Décodage en liste et application à la sécurité de l'information

Morgan Barbier

École Polytechnique Laboratoire d'informatique

THÈSE DE DOCTORAT

présentée à
L'ÉCOLE POLYTECHNIQUE

pour obtenir le titre de
DOCTEUR ÈS SCIENCES

Spécialité : **Informatique**

soutenue le 2 décembre 2011
par

Morgan BARBIER

Décodage en liste et application à la sécurité de l'information

Jury

Rapporteurs

Thierry BERGER	Université de Limoges
Grigory KABATIANSKY	Institute for Information Transmission Problems
Pascal VÉRON	Université du Sud Toulon-Var

Directeurs de Thèse

Daniel AUGOT	Directeur de recherche INRIA
Caroline FONTAINE	Chargée de recherche CNRS

Examinateurs

Carlos MUNUERA	University of Valladolid
Catuscia PALAMIDESSI	Directrice de recherche INRIA

Remerciements

Bien qu'une thèse est un projet personnel, je n'aurais jamais pu la réaliser sans l'aide de nombreuses personnes.

Tout d'abord, merci à tous les membres de mon jury de m'avoir consacré du temps qui leur est très précieux, ainsi que leur présence durant ma soutenance malgré les longues distances. Je pense évidemment à Carlos Munuera, qui a fait le voyage dans la journée en sacrifiant deux nuits dans les trains. Tout comme Grigory Kabatiansky venant d'Abu Dhabi. Je ne peux oublier également Thierry Berger et Pascal Véron qui se sont libérés de leurs obligations pour faire l'aller-retour et surtout, avec Grigory Kabatiansky, ils ont lu minutieusement mon mémoire, ce qui a amélioré considérablement la qualité.

Évidemment, un grand MERCI à Daniel Augot et Caroline Fontaine, qui m'ont témoignés de leur confiance en m'acceptant en thèse, et m'ont prodigués de nombreux conseils durant ces trois années. Je ne peux cacher que j'ai beaucoup appris à leurs côtés, et j'espère continuer même après ma thèse. J'ai été aussi extrêmement sensible à leurs qualités humaines et a leurs encouragements.

J'ai eu la chance de co-écrire des articles avec des scientifiques brillants, grâce à qui j'ai pu apprendre la rigueur et la rédaction scientifique. Encore une fois, merci à Daniel Augot et Caroline Fontaine, qui m'ont transmis leur savoir faire. Et merci à Paulo Barreto, Pierre-Louis Cayrel, Christophe Chabot, Carlos Munuera et Guillaume Quintin pour les projets et les échanges très enrichissants.

J'aimerais également prendre le temps de remercier tout le laboratoire LIX, qui ont su se montrer accueillant et me mettre dans les meilleures dispositions pour travailler. Ainsi que les conversations délirantes que j'ai pu avoir avec James Régis, notre ingénieur système préféré (et unique !), avec qui nous avons bien évidemment jamais lancé de troll !

L'intégration a été très facile dans l'équipe Crypto du LIX et TANC de L'INRIA. Tous les membres m'ont mis à l'aise très rapidement. Je suis très content et fier d'avoir fait parti de cette équipe. J'ai beaucoup apprécié les conversations techniques ou humoristiques durant les pauses avec les membres de l'équipe dont Daniel Augot, Alain

Couvreur, Cécile Gonçalves, Jérôme Milan, François Morain, Guillaume Quintin et Ben Smith. J'ai eu une grosse interaction (humoristique) avec les doctorants de l'équipe "du dessus", comme Jérémy Berthomieu et Romain Lebreton. Avec Romain et Van Du, nous avons aussi partagé la "passion" du ballon rond – dont notre niveau ressemblait plus à pièce de théâtre (tragique ou comique ?).
L'équipe TANC est très proche de l'équipe SECRET, ce qui m'a permi d'avoir des conversations très enrichissante et drôle – l'un n'empêche pas l'autre, "enfin je m'comprends, je m'comprends !" – avec Ayoub Otmani, Nicolas Sendrier et Jean-Pierre Tillich.

Dans notre équipe, il y a évidemment des gens très brillants, mais aussi d'une gentillesse et d'un humour renversant ! Je pense bien sûr à Coincoin, qui a su éclairer nos après-midis et nos soirées avec son humour quintesque ! Il nous a converti "notre Jéjé" et "Mister Anderson" ! Plus sérieusement, il a toujours du temps pour nous enrichir de ses larges connaissances, partager son bureau avec lui m'a permi d'apprendre beaucoup de choses aussi bien en maths qu'en info. Pendant ces trois ans, une personne était là, à mes côtés, dévoué, toujours prêt à aider, à rendre service, Jéjé. En plus de ses talents scientifiques, il est également un bon écrivain, ce qui lui a valu de relire la quasi-totalité de mes documents ! En plus de ses bons conseils littéraires, il a su égayer nos journées par son humour et ses remarques épicées ! Et Mister Anderson est toujours prêt à partager ses expériences, jamais avare de bons conseils et son humour décalquant ! Je ne peux pas oublier mes copines de clopes, Alex et Cécile, on a partagé des bons moments de détentes, propices aux confidences. C'est pendant ces moments qu'on a pu se plaindre et partager nos divers conseils en tout genre.

Une ÉNORME pensée à ma famille qui a toujours cru en moi et soutenu, malgré mon isolement géographique et trop souvent téléphonique ! J'ai envie de leur dire "Oups, désolé pour tous les anniversaires oubliés !" mais surtout MERCI. À ma maman, mon papa, ma soeur, mes grands-parents, ma tante, mon cousin et mes 4 neveux et nièces : tous plus adorables les uns que les autres ! Ils se reconnaîtront, enfin j'espère ! Et non, Johann je ne t'ai pas oublié ! Je te réserve une attention toute particulière. Outre d'être un grand frère exemplaire, tu as été un modèle sur lequel je me suis appuyé constamment. Il est évident que je n'aurais jamais défendu une thèse sans toi. Tu as réussi à me contaminer, à me transmettre toute ta passion pour la recherche et la sécurité de l'information.

Et toi mon Totof, on n'est pas liés par des liens de sang, il est vrai ! Mais ça ne t'a pas empêché d'être toujours là dans les moments importants et de me consacrer un temps fou. On ne peut pas compter les heures que tu as passé pendant mon Master 2, à m'expliquer les cours de maths, mais surtout de codes ! Tu m'as transmis ton virus pour les codes et pour la recherche. Tu m'as beaucoup aidé dans ma vie professionnelle mais également dans ma vie personnelle, tu es un merveilleux ami !

Et toi mi Perikita, depuis qu'on se connaît, tu as toujours su trouver les mots justes quand j'en avais besoin de les entendre. Parce que tu me connais très bien et grâce à ta

patience, les difficultés sont plus faciles. Même quand tu en n'avais pas assez pour toi, tu me consacrais du temps pour m'aider dans tout ce que je voulais entreprendre. Je sais que mes absences répétées pour mes missions étaient difficiles à gérer pour toi. Tu m'as toujours encouragé et compris dans mes choix professionnels, même si cela nous rend la vie plus difficile...Merci.

À toutes les personnes que j'ai oubliées qui se reconnaîtront !

Contributions

Durant ma thèse, j'ai eu la chance de pouvoir contribuer à différents domaines. Je propose ici un résumé de mes contributions par thèmes : codes correcteurs d'erreurs, cryptographie et stéganographie basées sur les codes. Les articles sont classés par thèmes, puis par ordre chronologique décroissant.

Codes correcteurs d'erreurs

- Avec Christophe Chabot et Guillaume Quintin, on a montré l'équivalence entre les codes quasi-cycliques et les idéaux principaux de l'anneau $M_\ell(\mathbb{F}_q)[X]/(X^m - 1)$. On a montré également que tous les idéaux d'un tel anneau sont principaux. On a défini deux nouvelles classes de codes, les codes *quasi-BCH* et *quasi-évaluation*, pour lesquelles on a exhibé leurs paramètres ou des bornes inférieures. En s'appuyant sur la notion de code *folder*, on peut décoder certains codes *quasi-BCH* avec les algorithmes de décodage des codes BCH. Pour les autres, on a proposé un algorithme de décodage unique. Pour finir, à partir des codes quasi-évaluation on a construit 49 nouveaux codes ayant une plus grande distance minimale que les codes connus jusqu'à présent [7].

- En collaboration avec Daniel Augot et Alain Couvreur, on a proposé un algorithme de décodage en liste pour la famille des codes alternants [1]. Cette méthode à été redécouverte indépendamment de Tal et Roth ; elle permet de décoder un code alternant sur $\mathbb{F}_q$ jusqu'à la borne de Johnson q-aire. C'est, à notre connaissance, la méthode qui permet de décoder le plus d'erreurs à l'heure actuelle pour ce type de codes. On a fait la preuve de cet algorithme et on a étudié sa complexité temps. Une version courte du rapport de recherche a été acceptée à *IEEE Information Theory Workshop* (ITW) [2]. Ce travail est présenté dans la section 2.4.

- Grâce aux avancées effectuées sur le décodage en liste, on a proposé de comparer le rayon de recouvrement d'un code et la capacité de correction d'un algorithme de décodage de ce code [4]. Cette notion naturelle, a permis de définir une nouvelle classe de codes pour lesquels le problème du décodage complet est soluble en temps polynomial, alors que ce problème a été montré NP-difficile. On a proposé un lemme général pour trouver des codes binaires

dans cette nouvelle classe de codes. Ainsi, on est parvenu à trouver quelques codes pour lesquels le problème du décodage complet est soluble en temps polynomial. Cet article en version courte a été accepté à *Yet Another Conference on Cryptography* (YACC) et une version longue se trouve dans cette thèse, section 2.5.

Cryptographie

- Grâce à notre méthode de décodage en liste pour les codes alternants, on a proposé, avec Paulo Barreto, de réduire la taille de clé du cryptosystème de McEliece [5]. Le système de chiffrement de McEliece est réputé pour être l'un des cryptosystèmes à clé publique les plus rapides pendant les phases de chiffrement et de déchiffrement, mais la taille des clés publiques est bien plus volumineuse que les autres cryptosystèmes asymétriques. Pour certains niveaux de sécurité donnés – complexité des attaques, on exhibe un gain jusqu'à 21%. La réduction de clé obtenue est expliqué dans la section 3.3. Cet article à été accepté à *the IEEE Symposium on Information Theory* (ISIT).

- En coopération avec Pierre-Louis Cayrel, on a rappelé les principales attaques algébriques pour les systèmes de chiffrement à flots et la dernière attaque algébrique de cette époque, dûe à H. Raddum et I. Semaev, pour les cryptosystèmes en blocs. Cet article accepté à MajecSTIC a été réalisé pendant mon stage de Master [6], c'est pourquoi ce travail n'est pas expliqué dans ce manuscrit.

Stéganographie

- Avec Daniel Augot et Caroline Fontaine, on a proposé de modifier très légèrement le problème d'insertion classique, pour en assurer la réussite. On s'est intéressé tout particulièrement aux codes parfaits linéaires. De plus, on a adapté le schéma ZZW pour gérer dynamiquement les paramètres de notre schéma d'insertion. Dans cette thèse, section 4.5, on y présente également une borne minimale sur un paramètre important de notre schéma d'insertion pour tous les codes linéaires en toute généralité. Une présentation de ce travail se fera à la conférence *Institute of Mathematics and its Applications on Cryptography and Coding 2011* [3].

- En association avec Carlos Munuera, on a proposé d'exhiber des conditions nécessaires et suffisantes pour assurer l'insertion d'un message. On s'est intéressé aux codes linéaires en toute généralité en étudiant la distance duale. Puisque le raisonnement est de nature combinatoire, on a pu alors le généraliser aux codes systématiques, c'est une large famille de codes non forcément linéaires mais contenant tous les codes linéaires. Ce travail est détaillé dans ce manuscrit, section 4.4, et va apparaître dans le journal *Adances in Mathematics of Communications* [8].

Publications

[1] Daniel Augot, Morgan Barbier et Alain Couvreur. List-decoding of binary Goppa codes up to the binary Johnson bound. Rapport technique RR-7490, INRIA, décembre 2010.

[2] Daniel Augot, Morgan Barbier et Alain Couvreur. List-Decoding of binary Goppa codes up to the binary Johnson bound. *Dans 2011 IEEE Information Theory Workshop (IEEE ITW 2011)*, Paraty, Brésil, octobre 2011.

[3] Daniel Augot, Morgan Barbier et Caroline Fontaine. Ensuring message embedding in wet paper steganography. *Dans* Liqun Chen, éditeur. *IMA proceedings*. Lecture Notes in Computer Science, décembre 2011.

[4] Morgan Barbier. New set of codes for the maximum-likelyhood decoding problem. *Dans YACC*, octobre 2010.

[5] Morgan Barbier et Paulo Barreto. Key reduction of McEliece's cryptosystem using list decoding. *Dans 2011 IEEE International Symposium on Information Theory (ISIT2011)*, St. Pétersbourg, Russie, juillet 2011.

[6] Morgan Barbier et Pierre-Louis Cayrel. Attaques algébriques. *Dans MajecSTIC*, octobre 2008.

[7] Morgan Barbier, Christophe Chabot et Guillaume Quintin. On quasi-cyclic codes as a generalization of cyclic codes. En soumission, juillet 2011.

[8] Carlos Munuera et Morgan Barbier. Wet paper codes and the dual distance in steganography. *Advances in Mathematics of Communications*, 2011. À apparaître.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Introduction

Durant mon doctorat, je me suis très longuement intéressé aux méthodes de décodage des codes correcteurs, tout particulièrement aux algorithmes de décodage en liste. Dès le début de ma thèse, l'un de mes objectifs principaux était d'appliquer l'algorithme de décodage en liste de Wu [Wu08] aux codes de Goppa binaires pour atteindre la borne de Johnson binaire. Cet axe de recherche s'est malheureusement révélé infructueux. En collaboration avec Daniel Augot et Alain Couvreur, on a indépendamment redécouvert, un résultat méconnu dû à Tal et Roth [TR03], qui permet de décoder en liste jusqu'à la borne de Johnson binaire de ces codes. On détaille la complexité et les paramètres de cette méthode dans un rapport de recherche [ABC10] ; une version courte a également été acceptée dans une conférence internationale [ABC11].

Grâce à l'amélioration du rayon de décodage pour les codes alternants, on s'est intéressé au problème du décodage complet pour ces codes. On propose une nouvelle classe de codes pour laquelle ce problème, montré NP-difficile dans le cas général, se résout en temps polynomial en la longueur du code [Bar10]. Ensuite, on montre que cette classe n'est pas vide et contient des codes non trivaux. En effet, on montre, qu'en plus des codes parfaits linéaires, tous les codes de Reed-Muller du premier ordre poinçonnés et certains codes BCH binaires sont dans cette classe. Je me suis également intéressé aux codes de Goppa binaires, pour lesquelles seuls ceux à très faibles dimensions sont également dans notre classe. On a montré expérimentalement que certains codes BCH binaires, avec des paramètres intéressants, sont dans notre classe proposée.

En collaboration avec Paulo Barreto, on s'est intéressé aux conséquences de notre méthode de décodage en liste pour les codes alternants, à la cryptographie basée sur les codes, comme par exemple le cryptosystème de McEliece. L'une des étapes de chiffrement est basée sur l'insertion d'erreurs. Le nombre d'erreurs à ajouter est donné par le plus grand rayon de décodage du code utilisé. En utilisant notre méthode de décodage en liste qui augmente le rayon de décodage pour les codes alternants, on peut alors ajouter plus d'erreurs que précédemment [BB11]. Cela fournit un niveau de sécurité plus important pour des tailles de clés fixées. Réciproquement, cela permet à niveau de sécurité donné de réduire la taille des clés. Comme cette méthode de décodage s'applique à la grande famille des codes alternants, on propose une réduction de clé pour la version générique et dyadique de McEliece. Dans cet article on propose également

deux contre-mesures pour lesquelles l'attaque basée sur le calcul d'une base de Groebner échoue. On exhibe alors une réduction de clé de 4% pour la variante générique et de 21% pour la variante quasi-dyadique.

Outre l'application de la théorie des codes à la cryptographie, on s'est également intéressé à son application en stéganographie, tout particulièrement aux problèmes d'insertion d'un message basés sur le codage par syndrome. Cependant, cette technique peut échouer. Avec Carlos Munuera, on a regardé ce problème d'un point de vue théorique. On a exprimé des conditions nécessaires et suffisantes pour que le problème du codage par syndrome possède toujours une solution pour les codes linéaires [MB11]. Ces conditions sont fonction de la distance duale du code utilisé. On a également exprimé ces conditions en fonction de la hiérarchie des poids du code. On montre alors que les codes MDS sont des bons candidats pour le problème du codage par syndrome. En effet, plus le défaut de Singleton d'un code est petit, plus il est adapté pour ce problème. Grâce à la nature combinatoire de nos conditions, on est en mesure d'exhiber des conditions suffisantes mais non nécessaires pour l'existence de solution avec l'utilisation des codes systématiques – qui peuvent être vus comme une généralisation des codes linéaires. Ce qui montre que l'utilisation des codes systématiques peut être un meilleur choix pour les schémas stéganographiques basés sur le problème du codage par syndrome avec des positions verrouillées.

En collaboration avec Daniel Augot et Caroline Fontaine, on a également reformulé le problème du codage par syndrome borné avec des positions verrouillées pour garantir l'insertion [ABF11]. Cette reformulation est basée sur l'idée originale du schéma de signature de Courtois, Finiasz et Sendrier, qui consiste à faire varier une petite partie du syndrome. On a étudié ce schéma d'insertion pour tous les codes linéaires parfaits : les deux codes de Golay et les codes de Hamming. Cette reformulation entraîne une légère baisse de l'*embedding efficiency*. Cette perte est logarithmique en le nombre de positions verrouillées pour les codes de Hamming binaires. On a également suggéré une adaptation du schéma d'insertion ZZW utilisant notre méthode. De plus, des formules pour calculer les paramètres de notre reformulation du problème pour un code linéaire quelconque sont exhibées dans cette thèse.

Dans le chapitre 1, on rappelle les principales définitions et propriétés de la théorie des codes, et également les principales classes de codes. Ensuite dans le chapitre 2, on explique les principaux algorithmes de décodages, ainsi que notre méthode de décodage en liste pour les codes alternants. Ce chapitre se termine par la présentation de notre classe de codes construite pour le problème du décodage complet. Le chapitre 3 est dédié à la cryptographie, tout particulièrement aux cryptosystèmes basés sur la théorie des codes : McEliece ou de manière équivalente, Niederreiter. On y explique également la réduction de clé obtenue si on utilise notre méthode de décodage en liste. Enfin, dans le chapitre 4, on rappelle les principaux problèmes de stéganographie basés sur la théorie des codes. De plus, on présente nos conditions pour que l'insertion d'un

message basé sur le problème du codage par syndrome ait une solution. Ces conditions sont nécessaires et suffisantes pour les codes linéaires et juste suffisantes pour les codes systématiques non linéaires. Enfin, ce manuscrit de thèse se termine par le détail de notre reformulation du schéma d'insertion pour garantir l'existence d'une solution.

Chapitre 1

Codes correcteurs d'erreurs

Plan du chapitre

1.1 Introduction

Ce chapitre commence par un rappel sur les codes linéaires en toute généralité et sur une discussion sur leurs paramètres, ce qui introduit les notions de *bons codes*. Ensuite, on présente les définitions des principaux codes étudiés dans cette thèse : codes de Hamming, BCH, de Reed-Solomon, de Goppa et les codes systématiques – vus comme une généralisation des codes linéaires. Ce chapitre se termine par une rapide discussion sur les différents problèmes de décodage.

1.1.1 Codes linéaires

Dans cette thèse, on s'est intéressé principalement aux codes linéaires en blocs. C'est pourquoi on propose de rappeler les principales définitions et propriétés de cette classe de codes. On exhibe ensuite quelques propriétés de la forme matricielle d'un code – c'est-à-dire par des matrices génératrice et de parité. On continue par exposer la définition et

les propriétés de codes équivalents, on finit enfin par quelques propriétés des fonctions syndromes.

Paramètres

Soient q une puissance d'un nombre premier et $\mathbb{F}_q$ le corps fini à q éléments.

Définition 1.1 (Code linéaire)**.** Un $[n,k]_q$-*code linéaire*, noté $\mathcal{C}$, est un $\mathbb{F}_q$-sous espace vectoriel de $\mathbb{F}_q^n$ de dimension k. Un élément $c \in \mathcal{C}$ est un *mot de code*.

Quand la cardinalité du corps de base d'un code n'est pas utilisée, on allège la notation de $\mathcal{C}$ par $\mathcal{C}$ est $[n,k]$-code linéaire.

Définition 1.2 (Espace ambiant)**.** Soit $\mathcal{C}$ un $[n,k]_q$-code linéaire, $\mathbb{F}_q^n$ est appelé l'*espace ambiant* de $\mathcal{C}$. On appelle un *mot* v, un vecteur de l'espace ambiant $\mathbb{F}_q^n$ et on note $v = (v_1, \ldots, v_n)$.

Définition 1.3 (Support)**.** Soit $v \in \mathbb{F}_q^n$. Le *support* de v, noté $Supp(v)$, est donné par :

$$Supp(v) \triangleq \{i \ : \ v_i \neq 0\}.$$

Définition 1.4 (Poids de Hamming)**.** Soit $v \in \mathbb{F}_q^n$, le *poids de Hamming* de v, noté $w(v)$, est défini comme suit :

$$\begin{aligned} w(v) &\triangleq |\{i \ : \ v_i \neq 0\}| \\ &= |Supp(v)|. \end{aligned}$$

Définition 1.5 (Distance de Hamming)**.** Soient $u, v \in \mathbb{F}_q^n$, deux mots de l'espace ambiant. On définit la *distance de Hamming* de u et v, notée $d_H(u,v)$, de la manière suivante :

$$d_H(u,v) \triangleq |\{i \ : \ \forall i \in \{1, \ldots, n\}, \ u_i \neq v_i\}|.$$

Proposition 1.1. *La distance de Hamming est une distance au sens mathématique ; c'est-à-dire pour tous $x, y, z \in \mathbb{F}_q^n$, on a :*

1. $d_H(x,y) = d_H(y,x)$,
2. $d_H(x,y) = 0 \Longleftrightarrow x = y$,
3. $d_H(x,z) \leq d_H(x,y) + d_H(y,z)$.

Alors la distance de Hamming définit une métrique pour $\mathbb{F}_q^n$. Dans cette thèse, on ne s'intéresse qu'à la métrique de Hamming. Par un souci d'allègement de notation, on propose de noter simplement $d(x,y)$ la distance de Hamming entre x et y.

Corollaire 1.2. *Soient $u, v \in \mathbb{F}_q^n$. La distance de Hamming entre u et v est égale au poids de Hamming de $u - v$.*

Définition 1.6 (Distance minimale)**.** Soit $\mathcal{C}$ un $[n,k]_q$-code linéaire. La *distance minimale* du code $\mathcal{C}$, notée $d_{\mathcal{C}}$, est la plus petite distance entre deux mots de code distincts.

$$d_{\mathcal{C}} \triangleq \min_{c \neq c' \in \mathcal{C}} \{d(c,c')\}.$$

Quand il n'y a pas d'ambiguïté sur le code, on se permet de noter d la distance minimale de $\mathcal{C}$. On dit alors que $\mathcal{C}$ est un $[n,k,d]_q$-code linéaire ou plus simplement un $[n,k,d]$-code. La distance minimale d'un code linéaire est également le plus petit poids des mots de code non nuls.

Corollaire 1.3. *Soit $\mathcal{C}$ un $[n,k,d]$-code linéaire. La distance minimale d de $\mathcal{C}$ peut être donnée par :*

$$d = \min_{c \in \mathcal{C} \setminus \{0\}} \{w_H(c)\}.$$

Définition 1.7 (Capacité de correction)**.** Soit $\mathcal{C}$ un code linéaire de distance minimale d. La *capacité de correction* de $\mathcal{C}$ notée t est définie de la manière suivante :

$$t \triangleq \left\lfloor \frac{d-1}{2} \right\rfloor.$$

Théorème 1.4. *Soient $\mathcal{C}$ un $[n,k,d]_q$-code linéaire de capacité de correction t et $v \in \mathbb{F}_q^n$ un mot de l'espace ambiant. S'il existe un mot de code $c \in \mathcal{C}$ tel que $d(v,c) \leq t$ alors il est unique.*

Définition 1.8 (Boule de Hamming)**.** Soient $v \in \mathbb{F}_q^n$ et r un entier positif, on définit la *boule de Hamming* centré en v de rayon r par

$$\mathcal{B}(v,r) \triangleq \left\{ y \in \mathbb{F}_q^n \ : \ d(y,v) \leq r \right\}.$$

Proposition 1.5. *Soit r un entier positif. Pour tout élément $v \in \mathbb{F}_q^n$, la cardinalité de la boule de Hamming centrées en v et de rayon r est*

$$V_q(n,r) = |\mathcal{B}(v,r)| = \sum_{i=0}^{r} (q-1)^i \binom{n}{i}.$$

Proposition 1.6 (Borne de Hamming)**.** *Soit $\mathcal{C}$ un $[n,k]_q$-code linéaire de capacité de correction t, alors*

$$q^k \leq \frac{q^n}{V_q(n,t)}.$$

Théorème 1.7 (Borne de Plotkin)**.** *Soit $\mathcal{C}$ un $[n,k,d]_2$-code linéaire.*

- *Si d est pair et $2d > n$ alors $k < \log_2\left(2\left\lfloor \frac{d}{2d-n} \right\rfloor\right)$.*
- *Si d est impair et $2d+1 > n$ alors $k < \log_2\left(2\left\lfloor \frac{d+1}{2d+1-n} \right\rfloor\right)$.*
- *Si d est pair et $2d = n$ alors $k < \log_2(4d)$.*
- *Si d est impair et $2d+1 = n$ alors $k < \log_2(4d+4)$.*

Une autre quantité importante d'un code est le rayon de recouvrement, pour l'introduire, on définit d'abord la distance d'un point à un ensemble.

Définition 1.9 (Distance)**.** Soient E un ensemble inclus dans $\mathbb{F}_q^n$ et $v \in \mathbb{F}_q^n$, on définit la *distance* de v à E de la manière suivante :

$$d(v, E) \triangleq \min_{e \in E}\{d(v, e)\}.$$

Le rayon de recouvrement d'un code est la plus grande distance qui sépare un élément de son espace ambiant à ce code.

Définition 1.10 (Rayon de recouvrement)**.** Soit $\mathcal{C}$ un $[n, k]_q$-code linéaire, le *rayon de recouvrement* de $\mathcal{C}$, noté ρ, est défini comme la plus grande distance entre les points de l'espace ambiant du code et le code :

$$\begin{aligned} \rho &\triangleq \max_{v \in \mathbb{F}_q^n}\{d(v, \mathcal{C})\} \\ &= \max_{v \in \mathbb{F}_q^n}\left\{\min_{c \in \mathcal{C}}\{d(v, c)\}\right\}. \end{aligned}$$

Remarque 1.1. On peut également voir le rayon de recouvrement ρ comme le plus petit rayon tel que toutes les boules de rayon ρ centrées en les mots du code recouvrent totalement l'espace ambiant, ce qui explique son nom.

Définition 1.11 (Code parfait)**.** Soient $\mathcal{C}$ un code, t la capacité de correction et ρ le rayon de recouvrement de $\mathcal{C}$. Le code $\mathcal{C}$ est un *code parfait* si

$$t = \rho.$$

Remarque 1.2. Soit $\mathcal{C}$ un code parfait de capacité de correction t et d'espace ambiant $\mathbb{F}_q^n$, alors

$$\mathbb{F}_q^n = \bigsqcup_{c \in \mathcal{C}} \mathcal{B}(c, t).$$

On obtient alors l'égalité pour la *borne de Hamming*, c'est-à-dire

$$q^k = \frac{q^n}{V_q(n, t)}.$$

Formes matricielles

Définition 1.12 (Matrice génératrice)**.** Soit $\mathcal{C}$ un $[n, k]_q$-code linéaire, on appelle G une *matrice génératrice* de $\mathcal{C}$, une représentation matricielle de la fonction de $\mathbb{F}_q^k$ dans $\mathbb{F}_q^n$ ayant pour image le code $\mathcal{C}$. Cette application est alors

$$\begin{array}{ccc} \mathbb{F}_q^k & \longrightarrow & \mathbb{F}_q^n \\ m & \longmapsto & mG. \end{array}$$

On parle d'*une* matrice génératrice d'un code car il n'y a pas unicité. Une matrice génératrice d'un $[n,k]_q$-code linéaire est une matrice à k lignes et n colonnes à coefficients dans $\mathbb{F}_q$ de rang k.

Remarque 1.3. Soit G une matrice génératrice d'un code linéaire $\mathcal{C}$, alors les lignes de G forment une base de $\mathcal{C}$.

Définition 1.13 (Matrice de parité)**.** Soit $\mathcal{C}$ un $[n,k]_q$-code linéaire, on appelle H une *matrice de parité* de $\mathcal{C}$, si H^t est une représentation matricielle de la fonction de $\mathbb{F}_q^n$ dans $\mathbb{F}_q^{n-k}$ ayant pour noyau le code $\mathcal{C}$. Cette application est alors

$$\begin{array}{rcl} \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_q^{n-k} \\ v & \longmapsto & vH^t. \end{array}$$

De même que pour la matrice génératrice, il n'y a pas unicité de la matrice de parité d'un $[n,k]_q$-code linéaire. C'est une matrice à $(n-k)$ lignes et n colonnes d'éléments de $\mathbb{F}_q$ et de rang $(n-k)$.

Définition 1.14 (Code dual)**.** Soient $\mathcal{C}$ un $[n,k]$-code linéaire et H une matrice de parité de $\mathcal{C}$. Le *code dual* de $\mathcal{C}$ noté $\mathcal{C}^\perp$ est un $[n,n-k]$-code linéaire dont H est une matrice génératrice.

Remarque 1.4. Soit $\mathcal{C}$ un code linéaire alors $\mathcal{C} = \left(\mathcal{C}^\perp\right)^\perp$.

Comme la matrice génératrice est une représentation matricielle de l'application linéaire qui a pour image le code, on peut définir un code uniquement par une de ses matrices génératrices. Pour des arguments similaires, il en est de même pour la matrice de parité. Il est important de se rappeler qu'une matrice génératrice ou de parité n'est pas unique. La proposition suivante en est une bonne illustration.

Proposition 1.8. *Soient $\mathcal{C}$ un $[n,k]_q$-code linéaire de matrice de parité H, M une matrice sur $\mathbb{F}_q$ de taille $(n-k)\times(n-k)$ inversible et $\mathcal{C}'$ le code ayant pour matrice de parité MH, alors*

$$\mathcal{C} = \mathcal{C}'.$$

Proposition 1.9. *Soient $\mathcal{C}$ un $[n,k,d]$-code linéaire et H une matrice de parité de $\mathcal{C}$ alors tout sous-ensemble de $d-1$ colonnes de H est libre et il existe un sous-ensemble de d colonnes de H liées.*

Corollaire 1.10 (Borne de Singleton)**.** *Soit $\mathcal{C}$ un $[n,k,d]$-code linéaire, alors*

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

Définition 1.15 (Code MDS)**.** Un $[n,k,d]$-code linéaire est appelé *Maximum Distance Separable (MDS)* si $d = n-k+1$.

Définition 1.16 (Défaut de Singleton)**.** Soit $\mathcal{C}$ un $[n,k,d]$-code linéaire. La différence entre la borne de Singleton et la distance minimale du code est le *défaut de Singleton* qui est

$$n-k+1-d.$$

Définition 1.17 (Hiérarchie des poids)**.** Soient $\mathcal{C}$ un $[n,k]$-code linéaire et g tel que $1 \leq g \leq k$, le g^{e} poids de Hamming généralisé de $\mathcal{C}$ est

$$d_g(\mathcal{C}) = \min\{\#\text{Supp}(L) : L \text{ est un sous-espace vectoriel de } \mathcal{C} \text{ de dimension } g\}$$

où $\text{Supp}(L) = \cup_{\mathbf{x}\in L}\text{Supp}(\mathbf{x})$. La suite $d_1(\mathcal{C}), \ldots, d_k(\mathcal{C})$ est la *hiérarchie des poids* de $\mathcal{C}$.

Par soucis de lisibilité, on écrit $d_1, \ldots, d_k$ la hiérarchie des poids d'un $[n,k]$-code linéaire. Il y a deux propriétés importantes sur la hiérarchie des poids d'un code : la monotonie et la dualité.

Proposition 1.11 (Monotonie)**.** *Soient $\mathcal{C}$ un $[n,k,d]$-code linéaire et $d_1, \ldots, d_k$ sa hiérarchie des poids. Alors*

$$d_1(\mathcal{C}) < d_2(\mathcal{C}) < \cdots < d_k(\mathcal{C})$$

Proposition 1.12 (Dualité)**.** *Soient $\mathcal{C}$ un $[n,k,d]$-code linéaire, $d_1, \ldots, d_k$ la hiérarchie des poids de $\mathcal{C}$ et $d_1^\perp, \ldots, d_{n-k}^\perp$ la hiérarchie des poids du dual $\mathcal{C}^\perp$. Alors*

$$\{d_1, \ldots, d_k\} \cup \left\{n+1-d_1^\perp, \ldots, n+1-d_{n-k}^\perp\right\} = \{1, \ldots, n\}$$

Définition 1.18 (Rang MDS)**.** Soit $\mathcal{C}$ un $[n,k]$-code linéaire tel que son i^{e} poids de Hamming généralisé est $d_i = n-k+i$, on définit le *rang MDS* de $\mathcal{C}$ comme le plus petit entier g tel que $d_g = n-k+g$.

Remarque 1.5. Le rang MDS d'un code MDS est alors 1.

Équivalence de codes

Définition 1.19 (Codes équivalents)**.** Soient $\mathcal{C}_1$ et $\mathcal{C}_2$ deux codes linéaires. On dit que $\mathcal{C}_1$ et $\mathcal{C}_2$ sont *équivalents*, si $\mathcal{C}_2$ est obtenu par combinaison des transformations suivantes de $\mathcal{C}_1$

- une permutation de composantes,
- une multiplication par un élément de $\mathbb{F}_q^*$ sur une composante.

Théorème 1.13. *Soient G et G' deux matrices $(k \times n)$ à coefficients dans $\mathbb{F}_q$. Le code engendré par G est équivalent au code engendré par G' si G' est obtenue par une combinaison des transformations suivantes de G*

1. *permutation de lignes,*
2. *multiplication d'une ligne par un élément de $\mathbb{F}_q^*$,*
3. *addition de deux lignes,*
4. *permutation de colonnes,*
5. *multiplication d'une colonne par un élément de $\mathbb{F}_q^*$,*

Définition 1.20 (Forme systématique)**.** Soit G une matrice génératrice d'un code linéaire. On dit que G est sous *forme systématique* si elle est de la forme

$$G = \left(\begin{array}{c|c} I_k & A \end{array} \right),$$

où I_k est la matrice identité à k lignes et k colonnes et A est une matrice à k lignes et $(n-k)$ colonnes.

Définition 1.21 (Ensemble d'information)**.** Soit $\mathcal{C}$ un code de longueur n et de dimension k sur $\mathbb{F}_q^n$. L'ensemble $\mathcal{I} \subset \{1, \ldots, n\}$ est appelé *ensemble d'information* si $|\mathcal{I}| = k$ et

$$\forall c \neq c' \in \mathcal{C},\ \pi_{\mathcal{I}}(c) \neq \pi_{\mathcal{I}}(c'),$$

où $\pi_{\mathcal{I}}$ est la projection sur $\mathcal{I}$.

Lorsqu'une matrice génératrice d'un code linéaire est sous forme systématique, alors les k premières positions des mots de code sont des symboles d'information et les $(n-k)$ dernières sont des symboles de redondance. Le fait de dissocier les positions d'information avec les positions de redondance vient des codes systématiques.

Théorème 1.14. *Tout code linéaire admet un code équivalent ayant une matrice génératrice sous forme systématique.*

Proposition 1.15. *Soient $\mathcal{C}$ un code linéaire et G la matrice génératrice de $\mathcal{C}$ sous forme systématique, c'est-à-dire $G = \left(\begin{array}{c|c} I_k & A \end{array} \right)$. Alors une matrice de parité de $\mathcal{C}$ est*

$$H = \left(\begin{array}{c|c} -A^t & I_{n-k} \end{array} \right).$$

Fonctions syndromes

Définition 1.22 (Fonction syndrome)**.** Soient $\mathcal{C}$ un $[n,k]_q$-code linéaire et H une matrice de parité de $\mathcal{C}$. La *fonction syndrome* associée à H est définie comme suit :

$$\begin{array}{rcl} S_H : \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_q^{n-k} \\ v & \longmapsto & vH^t. \end{array}$$

Définition 1.23 (Syndrome)**.** Soient $\mathcal{C}$ un $[n,k]_q$-code linéaire et H une matrice de parité de $\mathcal{C}$. Pour tout élément v dans $\mathbb{F}_q^n$, on appelle *syndrome* de v la quantité $S_H(v)$.

Définition 1.24 (Coset leader)**.** Soient $\mathcal{C}$ un $[n,k]$-code linéaire, H une matrice de parité de $\mathcal{C}$ et $s \in \mathbb{F}_q^{n-k}$. Le *coset leader* de s est le mot de plus petit poids de l'espace ambiant qui a pour syndrome s.

Proposition 1.16. *Soient $\mathcal{C}$ un $[n,k,d]_q$-code linéaire, H une matrice de parité de $\mathcal{C}$, $s \in \mathbb{F}_q^{n-k}$ et $e \in \mathbb{F}_q^n$ le* coset leader *de s. Alors le mot du code $\mathcal{C}$ le plus proche de e est $\mathbf{0}$.*

Proposition 1.17. *Soient $\mathcal{C}$ un code linéaire et H une matrice de parité de $\mathcal{C}$. Tous les* coset leader *de $\mathcal{C}$ associés à H ont un poids de Hamming inférieur au rayon de recouvrement de $\mathcal{C}$.*

Théorème 1.18. *Soient $\mathcal{C}$ un code linéaire et t la capacité de correction de $\mathcal{C}$. Alors pour tout mot v de l'espace ambiant, la fonction syndrome restreinte à $\mathcal{B}(v,t)$ est injective.*

Démonstration. On propose de faire la démonstration par contradiction. Soient $v \in \mathbb{F}_q^n$ un élément de l'espace ambiant, H une matrice de parité de $\mathcal{C}$, et v_1, v_2 deux éléments différents de la boule centrée en v de rayon t, c'est-à-dire $v_1, v_2 \in \mathcal{B}(v,t)$. Supposons maintenant que le syndrome de v_1 est égal au syndrome de v_2 : $S_H(v_1) = S_H(v_2)$. Comme la fonction syndrome est linéaire, on a $S_H(v_1 - v_2) = 0$ et donc $(v_1 - v_2)$ est un mot de code. Par le corollaire 1.3, on en déduit que $w(v_1 - v_2) \geq d$. Étudions maintenant le poids de Hamming de $v_1 - v_2$. Comme $v_1, v_2 \in \mathcal{B}(v,t)$, il existe $e_1, e_2 \in \mathbb{F}_q^n$ tels que :

$$v_1 = v + e_1 \text{ avec } w(e_1) \leq t \text{ et } v_2 = v + e_2 \text{ avec } w(e_2) \leq t.$$

Ainsi

$$\begin{aligned} w(v_1 - v_2) &= w(v + e_1 - v - e_2), \\ &= w(e_1 - e_2), \\ &\leq w(e_1) + w(e_2), \\ &\leq 2t, \\ &< d. \end{aligned}$$

Ce qui contredit que $v_1 - v_2$ est un mot de code. □

Théorème 1.19. *Soient $\mathcal{C}$ un code linéaire et ρ le rayon de recouvrement de $\mathcal{C}$. Alors pour tout mot de l'espace ambiant v, la fonction syndrome restreinte sur $\mathcal{B}(v,\rho)$ est surjective.*

Démonstration. Soient $\mathcal{C}$ un $[n,k]_q$-code linéaire de rayon de recouvrement ρ et H une matrice de parité de $\mathcal{C}$. Soit $v \in \mathbb{F}_q^n$, un mot de l'espace ambiant. On va montrer que pour tout syndrome $s_1 \in \mathbb{F}_q^{n-k}$, il existe $v_1 \in \mathbb{F}_q^n$ tel que $S_H(v_1) = s_1$ et $d(v, v_1) \leq \rho$. Soient $s \in \mathbb{F}_q^{n-k}$ le syndrome de v et e le *coset leader* de $s_1 - s$. Alors

$$\begin{aligned} S_H(e + v) &= S_H(e) + S_H(v) \\ &= s_1 - s + s \\ &= s_1, \end{aligned}$$

et $d(e+v, v) = d(e, \mathbf{0}) = w(e) \leq \rho$, par la proposition 1.17. Il suffit de prendre $v_1 \triangleq e + v$. □

Corollaire 1.20. *Soit $\mathcal{C}$ un code parfait de capacité de correction t. Alors pour tout mot de l'espace ambiant v, la fonction syndrome définie sur $\mathcal{B}(v,t)$ est bijective.*

Définition 1.25 (Code translaté)**.** Soient $\mathcal{C}$ un $[n,k]_q$-code linéaire et $v \in \mathbb{F}_q^n$. Le *code translaté* de $\mathcal{C}$ engendré par v est l'ensemble

$$v + \mathcal{C} \triangleq \{v + c \ : \ c \in \mathcal{C}\}.$$

Un code translaté est aussi appelé *coset*, c'est de là que vient le terme de *coset leader*.

Théorème 1.21. *Soient $\mathcal{C}$ un $[n,k]_q$-code linéaire, H une matrice de parité de $\mathcal{C}$ et $v \in \mathbb{F}_q^n$, alors $c' \in v + \mathcal{C}$ si et seulement si $S_H(c') = S_H(v)$.*

Corollaire 1.22. *Soient $\mathcal{C}$ un $[n,k]_q$-code linéaire, H une matrice de parité de $\mathcal{C}$ et v un élément de $\mathbb{F}_q^n$. Alors on peut redéfinir*

$$v + \mathcal{C} = \left\{ y \in \mathbb{F}_q^n \ : \ S_H(y) = S_H(v) \right\}.$$

Distributions

Définition 1.26 (Distribution des poids)**.** Soit $\mathcal{C}$ un $[n,k]$-code linéaire, la *distribution des poids* de $\mathcal{C}$ est la suite $A_0, \ldots, A_n$, où A_i est le nombre de mots de code $\mathcal{C}$ de poids i.

À partir de la distribution des poids d'un code linéaire, on est capable de calculer la distribution des poids de son code dual.

Théorème 1.23. *Soient $\mathcal{C}$ un code linéaire et $A_0, \ldots, A_n$ la distribution des* coset leader *de $\mathcal{C}$, alors la distribution des* coset leader *de $\mathcal{C}^\perp$ est donnée par*

$$A_i^\perp = \frac{1}{\#\mathcal{C}} \sum_{j=0}^{n} A_j K_i(j),$$

où $K_i(X) = \sum_{j=0}^{i} (-1)^j \binom{X}{j} \binom{n-X}{i-j}$ le i^e polynôme de Krawtchouk.

Une autre distribution qui est importante dans un contexte de décodage est la distribution des poids des *coset leader*.

Définition 1.27 (Distribution des poids des *coset leader*)**.** Soit $\mathcal{C}$ un $[n,k]$-code linéaire, la *distribution des poids des coset leader* est la suite $\alpha_0, \ldots, \alpha_n$, où α_i est le nombre de *coset leader* de poids i.

Proposition 1.24. *Soient $\mathcal{C}$ un $[n,k,d]_q$-code linéaire et t sa capacité de correction alors pour tout $i \leq t$ le nombre de* coset leader *de poids i est*

$$\alpha_i = (q-1)^i \binom{n}{i}.$$

La proposition précédente donne exactement la distribution des *coset leader* pour tous les i inférieurs à t, la capacité de correction de $\mathcal{C}$. Cependant, le calcul de α_i pour $i > t$ est un problème classique en théorie des codes, il est considéré comme difficile. Lorsqu'on est en mesure de connaître la distribution des poids des *coset leader*, on peut calculer le rayon de recouvrement moyen.

Définition 1.28 (Rayon de recouvrement moyen)**.** Soit $\mathcal{C}$ un $[n,k]_q$-code linéaire, alors le *rayon de recouvrement moyen* de $\mathcal{C}$ est

$$\tilde{\rho} = q^{k-n} \sum_{i=1}^{n} i\alpha_i.$$

Un problème de recherche consiste à calculer la distribution des poids des *coset leader* d'un code linéaire à partir de la distribution des poids des *coset leader* du code dual [JP09].

1.1.2 Bons codes

En théorie des codes correcteurs, il existe plusieurs notions de bons codes. Plus exactement, il existe deux définitions qui correspondent à deux points de vue différents. L'un d'entre eux consiste à regarder une famille de codes et de regarder si le rendement et la distance minimale normalisée sont strictement positifs lorsque n la longueur tend vers l'infini. C'est ce qui est appelé des *codes asymptotiquement bons*. Cette famille de codes est difficile à trouver, ils peuvent être très intéressants pour des applications qui nécessitent des fortes longueurs de codes. L'autre point de vue consiste pour un corps, une longueur et une dimension fixées, de construire un code avec la plus haute distance minimale.

Codes asymptotiquement bons

Définition 1.29 (Codes asymtotiquement bons)**.** Soient q fixé, une puissance d'un nombre premier, $(\mathcal{C}_n)$ une famille de codes sur $\mathbb{F}_q$ indexés par leur longueurs, $d(\mathcal{C}_n)/n$ leur distance minimale normalisée et $R_n = k_n/n$ leur rendement. La famille de codes $(\mathcal{C}_n)$ est *asymptotiquement bonne* si

$$\begin{cases} \lim_{n\to+\infty} d(\mathcal{C}_n)/n > 0, \\ \lim_{n\to+\infty} R_n = k_n/n > 0. \end{cases}$$

Exemple 1.6. Les codes de Reed-Muller du premier ordre, les $RM(1,m)$ sont des $[2^m, m+1, 2^{m-1}]_2$-codes linéaires. Alors la distance minimale normalisée est $d(\mathcal{C})_n/n = 1/2$ et le rendement est $R_n = k_n/n = (\log_2(n)+1)/n$. Il est clair que le rendement tend vers zéro lorsque n tend vers l'infini. Les codes de Reed-Muller du premier ordre ne sont donc pas des codes asymptotiquement bons.

Exemple 1.7. Les codes de Hamming binaires de paramètre m sont des $[2^m-1, 2^m-1-m, 3]_2$-codes linéaires, ses paramètres sont montrés dans la section 1.2. On a alors que $R_n = k_n/n \approx 1 - \log_2(n)/n$ et $d(\mathcal{C}_n)/n = 3/n$. On a clairement que la limite de la distance minimale normalisée est zéro. Donc les codes de Hamming ne sont pas des codes asymptotiquement bons.

Les codes de Reed-Muller du premier ordre ne constituent pas une famille asymptotiquement bonne à cause du rendement, pour les codes de Hamming c'est la distance minimale normalisée qui fait défaut. On voit qu'en général, il est difficile de satisfaire les deux conditions en même temps.

Exemple 1.8. Les codes de Reed-Solomon sont des codes MDS, ils sont donc des $[n, k, n-k+1]_q$-codes linéaires. On prouve ses paramètres dans la section 1.4. Si on prend comme famille les codes de Reed-Solomon de rendement $0,5$. On a $R_n = k_n/n = 0,5$ et pour distance minimale normalisée $d(\mathcal{C}_n)/n = 1 - R_n + 1/n$. Les

limites pour le rendement et pour la distance minimale normalisée sont strictement positives. Cependant on voit dans la section 1.4 que leur défaut principal est la taille du corps sur lequel ils sont définis, c'est-à-dire $q > n$, or q est fixé, on en déduit alors que les codes de Reed-Solomon ne sont pas des codes asymptotiquement bons.

Paramètres fixés

L'idée est de se donner un corps de base, une longueur ainsi qu'une dimension, et de construire des codes avec ces paramètres, tels que leur distance minimale soit supérieure aux codes précédents. Pour cela le site `codetables.de` de Markus Grassl est une formidable base de données mise à jour régulièrement [Gra07]. Avec Christophe Chabot et Guillaume Quintin, on a proposé des nouvelles classes de codes quasi-cycliques à partir desquelles on arrive à construire 49 nouveaux codes qui ont des distances minimales plus grandes que celles connues auparavant [BCQ11]. En effet, on est arrivé à construire 5 nouveaux codes sur $\mathbb{F}_4$ et en les transformant avec le poinçonnage décrit dans [GW04], Markus Grassl a réussi a exhiber 44 autres nouveaux codes à partir de nos 5 premiers. Les 49 nouveaux codes sont listés dans la table 1.1.

Nouveaux bons codes sur $\mathbb{F}_4$				
$[171, 11, 109]_{\mathbb{F}_4}$	$[172, 11, 110]_{\mathbb{F}_4}$	$[173, 11, 110]_{\mathbb{F}_4}$	$[174, 11, 111]_{\mathbb{F}_4}$	$[175, 11, 112]_{\mathbb{F}_4}$
$[176, 11, 113]_{\mathbb{F}_4}$	$[177, 11, 114]_{\mathbb{F}_4}$	$[178, 11, 115]_{\mathbb{F}_4}$	$[179, 11, 115]_{\mathbb{F}_4}$	$[180, 11, 116]_{\mathbb{F}_4}$
$[181, 11, 117]_{\mathbb{F}_4}$	$[182, 11, 118]_{\mathbb{F}_4}$	$[183, 11, 119]_{\mathbb{F}_4}$	$[184, 10, 121]_{\mathbb{F}_4}$	$[184, 11, 120]_{\mathbb{F}_4}$
$[185, 10, 122]_{\mathbb{F}_4}$	$[185, 11, 121]_{\mathbb{F}_4}$	$[186, 10, 123]_{\mathbb{F}_4}$	$[186, 11, 122]_{\mathbb{F}_4}$	$[187, 10, 124]_{\mathbb{F}_4}$
$[187, 11, 123]_{\mathbb{F}_4}$	$[188, 10, 125]_{\mathbb{F}_4}$	$[188, 11, 124]_{\mathbb{F}_4}$	$[189, 10, 126]_{\mathbb{F}_4}$	$[189, 11, 125]_{\mathbb{F}_4}$
$[190, 10, 127]_{\mathbb{F}_4}$	$[190, 11, 126]_{\mathbb{F}_4}$	$[191, 10, 128]_{\mathbb{F}_4}$	$[191, 11, 127]_{\mathbb{F}_4}$	$[192, 11, 128]_{\mathbb{F}_4}$
$[193, 11, 128]_{\mathbb{F}_4}$	$[194, 11, 128]_{\mathbb{F}_4}$	$[195, 11, 128]_{\mathbb{F}_4}$	$[196, 11, 129]_{\mathbb{F}_4}$	$[197, 11, 130]_{\mathbb{F}_4}$
$[198, 11, 130]_{\mathbb{F}_4}$	$[199, 11, 131]_{\mathbb{F}_4}$	$[200, 11, 132]_{\mathbb{F}_4}$	$[201, 10, 133]_{\mathbb{F}_4}$	$[201, 11, 132]_{\mathbb{F}_4}$
$[202, 10, 134]_{\mathbb{F}_4}$	$[202, 11, 132]_{\mathbb{F}_4}$	$[203, 10, 135]_{\mathbb{F}_4}$	$[204, 10, 136]_{\mathbb{F}_4}$	$[204, 11, 133]_{\mathbb{F}_4}$
$[205, 11, 134]_{\mathbb{F}_4}$	$[210, 11, 137]_{\mathbb{F}_4}$	$[213, 11, 139]_{\mathbb{F}_4}$	$[214, 11, 140]_{\mathbb{F}_4}$	

TABLE 1.1 – Nos 49 nouveaux codes sur $\mathbb{F}_4$ battant les anciennes bornes (B., Chabot, Quintin - 2011) [BCQ11].

1.2 Codes de Hamming

Richard Hamming proposa dans les années 1940, lorsqu'il travaillait dans les laboratoires de Bell, les codes dits de Hamming. Ce sont des codes qui sont importants tant d'un point de vue historique que d'un point de vue théorique. C'est la seule famille de codes non-triviaux qui sont à la fois linéaires et parfaits. On propose de définir les codes de Hamming par leur matrice de parité.

Définition 1.30 (Code de Hamming)**.** Soit m un entier positif. Les colonnes d'une matrice de parité du *code de Hamming* sont tous les vecteurs non nuls, libres deux à deux et de longueur m sur $\mathbb{F}_q$.

Proposition 1.25. *Soit $\mathcal{C}$ un code de Hamming défini sur $\mathbb{F}_q$ et de paramètre m. Alors la longueur de $\mathcal{C}$ est $n \triangleq (q^m-1)/(q-1)$, sa dimension est $n-m$ et sa distance minimale est 3.*

Exemple 1.9. Soit $m = 3$, alors une matrice de parité du code de Hamming binaire de longueur $n = 2^3 - 1$ est

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Proposition 1.26. *Le rayon de recouvrement d'un code de Hamming est égal à 1.*

Comme le rayon de recouvrement d'un code de Hamming est égale à sa capacité de correction $t \triangleq \left\lfloor \frac{d-1}{2} \right\rfloor$, on obtient le corollaire suivant.

Corollaire 1.27. *Un code de Hamming est un code parfait.*

1.3 Codes BCH

Indépendemment, le mathématicien français Alexis Hocquenghem dans [Hoc59] et les mathématiciens indiens Raj Chandra Bose et Dwijendra Chaudhuri dans [BC60] ont introduit les codes appelés aujourd'hui BCH. Dans [BC60], les auteurs sont arrivés à la définition des codes BCH, en s'intéressant à des sous-codes des codes de Hamming.

1.3.1 Définitions

Définition 1.31 (Code cyclique)**.** Soit $\mathcal{C}$ un code de longueur n. Le code $\mathcal{C}$ est un *code cyclique* si et seulement si

$$c \triangleq (c_1, \ldots, c_n) \in \mathcal{C} \Longrightarrow (c_n, c_1, \ldots, c_{n-1}) \in \mathcal{C}.$$

Notation 1.10. Comme $\mathbb{F}_{q^m}^n$ est isomorphe en tant qu'espace vectoriel à l'espace vectoriel des polynômes univariés de $\mathbb{F}_{q^m}$ de degré strictement inférieur à n, on a pour tout $\alpha \in \mathbb{F}_{q^m}$

$$\begin{array}{rcccl} ev_\alpha : \mathbb{F}_{q^m}^n & \longrightarrow & \mathbb{F}_{q^m}[X] & \longrightarrow & \mathbb{F}_{q^m} \\ v = (v_1, \ldots, v_n) & \longmapsto & \sum_{i=1}^n v_i X^{i-1} & \longmapsto & \sum_{i=1}^n v_i \alpha^{i-1}. \end{array}$$

Définition 1.32 (Code BCH – Code d'annulation)**.** Soient q le cardinal du corps, n, m, δ, b quatre entiers tels que $pgcd(q, n) = 1$, m le plus petit entier tel que $q^m \equiv 1 \mod n$ et α une racine primitive n^{e} de l'unité sur $\mathbb{F}_{q^m}$. Alors le *code BCH* est défini par

$$\mathcal{C}_{BCH} = \left\{ v \in \mathbb{F}_q^n \ : \ \forall i \in \{0, \ldots, \delta - 2\}, \ ev_{\alpha^{b+i}}(v) = 0 \right\}.$$

Lemme 1.28. *Soient $\mathcal{C}$ un code BCH et H la matrice définie par*

$$H = \begin{pmatrix} 1 & \alpha^b & (\alpha^b)^2 & \dots & (\alpha^b)^{n-1} \\ 1 & \alpha^{b+1} & (\alpha^{b+1})^2 & \dots & (\alpha^{b+1})^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \alpha^{b+\delta-2} & (\alpha^{b+\delta-2})^2 & \dots & (\alpha^{b+\delta-2})^{n-1} \end{pmatrix}.$$

Alors la matrice constituée des éléments de H développés en colonne sur $\mathbb{F}_q$ est une matrice de parité de $\mathcal{C}$.

Définition 1.33 (Code BCH primitif au sens strict)**.** Avec les notations précédentes, le code BCH défini est appelé un code BCH au *sens strict* si $b = 1$; il est dit *primitif* si $n = q^m - 1$.

Définition 1.34 (Polynôme générateur)**.** Avec les notations précédentes, on appelle *polynôme générateur* d'un code BCH le polynôme $g(X) \in \mathbb{F}_q[X]$ défini par

$$g(X) \triangleq ppcm(m_b(X), \dots, m_{b+\delta-2}(X)),$$

où $m_i(X)$ est le polynôme minimal de α^i.

Proposition 1.29. *Soit un code BCH de polynôme générateur $g(X)$. Alors une matrice génératrice du code est*

$$G \triangleq \begin{pmatrix} g_0 & g_1 & \dots & g_{\deg(g)} & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_{\deg(g)} & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \dots & \ddots & 0 \\ 0 & \dots & 0 & g_0 & g_1 & \dots & g_{\deg(g)} \end{pmatrix},$$

où g_i est le coefficient monomial de degré i de $g(X)$.

Proposition 1.30. *Soit $\mathcal{C}$ un code BCH de polynôme générateur $g(X)$ et de distance construite δ, alors $\dim(\mathcal{C}) = n - \deg(g)$ et la distance minimale a pour minoration $d_{\mathcal{C}} \geq \delta$.*

1.3.2 Équation clé

L'équation qu'on présente ici, appelée *équation clé*, permet de mettre en relation des polynômes qui caractérisent l'erreur et le syndrome. En résolvant cette équation, on est alors en mesure de déterminer l'erreur et ainsi le mot de code original. Quelques algorithmes de décodage résolvent cette équation pour retrouver le mot de code transmis – par exemple la méthode d'Euclide et celle de Berlekamp-Massey.

Contexte de décodage

Soient $\mathcal{C}$ un $[n,k]_q$-code BCH au sens strict et H une matrice de parité de $\mathcal{C}$. Supposons qu'on souhaite transmettre le mot de code $c \in \mathcal{C}$ et que, durant la transmission, une erreur provoquée par le canal se produit. Le récepteur est alors en possession du mot reçu

$$y \triangleq c + e,$$

avec $y, e \in \mathbb{F}_q^n$ et $c \in \mathcal{C}$. Le problème est de retrouver c à partir de la connaissance du code $\mathcal{C}$ et du mot reçu y. Si on applique la fonction syndrome associée à H sur y, on obtient :

$$\begin{aligned} S_H(y) &= yH^t, \\ &= (c+e)H^t, \\ &= eH^t. \end{aligned}$$

D'après la définition 1.13 de la matrice de parité, H est une représentation matricielle d'une application linéaire qui a pour noyau le code, en appliquant la fonction syndrome à un mot reçu y, on obtient alors de l'information uniquement sur l'erreur. De plus, en supposant que le poids de e soit inférieur à t la fonction syndrome est injective, théorème 1.18. On est donc en mesure de déterminer l'unique mot de code c à distance t de y.

Équation clé

L'erreur e est un élément de $\mathbb{F}_q^n$, donc on peut l'écrire sous la forme vectorielle $e = (e_1, \ldots, e_n)$. On note w le poids de l'erreur e, c'est-à-dire $Supp(e) = \{i_1, \ldots, i_w\}$. Pour alléger les notations et pour également respecter la présentation standard de l'équation de clé, on note

$$\forall j \in \{1, \ldots, w\}\,,\ x_j \triangleq \alpha^{i_j - 1} \text{ et } z_j \triangleq e_{i_j}.$$

On définit ensuite deux polynômes importants : le polynôme *localisateur* Λ et le polynôme *évaluateur* L de la manière suivante :

$$\Lambda(X) \triangleq \prod_{i=1}^{w} (1 - X x_i)$$

et

$$L(X) \triangleq \sum_{i=1}^{w} x_i z_i \prod_{j=1,\ j \neq i}^{w} (1 - X x_j).$$

On remarque que les racines du polynôme localisateur $\Lambda(X)$ sont particulières, toutes sont des puissances de α. Si $j \in \{1, \ldots, w\}$, $\alpha^{-i_j - 1}$ est une racine de $\Lambda(X)$, ceci signifie qu'il y a une erreur à la position i_j. Les racines de ce polynôme, appelé polynôme localisateur, permettent de connaître la position des erreurs. De plus, si on évalue le

polynôme évaluateur $L(X)$ en α^{-i_j-1} cela nous donne $\prod_{i\in Supp(e)\setminus\{i_j\}}(\alpha^{i_j-1}-\alpha^{i-1})e_{i_j}$, on est ainsi capable de connaître la valeur de l'erreur en cette position.

Dans un premier temps, on propose d'exhiber une relation entre le syndrome et ces deux polynômes. Pour ce faire, on propose de diviser le polynôme évaluateur $L(X)$ par le polynôme localisateur $\Lambda(X)$.

$$\begin{aligned}
L(X)/\Lambda(X) &= \sum_{i=1}^{w} x_i z_i \prod_{j=1,\ j\neq i}^{w} (1-Xx_j) \ / \ \prod_{i=1}^{w}(1-Xx_i),\\
&= \sum_{i=1}^{w} \frac{x_i z_i}{(1-Xx_i)},\\
&= \sum_{i=1}^{w} x_i z_i \left(\sum_{j=1}^{\infty}(Xx_i)^{j-1}\right),\\
&= \sum_{j=1}^{\infty} X^{j-1} \sum_{i=1}^{w}\left(z_i x_i^j\right),\\
&= \sum_{j=1}^{\infty} X^{j-1} ev_{\alpha^j}(e),\\
&\triangleq S_\infty(X),\\
L(X) &= \Lambda(X) S_\infty(X).
\end{aligned}$$

On appelle $S_\infty(X)$ la *série syndrome* de l'erreur e. Par la définition 1.32, on sait que $\forall j \in \{1,\dots,\delta-1\}$, $ev_{\alpha^j}(c)=0$ et donc $ev_{\alpha^j}(y)=ev_{\alpha^j}(e)$. Le destinataire est capable de calculer les $\delta-1$ premiers termes de $S_\infty(X)$. De plus, la dernière égalité devient modulo X^δ :

$$L(X) \equiv \Lambda(X) S_\delta(X) \mod X^\delta \tag{1.1}$$

où $S_\delta(X) \triangleq S_\infty(X) \mod X^\delta$ et est de degré au plus $\delta-1$. Cette équation est appelée *équation clé*.

1.4 Codes de Reed-Solomon

On propose de définir les codes de Reed-Solomon comme des codes d'évaluation. Pour cela, on a besoin de poser les notations suivantes :

Notation 1.11. Soient $\mathbb{F}$ un corps et $k \in \mathbb{N}$. On note $\mathcal{P}_k(\mathbb{F})$ l'ensemble des polynômes dans $\mathbb{F}[X]$ de degré strictement inférieur à k

$$\mathcal{P}_k(\mathbb{F}) \triangleq \{P(X) \in \mathbb{F}[X] \ : \ \deg(P) < k\}.$$

Notation 1.12. Soit $S \triangleq (\alpha_1, \cdots, \alpha_n)$ une liste de n éléments de $\mathbb{F}$. On définit la fonction d'évaluation :

$$\begin{array}{rcl} ev_S : \mathbb{F}[X] & \longrightarrow & \mathbb{F}^n \\ P & \longmapsto & (P(\alpha_1), \cdots, P(\alpha_n)) \,. \end{array}$$

Définition 1.35 (Code de Reed-Solomon)**.** Le code de *Reed-Solomon* défini sur $\mathbb{F}_q$ de longueur n, de dimension k et de support $S \triangleq (\alpha_1, \cdots, \alpha_n) \in \mathbb{F}_q^n$, avec $\forall i \neq j,\ \alpha_i \neq \alpha_j$, est donné par

$$RS_q[n,k] \triangleq \{ev_S(P) \ : \ \forall P \in \mathcal{P}_k(\mathbb{F}_q)\}.$$

Le principal inconvénient des codes de Reed-Solomon est la taille du corps sur lequel ce code est défini qui est toujours plus grande que la longueur du code. En effet, la définition des codes des Reed-Solomon impose que les éléments du support du code soient tous distincts, ce qui a pour conséquence que $n \leq q$.

Théorème 1.31 (MDS)**.** *Un code de Reed-Solomon $RS_q[n,k]$ a pour distance minimale $n-k+1$. C'est donc un code MDS.*

Proposition 1.32. *Soit $\mathcal{C}$ un $[n,k]_q$-code de Reed-Solomon de support $\mathcal{S} = \{\alpha_1, \cdots, \alpha_n\}$. Alors une matrice génératrice de $\mathcal{C}$ est*

$$G = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_{n-1} & \alpha_n \\ \vdots & \vdots & & \vdots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_{n-1}^{k-1} & \alpha_n^{k-1} \end{pmatrix}.$$

Proposition 1.33. *Un code BCH avec $\delta = n-k$ est un code de Reed-Solomon ayant pour support les puissances successives d'une racine primitive n^e de l'unité.*

Proposition 1.34. *Soit $RS_q[n,k]$ un code de Reed-Solomon, son rayon de recouvrement est $\rho = n-k$.*

Démonstration. Soient $RS_q[n,k]$ un code de Reed-Solomon de support S et $v \in \mathbb{F}_q^n$ un mot de l'espace ambiant. On peut toujours construire par interpolation de Lagrange $P(X) \in \mathcal{P}_k$ tel que

$$\forall i \in \{1, \ldots, k\}\,,\ P(\alpha_i) = v_i.$$

On en déduit que le rayon de recouvrement ρ de $RS_q[n,k]$ est majoré par $n-k$.

On montre maintenant que $\rho \geq n-k$, pour démontrer l'égalité. Soient $P'(X) \in \mathbb{F}_q[X]$ de degré k tel que toutes ses racines soient dans $\mathbb{F}_q$ et $v \triangleq ev_S(P')$. On obtient alors

$$\rho \geq n - \max_{P(X) \in \mathcal{P}_k} \left\{ |Zero(P'(X) - P(X))| \right\}.$$

Comme $P(X) \in \mathcal{P}_k$ alors $\deg(P' - P) = k$. Ainsi on obtient que $\max_{P(X)\in\mathcal{P}_k} \{|Zero(P'(X) - P(X))|\} = k$ et donc

$$\rho \geq n-k.$$

□

Définition 1.36 (Code de Reed-Solomon généralisé)**.** Soient $\mathcal{C}$ un code de Reed-Solomon $RS_q[n,k]$ et $\beta_1,\dots,\beta_n \in \mathbb{F}_q^*$. Le code de *Reed-Solomon généralisé* de *coefficients multiplicateurs* (β_i) est défini par

$$GRS_q[n,k] \triangleq \left\{(\beta_1 c_1,\dots,\beta_n c_n) \in \mathbb{F}_q^n \ : \ c \in RS_q[n,k]\right\}.$$

On le note également $GRS_q\left((\alpha_i)_i,(\beta_i)_i\right)$.

Remarque 1.13. Un code de Reed-Solomon est un code de Reed-Solomon généralisé avec $\beta_i = 1,\ \forall i \in \{1,\dots,n\}$.

Proposition 1.35. *Le code de Reed-Solomon généralisé $GRS_q[n,k]$ de support S et de coefficients multiplicateurs $(\beta_i)_{i=1,\dots,n}$ est un code équivalent au code de Reed-Solomon $RS_q[n,k]$ de support S.*

Proposition 1.36. *Soit $\mathcal{C}$ un $[n,k]_q$-code de Reed-Solomon généralisé de support $\mathcal{S} = \{\alpha_1,\dots,\alpha_n\}$ et de coefficients multiplicateurs $(\beta_1,\dots,\beta_n)$. Alors une matrice génératrice de $\mathcal{C}$ est*

$$G = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_{n-1} & \alpha_n \\ \vdots & \vdots & & \vdots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_{n-1}^{k-1} & \alpha_n^{k-1} \end{pmatrix} \begin{pmatrix} \beta_1 & & & & 0 \\ & \beta_2 & & & \\ & & \ddots & & \\ & & & \beta_{n-1} & \\ 0 & & & & \beta_n \end{pmatrix}.$$

Proposition 1.37. *Soit $GRS_q[n,k]$ un code de Reed-Solomon généralisé, son rayon de recouvrement est $\rho = n-k$.*

1.5 Codes de Goppa classiques

Il existe deux classes de codes de Goppa : les codes de Goppa géométriques définis sur des courbes algébriques et les codes de Goppa classiques définis sur les corps finis [Gop70]. Dans cette thèse, on ne s'intéresse qu'aux codes de Goppa classiques.

Définition 1.37 (Code de Goppa)**.** Soient $\mathcal{L} \triangleq (\alpha_1,\dots,\alpha_n) \in \mathbb{F}_{q^m}^n$ et $G(X) \in \mathbb{F}_{q^m}[X]$ de degré r tels que $g(\alpha_i) \neq 0$ pour tous les α_i. Le *code de Goppa* de longueur n défini sur $\mathbb{F}_q$ est donné par

$$\Gamma_q(\mathcal{L},G) \triangleq \left\{ c \in \mathbb{F}_q^n \ : \ \sum_{i=1}^{n} \frac{c_i}{X-\alpha_i} \equiv 0 \mod G(X) \right\}.$$

Comme aucun α_i n'est une racine du polynôme de Goppa G, $X-\alpha_i$ est bien inversible modulo $G(X)$, pour tout α_i.

Définition 1.38 (Subfield subcode)**.** Soit $\mathcal{C}$ un code défini sur $\mathbb{F}_{q^m}$. Le *subfield subcode* $\mathcal{C}'$ de $\mathcal{C}$ sur $\mathbb{F}_q$ est défini par

$$\mathcal{C}' \triangleq \mathcal{C} \cap \mathbb{F}_q^n.$$

Définition 1.39 (Code alternant)**.** Un *code alternant* est un *subfield subcode* d'un code de Reed-Solomon généralisé.

Proposition 1.38. *Un code de Goppa $\Gamma_q(\mathcal{L}, G)$ est un code alternant. Plus précisément*

$$\Gamma_q(\mathcal{L}, G) = GRS_{q^m}(\mathcal{L}, (\beta_i)_{i=1,\dots,n}) \cap \mathbb{F}_q^n,$$

où $\beta_i \triangleq \frac{G(\alpha_i)}{\prod_{j\neq i}^n (\alpha_i - \alpha_j)}$ et la dimension du GRS est $n - r$.

Démonstration. Soit $c \in \Gamma_q(\mathcal{L}, G)$. On montre que $c \in GRS_{q^m}(\mathcal{L}, (\beta_i)_{i=1,\dots,n})$.

$$\begin{aligned}
\sum_{i=1}^n \frac{c_i}{X - \alpha_i} &\equiv 0 \mod G(X), \\
\prod_{i=1}^n \frac{1}{(X - \alpha_i)} \left(\sum_{i=1}^n c_i \prod_{j\neq i} (X - \alpha_j) \right) &\equiv 0 \mod G(X).
\end{aligned}$$

Donc il existe un polynôme $P(X) \in \mathbb{F}_{q^m}[X]$ tel que $\deg(P) \leq n - r - 1$ et

$$\sum_{i=1}^n c_i \prod_{j\neq i} (X - \alpha_j) = P(X)G(X).$$

On en déduit que pour tout i on obtient :

$$\begin{aligned}
c_i \prod_{j\neq i} (\alpha_i - \alpha_j) &= P(\alpha_i)G(\alpha_i), \\
c_i &= \frac{P(\alpha_i)G(\alpha_i)}{\prod_{j\neq i}(\alpha_i - \alpha_j)}, \\
&= P(\alpha_i)\beta_i.
\end{aligned}$$

On en conclut que $c \in GRS_{q^m}(\mathcal{L}, (\beta_i)_{i=1,\dots,n})$.

Soit $c \in GRS_{q^m}(\mathcal{L}, (\beta_i)_{i=1,\dots,n}) \cap \mathbb{F}_q^n$. On montre que $c \in \Gamma_q(\mathcal{L}, G)$. Il existe un

polynôme $P(X) \in \mathbb{F}_{q^m}[X]$ tel que $\deg(P) \leq n - r - 1$ et

$$\begin{aligned}
c_i &= P(\alpha_i)\beta_i, \\
&= \frac{P(\alpha_i)G(\alpha_i)}{\prod_{j\neq i}(\alpha_i - \alpha_j)}, \\
\sum_{i=1}^{n} c_i &= \sum_{i=1}^{n} \frac{P(\alpha_i)G(\alpha_i)}{\prod_{j\neq i}(\alpha_i - \alpha_j)}, \\
\sum_{i=1}^{n} \frac{c_i}{X-\alpha_i} &= \left(\prod_{i=1}^{n} \frac{1}{(X-\alpha_i)}\right) \underbrace{\sum_{i=1}^{n} P(\alpha_i)G(\alpha_i) \prod_{j\neq i} \frac{(X-\alpha_j)}{(\alpha_i-\alpha_j)}}_{A(X)},
\end{aligned}$$

où $A(X)$ est le polynôme d'interpolation de Lagrange en les n points $(\alpha_i, P(\alpha_i)G(\alpha_i))$, de degré $n-1$. Par unicité du polynôme d'interpolation de Lagrange, on obtient que $A(X) = P(X)G(X)$. On en déduit

$$\begin{aligned}
\sum_{i=1}^{n} \frac{c_i}{X-\alpha_i} &= \left(\prod_{i=1}^{n} \frac{1}{(X-\alpha_i)}\right) P(X)G(X), \\
\sum_{i=1}^{n} \frac{c_i}{X-\alpha_i} &\equiv 0 \mod G(X).
\end{aligned}$$

Alors $c \in \Gamma_q(\mathcal{L}, G)$. □

Proposition 1.39. *Soit $\mathcal{C}$ un code de Goppa de longueur n. Alors la dimension de $\mathcal{C}$ est supérieure ou égale à $n - mr$ et sa distance minimale est supérieure ou égale à $r+1$.*

Proposition 1.40. *Soit un polynôme de Goppa $G(X) \in \mathbb{F}_{q^m}[X]$ ne possédant pas de racine multiple sur $\mathbb{F}_{q^m}$. Alors*

$$\Gamma_q(\mathcal{L}, G^{q-1}) = \Gamma_q(\mathcal{L}, G^q).$$

Démonstration. Preuve dans [SKHN76] □

Corollaire 1.41. *Soit $\Gamma_q(\mathcal{L}, G^{q-1})$ un code de Goppa tel que le polynôme de Goppa $G(X)$ ne possède pas de racine multiple sur $\mathbb{F}_{q^m}$. Alors sa distance minimale construite est $qr+1$ au lieu de $(q-1)r+1$.*

Le corollaire précédent est vraiment intéressant pour les codes de Goppa binaires, car on obtient presque un doublement de la distance minimale.

Exemple 1.14. Soit $\mathcal{C}$ le code de Goppa binaire construit à partir $\mathbb{F}_{2^8} = \mathbb{F}_2(\alpha)$, avec $\alpha^4 + \alpha^3 + \alpha^2 + 1 = \alpha^8$, $\mathcal{L} = \mathbb{F}_{2^8}$ et $G(X) = X^5 + \alpha^{28}X^4 + \alpha^{132}X^3 + \alpha^{165}X^2 + \alpha^{217}X + \alpha^4$. Le polynôme de Goppa G est irréductible et donc sans racine multiple sur $\mathbb{F}_{2^8}$. Le code $\mathcal{C}$ a pour paramètre $[256, 216, 11]_2$, ces paramètres atteignent exactement les bornes minimales.

La proposition 1.40 permet au code $\Gamma(\mathcal{L}, G^{q-1})$ de jouir de la plus grande distance minimale du code $\Gamma(\mathcal{L}, G^q)$ et à $\Gamma(\mathcal{L}, G^q)$ de profiter de la plus grande dimension de $\Gamma(\mathcal{L}, G^{q-1})$.

1.6 Codes systématiques

Les codes systématiques sont une généralisation des codes linéaires. En effet, on montrera que tous les codes linéaires sont des codes systématiques.

Définition 1.40 (Code systématique)**.** Soient $k \leq n \in \mathbb{N}$ et $\mathcal{C} \subset \mathbb{F}_q^n$ un ensemble a q^k éléments. S'il existe un ensemble $\mathcal{I} \subset \{1, \ldots, n\}$ tel que $|\mathcal{I}| = k$ et $\pi_{\mathcal{I}}(\mathcal{C}) = \mathbb{F}_q^k$, l'ensemble $\mathcal{C}$ est $(n, k)_q$-*code systématique.*

Remarque 1.15. Dans la définition précédente, l'ensemble $\mathcal{I}$ est un ensemble d'information de $\mathcal{C}$.

Notation 1.16. Soient $\mathcal{C}$ un $(n, k)_q$-code systématique et $\mathcal{I}$ un ensemble d'information de $\mathcal{C}$. On note

$$\forall c \in \mathcal{C},\ (\pi_{\mathcal{I}}(c), \pi_{\bar{\mathcal{I}}}(c)) \triangleq c,$$

où $\pi_{\mathcal{I}}(c)$ est la projection de c sur $\mathcal{I}$.

Définition 1.41 (Fonction d'encodage)**.** Soient $\mathcal{C}$ un $(n, k)_q$-code systématique et $\mathcal{I}$ un ensemble d'information de $\mathcal{C}$. On définit une *fonction d'encodage* de $\mathcal{C}$ associée à $\mathcal{I}$

$$\begin{array}{rcl} \forall c \in \mathcal{C},\ \phi_{\mathcal{I}} : \mathbb{F}_q^k & \longrightarrow & \mathbb{F}_q^n \\ \pi_{\mathcal{I}}(c) & \longmapsto & c \end{array}$$

avec $\phi_{\mathcal{I}}$ injectif.

Lemme 1.42. *Soit $\mathcal{C}$ un code systématique de fonction d'encodage $\phi_{\mathcal{I}}$ associée à un ensemble d'information $\mathcal{I}$. Si $\phi_{\mathcal{I}}$ est linéaire alors $\mathcal{C}$ l'est aussi.*

Définition 1.42 (Fonction génératrice)**.** Soient $\mathcal{C}$ un $(n, k)_q$-code systématique et $\phi_{\mathcal{I}}$ une fonction d'encodage associée à un ensemble d'information $\mathcal{I}$ de $\mathcal{C}$. On définit la *fonction génératrice* $\sigma_{\mathcal{I}}$ de $\mathcal{C}$ comme

$$\begin{array}{rcl} \sigma_{\mathcal{I}} : \mathbb{F}_q^k & \longrightarrow & \mathbb{F}_q^{n-k}, \\ u & \longmapsto & \pi_{\bar{\mathcal{I}}}(\phi_{\mathcal{I}}(u)). \end{array}$$

Pour tout $c \in \mathcal{C}$ il existe $u \in \mathbb{F}_q^k$ tel que $\phi(u) = c$, on obtient alors

$$\mathcal{C} = \left\{ \phi_{\mathcal{I}}(u) = (u, \sigma_{\mathcal{I}}(u)) \ : \ \forall u \in \mathbb{F}_q^k \right\}.$$

Plus simplement, on notera aussi ϕ et σ une fonction encodage et réciproquement une fonction génératrice d'un code systématique.

Proposition 1.43. *Soient $\mathcal{C}$ un $(n,k)_q$-code systématique et σ une fonction génératrice de $\mathcal{C}$. Alors $\mathcal{C}$ est un $[n,k]_q$-code linéaire si et seulement si σ est linéaire.*

D'après la proposition précédente, on en déduit que tout $[n,k]_q$-code linéaire est un $(n,k)_q$-code systématique. On peut donc voir les codes systématiques comme une généralisation des codes linéaires, de plus certains codes systématiques exhibent des meilleurs paramètres comparés a la famille des codes linéaires – par exemple les codes de Preparata [Pre68], les codes de Kerdock [Ker72] prouvés systématiques par Mykkeltlveit dans [Myk72].

La notion de syndrome pour les codes linéaires est fortement liée à la matrice de parité choisie. On propose de généraliser la notion de syndrome pour les codes systématiques, dépendant cette fois de la fonction génératrice.

Définition 1.43 (Fonction syndrome)**.** Soient $\mathcal{C}$ un $(n,k)_q$-code systématique et $\sigma_{\mathcal{I}}$ la fonction génératrice associée à l'ensemble d'information $\mathcal{I}$ de $\mathcal{C}$. Alors la *fonction syndrome* associée à $\sigma_{\mathcal{I}}$ est

$$\begin{array}{rcl} S_{\mathcal{I}} : \mathbb{F}_q^n & \longrightarrow & \mathbb{F}_q^{n-k} \\ u & \longmapsto & \pi_{\bar{\mathcal{I}}}(u) - \sigma_{\mathcal{I}}(\pi_{\mathcal{I}}(u)). \end{array}$$

La proposition suivante nous dit qu'un code systématique est l'ensemble des zéros de la fonction syndrome, comme pour le cas des codes linéaires.

Proposition 1.44. *Soient $\mathcal{C}$ un code systématique et $\mathcal{I}$ un ensemble d'information de $\mathcal{C}$. Alors $C = Zero(S_{\mathcal{I}})$.*

Démonstration. Soit $c \in Zero(S_{\mathcal{I}})$, alors $\pi_{\bar{\mathcal{I}}}(c) = \sigma_{\mathcal{I}}(\pi_{\mathcal{I}}(c))$. D'après la définition 1.42, on conclut que l'élément $c \in \mathcal{C}$.

Soit $c \in \mathcal{C}$. Alors il existe $u \in \mathbb{F}_q^k$ tels que $c = \phi_{\mathcal{I}}(u) = (\pi_{\mathcal{I}}(c), \sigma_{\mathcal{I}}(\pi_{\mathcal{I}}(c)))$. Par la complémentarité des projections on en déduit que $\pi_{\bar{\mathcal{I}}}(c) = \sigma_{\mathcal{I}}(\pi_{\mathcal{I}}(c))$. Par la définition de la fonction syndrome, on obtient que $c \in Zero(S_\sigma)$. □

Comme pour les codes linéaires, les syndromes induisent une partition naturelle de l'espace ambiant.

Proposition 1.45. *Soit $\mathcal{C}$ un $(n,k)_q$-code systématique. Alors la famille $\left\{(0,v) + \mathcal{C} \ : \ v \in \mathbb{F}_q^{n-k}\right\}$ forme une partition de $\mathbb{F}_q^n$.*

Démonstration. Soient $v_1, v_2 \in \mathbb{F}_q^{n-k}$ et $c_1, c_2 \in \mathcal{C}$ tels que $(0,v_1) + c_1 = (0,v_2) + c_2$ et $\mathcal{I}$ un ensemble d'information de $\mathcal{C}$. Calculons la projection sur $\mathcal{I}$ de c_1.

$$\begin{array}{rcl} \pi_{\mathcal{I}}(c_1) & = & \pi_{\mathcal{I}}((0,v_1) + c_1) \\ & = & \pi_{\mathcal{I}}((0,v_2) + c_2) \\ & = & \pi_{\mathcal{I}}(c_2). \end{array}$$

Comme la projection sur $\mathcal{I}$ de c_1 est égale à la projection sur $\mathcal{I}$ de c_2, alors $c_1 = c_2$. On en déduit que $v_1 = v_2$. □

Pour les codes non-linéaires, on peut généraliser la distribution des *coset leader* à la distribution des distances.

Définition 1.44 (Distribution des distances)**.** Soit $\mathcal{C}$ un code systématique. Alors la *distribution des distances* de $\mathcal{C}$ est la suite A_i définie par

$$A_i = \frac{1}{|\mathcal{C}|} |\left\{(c, c') \in \mathcal{C}^2 : d(c, c') = i\right\}|.$$

Remarque 1.17. Si $\mathcal{C}$ est linéaire, alors la distribution des distances coïncide avec la définition de la distribution des poids. De plus, grâce au théorème de Delsarte on peut appliquer le théorème de McWilliams pour calculer la distribution des distances du code dual. Bien évidement le code dual n'existe pas, mais la distance duale $d^{\perp}$ est bien définie. On verra dans la sous-section 4.4.4, que la distance duale d'un code non-linéaire peut nous donner des informations combinatoires sur le code.

Proposition 1.46. *Soient $\mathcal{C}$ un $(n, k)_q$-code systématique et $s \in \mathbb{F}_q^{n-k}$. Alors la distribution des distances de $(\mathbf{0}, s) + \mathcal{C}$ est la même que celle du code $\mathcal{C}$.*

Démonstration. Soient $\mathcal{C}' = (\mathbf{0}, s) + \mathcal{C}$ et A_i la distribution des distances de $\mathcal{C}$. La distribution des distances de $\mathcal{C}'$, notée A'_i, est définie par

$$A'_i = \frac{1}{|\mathcal{C}'|} |\left\{(c'_1, c'_2) \in (\mathcal{C}')^2 : d(c'_1, c'_2) = i\right\}|.$$

Or $|\mathcal{C}'| = |\mathcal{C}|$ et pour tout $c'_1, c'_2 \in \mathcal{C}'$, il existe $c_1, c_2 \in \mathcal{C}$ tels que $c'_1 = c_1 + (\mathbf{0}, s)$ et $c'_2 = c_2 + (\mathbf{0}, s)$. On obtient alors

$$\begin{aligned} A'_i &= \frac{1}{|\mathcal{C}|} |\left\{(c_1, c_2) \in \mathcal{C}^2 : d(c_1 + (\mathbf{0}, s), c_2 + (\mathbf{0}, s)) = i\right\}| \\ &= A_i. \end{aligned}$$

□

Exemple 1.18. Dans la table 1.2, on liste tous les mots du code de Nadler [Nad62], qui est un $(12, 5)_2$-code systématique. Ce code est utilisé dans le système de radio SINCGARS [Ham96]. Van Lint a montré dans son livre [VL98] que le code de Nadler possède deux fois plus de mots de code comparé aux codes linéaires de paramètres équivalents, il a également montré la structure combinatoire de ce code [VL72].

On remarque qu'un ensemble d'information pour le code de Nadler est $\mathcal{I} \triangleq \{1, 2, 4, 7, 10\}$. En plus de l'énumération exhaustive donnée dans la table 1.2, on

011	100	100	100
101	010	010	010
110	001	001	001
100	011	100	100
010	101	010	010
001	110	001	001
100	100	011	100
010	010	101	010
001	001	110	001
100	100	100	011
010	010	010	101
001	001	001	110

111	010	100	001
111	001	010	100
111	100	001	010
010	111	001	100
001	111	100	010
100	111	010	001
100	001	111	010
010	100	111	001
001	010	111	100
001	100	010	111
100	010	001	111
010	001	100	111

011	011	011	011
101	101	101	101
110	110	110	110
000	111	111	111
111	000	111	111
111	111	000	111
111	111	111	000
000	000	000	000

TABLE 1.2 – Chaque ligne représente un mot du code de Nadler qui est un $(12,5)_2$-code systématique.

peut décrire les codes de Nadler par sa fonction σ et par l'ensemble d'information $\mathcal{I}$ – plus exactement par son complémentaire $\bar{\mathcal{I}}$. Soit $u \in \mathbb{F}_q^k$. Alors,

$$\begin{aligned}
\pi_{\{3\}}(\sigma(u)) &= u_1 + u_2 + u_3 + u_4 + u_5 + u_3u_4 + u_4u_5 + u_5u_3 \\
\pi_{\{5\}}(\sigma(u)) &= u_1 + u_2 + u_3 + (u_1 + u_5)(u_3 + u_4) \\
\pi_{\{6\}}(\sigma(u)) &= u_2 + u_4 + u_5 + u_1u_3 + u_3u_5 + u_5u_1 \\
\pi_{\{8\}}(\sigma(u)) &= u_1 + u_2 + u_4 + (u_1 + u_3)(u_4 + u_5) \\
\pi_{\{9\}}(\sigma(u)) &= u_2 + u_3 + u_5 + u_1u_3 + u_3u_4 + u_4u_1 \\
\pi_{\{11\}}(\sigma(u)) &= u_1 + u_2 + u_5 + (u_1 + u_4)(u_3 + u_5) \\
\pi_{\{12\}}(\sigma(u)) &= u_2 + u_3 + u_4 + u_1u_4 + u_4u_5 + u_5u_1.
\end{aligned}$$

1.7 Problème du décodage

Dans cette thèse, on a beaucoup étudié les algorithmes de décodage des codes linéaires. On propose tout d'abord de formaliser les quelques problèmes liés au décodage. Berlekamp, McEliece et Van Tilborg ont montré en 1978 que le problème de décodage d'un code aléatoire est NP-difficile [BMVT78]. Toutes les méthodes efficaces existantes sont destinées à des familles de codes précises et exploitent leur structure.

Problème 1.47 (Décodage borné)**.** *Soient $\mathcal{C}$ un code et pour tout v un mot de l'espace ambiant de $\mathcal{C}$. Le problème du* décodage borné *jusqu'à T consiste à trouver un mot de code $c \in \mathcal{C}$, s'il existe, tel que*

$$d(v, c) \leq T.$$

On présente maintenant le problème du décodage complet. C'est le problème de décodage le plus difficile. Le décodage des codes de Reed-Solomon a été largement étudié [WB86, Sud97, GS99, Wu08] et le problème du décodage complet pour ces codes a récemment été montré NP-difficile [GV05].

Problème 1.48 (Décodage complet)**.** *Soit $\mathcal{C}$ un code. Le problème du* décodage complet *consiste pour tout v un élément de l'espace ambiant de $\mathcal{C}$, à trouver $c \in \mathcal{C}$ un mot de code tel que*

$$d(v, c) = d(v, \mathcal{C}).$$

Bien entendu, ces deux problèmes de décodage sont liés. La proposition suivante en est une bonne illustration.

Proposition 1.49. *Soit $\mathcal{C}$ un code de rayon de recouvrement ρ. La résolution du problème du décodage complet implique la résolution du problème du décodage borné jusqu'à ρ.*

Problème 1.50 (Décodage par syndrome)**.** *Soient $\mathcal{C}$ un $[n, k]_q$-code linéaire, H une matrice de parité de $\mathcal{C}$ et $v \in \mathbb{F}_q^n$ un mot de l'espace ambiant. Le problème du* décodage par syndrome *consiste à retrouver le* coset-leader *e tel que $S_H(e) = S_H(v)$.*

Proposition 1.51. *Le problème du décodage complet est un problème équivalent au problème du décodage par syndrome.*

Démonstration. Soient $\mathcal{C}$ un $[n, k]_q$-code linéaire, $v \in \mathbb{F}_q^n$ un mot de l'espace ambiant et $c \in \mathcal{C}$ le mot de code le plus proche de v, alors le *coset-leader* du syndrome de v est $v - c$.

Soit v un élément de l'espace ambiant et e le *coset-leader* du syndrome de v, alors le mot de code le plus proche est $c = v + e$. □

Enfin, on distingue le problème du décodage unique et le problème du décodage en liste.

Problème 1.52 (Décodage unique)**.** *Soit $\mathcal{C}$ un code de capacité de correction t. Le* problème du décodage unique *est le problème du décodage borné jusqu'à t.*

Le décodage en liste est le problème de décodage qu'on a le plus approfondi, en voici une définition.

Problème 1.53 (Décodage en liste)**.** *Soit $\mathcal{C}$ un code. Le* problème du décodage en liste *jusqu'à T consiste pour tout mot de l'espace ambiant v, à trouver tous les mots de code $c_i \in \mathcal{C}$, s'il en existe, tels que $d(v, c_i) \leq T$.*

Chapitre 2

Algorithmes de décodage

Plan du chapitre

2.1 Introduction

La théorie des codes est une discipline récente. Il existe différents problèmes en théorie des codes qui ont été prouvés difficiles, dont le problème de décodage d'un code aléatoire [BMVT78]. Tout ce chapitre est dédié aux problématiques de décodages de certains codes structurés. On commence par présenter certains algorithmes de décodage unique. La section 2.3 est consacrée aux méthodes de décodage en liste les plus connues. On commence par présenter le rôle important des bornes de Johnson dans le contexte du

décodage en liste. Ensuite, on présente, suivant leur chronologie, différents algorithmes de décodage en liste.

2.2 Décodage unique

Cette sous-section est dédiée à la résolution du problème 1.52, page 28, pour les codes de Reed-Solomon et alternant. On présente deux méthodes différentes, la méthode d'Euclide résout l'équation clé – par exemple celle donnée pour les codes BCH, page 17 – pour trouver ainsi le polynôme localisateur et le polynôme évaluateur. En revanche, la méthode de Welch-Berlekamp ne résout pas l'équation clé, on la présente car les algorithmes de décodage en liste sont fortement basés sur cette méthode. Bien que ce soit un algorithme de décodage unique, il en n'est pas moins un élément essentiel dans le paysage du décodage en liste.

2.2.1 Algorithme d'Euclide

Algorithme d'Euclide Étendue

L'algorithme d'Euclide permet de calculer le pgcd de deux éléments a et b, sans pour autant en connaître leur factorisation [Euc]. Cet algorithme très connu est constitué par une multitude de divisions euclidiennes. Pour autant il ne permet pas de calculer les coefficients de Bézout, c'est-à-dire calculer u et v tel que

$$au + bv = pgcd(a, b).$$

Pour calculer les coefficients de Bézout à partir de l'algorithme d'Euclide, il suffit d'insérer tous les quotients et les restes constitués par la succession des divisions euclidiennes dans le sens inverse de l'exécution. On commence par quelques rappels sur la division euclidienne.

Définition 2.1 (Division euclidienne)**.** Soient $a, b \in \mathcal{A} \setminus \{0\}$ deux éléments d'un anneau euclidien de stathme v. Alors il existe un unique couple $(q, r) \in \mathcal{A}$ tel que

$$\begin{cases} a = bq + r \\ v(r) < v(b) \end{cases}$$

L'algorithme d'Euclide consiste à faire une nouvelle division euclidienne de b par r tant que le reste r n'est pas nul. Ainsi on obtient deux suites de $\mathcal{A}$ que l'on note (b_n) pour la suite de quotients et (r_n) la suite des restes. Par ce procédé, il est clair que $v(r_n)$ est une suite décroissante, comme un stathme est à valeur dans $\mathbb{N}$ alors il existe $N \in \mathbb{N}$ tel que $\forall n > N$, $v(r_n) = v(r_N) = 0$. Le plus petit N vérifiant cette propriété est le nombre de divisions euclidiennes effectuées. On obtient alors cette suite de divisions

euclidiennes :

$$\begin{cases} a &= bq_1 + r_1 \\ b &= r_1 q_2 + r_2 \\ r_1 &= r_2 q_3 + r_3 \\ &\vdots \\ r_{N-4} &= r_{N-3} q_{N-2} + r_{N-2} \\ r_{N-3} &= r_{N-2} q_{N-1} + r_{N-1} \\ r_{N-2} &= r_{N-1} q_N \end{cases}$$

En insérant la dernière égalité $r_{N-2} = r_{N-1} q_N$ dans celle qui la précède on obtient

$$\begin{aligned} r_{N-3} &= r_{N-1} q_N q_{N-1} + r_{N-1} \\ &= r_{N-1}(q_N q_{N-1} + 1). \end{aligned}$$

Et ainsi de suite jusqu'à remonter à la première division euclidienne qui divise a par b, on peut alors trouver les coefficients de Bézout. Avec les notations précédentes, le pgcd de a et de b est exactement r_{N-1}. Par ailleurs, le résultat retourné est unique à multiplication par un inversible près, voir l'algorithme 1.

Algorithme 1: Euclide étendu

Entrées : Deux éléments $a, b \in \mathcal{A}$.
Sorties : Le triplé $(r, u, v) \in \mathcal{A}^3$ tel que $pgcd(a, b) = r = au + bv$.

début
- $r_{-1} \longleftarrow a,\ r_0 \longleftarrow b;$
- $u_{-1} \longleftarrow 1, v_{-1} \longleftarrow 0;$
- $u_0 \longleftarrow 0,\ v_0 \longleftarrow 1;$
- $i \longleftarrow 1\,;$
- **tant que** $r_i \neq 0$ **faire**
  - $q \longleftarrow r_{i-1}/r_i\,;$
  - $r_{i+1} \longleftarrow r_{i-1} - qr_i\,;$
  - $u_{i+1} \longleftarrow u_{i-1} - qu_i\,;$
  - $v_{i+1} \longleftarrow v_{i-1} - qv_i\,;$
  - $i++\,;$
- **retourner** $(r_{i-1}, u_{i-1}, v_{i-1})$.

Application aux décodage

Grâce à l'algorithme d'Euclide étendu, on peut résoudre l'équation clé donnée par les codes BCH au sens strict, équation (1.1) à la page 19, c'est-à-dire calculer $L(X)$ et $\Lambda(X)$ tels que :

$$L(X) \equiv \Lambda(X) S_\delta(X) \mod X^\delta \tag{2.1}$$

Plus précisément, il existe un polynôme $u(X) \in \mathbb{F}_q[X]$ tel que

$$\Lambda(X) S_\delta(X) + u(X) X^\delta = L(X) \tag{2.2}$$

De plus, la fonction

$$\begin{array}{rcl} \deg : \mathbb{F}_q[X] & \longrightarrow & \mathbb{N} \cup \{-\infty\} \\ P(X) & \longmapsto & \deg(P) \end{array}$$

est un stathme euclidien. L'anneau des polynômes univariés à coefficients dans un corps fini est donc euclidien. On peut utiliser l'algorithme d'Euclide étendu pour essayer de retrouver le polynôme localisateur $\Lambda(X)$ et évaluateur $L(X)$. Pour cela, il suffit de mettre en entrée de cet algorithme le polynôme syndrome X^δ et $S_\delta(X)$ que l'on peut calculer à partir du mot reçu. Comme remarqué précédemment, la suite (r_n) est décroissante, alors on arrête les divisions euclidiennes à l'étape i, lorsque $\deg(r_{i-1}) \geq t$ et $\deg(r_i) < t$. En effet, la suite (r_n) joue le rôle du polynôme évaluateur $L(X)$, tel que $\deg(L) < t$. On a donc un triplé candidat (Λ, u, L) qui satisfait l'égalité (2.2). Comme énoncé dans la partie précédente, le résultat de cet algorithme est unique à multiplication par un inversible, dans notre cas, par un scalaire $\alpha \in \mathbb{F}_q$. En utilisant le fait que le terme constant de $\Lambda(x)$, le polynôme localisateur est 1. Il est donc facile de retrouver le bon triplé solution de l'égalité (2.2). On obtient alors l'algorithme 2.

Algorithme 2: Euclide étendu pour l'équation clé

Entrées : Deux éléments $S_\delta(X),\ X^\delta \in \mathbb{F}_q[X]$.
Sorties : Le couple $(\Lambda(X), L(X))$.

début
- $r_{-1} \longleftarrow S_\delta(X),\ r_0 \longleftarrow X^\delta$;
- $u_{-1} \longleftarrow 1, v_{-1} \longleftarrow 0$;
- $u_0 \longleftarrow 0,\ v_0 \longleftarrow 1$;
- $i \longleftarrow 1$;
- **tant que** $\deg(r_i) \geq (\delta/2)$ **faire**
  - $q \longleftarrow r_{i-1}/r_i$;
  - $r_{i+1} \longleftarrow r_{i-1} - qr_i$;
  - $u_{i+1} \longleftarrow u_{i-1} - qu_i$;
  - $v_{i+1} \longleftarrow v_{i-1} - qv_i$;
  - $i++$;
- $\Lambda(X) \longleftarrow u_i(X)\ \text{Coefficient}(u_i, 0)^{-1}$;
- $L(X) \longleftarrow r_i(x)\ \text{Coefficient}(u_i, 0)^{-1}$;
- **retourner** $(\Lambda(X), L(X))$.

2.2.2 Algorithme de Welch-Berlekamp

L'algorithme de Welch-Berlekamp est un algorithme dédié aux codes de Reed-Solomon [WB86]. Bien qu'il soit un algorithme de décodage unique, il est l'essence des algorithmes de décodage en liste. En effet, la majorité des algorithmes de décodage en liste reprennent le principe de la méthode proposée par Welch et Berlekamp.

Contexte de décodage

Comme c'est un algorithme de décodage unique, on suppose que le poids de l'erreur ajouté par le canal est inférieur à t, la capacité de correction du code de Reed-Solomon. En gardant les mêmes notations que dans la définition des codes de Reed-Solomon, définition 1.35 page 20, on a que pour tout mot d'un code de Reed-Solomon sur $\mathbb{F}_q$ de dimension k et de longueur n, il existe $P(X) \in \mathbb{F}_q[X]$ de degré strictement inférieur à k tel que $c = (P(\alpha_1), \ldots, P(\alpha_n))$. Alors

$$\begin{aligned} y &\triangleq c + e \\ &= (e_1 + P(\alpha_1), \ldots, e_n + P(\alpha_n)). \end{aligned}$$

Décoder le mot reçu y, c'est-à-dire, retrouver c revient donc à calculer le polynôme $P(X) \in \mathbb{F}_q[X]$ de degré strictement inférieur à k.

Décodage

Comme le poids de l'erreur $w(e)$ est inférieur ou égal à t, il y a au moins $n - t$ composantes de l'erreur qui sont donc nulles. Alors pour ces composantes on a $y_i = P(\alpha_i)$. L'idée qui ne paraît pas forcément naturelle est de calculer un polynôme bivarié $Q(X, Y) \in \mathbb{F}_q[X, Y]$ tel que

$$(\mathbf{IP}_{WB}) \triangleq \begin{cases} 0 \neq Q(X,Y) \triangleq Q_0(X) + YQ_1(X), \\ Q(\alpha_i, y_i) = 0, \ \forall i \in \{1, \ldots, n\}, \\ \deg(Q_0) \leq n - t - 1, \\ \deg(Q_1) \leq n - t - k. \end{cases}$$

Naturellement, on se demande comment un tel polynôme $Q(X, Y)$ nous permet de calculer le polynôme $P(X)$ correspondant au mot reçu y. La réponse est donnée par la proposition suivante.

Proposition 2.1. *Soit $Q(X, Y) \in \mathbb{F}_q[X, Y]$ satisfaisant les conditions de $(\mathbf{IP}_{WB})$, $P(X) \in \mathbb{F}_q[X]$ le polynôme associé au mot de code c et $w(e) \leq t$. Alors $Q(X, P(X))$ est le polynôme nul.*

Démonstration. Pour montrer cette proposition, on calcule tout d'abord le degré d'un tel polynôme.

$$\begin{aligned} \deg(Q(X, P(X))) &= \deg(Q_0(X) + P(X)Q_1(X)) \\ &= \max\{n - t - 1, n - k - t + k - 1\} \\ &= n - t - 1. \end{aligned}$$

Or, comme l'on suppose qu'il y a au plus t erreurs, alors $Q(X, P(X))$ a au moins $n - t$ racines. On est donc en présence d'un polynôme univarié qui a plus de racines que son degré, ce polynôme est forcément le polynôme nul. □

Cette proposition est très intéressante, car à partir du polynôme $Q(X,Y)$ on peut directement calculer le polynôme qui correspond au mot de code, c'est-à-dire $P(X)$. En effet :

$$\begin{aligned} 0 &= Q(X,P(X)) \\ &= Q_0(X) + P(X)Q_1(X) \\ P(X) &= -\frac{Q_0(X)}{Q_1(X)}. \end{aligned}$$

Une question toute aussi importante que la précédente est de savoir si un tel polynôme $Q(X,Y)$ existe toujours pour n'importe quel code de Reed-Solomon et pour n'importe quel mot reçu. La proposition suivante nous indique qu'un tel polynôme bivarié existe toujours.

Proposition 2.2. *Le polynôme bivarié $Q(X,Y) \in \mathbb{F}_q[X,Y]$ vérifiant $(\mathbf{IP}_{WB})$ existe toujours.*

Démonstration. On note N_{cst} le nombre de contraintes que $(\mathbf{IP}_{WB})$ engendre et on calcule le nombre N_{var} de coefficients de $Q(X,Y)$, c'est-à-dire le nombre de variables d'un tel système linéaire. D'une part

$$N_{cst} = n.$$

Et d'autre part

$$\begin{aligned} N_{var} &= \deg(Q_0) + 1 + \deg(Q_1) + 1 \\ &= n - t + n - t - k + 1 \\ &> n + (n-d) - (k-1) \\ &= n. \end{aligned}$$

On remarque donc que le système engendré par $(\mathbf{IP}_{WB})$ possède strictement plus de variables que de contraintes. □

Pour résumer, on est assuré de l'existence d'un tel polynôme $Q(X,Y)$ et on peut calculer le polynôme $P(X)$, car $Q(X,P(X))$ est le polynôme nul. On remarque que plus généralement, $Y - P(X)$ divise $Q(X,Y)$ et $P(X)$ est appelé facteur Y-linéaire.

Dans l'algorithme 3, la fonction d'interpolation peut être par exemple réalisée avec l'algorithme de Koetter [KVA03, Koe96], ou celui d'Alekhnovich [Ale05].

2.3 Décodage en liste

2.3.1 Borne de Johnson

Dans cette section, on présente différents algorithmes de décodage en liste. Grâce à de telles méthodes, on est capable de corriger plus d'erreurs que la capacité de correction t.

Algorithme 3: Welch-Berlekamp

Entrées : Le mot reçu $y \in \mathbb{F}_q^n$ et le code $\mathcal{C}$ de Reed-Solomon.
Sorties : Le mot de code $c \in \mathcal{C}$ s'il existe tel que $d(c, y) \leq t$, sous sa forme polynomiale.

début
$Q(X, Y) \longleftarrow$ Interpolation($\mathbf{IP}_{WB}, \mathcal{C}$),
retourner $-\frac{Q_0(X)}{Q_1(X)}$.

En contrepartie, ces algorithmes ne retournent pas un unique mot du code mais une liste de mots du code. Dans l'hypothèse que l'on soit capable de corriger autant d'erreurs que l'on souhaite, la taille de la liste des mots de code deviendrait potentiellement très grande. Si pour un $[n, k]_q$-code, on était en mesure de corriger n erreurs, alors la liste consisterait en tous les mots du code, la taille de la liste serait donc q^k. Évidement, la complexité d'une telle méthode serait exponentielle et la liste de tous les mots du code n'apporterait pas du tout d'information sur le mot de code envoyé. C'est dans ce sens que la borne de Johnson est très importante dans le contexte du décodage en liste. En effet, la borne de Johnson nous fournit une borne maximale sur le nombre d'erreurs que l'on peut corriger tout en s'assurant que la taille de la liste reste polynomiale. Pour cela, on a besoin du théorème suivant.

Théorème 2.3. *Soient $\mathcal{C}$ un $[n, k, d]_q$-code linéaire, $\delta, \gamma \in]0, 1[$ et $r \in \mathbb{F}_q^n$ tels que $d \triangleq ((q-1)/q)(1-\delta)n$ et $T \triangleq ((q-1)/q)(1-\gamma)n$. Si $\gamma^2 > \delta$ alors*

$$|\mathcal{B}_q(r, T) \cap \mathcal{C}| \leq \min\left\{n(q-1), \frac{1-\delta}{\gamma^2 - \delta}\right\}.$$

De plus, pour le cas où $\gamma^2 = \delta$, on a $|\mathcal{B}_q(r, T) \cap \mathcal{C}| \leq 2n(q-1) - 1$.

Démonstration. Une démonstration géométrique est dans la thèse de Guruswami [Gur05, Thm 3.1, p.35], ainsi qu'une démonstration probabiliste dans [Gur06, Thm. 3.1, p. 20]. □

Théorème 2.4 (Borne de Johnson)**.** *Soient $\mathcal{C}$ un $[n, k, d]_q$-code linéaire et $T \in \mathbb{N}$.*

- *Si*
$$T < J_q(n, d) \triangleq n\frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\frac{d}{n}}\right),$$
alors
$$|\mathcal{B}_q(r, T) \cap \mathcal{C}| \leq \min\left\{n(q-1), \frac{nd}{nd - 2T\left(n - \frac{qT}{2(q-1)}\right)}\right\}.$$
- *Si $T \triangleq J_q(n, d)$, alors $|\mathcal{B}_q(r, T) \cap \mathcal{C}| \leq 2n(q-1) - 1$.*

Démonstration. Avec les notations précédentes, on obtient γ et δ tels que

$$T = n\frac{q-1}{q}(1-\gamma) \text{ et } d = n\frac{q-1}{q}(1-\delta).$$

Si on suppose que $T < J_q(n,d)$ alors $\gamma > \sqrt{1-(q/(q-1)d/n)}$ et $\gamma^2 > \delta$. L'hypothèse du théorème 2.3 est alors vérifiée. De plus

$$\begin{aligned}\frac{1-\delta}{\gamma^2-\delta} &= \frac{nd}{n^2\frac{q-1}{q}\left(\frac{d}{n}\frac{q}{q-1} - 2\frac{T}{n}\frac{q}{q-1} + \frac{q^2}{(q-1)^2}\frac{T^2}{n^2}\right)} \\ &= \frac{nd}{nd - 2Tn + T^2\frac{q}{q-1}}.\end{aligned}$$

On obtient le résultat en appliquant le théorème 2.3. □

La quantité $J_q(n,d)$ est appelé *borne de Johnson.* On obtient pour les codes de Reed-Solomon $|\mathcal{B}_q(r,T) \cap \mathcal{C}| < n^2$. Ce qui améliore le résultat dérivé d'Agrell, Vardy et Zeger [AVZ00], qui ont montré pour $T < J_2(n,d)$, le nombre de mots de code dans n'importe quelle boule de rayon T est majoré en $\mathcal{O}(n^2)$.

Le résultat précédent définit une famille de bornes dépendant de q, n et d. À cela on propose de poser

$$\begin{aligned}J_\infty(n,d) &\triangleq \lim_{q\to+\infty} J_q(n,d) \\ &= \lim_{q\to+\infty} n\frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}\frac{d}{n}}\right) \\ &= n\left(1-\sqrt{1-\frac{d}{n}}\right).\end{aligned}$$

On l'appelle la borne de Johnson générique. Dans la sous-section 2.3.3, on verra que l'algorithme de Guruswami-Sudan atteint cette borne. Puisque cette quantité définit plusieurs bornes, on les compare dans la figure 2.1. Pour diverses valeurs de q on fait varier la distance minimale normalisée, c'est-à-dire d/n.

On remarque que si la distance minimale normalisée est inférieure à $1/2$, alors la borne de Johnson binaire est supérieure à toutes les autres bornes de Johnson. Les codes ayant une distance normalisée supérieure à $1/2$, impose donc un faible rendement k/n – borne de Plotkin. C'est pourquoi on regarde tout particulièrement l'intervalle défini entre 0 et $1/2$. Dans cet intervalle, on compare la borne de Johnson binaire, générique et $d/(2n)$ le rayon normalisé des algorithmes de décodage unique, figure 2.2. Cela nous permet d'avoir une représentation graphique du gain qu'apporte le décodage en liste par rapport au décodage unique.

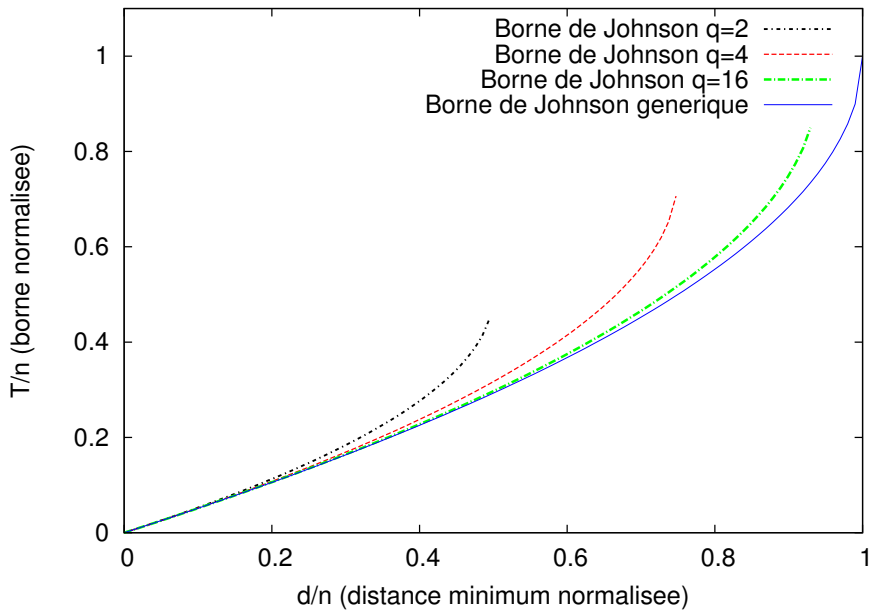

FIGURE 2.1 – Comparaison de la borne de Johnson pour q valant 2, 4, 16 et $+\infty$.

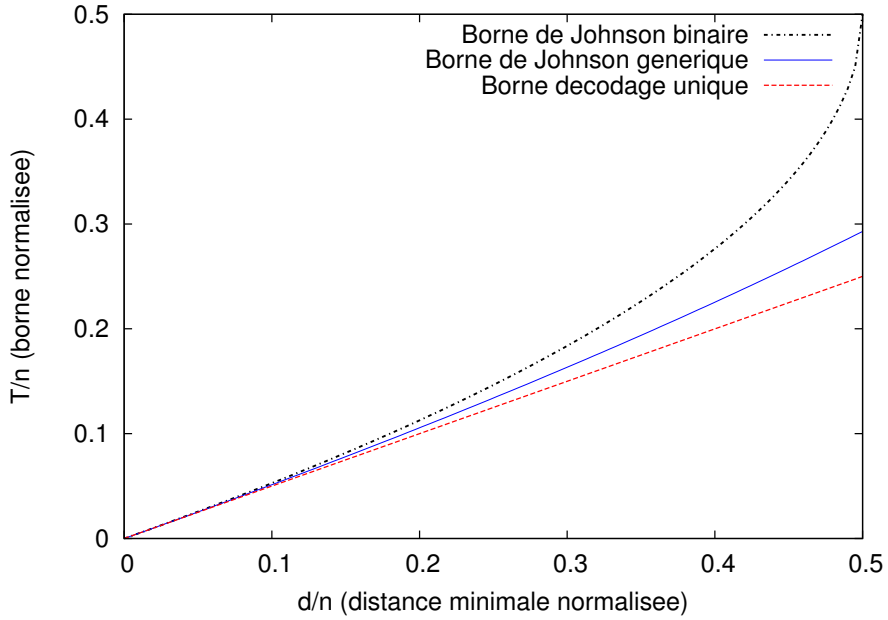

FIGURE 2.2 – Comparaison de la borne de Johnson binaire, générique et la borne du décodage unique.

La borne de Johnson binaire est plus grande, comparée à la borne de Johnson générique ainsi qu'à la borne de décodage unique. Cependant, cette borne ne fournit pas de moyen algorithmique pour atteindre cette borne. Le reste de cette section est dédiée à l'aspect algorithmique pour atteindre ces différentes bornes. La méthode proposée par Sudan a ouvert les portes à de telles méthodes. Ainsi, avec Guruswami, ils ont amélioré l'algorithme de Sudan pour les codes de Reed-Solomon pour atteindre la borne de Johnson générique. En 2008, Wu a proposé un algorithme de décodage en liste pour les BCH binaires qui atteint la borne de Johnson binaire. Ensuite, avec Daniel Augot et Alain Couvreur, nous avons proposé un algorithme générique pour tous les codes alternants sur $\mathbb{F}_q$, de façon à atteindre la borne de Johnson q-aire. Ce qui permet par exemple, de décoder en liste les codes de Goppa classiques binaires jusqu'à la borne de Johnson binaire.

2.3.2 Algorithme de Sudan

Madhu Sudan a proposé à la fin de l'année 1996, une extension à l'algorithme de Welch-Berlekamp [Sud97]. C'est alors le premier algorithme de décodage en liste pour les codes de Reed-Solomon. C'est une avancée capitale, car avec cette méthode, et pour certains codes de Reed-Solomon, on est capable de décoder au-delà de la capacité de correction du décodage unique.

Remarque

Voici la remarque importante de Sudan qui mène à son algorithme en partant de celui de Welch-Berlekamp. Si on souhaite décoder plus loin que la capacité de correction, alors dans certains cas – aussi rares soient-ils – l'algorithme doit pouvoir retourner une liste de mots de code et non plus un unique mot de code.

Soient $\mathcal{C}$ un $RS_q[n,k]$ code de Reed-Solomon et $P_1(X), P_2(X) \in \mathbb{F}_q[x]$ deux polynômes différents de degré strictement inférieur à k tels que l'algorithme de décodage doit retourner P_1 et P_2. Alors $Y - P_1(X)$ divise $Q(X,Y)$ – le polynôme d'interpolation de l'algorithme de Welch-Berlekamp, et de même $Y - P_2(X) \mid Q(X,Y)$. Comme on a supposé que P_1 est différent à P_2 alors

$$\boxed{\deg_Y(Q) > 1.}$$

Algorithme

L'idée de base de Madhu Sudan dans [Sud97] est d'augmenter le degré en Y du polynôme d'interpolation de l'algorithme de Welch-Berlekamp, que l'on note ℓ, pour décoder jusqu'à T erreurs. Le problème d'interpolation devient

$$(\mathbf{IP}_S) \triangleq \begin{cases} 0 \neq Q(X,Y) \triangleq \sum_{i=0}^{\ell} Q_i(X)Y^i, \\ Q(\alpha_i, y_i) = 0, \ \forall i \in \{1,\ldots,n\}, \\ \deg(Q_j) \leq n - T - 1 - j(k-1), \ \forall j \in \{0,\ldots,\ell\}. \end{cases}$$

On peut présenter les mêmes propositions que pour l'algorithme de Welch-Berlekamp, c'est-à-dire l'existence d'un tel polynôme $Q(X,Y)$ et la nullité du polynôme $Q(X,P(X))$, c'est ce que l'on se propose de démontrer dans le reste de cette partie. Tout d'abord, on a besoin du lemme suivant qui donne une minoration sur le nombre de coefficients du polynôme $Q(X,Y)$.

Lemme 2.5. *Soient a et b deux entiers strictement positifs et $Q(X,Y) \in \mathbb{F}_q[X,Y]$ tels que*

$$\begin{cases} Q(X,Y) & \triangleq & \sum_{j=0}^{\ell} Q_j(X)Y^j, \\ \deg(Q_j) & \leq & a - jb, \ \forall j \in \{0,\dots,\ell\}. \end{cases}$$

Alors le polynôme $Q(X,Y)$ a au moins $\frac{(a+1)^2-1}{2b}$ coefficients.

Démonstration. Avec les notations précédentes, on a que $a - \ell b > 0$ et $a - (\ell+1)b \leq 0$. Le nombre de coefficients est

$$\begin{aligned} \sum_{j=0}^{\ell} a + 1 - jb &= (\ell+1)\left(a + 1 - b\frac{\ell}{2}\right) \\ &> (\ell+1)\left(\frac{a}{2}+1\right) \\ &\geq \frac{(a+1)^2-1}{2b}. \end{aligned}$$

□

Proposition 2.6. *Si l'égalité suivante est satisfaite*

$$T < n\left(1 - \sqrt{2 - 2\frac{d}{n} + \frac{1}{n^2}}\right)$$

alors pour les codes de Reed-Solomon, il existe toujours un polynôme bivarié $Q(X,Y)$ satisfaisant $(\mathbf{IP}_S)$ pour tout mot reçu $y \in \mathbb{F}_q^n$.

Démonstration. Si le nombre de relations linéaires que $(\mathbf{IP}_S)$ définit est strictement plus grand que le nombre d'indéterminés, alors pour les codes de Reed-Solomon et pour tout mot reçu, il existe un polynôme $Q(X,Y)$ satisfaisant $(\mathbf{IP}_S)$. Grâce au lemme 2.5, on obtient

$$\begin{aligned} \frac{(n-T)^2-1}{2(k-1)} &> n \\ (n-T)^2 &> 2n(k-1)+1. \end{aligned}$$

Or pour les codes de Reed-Solomon, qui sont MDS, on a $n-k+1=d$, c'est-à-dire $n-d=k-1$. Ainsi

$$\begin{aligned} n-T &> \sqrt{2n(n-d)+1} \\ T &< n\left(1 - \sqrt{2 - 2\frac{d}{n} + \frac{1}{n^2}}\right), \end{aligned}$$

Alors

$$\frac{(n-T)^2-1}{2(k-1)} > n \Longleftrightarrow T < n\left(1-\sqrt{2-2\frac{d}{n}+\frac{1}{n^2}}\right),$$

on obtient ainsi le résultat. □

Maintenant, on sait sous quelle hypothèse un tel polynôme interpolateur existe. Comme pour la méthode de Welch-Berlekamp, pour retrouver nos mots de codes sous forme polynomiale, on utilise la proposition suivante.

Proposition 2.7. *Soit $P(X) \in \mathbb{F}_q[X]$ de degré strictement inférieur à k tel qu'il engendre le mot de code associé au mot reçu y. Alors $Q(X, P(X))$ est le polynôme nul.*

Démonstration. On propose de procéder de la même manière que pour la méthode proposée par Welch et Berlekamp. Tout d'abord, on calcule le degré de ce polynôme :

$$\begin{aligned}\deg(Q(X,P(X))) &\leq \max_{j=0}^{\ell}\{n-T-1-j(k-1)+j(k-1)\}\\ &= n-T-1.\end{aligned}$$

Puisque l'on souhaite corriger au plus T erreurs, on suppose qu'il y a moins de T erreurs et donc que le polynôme univarié $Q(X, P(X))$ a au moins $n-T$ racines. On conclut que $Q(X, P(X))$ est le polynôme nul. □

Algorithme 4: Algorithme de décodage en liste de Sudan

Entrées : Le mot reçu $y \in \mathbb{F}_q^n$ et le code de Reed-Solomon $\mathcal{C}$ de support $\mathcal{S}$.
Sorties : Une liste de mots de code c_i de $\mathcal{C}$, tels que $\forall i,\ d(v, c_i) \leq T$.

début
- $Q(X,Y) \longleftarrow$ Interpolation($\mathbf{IP}_S, \mathcal{C}$)
- $(P_1, \ldots, P_\ell) \longleftarrow$ Racine_YLineaire(Q(X,Y))
- $Candidats \longleftarrow \{\}$
- **pour** $i \in \{1, \ldots, \ell\}$ **faire**
  - **si** $d\,(ev_{\mathcal{S}}(P_i), y) \leq T$ **alors**
    - $Candidats \longleftarrow Candidats \cup \{(ev_{\mathcal{S}}(P_i)\}$
- **retourner** $Candidats$.

On représente le gain de cette méthode dans la figure 2.3. On a tracé le rayon de décodage maximum de l'algorithme de Sudan ainsi que celui de décodage unique.

Le gain de la méthode de Sudan n'est valable que pour les codes de Reed-Solomon à très faible rendement – inférieur à 1/5. Le véritable avantage de cette méthode, c'est qu'il ouvre les portes au décodage en liste pour les codes de Reed-Solomon.

2.3.3 Algorithme de Guruswami-Sudan

Guruswami en stage de Master au *Massachusetts Institute of Technology* (MIT) sous la direction de Sudan, a travaillé sur une amélioration de l'algorithme de Sudan, présenté

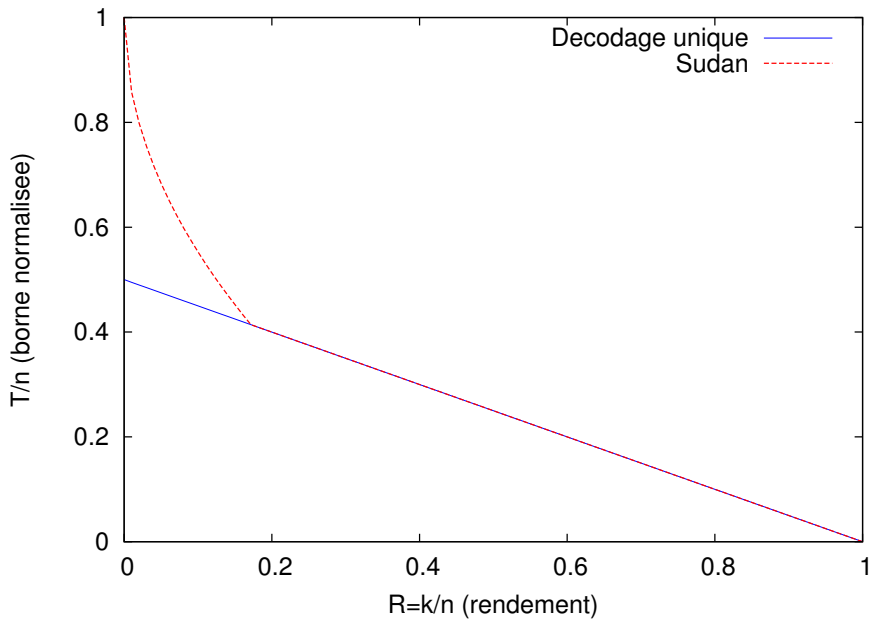

FIGURE 2.3 – Comparaison entre la capacité maximale de correction des algorithmes de décodage unique et de la méthode de Sudan.

dans la sous-section précédente, sous-section 2.3.2. Comme pour l'algorithme de décodage de Sudan, la méthode de Guruswami et Sudan repose sur une remarque simple qui permet de changer la phase d'interpolation et ainsi de décoder encore plus d'erreurs [GS99]. Cette méthode est dédiée aux codes de Reed-Solomon, en tant que codes d'évaluation et aux codes géométriques. Cependant, dans cette thèse on traite exclusivement de leur méthode pour les codes de Reed-Solomon.

Remarque

Soient $RS_q[n,k]$ un code de Reed-Solomon, $y \in \mathbb{F}_q^n$ un mot reçu, $T \in \mathbb{N}$ une borne de décodage et $P_1(X), P_2(X) \in \mathbb{F}_q[X]$ deux polynômes de degré inférieur à k tels que

- $ev_{\mathcal{S}}(P_1), ev_{\mathcal{S}}(P_2) \in \mathcal{B}(y,T)$,
- $\exists i_0 \in \{1,\ldots,n\}\,,\ P_1(\alpha_{i_0}) = P_2(\alpha_{i_0}) = y_{i_0}$.

D'après ce qui précède, on a également

$$Y - P_1(X) \mid Q(X,Y) \text{ et } Y - P_2(X) \mid Q(X,Y).$$

Alors, le point (α_{i_0}, y_{i_o}) est une racine d'ordre au moins deux du polynôme d'interpolation $Q(X,Y) \in \mathbb{F}_q[X,Y]$. L'idée de Guruswami et Sudan est d'augmenter la multiplicité des points d'interpolations du polynôme $Q(X,Y)$ dans l'algorithme de Sudan. On rappelle la définition d'une racine multiple de $Q(X,Y) \in \mathbb{F}_q[X,Y]$.

Définition 2.2 (Racine multiple)**.** Soient $(\alpha, \beta) \in (\mathbb{F}_q)^2$, $Q(X,Y) \in \mathbb{F}_q[X,Y]$ et $Q(X+\alpha, Y+\beta) = \sum_{i\geq 0,\ j\geq 0} q^*_{ij} X^i Y^j$. Le couple (α, β) est une *racine multiple* d'ordre s de $Q(X,Y)$ si

$$\begin{cases} Q(\alpha,\beta) = 0 \\ \forall (i,j),\ \text{tel que } i+j < s,\ q^*_{ij} = 0, \end{cases}$$

où s est le plus grand entier vérifiant cette propriété.

On dit que (α, β) est une racine d'ordre s de $Q(X,Y)$, ou que (α, β) est un zéro de $Q(X,Y)$ avec multiplicité s. Pour démontrer la correction de l'algorithme, on a besoin également du lemme suivant.

Lemme 2.8. *Soient s un entier, $a, b \in \mathbb{F}_q$ et $Q(X,Y) \in \mathbb{F}_q[X,Y]$ un polynôme tel que le point (a,b) est une racine de multiplicité s. Alors pour tout polynôme $P(X)$ non nul tel que $P(a) = b$, on a $(X-a)^s \mid Q(X, P(X))$.*

Démonstration. Le point (a,b) est une racine de $Q(X,Y)$ avec multiplicité s. Ainsi pour q^*_{ij}, définis par

$$Q^*(X,Y) \triangleq Q(X+a, Y+b) \triangleq \sum_{i\geq 0,\ j\geq 0} q^*_{ij} X^i Y^j,$$

on a $q^*_{ij} = 0$ pour tout i, j tels que $i + j < s$.

Montrer que $(X-a)^s \mid Q(X, P(X))$ est équivalent à montrer que $X^s \mid Q(X+a, P(X+a))$.

$$\begin{aligned} Q(X+a, P(X+a)) &= Q^*(X, P(X+a) - b), \\ &= Q^*(X, P_1(X)), \end{aligned}$$

avec $P_1(X) \triangleq P(X+a) - b$. De plus, $P_1(0) = P(a) - b = 0$ et donc il existe $P_2(X)$ tel que $P_1(X) = XP_2(X)$.

$$\begin{aligned} Q(X+a, P(X+a)) &= Q^*(X, XP_2(X)), \\ &= \sum_{i\geq 0,\ j\geq 0} q^*_{ij} X^i (XP_2(X))^j. \end{aligned}$$

Comme $q^*_{ij} = 0$ pour tous i et j tels que $i + j < s$, on en déduit que

$$\begin{aligned} X^s &\mid Q(X+a, P(X+a)) \\ (X-a)^s &\mid Q(X, P(X)). \end{aligned}$$

□

Algorithme

L'algorithme de Guruswami-Sudan est fortement basé sur la méthode de Sudan. Cependant dans l'algorithme de Guruswami-Sudan, la notion de multiplicité des points d'interpolation apparaît. Dans cette partie, on voit comment cette notion permet d'augmenter le rayon de décodage. En gardant les mêmes notations que précédemment, on commence par calculer le polynôme interpolateur $Q(X,Y)$ satisfaisant

$$(\mathbf{IP}_{GS}) \triangleq \begin{cases} 0 \neq Q(X,Y) \triangleq \sum_{i=0}^{\ell} Q_i(X)Y^i, \\ Q(\alpha_i, y_i) = 0, \text{ avec multiplicité } s, \ \forall i \in \{1,\dots,n\}, \\ \deg(Q_j) \leq s(n-T) - 1 - j(k-1), \ \forall j \in \{0,\dots,\ell\}. \end{cases}$$

De même que pour les algorithmes de Welsh-Berlekamp ou de Sudan, on peut montrer les deux mêmes propositions. La première assure l'existence d'un polynôme $Q(X,Y)$ satisfaisant $(\mathbf{IP}_{GS})$, la seconde montre que le polynôme recherché $P(X)$ est un facteur Y-linéaire de $Q(X,Y)$.

Proposition 2.9. *Si l'égalité suivante est satisfaite*

$$T < n\left(1 - \sqrt{1 - \frac{d}{n}}\right)$$

alors il existe un s assez grand tel qu'un polynôme $Q(X,Y)$ satisfaisant $(\mathbf{IP}_{GS})$ existe toujours pour tout mot reçu $y \in \mathbb{F}_q^n$ et tout codes de Reed-Solomon.

Démonstration. De même que pour les méthodes de Welch-Berlekamp et Sudan, si le nombre de contraintes linéaire que donne $(\mathbf{IP}_{GS})$ est plus grand que le nombre d'indéterminées, alors pour tout mot reçu d'un code de Reed-Solomon, un polynôme $Q(X,Y)$ satisfaisant $(\mathbf{IP}_{GS})$ existe toujours . Avec l'aide du lemme 2.5, on obtient

$$\begin{aligned} \frac{s^2(n-T)^2 - 1}{2(k-1)} &> n\binom{s+1}{2} \\ \frac{(n-T)^2 - \frac{1}{s^2}}{n-d} &> n\frac{s+1}{s} \\ n - T &> \sqrt{\left(n + \frac{n}{s}\right)(n-d) + \frac{1}{s^2}} \\ T &< n\left(1 - \sqrt{1 - \frac{d}{n} + \frac{n-d}{sn} + \frac{1}{s^2n^2}}\right). \end{aligned}$$

Pour s assez grand, on obtient la borne de Johnson générique :

$$T < n\left(1 - \sqrt{1 - \frac{d}{n}}\right),$$

alors pour un s assez grand on a :

$$\frac{s^2(n-T)^2-1}{2(k-1)} > n\binom{s+1}{2} \Longleftrightarrow T < n\left(1-\sqrt{1-\frac{d}{n}}\right),$$

ce qui conclut la preuve. □

Proposition 2.10. *Soit $P(X) \in \mathbb{F}_q[X]$ de degré strictement inférieur à k qui engendre le mot de code associé au mot reçu y. Alors $Q(X, P(X))$ est le polynôme nul.*

Démonstration. On propose de procéder de la même manière que pour les méthodes de Welch-Berlekamp et Sudan. Tout d'abord, on calcule le degré de ce polynôme :

$$\begin{aligned} \deg(Q(X,P(X))) &\leq \max_{j=0}^{\ell}\{s(n-T)-1-j(k-1)+j(k-1)\} \\ &= s(n-T)-1. \end{aligned}$$

Puisque l'on souhaite corriger au plus T erreurs, on suppose qu'il y a moins de T erreurs et donc que le polynôme univarié $Q(X, P(X))$ a au moins $s(n-T)$ racines. On conclut alors que $Q(X, P(X))$ est le polynôme nul. □

On remarque que si l'on effectue l'interpolation avec multiplicité $s = 1$, on retrouve exactement les mêmes conditions d'existence du polynôme interpolateur $Q(X, Y)$ et les mêmes conditions d'interpolation que pour l'algorithme de Sudan, sous-section 2.3.2.

Algorithme 5: Algorithme de décodage en liste de Guruswami-Sudan

Entrées : Le mot reçu $y \in \mathbb{F}_q^n$ et le code de Reed-Solomon $\mathcal{C}$ de support $\mathcal{S}$.
Sorties : Une liste de mots de code c_i de $\mathcal{C}$, tels que $\forall i,\ d(v, c_i) \leq T$.

début
- $Q(X,Y) \longleftarrow$ Interpolation($\mathbf{IP}_{GS}, \mathcal{C}$)
- $(P_1, \ldots, P_\ell) \longleftarrow$ Racine_YLineaire(Q(X,Y))
- $Candidats \longleftarrow \{\}$
- **pour** $i \in \{1, \ldots, \ell\}$ **faire**
  - **si** $d\,(ev_{\mathcal{S}}(P_i), y) \leq T$ **alors**
    - $Candidats \longleftarrow Candidats \cup \{(ev_{\mathcal{S}}(P_i)\}$
- **retourner** $Candidats$.

On suggère à partir de l'ancienne figure 2.3, de comparer la capacité maximale de décodage de la méthode de Guruswami et Sudan avec celles de Sudan et du décodage unique, voir figure 2.4. Le principal avantage de cette méthode est que l'on arrive à décoder au-delà de la capacité de correction, borne du décodage unique, pour tous les rendements des codes de Reed-Solomon. Ce qui n'est pas le cas avec l'algorithme de Sudan.

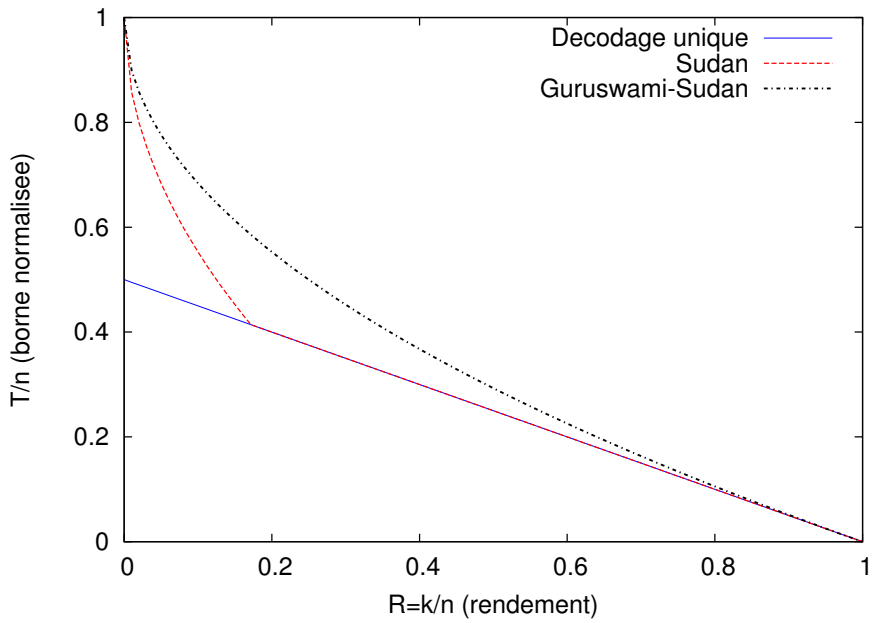

FIGURE 2.4 – Comparaison de la capacité maximale de correction des algorithmes de décodage unique, de Sudan et de la méthode de Gurusami-Sudan.

2.4 Notre décodage en liste pour les codes alternants

Dans cette sous-section, on reprend une idée présentée par Guruswami pour les codes géométriques dans son manuscrit de thèse [Gur05]. On l'applique aux codes alternants et tout particulièrement aux codes de Goppa binaires. En collaboration avec Daniel Augot et Alain Couvreur, on a indépendamment redécouvert, un résultat méconnu dû à Tal et Roth [TR03], qui permet de décoder en liste les codes alternants sur $\mathbb{F}_q$ jusqu'à la borne de Johnson q-aire.

Cette section fait l'objet d'un rapport de recherche [ABC10]. Une version courte a également été acceptée à la conférence ITW'11 [ABC11].

2.4.1 Principe de base

Cette méthode utilise le fait qu'un code alternant est un *subfield subcode* d'un code de Reed-Solomon généralisé, définition 1.39, page 22. L'idée principale est d'utiliser l'algorithme de Guruswami-Sudan et de changer la phase d'interpolation pour que les mots de code retournés par cette méthode restent dans le sous corps dans lequel est défini le code alternant.

2.4.2 Décodage en liste des codes alternants en général

Soit $\mathcal{C}$ le code alternant sur $\mathbb{F}_q$ que l'on souhaite décoder. Par la définition d'un code alternant, définition 1.39 page 22, il existe un code de Reed-Solomon généralisé GRS sur $\mathbb{F}_{q^m}$ de dimension k_{GRS} et de distance minimale d_{GRS} tel que

$$\mathcal{C} = GRS[n, k_{GRS}] \cap \mathbb{F}_q^n.$$

Le décodage en liste proposé permet de décoder $\mathcal{C}$ jusqu'à γn erreurs, où γ est précisé par la suite. Soit $y \in \mathbb{F}_q^n$ un mot reçu, associé au code alternant $\mathcal{C}$. Comme pour les algorithmes de décodage en liste de Sudan et de Guruswami-Sudan, cette méthode se décompose en deux étapes principales : l'interpolation et la recherche de racines. Comme les calculs se font sur des mots du code de Reed-Solomon, on rajoute une étape supplémentaire qui est la reconstruction. Pour élaborer cet algorithme, on définit une variable auxiliaire supplémentaire $s \in \mathbb{N} \setminus \{0\}$.

1. Interpolation : Calculer $Q(X,Y) = \sum_{j=0}^{\ell} Q_j(X)Y^j \in \mathbb{F}_{q^m}[X,Y]$ tel que
   (a) (non trivial) $Q(X,Y) \neq 0$;

   (b) (interpolation avec multiplicité variable)
   – $\mathrm{mult}(Q, (\alpha_i, y_i\beta_i^{-1})) \geq s(1-\gamma)$;
   – $\mathrm{mult}(Q, (\alpha_i, z\beta_i^{-1})) \geq \frac{s\gamma}{q-1}$, pour tout $z \in \mathbb{F}_q \setminus \{y_i\}$;

   (c) (degré pondéré)
   $\deg(Q_j) < sn\left((1-\gamma)^2 + \frac{\gamma^2}{q-1}\right) - j(k_{GRS}-1),\ \forall j \in \{0, \ldots, \ell\}$;

2. Recherche de racines : Trouver tous les facteurs $(Y - f(X))$ de $Q(X,Y)$, avec $\deg f(X) < k_{GRS}$;

3. Reconstruction : Calculer les mots du code $\mathcal{C}$ associés aux facteurs $f(X)$ – calculés à l'étape précédente.

4. Retourner seulement les mots de code qui sont à distance au plus γn de y.

Les conditions d'interpolation (1a), (1b) et (1c) forment le problème d'interpolation $\mathbf{IP}_{ABC}$. À partir du lemme 2.8, on peut déduire la proposition suivante, qui fournit la correction de l'algorithme.

Proposition 2.11 (Augot, B., Couvreur - 2010)**.** *Soient y le mot reçu et $Q(X,Y)$ satisfaisant les conditions 1a, 1b, et 1c ci-dessus. Soit $P(X)$ le polynôme tel que $\deg P(X) < k_{GRS}$. Si $d(ev_{L,B}(P(X)), y) \leq \gamma n$ alors $Q(X, f(X)) = 0$.*

Démonstration. Soit κ définit par $d(ev_{L,B}(P(X)), y) = \kappa n \leq \gamma n$. Soient l'ensemble $I \triangleq \{i, P(\alpha_i) = y_i\beta_i^{-1}\}$ et son complémentaire $\overline{I} \triangleq \{i, P(\alpha_i) \neq y_i\beta_i^{-1}\}$. On a $|I| = n(1-\kappa)$ et $|\overline{I}| = \kappa n$. Par le lemme 2.8 à la page 42, $Q(X, P(X))$ est divisible par

$$\prod_{i\in I}(X-\alpha_i)^{\lceil s(1-\gamma)\rceil} \times \prod_{j\in\overline{I}}(X-\alpha_j)^{\lceil s\gamma/(q-1)\rceil},$$

qui a pour degré $D = n(1-\kappa)\lceil s(1-\gamma)\rceil + n\kappa\left\lceil\frac{s\gamma}{q-1}\right\rceil$. Ce degré D est décroissant en fonction de κ pour $\gamma < \frac{q-1}{q}$. C'est une fonction affine par rapport à la variable κ ayant pour terme dominant

$$\kappa n\left(\left\lceil\frac{s\gamma}{q-1}\right\rceil - \lceil s(1-\gamma)\rceil\right) < 0.$$

Le minimum est atteint pour $\kappa = \gamma$. On obtient donc $D \geq sn\left((1-\gamma)^2 + \frac{\gamma^2}{q-1}\right)$.

Or, la condition (1c) sur le degré pondéré de $Q(X,Y)$ implique que $\deg(Q(X, P(X))) < sn\left((1-\gamma)^2 + \frac{\gamma^2}{q-1})\right)$. On en déduit que le polynôme univarié $Q(X, P(X))$ possède plus de racines que son degré, ainsi on obtient $Q(X, P(X)) = 0$. □

La proposition suivante nous assure l'existence d'un tel polynôme interpolateur $Q(X,Y)$.

Proposition 2.12 (Augot, B., Couvreur - 2010)**.** *Pour tout γ tel que*

$$\gamma < \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\frac{d_{GRS}}{n}}\right),$$

il existe s assez grand tel qu'un polynôme $Q(X,Y)$ satisfaisant les conditions 1a, 1b, et 1c, existe pour tout mot reçu $y \in \mathbb{F}_q^n$.

Démonstration. Pour démontrer cette proposition, il suffit de calculer une borne sur γ, telle que le système linéaire engendré par 1b et 1c a toujours plus d'indéterminées que d'équations pour tout mot reçu $y \in \mathbb{F}_q^n$. Cela donne l'inégalité suivante

$$\frac{s^2n^2((1-\gamma)^2 + \frac{\gamma^2}{q-1})^2}{2(k_{GRS}-1)} > \left(\binom{s(1-\gamma)+1}{2} + (q-1)\binom{s\frac{\gamma}{q-1}+1}{2}\right)n, \qquad (2.3)$$

qui peut être réécrite comme

$$\left((1-\gamma)^2 + \frac{\gamma^2}{q-1}\right)^2 > R'\left((1-\gamma)^2 + \frac{\gamma^2}{q-1} + \frac{1}{s}\right),$$

où $R' \triangleq \frac{k_{GRS}-1}{n}$. On trouve alors $\mu(\gamma) \triangleq (1-\gamma)^2 + \frac{\gamma^2}{q-1}$ qui doit satisfaire $\mu^2 - R'\mu - \frac{R'}{s} > 0$. Les racines de l'équation $\mu^2 - R'\mu - \frac{R'}{s} = 0$ sont

$$\mu_0 = \frac{R' - \sqrt{R'^2 + 4\frac{R'}{s}}}{2}, \quad \mu_1 = \frac{R' + \sqrt{R'^2 + 4\frac{R'}{s}}}{2}.$$

La fonction $\mu(\gamma)$ est décroissante pour l'intervalle considéré, c'est-à-dire $\gamma \in [0, 1 - \frac{1}{q}]$. Puisque la solution μ_0 est négative, elle ne peut pas convenir. Seule la solution μ_1 l'est. On doit avoir également $\mu(\gamma) > \mu_1$:

$$(1-\gamma)^2 + \frac{\gamma^2}{q-1} > \mu_1. \tag{2.4}$$

On obtient également deux racines pour l'équation $(1-\gamma)^2 + \frac{\gamma^2}{q-1} = \mu_1$, qui sont

$$\gamma_0 = \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}(1-\mu_1)}\right) \tag{2.5}$$

$$\gamma_1 = \frac{q-1}{q}\left(1 + \sqrt{1 - \frac{q}{q-1}(1-\mu_1)}\right). \tag{2.6}$$

Mais comme $\gamma_0 < \frac{q-1}{q}$, on a

$$\gamma < \gamma_0 = \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}(1-\mu_1)}\right). \tag{2.7}$$

Ainsi, quand $s \to \infty$, on obtient $\mu_1 \to R'$ et on a

$$\gamma < \tau = \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}(1-R')}\right) \tag{2.8}$$

En utilisant le fait que les codes de Reed-Solomon sont MDS, c'est-à-dire $k_{GRS} - 1 = n - d_{GRS}$, ou encore $R' = 1 - \frac{d_{GRS}}{n}$, on obtient

$$\tau = \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\frac{d_{GRS}}{n}}\right),$$

qui est exactement la borne de Johnson q-aire normalisée. □

Le rayon de décodage atteint grâce à cet algorithme est bien plus grand que ceux obtenus par les méthodes précédentes [GS99, Ber08]. La figure 2.5 nous permet de comparer les différents algorithmes de décodage pour les codes de Goppa binaires.

2.4.3 Analyse de la complexité

Notre principal souci ici est d'estimer s assez grand pour que le polynôme interpolateur puisse toujours satisfaire les conditions 1a, 1b et 1c. Tout d'abord on propose le lemme suivant qui nous permet d'exhiber une relation entre les paramètres de l'algorithme.

Lemme 2.13. *Soit* $\tau \triangleq \frac{q-1}{q}\left(1 - \sqrt{1 - \frac{q}{q-1}\frac{d_{GRS}}{n}}\right)$, *on a alors l'égalité suivante*

$$\frac{\tau^2}{q-1} + (1-\tau)^2 = 1 - \frac{d_{GRS}}{n}.$$

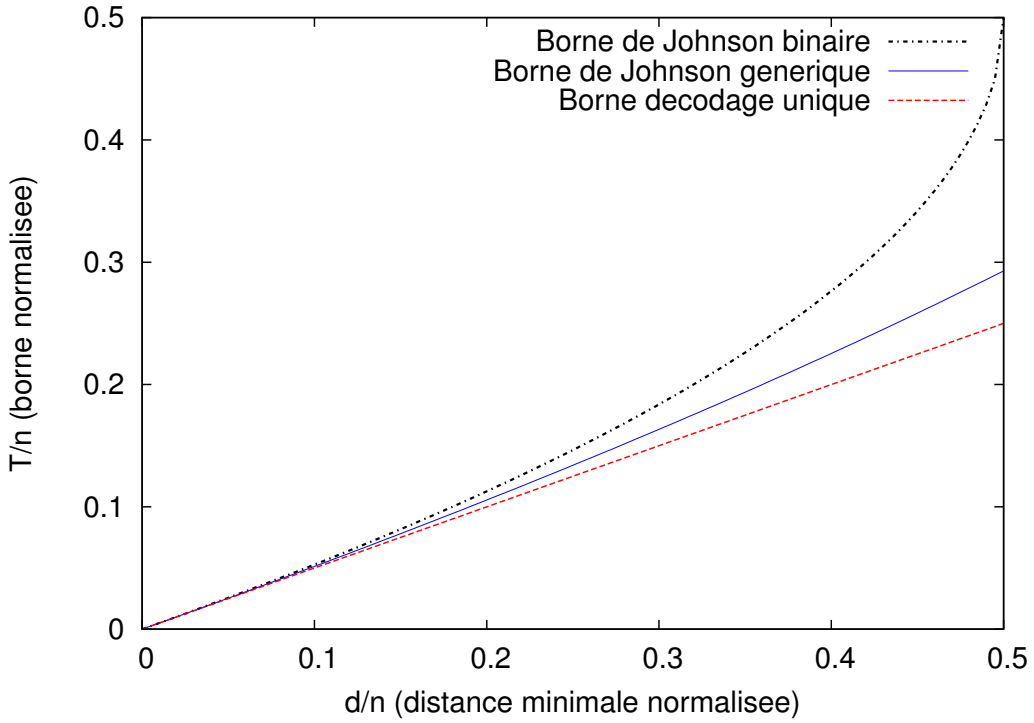

FIGURE 2.5 – Comparaison des rayons de décodage de différents algorithmes de décodage pour les codes de Goppa binaires. La borne de Johnson binaire est atteinte par notre méthode, la borne de Johnson générique par la méthode de Berstein [Ber08] et le décodage unique par les algorithmes de décodage unique, par exemple celui de Patterson.

Démonstration. On calcule partie par partie.

$$\begin{aligned}
\tau &\triangleq \frac{q-1}{q} - \frac{q-1}{q}\sqrt{1-\frac{q}{q-1}\frac{d_{GRS}}{n}} \\
\frac{\tau^2}{q-1} &= \frac{q-1}{q^2} + \frac{q-1}{q^2}\left(1-\frac{q}{q-1}\frac{d_{GRS}}{n}\right) - 2\frac{q-1}{q^2}\sqrt{1-\frac{q}{q-1}\frac{d_{GRS}}{n}}
\end{aligned}$$

$$\begin{aligned}
1-\tau &= \frac{1}{q} + \frac{q-1}{q}\sqrt{1-\frac{q}{q-1}\frac{d_{GRS}}{n}} \\
(1-\tau)^2 &= \frac{1}{q^2} + \left(\frac{q-1}{q}\right)^2\left(1-\frac{q}{q-1}\frac{d_{GRS}}{n}\right) + 2\frac{q-1}{q^2}\sqrt{1-\frac{q}{q-1}\frac{d_{GRS}}{n}}
\end{aligned}$$

Il ne reste plus qu'à recoller les deux parties et on obtient :

$$\begin{aligned}
(1-\tau)^2+\tau^2 &= \left(1-\frac{q}{q-1}\frac{d_{GRS}}{n}\right)\left(\frac{(q-1)^2}{q^2}+\frac{q-1}{q^2}\right)+\frac{1}{q} \\
&= \left(1-\frac{q}{q-1}\frac{d_{GRS}}{n}\right)\left(\frac{q-1}{q}\right)+\frac{1}{q} \\
&= 1-\frac{d_{GRS}}{n}.
\end{aligned}$$

□

Lemme 2.14 (Augot, B., Couvreur - 2010)**.** *Pour pouvoir décoder en liste jusqu'au rayon relatif $\gamma = (1-\varepsilon)\tau$, il suffit que la multiplicité auxiliaire s soit en $\mathcal{O}(\frac{1}{\varepsilon})$.*

Démonstration. On propose de repartir de l'équation (2.7) et on note $\gamma(s)$ le rayon de décodage que l'on peut atteindre avec cette méthode pour un s donné. On obtient alors

$$\gamma(s) = \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}\left(1-\mu_1(s)\right)}\right),$$

avec $\mu_1(s) \triangleq \frac{R'+\sqrt{R'^2+4\frac{R'}{s}}}{2}$. On sait que pour tout $x>0$, on a $\sqrt{1+x} < 1+\frac{x}{2}$. Avec les notations précédentes, on a

$$\begin{aligned}
\mu_1(s) &= \frac{R'+\sqrt{R'^2+4\frac{R'}{s}}}{2} \\
&= \frac{R'}{2}\left(1+\sqrt{1+\frac{4}{sR'}}\right) \\
&\leq \frac{R'}{2}\left(1+\left(1+\frac{4}{2sR'}\right)\right) \\
&= R'+\frac{1}{s}.
\end{aligned}$$

Pour alléger les équations, on pose $R'_q \triangleq 1 - \frac{q}{q-1}(1-R')$, ce qui donne

$$\begin{aligned}
\gamma(s) &= \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}\left(1-\mu_1(s)\right)}\right) && (2.9)\\
&\geq \frac{q-1}{q}\left(1-\sqrt{1-\frac{q}{q-1}\left(1-R'-\frac{1}{s}\right)}\right) && (2.10)\\
&= \frac{q-1}{q}\left(1-\sqrt{R'_q+\frac{q}{q-1}\frac{1}{s}}\right) && (2.11)\\
&= \frac{q-1}{q}\left(1-\sqrt{R'_q\left(1+\frac{q}{q-1}\frac{1}{s}\frac{1}{R'_q}\right)}\right) && (2.12)\\
&\geq \frac{q-1}{q}\left(1-\sqrt{R'_q}-\sqrt{R'_q}\frac{1}{2}\frac{q}{q-1}\frac{1}{s}\frac{1}{R'_q}\right) && (2.13)\\
&= \frac{q-1}{q}\left(1-\sqrt{R'_q}\right)-\frac{1}{2}\frac{1}{s\sqrt{R'_q}} && (2.14)\\
&= \tau_q(\delta)-\frac{1}{2s\sqrt{R'_q}} && (2.15)\\
&= \tau_q(\delta)\left(1-\frac{1}{2s\tau_q(\delta)\sqrt{R'_q}}\right). && (2.16)
\end{aligned}$$

Ainsi, pour atteindre le rayon de décodage relatif $\gamma = (1-\varepsilon)\tau$, il suffit de prendre

$$\begin{aligned}
s &= \frac{1}{2\varepsilon\tau\sqrt{R'_q}}\\
&= \frac{1}{2\varepsilon\tau\sqrt{\frac{-1}{q-1}(1-R')}}\\
&= \mathcal{O}\left(\frac{1}{\varepsilon}\right). && (2.17)
\end{aligned}$$

□

Comme dans les autres algorithmes de décodage en liste tels Sudan ou Guruswami-Sudan, la méthode se divise algorithmiquement en deux parties : la phase d'interpolation et la phase de recherche de racines. La deuxième partie, celle de la recherche de racines, est en complexité beaucoup plus faible comparativement aux différentes méthodes d'interpolation [RR00]. On s'intéresse à l'algorithme d'interpolation de Koetter [KVA03, Koe96], qui a pour complexité $\mathcal{O}(\ell C^2)$, où ℓ est le degré en Y de $Q(X,Y)$ et C et le nombre de contraintes linéaires qu'engendrent les contraintes de l'interpolation.

Corollaire 2.15 (Augot, B., Couvreur - 2010)**.** *La méthode de décodage en liste proposée est en $\mathcal{O}(\frac{1}{\varepsilon^5}n^2)$ opérations sur le corps pour décoder en liste jusqu'à $(1-\varepsilon)J_q(n,\delta)$ erreurs, la constante du grand-$\mathcal{O}$ dépendant seulement de q et de R'.*

Démonstration. On suppose qu'on souhaite décoder jusqu'à $n\gamma = (1-\varepsilon)J_q(n,\delta)$. Le nombre d'équation est alors en $\mathcal{O}(ns^2)$ – voir équation (2.3). Maintenant, ℓ le degré en Y de $Q(X,Y)$ est au plus

$$\frac{sn((1-\gamma)^2 + \frac{\gamma^2}{q-1})}{k-1} = \mathcal{O}(s), \tag{2.18}$$

avec $R' = \frac{k-1}{n}$ de fixé. Comme $s = \mathcal{O}(\frac{1}{\varepsilon})$ (voir (2.17)), on conclut que cette méthode est en $\mathcal{O}(n^2\varepsilon^{-5})$. La recherche de racine proposée dans [RR00] est en $\mathcal{O}(\ell^3k^2)$, on obtient alors la complexité $\mathcal{O}(\varepsilon^{-5}n^2)$, qui est plus petit que l'interpolation. □

Corollaire 2.16 (Augot, B., Couvreur - 2010)**.** *La méthode proposée atteignant la borne de Johnson q-aire $J_q(n,\delta)$, est en $\mathcal{O}(n^7)$ opérations sur le corps, la constante du grand-$\mathcal{O}$ dépendant uniquement de q et de R'.*

Démonstration. Il suffit de considérer le lemme 2.14 et le fait qu'on souhaite décoder jusqu'à $J_q(n,\delta)(1-\varepsilon)$, avec $\varepsilon = \mathcal{O}(\frac{1}{n})$. □

2.4.4 Application aux codes de Goppa binaires

D'après la proposition 1.38, les codes de Goppa sont des codes alternants. Il est alors clair que l'on peut appliquer la méthode présentée pour décoder les codes de Goppa. De plus, d'après la proposition 1.40, si le polynôme de Goppa $G(X)$ est sans facteur carré, on a l'égalité des deux codes de Goppa binaires suivants :

$$\Gamma_2(L,G) = \Gamma_2(L,G^2).$$

Ce code bénéficie de la dimension de $\Gamma_2(L,G)$ ainsi que de la distance minimale de $\Gamma_2(L,G^2)$. Avec notre méthode, on obtient alors pour les codes de Goppa binaires ayant un polynôme de Goppa sans facteur carré de degré t, un rayon de décodage de

$$\left\lceil \frac{1}{2}\left(n - \sqrt{n(n-4t-2)}\right)\right\rceil - 1.$$

Avec la proposition 1.40, cet algorithme de décodage en liste pour les codes alternants permet également de décoder jusqu'à la borne de Johnson q-aire pour les codes de Goppa définis sur $\mathbb{F}_q$.

Comme les codes alternants sont par définition des *subfield subcode* des codes de Reed-Solomon généralisés, on peut appliquer les méthodes existantes pour les codes de Reed-Solomon aux codes alternants : il suffit pour cela de supprimer dans la liste des mots de code retournée, ceux qui ne sont pas des mots du code alternant. On peut ainsi utiliser l'algorithme de décodage en liste de Guruswami-Sudan pour les codes alternants [GS99]. À ces deux méthodes, on compare l'algorithme de décodage en liste proposé par Berstein [Ber08] pour les codes de Goppa binaires ainsi que notre méthode dédiée aux codes alternants. Bien qu'il soit facile de connaître une borne inférieure sur la distance

minimale d'un code de Goppa, il est difficile de connaître sa valeur exacte. Le rayon de décodage de notre méthode est fortement lié à la distance minimale du code de Reed-Solomon généralisé – proposition 2.12. Ainsi, pour les codes de Goppa qui ont une distance minimale bien plus grande que la borne inférieure, on ne sait que les décoder jusqu'à la borne de Johnson q-aire dépendant de la distance minimale construite, et non la vraie distance minimale. Dans la table 2.1, on compare les différents algorithmes de décodage en liste dépendant de la longueur n et du degré du polynôme de Goppa t.

GS [GS99]	B [Ber08]	Notre méthode [ABC10]
$\left\lceil n - \sqrt{n(n-2t)} \right\rceil - 1$	$n - \sqrt{n(n-2t-2)}$	$\left\lceil \frac{1}{2}\left(n - \sqrt{n(n-4t-2}\right) \right\rceil - 1$

Table 2.1 – Comparaison des rayons de décodage des méthodes de décodage en liste en fonction de t le degré du polynôme de Goppa et de n la longueur du code.

Dans la table 2.2 on compare les différentes méthodes existantes. La méthode de Patterson adapte l'algorithme de Berlekamp-Massey à l'équation de clé engendrée par les code de Goppa [Pat75]. Cette méthode est un algorithme de décodage unique.

n	k	t [Pat75]	GS [GS99]	B [Ber08]	Notre méthode [ABC10]
512	197	35	36	37	38
512	107	45	47	48	50
512	17	55	58	59	63
1024	524	50	51	52	53
1024	424	60	62	62	74
1024	324	70	73	73	76
1024	224	80	83	84	88
1024	124	90	94	95	100
2048	948	100	103	103	105
2048	728	120	124	124	128
2048	508	140	145	146	151
2048	398	150	156	157	163
2048	288	160	167	167	175
2048	178	170	178	178	187

Table 2.2 – Comparaison de la capacité de correction de différents algorithmes pour les codes de Goppa binaires de longueur n, de dimension k et t le degré du polynôme de Goppa sans facteur carré.

Algorithme 6: Algorithme de décodage en liste pour les codes de Goppa

Entrées : Le mot reçu $y \in \mathbb{F}_q^n$ et le code $\Gamma(\mathcal{L}, G)$.
Sorties : Une liste de mots du code $c_i \in \Gamma(\mathcal{L}, G)$ tels que $d(c_i, y) \leq T$.

début
  $Q(X,Y) \longleftarrow$ Interpolation($\mathbf{IP}_{ABC}, \Gamma(\mathcal{L}, G)$)
  $(P_1, \ldots, P_\ell) \longleftarrow$ Racine_YLineaire(Q(X,Y))
  $Candidats \longleftarrow \{\}$
  pour $i \in \{1, \ldots, \ell\}$ **faire**
    si $d\left((\beta_1 P_i(\mathcal{L}_1), \ldots, \beta_n P_i(\mathcal{L}_n)), y\right) \leq T$ **alors**
      $Candidats \longleftarrow Candidats \cup \{(\beta_1 P_i(\mathcal{L}_1), \ldots, \beta_n P_i(\mathcal{L}_n))\}$
  retourner $Candidats$.

2.5 Nouvelle classe de codes pour le décodage complet

Dans cette partie on propose une définition d'une nouvelle classe de codes, appelée codes $\mathcal{A}$-*couverts*. La définition des codes $\mathcal{A}$-couverts est simple et très naturelle. La lettre $\mathcal{A}$ fait référence à un algorithme de décodage, bien souvent un algorithme de décodage en liste. On appelle un code $\mathcal{C}$ dans une famille $(\mathcal{C}_i)$, un code $\mathcal{A}$-*couvert* si l'algorithme $\mathcal{A}$ décode la famille $(\mathcal{C}_i)$ en temps polynomial – en les paramètres des codes – et que le rayon de recouvrement de $\mathcal{C}$ est inférieur à la capacité de correction de $\mathcal{A}$. Dans la sous-section 2.5.2 on définit notre classe de codes, on y montre que le problème du décodage complet pour cette famille de codes se résout en temps polynomial. Le cas des codes binaires est traité dans la sous-section 2.5.3. On y exhibe un lemme qui donne des critères généraux pour trouver des codes de cette classe. En appliquant ce lemme, on démontre que les code de Reed-Muller du premier ordre sont dans notre classe. Étant données les dernières avancées dans le décodage en liste des BCH binaires et Goppa binaires, on s'est intéressé également aux codes $\mathcal{A}$-couvert de cette nature. On a trouvé 12 codes de Goppa, malheureusement de très faible dimension, et 4 codes BCH binaires pour lesquels on sait maintenant résoudre le problème du décodage complet en temps polynomial. Nos résultats expérimentaux mettent en évidence également la difficulté de ce problème, par exemple pour les codes de Reed-Solomon.

Cette section à fait l'objet d'un article dans la conférence internationale YACC'10 [Bar10].

2.5.1 Introduction

En 1978, le problème de décodage pour un code en général a été montré NP-difficile. En 2005 Guruswami et Vardy ont démontré que le problème de décodage complet est NP-difficile pour les codes de Reed-Solomon [GV05]. En revanche, ce problème est résoluble pour la famille des codes parfaits linéaires en tenant compte des deux

observations :

- tous les codes parfaits linéaires sont connus [Tie73, ZL73] : les codes de Hamming, le code de Golay binaire de paramètre $[23, 12, 7]_2$, et le code de Golay ternaire de paramètre $[11, 6, 5]_3$;

- pour tous les codes parfaits linéaires, il existe des algorithmes de décodage unique en temps polynomial en les paramètres de ces codes.

Comme ces codes sont parfaits, leur capacité de correction t est égale à leur rayon de recouvrement ρ. De plus, pour tous ces codes il existe un algorithme de décodage qui résout le problème du décodage unique, voir le problème 1.52. Grâce à ces algorithmes, on sait alors résoudre le problème du décodage complet – problème 1.48. Compte tenu des dernières avancées en terme de décodage, à savoir l'algorithme de décodage en liste proposé par Wu [Wu08] et le notre – voir la section 2.4 ou [ABC10] – on se demande si ces méthodes peuvent résoudre le problème du décodage complet pour certains codes. D'un point de vu strictement algorithmique, on propose de faire un parallèle entre les codes parfaits et notre classe de codes, pour lesquels on sait décoder jusqu'au rayon de recouvrement.

2.5.2 Les codes $\mathcal{A}$-couverts

Définition 2.3 (Algorithme de décodage en liste)**.** Soit $\mathcal{C}$ un code. On appelle $\mathcal{A}$ *un algorithme de décodage en liste* de $\mathcal{C}$ jusqu'à $\tau_{\mathcal{A}}$ si pour tout mot de l'espace ambiant v, l'algorithme $\mathcal{A}(v)$ retourne tous les mots de codes $c \in \mathcal{C}$ tels que $d(v, c) \leq \tau_{\mathcal{A}}$.

Définition 2.4 (Algorithme de décodage en liste en temps polynomial)**.** Soient $\mathcal{C}_i$ une famille de codes de longueur n_i et $\mathcal{A}$ un algorithme de décodage en liste de la famille $\mathcal{C}_i$. On appelle $\mathcal{A}$ un *algorithme de décodage en liste en temps polynomial* si $\mathcal{A}$ s'exécute en $\mathcal{O}(n_i^c)$, avec $c \in \mathbb{N}$ pour tout $\mathcal{C}_i$.

Remarque 2.1. La taille de la liste d'un tel algorithme ne peut être que polynomiale également.

Définition 2.5 (Code $\mathcal{A}$-couvert)**.** Soient $\mathcal{C}$ un code de rayon de recouvrement ρ et $\mathcal{A}$ un algorithme de décodage en liste en temps polynomial de $\mathcal{C}$ jusqu'à $\tau_{\mathcal{A}}$. On appelle $\mathcal{C}$ un code $\mathcal{A}$-*couvert* si

$$\rho \leq \tau_{\mathcal{A}}.$$

On a défini la notion de code $\mathcal{A}$-couvert pour le problème du décodage complet, on obtient trivialement la proposition suivante.

Proposition 2.17 (B. - 2010)**.** *Soit $\mathcal{C}$ un code $\mathcal{A}$-couvert. Le problème du décodage complet est résoluble en temps polynomial en la longueur du code.*

D'un point de vue strictement algorithmique, on peut dire que les codes $\mathcal{A}$-couverts sont l'analogue des codes parfaits. Cependant, il est clair que de manière générale, les codes $\mathcal{A}$-couverts n'héritent pas des propriétés combinatoires des codes parfaits.

Décodage unique	Décodage en liste
Code parfait	Code $\mathcal{A}$-couvert
$\rho = t$	$\rho \leq \tau_{\mathcal{A}}$

FIGURE 2.6 – Codes parfait et code $\mathcal{A}$-couvert.

Une question naturelle est de savoir si cette classe des codes $\mathcal{A}$-couvert est seulement constituée des codes parfaits ou non.

2.5.3 Le cas binaire

Définition 2.6 (Code auto-complémentaire)**.** Soit $\mathcal{C}$ un code binaire. On dit que $\mathcal{C}$ est *auto-complémentaire* si pour tout mot de code $c \in \mathcal{C}$, son complémentaire est dans le code.

$$\forall c \in \mathcal{C}, \exists \bar{c} \in \mathcal{C} \text{ tel que } c + \bar{c} = \mathbf{1}.$$

Proposition 2.18. *Soit $\mathcal{C}$ un code linéaire binaire. Il est auto-complémentaire si et seulement si le mot tout à un est un mot du code.*

Théorème 2.19. *Soit $\mathcal{C}$ un code binaire, auto-complémentaire et de longueur n. Si sa distance duale, notée $d^{\perp}$, est plus grande ou égale à 2, alors son rayon de recouvrement ρ est plus petit ou égal à $\lfloor (n - \sqrt{n})/2 \rfloor$.*

Démonstration. Preuve détaillée dans [CHLL97, Théorème 9.2.2, page 241]. □

On remarque que cette borne supérieure du rayon de recouvrement de tels codes ressemble à la borne de Johnson binaire. Soit $\mathcal{C}$ un code linéaire binaire vérifiant les conditions du théorème 2.19 et tel qu'il existe $\mathcal{A}$ un algorithme de décodage en liste en temps polynomial qui décode $\mathcal{C}$ jusqu'à la borne de Johnson binaire. Alors une condition suffisante – mais pas nécessaire – pour que $\mathcal{C}$ soit $\mathcal{A}$-couvert est

$$\begin{aligned}
\frac{n-\sqrt{n}}{2} &\leq \frac{n}{2}\left(1-\sqrt{1-\frac{2d}{n}}\right) \\
\sqrt{n} &\geq n\sqrt{1-\frac{2d}{n}} \\
\sqrt{n} &\geq \sqrt{n(n-2d)} \\
d &\geq \frac{n-1}{2}.
\end{aligned}$$

Si d la distance minimale du code est impaire alors la dimension du code est majorée par la borne de Plotkin, théorème 1.7, page 7.

Parmi les algorithmes de décodage en liste présentés, les seuls à atteindre la borne de Johnson binaire sont l'algorithme de Wu [Wu08] pour les BCH binaires, ainsi que notre méthode pour tous les codes alternants binaires – section 2.4. Ces deux algorithmes de décodage en liste arrivent à atteindre la borne de Johnson binaire dépendant de la distance minimale construite et non de la vraie distance minimale. D'après toutes nos précédentes remarques, on obtient le corollaire suivant.

Corollaire 2.20 (B., 2010)**.** *Soit $\mathcal{C}$ un code binaire, auto-complémentaire, de longueur n, de distance construite $\delta \geq (n-1)/2$ et $d^{\perp} \geq 2$. S'il existe $\mathcal{A}$ un algorithme de décodage en liste en temps polynomial de $\mathcal{C}$ jusqu'à la borne de Johnson binaire, alors $\mathcal{C}$ est un code $\mathcal{A}$-couvert.*

Les BCH binaires

On s'intéresse ici aux BCH binaires. La notion de code $\mathcal{A}$-couvert fait référence à un couple : code et algorithme de décodage. On commence par présenter l'algorithme de décodage en liste en temps polynomial de Wu [Wu08] pour décoder les BCH binaires.

Wu : un algorithme de décodage en liste en temps polynomial. L'algorithme de Wu est détaillé en annexe B. Cette méthode décode jusqu'à la borne de Johnson binaire et s'exécute en temps polynomial, c'est un algorithme de décodage en liste en temps polynomial.

Les codes de Reed-Muller du premier ordre poinçonnés. Comme expliqué dans [MS77], on peut voir les codes de Reed-Muller du premier ordre poinçonnés comme des codes BCH binaires, ce qui a pour avantage de pouvoir appliquer l'algorithme de Wu à ces codes. Les codes de Reed-Muller du premier ordre poinçonnés sont des $[2^m - 1, m + 1, 2^{m-1} - 1]_2$-code linéaires pour $m > 1$.

Proposition 2.21 (B., 2010)**.** *Les codes de Reed-Muller du premier ordre poinçonnés sont Wu-couverts.*

Démonstration. On propose de vérifier les hypothèses du corollaire 2.20.

- Le dual d'un Reed-Muller du premier ordre poinçonné $(\mathcal{RM}(1, m)^*)^{\perp}$ est un $[2^m - 1, 2^m - 1 - m, 3]_2$-code de Hamming qui a pour distance minimale 3.
- Le code de Reed-Muller est le code simplexe augmenté par le vecteur tout a un. D'après la proposition 2.18, on en conclut que le code de Reed-Muller est un code auto-complémentaire.
- La distance minimale d'un code de Reed-Muller poinçonné du premier ordre est $\delta = 2^{m-1} - 1 = (n - 1)/2$.

Toutes les hypothèses du corollaire 2.20 sont bien vérifiées. □

Résultats. Pour dire si un code est couvert par un algorithme, il faut connaître son rayon de recouvrement ρ. Cependant, connaître le rayon de recouvrement est un problème difficile. On tire partie d'une recherche exhaustive réalisée en MAGMA [BCP97] ainsi que des résultats théoriques proposés dans [CHLL97]. Malheureusement, la recherche exhaustive permet uniquement de trouver des codes de petites longueurs, c'est-à-dire plus petit que 31. Dans la table 2.3, on peut trouver des codes de Hamming ainsi que des codes de Reed-Muller poinçonnés du premier ordre. Les codes de Hamming font partie de la famille des codes parfaits. On sait donc que le problème du décodage complet pour ces codes est résoluble. En outre, grâce à la proposition 2.21, on sait que les codes de Reed-Muller poinçonnés du premier ordre sont Wu ou GKZ-couvert.

La dimension d'un code de Reed-Muller poinçonné du premier ordre est logarithmique en sa longueur. Effectuer une recherche exhaustive sur les mots de codes est alors linéaire. On en déduit que la recherche exhaustive pour résoudre le problème du décodage complet pour de tels codes est aussi un algorithme de décodage en liste en temps polynomial.

n	k	d	ρ	τ	Commentaires
7	4	3	1	2	Hamming
15	11	3	1	1	Hamming
15	7	5	3	3	BCH Wu-couvert
15	5	7	5	5	RM(1,4)*
17	9	5	3	3	BCH Wu-couvert
23	12	7	3	4	BCH Wu-couvert
31	26	3	1	1	Hamming
31	11	11	7	7	BCH Wu-couvert
31	6	15	11	12	RM(1,5)*
63	57	3	1	1	Hamming
63	7	31	27	27	RM(1,6)*
127	8	63	55	57	RM(1,7)*

TABLE 2.3 – Exemples de $[n, k, d]_2$-codes de rayon de recouvrement ρ qui sont couverts par l'algorithme de Wu qui décode jusqu'à τ erreurs. Les rayons de recouvrement de tous les BCH binaires Wu-couverts jusqu'à la longueur 23 ont été trouvés par une recherche exhaustive. Les autres codes de longueur plus grande ont été trouvés grâce aux résultats théoriques. On remarque que certains BCH binaires trouvés sont en fait des codes de Hamming binaires ou des Reed-Muller poinçonnés du premier ordre.

Bien que notre espace d'investigation soit réduit par le calcul du rayon de recouvrement ρ et que le problème du décodage complet est attendu pour être asymptotiquement difficile, on réussit à trouver quatre codes BCH binaires $[15, 7, 5]_2$, $[17, 9, 5]_2$, $[23, 12, 7]_2$ et $[31, 11, 11]_2$ qui sont Wu-couvert.

Les Goppa binaires

Algorithme de décodage en liste en temps polynomial. Dans ce contexte, on recherche tout d'abord un algorithme qui décode le maximum d'erreurs pour les codes de Goppa binaires. Il s'avère que la méthode que l'on propose dans la section 2.4 décode plus d'erreurs que les autres algorithmes . De plus, notre étude de la complexité montre que c'est un algorithme de décodage en liste en temps polynomial.

Résultat. Comme on a un phénomène de doublement de la distance minimale pour les codes de Goppa définis par un polynôme de Goppa sans racine multiple, on regarde ces codes avec notre méthode de décodage en liste proposé en section 2.4. Un polynôme irréductible est sans racine multiple, on s'intéresse particulièrement aux codes de Goppa binaires engendrés par un polynôme de Goppa irréductible. On effectue une recherche exhaustive en fixant d'abord le degré de l'extension m et le degré du polynôme de Goppa r. Pour tous les codes de Goppa binaires engendrés par les polynômes irréductibles de degré r sur $\mathbb{F}_{2^m}$, on fait varier la longueur du code n et on retient tous les codes qui ont pour rayon de recouvrement ρ supérieur ou égal à τ la borne de Johnson binaire atteinte par notre méthode.

m	n	k	d	ρ	τ
4	10	2	5	4	4
	10	3	5	4	4
	14	2	7	6	6
	14	3	7	6	6
5	10	3	5	4	4
	14	2	7	6	6
	18	2	9	8	8
	18	3	9	8	8
	22	2	11	10	10
	22	3	11	10	10
6	10	2	5	4	4
	18	2	9	8	8

TABLE 2.4 – Exemples de codes $[n, k, d]_2$ construis sur $\mathbb{F}_{2^m}$ de rayon de recouvrement ρ qui sont couverts par l'algorithme décrit en section 2.4, qui décode jusqu'à τ erreurs. Tous les codes de Goppa binaires irréductibles ont été trouvés par recherche exhaustive.

On remarque que tous les codes de Goppa binaires irréductibles trouvés sont de dimension 2 ou 3. Une recherche exhaustive sur tous les mots de codes, aurait été plus efficace que notre algorithme de décodage. La faible dimension s'explique que pour décoder loin on doit être dans une gamme de paramètres où la distance minimale normalisée est assez haute. Ce qui impose d'avoir un rendement k/n assez bas.

On remarque également que différents jeux de paramètres, comme par exemple $[10, 2, 5]_2$, sont présentés plusieurs fois pour des degrés d'extension m différents. Comme ces codes ne sont pas équivalents, il s'agit bien de codes différents.

2.5.4 Conclusion

D'un point de vue strictement algorithmique, on propose une nouvelle classe de codes, les codes $\mathcal{A}$-couverts, pour lesquels le problème du décodage complet, connu pour être NP-difficile, se résout en temps polynomial. Bien que la difficulté de trouver de tels codes soit liée au calcul de leur rayon de recouvrement, on montre un lemme général pour les codes binaires et déduit que les codes de Reed-Muller poinçonnés du premier ordre sont couverts par différents algorithmes. Bien que nous ayons trouvé douze codes de Goppa binaires couverts de dimensions ridiculement petites, on a réussi à trouver 4 BCH binaires de dimensions raisonnables. Ces résultats expérimentaux nous laissent espérer une intersection non vide avec notre nouvelle classe et les BCH binaires.

Chapitre 3

Application au cryptosystème de McEliece

Plan du chapitre

3.1 Introduction

La cryptologie est la discipline qui permet de sécuriser, d'assurer l'intégrité des données, l'authentification, etc... Elle se décompose en deux branches : la cryptographie qui a pour objectif de proposer des systèmes de protection, et la cryptanalyse qui a pour but de mener des attaques pour essayer de mettre à mal les systèmes proposés. Évidemment ces deux notions sont fortement liées, car pour proposer des systèmes de chiffrement il faut bien connaître les attaques qu'ils vont subir. Le schéma habituel proposé est le suivant : Alice souhaite transmettre un message *clair* à Bob et souhaite que si Ève intercepte la transmission, de quelque manière que ce soit, elle ne puisse pas avoir accès au contenu de ce message. Alice chiffre son message pour Bob à l'aide d'un système de chiffrement et d'une clé, elle obtient ainsi le *chiffré* correspondant. Elle envoie donc le chiffré à Bob. Bob doit être la seule personne capable de pouvoir retrouver le message clair. Ceci est rendu possible grâce à l'utilisation d'une clé de déchiffrement qu'Ève ne

connaît pas. Ceci respecte les principes de Kerckhoffs datant de 1883 [Ker83], qui sont considérés aujourd'hui comme fondamentaux en cryptologie :

- la sécurité d'un système cryptographique repose uniquement sur la non divulgation de la clé de déchiffrement,
- le système cryptographique doit être au préalable étudié par des experts.

On doit retenir de ses principes, qu'il ne faut pas faire reposer la sécurité d'un système de chiffrement sur la méconnaissance que l'attaquant a des procédés de chiffrement et de déchiffrement. Au contraire, la sécurité doit seulement reposer sur la méconnaissance de la clé de déchiffrement et sur le fait que le système cryptographique est connu et fortement étudié. Dans certains cryptosystèmes la clé de chiffrement – utilisée par Alice pour engendrer le chiffré – et de déchiffrement – utilisée par Bob pour retrouver le message clair à partir du chiffré – sont identiques. Ce système de chiffrement est appelé un cryptosystème *symétrique*. Par opposition, un cryptosystème est dit *asymétrique* lorsque la clé de chiffrement est différente de celle de déchiffrement. Évidemment, ces deux clés sont fortement liées l'une à l'autre. Elles sont calculées grâce à une fonction à sens unique avec trappe. C'est une fonction qui est facile à calculer, à inverser lorsque l'on connaît la trappe, mais très difficile à inverser lorsque l'on ne connaît pas la trappe. Par exemple, dans le cryptosystème RSA, la clé publique est la multiplication de deux grands premiers et la clé privée associée est l'un de ces premiers.

3.1.1 Cryptosystème symétrique

Historiquement, les premières cryptosystèmes sont symétriques.

Définition 3.1 (Cryptosystème symétrique)**.** Soit $\mathcal{K}$ la clé secrète. Un *cryptosystème symétrique* est défini par une fonction de chiffrement paramétrée par la clé $\mathcal{K}$, $E_{\mathcal{K}}$ et une fonction de déchiffrement paramétrée par la clé $\mathcal{K}$, $D_{\mathcal{K}}$ telles que pour tout message clair m, on ait

$$D_{\mathcal{K}}(E_{\mathcal{K}}(m)) = m.$$

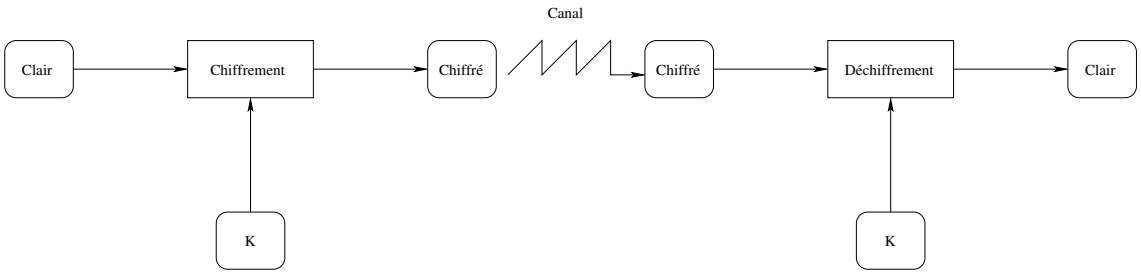

FIGURE 3.1 – Schéma d'un cryptosystème symétrique.

Il existe des cryptosystèmes symétriques dit *chiffrement par blocs* qui découpent le clair en blocs de même longueur et chiffre bloc par bloc, comme le DES ou l'AES. Il existe également des cryptosystèmes symétriques qui ne découpent pas le clair en blocs mais gèrent le clair comme un flux de données, ce sont des *chiffrements à flot*, par

exemple E0 et A5/1, qui sont respectivement les standards de chiffrement du *bluetooth* et du *GSM*. Une des grandes particularités des cryptosystèmes symétriques est leur rapidité de chiffrement et de déchiffrement, surtout pour le cas des chiffrements à flot. L'inconvénient majeur des cryptosystèmes symétriques est que l'expéditeur et le destinataire, c'est-à-dire Alice et Bob, doivent se mettre d'accord au préalable sur un secret commun : la clé $\mathcal{K}$. Ce qui présuppose l'existence d'un canal de communication sûr.

3.1.2 Cryptosystème asymétrique

Les cryptosystèmes asymétriques sont plus récents. Il a fallu attendre 1976 pour que Diffie et Hellman proposent l'idée d'avoir des clés de chiffrement et de déchiffrement différentes [DH76]. Cependant dans cet article, aucun exemple de cryptosystème n'a été présenté. Il a fallu attendre que Rivest, Shamir et Adleman proposent en 1978 un schéma de signature et de chiffrement asymétrique [RSA78], appelé RSA.

Définition 3.2 (Cryptosystème asymétrique)**.** Soient $\mathcal{K}_P$ la clé publique, $\mathcal{K}_S$ la clé secrète associée. Un *cryptosystème asymétrique* est défini par une fonction de chiffrement paramétrée par la clé publique $E_{\mathcal{K}_P}$ et une fonction de déchiffrement paramétrée par la clé secrète $D_{\mathcal{K}_S}$, telles que pour tout message clair m, on ait

$$D_{\mathcal{K}_S}(E_{\mathcal{K}_P}(m)) = m.$$

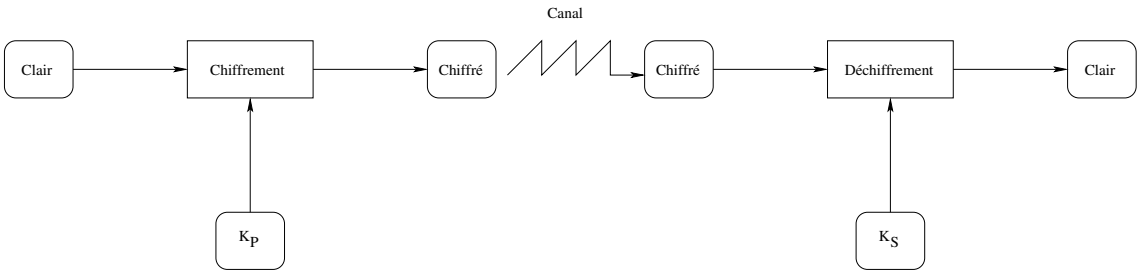

FIGURE 3.2 – Schéma d'un cryptosystème asymétrique.

Quand on compare les cryptosystèmes asymétriques aux symétriques, les cryptosystèmes asymétriques ont un atout non négligeable : on ne suppose pas au préalable l'existence d'un canal de communication sûr. On suppose en effet que la clé publique est accessible par tout le monde, par exemple stockée sur un serveur sécurisé. Cependant, les cryptosystèmes asymétriques sont plus lents comparés aux cryptosystèmes symétriques. En pratique, on utilise un cryptosystème asymétrique pour échanger les clés d'un cryptosystème symétrique, dans ce contexte la cryptographie asymétrique joue le rôle de canal de communication sûr. Ensuite, le message clair est alors chiffré à l'aide d'un cryptosystème symétrique. Ceci permet de bénéficier à la fois des avantages des cryptosystèmes symétrique et asymétrique.

L'attaquant Ève peut, comme tout à chacun, avoir accès à la clé publique. Les cryptosystèmes à clés publiques reposent sur des problèmes difficiles à résoudre : inverser une fonction à sens unique sans connaître de trappe. Il est donc difficile pour Éve de retrouver la clé privée à partir de la clé publique.

Dans le reste de ce chapitre, on présente le cryptosystème asymétrique de McEliece [McE03]. On commence par décrire le système, pour ensuite expliquer sur quel problème difficile de la théorie des codes correcteurs il repose. On finit par un rappel des différentes cryptanalyses connues sur ce système.

3.2 Cryptosystème de McEliece

Le cryptosystème de McEliece est le premier cryptosystème asymétrique basé sur la théorie des codes correcteurs [McE78].

3.2.1 McEliece

Dans un cryptosystème asymétrique, le chiffré – obtenu grâce à la clé publique – se déchiffre grâce à la clé privée. Il existe donc un lien très fort entre la clé privée et la clé publique. C'est pourquoi on commence par présenter la génération des clés dans le cryptosystème de McEliece. On explique ensuite l'étape de chiffrement, puis la phase de déchiffrement.

Génération des clés

L'article original de McEliece qui présente son cryptosystème utilise les codes de Goppa classiques binaires [McE78]. Jusqu'à présent, cette version résiste aux cryptanalyses connues, à taille des paramètres près [BLP08]. Le cryptosystème de McEliece est basé sur un problème difficile de la théorie des codes : décodage d'un code aléatoire. En toute généralité, on peut donc supposer que l'on peut instancier le cryptosystème avec n'importe quel code ce qui donne lieu à différentes variantes. Cependant, on verra dans la sous-section 3.2.3 que certaines variantes utilisant des codes fortement structurés ont été cryptanalysées.

Algorithme 7: Génération des clés pour McEliece

Entrées : Un $[n, k]$-code $\mathcal{C}$ pour lequel on sait décoder t erreurs.
Sorties : Un couple de clés publique et privée $(\mathcal{K}_P, \mathcal{K}_S)$.

début
- $G \longleftarrow$ une matrice $k \times n$ génératrice de $\mathcal{C}$,
- $S \longleftarrow$ une matrice aléatoire inversible $k \times k$,
- $P \longleftarrow$ une matrice de permutation $n \times n$, choisie aléatoirement,
- $G' \longleftarrow SGP$,
- $\mathcal{K}_P \longleftarrow (G', t)$,
- $\mathcal{K}_S \longleftarrow (S^{-1}, G, P^{-1})$,
- **retourner** $(\mathcal{K}_P, \mathcal{K}_S)$.

On remarque que les clés publique et privée du cryptosystème de McEliece sont des matrices : matrice génératrice d'un $[n, k]$-code $\mathcal{C}$, matrice de permutation et matrice inversible. Pour avoir une sécurité raisonnable, les tailles des matrices doivent être assez importantes, cela implique des tailles de clés conséquentes. C'est l'inconvénient principal du cryptosystème de McEliece, et c'est pour cela qu'aujourd'hui ce système de chiffrement n'est pas souvent utilisé dans la vie courante. Par exemple, le réseau de communication pair-à-pair *Entropy* utilise ce système pour chiffrer les communications [Cay08].

Chiffrement

La phase de chiffrement est très simple, il suffit de multiplier le clair m par la matrice de la clé publique G' et de modifier aléatoirement t composantes du résultat. Plus exactement, on ajoute une erreur de poids t au mot du code engendré par le clair m.

Algorithme 8: Chiffrement avec le cryptosystème de McEliece

Entrées : Le clair m et la clé publique $\mathcal{K}_P = (G', t)$.
Sorties : Le chiffré m'.

début
- $c' \longleftarrow mG'$,
- $e \longleftarrow$ vecteur aléatoire de poids t,
- $m' \longleftarrow c' + e$,
- **retourner** m'.

Un idée astucieuse utilisée dans l'implémentation de Biswas et Sendrier est de non plus choisir un vecteur d'erreurs aléatoire, mais de l'utiliser pour faire passer de l'information également [Sun00, BS08]. De plus, ils proposent d'utiliser la matrice génératrice du code sous sa forme systématique, même si cela impose de travailler sur un code équivalent. Cette version est appelée *HyMES* pour *Hybrid McEliece Encryption Scheme*.

Déchiffrement

Le déchiffrement est un peu plus compliqué que le chiffrement, mais reste néanmoins assez simple. Il consiste à annuler l'effet de la permutation, puis à décoder le mot obtenu pour supprimer les erreurs ajoutées durant le chiffrement, enfin par annuler l'effet de la matrice S.

Algorithme 9: Déchiffrement avec McEliece

Entrées : Le chiffré m' et la clé privée $\mathcal{K}_S = (S^{-1}, G, P^{-1})$.
Sorties : Le clair m.

début
- $c'' \longleftarrow m'P^{-1}$,
- $m'' \longleftarrow$ décoder c'',
- $m \longleftarrow m''S^{-1}$,
- **retourner** m.

On remarque que le déchiffrement repose essentiellement sur un algorithme de décodage. Maintenant, on peut vérifier que c'est bien un cryptosystème, c'est-à-dire qu'à partir des bonnes clés et des bons algorithmes de chiffrement et de déchiffrement, on est bien en mesure de retrouver le message clair m. Soient $G' = SGP$ la matrice de la clé publique et e le vecteur d'erreur ajouté pendant le chiffrement. On a alors :

$$\begin{aligned} m' &= mSGP + e, \\ m'P^{-1} &= mSG + eP^{-1}. \end{aligned}$$

Comme e est un vecteur d'erreur de poids t et P une permutation alors eP^{-1} est un autre vecteur d'erreur de poids toujours égal à t, donc

$$\begin{aligned} dec(m'P^{-1}) &= mS, \\ dec(m'P^{-1})S^{-1} &= m. \end{aligned}$$

3.2.2 Problème difficile sous-jacent

Les multiplications par une matrice inversible, puis par une matrice de permutation masquent la structure du code de Goppa, engendrant un code de Goppa équivalent, mais qui semble être un code aléatoire. Pendant très longtemps, on a pensé que la sécurité du cryptosystème de McEliece reposait sur la difficulté de distinguer un code de Goppa d'une matrice aléatoire et que l'on ne savait pas décoder un code sans en connaître sa structure. En réalité, la première assertion est fausse. En effet, Faugère, Otmani, Tillich et Perret ont montré qu'on peut distinguer un code de Goppa avec un fort rendement d'un code aléatoire [FOPT10b]. Même si l'on arrive à distinguer un code de Goppa d'un code aléatoire, on ne peut retrouver la matrice de permutation P et la matrice inversible S, c'est-à-dire retrouver la bonne structure du code de Goppa originalement utilisé. Sans la connaissance de cette structure, il est très difficile de

décoder ce code.

Une dizaine d'années après la publication du cryptosystème de McEliece, Niederreiter a proposé un schéma similaire reposant sur une matrice de parité au lieu d'une matrice génératrice du code [Nie86]. L'équivalence de la sécurité entre ces deux cryptosystèmes a été montrée dans [LDW94]. Sans perte de généralité, on présente uniquement le cryptosystème de McEliece.

3.2.3 Cryptanalyses du cryptosystème de McEliece

Il existe deux types de cryptanalyses du cryptosystème de McEliece. La première, l'*attaque par décodage* qui tente de décoder un code quelconque sans en connaître la structure. C'est un problème NP-complet. La seconde, l'*attaque structurelle*, est dédiée aux variantes de McEliece qui utilisent des codes fortement structurés.

Attaque par décodage

La première apparition de ce type d'attaque a été introduite par McEliece lui-même lors de la publication de son cryptosystème [McE78]. Cette attaque porte également le nom d'*Information Set Decoding (ISD)*. On introduit les notations qui sont utilisées dans l'algorithme 10 qui présente l'attaque par ensemble d'information.

Notation 3.1. Soient $\mathcal{I} \subset \{1, \ldots, n\}$, G une matrice à n colonnes et y un vecteur de n composantes. On note

- $G_{\mathcal{I}}$ la matrice composée des colonnes de G indexées par $\mathcal{I}$,
- $y_{\mathcal{I}}$ la projection de y sur $\mathcal{I}$.

Algorithme 10: Décodage par ensemble d'information

Entrées : Une matrice génératrice G de $\mathcal{C}$ et le mot reçu $y \in \mathbb{F}_q^n$.
Sorties : L'erreur e.

début
  $y' \longleftarrow (1, \ldots, 1)$,
  répéter
    $\mathcal{I} \longleftarrow k$ positions aléatoires de $\{1, \ldots, n\}$,
    si $G_{\mathcal{I}}$ *est inversible* **alors** `/* ` $\mathcal{I}$ `est un ensemble d'information */`
      $G' \longleftarrow G_{\mathcal{I}}^{-1} G$,
      $y' \longleftarrow y - y_{\mathcal{I}} G'$,
  jusqu'à $w(y') \leq t$;
  retourner $e = y'$.

On peut remarquer que cette attaque repose sur l'hypothèse que l'intersection entre l'ensemble d'information $\mathcal{I}$ et le support de l'erreur soit vide. Lee et Brickell proposent différentes astuces pour relaxer très légèrement cette hypothèse et permettre un petit

nombre d'erreurs p dans l'ensemble d'information [LB88]. Une première approche probabiliste a été introduite par Leon [Leo88]. Sa méthode suit le même raisonnement que précédemment mais ajoute que le poids de l'erreur doit être nul dans une petite partie dans l'ensemble de redondance. Dans [Ste89], Stern améliore la méthode de Leon en séparant l'ensemble d'information $\mathcal{I}$ en deux sous-ensembles, ce qui lui permet de relâcher encore plus la condition sur le poids de l'erreur dans l'ensemble d'information, passant de p à $2p$. Toutes ces méthodes permettent de relaxer la condition que l'intersection du support de l'erreur et de l'ensemble d'information soit vide. Sinon il faut tirer aléatoirement un autre ensemble d'information et tout recommencer.

Ce qui explique l'idée principale de la méthode de Canteaut et de Chabaud, de ne plus prendre aléatoirement un ensemble d'information, mais essayer d'en construire un nouveau par rapport aux précédents pour alléger le calcul de l'inverse de $G'_{\mathcal{I}}$ [CC98]. Bernstein, Lange et Peters ont ensuite proposé de garder en mémoire certains calculs pour les réutiliser dans les calculs suivants, ce qui permet de cryptanalyser en pratique la variante originale proposée par McEliece [BLP08, McE78], un code de Goppa binaire de paramètres $[1024, 524, 50]$, ce qui avait déjà été annoncé dans le domaine du réalisable [CS98]. Dernièrement, dans [BLP10], les auteurs ont proposé de modifier la contrainte de Leon pour la relâcher tout en contrôlant le poids de l'erreur dans une partie de l'ensemble de redondance.

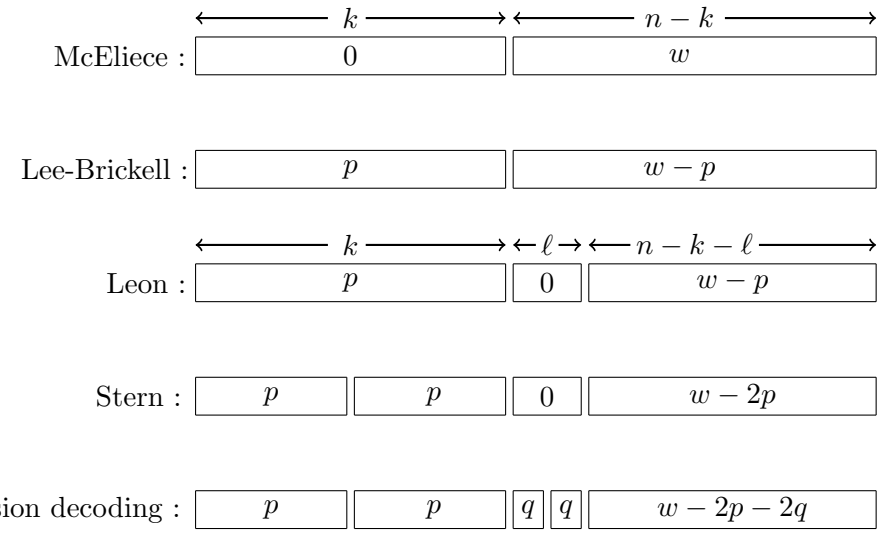

FIGURE 3.3 – Répartition du poids de l'erreur selon différentes attaques.

La figure 3.3 compare la répartition des poids possibles de l'erreur pour les différentes attaques énoncées précédemment. Évidemment on n'y trouve pas la méthode de Canteaut et Chabaud, car cette méthode ne change pas la répartition du poids de l'erreur. Cette figure est reprise de [OS09, BLP10].

Attaque structurelle

Les attaques structurelles visent des variantes précises de McEliece. Ce type d'attaque tente de retrouver la clé secrète de la clé publique, en profitant au maximum de la structure des codes utilisés. Comme on a pu le voir précédemment, l'inconvénient majeur du cryptosystème de McEliece est la taille de ses clés. Beaucoup d'auteurs ont essayé d'utiliser des codes très structurés, dans le but de limiter la clé publique à une sous-matrice de la matrice génératrice, qui permet d'en reconstruire le reste. En 2005, Berger et Loidreau ont proposé d'utiliser des codes de Reed-Solomon généralisés avec une clé publique basée sur une matrice génératrice d'un sous-code [BL05]. Ce qui a donné lieu, en 2006, à une attaque ne remettant pas en cause la variante en elle-même mais les jeux des paramètres à utiliser [Wie06]. En 2005, Gaborit [Gab05] a obtenu une significative réduction de clé grâce à l'utilisation de sous-codes de BCH primitifs. En 2010, Otamni, Tillich et Dallot ont cryptanalysé cette variante [OTD10]. En 2008, Berger, Cayrel, Gaborit et Otmani ont constaté une forte diminution de la taille des clés si on utilise des codes alternants quasi-cycliques [BCGO09]. Le décodage des codes quasi-cycliques en toute généralité est un problème, ils ont proposé alors l'utilisation des codes quasi-cycliques alternants. Avec le même principe, Misoczki et Barreto ont regardé dans [MB09] la réduction de clé avec l'utilisation des codes quasi-dyadiques de Goppa, qui sont des codes de Goppa. Un an après, en 2010, Faugère, Otmani, Perret et Tillich ont mis à mal certains paramètres des variantes basées sur les codes alternants, grâce au calcul d'une base de Groebner [FOPT10a].

3.3 Notre réduction de clé pour McEliece

Comme on l'a remarqué précédemment, la taille des clés du cryptosystème de McEliece en est le principal défaut. Dans cette section, on montre comment l'utilisation d'un décodage en liste permet de réduire la taille des clés du cryptosystème de McEliece. De plus, on propose deux contre mesures à l'attaque structurelle introduite dans [FOPT10a] pour renforcer la variante dyadique. On observe alors un gain en la longueur des clés allant de 4% pour la variante générique, jusqu'à un gain de 21% pour la variante dyadique.

Ce travail, effectué en collaboration avec Paulo Barreto, a fait l'objet d'une présentation lors de la conférence internationale ISIT 2011 [BB11].

3.3.1 Décodage en liste des codes de Goppa binaires

Comme un type de cryptanalyse du cryptosystème de McEliece est intiment lié à la correction des erreurs, une idée naturelle pour augmenter la complexité de ce type d'attaque est d'ajouter, durant la phase de chiffrement, autant d'erreurs que le destinataire est capable de corriger. Le cryptosystème de McEliece est originalement basé sur les codes de Goppa. Différentes propositions ont été faites basées sur différents codes, leur forte structure permet de réduire la taille des clés mais les expose aux attaques algébriques. On commence par rappeler l'état de l'art du décodage des codes

de Goppa binaires et on continue par expliquer comment le décodage en liste peut-être appliquée dans le schéma de McEliece.

Le premier algorithme de décodage algébrique des codes de Goppa a été proposé en 1975 par Patterson [Pat75]. Cet algorithme, est une variante de la méthode de Berlekamp et Massey, en annexe A, qui s'exécute en temps quadratique en la longueur du code. La méthode de Patterson permet de décoder les codes de Goppa jusqu'à t, sa capacité de correction, c'est donc un algorithme de décodage unique. Comme les codes de Goppa classiques sont des codes alternants, c'est-à-dire des *subfield subcode* des codes de Reed-Solomon généralisés, proposition 1.38 page 22, on peut utiliser l'algorithme de décodage en liste de Guruswami-Sudan, sous-section 2.3.3. Cette méthode nous permet de corriger jusqu'à la borne de Johnson générique donnée par $n\left(1-\sqrt{1-\frac{2t}{n}}\right)$ erreurs, qui est plus grande que t, figure 3.4. En conséquence, ce type de décodage ne permet plus d'assurer l'unicité du mot de code retourné. On remarque que l'algorithme de Guruswami-Sudan n'a pas été conçu pour les codes de Goppa. En utilisant les propriétés des codes de Goppa binaires, Bernstein est capable d'étendre la méthode de Patterson pour exhiber un algorithme de décodage en liste jusqu'à $n\left(1-\sqrt{1-\frac{2t+2}{n}}\right)$ [Ber08], qui est plus grande que la borne de Johnson générique atteinte par la méthode de Guruswami et Sudan. Avec notre méthode de décodage en liste – sous-section 2.4 – on est capable pour les mêmes codes de Goppa binaires de corriger jusqu'à la borne de Johnson binaire : $\tau_2 \triangleq \frac{n}{2}\left(1-\sqrt{1-\frac{4t+2}{n}}\right)$, qui est plus grande que les deux bornes précédentes. Comme il est possible voir dans la figure 3.4, plus la distance normalisée est proche de $1/2$, plus la borne de Johnson binaire est importante comparée aux deux autres. On montre dans la sous-section 3.3.2, que l'utilisation des codes de Goppa binaires ayant une distance minimale normalisée un peu plus proche de $1/2$, permet alors de corriger plus d'erreurs et ainsi de réduire d'avantage la taille des clés.

En accord avec le corollaire 2.15, la complexité totale de l'algorithme pour décoder en liste les codes de Goppa binaires jusqu'à $(1-\epsilon)\tau_2$ erreurs, est $\mathcal{O}(n^2\epsilon^{-5})$, où τ_2 est la borne de Johnson binaire. Ainsi, la complexité exprimée permet alors de réaliser un compromis entre le nombre d'erreurs supplémentaires que l'on peut ajouter et la complexité. Ceci permet de contrôler la complexité totale de l'algorithme de décodage. Décoder jusqu'à τ_2 est assez coûteux pour les paramètres de codes utilisés pour le cryptosystème de McEliece. On peut alors, utiliser des algorithmes rapides pour de tels paramètres baissant considérablement la complexité. Il existe différentes méthodes d'interpolation diminuant ainsi la complexité asymptotique [Ale05, BB10b, BB10a]. De cette manière, la complexité de notre méthode de décodage en liste pour les codes de Goppa reste inférieure à la dernière méthode de décodage en liste proposée par Bernstein. Cette méthode est maintenant capable de décoder, comme la notre, jusqu'à la borne de Johnson binaire [Ber11].

Le cryptosystème de McEliece classique est sensible aux attaques par chiffrés

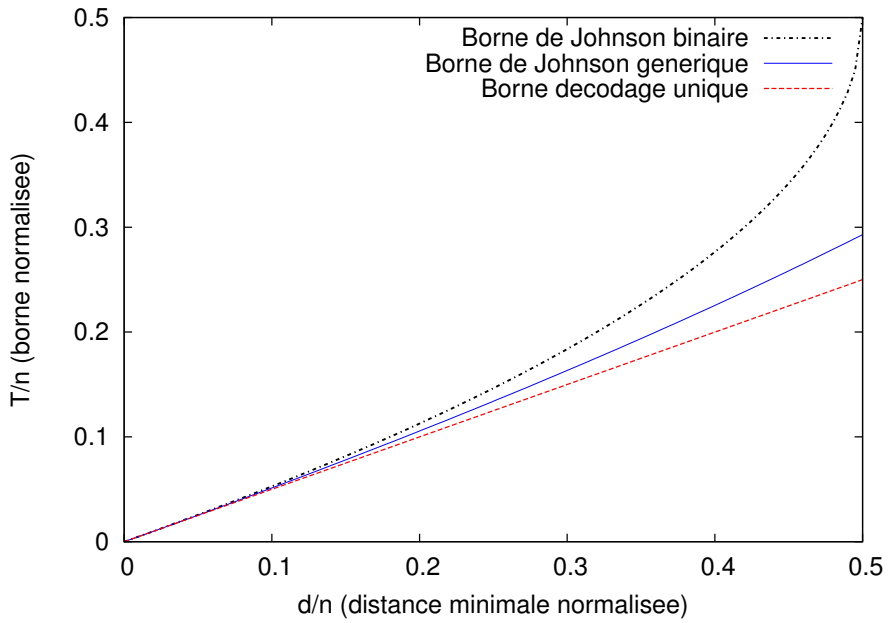

FIGURE 3.4 – Comparaison de la borne de Johnson binaire, de la borne de Johnson générique et de la borne du décodage unique.

choisis [VDVT02]. En effet, différents chiffrés peuvent être obtenus à partir d'un même message clair, un attaquant pourrait comparer ces différents chiffrés et ainsi extraire de l'information sur le clair. Différentes méthodes ont été proposées pour rendre les cryptosystèmes robustes contre ce genre d'attaque [EOS06, OS09, Poi00], menant alors aux variantes appelées *CCA2-secure*. Quand on ajoute plus d'erreurs que la capacité de correction, on n'est plus assuré de l'unicité du clair. Comme l'ont déjà noté les auteurs de [BLP08], les variantes *CCA2-secure* permettent de distinguer le clair original entre tous les candidats retournés par l'algorithme de décodage en liste utilisé durant l'étape de déchiffrement. En conséquence, il est possible de rendre la tâche de l'attaquant plus difficile en ajoutant plus d'erreurs que la capacité de correction. En utilisant les variantes *CCA2-secure* et l'état de l'art du décodage en liste des code de Goppa, les erreurs supplémentaires ajoutent seulement une petite charge supplémentaire au destinataire pour trouver le clair original.

3.3.2 Réduction des clés

Le chiffrement et le déchiffrement avec le cryptosystème de McEliece sont significativement plus rapides que les cryptosystèmes asymétriques plus répandus basés sur la théorie des nombres, tel que l'omniprésent RSA [MB09]. Le principal et peut-être le seul handicap qui empêche l'utilisation courante du cryptosystème de McEliece, est la taille des clés publiques trop conséquente. En suivant l'idée de [BLP08], on propose

d'aborder ce problème, non pas en utilisant des codes très structurés, comme c'est souvent le cas, mais en ajoutant autant d'erreurs que permet la méthode de décodage en liste – section 2.4. Pour une taille de clé donnée, ce procédé a pour conséquence d'augmenter le niveau de sécurité, tant que le nombre d'erreurs reste inférieur à la borne de Gilbert-Varshamov [OS09], cette hypothèse est toujours vérifiée en utilisant notre algorithme de décodage en liste. Symétriquement, cela permet d'utiliser des clés plus petites en gardant un niveau de sécurité similaire. L'utilisation de notre algorithme de décodage en liste nous mène alors à des tailles de clés réduites et à une augmentation du temps de déchiffrement.

On propose de s'attarder sur la famille des codes de Goppa binaires sans racine multiple, qui inclut la famille traditionnellement utilisée des codes de Goppa binaires irréductibles. Dans ce cas, grâce à la proposition 1.40, la capacité de correction t du code est égale à r le degré du polynôme de Goppa. L'algorithme de décodage qui a le plus grand rayon de décodage pour ces codes est étudié dans la section 2.4. Cette méthode de décodage en liste fonctionne pour tous les codes alternants, mais grâce à la proposition 1.40, on peut améliorer le rayon de décodage et aboutir même à des clés plus petites. On a cherché numériquement les paramètres des codes menant à de petites clés en ajoutant $\lceil \tau_2 \rceil - 1$ erreurs. On présente les avantages du décodage en liste sur les variantes générique ainsi que dyadique.

Variante générique

Les tables 3.1, 3.2 et 3.3 montrent la réduction des clés obtenues en utilisant l'algorithme de décodage en liste ayant le plus grand rayon de décodage – section 2.4 – pour les niveaux de sécurités cibles, appelés aussi *workfactors (WF)*, 2^{80}, 2^{112}, 2^{128}, 2^{192} et 2^{256}, ce qui correspond aux niveaux de sécurité standards adoptés pour les autres familles de cryptosystèmes [BBB+07]. Les codes utilisés sont paramétrés par m, le degré de l'extension où G et $\mathcal{L}$ sont définis, la longueur n, la dimension k et le degré r du polynôme de Goppa G. On fait également apparaître dans nos tables τ_2 la borne de Johnson binaire atteinte par notre algorithme de décodage en liste, le logarithme binaire de l'estimation du *workfactor*, la taille en bits des clés et le gain en pourcentage que procure l'utilisation du décodage en liste par rapport au décodage unique. L'estimation du *workfactor* s'effectue par rapport à l'attaque appelée décodage par *ball-collision* [BLP10]. Pour n, k et le poids d'erreur w donnés – avec $w = r$ ou $w = \tau_2$ – alors le *workfactor* est au moins

$$\mathrm{WF} = \min \left\{ \frac{1}{2} \binom{n}{w} \binom{n-k}{w-p}^{-1} \binom{k}{p}^{-1/2} : 0 \leq p \leq \min\{w, k\} \right\}.$$

Pour chaque *workfactor*, les tailles de clés du cryptosystème de McEliece sont données pour le décodage unique et le décodage en liste.

La table 3.1 traite le cas de la variante générique de McEliece où la taille des clés est donnée par $(n-k) \times k = mkr$.

Décodage	m	n	k	r	τ_2	$\log_2(WF)$	taille des clés	gain (%)
unique	11	1893	1431	42		80.025	661122	
liste	11	1876	1436	40	41	80.043	631840	4.43
unique	12	2887	2191	58		112.002	1524936	
liste	12	2868	2196	58	59	112.026	1475712	3.23
unique	12	3307	2515	66		128.007	1991880	
liste	12	3262	2482	65	66	128.021	1935960	2.81
unique	13	5397	4136	97		192.003	5215496	
liste	13	5269	4021	96	98	192.052	5018208	3.78
unique	13	7150	5447	131		256.002	9276241	
liste	13	7008	5318	130	133	257.471	8987420	3.11

TABLE 3.1 – Comparaison entre la taille de la clé publique de la variante générique de McEliece en utilisant du décodage unique et du décodage en liste pour des *workfactors* donnés.

Variante quasi-dyadique

L'attaque proposée par Faugère, Otmani, Perret et Tillich dans [FOPT10a] utilise le calcul de base de Groebner pour retrouver la clé privée à partir de la seule connaissance de la publique. Cette attaque a été spécialement construite pour cryptanalyser les variantes compactes de McEliece proposées dans [BCGO09, MB09], qui utilisent la structure des codes alternants. La variante proposée dans [MB09] utilise des codes de Goppa binaires dans la forme quasi-dyadique. L'attaque dans [FOPT10a] peut être appliquée à la variante quasi-dyadique et on peut donc retrouver une matrice équivalente à la clé privée, sous sa forme de code alternant. Cependant, cela ne suffit pas pour cryptanalyser définitivement cette variante, ainsi que la générique. En effet, l'attaque ne retrouve pas directement le polynôme de Goppa G de degré r qui est crucial pour décoder – voir [Pat75] et la section 2.4 – mais l'attaque permet de calculer une matrice de parité du code alternant – sans le polynôme de Goppa – de distance minimale construite $r+1$. En outre, lorsqu'un code de Goppa binaire sans racine multiple est utilisé, la clé privée est alors une matrice génératrice d'un code de distance minimale construite de $2r+1$, grâce à la proposition 1.40 page 23.

L'attaquant n'est donc pas capable de décoder avec la matrice calculée de cette manière, c'est-à-dire que l'attaquant ne pourra pas immédiatement déchiffrer. Effectivement, la matrice de parité d'un code alternant permet de décoder jusqu'à $r/2$, alors que la phase de chiffrement a ajouté un vecteur erreur de poids r. Cependant, retrouver le polynôme de Goppa permettrait de retrouver les r erreurs insérées. Cette attaque retrouve n/r variables Y et n variables X telles que $Y_i = G(X_i)^{-1}$. Donc si on impose que $r > \frac{n}{r}$, c'est-à-dire $r^2 > n$, on ne peut pas interpoler le polynôme unitaire G à partir des variables Y et X. En conséquence, cette attaque est pour l'instant incomplète. Bien que cette méthode démontre la faiblesse des codes fortement structurés, cette

attaque ne cryptanalyse pas totalement la variante de McEliece basée sur la forme dyadique, pour le moment. De plus, comme énoncé dans [FOPT10a], l'attaque n'est pas réalisable dès que le degré de l'extension m est minoré par 16. Travaillant avec un tel degré d'extension, la taille des clés publiques augmente très légèrement comparée aux paramètres proposés dans [MB09], mais reste toujours drastiquement plus petite que celle de la version générique, ce que montre les tables 3.1, 3.2 et 3.3.

Décodage	m	n	k	r	τ_2	$\log_2(WF)$	taille des clés	gain (%)
unique	11	1792	1088	64		82.518	11968	
liste	11	1728	1024	64	67	82.976	11264	5.88
unique	12	2944	1408	128		116.735	16896	
liste	13	2816	1280	128	134	113.896	15360	9.09
liste	13	7680	1024	512	552	113.084	13312	21.21
unique	12	3200	1664	128		131.235	19968	
liste	12	3072	1536	128	134	129.745	18432	7.69
unique	13	5888	2560	256		205.804	33280	
liste	13	5632	2304	256	269	199.473	29952	10.00
unique	15	11264	3584	512		279.002	53760	
liste	15	10752	3072	512	539	258.223	46080	14.29

Table 3.2 – Comparaison entre la taille de la clé publique de la variante quasi-dyadique de McEliece avec $r^2 > 2^m - r$ en utilisant du décodage unique et du décodage en liste pour des *workfactors* donnés.

Décodage	m	n	k	r	τ_2	$\log_2(WF)$	taille des clés	gain (%)
unique	16	5120	1024	256		81.765	16384	
liste	16	5120	1024	256	270	86.216	16384	0
unique	16	3840	1792	128		113.785	28672	
liste	16	5632	1536	256	269	116.400	24576	14.29
unique	16	5888	1792	256		132.470	28672	
liste	16	9728	1536	512	542	133.534	24576	14.29
unique	16	10752	2560	512		199.067	40960	
liste	16	10752	2560	512	539	209.414	40960	0
unique	16	11776	3584	512		264.846	57344	
liste	16	19456	3072	1024	1085	267.203	49152	14.29

Table 3.3 – Comparaison entre la taille de la clé publique de la variante quasi-dyadique de McEliece avec $m \geq 16$ en utilisant du décodage unique et du décodage en liste pour des *workfactors* donnés.

Dans la table 3.2, on regarde le cas de la variante quasi-dyadique avec l'hypothèse

$r^2 > 2^m - r$, qui est même une condition plus contraignante que $r^2 > n$. En choisissant r comme une puissance de 2, la taille de la clé publique devient alors mk [MB09]. On peut alors atteindre une réduction de clé allant jusqu'à 21%. Les résultats de la variante quasi-dyadique avec l'hypothèse $m \geq 16$ sont présentés dans la table 3.3. On obtient une meilleure réduction que pour la variante générique. En effet, nos expérimentations ont montré une réduction de clé souvent au-dessus de 14%, ou augmente considérablement le *workfactor*. Puisque le nombre d'ensemble de paramètres de codes quasi-dyadiques est assez faible, avec m le degré de l'extension fixé, il est difficile de se fixer un niveau de sécurité et de diminuer la taille des clés. Cependant, lorsque l'on ne peut pas réduire la taille de la clé, un gain conséquent du niveau de sécurité est observé.

La table 3.4 montre les tailles de clés recommandées pour les cryptosystèmes basés sur le problème du logarithme discret sur les corps finis, pour différents niveaux de sécurité [BBB+07, OH04]. Par comparaison, on a également inclu les plus petites tailles de clé obtenues avec les variantes de McEliece, bien qu'en toute impartialité, il faut souligner le manque actuel de perspective pour comparer la sécurité d'une variante du cryptosystème de McEliece récente avec le problème du logarithme discret. Pendant que les tailles des clés du cryptosystème de McEliece restent toujours plus grandes, le ratio entre les deux tailles de clé diminue significativement lorsque le niveau de sécurité augmente. De plus, le coût du chiffrement et du déchiffrement du système de McEliece augmentent plus doucement avec le niveau de sécurité comparé aux cryptosystèmes basés sur le problème du logarithme discret ou sur la factorisation de grands entiers [BS08, MB09].

Niveau de sécurité	Logarithme discret	McEliece	ratio
80	1024	11264	11.0
112	2048	13312	6.5
128	3072	18432	6.0
192	7680	29952	3.9
256	15360	46080	3.0

Table 3.4 – Comparaison des tailles des clés des cryptosystèmes basés sur le logarithme discret sur les corps finis – ou la factorisation d'entiers – et le cryptosystème de McEliece utilisant le décodage en liste.

3.3.3 Conclusion

Grâce à l'étude récente sur le décodage en liste des codes de Goppa binaires – [ABC10] ou section 2.4 – on compare les tailles des clés publiques pour différentes variantes du cryptosystème de McEliece. On montre que l'utilisation de codes qui peuvent être décodables en liste dans le cryptosystème de McEliece, donne un bénéfice non négligeable. On explique également comment rendre la variante quasi-dyadique

sûre contre les attaques connues jusqu'à aujourd'hui, tout en réduisant encore la taille des clés grâce à l'utilisation d'algorithmes de décodage en liste. Par exemple, pour un niveau de sécurité de 2^{80}, le décodage en liste réduit la taille de clé de 631 840 bits pour la variante générique à 11 264 bits pour la variante quasi-dyadique. De plus, contrairement aux précédentes tentatives de réduction de clé de McEliece, l'utilisation du décodage en liste n'introduit aucune nouvelle structure qui pourrait être utilisée pour mener à bien une cryptanalyse.

Chapitre 4

Stéganographie basée sur les codes

Plan du chapitre

4.1 Introduction

Premièrement, on rappelle le contexte de la stéganographie et on pose les premières définitions. Ensuite on présente les problèmes standards de la stéganographie basée sur les codes. On montre que de tels problèmes reviennent à tenter de résoudre des systèmes linéaires qui n'ont pas forcément de solutions. On donne alors des conditions nécessaires et suffisantes pour que les systèmes associés aux problèmes de stéganographie possèdent

toujours au moins une solution [MB11]. Enfin, on présente notre nouveau schéma d'insertion [ABF11], qui suit l'idée du système de signature de Courtois, Finiasz et Sendrier [CFS01], pour lequel les systèmes linéaires associés ont une solution.

4.1.1 Contexte

La stéganographie est la discipline qui a pour objectif principal de cacher, aussi furtivement que possible, une information dans un support. Si Alice envoie à Bob un message stéganographié dans un support et si Ève, une espionne, intercepte le support, on souhaite qu'Ève ne puisse pas se douter de l'existence même d'un message caché dans le support. Contrairement au chiffrement, dont l'objectif est de rendre un message inintelligible, la stéganographie vise à cacher l'existence même du message. Dans un contexte de chiffrement, Ève sait qu'un message est transmis.

Dans un contexte stéganographique, l'expéditeur a à sa disposition le support, appelé le *cover-medium*, le message et une clé d'insertion. Pour insérer le message, on commence par extraire des bits du support dépendants de la clé d'insertion donnant le *cover-data*, il existe différents types d'extraction qui dépendent évidemment du *cover-medium*. Par exemple celle dit des *Least Significant Bits (L.S.B)*, qui n'extrait du support que les bits de petites significations – les bits de poids faibles dans le codage binaire de chaque composante *RGB* – puis on applique à cette suite de bits une permutation donnée par la clé secrète. On modifie le *cover-data* pour obtenir le *stego-data*, grâce à la fonction d'insertion Emb qui prend en entrée un bloc du message, un bloc du *cover-data* et qui retourne un bloc du *stego-data*. On remplace dans le *cover-medium* le *cover-data* par le *stego-data* et on obtient ainsi le *stego-medium*. L'émetteur envoie au destinataire le *stego-medium* et grâce à la clé d'extraction, le destinataire peut lire le message.

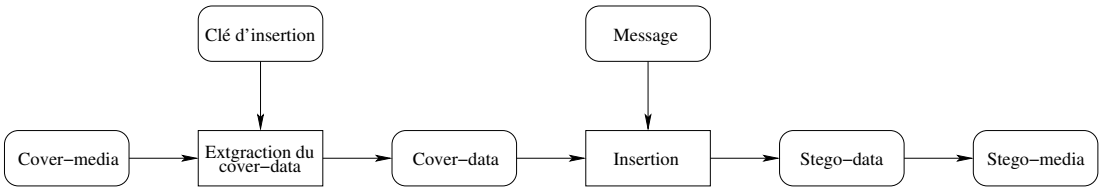

FIGURE 4.1 – Insertion considérée.

La stéganographie et le chiffrement ne partagent pas les mêmes objectifs, cependant elles ne sont pas pour autant antagonistes. Bien au contraire, on suppose souvent, que le message que l'on cache est chiffré. Cela a pour conséquence, non négligeable, que l'on suppose que la suite de bits du message suit une loi de Bernoulli de paramètre 1/2. On peut donc même dire dans ce sens, que le chiffrement sert les intérêts de la stéganographie. Dans le schéma général d'envoi et de réception d'un message stéganographié, on suppose l'existence de deux couples de clés. L'un pour le chiffrement et l'autre pour la stéganographie.

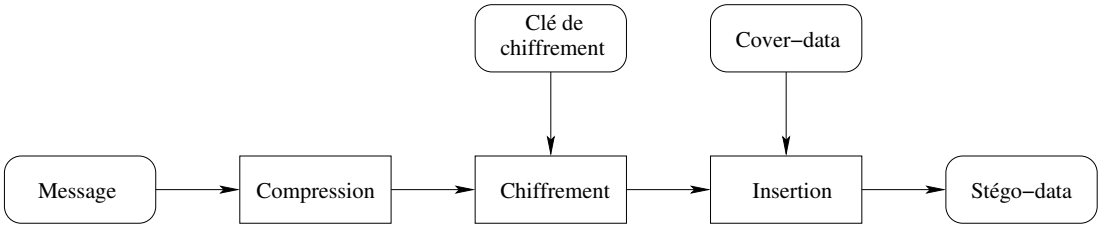

FIGURE 4.2 – Modèle type d'insertion d'un message dans un support

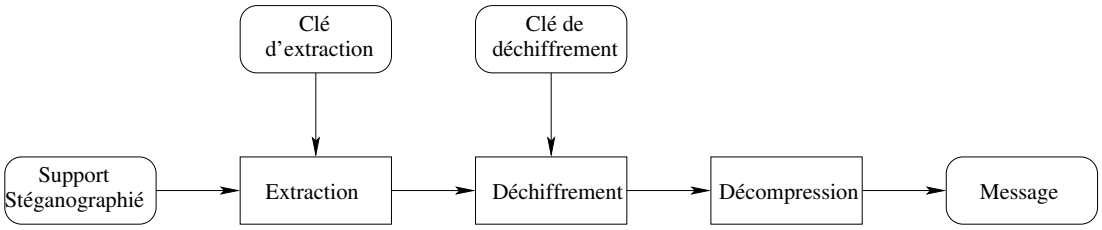

FIGURE 4.3 – Modèle type d'extraction d'un message dans un support stéganographié

Comme les primitives cryptographiques sont mises à l'épreuve des techniques de cryptanalyse, les schémas de stéganographie sont également mis à l'épreuve des techniques de stéganalyse. Ces dernières consistent à déterminer si un document contient ou non, un message caché.

4.1.2 Premières définitions

On présente les principales définitions de la stéganographie.

Définition 4.1 (Schéma stéganographique)**.** Soient $n, r \in \mathbb{N}$ tels que $n \geq r$, $\mathcal{A}$ un alphabet fini, $\mathbf{v} \in \mathcal{A}^n$ le *cover-data* et $\mathbf{m} \in \mathcal{A}^r$ le message. Un *schéma stéganographique* $\mathcal{S}$ est défini par le couple de fonctions : Emb fonction d'insertion et Ext fonction d'extraction telles que

$$Ext(Emb(\mathbf{v}, \mathbf{m})) = \mathbf{m}.$$

Dans la suite, on suppose que les schémas stéganographiques sont binaires, c'est-à-dire que $\mathcal{A} = \mathbb{F}_2$.

On peut construire différents schémas stéganographiques, il est donc important de pouvoir les comparer. Une des mesures les plus importante est l'*embedding efficiency*.

Définition 4.2 (*Embedding efficiency*)**.** Soient $\mathbf{v}$ le *cover-data*, $\mathbf{m}$ le message de r bits et $\mathcal{S}$ un schéma de stéganographie qui modifie $\mathbf{v}$ en au plus T bits. Alors l'*embedding*

efficiency de $\mathcal{S}$ est

$$e \triangleq \frac{r}{T}.$$

Par des raisonnements combinatoires, on obtient une borne supérieure pour l'*embedding efficiency*, mettant en relation l'*embedding efficiency* et le *payload relatif*.

Définition 4.3 (Payload relatif)**.** Soient $\mathbf{v}$ le *cover-data* de n bits, $\mathbf{m}$ le message de r bits et $\mathcal{S}$ un schéma de stéganographie qui modifie $\mathbf{v}$. Alors le *payload relatif* de $\mathcal{S}$ est

$$a \triangleq \frac{r}{n}.$$

À *payload relatif* fixé, un schéma stéganographique sera d'autant meilleur que son *embedding efficiency* sera élevé. Cependant, il existe une borne supérieure sur l'*embedding efficiency* dépendant du *payload* relatif.

Proposition 4.1. *Soient $\mathcal{S}$ un schéma stéganographique q-aire, e l'*embedding efficiency *de $\mathcal{S}$ et a son* payload *relatif. Alors*

$$e \leq \frac{a}{\mathcal{H}_q^{-1}(a)},$$

où $\mathcal{H}_q^{-1}$ est la fonction inverse de la fonction entropie q-aire définie sur l'intervale $[0, \frac{q-1}{q}]$ par $\mathcal{H}_q(x) \triangleq x \log_q(q-1) - x \log_q(x) - (1-x)\log_q(1-x)$.

4.2 Codage par syndrome

En 1998, Crandall a proposé un schéma stéganographique dont la fonction d'insertion Emb repose sur la résolution du problème du codage par syndrome [Cra98]. Cette technique, basée sur la théorie des codes correcteurs d'erreurs, consiste à changer des bits du *cover-data* de telle sorte que le *stego-data* obtenu a pour syndrome le message que l'on souhaite transmettre.

Problème 4.2 (Codage par syndrome)**.** *Soient $\mathbf{v} \in \mathbb{F}_2^n$ le* cover-data*, H une matrice de parité d'un $[n,k]_2$-code linéaire et $\mathbf{m} \in \mathbb{F}_2^{n-k}$ le message que l'on souhaite insérer. On cherche un* stego-data *$\mathbf{y} \in \mathbb{F}_2^n$ tel que*

$$\mathbf{y}H^t = \mathbf{m}.$$

Pour une matrice de parité H, un syndrome $\mathbf{m}$ et un mot de l'espace ambiant $\mathbf{v}$ fixés, on cherche un mot de l'espace ambiant $\mathbf{y}$ tel que son syndrome – par H – soit égale à $\mathbf{m}$. Puisque l'on va remplacer le vecteur $\mathbf{v}$ par $\mathbf{y}$ et que l'on souhaite faire le moins de modifications possibles, alors le vecteur $\mathbf{y}$ doit être le plus proche possible du vecteur $\mathbf{v}$. On définit ainsi les fonctions d'insertion *Emb* et d'extraction *Ext* du schéma stéganographique basé sur ce problème de la manière suivante :

– $Emb(\mathbf{v}, \mathbf{m}) = \mathbf{y} = \mathbf{v} - cl(\mathbf{v}H^t - \mathbf{m})$
– $Ext(\mathbf{y}) = \mathbf{y}H^t$,

où $cl(\mathbf{x})$ est la fonction qui retourne le *coset leader* de $\mathbf{x}$. Pour s'assurer que c'est bien un schéma stéganographique, il suffit de vérifier que $\forall(\mathbf{v}, \mathbf{m}) \in \mathbb{F}_2^n \times \mathbb{F}_2^{n-k}$

$$\begin{aligned} Ext(Emb(\mathbf{v}, \mathbf{m})) &= \left(\mathbf{v} - cl(\mathbf{v}H^t - \mathbf{m})\right) H^t \\ &= \mathbf{v}H^t - cl(\mathbf{v}H^t - \mathbf{m})H^t \\ &= \mathbf{m}. \end{aligned}$$

Dans [Mun07], Munuera montre comment construire un schéma stéganographique à partir des codes linéaires et comment certaines propriétés des codes se répercutent sur les schémas stéganographiques en question. Dans sa proposition originale, Crandall propose les codes de Hamming binaires, car pour toutes les valeurs les *coset leader* sont faciles à calculer. En effet, le rayon de recouvrement de ces codes est toujours égal a 1 et donc les *coset leader* sont toujours de poids 1. Toutefois, en prenant d'autres codes linéaires, ce n'est généralement pas le cas, ce qui implique que pour réussir à calculer la fonction Emb, on a besoin tout d'abord de pré-calculer et de mémoriser la table entière des *coset leader*, cette technique est appelée décodage par tableau standard. Ce qui présente un énorme inconvénient. On préfère ainsi formuler l'insertion comme suit :

$$Emb(\mathbf{v}, \mathbf{m}) = \mathbf{y}' + dec(\mathbf{v} - \mathbf{y}'),$$

où $\mathbf{y}' \in \mathbb{F}_2^n$ tel que $\mathbf{y}'H^t = \mathbf{m}$ et *dec* est une fonction de décodage de $\mathcal{C}$. Calculons maintenant la distance entre le vecteur $\mathbf{v}$ et le vecteur ainsi obtenu $\mathbf{y}$.

$$\begin{aligned} d(\mathbf{v}, \mathbf{y}) &= d(\mathbf{v}, \mathbf{y}' + dec(\mathbf{v} - \mathbf{y}')) \\ &= d(\mathbf{v}, \mathbf{y}' + \mathbf{v} - \mathbf{y}' + \mathbf{e}) \\ &= w(\mathbf{e}), \end{aligned}$$

où le vecteur $\mathbf{e}$ est donné par $dec(\mathbf{v}-\mathbf{y}') = \mathbf{v}-\mathbf{y}'+\mathbf{e}$. Comme la deuxième présentation de la fonction d'insertion *Emb* fait intervenir une fonction de décodage et que le problème du décodage complet est un problème difficile, il faut pouvoir prendre un compte le rayon de décodage de *dec*. C'est dans ce sens que l'on introduit le problème suivant, qui n'est rien d'autre que le problème du codage par syndrome avec une condition supplémentaire sur la distance entre $\mathbf{v}$ le *cover-data* et $\mathbf{y}$ le *stego-data*.

Problème 4.3 (Codage par syndrome borné)**.** *Soient* $\mathbf{v} \in \mathbb{F}_2^n$ *le* cover-data*,* H *une matrice de parité d'un* $[n, k]_2$*-code linéaire,* $T \in \mathbb{N}$ *et* $\mathbf{m} \in \mathbb{F}_2^{n-k}$ *le message que l'on souhaite insérer. On cherche un* stego-data $\mathbf{y} \in \mathbb{F}_2^n$ *tel que*

$$\begin{cases} \mathbf{y}H^t = \mathbf{m} \\ d(\mathbf{v}, \mathbf{y}) \leq T. \end{cases}$$

Dans [GK03], Galand et Kabatiansky montrent le lien entre les schémas stéganographiques et les codes recouvrants : *covering code*. Le fait d'ajouter une

contrainte supplémentaire sur la distance entre $\mathbf{v}$ et $\mathbf{y}$ permet de tenir compte du rayon de décodage de $\mathcal{C}$, ainsi que de contrôler le nombre maximal de modifications. Cependant, si T est strictement plus petit que ρ, le rayon de recouvrement du code $\mathcal{C}$, alors un tel système ne possède pas forcément une solution. Dans sa proposition, Crandall résout le problème du codage par syndrome borné en utilisant les codes de Hamming binaires. Puisque l'on sait décoder les codes de Hamming binaires jusqu'à leur rayon de recouvrement $\rho = 1$, alors en utilisant ces codes, le problème du codage par syndrome borné jusqu'à 1 a toujours une solution. En effet, pour tout message $\mathbf{m} \in \mathbb{F}_2^{n-k}$, on peut toujours modifier un bit du *cover-data* $\mathbf{v} \in \mathbb{F}_2^n$, pour obtenir un *stego-data* $\mathbf{y} \in \mathbb{F}_2^n$ tel que $S_H(\mathbf{y}) = \mathbf{m}$ et $d(\mathbf{v}, \mathbf{y}) \leq 1$. L'*embedding efficiency* est alors

$$e = n - k = m,$$

en utilisant un code de Hamming binaire de longueur $2^m - 1$. Ce qui permet de choisir l'*embedding efficiency*. Un des logiciels de stéganographie utilisé est le logiciel F5, développé par Westfeld. Ce logiciel utilise cette méthode avec les codes de Hamming binaires. Le logiciel est expliqué en détails dans [Wes01].

4.3 Codage par syndrome avec positions verrouillées

En 2005, la notion de *Wet paper* a été introduite par Fridrich, Goljan, Lisonek et Soukal dans [FGLS05]. Selon le *cover-media* et le message, cette notion consiste à interdire la modification de certaines composantes du *cover-data*. C'est-à-dire que le *stego-data* et le *cover-data* ont la même valeur pour ces composantes. Cette technique permet de diminuer la distorsion causée par l'insertion, Par exemple en interdisant la modification de bits dans une zone homogène du *cover-medium* : le ciel bleu dans une image. On obtient ainsi le problème du codage par syndrome avec positions verrouillées qui est basé sur le problème du codage par syndrome en ajoutant une condition supplémentaire, le rendant plus difficile à résoudre. Dans la suite on note $\mathcal{W}$ l'ensemble des positions qui ne doivent pas être modifiées, $\mathcal{D}$ l'ensemble des positions modifiables, $\ell \triangleq |\mathcal{W}|$ et $\delta \triangleq |\mathcal{D}|$. On a donc que $\mathcal{W} = \{1, \ldots, n\} \setminus \mathcal{D}$ et $\ell + \delta = n$. L'ensemble $\mathcal{W}$ est choisi selon le *cover-media* et le message. On obtient alors le problème suivant.

Problème 4.4 (Codage par syndrome avec positions verrouillées)**.** *Soient* $\mathbf{v} \in \mathbb{F}_2^n$ *le* cover-data*, H une matrice de parité d'un* $[n, k]_2$*-code linéaire,* $\mathcal{W} \subset \{1, \ldots, n\}$ *et* $\mathbf{m} \in \mathbb{F}_2^{n-k}$ *le message que l'on souhaite insérer. On cherche le* stego-data $\mathbf{y} \in \mathbb{F}_2^n$ *tel que*

$$\begin{cases} \mathbf{y}H^t = \mathbf{m} \\ \mathbf{v}_i = \mathbf{y}_i, \ \forall i \in \mathcal{W}. \end{cases}$$

De même que précédemment, on peut rajouter au problème de codage par syndrome avec positions verrouillées, une contrainte supplémentaire sur le nombre de positions que l'on s'autorise à modifier.

Problème 4.5 (Codage par syndrome borné avec positions verrouillées)**.** *Soient* $\mathbf{v} \in \mathbb{F}_2^n$ *le cover-data,* H *une matrice de parité d'un* $[n,k]_2$*-code linéaire,* $T \in \mathbb{N}$*,* $\mathcal{W} \subset \{1,\dots,n\}$ *et* $\mathbf{m} \in \mathbb{F}_2^{n-k}$ *le message que l'on souhaite insérer. On cherche le stego-data* $\mathbf{y} \in \mathbb{F}_2^n$ *tel que*

$$\begin{cases} \mathbf{y}H^t = \mathbf{m} \\ d(\mathbf{v},\mathbf{y}) \leq T \\ \mathbf{v}_i = \mathbf{y}_i, \ \forall i \in \mathcal{W}. \end{cases}$$

Puisque le problème 4.3 peut être insoluble, il est évident que l'insertion basé sur le problème 4.5, plus contraint, ne peut qu'avoir une probabilité d'échec supérieure. Le système linéaire posé par ce problème, est donné par une sous-matrice de la matrice de parité du code utilisé. La sous-matrice est définie par rapport à $\mathcal{W}$, l'ensemble des positions verrouillées. On obtient alors le système suivant à résoudre

$$\begin{aligned} \mathbf{y}H^t &= \mathbf{m}, \\ \mathbf{y}_{|\mathcal{D}}H^t_{|\mathcal{D}} + \mathbf{y}_{|\mathcal{W}}H^t_{|\mathcal{W}} &= \mathbf{m}, \\ \mathbf{y}_{|\mathcal{D}}H^t_{|\mathcal{D}} &= \mathbf{m} - \mathbf{y}_{|\mathcal{W}}H^t_{|\mathcal{W}}, \end{aligned}$$

où $\mathbf{y}_{|\mathcal{W}}$ représente la projection de $\mathbf{y}$ sur $\mathcal{W}$ et $H_{|\mathcal{W}}$ représente la sous matrice de H en ne gardant que les colonnes indexées par $\mathcal{W}$. Il est difficile d'exploiter en toute généralité la structure de n'importe quelle sous-matrice, de n'importe quelle matrice de parité et pour n'importe quel code. De plus, dans [FGS06], les auteurs ont montré que les codes aléatoires sont asymptotiquement bons pour l'*embedding efficiency* comparés à la borne donnée dans la proposition 4.1. C'est pour ces deux raisons que l'on propose de regarder le rang d'une matrice aléatoire comme indicateur pour étudier la résolubilité du système. Le théorème suivant fournit une minoration de la probabilité qu'une matrice aléatoire soit de rang plein.

Théorème 4.6. *Soit* M *une matrice aléatoire de* a *colonnes et* b *lignes à coefficients dans* $\mathbb{F}_q$ *tels que* $a \geq b$*. Alors*

$$\mathbb{P}\left(rang(M) = b\right) \geq \begin{cases} 0.288, & si\ a = b\ et\ q = 2, \\ 1 - \frac{1}{q^{a-b}(q-1)}, & sinon. \end{cases}$$

Démonstration. Voir [BGL01]. □

Dans cette thèse, on a beaucoup regardé le problème de résolubilité de ce système. Que faire lorsque l'insertion est impossible ? Différentes propositions ont été faites, dont celle de changer de *cover-medium* autant de fois qu'il le faut pour que le message s'insère. En effet, si on change le *cover-medium*, il est clair que le *cover-data* est aussi changé. On espère ainsi que le nouveau problème possède une solution. Ce procédé présuppose d'être en possession d'une grande base de données de supports durant l'insertion, qui peut-être une hypothèse plus ou moins forte selon le contexte dans lequel on se place. Une autre idée est de relâcher les conditions, par exemple en effectuant plus de modifications qu'autorisées initialement, ou encore en modifiant des positions verrouillées. Dans les

deux cas on augmente la probabilité de détection de l'attaquant et on affaibli donc le système. Cette question est donc primordiale, d'autant que la probabilité que l'insertion échoue est non négligeable. Or, elle a pour l'instant peu retenue l'attention et c'est pourquoi elle fait un des objets d'études principaux de cette thèse.

4.4 *Wet paper* et la distance duale

Dans cette section, on a considéré le problème du codage par syndrome avec des positions verrouillées, problème 4.4. On exhibe des conditions nécessaires et suffisantes pour que ce problème ait toujours une solution. La sous-section 4.4.1 est dédiée au cas particulier des codes linéaires. On calcule ensuite l'*overhead* moyen – la différence entre le nombre maximal et le nombre moyen de positions verrouillées – pour ces codes dans la sous-section 4.4.2. On remarque dans la sous-section 4.4.3 que le défaut de Singleton joue un rôle important et on généralise les résultats obtenus dans la première sous-section grâce à la hiérarchie des poids. Les résultats obtenus sont de nature combinatoire, c'est pourquoi on a réussi à généraliser nos résultats aux codes systématiques non-linéaires en sous-section 4.4.4.

Cette section à fait l'objet d'une publication qui va apparaître dans le journal *Advances in Mathematics of Computations* [MB11].

4.4.1 Condition nécessaire et suffisante pour l'existence de solutions

On commence par énoncer le problème du codage par syndrome avec positions verrouillées – problème 4.4. Soient $\mathcal{C}$ un $[n, n-r]_q$-code linéaire de matrice de parité H, $\mathbf{v} \in \mathbb{F}_q^n$, $\mathbf{m} \in \mathbb{F}_q^r$ et $\mathcal{W} \subset \{1, \ldots, n\}$. On cherche alors le vecteur $\mathbf{y} \in \mathbb{F}_q^n$ tel que

$$[S] : \begin{cases} \mathbf{y}H^T & = \mathbf{m}, \\ \mathbf{y}_i & = \mathbf{v}_i, \quad \text{si } i \in \mathcal{W}. \end{cases}$$

On pose $\mathcal{D} \triangleq \{1, \ldots, n\} \setminus \mathcal{W}$ tel que $|\mathcal{D}| = \delta \geq r$, la co-dimension de $\mathcal{C}$ – la dimension du code dual de $\mathcal{C}$. Soit G une matrice génératrice de $\mathcal{C}$. On note $\mathcal{C}^\perp$ le code dual de $\mathcal{C}$. Le code $\mathcal{C}^\perp$ a donc H pour une matrice génératrice et G pour matrice de parité. On note $d^\perp$, la distance duale de $\mathcal{C}$, qui est la distance minimale de $\mathcal{C}^\perp$. Pour $\mathbf{m} \in \mathbb{F}_2^r$, on note $\mathrm{cl}(\mathbf{m})$ le *coset leader* de $\{\mathbf{y} \in \mathbb{F}_2^n : \mathbf{y}H^T = \mathbf{m}\}$. Comme $\mathbf{y}H^T$ est un syndrome, le système $[S]$ a une solution s'il existe $\mathbf{y} \in \mathrm{cl}(\mathbf{m}) + \mathcal{C}$ tel que $\pi_{\mathcal{W}}(\mathbf{y}) = \pi_{\mathcal{W}}(\mathbf{v})$, où $\pi_{\mathcal{W}}$ est la projection sur les coordonnées de $\mathcal{W}$. De manière équivalente, le système $[S]$ a une solution si et seulement si $\pi_{\mathcal{W}}(\mathbf{v}) \in \pi_{\mathcal{W}}(\mathrm{cl}(\mathbf{m}) + \mathcal{C})$. Soit M une matrice avec n colonnes, on note alors $M_{|\mathcal{W}}$ la matrice obtenue en enlevant les colonnes de M indicées par $\mathcal{D}$, c'est-à-dire en ne gardant que les colonnes de $\mathcal{W}$.

Lemme 4.7 (Munuera, B. - 2011)**.** *Soient $\mathcal{C}$ un $[n, n-r]_q$-code linéaire, $\mathcal{W}$ un sous ensemble de $\{1, \ldots, n\}$ et G une matrice génératrice de $\mathcal{C}$. Alors*

$$\forall \mathbf{x} \in \mathbb{F}_q^n, \ |\pi_{\mathcal{W}}(\mathbf{x} + \mathcal{C})| = q^{rang(G_{|\mathcal{W}})}.$$

Démonstration. On a

$$\begin{aligned} |\pi_{\mathcal{W}}(\mathbf{x}+\mathcal{C})| &= |\pi_{\mathcal{W}}(\mathbf{x})+\pi_{\mathcal{W}}(\mathcal{C})| \\ &= |\pi_{\mathcal{W}}(\mathcal{C})|. \end{aligned}$$

Comme $\pi_{\mathcal{W}}(\mathcal{C})$ est un espace vectoriel de dimension $\mathrm{rang}(G_{|\mathcal{W}})$, alors

$$|\pi_{\mathcal{W}}(\mathbf{x}+\mathcal{C})| = q^{\mathrm{rang}(\mathrm{G}_{|\mathcal{W}})}.$$

□

Corollaire 4.8 (Munuera, B. - 2011)**.** *Soient $\mathcal{C}$ un $[n, n-r]_q$-code linéaire, $\mathcal{W} \subset \{1,\ldots,n\}$ et G une matrice génératrice. Alors le système $[S]$ a une solution pour n'importe quels vecteurs $\mathbf{v} \in \mathbb{F}_q^n$ et $\mathbf{m} \in \mathbb{F}_q^r$ si et seulement si la matrice $G_{|\mathcal{W}}$ est de rang plein, soit $rang(G_{|\mathcal{W}}) = \ell$.*

Démonstration. Pour n'importe quels vecteurs fixés $\mathbf{v}$ et $\mathbf{m}$, le système $[S]$ a une solution si et seulement si $\pi_{\mathcal{W}}(\mathbf{v}) \in \pi_{\mathcal{W}}(\mathrm{cl}(\mathbf{m})+\mathcal{C})$. Pour tous les vecteurs $\mathbf{v}$ et $\mathbf{m}$ le système $[S]$ a une solution si et seulement si

$$|\pi_{\mathcal{W}}(\mathbb{F}_q^n)| = |\pi_{\mathcal{W}}(\mathrm{cl}(\mathbf{m})+\mathcal{C})| \Longleftrightarrow \mathrm{rank}(G_{|\mathcal{W}}) = \ell.$$

C'est-à-dire $G_{|\mathcal{W}}$ soit de rang plein. □

Il est donc intéressant de trouver des conditions pour que la matrice $G_{|\mathcal{W}}$ soit de rang plein. C'est ce que l'on propose dans le lemme suivant.

Lemme 4.9 (Munuera, B. - 2011)**.** *Soient $\mathcal{C}$ un $[n, n-r]_q$-code linéaire, $\mathcal{W} \subset \{1,\ldots,n\}$ et G une matrice génératrice de $\mathcal{C}$. Alors $G_{|\mathcal{W}}$ est de rang plein si et seulement si il n'y a pas de mot du code dual $\mathcal{C}^{\perp}$ dont le support est inclus dans $\mathcal{W}$.*

Démonstration. Soient $c \in \mathcal{C}^{\perp} \setminus \{\mathbf{0}\}$ et $g_1,\ldots,g_n$ les colonnes de G. Comme G est une matrice de parité de $\mathcal{C}^{\perp}$, on obtient

$$\begin{aligned} cG^t &= 0 \\ &= \sum_{i=1}^{n} c_i g_i \\ &= \sum_{j \in Supp(c)} c_j g_j. \end{aligned}$$

Si on suppose maintenant que $Supp(c) \subset \mathcal{W}$ alors

$$\begin{aligned} cG^t &= cG^t_{|\mathcal{W}} \\ &= 0. \end{aligned}$$

De cette manière, il existe une combinaison linéaire des colonnes de $G_{|\mathcal{W}}$ qui est égale à zéro et donc $G_{|\mathcal{W}}$ ne peut pas être de rang plein.

De plus, ce raisonnement peut être inversé pour montrer l'équivalence. Effectivement, si $G_{|\mathcal{W}}$ n'est pas de rang plein, alors il existe une combinaison linéaire des colonnes de $G_{|\mathcal{W}}$ qui est égale à zéro. On en déduit que les coefficients de cette relation sont un mot du code dual $\mathcal{C}^{\perp}$ dont son support est contenu dans $\mathcal{W}$. □

Plus généralement, s'il existe w mots indépendants de $\mathcal{C}^{\perp}$ dont le support est dans $\mathcal{W}$, alors on obtient $\text{rank}(G_{\mathcal{W}}) = \ell - w$. Ce qui suggère que l'énumération des poids de $\mathcal{C}$ joue également un rôle dans l'étude de solvabilité du système $[S]$. Cette étude est menée dans la sous-section 4.4.3.

Théorème 4.10 (Munuera, B. - 2011)**.** *Soit $\mathcal{C}$ un $[n, n-r]_q$-code linéaire. Alors le système $[S]$ a une solution pour n'importe quels $\mathbf{v} \in \mathbb{F}_q^n, \mathbf{m} \in \mathbb{F}_q^r$ et $\mathcal{W} \subset \{1, \ldots, n\}$ avec $|\mathcal{W}| = n - \delta$, si et seulement si $\delta \geq n - d^{\perp} + 1$. De plus, il y a exactement $q^{\delta - r}$ solutions.*

Démonstration. Si $\delta \geq n - d^{\perp} + 1$ alors

$$\begin{aligned} |\mathcal{W}| &= n - \delta \\ &\leq n - n + d^{\perp} - 1 \\ &< d^{\perp}. \end{aligned}$$

En utilisant la proposition 1.9 – page 9, on en déduit qu'il ne peut pas exister c' un mot non nul du code $\mathcal{C}^{\perp}$ tel que $Supp(c') \subset \mathcal{W}$.

Soit G une matrice de parité de $\mathcal{C}^{\perp}$. Si on prend c' un mot du code $\mathcal{C}^{\perp}$ et un ensemble $\mathcal{W}$ de cardinal $n - \delta$ contenant le support de c', alors $G_{|\mathcal{W}}$ ne peut pas être de rang plein, et il existe au moins un couple $(\mathbf{v}, \mathbf{m}) \in \mathbb{F}_q^n \times \mathbb{F}_q^r$ tel que le système $[S]$ ne possède pas de solution.

Quand $\text{rang}(G_{\mathcal{W}}) = n - \delta$, alors le nombre de solutions est $|\mathcal{C}|/|\pi_{\mathcal{W}}(\mathcal{C})| = q^{\delta - r}$. □

La méthode de l'insertion du codage par syndrome avec un code $\mathcal{C}$ de paramètres $[n, n-r]_q$, peut gérer au plus $d^{\perp} - 1$ positions verrouillées pour insérer r symboles d'information, théorème 4.10. On définit τ le *seuil de positions verrouillées* comme le nombre de maximal de positions verrouillées pour que le système $[S]$ ait toujours une solution. D'après le théorème 4.10, le seuil pour $\mathcal{C}$ est $\tau = n - d^{\perp} + 1$. On remarque que le *strict overhead*, le nombre supplémentaire de positions modifiables est $\tau - r = n - r + 1 - d^{\perp}$ n'est rien d'autre que le défaut de Singleton de $\mathcal{C}^{\perp}$. Le lien avec la hiérarchie des poids se fait encore sentir.

Exemple 4.1. Soit un code de Hamming binaire de redondance s et de longueur $n = 2^s - 1$. La distance duale de ce code est $d^{\perp} = 2^{s-1}$. On peut alors insérer s bits d'information dans le support de longueur n avec $2^{s-1} \approx n/2$ positions modifiables. Pour voir à quel point on ne peut pas avoir plus de positions verrouillées tout en ayant une certitude que le système $[S]$ ait une solution, on considère une matrice de parité

dont toutes les colonnes sont la représentation binaire des entiers $1, \ldots, 2^s - 1$. En enlevant par exemple les 2^{s-1} dernières colonnes de H, on obtient que la dernière ligne de H est la ligne 0. Ceci donne exactement le nombre maximal de positions verrouillées, que peut gérer l'insertion basée sur le problème du codage par syndrome avec positions verrouillées en utilisant un code de Hamming binaire. Dans la section 4.5, on présente une nouvelle méthode d'insertion qui atteint exactement cette borne pour ces codes.

Exemple 4.2. Il est clair que plus le défaut de Singleton est petit, meilleurs sont les paramètres. Mais en général, il n'est pas simple de construire des codes avec des défauts de Singleton bornés. Cependant, les codes géométriques construits sur une courbe algébrique et deux diviseurs rationnels le permettent [Mun94]. Il est bien connu que le défaut de Singleton d'un code géométrique provenant d'une courbe, est borné par le genre de la courbe. Il est alors possible de construire un schéma stéganographique avec un *overhead* aussi petit que l'on souhaite.

4.4.2 Calcul du *overhead*

Une autre tâche importante est de calculer le *overhead* moyen, c'est-à-dire $\tilde{m} = \delta - r$ pour une solution en supposant $\mathcal{C}, \mathcal{W}, \mathbf{c}$ et $\mathbf{m}$ aléatoires – en gardant les notations précédentes. On se place maintenant dans le cas des codes linéaires binaires. On obtient alors une estimation de la probabilité d'avoir une solution. On note $\mathrm{avrank}(t, s)$ le rang moyen d'un matrice aléatoire M de taille $t \times s$.

Proposition 4.11 (Munuera, B. - 2011)**.** *Pour $\mathcal{C}, \mathcal{W}, \mathbf{c}$ et $\mathbf{m}$ aléatoires, la probabilité que δ symboles modifiables soient suffisants pour transmettre $r \leq \delta$ symboles du message est*

$$p \geq 2^{\mathrm{avrank}(n-r, n-\delta)-(n-\delta)}.$$

Démonstration. D'après la Section 4.4.1, étant donné $\mathcal{C}, \mathcal{W}, \mathbf{c}$ et $\mathbf{m}$ comme précédemment, la probabilité que le système $[S]$ possède une solution est

$$\mathrm{prob}\left(\pi_{\mathcal{W}}(\mathbf{c}) \in \pi_{\mathcal{W}}(\mathrm{cl}(\mathbf{m}) + \mathcal{C})\right) = \frac{|\pi_{\mathcal{W}}(\mathcal{C})|}{2^{n-\delta}}.$$

Alors

$$p = \frac{E[|\pi_{\mathcal{W}}(\mathcal{C})|]}{2^{n-\delta}}$$

où $\mathcal{W}$ et $\mathcal{C}$ sont pris uniformément aléatoire et $E[|\pi_{\mathcal{W}}(\mathcal{C})|]$ est la variable aléatoire correspondant à $|\pi_{\mathcal{W}}(\mathcal{C})|$. On rappel, d'après la borne de Jensen, si X est une variable aléatoire et ϕ est une fonction convexe, on a $\phi(E[X]) \leq E[\phi(X)]$, voir [Ham94], Section 2.5. Comme $\#\pi_{\mathcal{W}}(\mathcal{C}) = 2^{\mathrm{rank}(G_{\mathcal{W}})}$ et la fonction exponentielle est convexe, on a

$$p = \frac{E[2^{\mathrm{rank}(G_{\mathcal{W}})}]}{2^{n-\delta}} \geq \frac{2^{\mathrm{avrank}(G_{\mathcal{W}})}}{2^{n-\delta}}.$$

□

On a donc maintenant besoin d'outils pour calculer le rang moyen d'une matrice aléatoire. Le théorème 4.12 nous permet d'en avoir une estimation. La propriété des rangs a déjà été étudiée en théorie de l'information [SB10]. Comme montré dans [FGS05, FGS06], ces résultats nous permettent de donner une estimation du *overhead* moyen. Comme $G_{|\mathcal{W}}$ est une $(n-r)\times(n-\delta)$-matrice et $(n-r)-(n-\delta)=\delta-r$, alors $\tilde{m}$ peut être vu comme le nombre minimum de lignes supplémentaires afin que la matrice soit de rang plein. Soient t, m deux entiers non négatifs et $M_{t+m,t}$ une $(t+m)\times t$-matrice aléatoire avec $m \geq 0$.

Théorème 4.12. *Soit $M_{t+m,t}$ une matrice aléatoire dont tous ses éléments sont dans $\mathbb{F}_2$ et suivent une loi uniforme. Alors*

$$\lim_{t\to\infty} \operatorname{prob}\left(\operatorname{rank}(M_{t+m,t}) = t-s\right) = \prod_{j=s+m+1}^{\infty}\left(1-2^{-j}\right)/2^{s(s+m)}\prod_{j=1}^{s}\left(1-2^{-j}\right).$$

En particulier

$$\lim_{t\to\infty} \operatorname{prob}\left(\operatorname{rank}(M_{t+m,t}) = t\right) = \prod_{j=m+1}^{\infty}\left(1-2^{-j}\right).$$

Démonstration. Voir [Coo00, SB10]. □

Grâce à ce théorème on obtient une estimation numérique de la fonction avrank, et donc de la probabilité que le système $[S]$ admette une solution pour $\mathcal{C}, \mathcal{W}, \mathbf{c}$ et $\mathbf{m}$ aléatoires. Plus de détails peuvent être trouvés sur ces estimations dans [SW06]. Le théorème 4.12 nous permet de calculer le nombre de lignes supplémentaires pour avoir une matrice de rang plein [SB10]. Pour m positif on définit

$$Q_m = \prod_{j=m+1}^{\infty}\left(1-2^{-j}\right).$$

Alors la probabilité qu'il y a besoin exactement m lignes supplémentaires pour obtenir une $(t+m)\times t$-matrice aléatoire de rang plein est $Q_m - Q_{m-1}$. Comme $Q_{m-1} = ((2^m-1)/2^m)Q_m$ on a $Q_m - Q_{m-1} = Q_m/2^m$, et donc la moyenne du nombre de lignes supplémentaires est

$$\tilde{m} = \sum_{m=1}^{\infty} m(Q_m - Q_{m-1}) = \sum_{m=1}^{\infty}\frac{m}{2^m}Q_m.$$

Comme cette série est majorée par une série arithmético-géométrique convergente, la série précédente est également convergente. En utilisant la formule $\sum_{m=1}^{\infty} mx^m = x/(1-x)^2$ avec $|x|<1$, on obtient :

$$\tilde{m} = \sum_{m=1}^{\infty}\frac{m}{2^m}Q_m < \sum_{m=1}^{\infty}\frac{m}{2^m} = 2.$$

Un calcul direct montre que $\tilde{m} \approx 1.6067$. Pour conclure, le *overhead* moyen est de 1.6067 et, pour n assez grand, δ bits modifiables sont suffisants pour transmettre $r \approx \delta - 1.6067$ bits d'information.

4.4.3 Résolution du système et les poids de Hamming généralisés

Soient $\mathcal{C}$ un $[n, n-r]_2$-code linéaire et $\mathcal{C}^\perp$ le dual de $\mathcal{C}$. Comme précédemment, on note $d^\perp$ la distance duale de $\mathcal{C}$. La distance duale peut être exprimée en fonction de la hiérarchie des poids de $\mathcal{C}$.

Proposition 4.13 (Munuera, B. - 2011)**.** *Soient $\mathcal{C}$ un $[n, n-r]_2$-code linéaire, g le rang MDS de $\mathcal{C}$ et $\delta \geq g + r - 1$. Alors pour tout $\mathbf{v} \in \mathbb{F}_2^n, \mathbf{m} \in \mathbb{F}_2^r$ et $\mathcal{W} \subset \{1, \ldots, n\}$ avec $|\mathcal{W}| = n - \delta$, le système $[S]$ associé a une solution.*

Démonstration. Grâce à la propriété de dualité du rang MDS de $\mathcal{C}$, on déduit

$$\begin{aligned} g - 1 &= n - r - d^\perp + 1 \\ g &= n - r - d^\perp + 2. \end{aligned}$$

La proposition 4.10 nous permet de conclure. □

Cette proposition nous mène à considérer les codes de faible rang MDS. Les codes MDS ont été proposés comme de bons candidats pour le codage par syndrome avec positions verrouillées, par exemple les codes de Reed-Solomon [FG09]. Le principal inconvénient des codes de Reed-Solomon est que la longueur doit être inférieure au nombre d'éléments du corps. Comme les codes MDS sont rares, il est raisonnable de considérer alors des codes de rang MDS supérieur à un. Dans ce sens, les codes géométriques – voir exemple 4.2 et [Mun94] – peuvent constituer une bonne alternative. La proposition suivante permet de généraliser la proposition 4.10 grâce aux poids de Hamming généralisés.

Proposition 4.14 (Munuera, B. - 2011)**.** *Soient $\mathcal{C}$ un $[n, n-r]$-code linéaire, G une matrice génératrice de $\mathcal{C}$ et $\mathcal{W} \subset \{1, \ldots, n\}$ avec $|\mathcal{W}| = n - \delta$. Pour tout g tel que $n - r \geq g \geq \delta - r$ et $d_g > \delta \geq r$, on a rank$(G_{\mathcal{W}}) \geq n - \delta - g + 1$.*

Démonstration. Soit $\mathcal{C}_{\mathcal{W}}^\perp$ le code ayant pour matrice de parité $G_{|\mathcal{W}}$. On a alors $\mathrm{rank}(G_{|\mathcal{W}}) = n - \delta - \dim(\mathcal{C}_{\mathcal{W}}^\perp)$.

On propose maintenant de majorer la dimension de $\mathcal{C}_{\mathcal{W}}^\perp$. Comme on a supposé $d_g > \delta$, on a $n - d_g + 1 < n - \delta + 1$. De plus, par les propriétés de monotonie et de dualité, il y a au plus $(g-1)$ valeurs de d_i tels que $n - d_i + 1 \geq n - \delta + 1$ pour $i \leq g - 1$. Par recouvrement de l'intervalle $[n - \delta + 1, n]$ qui possède δ éléments, il y a donc au moins $\delta - g + 1$ poids de la hiérarchie de $\mathcal{C}^\perp$ qui sont dans $[n - \delta + 1, n]$. Par la propriété de monotonie de la hiérarchie des poids, on déduit que $d_{r-\delta+g}^\perp \geq n - \delta + 1$. Or, on a supposé que $r - \delta \leq 0$, et donc en appliquant encore la propriété de la monotonie de la hiérarchie des poids, on obtient :

$$\begin{aligned} d_g^\perp &\geq d_{r-\delta+g}^\perp \\ &\geq n - \delta + 1. \end{aligned}$$

Mais comme la longueur de $\mathcal{C}_{\mathcal{W}}^{\perp}$ est $n-\delta$ alors on en déduit que $dim(C_{\mathcal{W}}^{\perp}) \leq g-1$. On en conclut que $\text{rank}(G_{|\mathcal{W}}) \geq n-\delta-g+1$. □

4.4.4 Une généralisation aux codes systématiques

Dans cette section, on étend le codage par syndrome avec positions verrouillées à la large famille des codes systématiques. On montre que les schémas de stéganographie basés sur de tels codes se comportent essentiellement de la même manière que pour les codes linéaires. L'utilisation des codes systématiques a déjà été suggérée par Brierbauer et Fridrich dans [BF08], où les schémas stéganographiques basés sur les codes systématiques de Nordstrom-Robinson sont traités. Ici on propose une étude plus fine et plus générale en montrant le lien entre des schémas de stéganographie basés sur les codes systématiques, les fonctions résilientes ainsi que les tableaux orthogonaux. On accorde une attention particulière à l'analogie de la proposition 4.10, ce qui montre sa nature combinatoire.

Schémas stéganographiques basés sur les codes systématiques

Une petite introduction aux codes systématiques a été faite dans la section 1.6. Dans la suite de cette sous-section on appelle dec une fonction de décodage qui résout le problème du décodage complet, problème 1.48 page 28. Soit $\mathcal{C} \subset \mathbb{F}_2^n$, la fonction de décodage est définie par

$$\begin{array}{rcl} \text{dec} : \mathbb{F}_2^n & \longrightarrow & \mathcal{C} \\ \mathbf{x} & \longmapsto & \text{dec}(\mathbf{x}), \end{array}$$

tel que $d(\mathbf{x}, \text{dec}(\mathbf{x})) = d(\mathbf{x}, \mathcal{C})$. C'est-à-dire que la fonction de décodage retourne le mot du code le plus proche de $\mathbf{x}$. S'il n'y a pas unicité, alors il en retourne un au hasard.

Proposition 4.15. *Soient $\mathcal{C} \subset \mathbb{F}_2^n$ un code systématique,* dec *une fonction de décodage de $\mathcal{C}$ et $\mathbf{z} \in \mathbb{F}_2^n$ un élément de l'espace ambiant. Alors*

$$\begin{array}{rcl} \text{dec}_{\mathbf{z}} : \mathbb{F}_2^n & \longrightarrow & \mathbf{z}+\mathcal{C} \\ \mathbf{x} & \longmapsto & \mathbf{z}+\text{dec}(\mathbf{x}-\mathbf{z}), \end{array}$$

est une fonction de décodage pour le code $\mathbf{z}+\mathcal{C}$.

Démonstration. Si on suppose que la fonction de décodage $\text{dec}_{\mathbf{z}}$ n'est pas une fonction de décodage du code $\mathbf{z}+\mathcal{C}$ alors il existe $\mathbf{x} \in \mathbb{F}_2^n$ tel que

$$\begin{array}{rcl} d(\mathbf{x}, \text{dec}_{\mathbf{z}}(\mathbf{x})) & > & d(\mathbf{x}, \mathbf{z}+\mathcal{C}) \\ d(\mathbf{x}-\mathbf{z}, \text{dec}(\mathbf{x}-\mathbf{z})) & > & d(\mathbf{x}-\mathbf{z}, \mathcal{C}). \end{array}$$

On en déduit alors que dec ne peut pas être une fonction de décodage pour le code $\mathcal{C}$. □

Soient $\mathcal{C}$ un $(n, n-r)$-code systématique, σ la fonction génératrice de $\mathcal{C}$ et dec une fonction de décodage pour $\mathcal{C}$. Grâce aux propositions 1.45 et 4.15, il est facile d'en déduire un schéma stéganographique $\mathcal{S} = \mathcal{S}(\mathcal{C})$. Soient $\mathbf{v} \in \mathbb{F}_2^n$ le *cover-data* et $\mathbf{m} \in \mathbb{F}_2^r$ le message

que l'on souhaite cacher dans $\mathbf{v}$. Le schéma stéganographique est défini par les fonctions d'insertion et d'extraction suivantes

$$\begin{array}{rcl} \mathrm{Emb} : \mathbb{F}_2^n \times \mathbb{F}_2^r & \longrightarrow & \mathbb{F}_2^n \\ (\mathbf{v}, \mathbf{m}) & \longmapsto & \mathrm{dec}_{(\mathbf{0},\mathbf{m})}(\mathbf{v}) = (\mathbf{0}, \mathbf{m}) + \mathrm{dec}(\mathbf{v} - (\mathbf{0}, \mathbf{m})) \end{array}$$

$$\begin{array}{rcl} \mathrm{Ext} : \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2^r \\ \mathbf{y} & \longmapsto & S_\sigma(\mathbf{y}). \end{array}$$

En effet, pour tout couple $(\mathbf{v}, \mathbf{m}) \in \mathbb{F}_2^n \times \mathbb{F}_2^r$, on a

$$\begin{array}{rcl} \mathrm{Ext}\,(\mathrm{Emb}(\mathbf{v}, \mathbf{m})) & = & S_\sigma(\mathrm{dec}_{(\mathbf{0},\mathbf{m})}(\mathbf{v})) \\ & = & \mathbf{m}. \end{array}$$

Pour faire l'analogie avec les codes linéaires, la fonction d'insertion est $\mathrm{Emb}(\mathbf{v}, \mathbf{m}) = \mathbf{v} - \mathrm{cl}(\mathbf{v}H^T - \mathbf{m})$. On remarque que pour réussir cette insertion, il est nécessaire d'avoir une table qui fasse correspondre pour chaque syndrome son *coset leader*, ce qui impose de faire du décodage par tableau standard. Alors que la formulation précédente permet d'utiliser la fonction de décodage souhaitée.

On propose maintenant de regarder les paramètres du schéma stéganographique engendré par un $(n, n-r)_2$-code systématique. La longueur du *cover-data* est n, pour insérer un message de longueur r. Pour calculer le rayon d'insertion et le nombre moyen de changements par insertion, on a besoin de rappeler quelques concepts de la théorie des codes.

Proposition 4.16 (Munuera, B. - 2011)**.** *Soient $\mathcal{C}$ un $(n, n-r)$-code systématique et $\mathcal{S}$ le schéma stéganographique basé sur $\mathcal{C}$. Alors le nombre maximal de positions que peut modifier $\mathcal{S}$ est le rayon de recouvrement de $\mathcal{C}$ utilisé et le nombre moyen de changements $R_a(\mathcal{S})$ est le rayon de recouvrement moyen de $\mathcal{C}$.*

Démonstration. Soient $\mathbf{v}$ le *cover-data* et $\mathbf{m}$ le message, par la définition de la fonction d'insertion Emb, on obtient

$$\begin{array}{rcl} d(\mathbf{v}, \mathrm{Emb}(\mathbf{v}, \mathbf{m})) & = & d(\mathbf{v}, (\mathbf{0}, \mathbf{m}) + \mathrm{dec}(\mathbf{v} - (\mathbf{0}, \mathbf{m}))) \\ & = & d(\mathbf{v} - (\mathbf{0}, \mathbf{m}), \mathcal{C}). \end{array}$$

□

Exemple 4.3. On reprend l'exemple 1.18, avec le code 2-correcteur de Nadler, la distribution des poids des *coset leader* est $\alpha_0 = 1$, $\alpha_1 = 12$, $\alpha_2 = 66$, $\alpha_3 = 46$, $\alpha_4 = 3$ et $\alpha_i = 0$ pour $i = 5, \ldots, 12$. Alors le rayon de recouvrement moyen est $\tilde{\rho}(\mathcal{N}) = 2.296875$. Le schéma stéganographique basé sur ce code permet d'insérer 7 bits de message dans un vecteur donné de longueur 12, en changeant au pire 4 positions, et 2.296875 en moyenne.

Schémas stéganographiques, fonctions résilientes et tableaux orthogonaux

Les codes systématiques nous permettent de faire une connexion entre les schémas stéganographiques et deux outils d'importance en mathématiques discrètes : les fonctions résilientes et les tableaux orthogonaux. On commence par leur définitions.

Définition 4.4 (Fonction t-résiliente)**.** Soient $f : \mathbb{F}_2^n \to \mathbb{F}_2^r$ une fonction et $t < n$ un entier. La fonction f est *t-résiliente* si pour tout ensemble $\mathcal{T} \subset \{1, \ldots, n\}$ tel que $|\mathcal{T}| = t$ et pour tout $\mathbf{t} \in \mathbb{F}_2^t$, toutes les sorties possibles de $f(\mathbf{x})$ avec $\pi_{\mathcal{T}}(\mathbf{x}) = \mathbf{t}$ sont équiprobables. En d'autre termes, si $\forall \mathcal{T} \subset \{1, \ldots, n\}$ tel que $|\mathcal{T}| = t,\ \mathbf{y} \in \mathbb{F}_2^r$ on a

$$\mathbb{P}(f(\mathbf{x}) = \mathbf{y} \mid \pi_{\mathcal{T}}(\mathbf{x}) = \mathbf{t}) = \frac{1}{2^r}.$$

Les fonctions résilientes jouent un rôle important en cryptographie et sont fortement liées aux tableaux orthogonaux [Sti93].

Définition 4.5 (Tableau orthogonal)**.** Soient λ, t, n trois entiers. Un *tableau orthogonal*, noté $OA_\lambda(t, n)$ est une $\lambda 2^t \times n$ matrice sur $\mathbb{F}_2$, tel que dans n'importe quel ensemble de t colonnes, chaque 2^t vecteurs de $\mathbb{F}_2^t$ est exactement dans λ lignes.

Définition 4.6 (Grand ensemble de tableaux orthogonaux)**.** Un *grand ensemble de tableaux orthogonaux* est un ensemble de $2^{n-t}/\lambda$ tableaux orthogonaux $OA_\lambda(t, n)$ tels que tout vecteur de $\mathbb{F}_2^n$ est présent une fois comme une ligne de l'un des OA de l'ensemble.

En voyant les lignes de ces tableaux comme des vecteurs de $\mathbb{F}_2^n$, un grand ensemble de tableaux orthogonaux donne une partition de $\mathbb{F}_2^n$ [Sti93].

Il y a une forte connexion entre les tableaux orthogonaux et les codes correcteurs [MS77, Chap. 5, Sec. 5]. En effet, soit $\mathcal{C}$ un $[n, n-r]$-code linéaire. Alors – en accord avec la proposition 4.10 – le tableau ayant tous les mots du code $\mathcal{C}$ comme lignes est un $OA_{2^{n-r-d^\perp+1}}(d^\perp - 1, n)$. Delsarte a observé un résultat similaire pour les codes non-linéaires. Pour autant, si le code $\mathcal{C}$ n'est pas linéaire, alors le code dual n'existe pas. Cependant, la distance duale peut toujours être définie à partir de la distribution de poids de $\mathcal{C}$ et de la transformée de McWilliams [MS77].

Si $\mathcal{C}$ est un code linéaire et $A_0^\perp, \ldots, A_n^\perp$ est la distribution des poids de $\mathcal{C}^\perp$. Si $\mathcal{C}$ est un $(n, n-r)$-code systématique, alors le tableau ayant tous les mots du code $\mathcal{C}$ comme lignes est un $OA_{2^{n-r-d^\perp+1}}(d^\perp - 1, n)$. De plus, dans ce cas, tous les translatés $(\mathbf{0}, \mathbf{v}) + \mathcal{C}$ ont la même distribution des distances et donc la même distance duale.

Proposition 4.17 (Munuera, B. - 2011)**.** *Soient $\mathcal{C}$ un $(n, n-r)$-code systématique, σ sa fonction génératrice et $d^\perp$ sa distance duale. Soient $\mathbf{v} \in \mathbb{F}_2^r$ et $M_{\mathbf{v}}$ un tableau ayant tous les mots de $(\mathbf{0}, \mathbf{v}) + \mathcal{C}$ comme lignes. Alors*
(a) L'ensemble $\{M_{\mathbf{v}} \mid \mathbf{v} \in \mathbb{F}_2^r\}$ est un large ensemble de $OA_{2^{n-r-d^\perp+1}}(d^\perp - 1, n)$.
(b) La fonction syndrome $S_\sigma((\mathbf{u}, \mathbf{w})) = \mathbf{w} - \sigma(\mathbf{u})$ est une fonction $(d^\perp - 1)$-résiliente.

Démonstration. Soient $\mathcal{T} \subset \{1, \ldots, n\}$ avec $|\mathcal{T}| = d^{\perp} - 1$ et $\mathbf{t} \in \mathbb{F}_2^t$.
(a) D'après le théorème de Delsarte, tous les 2^t vecteurs possibles $\mathbf{t}$ sont présents dans exactement $2^{n-r-d^{\perp}+1}$ lignes de $\pi_{\mathcal{T}}(\mathcal{C})$. Comme les translations ne changent pas la distribution des distances, alors il en est de même pour les codes translatés $(\mathbf{0}, \mathbf{v}) + \mathcal{C}$.

(b) La partie (b) est une conséquence de (a), toutes les sorties possibles de $S_\sigma(\mathbf{y})$ avec $\pi_{\mathcal{T}}(\mathbf{y}) = \mathbf{t}$ sont équiprobables [Sti93]. □

Positions verrouillées aves les codes systématiques

On propose maintenant de regarder le problème du codage par syndrome avec des positions verrouillées dans le contexte des codes systématiques. Soient $\mathbf{v} \in \mathbb{F}_2^n$ le *cover-data* et $\mathbf{m} \in \mathbb{F}_2^u$ le secret que l'on souhaite insérer dans $\mathbf{v}$, $\mathcal{W} \subseteq \{1, \ldots, n\}$ désignant l'ensemble des $n - \delta$ positions verrouillées qui ne peuvent être altérées durant l'insertion. On considère un $(n, n-r)$-code systématique $\mathcal{C}$ de fonction génératrice σ et S_σ la fonction syndrome de $\mathcal{C}$. Comme dans le cas linéaire, la fonction d'extraction est la fonction syndrome et la fonction d'insertion est définie par $\mathrm{Emb}(\mathbf{v}, \mathbf{m}) = \mathbf{y}$ avec

$$[SS] : \begin{cases} S_\sigma(\mathbf{y}) & = \mathbf{m}, \\ \mathbf{y}_i & = \mathbf{c}_i \ \text{ si } i \in \mathcal{W} \end{cases}$$

Comme dans le cas linéaire, le seuil de positions verrouillées de $\mathcal{C}$ est le nombre maximal de positions verrouillées tel que le système $[SS]$ possède toujours une solution. Une telle solution existe si et seulement si $\pi_{\mathcal{W}}(\mathbf{v} + (\mathbf{0}, \mathbf{m})) \in \pi_{\mathcal{W}}(\mathcal{C})$. Dans ce cas, si $\mathbf{c} \in \mathcal{C}$ verifie $\pi_{\mathcal{W}}(\mathbf{c}) = \pi_{\mathcal{W}}(\mathbf{v} + (\mathbf{0}, \mathbf{m}))$, alors $\mathbf{y} = \mathbf{c} + (\mathbf{0}, \mathbf{m})$ est une solution.

Proposition 4.18 (Munuera, B. - 2011)**.** *Soit $\mathcal{C}$ un $(n, n-r)_2$-code systématique. Si $\delta \geq n - d^{\perp} + 1$, alors le système $[SS]$ a une solution pour tous $\mathbf{v} \in \mathbb{F}_2^n$, $\mathbf{m} \in \mathbb{F}_2^r$ et $\mathcal{W} \subset \{1, \ldots, n\}$ avec $|\mathcal{W}| = n - \delta$. De plus, dans ce cas le système $[SS]$ a exactement $2^{\delta - r}$ solutions.*

Démonstration. Pour tout $\mathbf{v} \in \mathbb{F}_2^n$, $\mathbf{m} \in \mathbb{F}_2^r$ et $\mathcal{W} \subset \{1, \ldots, n\}$, il existe une solution au système $[SS]$ si et seulement si $\pi_{\mathcal{W}}(\mathcal{C}) = \mathbb{F}_2^{n-\delta}$. D'après la proposition 4.17, qui est basée sur le théorème de Delsarte, on a

$$\begin{aligned} |\mathcal{W}| &\geq d^{\perp} - 1 \\ \delta &\geq n - d^{\perp} - 1. \end{aligned}$$

□

Alors pour les codes systématiques le seuil de positions verrouillées τ vérifie $\tau \geq n - d^{\perp} + 1$. Pour les codes linéaires, le théorème 4.10 donne l'égalité, c'est-à-dire $\tau = n - d^{\perp} + 1$. La différence importante entre les codes linéaires et les codes systématiques non-linéaires, c'est que le code dual $\mathcal{C}^{\perp}$ n'existe pas toujours. De plus, pour les codes linéaires la condition suffisante pour assurer l'existence d'une solution est également

une condition nécessaire. Cependant, pour les codes systématiques non-linéaires, cette condition est simplement suffisante mais pas nécessaire. En conséquence, il peut arriver qu'un schéma stéganographique basé sur un code systématique non-linéaire ait un seuil de positions verrouillées $\tau > n - d^{\perp} + 1$. L'exemple 4.4 en est une bonne illustration.

Exemple 4.4. On regarde maintenant la distribution des distances pour le code de Nadler $\mathcal{C}$, ainsi que pour son dual $\mathcal{C}^{\perp}$, en table 4.1.

distance	0	1	2	3	4	5	6	7	8	9	10	11	12
Distribution de $\mathcal{C}$	1	0	0	0	0	12	12	0	3	4	0	0	0
Distribution de $\mathcal{C}^{\perp}$	1	0	0	4	18	36	24	12	21	12	0	0	0

TABLE 4.1 – Distributions des distances et des distances duales du code de Nadler.

On remarque que la distance duale $d^{\perp} = 3$. Soit $\mathcal{S}$ un schéma stéganographique basé sur ce code. Alors $[SS]$ le système associé a une solution pour tout $\mathbf{v} \in \mathbb{F}_2^{12}$, $\mathbf{m} \in \mathbb{F}_2^5$ et $\mathcal{W} \subset \{1, \ldots, 12\}$ si $|\mathcal{W}| \leq 2$. En effet, d'après la proposition 4.18, c'est une condition suffisante, mais pas nécessaire. En faisant un test exhaustif on peut remarquer que pour tout ensemble de 4 colonnes de $\mathcal{C}$ tous les 2^4 vecteurs de $\mathbb{F}_2^4$ sont présents. En accord avec la borne de Bush [HSS99], c'est le nombre maximal de colonnes possibles pour lequel cela peut arriver. On en déduit que le système $[SS]$ a toujours une solution quand le nombre de positions verrouillées est au plus 4. En comparant avec les codes linéaires de mêmes paramètres, on trouve un $[12, 7, 4]_2$-code linéaire [Gra07], et le nombre maximal de positions verrouillées est alors exactement de 3.

La précédente proposition 4.18 et l'exemple 4.4 suggèrent l'utilisation des codes systématiques non-linéaires dans le problème du codage par syndrome avec positions verrouillées. Le principal inconvénient de l'utilisation de tels codes est le coût calculatoire pour résoudre le problème $[SS]$. Mais résoudre un système d'équations booléennes est un problème important en algèbre calculatoire et en informatique. Il existe différentes méthodes disponibles, dont certaines sont très efficaces quand le nombre de variables n'est pas trop grand [CGY08, Kei06]. Bien évidement, à paramètres identiques, le coût calculatoire pour résoudre $[SS]$ le système non-linéaire est toujours plus grand que le système linéaire : $[S]$. L'utilisation des codes systématiques peut être intéressant quand le gain de sécurité – qui est croissante en le nombre de positions verrouillées – est plus important que le coût calculatoire engendré.

4.4.5 Conclusion

Dans cette section on a obtenu des conditions nécessaires et suffisantes pour assurer la solution du problème du codage par syndrome avec des positions verrouillées. Ces conditions ont été d'abord exhibées pour les codes linéaires en fonction de la distance duale. Dans un deuxième temps, on a calculé l'*overhead* ainsi que l'*overhead* moyen.

Un lien évident avec le défaut de Singleton a été établi lors de la première partie, on s'est donc intéressé aux poids de Hamming généralisés. Enfin, on a étendu aux codes systématiques – qui est une généralisation des codes linéaires – les conditions trouvées précédemment. Ceci prouve la nature combinatoire de nos résultats. On s'aperçoit que les conditions nécessaires et suffisantes dans le cas des codes linéaires deviennent dans le cas non-linéaire des conditions suffisantes mais plus nécessaires. On en déduit donc, comme montre l'exemple 4.4, que les codes systématiques non-linéaires représentent de bons candidats pour le problème du codage par syndrome avec positions verrouillées, même si cela augmente le coût calculatoire de l'insertion.

4.5 Première méthode sans échec

Comme on a pu remarquer auparavant, le problème du codage par syndrome borné n'a pas toujours de solution. Comme le problème du codage par syndrome borné avec des positions verrouillées comporte des contraintes supplémentaires, il est clair qu'il en est de même pour ce dernier, comme cela a été remarqué dans [FGS06]. Dans [SW06], Schönfeld et Winkler proposent d'utiliser les codes BCH binaires. Ils ont montré expérimentalement que les codes BCH binaires et les codes de Hamming binaires, avaient une plus petite probabilité d'échec que les codes aléatoires binaires à longueur et dimension identiques. Comme ces résultats sont de nature expérimentale, ils n'ont pu les obtenir que pour des petites tailles de codes. Dans [FJF10], Filler, Judas et Fridrich proposent d'utiliser les codes treillis, cependant, ils buttent sur le même problème. Ils proposent soit de restreindre les codes a une sous-famille de codes, pour lesquelles la probabilité que l'insertion échoue est faible, soit de modifier des positions verrouillées. La proposition d'utiliser des codes de Reed-Solomon de Fontaine et Galand semble mieux répondre à ce problème [FG09]. En effet, les codes de Reed-Solomon sont des codes MDS, ainsi toutes les sous-matrices d'au moins $n-k$ colonnes d'une matrice de parité sont de rang plein. Et donc le schéma stéganographique obtenu à partir des codes de Reed-Solomon permet d'avoir une solution au problème du codage par syndrome avec des positions verrouillées tant que $\ell \leq k$. Cependant, les codes de Reed-Solomon sont définis sur $\mathbb{F}_q$ tel que $q \geq n$ et pour bénéficier des avantages que procurent ces codes, il faut au préalable redéfinir le contexte stéganographique en q-aire et non plus en binaire.

Dans cette section, on modifie très légèrement le problème du codage par syndrome borné avec des positions verrouillées, pour s'assurer de l'existence de solution. Cette méthode étant tout à fait générale, on peut donc l'appliquer à différents codes binaires et q-aires. L'assurance de l'insertion est obtenue au prix d'une légère baisse d'*embedding efficiency*. On montre que la perte d'*embedding efficiency* pour les codes de Hamming binaires est logarithmique en le nombre de positions verrouillées.

Ce travail à fait l'objet d'une présentation à la conférence *Institute of Mathematics and its applications on Cryptography and Coding 2011* [ABF11].

4.5.1 Reformulation du problème

Partant du constat que l'insertion avec des positions verrouillées a une probabilité d'échec non-nulle, on introduit alors une variante garantissant l'insertion du message. Cette idée est fortement inspirée du système de signature de Courtois, Finiasz et Sendrier [CFS01].

Problème 4.19 (Codage par syndrome partiel avec positions verrouillées. Augot, B., Fontaine - 2011)**.** *Soient* $\mathbf{v} \in \mathbb{F}_2^n$ *le cover-data,* H *une matrice de parité d'un* $[n,k]_2$*-code linéaire,* $T, r \in \mathbb{N}$*,* $\mathcal{W} \subset \{1,\ldots,n\}$ *et* $\mathbf{m} \in \mathbb{F}_2^{n-k-r}$ *le message que l'on souhaite insérer. On cherche un stego-data* $\mathbf{y} \in \mathbb{F}_2^n$ *tel que*

1. $\mathbf{y}H^t = (\mathbf{m}||\mathbf{R})$*, où* $\mathbf{R} \in \mathbb{F}_2^r$ *est un vecteur aléatoire et* $||$ *est la concaténation,*
2. $d(\mathbf{v},\mathbf{y}) \leq T$
3. $\mathbf{v}_i = \mathbf{y}_i,\ \forall i \in \mathcal{W}$*.*

Relaxer le problème comme proposé ci-dessus permet d'augmenter le degré de liberté du système et de s'assurer qu'il comporte toujours une solution. On peut contrôler ce degré de liberté en modifiant la taille r de la partie aléatoire. La garantie de l'insertion est alors obtenue au détriment de l'*embedding efficiency*. On étudie la perte en l'*embedding efficiency*. On note e, respectivement e', l'*embedding efficiency* de la méthode standard, respectivement de notre méthode. On a alors :

$$\begin{aligned}\frac{e'-e}{e} &= \frac{\frac{n-k-r}{T} - \frac{n-k}{T}}{\frac{n-k}{T}},\\ &= -\frac{r}{n-k}.\end{aligned}$$

La perte de l'embedding efficiency est alors $\frac{r}{n-k}$. Il est clair que l'on a intérêt à choisir r le plus petit possible.

4.5.2 Codes linéaires parfaits

Pour calculer la plus petite valeur de r qui garantit l'insertion, on commence dans un premier temps par calculer N_1, le nombre de syndromes différents atteignables par la condition du problème du codage par syndrome partiel avec positions verrouillées et N_2, le nombre de syndromes différents atteignables par les conditions deux et trois de ce problème. Pour s'assurer que notre nouveau système admet une solution pour tout m, on va choisir r tel que la somme de N_1 et de N_2 soit toujours plus grande que la cardinalité de l'ensemble de tous les syndromes, c'est-à-dire 2^{n-k}. Avant de résoudre notre reformulation du problème en toute généralité, on commence par s'intéresser au cas particulier des codes linéaires parfaits.

Résolution

La fonction syndrome associée est alors bijective – corollaire 1.20 – ce qui rend dans un premier temps l'analyse plus simple. Ceci nous permet de mettre en avant une condition suffisante pour que le problème 4.19 admette une solution :

Proposition 4.20 (Augot, B., Fontaine - 2011)**.** *Pour un $[n, k, d]_q$-code linéaire parfait donné, de rayon de recouvrement $\rho = \lfloor \frac{d-1}{2} \rfloor$. En gardant les notations précédentes, si l'inégalité*

$$q^{n-k} + 1 \leq q^r + \sum_{i=0}^{\rho} (q-1)^i \binom{n-\ell}{i}, \tag{4.1}$$

est vérifiée, alors il existe un vecteur $\mathbf{y} \in \mathbb{F}_q^n$ solution du problème 4.19. Dans ce cas, le problème 4.19 admet toujours une solution en $\mathbf{y}$ pour tout $\mathbf{m} \in \mathbb{F}_2^{n-k-r}$.

Démonstration. Soient N_1 – respectivement N_2 – le nombre de syndromes différents engendrés par le sous-ensemble de $\mathbb{F}_q^n$ satisfaisant la condition (1) du problème 4.19 — respectivement les conditions (2) et (3). Si

$$N_1 + N_2 > q^{n-k}, \tag{4.2}$$

alors il existe $\mathbf{y}$ qui satisfait les conditions (1), (2), et (3). Le nombre de syndromes différents atteignables par la première contrainte est q^r. Gardant à l'esprit qu'il y a ℓ composantes verrouillées et que la fonction syndrome restreinte à $\mathcal{B}(\mathbf{x}, \rho)$ est bijective, alors

$$N_2 = \sum_{i=0}^{\rho} (q-1)^i \binom{n-\ell}{i}.$$

En combinant avec la condition suffisante (4.2), on obtient le résultat. □

Codes de Golay

Il existe deux codes de Golay parfaits qui sont le code de Golay binaire $[23, 12, 7]_2$ et le code de Golay ternaire $[11, 6, 5]_3$. On présente la résolution de notre nouveau problème 4.19 avec ces deux codes.

Code de Golay binaire. On commence cette étude par le cas du code de Golay binaire $[23, 12, 7]_2$. L'inégalité de la proposition 4.20 nous donne

$$r \geq \log_2 \left(1 + \frac{796}{3}\ell - \frac{23}{2}\ell^2 + \frac{1}{6}\ell^3 \right). \tag{4.3}$$

Code de Golay ternaire. Le code de Golay ternaire a pour paramètre$[11, 6, 5]_3$. Toujours en utilisant la proposition 4.20, on obtient :

$$r \geq \log_3 \left(1 + 44\ell - 2\ell^2 \right). \tag{4.4}$$

Les équations 4.3 et 4.4 n'illustrent pas forcément bien le phénomène. Les figures 4.4 et 4.5 montrent que le nombre de bits, ou symboles, disponibles pour l'insertion se dégrade rapidement avec le nombre de positions verrouillées.

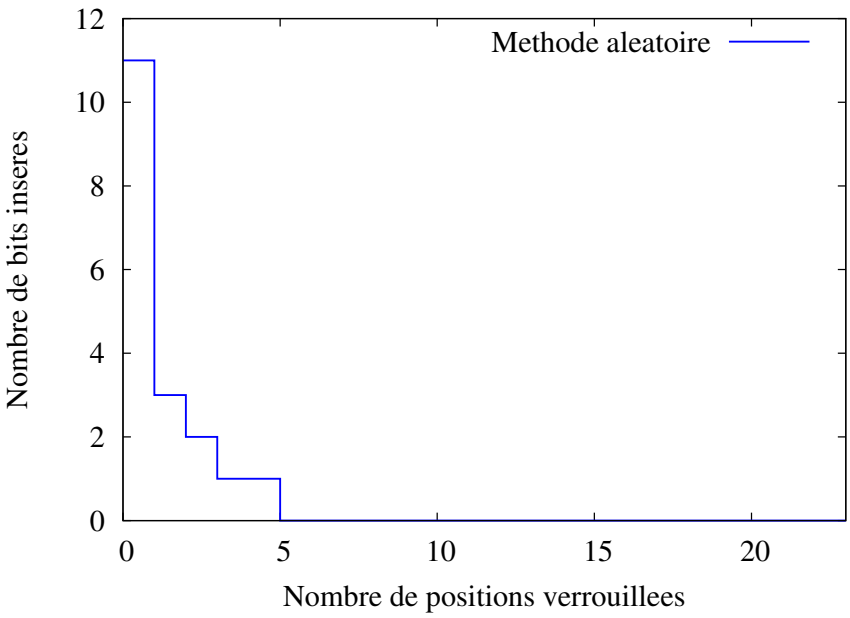

FIGURE 4.4 – Nombre de bits insérés en fonction du nombre de positions verrouillées pour le code de Golay binaire $[23, 12, 7]_2$.

Codes de Hamming

Calcul de r. Soit $\mathcal{C}$ un code de Hamming sur $\mathbb{F}_q$. Sa longueur est $n = (q^m-1)/(q-1)$, sa dimension est $k = n - m$, sa distance minimale est $d = 3$ et son rayon de recouvrement est $\rho = 1$. On souhaite minimiser r – la longueur du vecteur aléatoire – tant que le problème 4.19 admette une solution. Tout d'abord, le nombre N_1 de différents syndromes satisfaisant la première condition est q^r. Comme les codes de Hamming sont parfaits, c'est-à-dire $T = \lfloor (d-1)/2 \rfloor = \rho = 1$, alors le nombre de syndromes différents satisfaisant les deux dernières conditions est

$$\begin{aligned} N_2 &= \sum_{i=0}^{1} (q-1)^i \binom{n-\ell}{i} \\ &= q^m - (q-1)\ell. \end{aligned}$$

Comme $q^{n-k} = q^m$ et $(q^m - 1)/(q-1) = n$, on obtient une borne inférieure sur r :

$$\begin{aligned} N_1 + N_2 &\geq q^{n-k} + 1 \\ q^r + (q^m - (q-1)\ell) &\geq q^m + 1 \\ r &\geq \log_q((q-1)\ell + 1). \end{aligned} \tag{4.5}$$

Dans le cas extrême du problème du codage par syndrome sans positions verrouillées, on a alors $\ell = 0$ alors la taille de la partie aléatoire est $r = \log_q(1) = 0$, comme attendu.

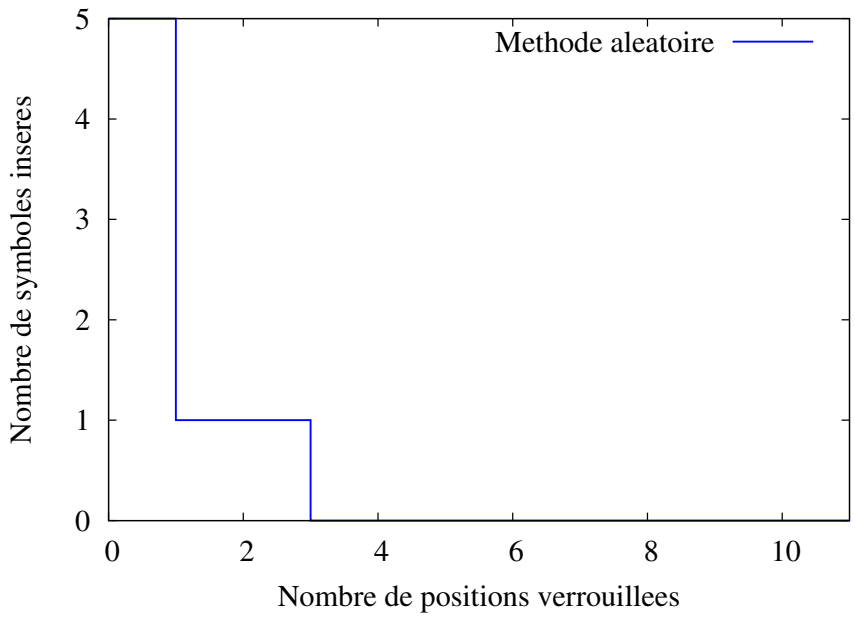

FIGURE 4.5 – Nombre de symboles insérés en fonction du nombre de positions verrouillées pour le code de Golay ternaire $[11, 6, 5]_3$.

Dans un autre cas extrême où toutes les positions sont verrouillées , tels que $\ell = n$, la taille de la partie aléatoire est alors $r = \log_q(q^m - 1 + 1) = m = n - k$. Sans surprise, si toutes les positions sont verrouillées, alors on ne peut pas être certain d'insérer n'importe quel message.

Analyse des paramètres

On regarde maintenant comment maximiser le nombre de positions verrouillées, tout en gardant $n - k - r$ le nombre de symboles du message à insérer strictement positif. Cela revient à trouver le plus grand ℓ tel que $n - k - r = 1$. Pour les codes de Hamming, $\ell_{\max}$ est donné par

$$\begin{aligned} m - 1 &= \log_q((q-1)\ell_{\max} + 1), \\ \ell_{\max} &= \left\lfloor \frac{q^{m-1} - 2}{q - 1} \right\rfloor, \\ \ell_{\max} &\approx \frac{n}{q}. \end{aligned}$$

Avec notre méthode et les codes de Hamming binaires, si pas plus de 50% des positions sont verrouillées, on est certain de pouvoir insérer au moins un bit d'information. La plus petite valeur de r vérifiant l'inégalité (4.5) est $r = \lceil \log_q((q-1)\ell + 1) \rceil$. En d'autres termes, le nombre minimum de symboles aléatoires nécessaires pour

garantir l'insertion de n'importe quel message dans n'importe quel *cover-data* avec notre méthode, est logarithmique en le nombre de positions verrouillées. Grâce à cette méthode d'insertion, on atteint exactement le seuil de positions verrouillées d'un code de Hamming binaire, exemple 4.1.

Notre méthode aléatoire permet de résoudre le problème 4.5 avec un taux de succès égal à 100%, tandis que la méthode traditionnelle exhibe une probabilité d'échec non nulle qui augmente exponentiellement avec le nombre de bloc du message et le nombre de positions verrouillées. Il est cependant évident que l'*embedding efficiency* de notre méthode souffre d'une légère perte. La perte pour les codes de Hamming en fonction de ℓ le nombre de positions verrouillées, est

$$\frac{-\lceil \log_q((q-1)\ell+1)\rceil}{n-k}.$$

En utilisant ici les codes de Hamming binaires, la perte relative de l'*embedding efficiency* est de $\frac{\lceil \log_q((q-1)\ell+1)\rceil}{n-k}$. Cette perte est illustrée dans la figure 4.6, pour le code de Hamming binaire de longueur 31. On présente également une comparaison entre la probabilité de réussite de la méthode traditionnelle et la notre, en utilisant le $[15, 11, 3]_2$-code de Hamming. Cette comparaison a été réalisée pour 1, 3 et 5 positions verrouillées figure 4.7.

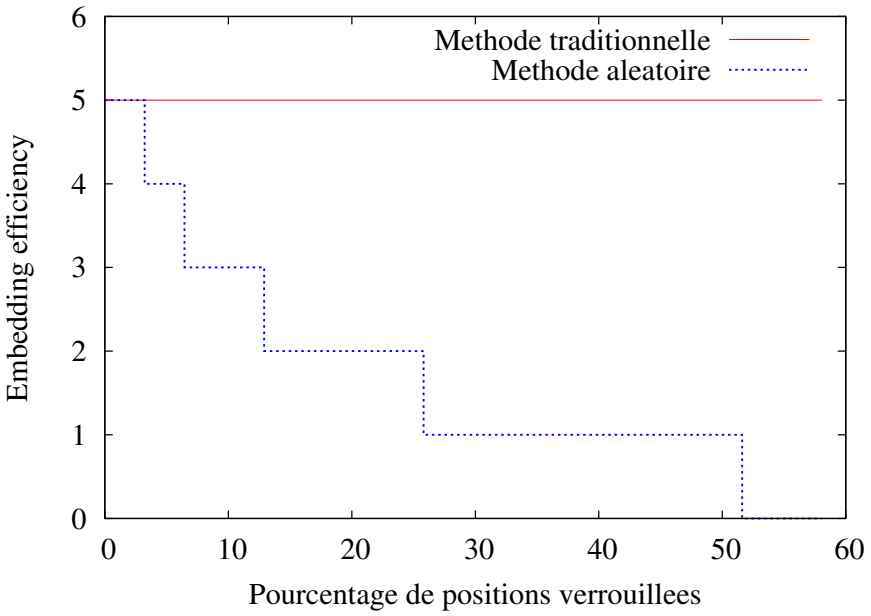

FIGURE 4.6 – Comparaison de l'embbeding efficiency en fonction du nombre de positions verrouillées pour le code binaire de Hamming de longueur 31.

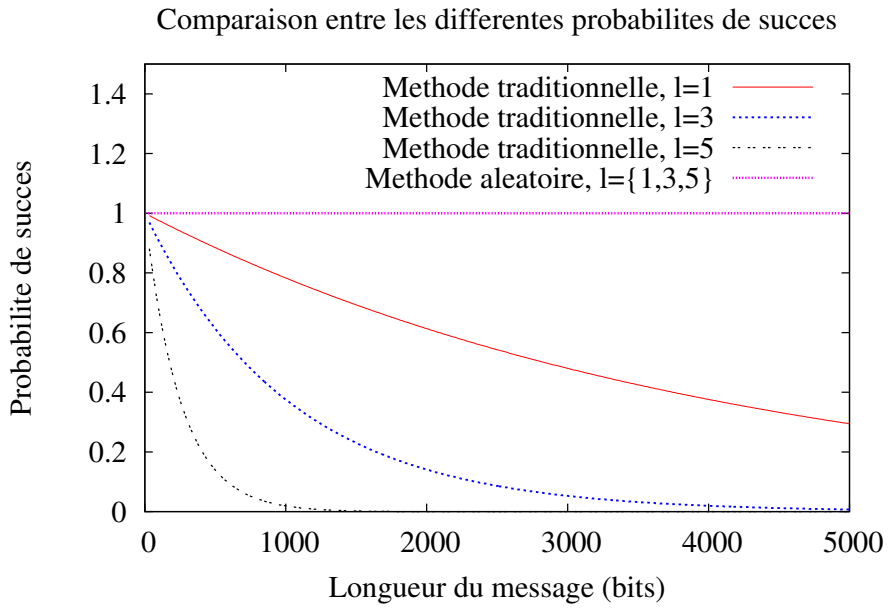

FIGURE 4.7 – Comparaison de la probabilité globale de réussite de la méthode traditionnelle et de notre méthode aléatoire en utilisant le $[15, 11, 3]_2$-code de Hamming avec 1, 3 et 5 positions verrouillées.

Pour être tout à fait honnête, l'*embedding efficiency* et la probabilité de réussite de ces méthodes doivent être analysés conjointement. L'*embedding efficiency* de la méthode traditionnelle ne diminue pas, mais sa probabilité de réussite décroît exponentiellement avec le nombre de blocs du message et avec le nombre de positions bloquées. Tandis que notre méthode exhibe une probabilité de réussite exactement égal à un tant que $\ell \leq \ell_{max}$, l'*embedding efficiency* diminue logarithmiquement en le nombre de positions verrouillées. On propose dans la table 4.2 de comparer la probabilité de réussite et de l'*embedding efficiency* de la méthode traditionnelle qui résout le problème du codage par syndrome avec des positions verrouillées, mais non borné et notre méthode aléatoire. Pour ces deux méthodes on fait varier le nombre de blocs du message à insérer et le nombre de positions verrouillées. On compare ces deux méthodes bien qu'elles résolvent deux problèmes légèrement différents. Notre méthode résout un problème avec plus de contraintes, donc plus compliqué. La probabilité de réussite de la méthode traditionnelle est calculée à l'aide du théorème 4.6. On peut noter que pour les messages très courts, par exemple 64 bits soit 8 lettres encodées sur 8 bits, la probabilité de la réussite de l'insertion du message entier est très proche de notre probabilité de réussite, c'est-à-dire 100%. Cependant, pour des messages raisonnablement longs, par exemple 1024 bits soit 128 lettres encodées sur 8 bits, la probabilité d'insérer tout le message est très loin de 100%, alors que notre méthode atteint les 100% avec une perte de 50% de l'embedding efficiency.

Longueur du message (en bits)	ℓ	Traditionnelle		Aléatoire	
		Proba réussite	e	Proba réussite	e
2^6 (64)	1 (6.67%)	≈ 0.98	4	1	3
2^6 (64)	3 (20%)	≈ 0.93	4	1	2
2^9 (512)	1 (6.67%)	≈ 0.88	4	1	3
2^9 (512)	3 (20%)	≈ 0.60	4	1	2
2^{10} (1 Kb)	1 (6.67%)	≈ 0.77	4	1	3
2^{10} (1 Kb)	3 (20%)	≈ 0.36	4	1	2

TABLE 4.2 – Comparaison de la probabilité globale de réussite et de l'embedding efficiency de la méthode traditionnelle et de notre méthode aléatoire, en utilisant un code de Hamming de paramètre $[15, 11, 3]_2$.

4.5.3 Cas général des codes linéaires

Résolution du nouveau problème

Quand le code utilisé n'est pas parfait, la fonction syndrome restreinte à une boule de rayon ρ, le rayon de recouvrement, n'est pas injective. Le calcul du nombre de syndromes distincts satisfaisant les deux dernières contraintes du problème 4.5 est alors plus difficile. Pour résoudre ce problème, on introduit $Nb_{\mathbf{v},T}$ qui est le nombre maximum d'éléments de la boule $\mathcal{B}(\mathbf{v}, T)$ qui donnent le même syndrome.

Proposition 4.21 (Augot, B., Fontaine - 2011)**.** *Si l'inégalité*

$$q^{n-k} + 1 \leq q^r + \frac{\sum_{i=0}^{T}(q-1)^i \binom{n-\ell}{i}}{Nb_{\mathbf{v},T}}, \tag{4.6}$$

est satisfaite, alors il existe un vecteur $\mathbf{y} \in \mathbb{F}_q^n$ *qui est solution du problème 4.5. Autrement dit, satisfaire cette inégalité est une condition suffisante pour assurer l'existence d'une solution au problème 4.5.*

Démonstration. Le calcul du nombre de différents syndromes engendrés par la première condition est clairement q^r. Toutefois, le calcul de N_2 n'est pas aussi facile. Gardant en tête que ℓ positions sont verrouillées, une borne inférieure sur N_2 est obtenue en divisant le nombre d'éléments de n'importe quelle boule de rayon T par le nombre maximal d'éléments qui donnent le même syndrome. On a alors

$$\frac{\sum_{i=0}^{T}(q-1)^i \binom{n-\ell}{i}}{Nb_{\mathbf{v},T}}.$$

En combinant avec la condition suffisante (4.2) on obtient

$$q^{n-k} + 1 \leq q^r + \frac{\sum_{i=0}^{T}(q-1)^i \binom{n-\ell}{i}}{Nb_{\mathbf{v},T}}.$$

□

On doit alors calculer la quantité r telle que l'inégalité (4.6) soit vérifiée, ce qui implique l'estimation de $Nb_{\mathbf{v},T}$. La proposition suivante nous donne une majoration de cette quantité $Nb_{\mathbf{v},T}$. Soit $V(T)$ le volume de n'importe quelle boule de rayon T.

Proposition 4.22 (Augot, B., Fontaine - 2011)**.** *Soit $\mathcal{C}$ un $[n,k,d]_q$ code linéaire de capacité de correction t et T un entier. Pour tout $\mathbf{v} \in \mathbb{F}_q^n$, on obtient l'inégalité suivante*

$$Nb_{\mathbf{v},T} \leq \frac{V(T+t)}{V(t)}. \tag{4.7}$$

Démonstration. Soit $\mathbf{v}$ un vecteur dans $\mathbb{F}_q^n$. On définit

$$\mathcal{Y}_{\mathbf{v},\mathbf{s}} \triangleq \left\{\mathbf{y} \in \mathcal{B}(\mathbf{v},T) \ : \ \mathbf{y}H^t = \mathbf{s}\right\},$$

où $\mathbf{s} \in \mathbb{F}_q^{n-k}$. On définit également $\mathbf{s}_0 \in \mathbb{F}_q^{n-k}$ tel que $|\mathcal{Y}_{\mathbf{v},\mathbf{s}_0}| = Nb_{\mathbf{v},T}$. Clairement, on a

$$\bigcup_{\mathbf{y}\in\mathcal{Y}_{\mathbf{v},\mathbf{s}_0}} \mathcal{B}(\mathbf{y},t) \quad \subset \quad \mathcal{B}(\mathbf{v},T+t). \tag{4.8}$$

Soient $\mathbf{y},\mathbf{y}' \in \mathcal{Y}_{\mathbf{v},\mathbf{s}_0}$. Alors,

$$\begin{aligned} \mathbf{y}H^t &= \mathbf{y}'H^t = \mathbf{s}_0, \\ (\mathbf{y}-\mathbf{y}')H^t &= 0. \end{aligned}$$

Donc $\mathbf{y}-\mathbf{y}' \in \mathcal{C}$ et on en déduit que $d(\mathbf{y},\mathbf{y}') \geq 2t+1$. En conséquence

$$\mathcal{B}(\mathbf{y},t) \cap \mathcal{B}(\mathbf{y}',t) = \emptyset.$$

D'après (4.8), on obtient

$$\begin{aligned} \sum_{\mathbf{y}\in\mathcal{Y}_{\mathbf{v},\mathbf{s}_0}} V(t) &\leq V(T+t), \\ Nb_{\mathbf{v},T}V(t) &\leq V(T+t). \end{aligned}$$

ce qui conclut la preuve. □

Combinant la proposition 4.22 et l'inégalité (4.6), on obtient une autre condition suffisante :

$$q^{n-k}+1 \leq q^r + \left(\sum_{i=0}^{T}(q-1)^i\binom{n-\ell}{i}\right)\frac{V(t)}{V(T+t)}. \tag{4.9}$$

Mise à part la quantité r, toutes les autres quantités sont directement connues. En d'autres mots, on obtient la formule close

$$r \geq \log_q\left(q^{n-k}+1-\left(\sum_{i=0}^{T}(q-1)^i\binom{n-\ell}{i}\right)\frac{V(t)}{V(T+t)}\right). \tag{4.10}$$

La borne supérieure donnée par la proposition 4.22 est très large. Par exemple, on ne parvient pas avec cette borne, appliquée aux codes BCH binaires 2 correcteurs, à insérer de l'information. Clairement, si on est capable de l'améliorer, alors l'inégalité de la condition suffisante serait plus fine et le stégosystème aurait une meilleure *embedding efficiency.*

4.5.4 Construction ZZW pour insérer les paramètres dynamiques

Dans l'approche donnée que l'on vient de décrire, l'expéditeur et le destinataire doivent fixer à l'avance la valeur de r. En effet, le destinataire doit savoir quelle partie du syndrome est aléatoire. Or, ceci n'est pas vraiment satisfaisant dans un schéma de positions verrouillées, pour lequel le destinataire ne doit pas avoir à connaître les détails des paramètres d'insertion. On propose dans cette sous-section, une variante du schéma de ZZW [ZZW08], qui est capable de transmettre simultanément la valeur de r, fixée dynamiquement en fonction du *cover-data*.

Notre schéma

On considère que l'on manipule n blocs de $2^p - 1$ bits, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, comme illustré en figure 4.8. Chaque bloc $\mathbf{x}_i$ est un vecteur binaire de longueur $2^p - 1$, vu comme une colonne. Soit $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ le vecteur binaire dont la i^{e} coordonnée v_i est le bit de parité de la colonne $\mathbf{x}_i$. On utilise le vecteur – virtuel – $\mathbf{v}$ pour insérer de l'information additionnelle, et en même temps les vecteurs $\mathbf{x}_i$ sont utilisés pour faire du codage par syndrome.

Notre schéma est triple : codage par syndrome sur les vecteurs $\mathbf{x}_i$ en utilisant la matrice de parité H_1 d'un premier code de Hamming binaire de longueur $2^p - 1$, avec notre méthode de codage par syndrome partiel. Les vecteurs $\mathbf{s}_i$ sont les syndromes des vecteurs $\mathbf{x}_i$ du code de Hamming binaire de longueur $2^p - 1$. Le second codage par syndrome considère les vecteurs $\mathbf{s}_i$ comme des symboles q–aires et la matrice utilisée est une matrice de parité H_q d'un code de Reed-Solomon q–aire. On appelle les n premières insertions, l'insertion par H_1, et la seconde l'insertion par H_q. Finalement, on utilise le vecteur $\mathbf{v}$ pour insérer de l'information dynamique, qui sont r le nombre de bits aléatoires et f le nombre d'échec dans l'insertion par H_1. On appelle la dernière étape, l'insertion par H_2, où H_2 est la matrice de parité d'un second code de Hamming binaire, mais plus court.

On suppose que r est borné par le schéma, c'est-à-dire $r \leq r_{\max}$, où $r_{\max}$ est un paramètre du schéma, que l'on détaille par la suite.

Insertion

Inspection. Chaque colonne $\mathbf{x}_1, \ldots, \mathbf{x}_n$ est vérifiée, pour trouver le nombre de positions modifiables dans chacune des colonnes. Cela permet de déterminer la taille r de la partie aléatoire de notre méthode, qui doit être la même pour chacune des colonnes. Cela a pour effet de déterminer les colonnes $\mathbf{x}_i$ pour lesquelles l'insertion par H_1 est possible. On pose f le nombre de vecteurs $\mathbf{x}_i$ pour lesquels l'insertion par H_1 a échoué.

Verrouillage des positions. Pour chacune des $n - f$ colonnes $\mathbf{x}_i$ où l'insertion par H_1 est possible, il y a un syndrome $\mathbf{s}_i$ de p bits, où les r derniers bits sont aléatoires, c'est-à-dire verrouillés et les $p - r$ premiers bits sont sous contrôle. On considère alors

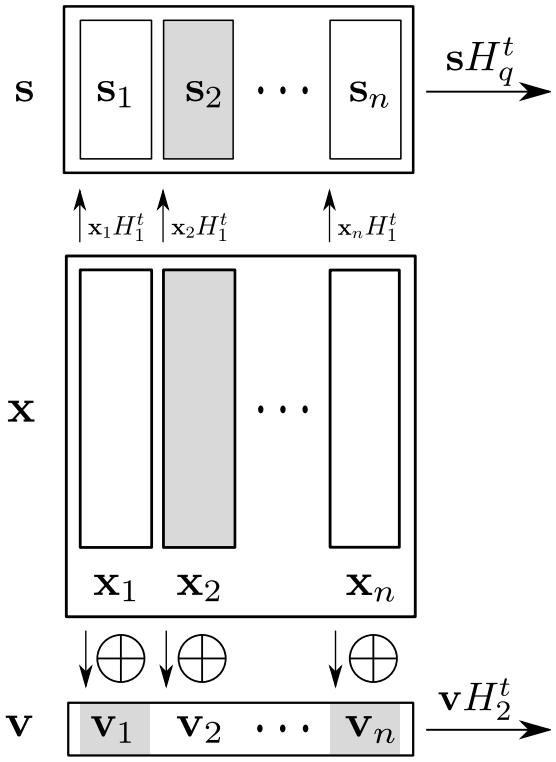

FIGURE 4.8 – Graphique de notre schéma inspiré de ZZW. Un syndrome $\mathbf{s}_i$ est considéré verrouillé pour l'insertion par H_q lorsque l'insertion par H_1 a échoué. Alors le bit correspondant v_i du vecteur $\mathbf{v}$ est modifiable pour l'insertion par H_2. Les données verrouilléss sont grisées sur la figure.

ces blocs de $p-r$ bits comme des symboles de $\mathbb{F}_q$, où $q = 2^{p-r}$. On obtient donc un codage par syndrome avec f positions verrouillées et $n-f$ symboles q–aires modifiables.

Insertion avec des positions verrouillées. En utilisant un code de Reed-Solomon sur l'alphabet $\mathbb{F}_q$, on peut insérer$(n-f)$ symboles q-aires, en utilisant H_q une $n \times (n-f)$ matrice de parité q-aire de ce code. On remarque que la taille de cette matrice dépend de f, qui est un paramètre dynamique du schéma.

Insertion des paramètres du schéma. On doit maintenant insérer les paramètres dynamiques r et f qui sont inconnus du destinataire. Pour cela on suggère d'utiliser le vecteur virtuel $\mathbf{v}$ de la méthode de ZZW. Pour ce canal, les bits modifiables $\mathbf{v}_i$ correspondent aux colonnes $\mathbf{x}_i$ où l'insertion par H_1 a échouée et pour lesquelles il y a au moins un bit modifiable dans $\mathbf{x}_i$. Un second code de Hamming est utilisé avec la matrice de parité H_2 pour cette insertion. Le destinataire est alors en mesure de retrouver r et f, les paramètres du schéma. Les valeurs de ces paramètres sont données plus loin.

Récupération

Extraction par H_2. Tout d'abord calculer $\mathbf{v}$ et utiliser la matrice de parité H_2 du code de Hamming pour extraire r et f les paramètres dynamiques du schéma.

Extraction par H_1. Extraire les syndromes de toutes les colonnes $\mathbf{x}_i$ à l'aide de la matrice de parité H_1 et garder seulement les $p-r$ premiers bits de chaque syndrome, pour ainsi construire des symboles q–aires.

Extraction par H_q. Construire la matrice de parité H_q d'un $[n,f]_q$-code de Reed-Solomon, avec $q = 2^{p-r}$. En utilisant cette matrice, on obtient alors un vecteur q–aire de $(n-f)$ symboles q-aires, qui est en fait de l'information insérée.

Analyse

Notre schéma est soumis à différentes contraintes. Tout d'abord, un code de Reed-Solomon de longueur n sur $\mathbb{F}_{2^{p-r}}$ existe si et seulement si $n \leq 2^{p-r}$. Donc $n \leq 2^{p-r_{\max}}$. On fixe alors $n = 2^{p-r_{\max}} - 1$ et on note $u = p - r_{\max}$.

Alors le code de Hamming binaire $[2^u-1, 2^u-u-1]_2$ ayant pour matrice de parité H_2 est utilisé pour insérer dans le vecteur $\mathbf{v}$, avec f symboles modifiables. Du théorème 4.10, on doit également avoir

$$f \geq 2^{u-1}, \tag{4.11}$$

ce qui implique que certaines colonnes $\mathbf{x}_i$ doivent être déclarées artificiellement verrouillées par H_q, pour satisfaire l'équation 4.11. De plus on doit également avoir

$$u = \lceil \log r_{\max} \rceil + \lceil \log f \rceil , \tag{4.12}$$

pour être capable d'insérer à la fois r et f. Comme $f \leq 2^u - 1$, on a $\lceil \log f \rceil = u$. L'équation 4.12 devient alors $u = \lceil \log r_{\max} \rceil + u$, ce qui est clairement impossible. Pour remédier à cela, au lieu d'insérer f, on propose d'insérer sa valeur relative $f_u = \frac{f}{2^u} \in [1/2, 1]$, pour une précision fixée, c'est-à-dire o bits, avec o petit. Alors l'équation 4.12 est remplacée par

$$u = \lceil \log r_{\max} \rceil + o, \tag{4.13}$$

$$p = r_{\max} + \lceil \log r_{\max} \rceil + o, \tag{4.14}$$

qui est, cette fois-ci, une condition facile à satisfaire. Il l'est aussi possible par construction d'utiliser le vecteur tout a un, qui est en dehors de l'intervalle pour déclarer un échec d'insertion.

Remarque 4.5. Le schéma s'adapte alors au support. Par exemple, pour une image donnée, r et f peuvent être différents sur certaines positions de l'image.

Pour conclure cette partie, le nombre de bits que l'on peut alors insérer avec cette méthode est majoré par $(n-f)(p-r) \leq 2^{u-1}(p-r)$, où r et f sont des paramètres dynamiques.

4.5.5 Conclusion

Comme la méthode traditionnelle peut échouer, on a proposé ici une méthode différente qui n'échoue jamais. On créé un degré de liberté dans le système linéaire associé au problème d'insertion, en rajoutant une partie aléatoire dans le syndrome. On détaille ce procédé et ses performances pour les codes de Hamming, les deux codes parfaits de Golay et on donne une formule close pour résoudre notre problème pour n'importe quel code linéaire. Notre méthode souffre d'une légère baisse de l'*embedding efficiency*, mais elle peut insérer n'importe quel message dans n'importe quel support en minimisant la distorsion. Pour la famille des codes de Hamming binaires, la perte en *embedding efficiency* est logarithmique en fonction du nombre de positions verrouillées, ce qui est satisfaisant. De plus, on remarque que la méthode traditionnelle exhibe une faible probabilité de réussite globale pour l'insertion d'un grand nombre de blocs du message, alors que notre méthode permet une insertion garantie. Ainsi notre méthode est tout à fait pertinente en pratique. Une perspective intéressante à ce travail serait d'améliorer la minoration de la taille de la partie aléatoire pour les codes linéaires en général. Ce qui permettrait ainsi de rendre cette technique réalisable pour beaucoup plus de codes et d'augmenter l'*embedding efficiency*.

Conclusion

Dans cette thèse, on a abordé trois thèmes distincts. Dans un premier temps, on s'est intéressé aux problèmes de décodage des codes correcteurs. Dans un deuxième temps, on a considéré le système de chiffrement de McEliece. Finalement, on a traité les problèmes d'insertion en stéganographie basée sur les codes correcteurs. Bien que ces thèmes soient apparemment distincts, on a montré comment une amélioration du rayon de décodage pour les codes de Goppa binaires permet de réduire la clé publique du cryptosystème de McEliece. Enfin, en stéganographie on a regardé le problème du codage par syndrome. Cependant, le problème d'insertion d'un message basé sur les codes correcteurs n'admet pas toujours de solution. Malgré que ce problème soit primordial, il n'a pas reçu beaucoup d'attention dans la littérature, c'est pourquoi il a fait l'un des principaux objets d'analyses de cette thèse.

Décodage

En collaboration avec Daniel Augot et Alain Couvreur, on a redécouvert indépendamment une méthode méconnue de Tal et Roth. C'est un algorithme de décodage en liste qui permet de décoder les codes alternants q-aire jusqu'à la borne de Johnson q-aire. Cette méthode est fortement basée sur l'algorithme de décodage en liste de Guruswami-Sudan et sur une technique de décodage souple appliquée au décodage dur. On a étudié les paramètres de cette méthode et la complexité en temps. Puisque les méthodes de décodage en liste sont généralement coûteuses en temps de calcul, un axe important de travail pourrait être d'améliorer la complexité de notre méthode, en essayant par exemple de diminuer les multiplicités de l'interpolation. De plus, on a regardé également le problème de décodage complet d'un point de vue algorithmique. On a défini une nouvelle classe de codes, les codes $\mathcal{A}$-couverts, pour lesquels le problème du décodage complet se résout en temps polynomial en les paramètres du code. En plus des codes parfaits linéaires, on a montré que les codes poinçonnés de Reed-Muller du premier ordre sont dans cette classe. De manière expérimentale, on a trouvé, en plus des codes de Goppa binaires de très faibles dimensions, des codes BCH binaires cette fois-ci de dimensions raisonnables. Ces résultats nous laissent espérer une intersection non vide entre la classe que l'on propose et les codes BCH binaires. Pour poursuivre ce travail, on pourrait tenter de montrer la conjecture précédente sur les BCH binaires –

ou de montrer son impossibilité – plus généralement, essayer de trouver d'autres codes qui sont dans notre classe.

Cryptographie

On a regardé également les systèmes de chiffrement basés sur les codes correcteurs, par exemple le cryptosystème à clé publique de McEliece. Un de ses principaux inconvénients – voire le seul – est la taille des clés publiques. Beaucoup d'auteurs ont suggéré l'utilisation des codes structurés pour diminuer la taille des clés. Cependant, ces techniques sont sujettes aux attaques structurelles qui essaient de retrouver la clé privée de la clé publique en utilisant la structure du code utilisé. On suggère avec Paulo Barreto, d'appliquer la méthode de décodage en liste présentée dans ce manuscrit au schéma de McEliece, pour diminuer la taille des clés. En effet, on peut rajouter autant d'erreurs durant la phase de chiffrement, que l'on peut corriger pendant l'étape de déchiffrement. Bien que les attaques de types décodages soient alors plus difficiles, on n'ajoute aucune structure qui pourrait faciliter une attaque structurelle. Ajouter alors plus d'erreurs que la capacité de correction, permet d'augmenter le niveau de sécurité pour une taille de clé fixée et, réciproquement, permet à niveau de sécurité donnée de diminuer la taille des clés du cryptosystème. De plus, on montre que la variante quasi-dyadique n'est pas totalement cryptanalysée pour le moment. On observe alors un gain allant jusqu'à 21% pour la variante dyadique en utilisant un algorithme de décodage en liste. Une perspective attrayante serait de chercher une famille de codes avec un algorithme de décodage en liste et n'exhibant pas une trop grand structure, telle que les clés publiques du cryptosystème de McEliece basées sur ces codes soient encore plus petites.

Stéganographie

Une autre application de la théorie des codes à la sécurité de l'information est la stéganographie. On s'est également intéressé aux problèmes de codage par syndrome avec positions verrouillées, tout particulièrement à l'existence de solutions lors de l'étape de l'insertion. Avec Carlos Munuera, on a exhibé des conditions nécessaires et suffisantes à l'existence de solution pour ce problème, basé sur les codes linéaires dépendant de la distance duale. On a reformulé nos conditions en fonction du rang MDS des codes utilisés. Comme conséquence directe, on a montré que l'utilisation d'un code de faible rang MDS est un bon candidat pour le problème du codage par syndrome. De plus, on a étendu nos conditions pour les codes systématiques, vus comme une généralisation des codes linéaires. On observe que les conditions obtenues sont des conditions suffisantes mais non nécessaires et on présente un exemple avec le code de Nadler telle que cette condition ne soit pas nécessaire. On en déduit que les codes systématiques non linéaires sont également de bons candidats pour l'insertion basée sur

le problème du codage par syndrome avec des positions verrouillées. Pour approfondir ce travail, on pourrait chercher une borne supérieure sur le nombre de changements nécessaires pour réussir l'insertion. Puisque ces conditions n'apportent pas de solution algorithmique pour résoudre avec certitude le problème du codage par syndrome avec positions verrouillées, on a présenté avec Daniel Augot et Caroline Fontaine, une nouvelle méthode d'insertion. Cette technique est fortement basée sur le schéma de signature de Courtois, Finiasz et Sendrier. Elle consiste à rendre aléatoire une partie du syndrome, ce qui a pour effet de créer un degré de liberté dans le système linéaire à résoudre. Puisque le message n'est plus le syndrome tout entier, mais seulement une partie, cette méthode souffre alors d'une légère perte en *embedding efficiency*. On a montré que cette perte est logarithmique en le nombre de positions verrouillées pour les codes de Hamming binaires. Puisque notre méthode est paramétrée, on reformule également le schéma de ZZW pour gérer dynamiquement les nouveaux paramètres de notre schéma et les transmettre au destinataire. La majoration des paramètres de notre méthode est grossière pour tout code linéaire ; un travail significatif reste nécessaire pour affiner notre estimation.

Annexe A

Algorithme de Berlekamp-Massey

Une alternative à l'algorithme d'Euclide étendu est l'algorithme de Berlekamp-Massey. Comparé à celle d'Euclide étendu, la méthode de Berlekamp-Massey est itérative. C'est-à-dire que si l'on est en mesure de calculer plus de syndromes, l'algorithme de Berlekamp-Massey ne nécessite pas de reprendre tous les calculs depuis le début, contrairement à la méthode d'Euclide étendue. De plus, l'algorithme de décodage en liste de Wu [Wu08] expliqué en annexe B, permet d'étendre l'algorithme de Berlekamp-Massey à partir d'une relation des états internes de cet algorithme [Ber84]. Dans cette partie, on montre cette relation importante. Pour ce faire on propose de reprendre l'équation clé et de garder les mêmes notations que dans [Ber84].

Notations et contexte

On propose de représenter l'équation clé des BCH binaires au sens strict.

Notation A.1. On note

– l'erreur

$$e(X) = \sum_{i=0}^{n-1} e_i X^i,$$

– le mot reçu

$$y(X) = \sum_{i=0}^{n-1} y_i X^i.$$

Comme précédemment, on remarque que les codes BCH de capacité de correction t – vu comme codes d'annulations – permettent d'écrire :

$$\forall j \in \{1, \ldots, 2t\},\ y(\alpha^j) = \sum_{i=0}^{n-1} e_i (\alpha^j)^i.$$

On peut alors définir le polynôme syndrome coefficient par coefficient, de la manière

suivante. Soit $w = w(e)$, on a alors

$$y(\alpha^j) = \sum_{i=0}^{n-1} e_i(\alpha^j)^i = \sum_{i=1}^{w} x_i = S_j.$$

Où tous les x_i déterminent les positions des erreurs. La série syndrome se définit donc par

$$S(X) \triangleq \sum_{j=1}^{\infty} S_j X^j.$$

Dans ce contexte, le polynôme localisateur est

$$\Lambda(X) \triangleq \prod_{i=1}^{w}(1 - x_i X) = 1 + \sum_{i=1}^{w} \Lambda_i X^i.$$

On calcule alors l'équation clé suivante :

$$\begin{aligned}
S(X) &= \sum_{j=1}^{\infty} S_j X^j, \\
&= \sum_{j=1}^{\infty}\sum_{i=1}^{w} x_i^j X^j, \\
&= \sum_{i=1}^{w} \frac{x_i X}{1 - x_i X}, \\
S(X)\Lambda(X) &= \sum_{i=1}^{w} \frac{x_i X}{1 - x_i X} \prod_{j=1}^{w}(1 - x_j X), \\
&= \sum_{i=1}^{w} x_i X \prod_{j \neq i}(1 - x_j X), \\
(1 + S(X))\Lambda(X) &= \Lambda(X) + \sum_{i=1}^{w} x_i X \prod_{j \neq i}(1 - x_j X) \triangleq L(X).
\end{aligned}$$

On obtient donc l'équation de clé

$$(1 + S(X))\Lambda(X) = L(X). \tag{A.1}$$

On ne connaît que les $2t$ premiers syndromes, on obtient donc l'équation de clé suivante

$$(1 + S(X))\Lambda(X) \equiv L(X) \mod X^{2t+1}. \tag{A.2}$$

Idée de résolution itérative

Si on a un couple $\Lambda^{(i)}(X)$ et $L^{(i)}(X)$ qui vérifie

$$(1+S(X))\Lambda^{(i)}(X) \equiv L^{(i)}(X) \mod X^{i+1}, \tag{A.3}$$

alors il est naturel de se demander comment passer au rang $i+1$, c'est-à-dire comment trouver $\Lambda^{(i+1)}$ et $L^{(i+1)}$ tels que

$$(1+S(X))\Lambda^{(i+1)}(X) \equiv L^{(i+1)}(X) \mod X^{i+2}.$$

On écrit

$$(1+S(X))\Lambda^{(i)}(X) \equiv L^{(i)}(X) + \Delta^{(i)}X^{i+1} \mod X^{i+2},$$

où $\Delta^{(i)}$ représente le coefficient de degré $i+1$ du produit $(1+S(X))\Lambda^{(i)}(X)$. Si $\Delta^{(i)}$ est nul alors clairement la mise à jour est

$$\begin{cases} \Lambda^{(i+1)}(X) = \Lambda^{(i)}(X) \\ L^{(i+1)}(X) = L^{(i)}(X). \end{cases}$$

De plus, on pose deux polynômes auxiliaires $\tau^{(i)}(X)$ et $\gamma^{(i)}(X)$ qui vérifient

$$(1+S(X))\tau^{(i)}(X) \equiv \gamma^{(i)}(X) \mod X^{i+1}. \tag{A.4}$$

Alors la mise à jour des polynômes est

$$\begin{cases} \Lambda^{(i+1)}(X) = \Lambda^{(i)}(X) - \Delta^{(i)}X\tau^{(i)}(X) \\ L^{(i+1)}(X) = L^{(i)}(X) - \Delta^{(i)}X\gamma^{(i)}(X). \end{cases}$$

En effet, on a l'égalité

$$\begin{aligned} (1+S(X))\Lambda^{(i+1)}(X) &= (1+S(X))\Lambda^{(i)}(X) - (1+S(X))\Delta^{(i)}X\tau^{(i)}(X), \\ &\equiv L^{(i)}(X) - \Delta^{(i)}X\gamma^{(i)}(X) \mod X^{i+2}, \\ &\equiv L^{(i+1)} \mod X^{i+2}. \end{aligned}$$

De plus, pour éviter lors de la mise à jour que le degré de $\Lambda^{(i+1)}$ ne diminue par rapport au degré de $\Lambda^{(i)}$, on définit pour chaque i la borne $D(i)$ et la fonction booléenne $B(i)$ de la façon suivante

- $\deg \Lambda^{(i)} \leq D(i)$, avec l'égalité si $B(i) = 1$,
- $\deg \gamma^{(i)} \leq i - D(i) - 1 + B(i)$, avec l'égalité si $B(i) = 1$,
- $\deg \tau^{(i)} \leq i - D(i)$, avec l'égalité si $B(i) = 0$,
- $\deg L^{(i)} \leq D(i) - B(i)$, avec l'égalité si $B(i) = 0$.

Algorithme et relations des états internes

En suivant l'idée ci-dessus, voici le pseudo code de la méthode de Berlekamp et de Massey, algorithme 11.

Algorithme 11: Algorithme de Berlekamp-Massey

Entrées : S le polynôme syndrome
Sorties : Λ le polynôme localisateur et L le polynôme évaluateur

début
  Initialisation
  – $\Lambda^{(0)} = \tau^{(0)} = L^{(0)} = 1$,
  – $\gamma^{(0)} = 0$,
  – $D(0) = B(0) = i = 0$.
  tant que S_{i+1} *est définie* **faire**
    $\Delta^{(i)} \longleftarrow$ coefficient de degré $i+1$ de $(1+S)\Lambda^{(i)}$
    $\Lambda^{(i+1)} \longleftarrow \Lambda^{(i)} - \Delta^{(i)} X \tau^{(i)}$
    $L^{(i+1)} \longleftarrow L^{(i)} - \Delta^{(i)} X \gamma^{(i)}$
    si $(\Delta^{(i)} = 0)$ ***ou*** $(D(i) > \frac{i+1}{2})$
    ou $((\Delta^{(i)} \neq 0)$ ***et*** $(D(i) = \frac{i+1}{2})$ ***et*** $(B(i) = 0))$ **alors**
      $(D(i+1), B(i+1)) \longleftarrow (D(i), B(i))$
      $(\tau^{(i+1)}, \gamma^{(i+1)}) \longleftarrow (X\tau^{(i)}, X\gamma^{(i)})$
    sinon
      $(D(i+1), B(i+1)) \longleftarrow (i+1-D(i), 1-B(i))$
      $(\tau^{(i+1)}, \gamma^{(i+1)}) \longleftarrow (\frac{\Lambda^{(i)}}{\Delta^{(i)}}, \frac{L^{(i)}}{\Delta^{(i)}})$
    $i \longleftarrow i+1$
  retourner $(\Lambda^{(i)}, L^{(i)})$.

Pour tout i, on a

$$(1+S(X))\Lambda^{(i)}(X) \equiv L^{(i)}(X) \mod X^{i+1}, \tag{A.5}$$

qui est un invariant de boucle, ce qui prouve la correction de cette méthode. Maintenant, on sait utiliser l'algorithme de Berlekamp-Massey pour décoder jusqu'à la capacité de correction. Les relations existantes entre les états internes des variables dans la méthode de Berlekamp et Massey nous permettent de corriger encore plus d'erreurs. Cette méthode est alors un algorithme de décodage en liste qui a été proposés en 2008, longtemps après la méthode de Berlekamp et Massey. Bien que l'algorithme de décodage en liste de Wu soit expliqué en annexe B, ici on démontre les relations internes utilisées.

Théorème A.1. *Pour chaque i, on a*

$$\Lambda^{(i)}(0) = L^{(i)}(0) = 1, \tag{A.6}$$

$$(1+S(X))\Lambda^{(i)}(X) \equiv L^{(i)}(X) + \Delta^{(i)} X^{i+1} \mod X^{i+2}, \tag{A.7}$$

$$(1+S(X))\tau^{(i)}(X) \equiv \gamma^{(i)}(X) + X^{i} \mod X^{i+1}. \tag{A.8}$$

Démonstration. On montre l'égalité (A.6) par récurrence.

– Rang initial
$\Lambda^{(0)}(0) = L^{(0)}(0) = 1.$

– Rang $i+1$: on suppose l'égalité vraie au rang i.

$$\begin{aligned} \Lambda^{(i+1)}(0) &= \Lambda^{(i)}(0) - 0 = 1, \\ L^{(i+1)}(0) &= L^{(i)}(0) - 0 = 1, \end{aligned}$$

L'égalité (A.6) est donc bien vérifiée.

Montrons les égalités (A.7) et (A.8) par double récurrence.

– Rang initial

$$\begin{aligned} (1+S(X)) &\equiv 1 \mod X, \\ (1+S(X)) &\equiv 1 + \Delta^{(1)} X \mod X^2, \end{aligned}$$

par définition de $\Delta^{(1)}$ et de $S(X)$.

– Rang $i+1$: on suppose les égalités vraies au rang i.
  – Pour l'égalité (A.7)

$$\begin{aligned} (1+S)\Lambda^{(i+1)} &= (1+S)\Lambda^{(i)} - (1+S)\Delta^{(i)}\tau^{(i)}X, \\ &\equiv L^{(i)} + \Delta^{(i)}X^{i+1} - \Delta^{(i)}X\gamma^{(i)} - \Delta^{(i)}X^{i+1} \mod X^{i+2}, \\ &\equiv L^{(i)} - \Delta^{(i)}X\gamma^{(i)} \mod X^{i+2}, \\ &\equiv L^{(i+1)} \mod X^{i+2}, \\ &\equiv L^{(i+1)} + \Delta^{(i+1)}X^{i+2} \mod X^{i+3}. \end{aligned}$$

  – Pour l'égalité (A.8)
    – $\tau^{(i+1)} = X\tau^{(i)}$

$$\begin{aligned} (1+S)\tau^{(i+1)} &= (1+S)\tau^{(i)}X, \\ &\equiv \gamma^{(i)}X + X^{i+1} \mod X^{i+2}, \\ &\equiv \gamma^{(i+1)} + X^{i+1} \mod X^{i+2}. \end{aligned}$$

    – $\tau^{(i+1)} = \frac{\Lambda^{(i)}}{\Delta^{(i)}}$

$$\begin{aligned} (1+S)\tau^{(i+1)} &= \frac{(1+S)\Lambda^{(i)}}{\Delta^{(i)}}, \\ &\equiv \frac{L^{(i)}}{\Delta^{(i)}} + X^{i+1} \mod X^{i+2}, \\ &\equiv \gamma^{(i+1)} + X^{i+1} \mod X^{i+2}. \end{aligned}$$

Les égalités (A.7) et (A.8) sont bien vérifiées. □

Théorème A.2. *Pour chaque i on a*

$$L^{(i)}\tau^{(i)} - \Lambda^{(i)}\gamma^{(i)} = X^i.$$

Démonstration. En multipliant les égalités (A.7) et (A.8), on obtient

$$\begin{aligned} (1+S)\tau^{(i)}L^{(i)} &\equiv (1+S)(\Lambda^{(i)}\gamma^{(i)} + \Lambda^{(i)}(0)X^i) \mod X^{i+1}, \\ \underbrace{\tau^{(i)}L^{(i)} - \Lambda^{(i)}\gamma^{(i)}}_{A^{(i)}} &\equiv X^i \mod X^{i+1} \end{aligned}$$

Or

$$\begin{aligned} \deg A^{(i)} &\leq \max\{\deg(\tau^{(i)}L^{(i)}), \deg(\Lambda^{(i)}\gamma^{(i)})\}, \\ &\leq \max\{i - D(i) + D(i) - B(i), D(i) + i - D(i) - 1 + B(i)\}, \\ &\leq i. \end{aligned}$$

On en déduit donc

$$L^{(i)}\tau^{(i)} - \Lambda^{(i)}\gamma^{(i)} = X^i.$$

□

Le théorème suivant donne la relation qui permet de mettre en relation le vrai polynôme localisateur $\Lambda(X)$ avec les sorties de la méthode de Berlekamp et Massey, même s'il y a plus d'erreurs que la capacité de correction t.

Théorème A.3. *Soient $\Lambda(X)$ et $L(X)$ qui vérifient*
- $\Lambda(0) = 1$,
- $(1 + S(X))\Lambda(X) \equiv L(X) \mod X^{i+1}$,

alors il existe

$$\Lambda(X) = U(X)\Lambda^{(i)}(X) + V\tau^{(i)}(X), \tag{A.9}$$
$$L(X) = U(X)L^{(i)}(X) + V\gamma^{(i)}(X), \tag{A.10}$$

tels que
- $U(0) = 1$,
- $V(0) = 0$,
- $\deg U \leq D - D(i)$,
- $\deg V \leq D - i + D(i)$,

où $D = \max\{\deg(\Lambda), \deg(L)\}$.

Démonstration. – Définition de $U(X)$, pour cela on part de l'hypothèse

$$\begin{aligned}
(1+S)\Lambda &\equiv L \mod X^{i+1},\\
(1+S)\Lambda\tau^{(i)} &\equiv L\tau^{(i)} \mod X^{i+1},\\
\Lambda(\gamma^{(i)}+X^i) &\equiv L\tau^{(i)} \mod X^{i+1},\\
\Lambda\gamma^{(i)}+\Lambda(0)X^i &\equiv L\tau^{(i)} \mod X^{i+1},\\
L\tau^{(i)}-\Lambda\gamma^{(i)} &= X^i(1+U'X),\\
L\tau^{(i)}-\Lambda\gamma^{(i)} &= X^iU \quad \Longrightarrow U(0)=1.
\end{aligned} \tag{A.11}$$

De plus

$$\begin{aligned}
\deg U &\leq \max\{\deg(\tau^{(i)}L),\deg(\Lambda\gamma^{(i)})\}-i,\\
&\leq \max\{i-D(i)+D, D+i-D(i)-1+B(i)\}-i,\\
&\leq i-D(i)+D-i,\\
&\leq D-D(i).
\end{aligned}$$

– Définition de $V(X)$, on part également de l'hypothèse

$$\begin{aligned}
(1+S)\Lambda &\equiv L \mod X^{i+1},\\
(1+S)L^{(i)}\Lambda &\equiv LL^{(i)} \mod X^{i+1},\\
(1+S)L^{(i)}\Lambda &\equiv L(1+S)\Lambda^{(i)} \mod X^{i+1},\\
L^{(i)}\Lambda &\equiv L\Lambda^{(i)} \mod X^{i+1},\\
L\Lambda^{(i)}-L^{(i)}\Lambda &= V'X^{i+1},\\
L\Lambda^{(i)}-L^{(i)}\Lambda &= -VX^i \quad \Longrightarrow V(0)=0.
\end{aligned} \tag{A.12}$$

De plus

$$\begin{aligned}
\deg V &\leq \max\{\deg(L\Lambda^{(i)}),\deg(L^{(i)}\Lambda)\}-i,\\
&\leq \max\{D+D(i), D(i)-B(i)+D\}-i,\\
&\leq D+D(i)-i.
\end{aligned}$$

– En multipliant l'égalité (A.11) par $\Lambda^{(i)}$ et (A.12) par $-\tau^{(i)}$, on obtient

$$\begin{aligned}
X^i(\Lambda^{(i)}U+\tau^{(i)}V) &= \Lambda^{(i)}L\tau^{(i)}-\Lambda^{(i)}\Lambda\gamma^{(i)}-\tau^{(i)}L\Lambda^{(i)}+\tau^{(i)}L^{(i)}\Lambda,\\
&= \Lambda(\tau^{(i)}L^{(i)}-\Lambda^{(i)}\gamma^{(i)}),\\
\Lambda^{(i)}U+\tau^{(i)}V &= \Lambda \qquad \text{d'après le théorème } A.2.
\end{aligned}$$

– De même, en multipliant l'égalité (A.11) par $L^{(i)}$ et (A.12) par $-\gamma^{(i)}$, on a

$$L = UL^{(i)}+V\gamma^{(i)}.$$

□

Ici on montre une relation particulière sur les syndromes des codes BCH binaires, dûe au Frobenius, que l'on peut par la suite exploiter, aussi bien fois dans l'algorithme de Berlekamp-Massey que dans celui de Wu. Pour la suite on a besoin des notations suivantes.

Notation A.2. Soit $P(X) \in \mathbb{F}_q[X]$, on note
- $\widehat{P}(X)$ la partie paire de $P(X)$,
- $\widetilde{P}(X)$ la partie impaire de $P(X)$.

Pour illustrer les notations précédentes, on se propose de traiter un exemple.

Exemple A.3. Soit $P(X) = 1 + X + X^2 + X^5 \in \mathbb{F}_2[X]$, alors
- $\widehat{P}(X) = 1 + X^2$,
- $\widetilde{P}(X) = X + X^5$.

Dans le cas binaire, en gardant les mêmes notations on a

$$S_i = \sum_{i=1}^{e} X_i^i$$

et

$$\begin{aligned} S_{2i} &= \sum_{i=1}^{e} X_i^{2i}, \\ &= \left(\sum_{i=1}^{e} X_i^i\right)^2, \\ &= S_i^2. \end{aligned}$$

On a donc l'égalité $S^2 = \widehat{S}$.

Proposition A.4. *Soient S et y tels que $(1+S)(1+y) = 1$, alors*

$$\widehat{y} = 0 \quad \Leftrightarrow \quad \widehat{S} = S^2.$$

Démonstration. En décomposant l'hypothèse de départ on a

$$\begin{aligned} 1 &= (1 + \widehat{S} + \widetilde{S})(1 + \widehat{y} + \widetilde{y}), \\ &= \underbrace{1 + \widehat{S} + \widetilde{S} + \widehat{y} + \widehat{y}\widehat{S} + \widehat{y}\widetilde{S} + \widetilde{y} + \widetilde{y}\widehat{S} + \widetilde{y}\widetilde{S}}_{A}. \end{aligned}$$

En décomposant A en partie paire et impaire on a

$$\begin{aligned} \widehat{A} &= 1 + \widehat{S} + \widehat{y} + \widehat{y}\widehat{S} + \widetilde{y}\widetilde{S}, \\ &= (1 + \widehat{S})(1 + \widehat{y}) + \widetilde{y}\widetilde{S} \end{aligned}$$

et

$$\begin{aligned}\widetilde{A} &= \widetilde{S} + \widehat{y}\widetilde{S} + \widetilde{y} + \widetilde{y}\widehat{S},\\ &= \widetilde{S}(1+\widehat{y}) + \widetilde{y}(1+\widehat{S}).\end{aligned}$$

En calculant $(1+\widehat{S})\widehat{A} - \widetilde{S}\widetilde{A}$ on obtient

$$\begin{aligned}1+\widehat{S} &= (1+\widehat{S})\left((1+\widehat{S})(1+\widehat{y}) + \widetilde{y}\widetilde{S}\right) - \widetilde{S}^2(1+\widehat{y}) - \widetilde{S}\widetilde{y}(1+\widehat{S}),\\ &= (1+\widehat{y})(1+\widehat{S}^2 - \widetilde{S}^2),\\ \widehat{y} &= \frac{1+\widehat{S}}{1+\widehat{S}^2-\widetilde{S}^2} - 1.\end{aligned}$$

En caractéristique 2, on en déduit

$$\begin{aligned}\widehat{y} = 0 &\iff 1+\widehat{S} = 1+\widehat{S}^2+\widetilde{S}^2,\\ \widehat{y} = 0 &\iff \widehat{S} = S^2.\end{aligned}$$

□

Théorème A.5. *Si on a l'égalité suivante $\widehat{S} = S^2$ dans un corps de caractéristique 2, alors*

- $L^{(i)} = \widehat{\Lambda}^{(i)}$,
- $\gamma^{(i)} = \begin{cases} \widehat{\tau}^{(i)}, & \textit{si } i \textit{ est impaire} \\ \widetilde{\tau}^{(i)}, & \textit{si } i \textit{ est paire} \end{cases}$,
- $\Delta^{(i)} = 0$, *si i est impaire.*

Démonstration. Il est clair qu'à l'état initial $i = 0$ les égalités sont vérifiées. On suppose vraies les égalités au rang $2i$: $L^{(2i)} = \widehat{\Lambda}^{(2i)}$ et $\gamma^{(2i)} = \widetilde{\tau}^{(2i)}$.

-

$$\begin{aligned}L^{(2i+1)} &= L^{(2i)} - \Delta^{(2i)} X\gamma^{(2i)}\\ &= \widehat{\Lambda}^{(2i)} - \Delta^{(2i)} X\widetilde{\tau}^{(2i)}\\ &= \widehat{\Lambda}^{(2i)} - \Delta^{(2i)}\widehat{X\tau}^{(2i)}\\ &= \widehat{\Lambda}^{(2i+1)}.\end{aligned}$$

- $\gamma^{(2i+1)}$:
  - $\gamma^{(2i+1)} = X\gamma^{(2i)} = X\widetilde{\tau}^{(2i)} = \widehat{X\tau}^{(2i)} = \widehat{\tau}^{(2i+1)}$.
  - $\gamma^{(2i+1)} = \frac{L^{(2i)}}{\Delta^{(2i)}} = \frac{\widehat{\Lambda}^{(2i)}}{\Delta^{(2i)}} = \widehat{\tau}^{(2i+1)}$.

– $\Delta^{(2i+1)}$:

$$\begin{aligned}
(1+S)\Lambda^{(2i+1)} &\equiv \widehat{\Lambda}^{(2i+1)} + \Delta^{(2i+1)}X^{2i+2} \mod X^{2i+3},\\
\Lambda^{(2i+1)} &\equiv \widetilde{y}\widehat{\Lambda}^{(2i+1)} + \widehat{\Lambda}^{(2i+1)} + \Delta^{(2i+1)}X^{2i+2} \mod X^{2i+3},\\
\underbrace{\widetilde{\Lambda}^{(2i+1)} + \widetilde{y}\widehat{\Lambda}^{(2i+1)}}_{impaire} &\equiv \underbrace{\Delta^{(2i+1)}X^{2i+2}}_{paire} \mod X^{2i+3},
\end{aligned}$$

$\Longrightarrow \Delta^{(2i+1)} = 0.$

– $\Lambda^{(2i+2)} = \Lambda^{(2i+1)} + 0.$
– $L^{(2i+2)} = L^{(2i+1)} = \widehat{\Lambda}^{(2i+1)} = \widehat{\Lambda}^{(2i+2)}.$
– $\gamma^{(2i+2)} = X\gamma^{(2i+1)} = X\widehat{\tau}^{(2i+1)} = \widetilde{X\tau}^{(2i+1)} = \widetilde{\tau}^{(2i)}$

□

Ce qui nous permet d'en déduire un équivalent du théorème A.3 dans le cas binaire. Ce résultat est largement exploité par l'algorithme de Wu [Wu08].

Lemme A.6. *D'après les égalités (A.2), (A.4), le théorème A.3 et le théorème précédent on en déduit*

– $\Lambda = U\Lambda^{(2t)} + V\tau^{(2t)}$,

– $(1+S)\Lambda^{(2t)} \equiv \widehat{\Lambda}^{(2t)} \mod X^{2t+1}$,

– $(1+S)\tau^{(2t)} \equiv \widetilde{\tau}^{(2t)} \mod X^{2t+1}$.

Théorème A.7. *Pour chaque i, on a $\deg(\Lambda^{(i)}) = D(i)$ et $\deg(\tau^{(i)}) = i - D(i)$.*

Démonstration. Clairement à l'étape $i = 0$ les égalités sont vérifiées.
On suppose les égalités vraies.

– Si $\deg \Lambda^{(i)} \neq 1 + \deg \tau^{(i)}$

Alors $\deg \Lambda^{(i+1)} = \max\{D(i), 1 + i - D(I)\}$

– Si $\deg \Lambda^{(i+1)} = D(i)$

$$\Longrightarrow \begin{cases} \tau^{(i+1)} &= X\tau^{(i)} \Longrightarrow \deg \tau^{(i+1)} = i + 1 - \underbrace{D(i)}_{=D(i+1)},\\ D(i+1) &= D(i). \end{cases}$$

– Si $\deg \Lambda^{(i+1)} = 1 + i - D(i)$

$$\Longrightarrow \begin{cases} D(i+1) &= i + 1 - D(i),\\ \tau^{(i+1)} &= \frac{\Lambda^{(i)}}{\Delta^{(i)}} \Longrightarrow \deg \tau^{(i+1)} = D(i) = i + 1 - D(i+1). \end{cases}$$

– Si $\deg \Lambda^{(i)} = 1 + \deg \tau^{(i)}$

Alors

$$\begin{aligned} D(i) &= 1 + i - D(i), \\ i &= 2D(i) + 1 \Longrightarrow i \text{ est impaire donc } \Delta^{(i)} = 0. \end{aligned}$$

Ce qui revient à refaire le cas $\deg \Lambda^{(i+1)} = D(i)$.

□

Lemme A.8. *D'après le lemme A.6 on a*

$$(1+S)\Lambda = U\widehat{\Lambda}^{(2t)} + V\widetilde{\tau}^{(2t)}. \tag{A.13}$$

Démonstration. D'après le lemme A.6 on a

$$(1+S)\Lambda = \underbrace{U\widehat{\Lambda}^{(2t)} + V\widetilde{\tau}^{(2t)}}_{A} \mod X^{2t+1}.$$

Or, on sait que $\deg((1+S)\Lambda) = \deg(\widehat{\Lambda}) \leq 2t$ et

$$\begin{aligned} \deg(A) &\leq \max\{\deg(U\Lambda^{(2t)}), \deg(V\tau^{(2t)})\}, \\ &\leq \max\{D - D(i) + D(i), D + D(i) - i + i - D(i)\}, \\ &\leq D, \\ &\leq 2t. \end{aligned}$$

On a donc l'égalité polynomiale

$$(1+S)\Lambda = U\widehat{\Lambda}^{(2t)} + V\widetilde{\tau}^{(2t)}.$$

□

Théorème A.9. *Dans l'égalité (A.13) on a*

$$U = \widehat{U} \text{ et } V = \widetilde{V}.$$

On en déduit

$$(1+S)\Lambda = \widehat{U}\widehat{\Lambda}^{(2t)} + \widetilde{V}\widetilde{\tau}^{(2t)} \tag{A.14}$$

Démonstration. On a d'après les égalités (A.13) et (A.9)

$$\begin{aligned} \widehat{\Lambda} &= U\widehat{\Lambda}^{(2t)} + V\widetilde{\tau}^{(2t)}, \\ \Lambda &= U\Lambda^{(2t)} + V\tau^{(2t)}, \end{aligned}$$

en additionnant ces deux égalités on obtient

$$\widetilde{\Lambda} = U\widetilde{\Lambda}^{(2t)} + V\widehat{\tau}^{(2t)},$$

on a donc le système suivant à résoudre

$$(\mathcal{S})\left\{\begin{array}{rcl}\widehat{\Lambda} & = & U\widehat{\Lambda}^{(2t)}+V\widetilde{\tau}^{(2t)},\\ \widetilde{\Lambda} & = & U\widetilde{\Lambda}^{(2t)}+V\widehat{\tau}^{(2t)},\end{array}\right. \quad \text{On obtient}$$

$$U = Det^{-1}\begin{vmatrix}\widehat{\Lambda} & \widetilde{\tau}^{(2t)}\\ \widetilde{\Lambda} & \widehat{\tau}^{(2t)}\end{vmatrix} = Det^{-1}(\underbrace{\widehat{\Lambda}\widehat{\tau}^{(2t)}}_{paire}-\underbrace{\widetilde{\Lambda}\widetilde{\tau}^{(2t)}}_{paire}).$$

Lorsque le déterminant est non nul, alors U est paire.

$$V = Det^{-1}\begin{vmatrix}\widehat{\Lambda}^{(2t)} & \widehat{\Lambda}\\ \widetilde{\Lambda}^{(2t} & \widetilde{\Lambda}\end{vmatrix} = Det^{-1}(\underbrace{\widetilde{\Lambda}\widehat{\Lambda}^{(2t)}}_{impaire}-\underbrace{\widehat{\Lambda}\widetilde{\Lambda}^{(2t)}}_{impaire}).$$

Donc si Det est non nul, alors V est impaire. On calcule Det

$$Det = \underbrace{\widehat{\Lambda}^{(2t)}\widehat{\tau}^{(2t)}}_{A}-\underbrace{\widetilde{\Lambda}^{(2t)}\widetilde{\tau}^{(2t)}}_{B}.$$

On propose de calculer les degrés des parties A et B.

– Si $D(2t)$ est paire

$$\begin{array}{rcl}\deg A & = & D(2t)+2t-D(2t),\\ & = & 2t.\end{array}$$

$$\begin{array}{rcl}\deg B & \leq & D(2t)-1+2t-D(2t)-1,\\ & \leq & 2t-2.\end{array}$$

Donc Det est non nul.

– Si $D(2t)$ est impaire

$$\begin{array}{rcl}\deg A & \leq & D(2t)-1+2t-D(2t)-1,\\ & \leq & 2t-2.\end{array}$$

$$\begin{array}{rcl}\deg B & = & D(2t)+2t-D(2t),\\ & = & 2t.\end{array}$$

Donc le déterminant est non nul.

□

Annexe B

Algorithme de Wu

Wu propose un nouvel algorithme de décodage en liste pour les codes de Reed-Solomon et les codes BCH binaires [Wu08]. Il est très important pour diverses raisons : premièrement cette méthode travaille avec des multiplicités moindres, comparées à celles de Guruswami-Sudan ce qui permet de baisser la complexité ; deuxièmement il est basé sur une philosophie différente. En effet, la méthode proposée par Wu en 2008, étend l'algorithme de Berlekamp-Massey, voir annexe A, et exploite une relation entre le vrai polynôme localisateur et les sorties de l'algorithme de Berlekamp-Massey.

Principe

La méthode de décodage de Berlekamp et de Massey est une méthode de décodage unique. Alors si le poids de l'erreur est supérieure à t, la capacité de correction, les polynômes localisateur et évaluateur retournés par cette méthode, ne sont pas corrects. On appelle alors *vrais* polynômes localisateur et évaluateur, ceux associés à l'erreur. Soient $\Lambda(X)$ le vrai polynôme localisateur et $L(X)$ le vrai polynôme évaluateur alors d'après le théorème A.3 à la page 118, il existe le couple de polynômes $(U(X), V(X))$ tel que

$$\begin{aligned} \Lambda(X) &= U(X)\Lambda^{(i)}(X) + V(X)\tau^{(i)}(X), \\ L(X) &= U(X)L^{(i)}(X) + V(X)\gamma^{(i)}(X), \end{aligned}$$

et

- $U(0) = 1$,
- $V(0) = 0$,
- $\deg U \leq D - D(i)$,
- $\deg V \leq D - i + D(i)$,

où $D = \max\{\deg(\Lambda), \deg(L)\}$ et $\Lambda^{(i)}, L^{(i)}$, respectivement $\tau^{(i)}$ et $\gamma^{(i)}$, sont les sorties, respectivement les états internes de l'algorithme de Berlekamp-Massey. S'il y a trop d'erreurs, la méthode de Berlekamp-Massey n'est pas en mesure de retourner les bons polynômes localisateurs et évaluateurs. Cependant, une relation très importante lie les

sorties de cet algorithme aux vrais polynômes. La méthode proposée par Wu consiste à retrouver les coefficients de cette relation. Comme X est un facteur de $V(X)$, on obtient

$$\begin{aligned}\Lambda(X) &= U(X)\Lambda^{(i)}(X) + XV'(X)\tau^{(i)}(X),\\ \frac{\Lambda(X)}{X\tau^{(i)}(X)} &= U(X)\frac{\Lambda^{(i)}(X)}{X\tau^{(i)}(X)} + V'(X).\end{aligned}$$

La dernière ligne a l'avantage de rassembler tout les termes que l'on connaît. On sait que les racines de $\Lambda(X)$ sont des puissances inverses de α^{-j}, où α est une racine primitive n^{e} de l'unité sur $\mathbb{F}_q$. Pour calculer les coefficients de cette relation, c'est-à-dire $U(X)$ et $V'(X)$, on procède par interpolation sur les points

$$\left(\alpha^{-j}, y_j\right), \text{ où } y_j \triangleq -\frac{\Lambda^{(i)}(\alpha^{-i})}{\alpha^{-i}\tau^{(i)}(\alpha^{-i})}, \ \forall j \in \{1, \ldots, n\}.$$

On calcule alors le polynôme bivarié $Q(X,Y) \triangleq \sum_{l=0}^{\ell} Q_l(X)Y^l$ tel qu'il s'annule à tous les points (α^{-j}, y_j) avec multiplicité s. La définition de la multiplicité est rappelée dans la définition 2.2 à la page 42.

Remarque B.1. La méthode de Wu utilise donc, comme les autres algorithmes de décodage en liste, une interpolation bivariée. Cette interpolation est paramétrée par un degré maximum sur chaque $Q_l(X)$, défini par $L_\Lambda - L_{X\tau}$ – qui sont retournés par la méthode de Berlkamp et Massey. Dans l'algorithme de décodage en liste de Wu, ce degré peut être négatif. Wu suggère alors de modifier l'algorithme d'interpolation de Koetter [Koe96] pour s'adapter a ce problème. Cependant tous les algorithmes ne permettent pas de faire cela. Toutefois, en réunissant les termes de la manière suivante

$$\frac{\Lambda(X)}{\Lambda^{(i)}(X)} = U(X) + V'(X)\frac{X\tau^{(i)}(X)}{\Lambda^{(i)}(X)},$$

le degré maximum devient alors $L_{X\tau} - L_\Lambda > 0$ et les points d'interpolations deviennent

$$\left(\alpha^{-j}, y_j\right), \text{ où } y_j \triangleq -\frac{\alpha^{-j}\tau^{(i)}(\alpha^{-j})}{\Lambda^{(i)}(\alpha^{-j})}, \ \forall j \in \{1, \ldots, n\}.$$

Algorithme

Algorithme 12: Algorithme de décodage en liste de Wu

Entrées : Le mot reçu $y \in \mathbb{F}_q^n$ et le code de Reed-Solomon.
Sorties : Une liste de polynômes localisateurs et évaluateurs $\{(\Lambda, L)\}$.

début
- $S \longleftarrow$ Syndrome(y)
- $(\Lambda^{(i)}, \tau^{(i)}, L^{(i)}, \gamma^{(i)} L_\Lambda, L_{X\tau}) \longleftarrow$ Berlekamp-Massey(Syndrome(y))
- **si** *($L_\Lambda = \deg(\Lambda^{(i)})$)* ***et*** *($\Lambda^{(i)}$ possède L_Λ racines différentes non nulles)* **alors**
  - **retourner** $\{\Lambda^{(i)}\}$
- **si** *($n - k - T \geq L_\Lambda$)* ***ou*** *($L_\Lambda > T$)* **alors**
  - **retourner** *Échec*
- $Q(X, Y) \longleftarrow$ Interpolation$(\mathbf{IP}_{Wu}, \Lambda^{(i)}, \tau^{(i)}, \mathcal{C})$
- $((U_1, V_1'), \ldots, (U_\ell, V_\ell')) \longleftarrow$ Racine_YLineaire$(Q(X, Y))$
- **pour tous les** $j \in \{1, \ldots, \ell\}$ **faire**
  - $\Lambda_j(X) \longleftarrow \Lambda^{(i)}(X) U_j(X) + X V_j'(X) \tau^{(i)}$
  - $L_j(X) \longleftarrow L^{(i)}(X) U_j(X) + X V_j'(X) \gamma^{(i)}$
- **retourner** $\{(\Lambda_j, L_j)\}$.

Théorème B.1. *Pour un $RS_q[n, k, d]$ code de Reed-Solomon, cette méthode peut corriger jusqu'à la borne de Johnson générique*

$$n\left(1 - \sqrt{1 - \frac{d}{n}}\right).$$

Le rayon de décodage de cet algorithme est le même que celui de Guruswami-Sudan. Néanmoins, Wu montre également que la complexité de sa méthode est inférieure à celle de Guruswami-Sudan. De plus, Wu réussit à augmenter le rayon de décodage pour les BCH binaires. En effet, grâce au lemme A.6, page 122, on en déduit que

$$\Lambda(X) = U(X^2)\Lambda^{(i)}(X) + X^2 V'(X^2)\tau^{(i)}(X).$$

Théorème B.2. *Pour un $[n, k, d]_2$ code BCH, cette méthode peut corriger jusqu'à la borne de Johnson binaire*

$$\frac{n}{2}\left(1 - \sqrt{1 - 2\frac{d}{n}}\right).$$

Annexe C

Code source

Dans cette annexe, on présente quelques sources en un langage de calcul formel : Magma [BCP97]. On donne quelques exemples d'implémentation des algorithmes de décodage introduits précédement, elles n'ont pas la prétention d'être rapides. On commence par introduire l'algorithme d'interpolation de Koetter [Koe96], ensuite différents algorithmes de décodage pour les codes de Reed-Solomon : Welsh-Berlekamp, Sudan et Guruswami-Sudan. On termine cette annexe par l'implémentation de notre méthode proposée pour décoder en liste les codes de Goppa binaires.

C.1 Algorithme de Koetter

```
/**************************************************/
/**************************************************/
/*              Weighted_Degree                   */
/**************************************************/
/**************************************************/
/*  Compute the (1,nu)-weighted degree of  a      */
/*  bivariate polynomial.                         */
/**************************************************/
/* Input:                                         */
/*   P  bivariate polynomial                      */
/*   nu parameter of (1,nu)-weighted-degree       */
/*                                                */
/* Output:                                        */
/*   The (1,nu)-weighted degree of the bivariate */
/*   polynomial P.                                */
/**************************************************/
function Weighted_Degree(P,nu)

    R<X,Y> := Parent(P);
```

```
    list   := Terms(P,Y);
    nb     := #list;

    i:=1;
    while( IsZero(list[i]) ) do
      i +:= 1;
    end while;

    var_return := Degree(list[i],X) + (nu * (i-1));
    for j in [i+1..nb] do
      D := Degree(list[j],X) + (nu * (j-1));

      if (D ge var_return) then
        var_return := D;
      end if;
    end for;

    return(var_return);
end function;



/**************************************************/
/**************************************************/
/*          smallest_weighted_degree             */
/**************************************************/
/**************************************************/
/*  Return the index of the polynomial which has */
/*  the smallest weighted degree in the list of  */
/*  bivariate polynomials.                       */
/**************************************************/
/* Input:                                        */
/*   g     list of n bivariate polynomials       */
/*   index a subset of [1,n]                     */
/*   nu    parameter of (1,nu)-weighted-degree   */
/*                                               */
/* Output:                                       */
/*   The index of the bivariate polynomial which */
/*   has the smallest weighted degree among      */
/*   the subset index.                           */
/**************************************************/
function smallest_weighted_degree(g,index,nu)
```

```
    Deg        := Weighted_Degree(g[index[1]],nu);
    var_return := index[1];

    for i in index do
      current_deg := Weighted_Degree(g[i],nu);

      if (Deg gt current_deg) then
        Deg        := current_deg;
        var_return := i;
      end if;

    end for;

    return(var_return);
end function;

/**************************************************/
/**************************************************/
/*           Koetter_Interpolation               */
/**************************************************/
/**************************************************/
/*  Koetter's interpolation algorithm            */
/**************************************************/
/* Input:                                        */
/*   list         list of point (x_i,y_i)        */
/*   multiplicity list of multipliciti m_i       */
/*   nu           the weighted degree            */
/*   L            the Y degreee                  */
/*   X            first indeterminate of the     */
/*                bivariate polynomial ring      */
/*   Y            second indeterminate of the    */
/*                bivariate polynomial ring      */
/*                                               */
/* Output:                                       */
/*   The bivariate polynomial which passes       */
/*   throught all points (x_i,y_i) with          */
/*   multiplicity m_i. The Y degree of the       */
/*   returned polynomial is at most L and its    */
/*   weighted degree is (1,nu).                  */
/**************************************************/
function Koetter_Interpolation(list, multiplicity, nu, L, X, Y)
```

```
if (#list ne #multiplicity) then
   list;
   multiplicity;
   nu;
   L;
   "Koetter interpolation : problem of dimension in arguments";
   exit;
end if;

/* Initialization */
n := #list;
g := [ ];
for j in [0..L] do
    g[j+1] := Y^(j);
end for;

mylist := [];
mylist[1] := [];
mylist[1][1] := Parent(list[1][1]) ! 0;
mylist[1][2] := Parent(list[1][1]) ! 0;
for i in [1..n] do
   mylist[i+1] := list[i];
end for;

delta := [];
for i in [1..n] do

    for j in [0..L] do
      g[j+1] := Evaluate(g[j+1],
        <X-mylist[i][1]+mylist[i+1][1],Y-mylist[i][2]+mylist[i+1][2]>);
    end for;

    for r in [0..(multiplicity[i] - 1)] do

      for s in [0..(multiplicity[i] - 1 - r)] do

        J := [];
        for j in [0..L] do

          delta[j+1] := MonomialCoefficient(g[j+1],X^r * Y^s);


          if ( not IsZero(delta[j+1]) ) then
```

```
                J[(#J)+1] := j+1;
              end if;


           end for; // end j


          if (#J ne 0) then
            j_star := smallest_weighted_degree(g,J,nu);
            f      := g[j_star];
            delt   := delta[j_star];

            for j in J do

              if (j eq j_star) then
                g[j] := f*X;
              else
                g[j] := (delt * g[j]) - (delta[j] * f);
              end if;
           end for; // end j
         end if;
      end for; // end s
    end for; // end r
  end for; // end i

  /* Research of the smallest weighted-degree polynomial */
  j_star := smallest_weighted_degree(g,[1..(L+1)],nu);
  return(Evaluate(g[j_star],<X-list[n][1],Y-list[n][2]>));

end function;
```

C.2 Algorithmes de décodage des codes de Reed-Solomon

```
/* Degree of extension field */
m := 4;

/* Code dimension */
k := 6;

/* Code length */
n := 2^m-1;
```

```
/* Minimum distance */
d := n-k+1;

/* Correction capacity */
t := Floor((d-1) / 2);

/* Weight of error */
w := t+1;

load "koetter.mag";
K<alpha> := GF(2,m);
Supp     := [alpha^i : i in [0..n-1]];
UPR<Z>   := PolynomialRing(K);
MPR<X,Y> := PolynomialRing(K,2);


/**************************************************/
/**************************************************/
/*                  Random_Upol                   */
/**************************************************/
/**************************************************/
/* Pick randomly a polynomial of degree           */
/* strictly less than k.                          */
/**************************************************/
/* Input:                                         */
/*   ind is the indeterminate                     */
/*   k   the upper bound on the degree            */
/*                                                */
/* Output:                                        */
/*   A random univariate polynomial with          */
/*   indeterminate ind and its degree is          */
/*   strictly less than k.                        */
/**************************************************/
function Random_Upol(ind, k)
    K := CoefficientRing(ind);

    P := UPR ! 0;
    for i in [1..k] do
        P := P + Random(K)*ind^(i-1);
    end for;

    return(P);
end function;
```

```
/**************************************************/
/**************************************************/
/*                 Codeword                       */
/**************************************************/
/**************************************************/
/* Construct the Reed-Solomon codeword.           */
/**************************************************/
/* Input:                                         */
/*   Supp the support of the RS code              */
/*   Pol  the message to encode in the            */
/*        polynomial form                         */
/*                                                */
/* Output:                                        */
/*   The codeword of a RS code defined by the     */
/*   support Supp. The returned codeword is       */
/*   codeword associated to polynomial Pol.       */
/**************************************************/
```

```
function Codeword(Supp, Pol)

    codeword := [];
    n := #Supp;

    for i in [1..n] do

codeword[i] := Evaluate(Pol,Supp[i]);
    end for;

    return(Vector(codeword));
end function;
```

```
/**************************************************/
/**************************************************/
/*                Random_Error                    */
/**************************************************/
/**************************************************/
/* Construct a random error vector.               */
/**************************************************/
/* Input:                                         */
/*   field the field where is defined the         */
/*         components                             */
/*   n     the length of the vector               */
/*   w     the weight of the vector               */
/*                                                */
/* Output:                                        */
/*   A random error vector of (field)^n. Its      */
/*   Hamming weight is exactly w.                 */
/**************************************************/
function Random_Error(field, n, w)
    error_vector := [];
    error_vector := [field ! 0 : i in [1..n]];

    for i in [1..w] do
       elt := Random(field);

       while(IsZero(elt)) do
          elt := Random(field);
       end while;

       index := Random(n-1)+1;
       while(not IsZero(error_vector[index])) do
          index := Random(n-1)+1;
       end while;

       error_vector[index] := elt;
   end for;

   return(Vector(error_vector));
end function;
```

```
/**************************************************/
/**************************************************/
/*                YLinear_Roots                   */
/**************************************************/
/**************************************************/
/* Compute the Y linear roots. Let Q(X,Y) a       */
/* bivariate polynomial, a Y linear root, is a    */
/* univariate polynomial f(X) such that           */
/*  Y - f(X) | Q(X,Y)                             */
/**************************************************/
/* Input:                                         */
/*   k the upper bound on the degree of Y linear */
/*     roots                                      */
/*   Q is the bivariate polynomial                */
/*   X is the first indeterminate of the          */
/*     bivariate polynomial ring.                 */
/*   Y is the second indeterminate of the         */
/*     bivariate polynomial ring.                 */
/*   Z is the indeterminate of the univariate     */
/*     polynomial ring                            */
/*                                                */
/* Output:                                        */
/*   List of all Y linear roots of Q(X,Y) of      */
/*   degree strictly less than k.                 */
/**************************************************/
function YLinear_Roots(k, Q, X, Y, Z)

    facto     := Factorization(Q);
    card      := #facto;
    codewords := [];

    for i in [1..card] do
        candidat := facto[i][1];

        if((Degree(candidat,X) lt k) and (Degree(candidat,Y) eq 1))
        then
            coeff                   := Coefficient(candidat,Y,1);
            codewords[#codewords+1] := Evaluate(candidat,<Z,0>)/coeff;
        end if;
    end for;

    return(codewords);
end function;
```

```
/**************************************************/
/**************************************************/
/*                      WB                        */
/**************************************************/
/**************************************************/
/*  The Welsh-Berlekamp's unambiguous decoding.  */
/**************************************************/
/* Input:                                         */
/*   y    the received word                       */
/*   Supp the support of the RS code              */
/*   k    the dimension of the RS code            */
/*                                                */
/* Output:                                        */
/*   Either the codeword at Hamming distance t,   */
/*   the correction capacity, either a failure.   */
/**************************************************/
function WB (y, Supp, k)
    n    := #Supp;
    list := [[Supp[i],y[i]] : i in [1..n]];

    Q     := Koetter_Interpolation(list, [1 : i in [1..n]], k-1, 1, X, Y);
    facto := Coefficients(Q,Y);
    Pol   := Evaluate((facto[1] div facto[2]),<Z,0>);

    return(Codeword(Supp, Pol));
end function;
/**************************************************/
/**************************************************/
/*                    Sudan                       */
/**************************************************/
/**************************************************/
/*  The Sudan's list decodind algorithm.         */
/**************************************************/
/* Input:                                         */
/*   y     the received word                      */
/*   bound the maximum number of errors           */
/*   Supp  the support of the RS code             */
/*   k     the dimension of the RS code           */
/*                                                */
/* Output:                                        */
/*   Either list of codewords at Hamming          */
/*   distance less than bound, either a failure. */
/**************************************************/
```

```
function Sudan(y, bound, Supp, k)
    field      := Parent(y[1]);
    U_P_R<Z>   := PolynomialRing(field);
    M_P_R<X,Y> := PolynomialRing(field,2);

    n    := #Supp;
    list := [[Supp[i],y[i]] : i in [1..n]];

    /* Compute the maximum size of the list     */
    /* in function of bound, that is also the   */
    /* Y degree of the interpolation polynomial */
    /* Q(X,Y).                                  */
    L := 1;
    while(n-bound - 1 - L*(k-1) gt 0) do

        L := L + 1;
    end while;
    L := L-1;

    Q := Koetter_Interpolation(list, [1 : i in [1..n]], k-1, L, X, Y);

    candidat  := YLinear_Roots(k, Q, X, Y, Z);
    codewords := [];
    nb        := #candidat;
    for i in [1..nb] do

        c := Codeword(Supp, candidat[i]);
        if (Weight(y-c) le bound) then
            codewords[#codewords+1] := c;
        end if;
    end for;

    return(codewords);
end function;
```

```
/**************************************************/
/**************************************************/
/*                      GS                        */
/**************************************************/
/**************************************************/
/*  The Guruswami-Sudan's list decodind          */
/*   algorithm.                                  */
/**************************************************/
/* Input:                                        */
/*   y     the received word                     */
/*   bound the maximum number of errors          */
/*   Supp  the support of the RS code            */
/*   k     the dimension of the RS code          */
/*                                               */
/* Output:                                       */
/*   Either list of codewords at Hamming         */
/*   distance less than bound, either a failure. */
/**************************************************/
function GS (y, bound, Supp, k)
    field      := Parent(y[1]);
    U_P_R<Z>   := PolynomialRing(field);
    M_P_R<X,Y> := PolynomialRing(field,2);

    n    := #Supp;
    list := [[Supp[i],y[i]] : i in [1..n]];

    /* Compute the parameters of interpolation  */
    /* step:                                    */
    /* s the multiplicty,                       */
    /* L the Y degree.                          */
    s := 1;
    while((n-bound) le (Sqrt((n + n/s)*(k-1) + (1/s)^2))) do
        s := s + 1;
    end while;

    L := 1;
    while(s*(n-bound) - 1 - L*(k-1) gt 0) do
        L := L + 1;
    end while;
    L := L-1;

    Q := Koetter_Interpolation(list, [s : i in [1..n]], k-1, L, X, Y);
```

```
    candidat  := YLinear_Roots(k, Q, X, Y, Z);
    codewords := [];
    nb        := #candidat;
    for i in [1..nb] do
        c := Codeword(Supp, candidat[i]);
        if (Weight(y-c) le bound) then
            codewords[#codewords+1] := c;
        end if;
    end for;

    return(codewords);
end function;



P := Random_Upol(Z,k);
c := Codeword(Supp,P);
e := Random_Error(K,n,w);
y := c + e;

Sudan_bound := Ceiling(n*(1 - Sqrt(2 - 2 * d/n + 1/n^2)))-1;
GS_bound    := Ceiling(n*(1 - Sqrt(1 - d/n)))-1;


"The [",n,", ",k,", ", d,"]-Reed-Solomon code over F",2^m;
"Welsh-Berlekamp's bound\t: ",t;
"Sudan's bound\t\t: ",Sudan_bound;
"Guruswami-Sudan's bound\t: ",GS_bound;
"
Codeword\t: ",c;
"Received word\t: ",y;
"Error\t\t: ",e;
"Weight of error\t: ",w;

if(w le t) then
    "=> Welsh-Berlekamp's method used";
    Pol := [WB(y,Supp,k)];
elif(w le Sudan_bound) then
    "=> Sudan's method used";
    Pol := Sudan (y, w, Supp, k);
elif(w le GS_bound) then
    "=> Guruswami-Sudan's method used";
    Pol := GS(y, w, Supp, k);
```

```
else
    "Number of errors greater than the biggest bound :",
    w," > ",GS_bound;
    exit;
end if;
"Successfull : ",
c in Pol;
Pol;
c;

exit;
```

C.3 Algorithme de décodage des codes de Goppa binaires

```
/* Degree of Goppa polynomial */
r := 3;


/* Degree of extension field */
m := 4;



load "koetter.mag";
K         := GF(2);
KK<alpha> := GF(2,m);
UPR<Z>    := PolynomialRing(KK);


G         := IrreduciblePolynomial(KK,r);
L         := [elt : elt in KK];
n         := #L;
C         := GoppaCode(L,G);
```

```
/**************************************************/
/**************************************************/
/*                Random_Error                    */
/**************************************************/
/**************************************************/
/* Construct a random error vector.               */
/**************************************************/
/* Input:                                         */
/*   field the field where is defined the         */
/*         components                             */
/*   n     the length of the vector               */
/*   w     the weight of the vector               */
/*                                                */
/* Output:                                        */
/*   A random error vector of (field)^n. Its      */
/*   Hamming weight is exactly w.                 */
/**************************************************/
function Random_Error(field, n, w)
    error_vector := [];
    error_vector := [field ! 0 : i in [1..n]];

    for i in [1..w] do
    elt := Random(field);
    while(IsZero(elt)) do
        elt := Random(field);
    end while;

    index := Random(n-1)+1;
    while(not IsZero(error_vector[index])) do
        index := Random(n-1)+1;
    end while;

    error_vector[index] := elt;
    end for;

    return(Vector(error_vector));
end function;
```

```
/**************************************************/
/**************************************************/
/*                YLinear_Roots                   */
/**************************************************/
/**************************************************/
/* Compute the Y linear roots. Let Q(X,Y) a      */
/* bivariate polynomial, a Y linear root, is a   */
/* univariate polynomial f(X) such that          */
/*  Y - f(X) | Q(X,Y)                            */
/**************************************************/
/* Input:                                        */
/*   k the upper bound on the degree of Y linear */
/*     roots                                     */
/*   Q is the bivariate polynomial               */
/*   X is the first indeterminate of the         */
/*     bivariate polynomial ring.                */
/*   Y is the second indeterminate of the        */
/*     bivariate polynomial ring.                */
/*   Z is the indeterminate of the univariate    */
/*     polynomial ring                           */
/*                                               */
/* Output:                                       */
/*   List of all Y linear roots of Q(X,Y) of     */
/*   degree strictly less than k.                */
/**************************************************/
function YLinear_Roots(k, Q, X, Y, Z)

    facto     := Factorization(Q);
    card      := #facto;
    codewords := [];

    for i in [1..card] do

       candidat := facto[i][1];
       if((Degree(candidat,X) lt k) and (Degree(candidat,Y) eq 1)) then
          coeff                   := Coefficient(candidat,Y,1);
          codewords[#codewords+1] := Evaluate(candidat,<Z,0>)/coeff;
       end if;
    end for;

    return(codewords);
end function;
```

```
/**************************************************/
/**************************************************/
/*                 GRS_Codeword                   */
/**************************************************/
/**************************************************/
/* Construct the Generalized Reed-Solomon         */
/* codeword.                                      */
/**************************************************/
/* Input:                                         */
/*   Supp the support of the GRS code             */
/*   B    the multiplicator coefficients of the   */
/*        GRS code                                */
/*   Pol  the message to encode in the            */
/*        polynomial form                         */
/*                                                */
/* Output:                                        */
/*   The codeword of a GRS code defined by the    */
/*   support Supp and the multiplicator           */
/*   coefficients B. The returned codeword is     */
/*   codeword associated to polynomial Pol.       */
/**************************************************/
function GRS_Codeword(Supp, B, Pol)

    codeword := [];
    n        := #Supp;
    for i in [1..n] do
        codeword[i] := B[i]*Evaluate(Pol,Supp[i]);
    end for;

    return(Vector(codeword));

end function;
```

```
/**************************************************/
/**************************************************/
/*            ABC_Binary_Goppa_Code               */
/**************************************************/
/**************************************************/
/*  The method proposed to list decode the       */
/*  binary Goppa codes                           */
/**************************************************/
/* Input:                                        */
/*   y     the received word                     */
/*   bound the maximum number of errors          */
/*   L     the support of the Goppa code         */
/*   G     the Goppa polynomial                  */
/*                                               */
/* Output:                                       */
/*   Either list of codewords at Hamming         */
/*   distance less than bound, either a failure. */
/**************************************************/
function ABC_Binary_Goppa_Code(y, bound, L, G)

    if (IsSquarefree(G)) then
        return(ABC_Binary_Goppa_Code(y, bound, L, G^2));
    end if;

    n          := #L;
    kGRS       := n - Degree(G);
    Field      := Parent(L[1]);
    U_P_R<Z>   := PolynomialRing(Field);
    M_P_R<X,Y> := PolynomialRing(Field,2);

    /* Construction of the multiplicator coefficients of the */
    /* associated Generalized Reed-Solomon code.             */
    B := [];
    for i in [1..n] do
        B[i] := Evaluate(G,L[i])/&*[L[i] - L[j] : j in [1..i-1] cat [i+1..n]];
    end for;

    /* Construction of the interpolation point list. */
    list := [];
    for i in [1..n] do
        list[i]   := [L[i],y[i]/B[i]];
        list[i+n] := [L[i],(y[i]+1)/B[i]];
    end for;
```

```
    gamma := bound/n;
    delta := (1-gamma)^2 + gamma^2;


    /* Compute the parameters of interpolation  */
    /* step:                                    */
    /* s  the main factor in the multiplicty,   */
    /* Dy the Y degree.                         */
    s := 1;
    while (delta^2 le ((kGRS-1)/n)*(delta + (1/s))) do
        s := s+1;
    end while;

    mult1 := Ceiling(s*(1-gamma));
    mult2 := Ceiling(s*gamma);
    mult := [mult1 : i in [1..n]] cat [mult2 : i in [1..n]];

    Dy := 1;
    while((s*n*delta - Dy*(kGRS-1) gt 0)) do
        Dy := Dy + 1;
    end while;
    Dy := Dy-1;

    Q        := Koetter_Interpolation(list, mult, kGRS-1, Dy, X, Y);
    YRoots   := YLinear_Roots(kGRS, Q, X, Y, Z);

    codewords := [];
    for i in [1..#YRoots] do
        candidat := GRS_Codeword(L,B,YRoots[i]);
        if (Weight(y - candidat) le bound) then
            codewords[#codewords+1] := candidat;
        end if;
    end for;

    return(codewords);
end function;
```

```
bound := Ceiling(n*(1 - Sqrt(1 - (4*r+2)/n))/2)-1;
w     := bound - 1;
c     := Random(C);
e     := Random_Error(GF(2),n,w);
y     := c + e;
"The binary Goppa code :";
C;
"";
"The codeword\t\t: ",c;
"The received word\t: ",y;
"The error vector\t: ",e;
"of weight\t\t: ",Weight(e);

"";
codewords := ABC_Binary_Goppa_Code(y, w, L, G);
"Successfull :";
c in codewords;
codewords;
c;

exit;
```

Index

Bibliographie

[ABC10] Daniel Augot, Morgan Barbier, et Alain Couvreur. List-decoding of binary Goppa codes up to the binary Johnson bound. Rapport technique RR-7490, INRIA, décembre 2010.

[ABC11] Daniel Augot, Morgan Barbier, et Alain Couvreur. List-Decoding of binary Goppa codes up to the binary Johnson bound. *Dans 2011 IEEE Information Theory Workshop (IEEE ITW 2011)*, Paraty, Brésil, octobre 2011.

[ABF11] Daniel Augot, Morgan Barbier, et Caroline Fontaine. Ensuring message embedding in wet paper steganography. *Dans* Liqun Chen, éditeur. *IMA proceedings*. Lecture Notes in Computer Science, décembre 2011.

[Ale05] Michael Alekhnovich. Linear diophantine equations over polynomials and soft decoding of Reed-Solomon codes. *IEEE Trans. on Information Theory*, 51(7):2257 –2265, juillet 2005.

[AVZ00] Erik Agrell, Alexander Vardy, et Kenneth Zeger. Upper bounds for constant-weight codes. *IEEE Trans. on Information Theory*, 46(7):2373 –2395, novembre 2000.

[Bar10] Morgan Barbier. New set of codes for the maximum-likelyhood decoding problem. *Dans YACC*, octobre 2010.

[BB10a] Peter Beelen, et Kristian Brander. Efficient list decoding of a class of algebraic-geometry codes. *Advances in Mathematics of Communication*, 4(4):485–518, 2010.

[BB10b] Peter Beelen, et Kristian Brander. Key equations for list decoding of Reed-Solomon codes and how to solve them. *J. Symb. Comput.*, 45:773–786, juillet 2010.

[BB11] Morgan Barbier, et Paulo Barreto. Key reduction of McEliece's cryptosystem using list decoding. *Dans 2011 IEEE International Symposium on Information Theory Proceedings (ISIT2011)*, St. Petersbourg, Russie, juillet 2011.

[BBB+07] Elaine Barker, William Barker, William Burr, William Polk, et Miles Smid. Recommendation for key management – part 1 : General (revised). *NIST Special Publication 800-57*, (1/3):1–142, 2007.

[BC60] Raj Chandra Bose, et Dwijendra Chaudhuri. On a class of error correcting binary group codes. *Information and Computation/information and Control*, 3:68–79, 1960.

[BCGO09] Thierry Berger, Pierre-Louis Cayrel, Philippe Gaborit, et Ayoub Otmani. Reducing key length of the McEliece cryptosystem. *Dans* Bart Preneel, éditeur. *Progress in Cryptology – AFRICACRYPT 2009*, volume 5580 de *Lecture Notes in Computer Science*, pages 77–97. Springer Berlin / Heidelberg, 2009.

[BCP97] Wieb Bosma, John Cannon, et Catherine Playoust. The Magma Algebra System I : The User Language. *Journal of Symbolic Computation*, 24(3-4):235–265, octobre 1997.

[BCQ11] Morgan Barbier, Christophe Chabot, et Guillaume Quintin. On quasi-cyclic codes as a generalization of cyclic codes. En soumission, 2011.

[Ber84] Elwyn Berlekamp. *Algebraic coding theory.* Aegean Park Press, Laguna Hills, CA, USA, seconde édition, 1984.

[Ber08] Daniel Bernstein. List decoding for binary Goppa codes. `http://cr.yp.to/codes/goppalist-20110303.pdf`, 2008.

[Ber11] Daniel Bernstein. Simplified high-speed high-distance list decoding for alternant codes. mars 2011.

[BF08] Jürgen Bierbrauer, et Jessica Fridrich. Constructing good covering codes for applications in steganography. *Dans* Yun Shi, éditeur. *Transactions on Data Hiding and Multimedia Security III*, volume 4920 de *Lecture Notes in Computer Science*, pages 1–22. Springer Berlin / Heidelberg, 2008.

[BGL01] Richard Brent, Shuhong Gao, et Alan Lauder. Random Krylov spaces over finite fields. *SIAM J. Discrete Math*, 16:276–287, 2001.

[BL05] Thierry Berger, et Pierre Loidreau. How to mask the structure of codes for a cryptographic use. *Designs, Codes and Cryptography*, 35:63–79, 2005.

[BLP08] Daniel Bernstein, Tanja Lange, et Christiane Peters. Attacking and defending the McEliece cryptosystem. *Dans* Johannes Buchmann, et Jintai Ding, éditeurs. *Post-Quantum Cryptography*, volume 5299 de *Lecture Notes in Computer Science*, pages 31–46. Springer Berlin / Heidelberg, 2008.

[BLP10] Daniel Bernstein, Tanja Lange, et Christiane Peters. Smaller decoding exponents : ball-collision decoding. Cryptology ePrint Archive, Report 2010/585, 2010.

[BMVT78] Elwyn Berlekamp, Robert McEliece, et Henk Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. on Information Theory*, IT-24(3):384–386, mai 1978.

[BS08] Bhaskar Biswas, et Nicolas Sendrier. McEliece cryptosystem implementation : Theory and practice. *Dans* Johannes Buchmann, et Jintai Ding, éditeurs. *Post-Quantum Cryptography*, volume 5299 de *Lecture Notes in Computer Science*, pages 47–62. Springer Berlin / Heidelberg, 2008.

[Cay08] Pierre-Louis Cayrel. *Construction et optimisation de cryptosystèmes basés sur les codes correcteurs d'erreurs.* Thèse de doctorat, Université de Limoges, 2008.

[CC98] Anne Canteaut, et Florent Chabaud. A new algorithm for finding minimum-weight words in a linear code : application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Trans. on Information Theory*, 44(1):367 –378, janvier 1998.

[CFS01] Nicolas Courtois, Matthieu Finiasz, et Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. *Dans* Colin Boyd, éditeur. *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 de *Lecture Notes in Computer Science*, pages 157–174. Springer Berlin / Heidelberg, 2001.

[CGY08] Fengjuan Chai, Xiao-Shan Gao, et Chunming Yuan. A characteristic set method for solving Boolean equations and applications in cryptanalysis of stream ciphers. *J. Syst. Sci. Complex.*, 21(2):191–208, 2008.

[CHLL97] Gérard Cohen, Iiro Honkala, Simon Litsyn, et Antoine Lobstein. *Covering codes*, volume 54 de *North-Holland Mathematical Library.* North-Holland Publishing Co., Amsterdam, 1997.

[Coo00] C. Cooper. On the rank of random matrices. *Random Structures & Algorithms*, 16(2):209–232, 2000.

[Cra98] Ron Crandall. Some notes on Steganography, 1998. Posted on the steganography mailing list.

[CS98] Anne Canteaut, et Nicolas Sendrier. Cryptanalysis of the original McEliece cryptosystem. *Dans* Kazuo Ohta, et Dingyi Pei, éditeurs. *Advances in Cryptology — ASIACRYPT'98*, volume 1514 de *Lecture Notes in Computer Science*, pages 187–199. Springer Berlin / Heidelberg, 1998.

[DH76] Whitfield Diffie, et Martin Hellman. New directions in cryptography. *IEEE Trans. on Information Theory*, 22(6):644 – 654, novembre 1976.

[EOS06] Daniela Engelbert, Raphael Overbeck, et Arthur Schmidt. A summary of McEliece-type cryptosystems and their security. Cryptology ePrint Archive, Report 2006/162, 2006.

[Euc] Euclide. Éléments. `http://gallica.bnf.fr/`.

[FG09] Caroline Fontaine, et Fabien Galand. How Reed-Solomon codes can improve steganographic schemes. *EURASIP J. Inf. Secur.*, 2009:1–10, 2009.

[FGLS05] Jessica Fridrich, Miroslav Goljan, P. Lisonek, et David Soukal. Writing on wet paper. *Signal Processing, IEEE Transactions on*, 53(10):3923 – 3935, octobre 2005.

[FGS05] Jessica Fridrich, Miroslav Goljan, et David Soukal. Efficient wet paper codes. *Dans Information Hiding*, pages 204–218, 2005.

[FGS06] Jessica Fridrich, Miroslav Goljan, et David Soukal. Wet paper codes with improved embedding efficiency. *Information Forensics and Security, IEEE Transactions on*, 1(1):102 – 110, mars 2006.

[FJF10] Tomas Filler, Jan Judas, et Jessica Fridrich. Minimizing embedding impact in steganography using trellis-coded quantization. volume 7541, page 754105. SPIE, 2010.

[FOPT10a] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, et Jean-Pierre Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. *Dans* Henri Gilbert, éditeur. *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 de *Lecture Notes in Computer Science*, pages 279–298. Springer Berlin / Heidelberg, 2010.

[FOPT10b] Jean-Charles. Faugère, Ayoub Otmani, Ludovic Perret, et Jean-Pierre Tillich. A distinguisher for high rate McEliece cryptosystems. Eprint Report 2010/331, 2010.

[Gab05] Philippe Gaborit. Shorter keys for code based cryptography. *Dans Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, mars 2005.

[GK03] Fabien Galand, et Gregory Kabatiansky. Information hiding by coverings. *Dans Information Theory Workshop, 2003. Proceedings. 2003 IEEE*, pages 151 – 154, mars 2003.

[Gop70] Valery Goppa. A New Class of Linear Error Correcting Codes. *Problemy Peredachi Informatsii*, 6:24–30, septembre 1970.

[Gra07] Markus Grassl. Bounds on the minimum distance of linear codes and quantum codes. Online available at http ://www.codetables.de, 2007. Accessed on 2011-04-19.

[GS99] Venkatesan Guruswami, et Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. on Information Theory*, 45(6):1757–1767, 1999.

[Gur05] Venkatesan Guruswami. *List Decoding of Error-Correcting Codes : Winning Thesis of the 2002 ACM Doctoral Dissertation Competition*. Lecture Notes in Computer Science. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[Gur06] Venkatesan Guruswami. *Algorithmic results in list decoding*, volume 2. Foundations and Trends in Theoretical Computer Science, 2006.

[GV05] Venkatesan Guruswami, et Alexander Vardy. Maximum-likelihood decoding of Reed-Solomon codes is NP-hard. *Dans SODA '05 : Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 470–478, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[GW04] Markus Grassl, et Greg White. New good linear codes by special puncturings. *Dans Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, page 454, juillet 2004.

[Ham94] Richard Hamming. *The art of probability for scientists and engineers*. Westview Press, 1994. New York.

[Ham96] Bradley Hamilton. SINCGARS system improvement program (sip) specific radio improvements. *Dans Tactical Communications Conference, Proceedings of the 1996*, pages 397 –406, mai 1996.

[Hoc59] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Dans Chiffres (Paris)*, page 2 :147–156, septembre 1959.

[HSS99] A. Hedayat, Neil Sloane, et John Stufken. *Orthogonal Arrays : Theory and Applications*. Springer Verlag, New York, 1999.

[JP09] Relinde Jurrius, et Ruud Pellikaan. The extended coset leader weight enumerator. *Dans WIC*, Benelux, 2009. 30-th Symposium on Information Theory.

[Kei06] Misa Keinänen. *Techniques for solving boolean equation systems*. Thèse de doctorat, University of Helsinki, 2006.

[Ker83] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, janvier 1883.

[Ker72] A. Kerdock. A class of low-rate nonlinear binary codes. *Information and Control*, 20(2):182 – 187, 1972.

[Koe96] Ralph Koetter. *On algebraic decoding of algebraic-geometric and cyclic codes*. Thèse de doctorat, Linköping University, 1996.

[KVA03] Ralf Koetter, Alexander Vardy, et Arshad Ahmed. Efficient interpolation and factorization in algebraic soft-decision decoding of Reed-Solomon codes. *Dans Information Theory, 2003. Proceedings. IEEE International Symposium on*, page 365, juillet 2003.

[LB88] P. J. Lee, et E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. *Dans Lecture Notes in Computer Science*, pages 275–280, New York, NY, USA, 1988. Springer-Verlag New York, Inc.

[LDW94] Yuan Xing Li, Robert Deng, et Xin Mei Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Trans. on Information Theory*, 40(1):271 –273, janvier 1994.

[Leo88] Jeffrey Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. on Information Theory*, 34(5): 1354 –1359, septembre 1988.

[MB09] Rafael Misoczki, et Paulo S. L. M. Barreto. Compact McEliece keys from Goppa codes. *Dans Selected Areas in Cryptography – SAC 2009*, volume 5867 de *Lecture Notes in Computer Science*, pages 276–392. Springer, 2009.

[MB11] Carlos Munuera, et Morgan Barbier. Wet paper codes and the dual distance in steganography. *Advances in Mathematics of Communications*, 2011. À apparaître.

[McE78] Robert McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, 1978.

[McE03] Robert McEliece. The Guruswami-Sudan decoding algorithm for Reed-Solomon codes. Rapport technique, The Interplanetary Network Progress Report 42-153, avril 2003. http ://ipnpr.jpl.nasa.gov/progress_report/42-153/153F.pdf.

[MS77] Florence Jessie MacWilliams, et Neil James Alexander Sloane. *The theory of error-correcting codes. II.* North-Holland Publishing Co., Amsterdam, 1977. North-Holland Mathematical Library, Vol. 16.

[Mun94] Carlos Munuera. On the generalized Hamming weights of geometric Goppa codes. *IEEE Trans. on Information Theory*, 40(6):2092 –2099, novembre 1994.

[Mun07] Carlos Munuera. Steganography and error-correcting codes. *Signal Processing*, 87(6):1528 – 1533, 2007.

[Myk72] J. Mykkeltveit. A note on Kerdock codes. Rapport technique, Communications Systems Research Section, mars 1972. TR 32-1526.

[Nad62] Morton Nadler. A 32-point $n = 12$, $d = 5$ code. *Information Theory, IRE Transactions on*, 8(1):58, janvier 1962.

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, pages 15(2) :159–166, 1986.

[OH04] Hilarie Orman, et Paul Hoffman. *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys.* Purple Streak Development and VPN Consortium, avril 2004.

[OS09] Raphael Overbeck, et Nicolas Sendrier. Code-based cryptography. *Dans* Daniel Bernstein, Johannes Buchmann, et Erik Dahmen, éditeurs. *Post-Quantum Cryptography*, pages 95–145. Springer Berlin / Heidelberg, 2009.

[OTD10] Ayoub Otmani, Jean-Pierre Tillich, et Léonard Dallot. Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. *Mathematics in Computer Science*, 3:129–140, 2010.

[Pat75] Nicolas Patterson. The algebraic decoding of Goppa codes. *IEEE Trans. on Information Theory*, 21(2):203 – 207, mars 1975.

[Poi00] David Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. *Dans* Hideki Imai, et Yuliang Zheng, éditeurs. *Public Key Cryptography*, volume 1751 de *Lecture Notes in Computer Science*, pages 129–146. Springer Berlin / Heidelberg, 2000.

[Pre68] Franco Preparata. A class of optimum nonlinear double-error-correcting codes. *Information and Control*, pages 378–400, 1968.

[RR00] Ronny Roth, et Gitit Ruckenstein. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. *IEEE Trans. on Information Theory*, 46(1):246–257, 2000.

[RSA78] Ronald Rivest, Adi Shamir, et Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21:120–126, février 1978.

[SB10] Chris Studholme, et Ian Blake. Random matrices and codes for the erasure channel. *Algorithmica*, 56:605–620, 2010. 10.1007/s00453-008-9192-0.

[SKHN76] Yasuo Sugiyama, Masao Kasahara, Shigeichi Hirasawa, et Toshihikoko Namekawa. Further results on Goppa codes and their applications to constructing efficient binary codes. *IEEE Trans. on Information Theory*, 22(5):518 – 526, septembre 1976.

[Ste89] Jacques Stern. A method for finding codewords of small weight. *Dans* Gérard Cohen, et Jacques Wolfmann, éditeurs. *Coding Theory and Applications*, volume 388 de *Lecture Notes in Computer Science*, pages 106–113. Springer Berlin / Heidelberg, 1989.

[Sti93] Douglas Stinson. Resilient functions and large sets of orthogonal arrays. *Congressus Numericus*, 92:105–110, 1993.

[Sud97] Madhu Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180 – 193, 1997.

[Sun00] Hung-Min Sun. Enhancing the security of the McEliece public-key crypto system. *Journal of Information Science and Engineering*, 16:799–812, 2000.

[SW06] Dagmar Schönfeld, et Antje Winkler. Embedding with syndrome coding based on BCH codes. *Dans Proceedings of the 8th workshop on Multimedia and security*, MM&Sec '06, pages 214–223, New York, NY, USA, 2006. ACM.

[Tie73] Aimo Tietäväinen. On the nonexistence of perfect codes over finite fields. *SIAM J. Appl. Math.*, 24:88–96, 1973.

[TR03] Ido Tal, et Ronny Roth. On list decoding of alternant codes in the Hamming and Lee metrics. *IEEE Trans. on Information Theory*, juin 2003.

[VDVT02] Eric Verheul, Jeroen Doumen, et Henk Van Tilborg. Sloppy Alice attacks ! Adaptive chosen ciphertext attacks on the McEliece public-key cryptosystem. *Dans Information, coding and mathematics : proceedings of workshop honoring prof. Bob McEliece on his 60th birthday / ed. by Mario Blaum*, pages 99–119, Boston, 2002. Kluwer Academic Publishers.

[VL72] Jack Van Lint. A new description of the Nadler code. *IEEE Trans. on Information Theory*, 18(6):825 – 826, novembre 1972.

[VL98] Jack Van Lint. *Introduction to Coding Theory.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd édition, 1998.

[WB86] Lloyd Welch, et Elwyn Berlekamp. Error correction for algebraic block codes. (4633470), décembre 1986.

[Wes01] Andreas Westfeld. F5 - A steganographic algorithm. *Dans* Ira Moskowitz, éditeur. *Information Hiding*, volume 2137 de *Lecture Notes in Computer Science*, pages 289–302. Springer Berlin / Heidelberg, 2001.

[Wie06] Christian Wieschebrink. An attack on a modified Niederreiter encryption scheme. *Dans* Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, et Tal Malkin, éditeurs. *Public Key Cryptography - PKC 2006*, volume 3958 de *Lecture Notes in Computer Science*, pages 14–26. Springer Berlin / Heidelberg, 2006.

[Wu08] Yingquan Wu. New list decoding algorithms for Reed-Solomon and BCH codes. *IEEE Trans. on Information Theory*, 54(8):3611–3630, 2008.

[ZL73] Victor Zinoviev, et Victor Leontiev. The nonexistence of perfect codes over Galois fields. *Contr. Inform. Theory*, (2):123–132, 1973.

[ZZW08] W. Zhang, X. Zhang, et Shuozhong Wang. Maximizing steganographic embedding efficiency by combining Hamming codes and wet paper codes. *Dans Proc. of the 10th International Worksop on Information Hiding*, volume 5284 de *Lecture Notes in Computer Science*, pages 60–71. Springer-Verlag, 2008.