h_{ij} c_j x_j = 0.$$

Chacune des $n - k$ lignes de la matrice de parité nous fournit une équation linéaire sur les $c_1, \ldots, c_n$. En utilisant plusieurs mots de code, on obtient suffisamment d'équations pour résoudre le système linéaire, et calculer ainsi les constantes de multiplication directionnelle.

3.3.4 Genre 1 : Codes elliptiques et attaque de Minder

Les courbes de genre 1 sont appelées courbes elliptiques, et les codes géométriques associés sont appelés codes elliptiques. En 2007, L. Minder a mis au point dans [32] une attaque rapide contre ces codes. L'attaque de Minder est essentiellement la même, en légèrement plus simple, que celle que nous avons mise au point en genre 2. Aussi, je ne décrirai pas l'attaque de Minder en détail, mais je vais me contenter de souligner les spécificités du genre 1 :

– Sur une courbe elliptique, la Jacobienne est isomorphe au groupe des points rationnels de la courbe. Tous les calculs que nous effectuons dans la Jacobienne en genre 2 sont effectués directement avec les points de la courbe en genre 1. Les calculs sont essentiellement les mêmes, mais le formalisme du genre 2 est plus général.

– En utilisant une translation comme isomorphisme de courbes, Minder montre que, lors de son attaque, on peut considérer que le diviseur du code est de la forme $\Delta = k\langle\mathcal{O}\rangle$. Ainsi, dans la première étape de l'algorithme, les mots de poids faibles fournissent directement des équations sur les points de la courbe. De plus, l'expression de Δ dans la Jacobienne est triviale, on n'a pas besoin de recourir un test statistique pour retrouver sa valeur. Cette simplification n'est plus valable à partir du genre 2.

– Lors de la deuxième étape de l'algorithme, Minder retrouve les ordonnées des points d'évaluation du code en utilisant des alignements de points de la courbe (basiquement, des sommes nulles de 3 termes de la Jacobienne). À partir du genre 2, on ne sait pas reconnaitre dans la Jacobienne si 3 ou plusieurs points sont alignés. La condition plus générale que nous utilisons devient alors "$2g + 1$ points de la courbe sont interpolés par un polynôme de degré au plus g".

L'algorithme de Minder fonctionne si le code utilisé vérifie les trois mêmes hypothèses qu'en genre 2. Dans ce cas, en utilisant les spécificités ci-dessus, Minder donne un algorithme qui s'exécute en $O(n^4)$ opérations binaires, et qui retrouve la structure du code avec une grande probabilité, comme le montre l'implémentation qu'il en a réalisée.

Théorème 13 (Minder)**.** *Soit $\mathcal{C}$ un code géométrique de longueur n et de dimension k sur $\mathbb{F}_q$, connu uniquement par une matrice génératrice G. On suppose que $\mathcal{C}$ est construit sur une courbe de genre 1, et vérifie les trois hypothèses ci-après. Alors, on sait, avec grande probabilité, construire en $O(n^3)$ opérations $\mathcal{X}, \Delta \in Div(\mathcal{X}), \boldsymbol{P} \in \mathcal{X}(\mathbb{F}_q)$, et $\boldsymbol{c} \in \mathbb{F}_q^n$ tels que $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, \boldsymbol{P}, \boldsymbol{c})$.*

3.4 Cryptanalyse des codes hyperelliptiques en genre 2

L'attaque présentée ici est le résultat d'un travail de collaboration que j'ai effectué avec Lorenz Minder. Les résultats n'ont pas encore été publiés, mais ont fait l'objet d'une présentation à ACCT en 2008 [16].

Cette attaque est heuristique en temps polynomial : l'algorithme retrouve la structure du code géométrique en un temps polynomial ($O(n^4)$), mais pas de façon certaine. Cependant, sur un code généré aléatoirement, la probabilité d'échec de l'algorithme est extrêmement proche de zéro. L'idée sous-jacente est de déduire des informations sur la courbe à partir de la répartition des positions nulles sur des mots de poids faible.

Nous avons cependant besoin de trois hypothèses additionnelles sur le code géométrique utilisé.

- Premièrement, on suppose que le code contient beaucoup de points d'évaluation, i.e., que n est proche du nombre de points rationnels de $\mathcal{X}$. Cette hypothèse semble raisonnable en cryptologie, puisque le principal intérêt de construire un code sur une courbe hyperelliptique est que la courbe contient plus de points que le corps de base, autrement dit que sur un corps donné, on peut construire un code de longueur élevée ($n \leq q + 2g\sqrt{q}$ au lieu de $n \leq q$, cf corollaire 2). Il semble donc contre-productif de réduire artificiellement la longueur du code utilisé, mais une telle réduction, surtout si elle est faite intelligemment, semble être une des mesures les plus prometteuses pour neutraliser l'attaque. Cependant, un cryptosystème utilisant un nombre réduit de points d'évaluation perd en efficacité, ce qui le classera sans doute inférieur à d'autres cryptosystèmes utilisant des codes.

- Deuxièmement, on suppose que le cardinal de la jacobienne de la courbe et le degré du diviseur du code sont premiers entre eux ($\gcd(k + g - 1, |\mathcal{G}|) = 1$). Cette condition n'est pas limitante, puisqu'on peut la forcer en considérant un sous-code (on réduit k en supprimant une ligne de la matrice génératrice). Evidemment, puisque le cardinal de la Jacobienne est a priori inconnu, on ne procèdera à une telle coupe qu'après un échec de l'algorithme. Grâce à cette supposition, on sait que Δ est équivalent à un diviseur de la forme $(k+g-1)\Delta_0$, où $\Delta_0 \in \mathrm{Div}(\mathcal{X})$ est un diviseur de degré 1.

En effet, si P est un point arbitraire de la courbe, $\Delta-(k+g-1)\langle P\rangle$ est un diviseur de degré 0. Puisque $\gcd(k+g-1,|\mathcal{G}|)=1$, il existe Δ' de degré 0, tel que $\Delta-(k+g-1)\langle P\rangle=(k+g-1)\Delta'$ dans la Jacobienne. D'où l'existence de $\Delta_0=\langle P\rangle+\Delta'$.

– Troisièmement, on suppose que la distance minimale vaut véritablement $d=n-k-g+1$ (alors qu'on sait seulement $n-k-g+1\leq d\leq n-k+1$). On suppose de plus que les mots de poids minimum sont nombreux et faciles à construire. Des résultats empiriques montrent que cette supposition est vérifiée en pratique. Dans la section suivante se trouve une étude du nombre du coût heuristique de la recherche d'un mot de poids faible en genre g.

Théorème 14. *Soit $\mathcal{C}$ un code géométrique de longueur n et de dimension k sur $\mathbb{F}_q$, connu uniquement par une matrice génératrice G. On suppose que $\mathcal{C}$ est construit sur une courbe de genre 2, et vérifie les trois hypothèses ci-dessus. Alors, l'algorithme suivant permet, avec grande probabilité, de construire en $O(n^3)$ opérations de corps $\mathcal{X},\Delta\in Div(\mathcal{X}),\boldsymbol{P}\in\mathcal{X}(\mathbb{F}_q)$, et $\boldsymbol{c}\in\mathbb{F}_q^n$ tels que $\mathcal{C}=\mathrm{AGC}(\mathcal{X},\Delta,\boldsymbol{P},\boldsymbol{c})$.*

Afin de fixer les idées sur les paramètres des codes que nous attaquons, rappelons que dans [27], les auteurs proposent l'emploi d'un code [171,109,61] construit à partir d'une courbe hyperelliptique de genre 2 sur $\mathbb{F}_{2^7}$. Selon eux, en fixant $n=171$, $k=109$, et $q=128$, les attaques devraient coûter 2^{80} opérations binaires.

3.4.1 Présentation de l'attaque

Notre algorithme se compose de quatre étapes :

1. Retrouver la structure du groupe des diviseurs. Nous commençons par rassembler un grand nombre de mots de code de poids minimum afin d'en déduire des équation linéaires sur les éléments de la Jacobienne. Avec suffisamment d'équations, nous sommes capables de déduire la structure de groupe de la Jacobienne, ainsi que la valeur dans ce groupe des diviseurs liés aux points d'évaluation.

2. Retrouver l'équation de la courbe. A partir de ce que l'on sait sur la Jacobienne, on construit quelques mots de code de poids faible, et on en déduit des conditions sur les coordonnées de quelques points de la courbe.

   Ensuite, on fait une supposition sur les coordonnées de quelques points, et on calcule les coordonnées d'un plus grand ensemble de points en utilisant les conditions établies juste avant. Si notre supposition est exacte, on peut tracer une courbe hyperelliptique passant par les points dont nous avons calculé les coordonnées. Si ce n'est pas le cas, on refait cette étape avec une supposition différente.

3. Retrouver les coordonnées des points d'évaluation. Maintenant, nous sommes capables, à partir de la structure de la Jacobienne, et de l'équation de la courbe, de calculer les coordonnées de tous les points de l'ensemble d'évaluation, ainsi que le diviseur du code.

4. Calculer les constantes de multiplication directionnelle. Nous avons construit un code qui est une déformation (par multiplication directionnelle) du code qui nous était donné. Il ne nous reste plus qu'à calculer les constantes de cette déformation.

3.4.2 Première étape : Retrouver la structure de la Jacobienne

En entrée de l'algorithme, on se donne G, une matrice génératrice $k \times n$ du code géométrique $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, \boldsymbol{P}, \boldsymbol{c})$, où la courbe $\mathcal{X}$ est de genre 2. À partir du nombre de colonnes de G, il est trivial de déduire la valeur de $\deg(\Delta) = k + g - 1$.

D'après le théorème 4, on sait qu'il existe un isomorphisme inconnu φ de $\mathrm{Jac}(\mathcal{X})$ vers $\mathcal{G} = \frac{\mathbb{Z}}{d_1\mathbb{Z}} \times \cdots \times \frac{\mathbb{Z}}{d_4\mathbb{Z}}$. Notre but, pour cette étape, est de retrouver les valeurs de $d_1, \ldots, d_4$, c'est-à-dire la structure de la Jacobienne. Au passage, nous en profiterons pour calculer dans $\mathcal{G}$ les valeurs des $z_i = \varphi(P_i - \langle\mathcal{O}\rangle))$.

Pour cela, on génère des mots de code de poids minimum, c'est à dire des $\mathbf{x} \in \mathcal{C}$ tels que $|\mathbf{x}| = n - k - g + 1$. Pour un code général, la recherche de mots de poids faible est un problème difficile, mais pour des codes géométriques de genre 2, il nous suffit d'effectuer en moyenne $O(n^2)$ opérations de corps pour

générer un mot de poids faible, en utilisant une méthode du type Ensemble d'Information. La méthode utilisée est détaillée et sa complexité est analysée en fonction du genre g dans la section 3.5.1.

Soit $\mathbf{x}$ un tel mot de code, et on appelle $f \in \mathcal{L}(\Delta)$ la fonction rationnelle associée (i.e., la fonction telle que $\mathbf{x}_i = f(P_i)$ pour $1 \leq i \leq n$). $\mathbf{x}$ admet $k+g-1$ positions nulles, que l'on numérote $i_1, \ldots, i_{k+g-1}$. Alors la fonction f vérifie

$$\mathrm{div}(f) \geq -\Delta, \text{ et } f(P_{i_1}) = \cdots = f(P_{i_{k+g-1}}) = 0.$$

Comme P_{i_1} n'est pas dans le support de Δ, on en déduit

$$\mathrm{div}(f) \geq \langle P_{i_1} \rangle - \Delta.$$

Par un raisonnement similaire, on montre

$$\mathrm{div}(f) \geq \sum_{j=1}^{k+g-1} \langle P_{i_j} \rangle - \Delta.$$

Or, dans cette inégalité, les deux diviseurs sont de degré 0, on a donc en fait une égalité :

$$\mathrm{div}(f) = \sum_{j=1}^{k+g-1} \langle P_{i_j} \rangle - \Delta. \tag{3.16}$$

Le diviseur $\mathrm{div}(f)$ est évidemment un diviseur principal. De plus, la condition $\gcd(k+g-1, |\mathcal{G}|) = 1$ nous permet de supposer que Δ est équivalent à un diviseur de la forme $(k+g-1)\Delta_0$. On sait donc que $\sum_{j=1}^{k+g-1} (\langle P_{i_j} \rangle - \Delta_0)$ est un diviseur principal.

Si on se place dans la Jacobienne, et qu'on note $\tilde{z}_i = \varphi(\langle P_i \rangle - \Delta_0) \in \mathcal{G}$, cette propriété se traduit naturellement par l'équation

$$\sum_{j=1}^{k+g-1} \tilde{z}_{i_j} = 0. \tag{3.17}$$

En calculant plusieurs mots, on accumule des équations linéaires sur les $\tilde{z_i}$. Si l'on suppose que les $\tilde{z_i}$ engendrent $\mathcal{G}$ (ceci est raisonnable, car on a supposé n grand, et $\mathcal{G}$ admet une partie génératrice à 4 éléments), le calcul de la forme normale de Smith associée aux z_i nous fournit les valeurs de $d_1, \ldots, d_4$, et donc la structure de $\mathcal{G}$. Pour en savoir plus sur le calcul général de la forme normale de Smith, on pourra se rapporter à [8].

Pour que le calcul des invariants de Smith aboutisse au bon résultat, il faut que l'on ait rassemblé suffisamment d'équations pour déterminer complètement la structure de $\mathcal{G}$ comme groupe abélien libre engendré par les $\tilde{z_i}$. Ceci est en fait impossible, car toutes les équations que l'on considère sont par construction de poids $k + g - 1$. Toute combinaison linéaire à coefficients entiers de ces équations aura donc un poids divisible par $k + g - 1$, et tout déterminant $n \times n$ extrait de ce système d'équation sera divisible par $k + g - 1$ (pour s'en convaincre, on peut effectuer la transformation élémentaire qui ajoute à la première colonne la somme de toutes les autres). Comme $k+g-1$ est premier avec $|\mathcal{G}|$, les invariants de Smith de notre système seront en fait $1, \ldots, 1, d_1, d_2, d_3, (k + g - 1)d_4$, desquels on déduira les valeurs de $d_1, \ldots, d_4$.

À ce détail près, il nous faut en moyenne $O(n)$ équations pour déterminer complètement $\mathcal{G}$. Générer les $O(n)$ mots de poids faible correspondants coûte donc $O(n^3)$ opérations dans $\mathbb{F}_q$.

Le calcul des invariants de Smith s'effectue par des transformations matricielles dans $\mathbb{Z}$, dont la plus coûteuse est le calcul d'un déterminant. Puisque le déterminant obtenu au final vaut $(k+g-1)d_1d_2d_3d_4 \leq (k+g-1)(\sqrt{q}+1)^{2g}$, tous les entiers manipulés peuvent être considérés de valeur absolue inférieure à $(k+g-1)(\sqrt{q}+1)^{2g} \leq nq^g \leq q^{g+2}$. En utilisant le lemme chinois, le calcul de déterminant requis dans $\mathbb{Z}$ peut être ramené au calcul de $g+2 = 4$ déterminants modulo des nombres premiers de l'ordre de q. Chaque déterminant se calculant en $O(n^3)$ opérations, le calcul des invariants de Smith coûte donc $O(n^3)$ opérations de corps.

Le calcul des invariants de Smith nous permet en même temps de déterminer les valeurs des $\tilde{z_i}$ dans $\mathcal{G}$.

C'est ici que l'on vérifie que le cardinal de la Jacobienne et $k+g-1$ sont premiers entre eux, en vérifiant que la valeur obtenue pour d_4 est première avec $k + g - 1$ et que la valeur obtenue pour $|\mathcal{G}|$ vérifie l'équation de Hasse-Weil (cf. théorème 3 et remarque 1).

On cherche maintenant à calculer dans $\mathcal{G}$ la valeur de $\delta_0 = \varphi(\Delta_0 - \langle\mathcal{O}\rangle)$. A cette fin, on effectue un test statistique sur les valeurs des $\tilde{z_i} + \tilde{z_j}$. En effet, si P_i et P_j sont des points opposés sur la courbe, on aura :

$$\begin{aligned} \tilde{z_i} + \tilde{z_j} &= \varphi(\langle P_i\rangle - \Delta_0 + \langle P_j\rangle - \Delta_0) \\ &= \varphi(\langle P_i\rangle + \langle P_j\rangle - 2\langle\mathcal{O}\rangle) + \varphi(2\langle\mathcal{O}\rangle - 2\Delta_0) \\ \tilde{z_i} + \tilde{z_j} &= -2\delta_0 \end{aligned}$$

L'hypothèse que presque tous les points rationnels de la courbe sont des points d'évaluation nous assure que dans l'ensemble d'évaluation, il y a de nombreux couples de points opposés, et que statistiquement, la valeur $-2\delta_0$ apparaitra souvent dans les sommes de deux points. Ce test s'effectue en $O(n^2)$ opérations dans $\mathcal{G}$.

On peut alors calculer les valeurs des $z_i = \varphi(\langle P_i\rangle - \langle\mathcal{O}\rangle)$ par la formule $z_i = \tilde{z_i} - \delta_0$.

3.4.3 Deuxième étape : Retrouver l'équation de la courbe

Le principe de cette étape est, comme dans l'attaque de Sidelnikov et Shestakov, ou celle de Minder, d'utiliser deux mots de code très proches afin de calculer les coordonnées des points d'évaluation. Cependant, le genre 2 apporte des complications importantes qui rendent cette étape délicate.

On génère deux mots de code $\mathbf{v}$ et $\mathbf{w}$ de poids $(n-k-g+1)$, tels que $\mathbf{v}$ et $\mathbf{w}$ ont exactement $k+g-3$ positions nulles communes. La génération de ces deux mots est facile maintenant que l'on connait les valeurs dans $\mathcal{G}$ des z_i et de δ_0 :

Pour ce faire, on construit un ensemble d'indices $I \subset [1,n]$ de taille $k+g-3$ tel que $\sum_{i\in I} z_i = (k+g-1)\delta_0$. L'ensemble I est construit en calculant dans $\mathcal{G}$ des sommes de z_i choisis au hasard. Comme $\mathcal{G}$ est de cardinal proche de q^2 (cf. théorème 3), on peut espérer construire I en moins de $O(n^2)$ opérations dans $\mathcal{G}$.

On sélectionne maintenant, parmi les points n'appartenant pas à I, deux paires de points opposés, i.e., $(i_1, i_2, j_1, j_2) \in [1,n] - I$ tels que $z_{i_1} + z_{i_2} = z_{j_1} + z_{j_2} = 0$.

On sait alors que

$$\sum_{i \in I \cup \{i_1, i_2\}} z_i - (k+g-1)\delta_0 = 0,$$

donc que $\sum_{i \in I \cup \{i_1, i_2\}} \langle P_i \rangle - (k+g-1)\Delta_0$ est un diviseur principal. La fonction rationnelle associée appartient à $\mathcal{L}((k+g-1)\Delta_0)$ et s'annule sur les $(P_i)_{i \in I \cup \{i_1, i_2\}}$. Par conséquent, on sait qu'il existe un mot de code $\mathbf{v} \in \mathcal{C}$ dont les positions nulles sont situées sur $I \cup \{i_1, i_2\}$. Ce mot se calcule facilement à partir de la matrice génératrice G. Par la même méthode, on calcule le mot de code $\mathbf{w} \in \mathcal{C}$ dont les positions nulles sont situées sur $I \cup \{j_1, j_2\}$.

Une fois les mots de code $\mathbf{v}$ et $\mathbf{w}$ générés, si l'on appelle f et h leurs fonctions rationnelles (inconnues) respectives associées dans $\mathcal{L}((k+g-1)\Delta_0)$, alors elles vérifient

$$\operatorname{div}\left(\frac{f}{h}\right) = \langle P_{i_1} \rangle + \langle P_{i_2} \rangle - \langle P_{j_1} \rangle - \langle P_{j_2} \rangle.$$

$$\operatorname{div}\left(\frac{f}{h}\right) = (\langle P_{i_1} \rangle + \langle P_{i_2} \rangle - 2\langle \mathcal{O} \rangle) - (\langle P_{j_1} \rangle + \langle P_{j_2} \rangle - 2\langle \mathcal{O} \rangle).$$

Comme on a affaire à des paires de points opposés, on sait que $\langle P_{i_1} \rangle + \langle P_{i_2} \rangle - 2\langle \mathcal{O} \rangle$ est le diviseur d'un polynôme rationnel de degré 1 en x et 0 en y. De même pour $\langle P_{j_1} \rangle + \langle P_{j_2} \rangle - 2\langle \mathcal{O} \rangle$.

On peut donc en déduire qu'il existe $a, b, c, d \in \mathbb{F}_q$ tels que

$$\frac{f}{h}(x, y) = \frac{ax+b}{cx+d}.$$

Or on sait aussi que pour tout i tel que $w_i \neq 0$, on a

$$\frac{f}{h}(P_i) = \frac{v_i}{w_i} = \frac{ax_i + b}{cx_i + d}. \tag{3.18}$$

Tous les v_i et w_i sont connus. Donc en connaissant les abscisses x_i de trois points (mettons, $P_{k_1}, P_{k_2}, P_{k_3}$), alors on peut déterminer les constantes a, b, c, d à partir de l'équation précédente appliquée aux indices k_1, k_2, k_3. A partir de ces constantes, on peut maintenant utiliser les équations pour retrouver les abscisses de nombreux points ($n - k - g - 2$ précisément).

Il nous faut maintenant retrouver les ordonnées des points d'évaluation. Pour cela, on va interpoler les points de la courbes par des polynômes de degré $g = 2$.

On cherche 5 indices différents $k_1, \ldots, k_5$, tels que l'on puisse vérifier dans $\mathcal{G}$: $z_{k_1} + z_{k_2} + z_{k_3} + z_{k_4} + z_{k_5} = 0$. De telles sommes sont faciles à trouver en additionnant des éléments au hasard dans $\mathcal{G}$. En effet, $\mathcal{G}$ est de cardinal environ q^2, et il existe $C_n^5 \gg q^2$ sommes différentes[2]. On peut donc espérer trouver une somme nulle de 5 indices différents en $O(n^2)$ opérations.

Si on a $z_{k_1} + z_{k_2} + z_{k_3} + z_{k_4} + z_{k_5} = 0$, on en déduit que le diviseur $\langle P_{k_1} \rangle + \cdots + \langle P_{k_5} \rangle - 5\langle \mathcal{O} \rangle$ est principal. La fonction associée est un polynôme de degré au plus 2 en x, exactement 1 en y, qui s'annule en $P_{k_1}, \ldots, P_{k_5}$. Ces 5 points vérifient donc la même équation de la forme $y = ax^2 + bx + c$, où a, b, c sont des constantes dans $\mathbb{F}_q$.

Si l'on suppose maintenant que l'on connait toutes les abscisses $x_{k_1}, \ldots, x_{k_5}$ et trois des ordonnées $y_{k_1}, y_{k_2}, y_{k_3}$, alors on peut utiliser l'équation précédente pour calculer deux ordonnées manquantes y_{k_4}, y_{k_5}.

En utilisant une autre somme nulle de 5 éléments dans $\mathcal{G}$, on peut utiliser les ordonnées déja calculées pour en déduire d'autres, et réitérer le processus jusqu'à connaitre les ordonnées d'un grand nombre de points. En choisissant bien les indices, il est raisonnable d'espérer retrouver les coordonnées de nombreux points en ne connaissant au départ que les ordonnées de 4 d'entre eux.

Une fois toutes ces équations établies, on peut, en supposant connues 3 abscisses et 4 ordonnées, retrouver les coordonnées d'un grand nombre de points d'évaluation. Il suffit de 8 points pour définir une courbe de genre 2 de façon unique. On peut donc calculer les coordonnées d'une douzaine de points et vérifier a posteriori la validité de notre supposition sur les valeurs des 3 abscisses et des 4 ordonnées. Si l'on obtient une courbe de genre 2 passant par tous les points, alors avec une très forte probabilité, on obtient la courbe $\mathcal{X}$ recherchée et les coordonnées calculées sont justes. Si l'on n'arrive pas à construire une courbe passant par tous ces points, alors notre supposition était fausse, il faut reprendre les calculs avec un nouveau jeu de valeurs pour ces 3 abscisses et 4 ordonnées.

[2]En fait, l'argument est plus profond : Une telle somme existe dans la Jacobienne si, en remplaçant y par $ax^2 + bx + c$ dans l'équation de la courbe $\mathcal{X}$, le polynôme de degré 5 en x obtenu est scindé sur $\mathbb{F}_q$, ce qui arrive une fois sur 5 !.

Algorithme 6 : Deuxième étape

Input : $n, k, g = 2, G, d_1, \ldots, d_4, z_1, \ldots, z_n, \delta_0$.
Output : F et G intervenant dans l'équation de $\mathcal{X}$, liste des coordonnées de certains points d'évaluations (k_i, x_{k_i}, y_{k_i})

Initialisation : *On construit* $I \subset [1, n]$ *de taille* $k + g - 3$ *tel que* $\sum_{i \in I} z_i = (k + g - 1)\delta_0$.
On sélectionne $(i_1, i_2, j_1, j_2) \in [1, n] - I$ *deux à deux distincts tels que* $z_{i_1} + z_{i_2} = z_{j_1} + z_{j_2} = 0$.
On construit $\mathbf{v}, \mathbf{w} \in \mathcal{C}^*$ *dont les positions nulles sont exactement respectivement* $I \cup \{i_1, i_2\}$ *et* $I \cup \{j_1, j_2\}$.
On construit un agenda d'évaluation AE, c'est à dire une suite finie d'indices $k_1, \ldots, k_{12} \in [1, n] - (I \cup \{j_1, j_2\})$ *et de quintuplets d'indices* $E_5, \ldots, E_{12} \in [1, n]^5$ *associé tels que :*
- *Pour tout* t, $k_t \notin I \cup \{j_1, j_2\}$
- *Pour tout* $t \geq 5$, *les 5 indices de* E_t *correspondent à une somme nulle de 5* z_i.
- *Pour tout* $t \geq 5$, E_t *contient l'indice* k_t *et au moins 3 indices présents avant* k_t *dans la suite* $k_1, \ldots, k_{12}$
- $z_{k_1}, \ldots, z_{k_{12}}$ *engendrent le groupe* $\mathcal{G}$

Procédure d'essai : *On choisit* $x_{k_1}, x_{k_2}, x_{k_3}, y_{k_1}, y_{k_2}, y_{k_3}, y_{k_4} \in \mathbb{F}_q$.
On utilise les valeurs $x_{k_1}, x_{k_2}, x_{k_3}$, *ainsi que les mots* $\mathbf{v}, \mathbf{w}$ *pour déterminer les valeurs de* $x_{k_4}, \ldots, x_{k_{12}}$.
On calcule dans l'ordre les valeurs de $y_{k_5}, \ldots, y_{k_{12}}$ *en utilisant l'agenda d'évaluation, les* x_{k_t} *et les* y_{k_t} *déja calculés.*
Une courbe de genre g passe-t-elle par les points $P_{k_1}, \ldots, P_{k_{12}}$ *?*
Si oui, on retourne l'équation de la courbe et les coordonnées des points.
Si non, on recommence la procédure d'essai avec de nouvelles valeurs.

Ce qui rend cette étape particulièrement technique, c'est que l'on doit établir l'ordre de calcul des coordonnées des points de la courbe avant de commencer les calculs proprement dits. L'algorithme commence donc par la construction d'un agenda d'évaluation. Cet agenda d'évaluation se compose d'une liste de couples (indice, équation). Cette liste représente à la fois l'ordre dans lequel seront calculées les ordonnées des points d'évaluation, et, pour chaque point, l'équation sur 5 éléments de la Jacobienne permettant de cal-

culer son ordonnée. L'agenda d'évaluation peut être construit, une fois pour toutes, à partir des valeurs des z_i dans $\mathcal{G}$, en $O(n)$ opérations dans $\mathcal{G}$. L'algorithme 6 explicite la façon dont est construit cet agenda d'évaluation.

Une fois l'agenda construit, on peut passer à la procédure d'essai. On choisit arbitrairement les abscisses des trois premiers points de l'agenda, les ordonnées des quatre premiers points. On calcule les abscisses de tous les points de l'agenda par l'équation 3.18. On calcule ensuite les ordonnées dans l'ordre de l'agenda, en utilisant les équations de l'agenda. On vérifie enfin si notre supposition était correcte en tentant d'interpoler les points de l'agenda par une courbe hyperelliptique de genre 2. En cas d'échec, on recommence avec un nouveau jeu de 7 valeurs.

On croirait qu'il faut en moyenne q^7 suppositions pour obtenir la courbe $\mathcal{X}$. Mais c'est ici que l'on utilise les isomorphismes de courbes pour réduire ce nombre. En effet, le corollaire 4 nous garantit que 2 abscisses et 3 ordonnées peuvent être choisies librement, et que le calcul a alors une chance sur deux d'aboutir. Il faut donc en moyenne $2q^2$ suppositions pour obtenir la courbe, ce qui donne à cette étape une complexité $O(n^2)$ opérations dans $\mathbb{F}_q$.

3.4.4 Troisième étape : Retrouver les coordonnées des points d'évaluation restants

On connait l'équation de la courbe hyperelliptique $\mathcal{X}$, ainsi que les coordonnées d'une douzaine de points P_{k_t} sur $\mathcal{X}$. On connait aussi les valeurs de tous les $z_i = \varphi(\langle P_i \rangle)$ où φ est un isomorphisme inconnu.

La troisième étape est alors assez facile. Pour chaque P_i dont les coordonnées sont inconnues, on écrit z_i comme une somme de z_{k_j} (on a imposé que ces éléments engendrent le groupe de la Jacobienne). On calcule la même somme avec des couples de points sur la courbe $\mathcal{X}$, et on obtient ainsi les coordonnées de P_i.

La valeur du diviseur Δ_0 est calculée de la même manière à partir de la valeur de $\delta_0 = \varphi(\Delta_0 - \langle \mathcal{O} \rangle)$.

En précalculant une base de la Jacobienne, et en utilisant des algorithmes d'exponentiation rapide, le coût de cette étape peut être réduit à $O(n \log n)$ multiplications sur le corps de base.

3.4.5 Quatrième étape : Calculer les constantes de multiplication directionnelle

Maintenant que l'on connait tout le reste, calculer les constantes directionnelles c_i est un simple problème d'algèbre linéaire, qui peut être résolu en $O(n^3)$ par la même méthode que pour le genre 0 :

On commence par calculer une matrice de parité H du code $\mathcal{C}$. On construit $\mathbf{x} = (x_1, \ldots, x_n)$ un mot du code $\mathrm{AGC}(\mathcal{X}, (k+g-1)\Delta_0, \mathbf{P})$. Alors $\mathbf{cx} \in \mathcal{C}$, donc $H\mathbf{cx}^T = 0$, soit pour tout $i \leq n-k$,

$$\sum_{j=1}^{n} h_{ij} c_j x_j = 0.$$

Chacune des $n-k$ lignes de la matrice de parité nous fournit une équation linéaire sur les $c_1, \ldots, c_n$. En utilisant plusieurs mots de code, on obtient suffisamment d'équations pour résoudre le système linéaire, et calculer ainsi les constantes de multiplication directionnelle.

3.5 Genres supérieurs

L'attaque développée pour le genre 2 peut, au prix d'adaptations mineures, être utilisée pour cryptanalyser un système utilisant des codes géométriques de genre $g \geq 3$. Comme nous allons le voir, la majeure partie de l'algorithme reste inchangée, au moins en terme de complexité. Cependant, quelques étapes voient leur complexité augmenter avec la valeur de g. Nous considèrerons donc le genre g comme une variable intervenant dans la complexité, en supposant néanmoins pour les calculs $g \ll n-k$.

Nous conservons également deux des hypothèses formulées en genre 2. Tout d'abord, on suppose que le code contient beaucoup de points d'évaluation, i.e., que n est proche du nombre de points rationnels de $\mathcal{X}$. On suppose de plus que le cardinal de la jacobienne de la courbe et le degré du diviseur du code sont premiers entre eux, soit $(\gcd(k+g-1, |\mathcal{G}|) = 1)$. Les arguments qui nous ont conduits à adopter ces hypothèses en genre 2 conservent leur validité en genre supérieur.

La troisième hypothèse que nous voudrions utiliser concerne la facilité du cryptanaliste à trouver des mots de code de poids faibles. Une étude est

nécessaire pour évaluer la possibilité et le coût de cette recherche.

Théorème 15. *Soit $\mathcal{C}$ un code géométrique de longueur n et de dimension k sur $\mathbb{F}_q$, connu uniquement par une matrice génératrice G. On suppose que $\mathcal{C}$ est construit sur une courbe dont le genre g est connu, et vérifie les deux premières hypothèses ci-dessus. Alors, l'algorithme suivant permet, avec grande probabilité, de construire $\mathcal{X}, \Delta \in Div(\mathcal{X}), \boldsymbol{P} \in \mathcal{X}(\mathbb{F}_q)$, et $\boldsymbol{c} \in \mathbb{F}_q^n$ tels que $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, \boldsymbol{P}, \boldsymbol{c})$.*

La complexité de l'algorithme est alors $O(n^3 g!(\frac{q}{n-k-g/2})^g + q^{g/2})$ opérations dans $\mathbb{F}_q$.

Cette attaque n'a été encore ni publiée ni présentée.

3.5.1 Coût de la recherche d'un mot de code de poids faible

Soit $\mathcal{C} = \mathrm{AGC}(\mathcal{X}, \Delta, (P_1, \ldots, P_n))$ un code géométrique de dimension k basé sur une courbe de genre g. On suppose que le diviseur Δ est équivalent à un diviseur de la forme $(k+g-1)\Delta_0$, où Δ_0 est de degré 1.

On essaie de trouver des mots de code de poids $n-k-g+1$ par une méthode de décodage par Ensemble d'Information. Par un procédé de diagonalisation, on écrit la matrice génératrice du code sous la forme :

$$G = \left(\begin{array}{ccc|c} 1 & & 0 & \\ & \ddots & & * \\ 0 & & 1 & \end{array}\right). \tag{3.19}$$

On considère, par exemple, le mot de code $\mathbf{x}$ obtenu en lisant la première ligne de cette matrice. Ce mot est par construction nul en $k-1$ positions, d'indices $2, \ldots, k$. Le mot $\mathbf{x}$ est nul sur g positions suivantes si et seulement si il existe une fonction rationnelle $f \in \mathcal{L}(\Delta)$ qui s'annule sur les points $P_2, \ldots, P_k$, ainsi que sur g points parmi $P_{k+1}, \ldots, P_n$. Une telle fonction existe si et seulement si il existe $i_1, \ldots, i_g \in [k+1, n]$ tels que :

$$\langle P_2 \rangle + \cdots + \langle P_k \rangle + \langle P_{i_1} \rangle + \cdots + \langle P_{i_g} \rangle - \Delta \tag{3.20}$$

est principal.

On note $\tilde{z_i}$ la classe d'équivalence du diviseur de degré 0 $(\langle P_i \rangle - \Delta_0)$ dans la jacobienne de la courbe $\mathrm{Jac}(\mathcal{X})$. Alors le mot $\mathbf{x}$ est de poids $n-k-g+1$ si et seulement si il existe $i_1, \ldots, i_g \in [k+1, n]$ tels que

$$\tilde{z_{i_1}} + \cdots + \tilde{z_{i_g}} = -\tilde{z_2} - \cdots - \tilde{z_k}. \tag{3.21}$$

La partie droite de l'égalité correspond à un élément fixé de la Jacobienne, dont le cardinal est environ égal à q^g. La partie gauche correspond au choix de g points parmi $n-k$. Par conséquent, la probabilité que $\mathbf{x}$ est de poids $n-k-g+1$ vaut $p \approx \frac{C_{n-k}^g}{q^g}$.

Sous l'hypothèse $g \ll n-k$, on en déduit $p \approx \frac{1}{g!}(\frac{n-k-g/2}{q})^g$.

Une diagonalisation de la matrice génératrice de $\mathcal{C}$ se fait en $O(k^2 n)$ et fournit k candidats pouvant être des mots de poids faible. Par conséquent, la recherche d'un mot de poids $n-k-g+1$ par cette méthode coûte en moyenne $O(\frac{kn}{p})$, soit $O(kng!(\frac{q}{n-k-g/2})^g)$ opérations sur le corps de base. Tant que le genre de la courbe reste relativement petit, il est parfaitement raisonnable d'espérer trouver des mots de poids faible par cette méthode.

3.5.2 Algorithme de cryptanalyse

Notre algorithme de cryptanalyse peut maintenant être généralisé en genre $g \geq 3$. Il y a peu de changements à effectuer pour cela, mais la complexité varie selon le genre de la courbe. Puisque l'algorithme a déja été détaillé en genre 2, nous nous contenterons d'expliciter les différences.

Première étape : Retrouver la structure de la Jacobienne

D'après le théorème 4, on sait que $\mathrm{Jac}(\mathcal{X}) \simeq \mathcal{G} = \frac{\mathbb{Z}}{d_1\mathbb{Z}} \times \cdots \times \frac{\mathbb{Z}}{d_{2g}\mathbb{Z}}$. On appelle φ l'isomorphisme inconnu de $\mathrm{Jac}(\mathcal{X})$ vers $\mathcal{G}$.

De plus, de par la condition $\gcd(k+g-1, |\mathcal{G}|) = 1$, on sait que le diviseur Δ est équivalent à un diviseur de la forme $(k+g-1)\Delta_0$, avec Δ_0 de degré 1.

On pose $z_i = \varphi(\langle P_i \rangle - \langle \mathcal{O} \rangle)$, $\tilde{z}_i = \varphi(\langle P_i \rangle - \Delta_0)$, et $\delta_0 = \varphi(\Delta_0 - \langle \mathcal{O} \rangle)$.

De la même manière qu'en genre 2, si $\mathbf{x}$ est un mot de code de poids $n-k-g+1$, les positions des zéros de $\mathbf{x}$ ne sont pas aléatoires, et permettent de déduire dans $\mathcal{G}$ une équation concernant les $\tilde{z}_i$.

Avec suffisamment d'équations, on retrouve les valeurs de $d_1, \ldots, d_{2g}$, ainsi que les valeurs des $\tilde{z}_i$. La même méthode qu'en genre 2 (calcul d'invariants de Smith, par calcul de $O(g)$ déterminants $n \times n$) s'applique ici, en $O(gn^3)$. Il nous faut $O(n)$ mots de poids faible, leur recherche coûte donc en moyenne $O(n^3 g!(\frac{g}{n-k-g/2})^g)$ opérations sur le corps de base.

On peut maintenant déterminer δ_0 (toujours par recherche statistique sur les $\tilde{z}_i + \tilde{z}_j$), et en déduire les valeurs des z_i, le tout en temps $O(n^2)$.

Deuxième étape : Retrouver l'équation de la courbe

Les équations pour retrouver les abscisses des points d'évaluation sont les mêmes qu'en genre 2 : on génère deux mots de code $\mathbf{v}$ et $\mathbf{w}$ de poids $(n-k-g+1)$, tels que $\mathbf{v}$ et $\mathbf{w}$ ont exactement $k+g-3$ positions nulles communes, et deux autres positions nulles sur des points opposés. On peut alors écrire : $\frac{v_i}{w_i} = \frac{ax_i+b}{cx_i+d}$, où a, b, c, d sont des constantes inconnues. Si l'on connait les valeurs de 3 abscisses $x_{k_1}, x_{k_2}, x_{k_3}$, on peut calculer ces constantes, puis déterminer les abscisses de nombreux autres points.

Cependant, les mots $\mathbf{v}$ et $\mathbf{w}$ sont plus difficiles à générer à mesure que g augmente. Le problème revient à trouver $I \subset [1, n]$ de taille $k+g-3$ tel que $\sum_{i \in I} z_i = (k+g-1)\delta_0$. En supposant que les sommes de $k+g-3$ points sont aléatoirement réparties sur la Jacobienne, un tel ensemble I existe avec une très forte probabilité (car il est raisonnable de supposer $C_n^{k+g-3} \gg |\mathcal{G}| \approx q^g$). Toujours sous la même hypothèse, en calculant des sommes au hasard et en utilisant le paradoxe des anniversaires, on devrait générer I en un temps moyen $O(\sqrt{|\mathcal{G}|}) = O(q^{g/2})$.

Les équations sur les ordonnées des points se compliquent également avec le genre. En effet, le polynôme y admet un pôle d'ordre $2g+1$ en $\mathcal{O}$. On cherche donc des sommes nulles de $2g+1$ points : $z_{k_1} + \cdots + z_{k_{2g+1}} = 0$. On en déduit alors que les points correspondants sont les zéros d'un polynôme de degré g en x et 1 en y : $y = \lambda_g x^g + \cdots + \lambda_0$. Il y a $g+1$ constantes à

déterminer dans ce polynôme, une équation de ce type nous permet donc, en connaissant $2g+1$ abscisses et $g+1$ ordonnées, de calculer g ordonnées inconnues.

La construction de l'agenda d'évaluation commence par le choix arbitraire de $g+2$ indices, que l'on note $k_1, \ldots, k_{g+2}$. Chaque somme nulle de $2g+1$ points permet d'ajouter g points à l'agenda d'évaluation, à condition que les $g+1$ restants en fassent déja partie. En utilisant 3 sommes nulles, l'agenda d'évaluation contient alors $4g+2$ indices. La complexité de génération est grossièrement égale à $O(q^{g/2})$ (voir plus loin).

La procédure d'essai prend en entrée, à chaque exécution, des valeurs arbitraires pour $x_{k_1}, x_{k_2}, x_{k_3}, y_{k_1}, \ldots, y_{k_{g+2}}$, et calcule les coordonnées complètes de $P_{k_1}, \ldots, P_{k_{4g+2}}$. D'après l'équation de la courbe $\mathcal{X}$, il suffit de $3g+3$ points pour définir celle-ci de façon unique. Il est donc très peu probable qu'un essai nous permette de reconstruire une courbe invalide. Les isomorphismes de courbes nous garantissent de retrouver une courbe valide en $O(n^2)$ essais.

Génération détaillée de l'agenda d'évaluation

Générer notre agenda d'évaluation revient à exhiber trois sommes nulles de $2g+1$ points dans la jacobienne, vérifiant quelques propriétés permettant d'enchainer les calculs des coordonnées. Tout d'abord, tous les indices concernés par ces sommes doivent appartenir au support des deux mots de poids faibles $\mathbf{v}$ et $\mathbf{w}$ générés plus haut. Cela restreint notre ensemble de choix à $n-k-g+1$ indices, mais ceci ne devrait pas être un problème.

La première somme nulle peut contenir $2g+1$ indices arbitrairement choisis dans le support. Par paradoxe des anniversaires, le coût de génération de cette somme est $O(q^{g/2})$. Parmi les $2g+1$ indices de cette somme, on en choisit arbitrairement $g+1$ qui constitueront l'initialisation de l'agenda d'évaluation. Le $(g+2^{\text{ieme}})$ point de l'initialisation de l'agenda sera déterminé par la seconde somme nulle.

La seconde somme nulle doit contenir g indices apparaissant dans la première somme, et les $g+1$ autres peuvent être choisis arbitrairement. Si une telle somme existe, on sait la trouver en $O(q^{g/2})$ opérations. Cependant, le nombre de sommes vérifiant ces conditions est en moyenne $n/(g+1)!$. Le $(g+2^{\text{ieme}})$ point de l'initialisation de l'agenda est choisi parmi les nouveaux indices introduits par la deuxième somme.

La troisième somme nulle doit contenir $g+1$ indices apparaissant dans les $3g+2$ déja construits, et g indices extérieurs. Une telle somme existe avec très forte probabilité, et on peut la trouver en $O(q^{g/2})$ opérations.

Tant que $(g+1)!$ est au plus du même ordre que n, on peut donc construire notre agenda d'évaluation en $O(q^{g/2})$ opérations. Si ce n'est pas le cas, on peut alléger la condition sur la seconde somme nulle, qui ne devra alors contenir que $g-1$ indices issus de la première somme. Cependant, dans ce cas, l'agenda d'évaluation est initialisé par $g+3$ points au lieu de $g+2$, et il faut alors effectuer $O(n^3)$ procédures d'essai au lieu de $O(n^2)$. À ce niveau-là, la procédure d'essai n'est pas limitante dans la complexité de l'algorithme, donc cet allègement ne modifie la complexité donné au théorème 15.

Troisième et quatrième étapes

Ici, l'algorithme fonctionne exactement comme en genre 2, avec une complexité $O(n^3)$.

3.6 Conclusion

Nous avons présenté dans ce chapitre une attaque polynomiale efficace ($O(n^4)$) contre les codes hyperelliptiques de genre 2. Cette attaque, entre autres, brise complètement les paramètres proposés par Janwa et Moreno dans [27] (un code [171,109,61] sur $\mathbb{F}_{2^7}$).

J'ai de plus montré que cette attaque se généralise à des codes définis sur des courbes de genre supérieur. L'algorithme présenté à la fin de ce chapitre est tout à fait efficace pour cryptanalyser un cryptosystème de McEliece basé des codes géométriques, tant que le genre est relativement faible. Utiliser des codes de genre suffisamment élevé pour neutraliser notre attaque paraît extrêmement difficile à gérer, et très peu optimal.

Les attaques présentées dans ce chapitre ne fonctionnent que sous certaines hypothèses raisonnables. On peut imaginer qu'un concepteur intelligent arriverait à rendre caduques nos hypothèses, mais une telle opération aurait un coût important sur l'efficacité du cryptosystème.

Ce que nous montrons surtout par nos attaques, c'est que les codes géométriques présentent une faiblesse structurelle importante. On peut en effet générer relativement facilement des mots de code de poids faible, par une version affaiblie de la propriété d'optimalité de ces codes. De plus, ces mots de poids faible correspondent à des éléments particuliers de la Jacobienne, et nous renseignent sur la structure du code. Nous montrons que cette faiblesse peut être exploitée par des attaques structurelles. Par conséquent, l'utilisation de codes géométriques en cryptographie semble pour le moins extrêmement risquée.

Chapitre 4

Métrique rang et applications

4.1 Introduction

La métrique rang a été proposée par Gabidulin en 1985 comme alternative à la métrique de Hamming [22]. Cette métrique est appelée ainsi car la distance correspond au rang d'un certain vecteur. Ainsi, la notion de position d'erreur n'a pas vraiment de sens, puisqu'une erreur de poids faible peut être non nulle sur toutes les positions du message.

Utiliser la métrique rang pour la détection d'erreurs est souvent peu intéressant, car les canaux réels de communication introduisent rarement une erreur dont le motif peut être modélisé efficacement en métrique rang. Cependant, il semble que les outils de cette métrique sont adaptés dans le cadre du *network coding* [44, 48].

Mais en cryptographie, la métrique rang prend tout son intérêt. Puisqu'une erreur de poids faible peut être étalée sur toutes les positions du mot de code, les algorithmes de décodage par ensemble d'information sont inutilisables. De fait, les meilleurs algorithmes de décodage général en métrique rang sont ceux d'Ourivski et Johannson [36], qui s'exécutent ou bien en $O(n^3 q^{(t-1)(m-t)+2})$, ou bien en $O(n^3 t^3 q^{(t-1)(k+1)})$. On peut donc espérer concevoir des cryptosystèmes dont la clé publique reste assez petite.

Dans cette optique, Gabidulin, Paramonov et Tretjakov ont proposé en 1991 dans [24], un système cryptographique utilisant la métrique rang. Par deux fois, il a été montré que ce cryptosystème était soumis à une attaque structurelle (par Gibson dans [25], puis par Overbeck dans [38]). Par deux fois, il a fallu adapter la définition et les paramètres du système afin de neu-

traliser cette attaque. Aussi, la communauté cryptographique considère avec méfiance le système GPT, et l'utilisation de la métrique rang en général. Cependant, l'attaque d'Overbeck n'est pas toute-puissante, et l'on peut s'en prémunir de façon simple, comme je le montre par la suite. Le cryptosystème GPT ainsi modifié utilise des clés publiques d'une vingtaine de kilobits seulement. Sa sécurité est toujours sujette à caution, mais, par la faible taille de sa clé publique, il représente une piste d'étude intéressante dans le domaine de l'utilisation des codes correcteurs en cryptographie.

Toujours dans cette optique, avec P. Loidreau, nous avons proposé et implémenté en MAGMA un autre cryptosystème utilisant la métrique rang, et le problème de reconstruction de polynômes linéaires. L'idée de notre système est de réduire la taille de la clé publique en ne masquant pas la structure du code de Gabidulin utilisée. Une partie de la matrice publique est alors énoncée sous forme systématique, sa taille est donc très faible. Le coeur de la clé publique est un mot situé à distance trop élevée du code de Gabidulin pour être décodé. Le décodage de ce mot est le secret utilisé par le destinataire pour déchiffrer.

Lorsque nous avons présenté notre cryptosystème dans [18], nous n'avions pas connaissance de la méthode d'attaque développée par Overbeck contre les systèmes utilisant des codes de Gabidulin. Aussi, les paramètres proposés alors rendent le système vulnérable à une telle attaque. Cependant, notre système résiste en fait assez bien à l'attaque Overbeck, et une petite adaptation des paramètres nous permet de proposer un système qui semble sûr, pour une clé publique d'environ 10 kilobits.

Par ailleurs, étant donnés ses applications possibles en cryptographie, je me suis également intéressé au problème du décodage en liste des codes de Gabidulin. Ces codes sont en effet très proches des codes de Reed-Solomon que l'on sait décoder en liste en temps polynomial (cf. [47] ou chapitre précédent). Or, on ne sait rien à ce sujet sur les codes de Gabidulin. Je n'ai pour ma part pas réussi à adapter les algorithmes de Sudan aux polynômes linéaires, mais j'ai pu établir dans [14] un équivalent de la borne de Johnson. Je montre en effet, en comptant les mots de code dans des boules, que si un algorithme de décodage en liste décode les codes de Gabidulin au-delà de la distance $t_0 = \frac{m+n}{2} - \sqrt{mk + \left(\frac{m-n}{2}\right)^2}$, alors cet algorithme ne peut pas s'exécuter en un temps polynomial en moyenne. Une autre application du même calcul est l'impossibilité de concevoir en métrique rang un schéma de signature du type décrit par Courtois, Finiasz et Sendrier dans [11].

Je commence par présenter les bases de la métrique rang dans la section 2, j'expose dans la section 3 mes résultats sur le décodage en liste des codes de Gabidulin, je rappelle dans la section 4 les résultats connus sur le système GPT, et je détaille dans la section 5 mes travaux sur le cryptosystème fondé sur le problème de reconstruction de polynômes linéaires.

4.2 Présentation de la métrique rang et des codes de Gabidulin

4.2.1 Métrique rang

Les définitions de la métrique rang et des codes de Gabidulin peuvent être trouvées par exemple dans [22].

Définition 28. *Soit $\mathbb{F}_q$ le corps fini à q éléments, et $\mathbb{F}_{q^m}$ une extension de ce corps.*
Soit $\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_n) \in \mathbb{F}_{q^m}^n$.
Pour tout j, on peut écrire $x_j = \begin{pmatrix} x_{1j} \\ \vdots \\ x_{mj} \end{pmatrix}$ comme vecteur de $\mathbb{F}_q$.
Visuellement, on voit alors $\mathbf{x}$ comme une matrice dans $\mathbb{F}_q$:

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

(x_{ij}) est alors une matrice à m lignes et n colonnes dans $\mathbb{F}_q$.
On définit alors le rang du vecteur $\mathbf{x}$ par : $rg(\mathbf{x}) \stackrel{def}{=} rg(x_{ij})$.

Proposition 20. *L'application $\mathbf{x}, \mathbf{y} \mapsto rg(\mathbf{x} - \mathbf{y})$ est une distance sur $\mathbb{F}_{q^m}^n$, appelée distance rang. En particulier, elle vérifie l'inégalité triangulaire.*

La métrique rang est plus grossière que la métrique de Hamming, c'est-à-dire, pour $\mathbf{x} \in \mathbb{F}_{q^m}^n$, on a $\mathrm{Rg}(\mathbf{x}) \leq HW(\mathbf{x})$. On peut évaluer la taille des

boules en métrique rang, en comptant les matrices à m lignes et n colonnes de rang fixé :

Proposition 21. *Soit* $\mathbf{y}$ *un vecteur de longueur* n *à coefficients dans* $\mathbb{F}_{q^m}$. *Soit* $\mathcal{S}(\mathbf{y}, t)$ *la sphère centrée en* $\mathbf{y}$ *et de rayon* t, *et* $\mathcal{B}(\mathbf{y}, t)$ *la boule centrée en* $\mathbf{y}$ *et de rayon* t. *On a :*

$$\#\mathcal{S}(\mathbf{y}, t) = \frac{\prod_{i,j=0}^{t-1} (q^n - q^j)(q^m - q^i)}{\prod_{i=0}^{t-1} q^t - q^i},$$

$$\#\mathcal{B}(\mathbf{y}, t) = \sum_{i=0}^{t} |\mathcal{S}(\mathbf{y}, i)|.$$

Corollaire 5. *Les cardinaux de* $\mathcal{S}(\mathbf{y}, t)$ *et de* $\mathcal{B}(\mathbf{y}, t)$ *ne dépendent pas du choix de* $\mathbf{y} \in \mathbb{F}_{q^m}^n$. *De plus, on peut vérifier les encadrements suivants :*

$$q^{(m+n-2)t-t^2} \leq \#\mathcal{S}(\mathbf{y}, t) \leq q^{(m+n+1)t-t^2},$$

$$q^{(m+n-2)t-t^2} \leq \#\mathcal{B}(\mathbf{y}, t) \leq q^{(m+n+1)t-t^2+1}$$

4.2.2 Algorithmes de décodage Ourivski-Johannson

Les codes linéaires étant définis de la même façon qu'au chapitre deux, on s'intéresse à la version rang du décodage borné d'un code linéaire quelconque. À cause de la définition de la métrique, les méthodes de décodage par ensemble d'information sont inapplicables.

En effet, la métrique rang "étale" les erreurs. Par exemple, le mot $(\alpha, \ldots, \alpha)$ est de rang 1, alors que toutes ses positions sont non nulles. Par conséquent, on ne peut pas décoder en déplaçant une fenêtre d'information et en espérant que l'erreur reste à l'extérieur. Il faut utiliser d'autres techniques, et c'est ce que proposent Ourivski et Johannson dans [36].

Proposition 22. *Soit $\mathcal{C}$ un code linéaire $[n,k]$ de distance rang minimale d engendré par la matrice G. Soit $\mathbf{y} = \mathbf{c} + \mathbf{e}$, avec $\mathbf{c} \in \mathcal{C}$, et $Rg(\mathbf{e}) \leq (d-1)/2$. Soit $\mathcal{C}'$ le code $[n, k+1]$ engendré par la matrice $G' = \begin{pmatrix} G \\ \mathbf{y} \end{pmatrix}$. Alors les vecteurs non nuls de rang minimum dans $\mathcal{C}'$ sont de la forme $\alpha\mathbf{e}, \alpha \in \mathbb{F}_{q^m}$.*

On peut donc décoder $\mathcal{C}$ en utilisant un algorithme de recherche de mot de poids faible sur $\mathcal{C}'$. Lorsque l'on a obtenu $\mathbf{e}' = \alpha\mathbf{e}$, on peut retrouver la valeur de α en constatant que $H\mathbf{e}'^T = \alpha H\mathbf{y}^T$.

Pour rechercher les mots de rang faible de $\mathcal{C}'$, on raisonne de la manière suivante :

Quitte à effectuer une permutation, on suppose que $[1, k+1]$ forme un ensemble d'information du code $\mathcal{C}'$. On peut alors considérer $G_{\text{syst}} = (I_{k+1}|R)$, et $H_{\text{syst}} = (-R^T|I_{n-k-1})$, les matrices génératrice et de parité de $\mathcal{C}'$ associées à cet ensemble d'information.

Les mots de $\mathbb{F}_{q^m}^n$ de rang inférieur ou égal à t sont, quant à eux, exactement les mots de la forme $(\beta_1, \ldots, \beta_t)U$, où les $(\beta_1, \ldots, \beta_t)$ sont des éléments de $\mathbb{F}_{q^m}$ qui forment une famille libre sur $\mathbb{F}_q$, et U est une matrice $t \times n$ à coefficients dans $\mathbb{F}_q$.

En écrivant $U = (U_1|U_2)$, un mot de code de rang $t \leq (d-1)/2$ correspond à une solution de l'équation $(\beta_1, \ldots, \beta_t)(U_2 - U_1R) = \mathbf{0}$, avec comme contrainte que les $(\beta_1, \ldots, \beta_t)$ sont linéairement indépendants sur $\mathbb{F}_q$. Il y a deux stratégies pour résoudre ce système :

- Énumération des bases : On énumère les familles libres $(\beta_1, \ldots, \beta_t)$. Par suite de symétries du système, on doit énumérer au plus $q^{(m-t)(t-1)}$ familles. Pour chacune d'entre elles, on tente de résoudre dans $\mathbb{F}_q$ le système précédent, à $mt(n-k-1)$ équations et nt inconnues.

- Énumération des coordonnées : On cherche plutôt une matrice $V = U_2 - U_1R$ qui ne soit pas de rang maximal (pour que son noyau à gauche soit non nul). On doit alors énumérer un ensemble de $q^{t(k+1)}$ matrices, et tester le rang de chacune. Lorsqu'une matrice de rang au plus $t-1$ est trouvée, on résout le système linéaire précédent.

Théorème 16 (Décodage d'Ourivski et Johannson)**.** *Soit $\mathcal{C}$ un code linéaire $[n,k]$ de distance rang minimale d, $\mathbf{y} \in \mathbb{F}_{q^m}$ un mot de longueur n et $t \leq$*

$(d-1)/2$ un entier. Alors on sait trouver, s'il existe, un mot du code $\mathcal{C}$ à distance rang au plus t du mot $\mathbf{y}$ de deux manières :

- *Par énumération des bases, en $O((k+t)^3 q^{(t-1)(m-t)+2})$.*
- *Par énumération des coordonnées, en $O((k+t)^3 t^3 q^{(t-1)(k+1)})$.*

Pour un code linéaire général, ces algorithmes sont les meilleurs connus à l'heure actuelle. Mais certains codes, comme les codes de Gabidulin, peuvent être décodés beaucoup plus rapidement si leur structure est identifiée.

4.2.3 Polynômes linéaires

La théorie des polynômes linéaires date de 1935 et est dûe à Oystein Ore, dans [34] et [35]. Celui-ci donne plusieurs algorithmes de calculs sur les polynômes linéaires. Dans ce paragraphe, on considère $\mathbb{F}_{q^m}$ comme un espace vectoriel de dimension m sur le corps $\mathbb{F}_q$.

Définition 29. *Un polynôme linéaire (ou q-polynôme) sur $\mathbb{F}_{q^m}$ est un polynôme de la forme :*

$$P(X) = a_k X^{q^k} + \cdots + a_\ell X^{q^\ell} + \cdots + a_1 X^q + a_0 X,$$

où $a_0, \ldots, a_k$ sont des éléments de $\mathbb{F}_{q^m}$. Si $a_k \neq 0$, l'entier k est appelé le q-degré de P, ou simplement degré.

Par la suite, on notera[1] $[\ell] = q^\ell$.

Proposition 23. *Muni des lois $+$ et $\circ$, l'ensemble des q-polynômes est un sous-anneau non commutatif de $\mathcal{L}(\mathbb{F}_{q^m})$, l'ensemble des applications $\mathbb{F}_q$-linéaires de $\mathbb{F}_{q^m}$ dans lui-même.*

Preuve . *Dans un $\mathbb{F}_{q^m}$-espace vectoriel, X^q est linéaire.*
En effet, $(a+b)^q = \sum_{\ell=0}^{q} C_q^\ell a^\ell b^{q-\ell}$
Or, $\forall \ell \in [1, q-1], C_q^\ell = \frac{q!}{\ell!(q-\ell)!}$ est divisible par q.
Donc $(a+b)^q = a^q + b^q$, donc X^q est linéaire. On généralise aux puissances supérieures par récurrence :

[1] Cependant, pour éviter une ambiguïté avec les notations de la section 4.4.2, si M est une matrice carrée, l'élévation de cette matrice à la puissance q^ℓ sera toujours notée M^{q^ℓ}. La notation $M^{[\ell]}$ désignera l'action du Frobenius sur chacun des éléments de la matrice.

$(a+b)^{q^{i+1}} = ((a+b)^{q^i})^q = (a^{q^i} + b^{q^i})^q = a^{[i+1]} + b^{[i+1]}$
Donc les q-polynômes sont des applications linéaires. De plus, X est un q-polynôme.
Si $A = \sum_{i=0}^{k} a_i X^{[i]}, B = \sum_{i=0}^{k'} b_i X^{[i]}$,
alors un calcul simple montre que $A \circ B = \sum_{i=0}^{k+k'} c_i X^{[i]}$, *où* $c_i = \sum_j a_j b_{i-j}^{[j]}$
$A \circ B$ *est donc un q-polynôme différent a priori de* $B \circ A$, *dont le degré est* $\deg_q(A) + \deg_q(B)$.
D'où la conclusion. $\diamondsuit$

Algorithmiquement, le point délicat est le calcul de $A \circ B$, à partir des coefficients de A et de B. Ce calcul est réalisé en $O(kk')$ multiplications dans $\mathbb{F}_{q^m}$.

Proposition 24 (Lien entre racines et degré)**.** *Si P est un q-polynôme de degré* $k \neq 0$, *alors dim*$(\ker(P)) \leq k$.
Inversement, si E est un sous-espace vectoriel de $\mathbb{F}_{q^m}$, *de dimension* $k \neq 0$ *sur* $\mathbb{F}_q$, *il existe un q-polynôme de degré k s'annulant sur E. De plus, si* $q = 2$, *on sait construire celui-ci en* $O(k^2)$ *multiplications dans* $\mathbb{F}_{q^m}$.

Preuve . *La première partie s'obtient par dénombrement des racines de P. Pour la réciproque, si E=Vect*$(v_1 \dots v_k)$, *on construit* $P_1 = X^q - v_1^{q-1} X$, P_1 *est de q-degré 1 et s'annulant en* v_1, $P_2 = (X^q - P_1(v_2)^{q-1} X) \circ P_1$, *etc. Alors* $P = P_n = (X^q - P_{n-1}(v_n)^{q-1} X) \circ \cdots \circ (X^q - v_1^{q-1} X)$ *s'annule sur E et est de q-degré k. Si* $q = 2$, *l'évaluation des coefficients de* P_n *se fait en* $O(k^2)$ *multiplications dans* $\mathbb{F}_{q^m}$. $\diamondsuit$

Proposition 25 (Divisions euclidiennes)**.** *Soit* A, B *deux polynômes linéaires,* $B \neq 0$, $\exists!(Q, Q', R, R')$ 4*-uplet unique de polynômes linéaires tel que*
$A(X) = Q(X) \circ B(X) + R(X), \deg_q(R) < \deg_q(B)$
$A(X) = B(X) \circ Q'(X) + R'(X), \deg_q(R') < \deg_q(B)$.
De plus, on peut construire ces q-polynômes en $O(\deg_q(B)(\deg_q(A) - \deg_q(B))$ *multiplications dans* $\mathbb{F}_{q^m}$.

Les algorithmes de division euclidienne sont décrits dans [34].

4.2.4 Codes de Gabidulin

On suppose maintenant $k \leq n \leq m$.
Si P est un polynôme linéaire et $\mathbf{x} \in \mathbb{F}_{q^m}^n$, on notera $P(\mathbf{x}) = (P(x_1), \ldots, P(x_n))$.

Définition 30 (Codes de Gabidulin)**.**
Soit $\mathbf{g} \in \mathbb{F}_{q^m}^n$ *une famille libre sur* $\mathbb{F}_q$ *(ie* $rg(\mathbf{g}) = n$*).*
Le code de Gabidulin de longueur n*, de dimension* k*, et de support* $\mathbf{g}$ *est l'ensemble des mots obtenus par évaluation d'un q-polynôme de degré au plus* $k-1$ *sur* $\mathbf{g}$ *:*

$$Gab(\mathbf{g}, k, n) \stackrel{def}{=} \{(P(g_1), \ldots, P(g_n)) = P(\mathbf{g}), \deg_q(P) \leq k-1\}$$

Les codes de Gabidulin sont en fait l'équivalent en métrique rang des codes de Reed Solomon pour la métrique de Hamming.

Proposition 26. *Gab*$(\mathbf{g}, k, n)$ *est un code linéaire optimal en métrique rang, i.e. c'est un code linéaire qui, à cardinal fixé, maximise la distance minimale. Sa distance minimale vaut* $n - k + 1$*.*

Preuve . *La distance minimale se déduit facilement de la propriété 24. Le code est optimal, car la métrique rang étant plus grossière que la métrique de Hamming, la distance rang minimale des codes linéaires vérifie l'inégalité de Singleton (cf. théorème 1)* $\diamondsuit$

Proposition 27. *Si* $\mathcal{C} = Gab(\mathbf{g}, k, n)$ *est un code de Gabidulin, alors le code dual* $\mathcal{C}^\perp$ *est aussi un code de Gabidulin, de longueur* n *et de dimension* $n-k$*. Son support* $\mathbf{h}$ *est une des solutions (au choix) du système de* $n-1$ *équations* $(\sum_{j=1}^{n} h_j g_j^{[i+1+k-n]})_{i=0\ldots n-2}$.

Une preuve de cette proposition est donnée dans [22].

4.2.5 Reconstruction de polynômes linéaires

Le problème de reconstruction de polynômes linéaires fut énoncé par Pierre Loidreau dans [29]. Le décodage des codes de Gabidulin est un cas particulier de celui-ci. De plus, la reconstruction des polynômes linéaires est

supposée difficile au-delà de la capacité de correction du code de Gabidulin associé, ce qui justifie son utilisation en cryptographie par la suite.

Définition 31 (**RPL**$(\mathbf{y}, \mathbf{g}, k, t)$)**.**
Trouver tous les couples (V, P) *de polynômes linéaires non nuls tels que* $\deg_q(V) \leq t$, $\deg_q(P) \leq k-1$, *et vérifiant* $V(y_i) = V \circ P(g_i)$, $i = 1, \ldots, n$.

Proposition 28. *Les polynômes linéaires* P *pour lesquels il existe* V *tel que* (V, P) *est une solution de* **RPL**$(\mathbf{y}, \mathbf{g}, k, t)$ *sont exactement ceux associés à un mot de* $Gab(\mathbf{g}, k, n)$ *dont la distance à* $\mathbf{y}$ *est plus petite que* t.

Preuve . *Si* (V, P) *est solution de* **RPL**$(\mathbf{y}, \mathbf{g}, k, t)$, *alors pour tout* i, $V(y_i) = V \circ P(g_i)$, *soit pour tout* i, $V(y_i - P(g_i)) = 0$. *Comme* $1 \leq \deg_q(V) \leq t$, *d'après la proposition 24, en posant* $e_i = y_i - P(g_i)$, *Vect*$(e_1, \ldots, e_n)$ *est un espace vectoriel de dimension au plus* t, *donc* $rg(\mathbf{e}) \leq t$. *Donc* $P(\mathbf{g}) = \mathbf{y} - \mathbf{e}$ *est un mot de* $Gab(\mathbf{g}, k, n)$ *dont la distance à* $\mathbf{y}$ *est plus petite que* t.
Réciproquement, si $\mathbf{y} = P(\mathbf{g}) + \mathbf{e}$, *et* $rg(\mathbf{e}) \leq t$, *alors on peut, toujours d'après la proposition 24, construire un* q*-polynôme* V *non nul tel que* $\deg_p(V) \leq t$ *et* $V(\mathbf{e}) = 0$. *Alors, pour* $i \leq n$, $V(y_i) = V(P(g_i)) + V(e_i) = V \circ P(g_i)$, *donc* (V, P) *est une solution de* **RPL**$(\mathbf{y}, \mathbf{g}, k, t)$. $\diamondsuit$

Résoudre **RPL**$(\mathbf{y}, \mathbf{g}, k, t)$ conduit à résoudre le système

$$V(y_i) = V \circ P(g_i), V \neq 0, i = 1, \ldots, n, \tag{4.1}$$

dont les inconnues sont les coefficients de polynômes linéaires $v_0, \ldots, v_t$, et $P_0, \ldots, P_{k-1}$. C'est un système quadratique de n équations à $k + t + 1$ inconnues. On ne connaît pas de résultat générique sur la complexité de tels systèmes (bien qu'une piste de résolution pourrait être l'utilisation de bases de Gröbner). On peut cependant linéariser celui-ci, obtenant ainsi le système suivant, plus large :

$$V(y_i) = N(g_i), i = 1, \ldots, n, V \neq 0, \tag{4.2}$$

dont les inconnues sont $V(X) = \sum_{i=0}^{t} v_i X^{[i]}$ et $N(X) = \sum_{i=0}^{k+t-1} n_i X^{[i]}$. Ce système est linéaire à n équations et $k + 2t + 1$ inconnues. Toute solution (V, P) de 4.1 donne une solution de 4.2 de la forme $(V, V \circ P)$. La réciproque n'est pas

forcément vraie. On cherche maintenant à résoudre le système 4.2 : On pose

$$\mathcal{V} = \begin{pmatrix} v_0 \\ \vdots \\ v_t \end{pmatrix}, \mathcal{N} = \begin{pmatrix} n_0 \\ \vdots \\ n_{k+t-1} \end{pmatrix},$$

$$S = \begin{pmatrix} g_1 & \dots & g_1^{[k+t-1]} & -y_1 & \dots & -y_1^{[t]} \\ \vdots & & \vdots & \vdots & & \vdots \\ y_n & \dots & g_n^{[k+t-1]} & -y_n & \dots & -y_n^{[t]} \end{pmatrix}.$$

Alors le système 4.2 se traduit de façon matricielle :

$$S \times \begin{pmatrix} \mathcal{N} \\ \mathcal{V} \end{pmatrix} = 0 \tag{4.3}$$

L'ensemble des solutions de 4.3 forme un $\mathbb{F}_{q^m}$-espace vectoriel $\mathcal{K}$. Selon la dimension de $\mathcal{K}$, on peut déduire des informations sur les solutions de $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, t)$:

- Si $\mathcal{K}$ est de dimension 0, alors la seule solution du système 4.3 est le vecteur nul. Celui ci se traduit en un polynôme nul, donc $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, t)$ n'a pas de solution.

- Si $\mathcal{K}$ est de dimension 1, toutes les solutions du système 4.3 sont colinéaires, et conduisent à la même solution de $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, t)$, à un facteur près dans V. On choisit donc une solution $(\mathcal{V}, \mathcal{N})$ du système linéarisé, et on effectue la division euclidienne à gauche : $N = V \circ q + R$. Si R est non nul, alors $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, t)$ n'a pas de solution ; si $R = 0$, alors $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, t)$ admet pour solution $(\lambda V, P), \lambda \in \mathbb{F}_{q^m} - \{0\}$.

- Si $\mathcal{K}$ est de dimension $s \geq 2$, le système 4.3 admet q^{ms} solutions mais il n'y a pas de lien évident entre celles-ci et les solutions de $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, t)$. La meilleure solution connue est d'essayer $q^{m(s-1)}$ solutions non colinéaires et de regarder lesquelles sont solutions de $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, t)$.

On peut noter que si $t > \frac{n-k}{2}$, c'est-à-dire si on se place au-delà de la capacité de décodage, S contient alors $k + 2t + 1 > n + 1$ colonnes pour n lignes, donc $\mathcal{K}$ est de dimension au moins 2. On se trouve alors toujours dans le troisième cas. Aussi, au-delà de la capacité de décodage, $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, t)$

est supposé un problème difficile, ce qui justifie son utilisation en cryptographie.

4.2.6 Algorithme de décodage des codes de Gabidulin

On s'intéresse maintenant au décodage en pratique des codes de Gabidulin. Pierre Loidreau, dans [29], a donné un algorithme permettant de trouver le mot de $\mathrm{Gab}(\mathbf{g}, k, n)$ le plus proche d'un mot donné $\mathbf{y}$, tant que sa distance est plus petite que la capacité de correction du code, $\frac{n-k}{2}$. J'ai implémenté en MAGMA cet algorithme, dont la spécificité est le précalcul de matrices utiles au décodage, afin de gagner du temps lors de l'exécution :

On suppose maintenant que $t = \frac{n-k}{2}$ et que $\mathbf{y}$ est de la forme $\mathbf{y} = P(\mathbf{g}) + \mathbf{e}$ où $rg(\mathbf{e}) \leq t$. On sait alors que $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, t)$ admet au moins une solution, et donc le système linéarisé aussi.
Réciproquement, soit (V, N) une solution du système linéarisé. On a alors :

$$V(\mathbf{y}) = N(\mathbf{g}),$$

soit

$$V(P(\mathbf{g})) + V(\mathbf{e}) = N(\mathbf{g}),$$

d'où

$$V(\mathbf{e}) = (N - V \circ P)(\mathbf{g}).$$

Or $rg(V(\mathbf{e})) \leq rg(\mathbf{e}) \leq \frac{n-k}{2}$, et $N - V \circ P$, de q-degré au plus $k+t-1 \leq \frac{n+k}{2} - 1$, ne peut avoir un rang si petit. Donc $N - V \circ P = 0$.

N'importe quelle solution non nulle de l'équation matricielle 4.3 permet alors d'obtenir P, par division euclidienne à gauche. De plus, comme une partie de S ne dépend que de $\mathbf{g}$, la résolution de cette équation peut être accélérée par des précalculs :
On écrit $S = \begin{pmatrix} G_1 & Y_1 \\ G_2 & Y_2 \end{pmatrix}$, où G_1 est une matrice carrée de taille $k + t$.
On veut alors résoudre $\begin{cases} G_1\mathcal{N} + Y_1\mathcal{V} = 0 \\ G_2\mathcal{N} + Y_2\mathcal{V} = 0 \end{cases}$
Or G_1 est inversible car les g_i sont indépendants sur $\mathbf{F}_q$. Le système est donc

équivalent au système suivant :

$$\begin{cases} \mathcal{N} = -G_1^{-1}(Y_1\mathcal{V}) \\ G_2(-G_1^{-1}(Y_1\mathcal{V})) + Y_2\mathcal{V} = 0 \end{cases}$$

Soit, en posant $U = -G_1^{-1}$, $T = -G_2G_1^{-1}$,

$$\begin{cases} \mathcal{N} = U(Y_1\mathcal{V}) \\ (TY_1 + Y_2)\mathcal{V} = 0 \end{cases} \tag{4.4}$$

Après un précalcul de U et T, on peut utiliser l'algorithme de décodage 7 :

Algorithme 7 : Berlekamp-Welch (codes de Gabidulin)

Input : $\mathbf{y}$ à distance au plus $\frac{n-k}{2}$ du code, U, T
Output : P polynôme linéaire tel que $\mathrm{Rg}(\mathbf{y} - P(\mathbf{g})) \leq \frac{n-k}{2}$

Résolution du système linéaire : *Calculer une solution $\mathcal{V}_0 \neq 0$ par multiplications matricielles et pivot de Gauss.*
Calculer $\mathcal{N}_0$ associée à partir de $\mathcal{V}_0$ par multiplications matricielles.

Division : *Calculer $P = V_0 \backslash N_0$, par division euclidienne à gauche.*
Retourner P.

Comme $n - k - t = t$ ou $t - 1$, la complexité de cet algorithme est en $O(t^2(k+t) + t^3 + t(k+t) + (k+t)^2 + t^2) = O(kt^2 + t^3 + k^2)$ multiplications dans $\mathbb{F}_{q^m}$.

Théorème 17 (Décodage des codes de Gabidulin)**.** *Soit $\mathcal{C} = Gab(\mathbf{g}, k, n)$ un code de Gabidulin de support connu. Soit $\mathbf{y} \in \mathbb{F}_{q^m}^n$ un mot reçu, et t un entier. Alors, si $t < \frac{n-k}{2}$, on sait décoder $\mathbf{y}$ dans $\mathcal{C}$ à distance t en $O(kt^2 + t^3 + k^2)$ multiplications dans $\mathbb{F}_{q^m}$. Sinon, le décodage à distance t est supposé de complexité exponentielle.*

4.3 Répartition des poids et décodage en liste

On a vu dans le chapitre précédent que les codes géométriques (et donc en particulier les codes de Reed-Solomon) admettent un algorithme de décodage

en liste en temps polynomial. Puisque les codes de Gabidulin partagent de nombreuses propriétés avec les codes de Reed-Solomon, il est naturel de se demander si ces codes peuvent également être décodés en liste.

Un algorithme de décodage en liste des codes de Gabidulin doit renvoyer la liste de tous les mots de code répondant au problème Décodage-rang en liste($\mathrm{Gab}_k(\mathbf{g}), \mathbf{y}, t$). Pour qu'un tel algorithme fonctionne en temps polynomial, la taille de la liste doit être majorée par un polynôme. Cette constatation semble triviale, cependant le nombre de mots contenus dans une sphère de rayon-rang d augmente exponentiellement avec d, et donc le nombre de mots de code également.

C'est pourquoi, à partir de maintenant, nous allons estimer le nombre de mots de code satisfaisant le problème de décodage Décodage-rang en liste($\mathrm{Gab}_k(\mathbf{g}), \mathbf{y}, t$), en fonction des valeurs de t, k, n, m definis auparavant. J'ai présenté ce travail dans [14] et [15].

4.3.1 Cas général

Le nombre total de mots de longueur n sur $\mathbb{F}_{q^m}$ vaut clairement :

$$|\mathbb{F}_{q^m}^n| = q^{mn}$$

Comme le code de Gabidulin considéré est un code linéaire de dimension k sur $\mathbb{F}_{q^m}$, le nombre total de mot de Gabidulin vaut :

$$|Gab_k(\mathbf{g})| = q^{mk}$$

Le nombre de mots à l'intérieur de la boule de décodage de rayon t $\mathcal{B}(\mathbf{y}, t)$ a été exprimé plus tôt, et on peut utiliser l'encadrement (5) :

$$q^{(m+n-2)t-t^2} \leq |\mathcal{B}(\mathbf{y}, t)| \leq q^{(m+n+1)t-t^2+1}.$$

Par conséquent, le nombre moyen de solutions au problème de décodage, quand $\mathbf{y}$ traverse l'espace $\mathbb{F}_{q^m}^n$ vaut :

$$D(t, k, n, m) = \frac{|Gab_k(\mathbf{g})| \times |E(B(\mathbf{y}, t))|}{|\mathbb{F}_{q^m}^n|},$$

dont on déduit

$$\frac{q^{mk} q^{(m+n-2)t-t^2}}{q^{mn}} \leq D(t, k, n, m) \leq \frac{q^{mk} q^{(m+n+1)t-t^2+1}}{q^{mn}},$$

$$q^{mk+tm+tn-2t-t^2-mn} \leq D(t,k,n,m) \leq q^{mk+tm+tn+t-t^2-mn+1}.$$

Comme nous l'attendions, à k, n, m fixés, ce nombre augmente exponentiellement avec t (en effet, on a $t \leq n \leq m$). Nous allons nous intéresser particulièrement à la borne t_0 telle que $D(t_0, k, n, m) = 1$. Si t est plus petit que cette borne, nous savons que le problème de décodage admet en moyenne moins d'une solution, et nous pouvons espérer mettre au point un algorithme de décodage en liste en temps polynomial. Mais, si t est plus grand que cette borne, la taille de la liste des solutions serait exponentielle en n. En conséquence, nous pouvons affirmer qu'il n'existe pas d'algorithme de décodage en liste des codes de Gabidulin à complexité polynomiale au délà de cette borne t_0.

4.3.2 Codes de longueur maximale

On utilise souvent les codes de Gabidulin à leur longueur maximale, ce qui signifie $m = n$. Dans ce cas, on peut réécrire l'encadrement précédent :

$$q^{nk-(n-t)^2-2t} \leq D(t,k,n,m) \leq q^{nk-(n-t)^2+t+1}.$$

On en déduit un encadrement de la borne t_0 :

$$n - 1 - \sqrt{nk - 2n + 1} \leq t_0 \leq n + 1/2 - \sqrt{nk + n + 5/4}.$$

Asymptotiquement, on obtient donc

$$t_0 \approx n - \sqrt{nk}.$$

Ce résultat est remarquable, car la borne obtenue est la même que la borne du décodage en liste des codes géométriques du chapitre précédent, ce qui souligne encore les similarités entre les codes de Gabidulin et de Reed-Solomon.

Cependant, bien que t_0 soit plus grand que la capacité de décodage $\frac{n-k}{2}$, cela ne prouve pas l'existence d'un algorithme de décodage en liste en temps polynomial, ce n'est qu'un argument dans cette direction.

En revanche, on peut affirmer de façon sûre qu'il est impossible de décoder les codes de Gabidulin en temps polynomial, au-delà de cette borne $n - \sqrt{nk}$, car la liste des solutions serait elle-même de taille exponentielle.

4.3.3 Codes de faible longueur

Dans ce cas, l'encadrement précédent nous fournit des inégalités plus compliquées sur la borne t_0 :

$$\frac{m+n-2}{2} - \sqrt{mk + (\frac{m-n}{2})^2 - m - n + 1} \leq t_0$$

et

$$t_0 \leq \frac{m+n+1}{2} - \sqrt{mk + (\frac{m-n}{2})^2 + \frac{2m+2n+5}{4}}.$$

Ce qui nous donne, asymptotiquement,

$$t_0 \approx \frac{m+n}{2} - \sqrt{mk + \left(\frac{m-n}{2}\right)^2}.$$

On peut au passage remarquer que

$$t_0 \geq n - \sqrt{nk}$$

et que l'égalité est atteinte pour $m = n$.

Donc, dans le cas général, la borne $n - \sqrt{nk}$ (la meilleure que l'on puisse espérer pour des codes de Gabidulin de longueur maximale) reste atteignable par des algorithmes de décodage en liste en temps polynomial. Il est peut-être même possible de tirer parti d'un alphabet de taille plus élevée pour décoder un peu plus loin.

4.3.4 Synthèse

Théorème 18. *Pour $t > t_0$ (l'expression de t_0 étant donnée plus haut), le nombre moyen de solutions au problème Décodage-rang en liste($Gab_k(\mathbf{g}), \mathbf{y}, t$) ne peut être minoré par un polynôme en les paramètres du système.*

Nous avons montré que, si l'on considère une boule autour d'un mot reçu, le nombre moyen de mots de Gabidulin à l'intérieur reste faible en moyenne, même lorsque le rayon de décodage devient plus élevé que la capacité de correction du code. Nous avons même exprimé une borne théorique ($n - \sqrt{nk}$) pour que cela reste vrai.

Cette borne, malgré la densité plus importante des mots en métrique rang, conserve la même expression que la borne de Johnson en métrique de Hamming (cf. théorème 7). C'est une similarité de plus entre les codes de Gabidulin et les codes de Reed-Solomon, et cela laisse espérer qu'il est possible de mettre au point des algorithmes de décodage en liste en temps polynomial pour les codes de Gabidulin.

Pour poursuivre dans cette voie, il serait intéressant d'étudier plus avant la distribution du nombre de mots de code dans une boule. Nous ne nous sommes intéressés qu'à la moyenne de cette distribution, son maximum apporterait un éclairage sur le problème. Il faudrait de plus mieux comprendre les structures algébriques des polynômes linéarisés, pour voir s'il est possible de les rendre bivariés.

4.3.5 Schéma de signature

Une autre conséquence de ce calcul est, qu'a priori, on ne peut pas adapter un cryptosystème à clé publique utilisant la métrique rang en schéma de signature.

Classiquement, si l'on dispose d'une fonction de hachage h, pour adapter un mécanisme à clé publique en schéma de signature, on procède de la façon suivante : si m est le message que je veux signer, je calcule $h(m)$, que je considère comme un chiffré. J'utilise donc la clé secrète pour "déchiffrer" $h(m)$, et je génère ainsi $s = k_s(h(m))$. Le destinataire vérifie la validité du message signé (m, s), en testant l'égalité $h(m) = k_p(s)$.

Lorsque l'on utilise un cryptosystème basé sur les codes linéaires, la clé publique est généralement un code $[n, k, d]$ donné par une matrice génératrice G_{pub}, et la clé secrète est un algorithme de décodage du code G_{pub} à distance au plus $t \leq d/2$.

Cependant, quand il s'agit d'inverser le mécanisme pour signer un message, l'expansion du code nous pose un problème. En effet, si l'on calcule $h(m)$, on obtient un mot aléatoire de longueur n, rien ne nous assure que ce mot est décodable par notre clé secrète.

En métrique de Hamming, Courtois, Finiasz et Sendrier proposent dans [11] une solution qui fonctionne avec les codes de Goppa : On implémente un compteur i, on calcule le mot de longueur n $h(m, i)$, et on essaye de le

décoder. En cas d'échec, on incrémente le compteur et on recommence. En cas de réussite, on génère le message signé $(m, i, k_s(h(m, i)))$. Finiasz montre qu'en choisissant bien les paramètres, on effectue en moyenne $t!$ essais avant d'obtenir un mot valide (i.e. à distance au plus t du code G_{pub}). Pour des paramètres de sécurité corrects, on peut choisir t de l'ordre de 10, et générer une signature en $10! \approx 2^{24}$ opérations, ce qui rend le schéma utilisable.

En métrique rang, on ne peut pas utiliser cette méthode. Si $\mathcal{P}$ est la probabilité qu'un mot aléatoire de longueur n sur $\mathbb{F}_{q^m}$ soit à distance rang au plus $t \leq d/2$ du code généré par G_{pub}, on veut que $\mathcal{P}$ soit la plus élevée possible. Le cas le plus favorable serait celui où G_{pub} génère un code de distance-rang optimale (MRD). Or, si le code est MRD, tous les calculs effectués sur les codes de Gabidulin restent valides, et on a :

$$\mathcal{P} = D(t, k, n, m) \leq q^{mk+tm+tn+t-t^2-mn+1}.$$

En utilisant $q \geq 2$, $k \leq n - 2t$ et $n \leq m$, on a :

$$\mathcal{P} \leq q^{t(n-m-t+1)+1},$$

soit

$$\mathcal{P} \leq 2^{-t^2+t+1}.$$

Cette probabilité décroit extrêmement vite quand t croît, et donc la complexité d'un schéma de signature basé sur la métrique rang explose, ce qui le rend très vite inutilisable.

4.4 Cryptosystème GPT

4.4.1 Système

Le cryptosystème GPT est la première tentative d'utilisation de la métrique rang en cryptographie. Une première version fut présentée dans [24]. Suite à l'attaque de Gibson ([25]), Gabidulin et Ourivski présentèrent dans [23] une version généralisée du système, utilisant une matrice de distorsion. C'est cette version que nous étudierons ici.

– **Génération des clés :** On se place sur le corps $\mathbb{F}_{q^m}$, on choisit S, une matrice $k \times k$ inversible, à coefficients dans $\mathbb{F}_{q^m}$. On choisit un support $\mathbf{g} = (g_1, \ldots, g_n) \in \mathbb{F}_{q^m}^n$, et on construit $G = (g_j^{[i]})$, la matrice $k \times n$

génératrice du code de Gabidulin de dimension k et de support $\mathbf{g}$. On choisit une matrice de distorsion Z, de taille $k \times t_1$, à coefficients dans $\mathbb{F}_{q^m}$. On choisit enfin T, une matrice inversible de taille $(n+t_1)\times(n+t_1)$, à coefficients dans $\mathbb{F}_q$. Puisque les coefficients de T sont dans le corps de base, T est une isométrie pour la métrique rang.
La clé publique est

$$G_{\text{pub}} = S(G|Z)T,$$

matrice de taille $k \times (n + t_1)$, ainsi que la valeur de la capacité de correction du code $(n - k)/2$.
La clé secrète est la décomposition précise de G_{pub}.

– **Chiffrement :** Comme pour un système de McEliece classique, on génère $\mathbf{e}$, un vecteur aléatoire de rang $(n - k)/2$. On chiffre le message $\mathbf{m}$ en calculant

$$\mathbf{c} = \mathbf{m}G_{\text{pub}} + \mathbf{e}.$$

– **Déchiffrement :** En supposant que l'on a reçu $\mathbf{c} = \mathbf{m}G_{\text{pub}} + \mathbf{e}$, on calcule $\mathbf{c}T^{-1} = (\mathbf{m}S(G|Z) + \mathbf{e})T^{-1}$. En tronquant les t_1 dernières colonnes de $\mathbf{c}T^{-1}$, un algorithme de décodage du code G permet de déterminer $\mathbf{m}S$, puis $\mathbf{m}$.

Remarque 4. *Dans la présentation originelle, on n'utilisait pas de matrice de distorsion. Suite à l'attaque proposée par Gibson, on a ajouté la matrice de distorsion Z au système.*

Remarque 5. *D'autres variantes du système GPT originel ont été développées pour contrer l'attaque de Gibson. Dans [2], Berger et Loidreau proposent l'utilisation d'un sous-code d'un code de Gabidulin. Dans [37], les auteurs proposent l'utilisation de codes rangs réductibles combinés à une matrice de distorsion. Ces variantes ne seront pas décrites ici.*

4.4.2 Attaque structurelle

Raphaël Overbeck a présenté dans [38] une attaque structurelle contre le cryptosystème GPT. Le concepteur peut parer cette attaque mais doit, pour ce faire, choisir des paramètres assez élevés.

L'idée de l'attaque d'Overbeck utilise des propriétés intrinsèques aux codes de Gabidulin, et peut donc être généralisée à tout système utilisant ces familles de codes. C'est pourquoi nous allons présenter cette attaque en détails :

Si $M = (m_{i,j})$ est une matrice à coefficients dans $\mathbb{F}_{q^m}$, on définit la matrice $M^{[\ell]}$ comme celle obtenue en appliquant l'automorphisme de Frobenius d'ordre l à chacun de ses éléments : $M^{[\ell]} = (m_{i,j}^{q^\ell})$.

Et on définit la matrice $\Lambda_\ell(M)$ comme une matrice bloc à $k(\ell+1)$ lignes et n colonnes, de la façon suivante :

$$\Lambda_\ell(M) = \begin{pmatrix} M \\ M^{[1]} \\ \vdots \\ M^{[\ell]} \end{pmatrix}.$$

Proposition 29. *Si M est une matrice génératrice du code de longueur n $Gab_k(\mathbf{g})$, et ℓ un entier inférieur ou égal à $n-k$, alors $M^{[\ell]}$ génère le code $Gab_k(\mathbf{g}^{[\ell]})$. En conséquence, $\Lambda_\ell(M)$ est de rang $k+\ell$ seulement, et une sélection de $k+\ell$ lignes indépendantes de $\Lambda_\ell(M)$ permet d'obtenir une matrice génératrice de $Gab_{k+\ell}(\mathbf{g})$.*

Remarque 6. *Si $\mathcal{C}$ est un code de Gabidulin de longueur n et de dimension k mais de support inconnu, on peut retrouver facilement le support de $\mathcal{C}$ à partir d'une matrice de parité H du code. En effet, $\Lambda_{k-1}(H)$ est de rang $n-1$, et si on choisit un vecteur $\mathbf{x}$ non nul dans son noyau, alors en calculant $\mathbf{g} = \mathbf{x}^{[1-k]} = \mathbf{x}^{[m-k+1]}$, on obtient un support de $\mathcal{C}$.*

Nous allons maintenant utiliser ces opérateurs pour attaquer le système GPT. Soit $G_{\text{pub}} = S(G|Z)T$ la clé publique de taille $k \times (n+t_1)$ du cryptosystème. On s'intéresse à $\mathcal{G}_{\text{pub}} = \Lambda_{n-k-1}(G_{\text{pub}})$. On a :

$$\begin{pmatrix} G_{\text{pub}} \\ G_{\text{pub}}^{[1]} \\ \vdots \\ G_{\text{pub}}^{[n-k-1]} \end{pmatrix} = \begin{pmatrix} S & 0 & \dots & 0 \\ 0 & S^{[1]} & \ddots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & & S^{[n-k-1]} \end{pmatrix} \left(\begin{array}{c|c} G & Z \\ G^{[1]} & Z^{[1]} \\ \vdots & \vdots \\ G^{[n-k-1]} & G^{[n-k-1]} \end{array}\right) T,$$

que l'on écrit

$$\mathcal{G}_{\text{pub}} = \mathcal{S}(\mathcal{G}|\mathcal{Z})T.$$

- La matrice $\mathcal{S}$ est carrée de taille $k(n-k)$, inversible car chacun de ses blocs diagonaux est inversible.
- La matrice $\mathcal{G}$ est de taille $k(n-k) \times n$, et de rang $n-1$, d'après la proposition 29.
- La matrice $\mathcal{Z}$ est de taille $k(n-k) \times t_1$, son rang s vérifie $s \leq \min(k(n-k), t_1)$. De plus, si Z est aléatoire, et que $t_1 << k(n-k)$, il y a une forte probabilité que $s = t_1$.
- La matrice $(\mathcal{G}|\mathcal{Z})$ est donc de taille $k(n-k) \times (n+t_1)$, et son rang est inférieur à $n+s-1$, mais a de bonnes chances d'être égal à $n+t_1-1$.
- La matrice T est carrée de taille $n+t_1$, à coefficients dans $\mathbb{F}_q$ (donc, en particulier $T^{[i]} = T$). Cette matrice est inversible.

En conséquence, la matrice $\mathcal{G}_{\text{pub}}$ est de taille $k(n-k) \times (n+t_1)$, et son rang a de fortes probabilités de valoir $n+t_1-1$.

Théorème 19. *Si le rang de $\mathcal{G}_{pub}$ est exactement $n+t_1-1$, alors un attaquant peut, en temps polynomial, forger une clé secrète associée à G_{pub}.*

Ce théorème se démontre en utilisant la proposition suivante :

Proposition 30. *Si le noyau à droite* $\ker(\mathcal{G}_{pub})$ *est de dimension 1, alors*

- *il existe* $\mathbf{h} = (h_1, \ldots, h_n)$*, vecteur de* $\mathbb{F}_{q^m}$ *de rang* n *tel que* $\ker(\mathcal{G}_{pub}) = \{T^{-1}(\alpha\mathbf{h}|\mathbf{0})^T | \alpha \in \mathbb{F}_{q^m}\}$.

- *De plus, si* $\mathbf{y} \in \ker(\mathcal{G}_{pub})$*, et que* Q *est une matrice carrée de taille* $(n+t_1)$*, inversible, à coefficients dans* $\mathbb{F}_q$*, telle que* $Q\mathbf{y} = (\mathbf{x}|\mathbf{0})^T$*, alors* Q *vérifie* $TQ^{-1} = \begin{pmatrix} A & B \\ 0 & D \end{pmatrix}$*, avec* A *et* D *carrées inversibles et de taille respectivement* n *et* t_1.

Preuve . *Si le noyau de $\mathcal{G}_{pub}$ est de dimension 1, cela implique que* $\ker(\mathcal{G}|\mathcal{Z})$ *est également de dimension 1, et donc que $\mathcal{Z}$ est de rang t. Par conséquent, en prenant* $\mathbf{h}$ *le support du dual de* $Gab_{n-1}(\mathbf{g})$*, on vérifie* $\ker(\mathcal{G}|\mathcal{Z}) = \{(\alpha\mathbf{h}|\mathbf{0})^T | \alpha \in \mathbb{F}_{q^m}\}$*. De plus, par les propriétés des codes de Gabidulin,* $\mathbf{h}$ *est de rang n.*

Soient $\mathbf{y}$ *et* Q *vérifiant les conditions ci-dessus. De par la structure de* $\ker(\mathcal{G}_{pub})$*, on peut écrire* $(\mathbf{x}|\mathbf{0})^T = Q\mathbf{y} = QT^{-1}(\alpha\mathbf{h}|\mathbf{0})^T$*. Si l'on écrit* QT^{-1} *comme une matrice par blocs :* $QT^{-1} = \begin{pmatrix} A' & B' \\ C' & D' \end{pmatrix}$*, alors on a* $C'\mathbf{h}^T = \mathbf{0}^T$*. Comme* C' *est à coefficients dans* $\mathbb{F}_q$ *et que* $\mathbf{h}$ *est de rang* n*, on en déduit que* C' *est nulle.* QT^{-1} *est donc triangulaire par blocs, donc par passage à l'inverse,* TQ^{-1} *également.* $\diamondsuit$

Ainsi, lorsque $\ker(\mathcal{G}_{\text{pub}})$ est de dimension 1, l'attaquant choisit $\mathbf{y}$ non nul dans $\ker(\mathcal{G}_{\text{pub}})$, puis construit une matrice Q à coefficients dans $\mathbb{F}_q$ vérifiant le second point de la proposition 30. La condition sur le rang de $\mathbf{h}$ dans la première partie de la proposition garantit l'existence d'une telle matrice inversible, et on peut la construire en temps polynomial par de simples opérations matricielles.

On écrit alors $G_{\text{pub}}Q^{-1} = S(GA|Z') = (SGA|SZ')$, ce qui revient à dire que les n premières colonnes de la matrice $G_{\text{pub}}Q^{-1}$ forment une matrice génératrice d'un code de Gabidulin de longueur n, de dimension k, et de support $\mathbf{g}' = \mathbf{g}A$ inconnu. On peut facilement calculer ce support, par exemple en utilisant la remarque 6. En écrivant une matrice génératrice de ce code sous sa forme systèmatique, on obtient $G' = GA$, puis on peut calculer S et Z' à partir de l'équation ci-dessus.

Proposition 31. *Si le noyau à droite de* $\mathcal{G}_{pub}$ *est de dimension 1, les matrices* Q, G', S *et* Z'*, construites en temps polynomial par la méthode ci-dessus, vérifient toutes les propriétés de la clé secrète associée à* G_{pub}*, a savoir :*

- S *est carrée inversible de taille* k*, à coefficients dans* $\mathbb{F}_{q^m}$*.*
- G' *est la matrice génératrice, écrite sous forme systématique, d'un code de Gabidulin sur* $\mathbb{F}_{q^m}$ *de longueur* n *et de dimension* k*.*
- Z' *est une matrice de taille* $k \times t_1$ *à coefficients dans* $\mathbb{F}_{q^m}$*.*
- Q *est carrée inversible de taille* $n + t_1$*, à coefficients dans* $\mathbb{F}_q$*.*
- $G_{pub} = S(G'|Z')Q$*.*

Remarque 7. *Cette méthode de cryptanalyse fonctionne aussi contre les autres variantes de GPT, ce qui permet d'attaquer les paramètres proposés dans [2] et [37].*

4.4.3 Paramètres

Comme P. Loidreau le montre dans [29], il est possible de neutraliser l'attaque proposée par Overbeck. En effet, tout le principe de cette attaque repose sur le fait que, si $\mathcal{G}_{\text{pub}}$ est de rang exactement $n + t_1 - 1$, alors n'importe quel élément de son noyau est un mot dual de $\mathbf{g}T$. Si la dimension du noyau est $1 + d$, il faut retrouver un des q^m mots duaux possibles parmi $q^{m(1+d)}$ éléments, ce que l'on ne sait pas faire en mieux que $O(q^{md})$ opérations.

Pour cela, l'idée est d'imposer un rang faible dans le choix de la matrice de distorsion Z :

Proposition 32. *Soit $G_{pub} = S(G|Z)T$ la clé publique de taille $k \times (n + t_1)$ du cryptosystème, et soit d un entier. Si on a*

$$1 \leq Rg(Z) \leq \frac{t_1 - d}{n - k},$$

alors la dimension de $\ker(\mathcal{G}_{pub})$ *est supérieure ou égale à $1 + d$.*

Preuve . *Par construction de $\mathcal{Z}$, on a $Rg(\mathcal{Z}) \leq (n-k)Rg(Z)$. Donc $Rg(\mathcal{G}_{pub}) = Rg(\mathcal{G}) + Rg(\mathcal{Z}) \leq n - 1 + (n - k)Rg(Z)$. Par conséquent, $dim(\ker(\mathcal{G}_{pub})) \geq 1 + t_1 - (n - k)Rg(Z) \geq 1 + d$.* $\diamondsuit$

Malheureusement, cette condition impose notamment de choisir $t_1 > n - k$, ce qui implique que la taille de la clé publique devient importante. Tout est néanmoins relatif, la clé publique reste de faible taille pour un cryptosystème issu de la théorie des codes.

Proposition 33. *En utilisant, par exemple, le jeu de paramètres $m = n = 24$, $k = 12$, $t_1 = 40$, $Rg(Z) = 3$, toutes les attaques connues contre le système GPT ont une complexité supérieure à 2^{80}, y compris l'attaque d'Overbeck. La taille de la clé publique est alors de 18432 bits, pour une capacité de transmission utile de 19%.*

Remarque 8. *Les variantes de GPT résistent moins bien à l'attaque Overbeck, et leur réparation provoque une augmentation très importante de la taille des clés publique.*

4.5 Système basé sur le problème de reconstruction de polynômes linéaires

Une idée pour concevoir des cryptosystèmes à base de codes utilisant une clé publique de petite taille est de ne pas masquer la structure du code utilisé. En s'inspirant de cette idée, Augot et Finiasz ont présenté dans [1] un système utilisant les codes de Reed-Solomon. Ce cryptosystème, attaqué par Coron dans [10], fut l'objet d'une réparation utilisant l'opérateur trace. Cependant, cette réparation n'améliore pas vraiment sa sécurité, ce système est donc inutilisable en métrique de Hamming.

Pour attaquer la seconde version du cryptosystème, on doit décoder plusieurs mots dans des codes de Reed-Solomon au-delà de la borne de Johnson. Mais les positions d'erreurs sont les mêmes sur tous ces mots. Le cryptanaliste peut tirer parti de ce fait, et décoder tous les mots en même temps en dépassant la capacité de correction théorique connue pour un seul mot.

Cette notion de "positions d'erreurs identiques" étant intrinsèque à la métrique de Hamming, nous avons eu l'idée avec P. Loidreau de reprendre cette idée en métrique rang, en utilisant des codes de Gabidulin. Nous avons ainsi mis au point et implémenté en MAGMA un cryptosystème fondé sur le problème de reconstruction de polynômes linéaires, qui fut présenté, avec plus ou moins de détails, dans [12], [13], [17], et [18]. Ce cryptosystème semble résister aux attaques développées par Coron, et ses paramètres peuvent être adaptés pour tenir compte de l'attaque d'Overbeck.

4.5.1 Un premier essai

Afin de clarifier le fonctionnement de notre cryptosystème, je commence par en donner une première version, sans l'opérateur trace. Comme on le verra, cette version afaiblie est cependant vulnérable à une attaque sur le chiffré. L'opérateur trace permettra par la suite de neutraliser cette attaque.

Paramètres

On se place sur le corps $\mathbb{F}_{q^m}$, on considère $\mathbf{g}$ le support d'un code de Gabidulin de longueur n et de dimension k. W est le rang d'une grosse erreur, telle que $\mathbf{RPL}(\mathbf{y}, \mathbf{g}, k, W)$ est supposé difficile, soit $W > \frac{n-k}{2}$.

ω est le rang d'une petite erreur, vérifiant $\omega \leq \frac{n-W-k}{2}$.

Génération des clés

On génère dans $\mathbb{F}_{q^m}$ un polynôme linéaire unitaire P de q-degré $k-1$, et un n-uplet d'erreur $\mathbf{E}$, de rang W. $\mathbf{E}$ peut aussi être vu comme une matrice binaire à m lignes et n colonnes, de rang W. On calcule $\mathbf{c} = P(\mathbf{g})$
La clé publique est $\mathbf{z} = \mathbf{c} + \mathbf{E}$, ainsi que le vecteur $\mathbf{g}$.
La clé privée est le couple $(P, \mathbf{E})$.

Chiffrement

Le message m est un mot de longueur $k-1$ sur $\mathbb{F}_{q^m}$, que l'on exprime sous la forme d'un polynôme linéaire $m(X) = m_0 X + m_1 X^q + \cdots + m_{k-2} X^{q^{k-2}}$, de degré inférieur ou égal à $k-2$. On génère aléatoirement $\alpha \in \mathbb{F}_{q^m}$ et $\mathbf{e}$ un vecteur d'erreur de rang ω. Le chiffré y est alors donné par :

$$\mathbf{y} = m(\mathbf{g}) + \alpha \mathbf{z} + \mathbf{e}$$

Déchiffrement

Afin de déchiffrer, on projette $\mathbf{y}$ orthogonalement à $\mathbf{E}$. Pour cela, on calcule R, une matrice $n \times (n-W)$ de rang $n-W$ et à coefficients dans $\mathbb{F}_q$, telle que $\mathbf{E}R = 0$. Alors, $\mathbf{y}R$ est un mot de longueur $n-W$, à distance-rang au plus ω du code $\mathrm{Gab}(\mathbf{g}R, k, n-W)$. Un algorithme de décodage appliqué à ce mot nous permet de retrouver le polynôme linéaire $Q = m + \alpha P$. Le coefficient directeur de ce polynôme est α, on peut donc calculer $m = Q - \alpha P$.

Attaque sur le chiffré

Ce cryptosystème ne peut cependant pas être utilisé directement. Il est en effet vulnérable à une attaque en $O(n^3)$ de type Berlekamp, par déchiffrement et linéarisation. Soit $\mathbf{z}$ la clé publique, et $\mathbf{y}$ un chiffré intercepté. On veut résoudre :

$$\exists m, \alpha, \mathbf{e}, \begin{cases} \mathbf{y} = m(\mathbf{g}) + \alpha \mathbf{z} + \mathbf{e} \\ \deg_q(m) \leq k-2, rg(\mathbf{e}) \leq \omega \end{cases}$$

La résolution de ce système est toujours équivalente (cf. proposition 24) à la résolution de :

$$\exists m, \alpha, V, \begin{cases} V(\mathbf{y}) = V \circ m(\mathbf{g}) + V(\alpha \mathbf{z}) \\ \deg_q(m) \leq k-2, \deg_q(V) \leq \omega, V \neq 0 \end{cases} \tag{4.5}$$

En linéarisant $V \circ m$ en N et $V \circ (\alpha X)$ en V', on obtient le système plus large :

$$\exists N, V, V', \begin{cases} V(\mathbf{y}) = N(\mathbf{g}) + V'(\mathbf{z}) \\ \deg_q(N) \leq k + \omega - 2 \\ \deg_q(V) \leq \omega, V \neq 0 \\ \deg_q(V') \leq \omega, V' \neq 0 \end{cases} \tag{4.6}$$

Ce système d'équations linéaires comporte n équations et $k + 3\omega + 1$ inconnues (les coefficients de N, V, V'). Or on a $\omega \leq \frac{n-W-k}{2}$ et $W > \frac{n-k}{2}$, dont on déduit $k+3\omega+1 < n$. Le système 4.6 admet une solution non triviale par construction, et puisqu'il est surdéterminé, son espace de solutions a de bonnes chances (vérifiées en pratique) d'être de dimension 1. Dans ce cas, à partir de n'importe quelle solution non triviale de ce système, on déduit

$$\alpha = V \backslash V'$$

par division euclidienne à gauche, et par suite, le message m. Le total des calculs aura coûté $O(n^3)$ multiplications dans $\mathbb{F}_q$. Des simulations en MAGMA montrent que cette attaque fonctionne rapidement en pratique.

Ce brouillon de système est vulnérable à une attaque car sa clé ne se compose que d'un seul vecteur. La version que nous avons imaginée, et qui est présentée juste ensuite, utilise une clé composée de u vecteurs à coefficients dans $\mathbb{F}_{q^m}$. Une façon efficace de faire les calculs consiste à se représenter cette clé plutôt comme un unique vecteur à coefficients dans $\mathbb{F}_{q^{mu}}$, et d'utiliser l'opérateur trace pour effectuer des projections de $\mathbb{F}_{q^{mu}}$ sur $\mathbb{F}_{q^m}$.

4.5.2 Opérateur trace

On considère maintenant $\mathbb{F}_{q^{mu}}$, vu comme un $\mathbb{F}_{q^m}$-espace vectoriel de dimension u.

Définition 32. *On définit l'opérateur trace de $\mathbb{F}_{q^{mu}}$ vers $\mathbb{F}_{q^m}$ par :*

$$\forall x \in \mathbb{F}_{q^{mu}}, Tr(x) = x + x^{q^m} + x^{q^{2m}} + \cdots + x^{q^{m(u-1)}}$$

Proposition 34. *La trace est une forme $\mathbb{F}_{q^m}$-linéaire de $\mathbb{F}_{q^{mu}}$ dans $\mathbb{F}_{q^m}$. De plus,*

$$\phi : \begin{array}{ccc} \mathbb{F}_{q^{mu}} & \rightarrow & \mathcal{L}(\mathbb{F}_{q^{mu}}, \mathbb{F}_{q^m}) \\ x & \mapsto & y \mapsto Tr(xy) \end{array}$$

est un isomorphisme de $\mathbb{F}_{q^m}$-espaces vectoriels.

Preuve . $(Tr(x))^{q^m} = x^{q^m} + \cdots + x^{q^{m(u-1)}} + x^{q^{mu}}$
$(Tr(x))^{q^m} = x^{q^m} + \cdots + x^{q^{m(u-1)}} + x$
$(Tr(x))^{q^m} = Tr(x)$, *donc* $Tr(x) \in \mathbb{F}_{q^m}$ $\diamondsuit$

Ainsi, si $(\gamma_1, \ldots, \gamma_u)$ est une base de $\mathbb{F}_{q^{mu}}$, alors $(Tr(\gamma_i \gamma_j))_{i,j}$ est une matrice carrée de taille u, inversible, et dont l'inverse permet de retrouver les coordonnées de $\alpha \in \mathbb{F}_{q^{mu}}$ dans la base $(\gamma_1, \ldots, \gamma_u)$ à partir du vecteur $(Tr(\gamma_1 \alpha), \ldots, Tr(\gamma_u \alpha))$.

L'opérateur trace s'étend canoniquement aux vecteurs eu aux polynômes linéaires de $\mathbb{F}_{q^{mu}}$:

$$Tr(\mathbf{x}) = (Tr(x_1), \ldots, Tr(x_n))$$

$$Tr(\sum_{i=0}^{k} p_i X^{[i]}) = \sum_{i=0}^{k} Tr(p_i) X^{[i]}$$

Proposition 35. *Si $(g_1, \ldots, g_n) \in \mathbb{F}_{q^m}^n$, et P polynôme linéaire de $\mathbb{F}_{q^{mu}}$, alors $Tr(P(\mathbf{g})) = (Tr(P))(\mathbf{g})$*

4.5.3 Paramètres

L'entier q^m est la taille du corps intermédiaire $\mathbb{F}_{q^m}$, u est le degré de l'extension, q^{mu} est la taille du corps d'extension $\mathbb{F}_{q^{mu}}$, n est la longueur du code de Gabidulin utilisé, k sa dimension, et $\mathbf{g} \in \mathbb{F}_{q^m}^n$ son support (n points choisis dans le corps intermédiaire, formant une famille $\mathbb{F}_q$-libre).
W est le rang d'une grosse erreur, telle que le problème de reconstruction de polynômes linéaires est supposé difficile, soit $W > \frac{n-k}{2}$.
ω est le rang d'une petite erreur, pour laquelle le problème de reconstruction de polynômes linéaires est facile, soit $\omega \leq \frac{n-W-k}{2}$.

4.5.4 Génération des clés

On génère dans $\mathbb{F}_{q^{mu}}$ un polynôme linéaire P de q-degré $k-1$, tel que les coefficients $p_{k-1}, \ldots, p_{k-u}$ forment une base de $\mathbb{F}_{q^{mu}}$ sur $\mathbb{F}_{q^m}$, on calcule ensuite $\mathbf{c} = P(\mathbf{g})$. On génère un n-uplet d'erreur $\mathbf{E}$, à coefficients dans $\mathbb{F}_{q^{mu}}$, de rang W ($\mathbf{E}$ peut aussi être vu comme une matrice à coefficients dans $\mathbb{F}_q$, à mu lignes et n colonnes, de rang W).

- La clé publique est $\mathbf{K} = \mathbf{c} + \mathbf{E} \in \mathbb{F}_{q^{mu}}^n$, ainsi que le vecteur $\mathbf{g} \in \mathbb{F}_{q^m}^n$. La clé publique est donc de taille $nm(u+1)\log_2(q)$, mais peut être réduite à $nmu\log_2(q)$ si l'expression de $\mathbf{g}$ est incluse dans les spécifications du système.
- La clé privée est le couple $(P, \mathbf{E})$.

4.5.5 Chiffrement

Soit m un message de longueur $k-u$ sur $\mathbb{F}_{q^m}$, que l'on exprime sous la forme d'un polynôme linéaire $m(X) = m_0 X + m_1 X^q + \cdots + m_{k-u-1} X^{q^{k-u-1}}$, de degré inférieur ou égal à $k-u-1$. On génère aléatoirement $\alpha \in \mathbb{F}_{q^{mu}}$ et $\mathbf{e}$ un vecteur d'erreur de $\mathbb{F}_{q^m}$ de rang ω. Le chiffré $\mathbf{y}$ est alors :

$$\mathbf{y} = m(\mathbf{g}) + Tr(\alpha \mathbf{K}) + \mathbf{e} \in \mathbb{F}_{q^m}^n$$

Le chiffrement est effectué en $O((k-u)n)$ multiplications dans $\mathbb{F}_{q^m}$ et $O(n)$ multiplications dans $\mathbb{F}_{q^{mu}}$. En estimant une multiplication dans $\mathbb{F}_{q^{mu}}$ équivalente à u^2 multiplications dans $\mathbb{F}_{q^m}$, l'algorithme de chiffrement a une complexité de $O((u^2+k)n)$ multiplications dans $\mathbb{F}_{q^m}$.

4.5.6 Méthode de déchiffrement

Comme pour le premier cryptosystème, on va commencer par raccourcir le chiffré reçu en le projetant orthogonalement au vecteur d'erreur $\mathbf{E}$. Ainsi le mot plus court obtenu ne contiendra plus d'interférence liée à $\mathbf{E}$, et pourra donc être traité par un algorithme de décodage des codes de Gabdulin. On suppose que l'on dispose d'une matrice R à coefficients dans $\mathbf{F}_q$, à n lignes et $n-W$ colonnes, de rang $n-W$, et telle que $\mathbf{E}R = 0$. On peut ensuite effectuer les calculs suivants dans $\mathbb{F}_{q^m}$:

$$\begin{aligned}
\mathbf{y}R &= m(\mathbf{g})R + Tr(\alpha P(\mathbf{g}))R + Tr(\alpha \mathbf{E})R + \mathbf{e}R \\
\mathbf{y}R &= (m + Tr(\alpha P))(\mathbf{g}R) + Tr(\alpha \mathbf{E}R) + \mathbf{e}R \\
\mathbf{y}R &= (m + Tr(\alpha P))\mathbf{g}R + \mathbf{e}R
\end{aligned}$$

Mais $rg(\mathbf{e}R) \leq rg(\mathbf{e}) \leq w \leq \frac{n-W-k}{2}$

Donc un algorithme de décodage de $\mathrm{Gab}(\mathbf{g}R, k, n-W)$ appliqué à $\mathbf{y}R$ renverrait le polynôme linéaire $Q = m + Tr(\alpha P)$ dans $\mathbb{F}_{q^m}$.

Pour appliquer l'algorithme de décodage 7, si on note $\mathbf{h}$ le vecteur $\mathbf{g}R$, il faut précalculer les matrices U et T liées à $\mathbf{h}$. Ces matrices ne dépendent que de la clé privée et peuvent donc effectivement être précalculées.

Le décodage effectué, on obtient le p-polynôme $Q = m + Tr(\alpha P)$. Or comme m est de degré au plus $k-u-1$, et que $p_{k-1}, \ldots, p_{k-u}$ forment une base de $\mathbb{F}_{q^{mu}}$ sur $\mathbb{F}_{q^m}$, pour $i = k-u \ldots k-1, q_i = Tr(\alpha p_i)$ sont les coordonnées de α dans la base duale de $p_{k-1}, \ldots, p_{k-u}$. On obtiendra la valeur de α en multipliant ces coordonnées par une matrice de passage, précalculée elle aussi. On peut ainsi déduire α et donc :

$$m = Q - Tr(\alpha P).$$

4.5.7 Précalculs avant déchiffrement

À partir de $\mathbf{E}$, on calcule une matrice binaire R, carrée, inversible, de taille n, telle que $\mathbf{E}R$ est nulle sur les $n-W$ premières colonnes. En fait, dans la pratique, la matrice R est générée en même temps que $\mathbf{E}$. En effet, le vecteur d'erreur $\mathbf{E}$ peut être vu comme une matrice à coefficients dans $\mathbb{F}_q$, à mu lignes, n colonnes, de rang W. On génère aléatoirement une telle matrice en générant 2 matrices Q_1 et Q_2, carrées inversibles, de taille respective mu et n. Ensuite, on obtient $\mathbf{E} = Q_1 D Q_2$ où D est la matrice diagonale de rang W. Ainsi, $\mathbf{E}$ est équirépartie sur l'ensemble des vecteurs de rang W, et on peut calculer facilement $R = Q_2^{-1}$, en temps $O(n^3)$.

On doit aussi calculer les matrices U et T utilisées dans l'algorithme vu en 1.5. Pour ce faire, on calcule le vecteur $\mathbf{h} = \mathbf{g}R$. Ensuite on peut calculer :

$$U = -\begin{pmatrix} h_1 & \dots & h_1^{[k+\omega-1]} \\ \vdots & & \vdots \\ h_{k+\omega} & \dots & h_{k+\omega}^{[k+\omega-1]} \end{pmatrix}^{-1},$$

$$T = -\begin{pmatrix} h_{k+\omega+1} & \dots & h_{k+\omega+1}^{[k+\omega-1]} \\ \vdots & & \vdots \\ h_{n-W} & \dots & h_{n-W}^{[k+\omega-1]} \end{pmatrix} \times U,$$

en respectivement $O((k+\omega)^3)$ et $O(\omega(k+\omega)^2)$ multiplications dans $\mathbb{F}_{q^m}$.

Enfin, il reste à calculer la matrice qui permettra de retrouver α à partir de ses coordonnées dans la base duale de $p_{k-u}, \dots, p_{k-1}$. Le plus simple est de vérifier que $p_{k-u}, \dots, p_{k-1}$ forment bien une base de $\mathbb{F}_{q^{mu}}$ sur $\mathbb{F}_{q^m}$ en tentant d'inverser la matrice $(Tr(p_i p_j))_{i,j\in[k-u,k-1]}$. Alors l'inverse de cette matrice permet de passer des coordonnées dans la base duale aux coordonnées dans la base $p_{k-u}, \dots, p_{k-1}$. Multiplier cette matrice par la matrice des coordonnées de $p_{k-u}, \dots, p_{k-1}$ nous donnera ce que l'on cherche, en $O(u^3)$ multiplications.

4.5.8 Algorithme de déchiffrement

L'étape la plus coûteuse de cet algorithme est la troisième, qui se fait en $O(\omega^2(k+\omega))$ multiplications dans $\mathbb{F}_{q^m}$. La cinquième étape utilise $O((k+\omega)^2)$ multiplications, la septième coûte $O(u^2)$ multiplications, et la dernière $O(kn)$ (mais celle-ci est facultative). Les autres étapes sont plus rapides quels que soient les paramètres. La complexité de l'algorithme de déchiffrement est donc $O(\omega^2(k+\omega) + (k+\omega)^2 + u^2 + kn)$.

Proposition 36 (Efficacité du système)**.** *Pour un jeu de paramètres q, m, u, n, k, W, ω fixés, le cryptosystème fondé sur le problème de reconstruction des polynômes linéaires atteint les performances suivantes :*

- *La clé publique est de taille $nm(u+1)\log_2(q)$, mais peut être réduite à $nmu\log_2(q)$.*
- *Le taux de transmission efficace vaut $\frac{k-u}{n}$.*
- *Le chiffrement s'effectue en $O((u^2+k)n)$ multiplications dans $\mathbb{F}_{q^m}$.*
- *En utilisant des précalculs, le déchiffrement s'effectue en $O(\omega^2(k+\omega) + (k+\omega)^2 + u^2 + kn)$ multiplications dans $\mathbb{F}_{q^m}$.*

Algorithme 8 : Algorithme de déchiffrement

Input : Les paramètres du cryptosystème n, k, u, W, ω, un n-uplet $\mathbf{y}$ d'éléments de $\mathbb{F}_{q^m}$, le support $\mathbf{g} \in \mathbb{F}_{q^m}$, la clé privée $(P, \mathbf{E})$, la matrice R de $\mathbb{F}_p$ carrée de taille n adaptée à $\mathbf{E}$, les matrices U et T dans $\mathbb{F}_{q^m}$ respectivement de tailles $(k+\omega)^2$ et $(n-W-k-\omega) \times (k+\omega)$, la matrice Ω dans $\mathbb{F}_{q^m}$ de taille u^2 de passage de la base duale de $p_{k-u}, \ldots, p_{k-1}$ à la base canonique de $\mathbb{F}_{q^{mu}}$ / $\mathbb{F}_{q^m}$.

Output : L'unique $(k-u)$-uplet m, s'il existe, correspondant au chiffré $\mathbf{y}$.

Algorithme :

1. Calculer les $n-W$ premiers éléments de $\mathbf{y}R$. On obtient ainsi le $n-W$ uplet $\mathbf{y}'$ de $\mathbb{F}_{q^m}$.
2. Calculer dans $\mathbb{F}_{q^m}$ les matrices Y_1 et Y_2, à partir de $\mathbf{y}'$:

$$Y_1 = \begin{pmatrix} y'_1 & \cdots & {y'_1}^{q^\omega} \\ \vdots & & \vdots \\ y'_{k+\omega} & \cdots & {y'_{k+\omega}}^{q^\omega} \end{pmatrix}, Y_2 = \begin{pmatrix} y'_{k+\omega+1} & \cdots & {y'_{k+\omega+1}}^{q^\omega} \\ \vdots & & \vdots \\ y'_{n-W} & \cdots & {y'_{n-W}}^{q^\omega} \end{pmatrix}$$

3. Calculer $M = T \times Y_1 + Y_2$
4. Déterminer par pivot de Gauss $\mathcal{V} \neq 0 \in \ker(M)$. Si $\ker(M) = \{0\}$, retourner `Pas de solution`
5. Calculer $\mathcal{N} = U \times (Y_1 \times \mathcal{V})$
6. Effectuer la division euclidienne à gauche par l'algorithme de Ore des polynômes linéaires N et V associés : $N = V \circ q + R$. Si $R \neq 0$, retourner `Pas de solution`
7. Calculer $\alpha = \Omega \begin{pmatrix} q_{k-u} \\ \vdots \\ q_{k-1} \end{pmatrix}$
8. Calculer $m = q - Tr(\alpha P)$
9. Vérifier que $rg(\mathbf{y} - m(\mathbf{g}) - Tr(\alpha(P(\mathbf{g}) + \mathbf{E}))) \leq \omega$. Si c'est le cas retourner m, sinon retourner `Pas de solution`

Nous allons maintenant étudier la résistance du système aux diverses attaques connues en métrique rang.

4.5.9 Attaque par décodage général

Soit $\mathbf{K} \in \mathbb{F}_{q^{mu}}^n$ la clé publique du système. Soit $\gamma_1, \ldots, \gamma_u$ une base fixée de $\mathbb{F}_{q^{mu}}$ sur $\mathbb{F}_{q^m}$.

On notera, pour toute la suite, $\mathbf{K}_t = Tr(\gamma_t \mathbf{K})$, pour $t = 1 \ldots u$.

Les vecteurs $\mathbf{K}_t$ sont des vecteurs de $\mathbb{F}_{q^m}$ aisément calculables à partir de la clé publique $\mathbf{K}$.

Soit $\mathbf{y} = m(\mathbf{g}) + Tr(\alpha \mathbf{K}) + \mathbf{e} \in \mathbb{F}_q^n$ un chiffré intercepté. Alors, décrypter ce chiffré revient à décoder $\mathbf{y}$ à distance-rang $\omega = \mathrm{Rg}(\mathbf{e})$ dans le code linéaire $[n, k]$ généré par la matrice :

$$G_{\mathrm{pub}} = \begin{pmatrix} \mathbf{g} \\ \vdots \\ \mathbf{g}^{[k-u-1]} \\ \mathbf{K}_1 \\ \vdots \\ \mathbf{K}_u \end{pmatrix} \tag{4.7}$$

Un tel décodage peut être effectué par les algorithmes d'Ourivski-Johannson, sans exploiter la structure particulière de cette matrice. La complexité de ces décodages (cf. théorème 16) est alors :

- Par énumération des bases, en $O((k+\omega)^3 q^{(\omega-1)(m-\omega)+2})$.
- Par énumération des coordonnées, en $O((k+\omega)^3 \omega^3 q^{(\omega-1)(k+1)})$.

4.5.10 Attaque par déchiffrement et linéarisation

Cette attaque ressemble à ce que l'on a pu voir dans le début de la section contre la version affaiblie du système, et s'inspire d'une des idées de Coron contre le système Augot-Finiasz réparé (cf. [10]). Si les paramètres du système utilisant la trace sont mal choisis, cette attaque reste valide.

En reprenant les notations de la sous-section précédente, on veut résoudre :

$$\exists \mathbf{e}, m, \alpha_1, \ldots, \alpha_u, \begin{cases} \mathbf{y} = m(\mathbf{g}) + \sum_{t=1}^{u} \alpha_t \mathbf{K}_t + \mathbf{e}, \\ \deg_q(m) \leq k - u - 1, rg(\mathbf{e}) \leq \omega \end{cases} \tag{4.8}$$

Ce système est équivalent au système suivant :

$$\exists m, \alpha_1, \ldots, \alpha_u, V, \begin{cases} V(\mathbf{y}) = V \circ m(\mathbf{g}) + \sum_{t=1}^{u} \alpha_t V(\mathbf{K}_t), \\ \deg_q(m) \leq k - u - 1, \deg_q(V) \leq \omega \end{cases} \tag{4.9}$$

On peut ensuite tout linéariser, pour obtenir le système suivant, plus large :

$$\exists V, R_1, \ldots, R_u, N, \begin{cases} V(\mathbf{y}) = N(\mathbf{g}) + \sum_{t=1}^{u} R_t(\mathbf{K}_t), \\ \deg_q(V) \leq \omega, \deg_q(R_t) \leq \omega, \\ \deg_q(N) \leq k + \omega - u - 1 \end{cases} \tag{4.10}$$

Ce sytème se traduit en un système linéaire dans $\mathbb{F}_{q^m}$ à n équations et $\omega + 1 + u(\omega + 1) + k + \omega - u = k + (u + 2)\omega + 1$ inconnues. L'attaque peut aboutir en temps polynomial si l'espace solution peut être de dimension 1, i.e. si :

$$k + (u + 2)\omega \leq n. \tag{4.11}$$

Remarque 9. *Cette condition est toujours vérifiée quand u vaut 1 ou 2. Le cryptosystème doit donc utiliser un paramètre $u \geq 3$.*

Si la condition 4.11 n'est pas vérifiée, la meilleure méthode connue pour résoudre le système 4.8 reste d'essayer toutes les solutions non colinéaires de 4.10. Il faudra alors essayer en moyenne $q^{m(n-k-(u+2)\omega)}$ solutions.

4.5.11 Attaque par déchiffement et bases de Gröbner

Pour cette attaque, on reprend le système non linéarisé précédent :

Nombre de variables	Rang de l'erreur	Degré	Magma 2.11-2/F4
$u = 2$	$\omega = 2$	7	0.01s
	$\omega = 3$	15	0.340s
	$\omega = 4$	31	9s
	$\omega = 5$	63	500s
	$\omega = 6$	127	11 hours
$u = 3$	$\omega = 2$	7	0.06s
	$\omega = 3$	15	54s
	$\omega = 4$	31	15 hours

TAB. 4.1 – Simulations d'attaques par bases de Gröbner utilisant les paramètres $n = 36,\ q = 2^{36},\ k = 10,\ W = 14$

$$\exists m, \alpha_1, \ldots, \alpha_u, V, \begin{cases} V(\mathbf{y}) = V \circ m(\mathbf{g}) + \sum_{t=1}^{u} \alpha_t V(\mathbf{K}_t), \\ \deg_q(m) \leq k - u - 1, \deg_q(V) \leq \omega \end{cases}$$

Et on ne linéarise que $V \circ m$:

$$\exists N, V, \alpha_1, \ldots, \alpha_u, \begin{cases} V(\mathbf{y}) = N(\mathbf{g}) + \sum_{t=1}^{u} \alpha_t V(\mathbf{K}_t), \\ \deg_q(N) \leq k + \omega - u - 1, \deg_q(V) \leq \omega \end{cases}$$

Ce système, dans $\mathbb{F}_{q^m}$ est maintenant de degré 2, à n équations et $\omega + 1 + k + \omega - u + u = k + 2\omega + 1$ inconnues. À ce jour, les méthodes les plus rapides pour résoudre ce genre de système consistent à calculer les bases de Gröbner de l'idéal engendré par ces équations. Les librairies récentes de MAGMA incluent des implémentations efficaces des algorithmes F4 et F5. On a $k+2\omega+1 < n$, donc on est sur que ce genre d'algorithme termine, mais son temps de calcul peut être exponentiel. Le tableau 4.1 semble montrer que le temps de calcul croît beaucoup trop vite pour que l'attaque soit utilisable avec les paramètres cryptographiques que nous donnons plus loin.

4.5.12 Attaque algébrique

En métrique de Hamming, Coron donne une deuxième attaque contre le cryptosystème Augot-Finiasz réparé, qui couvre les choix de paramètres

tels que l'attaque par déchiffrement est évitée. Cette attaque algébrique se fait directement sur la clé publique, pour récupérer la clé secrète, par une variante de l'algorithme de Berlekamp-Welsh. En fait, au lieu d'essayer de décoder directement $\mathbf{K} = P(\mathbf{g}) + \mathbf{E}$ dans $\mathbb{F}_{q^{mu}}$, on décode en parallèle les u projections de cette équation dans $\mathbb{F}_{q^m}$. En métrique de Hamming, les erreurs sont projetées dans les mêmes positions, ce qui permet d'affaiblir la condition $n \geq k + 2W$ en $n \geq k + \frac{u+1}{u}W$.

En métrique rang, ce type de technique ne s'applique pas. $\mathbf{E}$ est un n-uplet d'erreur. Une information intéressante est le sous-espace vectoriel (de dimension W) contenant tous les éléments de ce n-uplet. Mais les projections de cet espace sur $\mathbb{F}_{q^m}$ ne semblent pas corrélées.

4.5.13 Attaque Overbeck

Cette méthode d'attaque, présentée dans [38], cryptanalyse effectivement la clé publique pour les paramètres que nous avions proposés dans [18]. Cependant, comme je le montre ici, il est possible de se prémunir contre cette attaque, au prix d'une augmentation très raisonnable de la taille de la clé publique.

En utilisant les notations précédentes, on a :

$$\mathbf{K}_t = Tr(\gamma_t \mathbf{K}) = P_t(\mathbf{g}) + \mathbf{E}_t \text{ pour } t = 1 \dots u.$$

Les $\mathbf{K}_t$ sont connus, mais ni les P_t, ni les $\mathbf{E}_t$. On écrit :

$$P_t = \sum_{j=0}^{k-1} p_{t,j} X^{[j]}.$$

On pose, en reprenant les notations de la section 4.4.2 :

$$\mathcal{K} = \Lambda_l \begin{pmatrix} \mathbf{K}_1 \\ \vdots \\ \mathbf{K}_u \end{pmatrix}.$$

La matrice $\mathcal{K}$ est une matrice à $u(l+1)$ lignes et n colonnes, à coefficients dans $\mathbb{F}_{q^m}$. De plus, $\mathcal{K}$ peut s'écrire sous la forme :

$$\mathcal{K} = \mathcal{P}\mathcal{G} + \mathcal{E}T, \text{ avec}$$

$$\mathcal{P} = \begin{pmatrix}
p_{1,0} & & \cdots & p_{1,k-1} & & 0 \\
\vdots & & & \vdots & & \vdots \\
p_{t,0} & & \cdots & p_{t,k-1} & & 0 \\
0 & p_{1,0}^{[1]} & \cdots & & p_{1,k-1}^{[1]} & 0 \\
\vdots & \vdots & & & \vdots & \vdots \\
0 & p_{t,0}^{[1]} & \cdots & & p_{t,k-1}^{[1]} & 0 \\
 & & \ddots & & & \\
 & & & \ddots & & \\
 & & & & \ddots & \\
0 & 0 & p_{1,0}^{[l]} & \cdots & & p_{1,k-1}^{[l]} \\
\vdots & \vdots & \vdots & & & \vdots \\
0 & 0 & p_{t,0}^{[l]} & \cdots & & p_{t,k-1}^{[l]}
\end{pmatrix},$$

$$\mathcal{G} = \begin{pmatrix}
g_1 & \cdots & g_n \\
\vdots & & \vdots \\
g_1^{[k+l-1]} & \cdots & g_n^{[k+l-1]}
\end{pmatrix},$$

$$\mathcal{E} = \begin{pmatrix}
0 & \cdots & 0 & e'_{1,1} & \cdots & e'_{1,W} \\
\vdots & & \vdots & \vdots & & \vdots \\
0 & \cdots & 0 & e'_{u(l+1),1} & \cdots & e'_{u(l+1),W}
\end{pmatrix},$$

et T une matrice carrée inversible de taille n, à coefficients dans $\mathbb{F}_q$.

Si la matrice $\mathcal{G}$ est de rang $k+l$, alors en considérant $k+l$ lignes linéairement indépendantes, on se ramène au cas d'une matrice génératrice d'un système GPT de paramètres $n' = n-W$, $k' = k+l$, $t'_1 = W$, $s' = \mathrm{Rg}(Z) = W$. Sous les conditions décrites dans la section 4.4.2, l'attaquant peut mettre au point un algorithme de décodage en temps polynomial pour $\mathcal{G}$, dont il déduit un algorithme pour G_{pub}.

Le paramètre l est librement choisi par l'attaquant, mais vérifie toujours $u(l+1) \geq k+l$ (à cause de la condition sur le rang de $\mathcal{G}$). De plus, pour que le système GPT auquel on se ramène ait un sens, il faut $k' < n'$, ce qui impose $k + l < n - W$.

Proposition 37. *Si les paramètres du système vérifient*

$$n - k - W \leq \frac{k-u}{u-1},$$

l'attaque Overbeck ne peut pas s'appliquer.

Preuve . *D'après la condition sur le rang de $\mathcal{G}$, le paramètre l choisi par l'attaquant vérifie $l \geq \frac{k-u}{u-1}$, donc $l \geq n - k - W$. Or, cette dernière inégalité est contradictoire avec l'inégalité $k + l < n - W$.* $\diamondsuit$

4.5.14 Discussion sur le choix des paramètres

Il faut choisir les paramètres pour se prémunir contre toutes les attaques présentées ci-dessus. Deux attaques en particulier sont menaçantes : celle sur le chiffré par linéarisation, et celle d'Overbeck sur la clé publique.

Théorème 20 (Résistance du cryptosystème aux attaques connues)**.** *Si les paramètres q, m, u, n, k, W, ω du système vérifient :*

- $n - k > (u + 2)\omega$,
- $n - k - W \leq \frac{k-u}{u-1}$,
- $n - k - W \geq 2\omega$,
- $W \geq \frac{n-k}{2}$,

Alors le cryptosystème fondé sur le problème de reconstruction des polynômes linéaires est bien défini, et n'est cryptanalysé en temps polynomial par aucune attaque connue.

Preuve . *Les deux premières inégalités neutralisent l'attaque par linéarisation sur le chiffré et l'attaque Overbeck sur la clé publique. La troisième garantit la difficulté du décodage de $\mathbf{K}$ comme un mot de code de Gabidulin. La quatrième assure le fonctionnement du déchiffrement.* $\diamondsuit$

Corollaire 6. *Pour un paramètre de complexité de 2^{80}, on peut utiliser, par exemple, les jeux de paramètres suivants pour le système :*

- *$q = 2$, $m = n = 56$, $u = 3$, $k = 28$, $W = 16$, $\omega = 6$, pour une clé publique de 12544 bits (9408 bits après inclusion du support dans les spécifications), et un taux de transmission efficace de $25/56 \approx 45\%$.*
- *$q = 2$, $m = n = 54$, $u = 4$, $k = 32$, $W = 13$, $\omega = 4$, pour une clé publique de 14580 bits (11664 bits après inclusion du support dans les spécifications), et un taux de transmission efficace de $28/54 \approx 52\%$.*

4.6 Conclusion

Nous avons étudié dans ce chapitre la possibilité d'utiliser la métrique rang pour définir des cryptosystèmes basés sur les codes. Nous avons montré que, contrairement aux idées reçues, les techniques de cryptanalyse en métrique rang ne permettent pas de casser tous les systèmes. Nous avons montré que certains cryptosystèmes résistent bien aux attaques connues, tout en conservant une clé publique de taille particulièrement faible par rapport à ce que l'on sait faire en métrique de Hamming.

En particulier, nous avons proposé un cryptosystème fondé sur le problème de reconstruction des polynômes linéaires. Ce cryptosystème semble résister à toutes les attaques connues, alors que sa clé publique est plus petite que celle de tous les autres systèmes existants basés sur les codes. De plus, nous avons présenté et programmé des algorithmes de chiffrement et de déchiffrement rapides pour ce système. En effet, les paramètres limitant en pratique dans les complexités de chiffrement et de déchiffrement sont respectivement en $O(kn)$ et $O(k\omega^2)$.

Évidemmment, nous ne sommes pas capables d'affirmer avec aplomb que ce système est absolument sûr d'un point de vue cryptographique. Néanmoins, il semble l'être, et les performances atteintes justifient que l'on se penche de plus près sur l'étude de la métrique rang.

Chapitre 5

Conclusions et perspectives

J'ai étudié durant ma thèse deux axes de recherche concernant la cryptographie basée sur les codes correcteurs. L'un concerne l'utilisation de codes géométriques dans des systèmes de type McEliece en métrique de Hamming. L'autre axe est l'utilisation de la métrique rang et des codes de Gabidulin en particulier. Je présente ici mes conclusions sur ces domaines d'étude, ainsi que des perspectives pour des recherches ultérieures.

5.1 Utilisation des codes géométriques

Nous avons montré, avec Lorenz Minder, que les codes géométriques sont facilement reconnaissables, et qu'on ne peut pas les utiliser tels quels dans un système de type McEliece. Nos algorithmes d'attaque peuvent certainement être neutralisés, il existe plusieurs pistes dans cette direction. On peut imaginer par exemple un poinçonnage intelligent du code utilisé, le choix de courbes particulières sur lesquelles nos suppositions sont forcément fausses, l'utilisation de courbes de genre très élevé, ou bien encore l'utilisation de nouvelles transformations pour masquer la structure du code.

Cependant, ces perspectives me semblent vouées à l'échec, car je pense que les codes géométriques sont irrémédiablement structurellement faibles. La propriété d'optimalité (ou de quasi-optimalité) qui fait l'attractivité de ces codes est précisément ce qui les rend vulnérables à des attaques structurelles. En effet, c'est parce que ces codes sont optimaux que l'on peut facilement générer des mots de code de poids minimal. Or, la connaissance des mots de poids faible d'un code renseigne énormément sur la structure de ce code. Les algorithmes que nous avons mis au point ne font qu'exploiter sur un cas

particulier ce principe général.

Nos algorithmes de cryptanalyse peuvent certainement être améliorés. Leur complexité peut peut-être être réduite, et il y a sans doute moyen d'affaiblir les hypothèses que nous devons faire pour que ces algorithmes fonctionnent. De plus, il serait extrêmement intéressant de programmer une implémentation efficace de ces algorithmes. Néanmoins, je pense que le résultat important dans le domaine (à savoir, la faiblesse structurelle des codes géométriques) est démontré par nos travaux, et qu'il ne reste plus qu'à affiner ce résultat.

5.2 Utilisation de la métrique rang

La métrique rang a toujours été considérée avec méfiance ou scepticisme par la communauté cryptographique. Comme, de plus, son intérêt en théorie de l'information est très limité, peu de travaux ont été écrits concernant la métrique rang. Il s'agit cependant selon moi d'un axe de recherche extrêmement intéressant, dans lequel de nombreux résultats importants attendent d'être découverts. De tels résultats, en plus de leurs applications directes dans la conception de système cryptographiques, nous permettraient sans doute au passage de mieux comprendre la théorie des codes en métrique de Hamming.

Tout d'abord, on connaît très peu de familles de codes intéressantes en métrique rang. Les seuls codes connus facilement décodables sont dérivés des codes de Gabidulin, et presque tous les sujets d'étude tournent autour de ces codes. Pour explorer plus avant la métrique rang, il serait nécessaire de se diversifier, et de mettre au jour d'autres familles de codes, complètement différentes.

Concernant les codes de Gabidulin, je me suis intéressé au problème du décodage en liste. Le résultat que je trouve laisse le sujet complètement ouvert, car il existe toujours un intervalle dans lequel on ne sait pas si on peut décoder en liste. Il faudrait se pencher plus avant sur la question et, ou bien exhiber un algorithme de décodage en liste, ou bien comprendre quelle différence fondamentale entre les codes de Reed-Solomon et de Gabidulin empêche la généralisation de l'algorithme de Sudan.

Au niveau des applications cryptographiques, les performances du sys-

tème GPT et du système fondé sur le problème de reconstruction des polynômes linéaires rendent ces deux cryptosystèmes extrêmement intéressants, très supérieurs aux implémentations classiques de McEliece. Les codes de Goppa présentent l'avantage d'avoir été étudiés en détail par de nombreux chercheurs depuis longtemps. J'aurais donc tendance à les considérer comme plus sûrs que les codes de Gabidulin pour la cryptographie. Il s'agit cependant là d'une question de foi, et la métrique rang améliore suffisamment les performances des cryptosystèmes pour ne pas devoir être rejetée sur des croyances. Il convient donc d'étudier ces systèmes plus en détail afin de rassembler de vrais arguments, dans un sens ou dans l'autre, sur leur sécurité.

La mise au point (ou au moins la tentative de mise au point) d'attaques directes contre GPT ou contre le système fondé sur le problème de reconstruction des polynômes linéaires est une voie de recherche intéressante dans ce sens. Un autre axe de recherche important est la formalisation de problèmes supposés difficiles en métrique rang, et la recherche de réductions de sécurité vers ces problèmes. Ce serait un premier pas vers la sécurité prouvée. Un problème qui semble revenir souvent dans les questions de sécurité est la recherche d'une solution particulière dans l'espace vectoriel des solutions d'un système linéaire. On sait très peu de choses de façon générale sur ce genre de problèmes. Des résultats dans ce domaine (par exemple, sur les bases de Gröbner) peuvent présenter un grand intérêt en cryptographie et dans d'autres domaines.

Pour résumer, je pense que la cryptographie en métrique rang est un domaine intéressant et ouvert, dans lequel il reste de nombreuses choses à faire.

Bibliographie

[1] D. Augot and M. Finiasz. A Public Key Encryption Scheme Based on the Polynomial Reconstruction Problem. In *EUROCRYPT*, number 2656 in LNCS, pages 222–233. Springer-Verlag, 2003.

[2] T. P. Berger and P. Loidreau. How to Mask the Structure of Codes for a Cryptographic use. *Designs, Codes and Cryptography*, 35 :63–79, 2005.

[3] T. G. Berry. Construction of Linear Systems on Hyperelliptic Curves. *Journal of Symbolic Computation*, 26 :315–327, 1998.

[4] B. Biswas and N. Sendrier. McEliece Cryptosystem Implementation : Theory and Practice. In *Post-Quantum Cryptography*, number 5299 in LNCS, pages 47–62. Springer-Verlag, 2008.

[5] A. Canteaut and F. Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code : Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1) :367–378, January 1998.

[6] A. Canteaut and N. Sendrier. Cryptanalysis of the original McEliece Cryptosystem. In *Advances in Cryptology - ASIACRYPT'98*, number 1514 in LNCS, pages 187–199. Springer-Verlag, 1998.

[7] D. G. Cantor. Computing in the Jacobian of a Hyperelliptic Curve. *Mathematics of Computation*, 48(177) :95–101, 1987.

[8] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer, 1993.

[9] Henri Cohen and Gerhard Frey, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC Press, 2005.

[10] J. S. Coron. Cryptanalysis of a Public-Key Encryption Scheme Based on the Polynomial Reconstruction Problem. In J. Zhou F. Bao, R. Deng, editor, *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography, PKC2004*, LNCS, pages 14–28, 2004.

[11] N. Courtois, M. Finiasz, and N. Sendrier. How to achieve a McEliece-based Digital Signature Scheme. Technical report, INRIA, http ://eprint.iacr.org/2001/010/, 2001.

[12] C. Faure. Étude d'un Cryptosystème à Clé Publique Basé sur le Problème de Reconstruction de Polynômes Linéaires. Master's thesis, Université Paris VI, 2004.

[13] C. Faure. Cryptosystème à Clé Publique Fondé sur le Problème de Reconstruction de Polynômes Linéaires. Journées "Codage et Cryptographie", 2005. Aussois.

[14] C. Faure. Average Number of Gabidulin Codewords Within a Sphere. In *Proceedings of the 10th international workshop on Algebraic and Combinatorial Coding Theory, ACCT 2006*, pages 86–89, 2006.

[15] C. Faure. Nombre Moyen de Mots de code de Gabidulin à l'Intérieur d'une Boule. Journées "Codage et Cryptographie", 2006. Eymoutiers.

[16] C. Faure. Cryptanalysis of the McEliece Cryptosystem over Hyperelliptic Codes. In L. Bassalygo and S. Dodunekov, editors, *Proceedings of the 11th international workshop on Algebraic and Combinatorial Coding Theory, ACCT 2008*, pages 99–107, 2008.

[17] C. Faure and P. Loidreau. A new Public-Key Cryptosystem Based on the Problem of Reconstructing p–Polynomials. In P. Charpin and O. Ytrehus, editors, *Proceedings of the 4th International workshop on Coding and Cryptography, WCC 2005*, pages 275–284, 2005.

[18] C. Faure and P. Loidreau. A new Public-Key Cryptosystem Based on the Problem of Reconstructing p–Polynomials. 3969 :304–315, 2006.

[19] G.-L. Feng and T. R. N. Rao. Decoding Algebraic Geometric Codes up to the designed Minimum Distance. *IEEE-IT*, 39 :37–45, 1993.

[20] M. Finiasz. *Nouvelles Constructions utilisant des Codes Correcteurs d'Erreurs en Cryptographie à Clef Publique*. PhD thesis, École Polytechnique, Oct. 2004.

[21] William Fulton. *Algebraic Curves : An Introduction to Algebraic Geometry*. Addison Wesley Publishing Company, 1974. Disponible en ligne : http ://www.math.lsa.umich.edu/ wfulton/CurveBook.pdf.

[22] E. M. Gabidulin. Theory of Codes with Maximal Rank Distance. *Problems of Information Transmission*, 21 :1–12, July 1985.

[23] E. M. Gabidulin and A. V. Ourivski. Modified GPT PKC with Right Scrambler. In Daniel Augot and Claude Carlet, editors, *Proceedings of the 2nd International workshop on Coding and Cryptography, WCC 2001*, pages 233–242, 2001.

[24] E .M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a Non-Commutative Ring and their Application in Cryptology. *LNCS*, 573 :482 – 489, 1991.

[25] J. K. Gibson. Severely Denting the Gabidulin Version of the McEliece Public-Key Cryptosystem. *Designs, Codes and Cryptography*, 6 :37–45, 1995.

[26] V. Guruswami and M. Sudan. Improved Decoding of Reed-Solomon Codes and Algebraic Geometry Codes. *IEEE-IT*, 45(6) :1757–1767, 1999.

[27] H. Janwa and O. Moreno. Public Key Cryptosystems Using Algebraic-Geometric Codes. *Designs, Codes and Cryptography*, 8(3), 1996.

[28] Y. X. Li, R. H. Deng, and X. M. Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1) :271–273, 1994.

[29] P. Loidreau. Métrique Rang et Cryptographie. Mémoire d'habilitation à diriger les recherches, 2007. Université Paris-VI.

[30] J. L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15(1) :122–127, January 1969.

[31] R. J. McEliece. A public-key Cryptosystem based on Algebraic Coding Theory. Technical report, Jet Propulsion Lab. DSN Progress Report, 1978.

[32] L. Minder. *Cryptography based on Error Correcting Codes*. PhD thesis, EPFL, Lausanne, 2007.

[33] H. Niederreiter. Knapsack-type Cryptosystems and Algebraic Coding Theory. *Problems of Control and Information Theory*, 15(2) :159–166, 1986.

[34] Ö. Ore. On a special class of Polynomials. *Transactions of the American Mathematical Society*, 35 :559–584, 1933.

[35] Ö. Ore. Contribution to the theory of finite fields. *Transactions of the American Mathematical Society*, 36 :243–274, 1934.

[36] A. Ourivski and T. Johannson. New Technique for Decoding Codes in the Rank Metric and Its Cryptography Applications. *Problems of Information Transmission*, 38(3) :237–246, September 2002.

[37] A. V. Ourivski, E. M. Gabidulin, B. Honary, and B. Ammar. Reducible rank codes and their applications to cryptography. *IEEE Transactions on Information Theory*, 49(12) :3289–3293, 2003.

[38] R. Overbeck. Structural Attacks for Public Key Cryptosystems based on Gabidulin Codes. *Journal of Cryptology*, 21 :280–301, 2008.

[39] S. Sakata, H. E. Jensen, and T. Høholdt. Generalized Berlekamp-Massey decoding of Algebraic Geometric Code up to half the Feng-Rao bound. *IEEE-IT*, 41(6) :1762–1768, Nov. 1995.

[40] N. Sendrier. On the Structure of a Randomly Permuted Concatenated Code. In *Proceedings of EUROCODE 94*, LNCS, pages 169–173, 1994.

[41] C. E. Shannon. Communication Theory of Secrecy Systems. *Bell system technical journal*, 28 :656–715, 1949.

[42] M. A. Shokrollahi and H. Wasserman. Decoding Algebraic-Geometric Codes beyond the Error-Correction Bound. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 241–248, 1998.

[43] V. M. Sidel'nikov and S. O. Shestakov. On cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics*, 4(3) :57–63, 1992. en russe.

[44] D. Silva, F.R. Kschischang, and R. Koetter. A Rank-Metric Approach to Error Control in Random Network Coding. *IEEE-IT*, 54(9) :3951–3967, Sep. 2008.

[45] J. H. Silverman. *The Arithmetic of Elliptic Curves*. Springer, 1986.

[46] J. Stern. A method for finding Codewords of Small Weight. In G. Cohen and J. Wolfmann, editors, *Coding theory and applications*, number 388 in LNCS, pages 106–113. Springer-Verlag, 1989.

[47] M. Sudan. Decoding of Reed-Solomon Codes beyond the Error-Correcting Bound. *Journal of Complexity*, 13 :180–193, 1997.

[48] N. Silberstein T. Etzion. Error-Correcting Codes in Projective Spaces via Rank-Metric Codes and Ferrers Diagrams. preprint, available at http ://arxiv.org/abs/0807.4846.

[49] M. Tsfasman, S. Vladut, and D. Nogin. *Algebraic Geometric Codes : Basic Notions*. American Mathematical Society, 2007.