THÈSE

présentée à

L'ÉCOLE POLYTECHNIQUE

pour obtenir le titre de

DOCTEUR EN SCIENCES

Spécialité :

Informatique

par

Pierre LOIDREAU

Sujet de la thèse :

**Étude et Optimisation de Cryptosystèmes à
Clé Publique Fondés sur la Théorie des
Codes Correcteurs**

Soutenue le 4 mai 2001 devant le jury composé de :

Mr	Jacques	STERN	Président
Mme	Pascale	CHARPIN	Directeur
MM.	Thierry	BERGER	Rapporteurs
	Grigory	KABATIANSKY	
MM.	Ernst	GABIDULIN	Examineurs
	François	MORAIN	
	Nicolas	SENDRIER	

Table des matières

Remerciements	9
Introduction	11
Notations	17
1 Cryptosystèmes à clé publique utilisant des codes correcteurs	19
1 Présentation générale	20
1.1 Cryptosystèmes de McELIECE et de NIEDERREITER	20
1.1.1 Cryptosystème de McELIECE	21
1.1.2 Cryptosystème de NIEDERREITER	24
1.1.3 Sécurité des cryptosystèmes	26
1.2 Attaques par décodage	29
1.2.1 Attaques passives	30
1.2.2 Attaques actives	35
2 Quels codes utiliser ?	37
2.1 Codes de REED-SOLOMON généralisés	38
2.1.1 Définition et propriétés	39
2.1.2 Attaque de SIDEL'NIKOV-SHESTAKOV	41
2.2 Codes de GOPPA classiques	44
2.2.1 Définitions	44
2.2.2 Propriétés générales	49
2.2.3 Propriétés des codes de GOPPA binaires	51
2.2.4 Une approche de la cryptanalyse structurelle des systèmes basés sur les codes de GOPPA	54
3 Conclusion	56
2 Clés faibles du cryptosystème de McELIECE	57
1 Principe général des attaques structurelles	58
1.1 Classes d'équivalence de codes et groupes d'automorphismes	58
1.1.1 Généralités sur les codes	59
1.1.2 Classes d'équivalence des codes de GOPPA	61
1.2 Attaques structurelles – généralités	63

	1.2.1	Description de l'attaque structurelle	63
	1.2.2	Attaque utilisant les codes étendus	65
	1.2.3	Application à la famille des codes de GOPPA irréductibles	68
1.3		Algorithme de séparation des supports	69
	1.3.1	Invariant de codes	70
	1.3.2	Description et complexité de $\mathcal{SSA}$	71
	1.3.3	Recouvrement de la permutation entre codes équivalents	74
2		Clés faibles du système	74
	2.1	Propriétés des codes de GOPPA de polynôme généra- teur binaire	76
		2.1.1 Groupe d'automorphismes	77
		2.1.2 Sous-code idempotent d'un code de GOPPA .	77
		2.1.3 Répartition des dimensions des hulls	81
	2.2	Détection des clés faibles	82
	2.3	Attaques structurelles contre les clés faibles	84
		2.3.1 Attaque générale	84
		2.3.2 Utilisation du sous-code idempotent	86
3		Conclusion	89
3		Propriétés de codes de GOPPA particuliers et application au renforcement du cryptosystème de McELIECE	91
	1	Codes dérivés de codes de GOPPA binaires	92
		1.1 Construction des codes s -projetés	92
			1.1.1 Groupe d'automorphismes des codes de GOPPA
			1.1.2 Sous-code des mots invariants
			93
		1.2 Propriétés des codes s -projetés	96
			1.2.1 Dimension des codes s -projetés
			96
			1.2.2 Distance minimale et algorithme de décodage
			103
			1.2.3 Codes de GOPPA et codes s -projetés
			105
	2	Décodage de motifs de poids élevés	105
		2.1 t -décodabilité par tour	106
		2.2 Construction d'ensembles t -décodables par tour	108
			2.2.1 Extensions de degré 5
			108
	3	Renforcement du système de McELIECE	111
		3.1 Description du schéma	111
		3.2 Sécurité du schéma	113
		3.3 Application aux extensions de degré 5	115
	4	Conclusion	117
4		Cryptosystèmes fondés sur la métrique rang	119
	1	Théorie de la métrique rang et codes optimaux	120
		1.1 Métrique rang : définitions et propriétés	121
			1.1.1 Distance rang minimale d'un code
			122

	1.1.2	Borne de SINGLETON	124
	1.1.3	Décodage de la métrique rang	127
	1.2	Codes de GABIDULIN	128
	1.2.1	Définition et propriétés	129
	1.2.2	Algorithme de décodage	133
	1.3	Sous-codes traces des codes de GABIDULIN	139
2		Cryptosystèmes utilisant la métrique rang	146
	2.1	Le cryptosystème GPT	147
	2.1.1	Description du cryptosystème	147
	2.1.2	Rôle de la matrice de distorsion	149
	2.1.3	Attaques par décodage	152
	2.2	Les attaques de GIBSON	154
	2.2.1	Première attaque	155
	2.2.2	Seconde attaque	162
	2.3	Un cryptosystème de type NIEDERREITER fondé sur la métrique rang	169
	2.3.1	Description du système	170
	2.3.2	Encodage et décodage des messages	172
	2.3.3	Quels codes utiliser ?	174
3		Conclusion	178
Conclusion et perspectives			179
A Factorisation des trinômes sur $\mathbf{F}_3$			183
1		Préliminaires	183
	1.1	Discriminant de polynômes et factorisation	183
2		Discriminant des trinômes sur $\mathbf{F}_3$	184
3		Degré congru à 0 mod 3	185
4		Degré congru à 1 mod 3	186
	4.1	Trinôme $x^n + x^k + 1$	187
	4.2	Trinôme $x^n + x^k - 1$	187
	4.3	Trinôme $x^n - x^k + 1$	189
	4.4	Trinôme $x^n - x^k - 1$	190
5		Degré congru à 2 mod 3	191
	5.1	Trinôme $x^n + x^k + 1$	191
	5.2	Trinôme $x^n + x^k - 1$	192
	5.3	Trinôme $x^n - x^k + 1$	192
	5.4	Trinôme $x^n - x^k - 1$	192
6		Tests d'irréductibilité	193
7		Trinômes primitifs	194
B Rappels de théorie des codes correcteurs			197
1		Éléments de corps finis	197
2		Fondamentaux de théorie des codes	198

Liste des tableaux

1.1	Système de chiffrement de McELIECE	21
1.2	Système de chiffrement de NIEDERREITER	24
1.3	Algorithme trivial de résolution du problème du décodage borné par t d'un code linéaire	30
1.4	Décodage par ensembles d'information	32
1.5	Algorithme de recouvrement d'un texte clair, chiffré deux fois par le même cryptosystème de McELIECE	36
1.6	Attaque de SIDEL'NIKOV-SHESTAKOV contre un cryptosys- tème de McELIECE	45
2.1	Attaque structurelle par énumération de l'espace des clés . . .	65
2.2	Attaque structurelle sur les codes étendus	67
2.3	Dimension des hulls de codes linéaires binaires.	74
2.4	Recouvrement de la permutation entre deux codes équivalents à l'aide de SSA	75
2.5	Répartition des dimensions des hulls des codes de GOPPA de longueur 1024, de polynôme générateur binaire	82
2.6	Répartition des dimensions des hulls des codes de GOPPA de longueur 512, de polynôme générateur binaire	83
2.7	Attaque structurelle contre les <i>clés faibles</i> lorsque l'espace des clés est la famille des codes de GOPPA irréductibles de degré t	85
2.8	Attaque réduite contre les clés faibles	88
3.1	Procédure de décodage d'un code s -projeté	104
3.2	Propriétés comparées de $\Gamma(\mathbf{L}, g)$ et du code 1-projeté $\tilde{I}(\mathbf{R}, g)$ pour la longueur $n = 128$	105
3.3	Comparaison asymptotique des paramètres des codes de GOPPA et des codes s -projetés.	105
3.4	Cryptosystème de McELIECE modifié	112
4.1	Algorithme de décodage des codes de GABIDULIN	140
4.2	Système de chiffrement GPT	148
4.3	Première attaque de GIBSON – cas où le paramètre de distor- sion $t_1 = 1$	161
4.4	Seconde attaque de GIBSON : réduction 2	167

4.5	Cryptosystème de NIEDERREITER fondé sur le décodage de la métrique rang	170
A.1	Valeurs de k pour lesquelles les trinômes sur $\mathbf{F}_3$ ont un nombre impair de facteurs	195

Remerciements

J'ai passé plusieurs années formidables ! Si un jour on m'avait dit que j'aurais la chance d'échoir dans un laboratoire où le prototype du chercheur ebourriffé, mal rasé, en pull marin et en sandalettes, et marmonnant dans sa barbe de trois jours des formules insolites n'existait pas, je ne l'aurais sans doute pas cru. Pourtant, je remercie tous les membres, ex et actuels, du projet CODES, de m'avoir ouvert les yeux et aiguisé mon goût pour la recherche en sirotant tous les matins dans une salle enfumée, un café bien serré.

D'abord, un grand merci à Pascale Charpin, qui, jeune, me recueillit, tandis que j'errais sans but, de place en place à la recherche d'un lieu où faire mon stage de DEA. Malgré mon air troglodyte, elle m'invita à séjourner quelque temps pour plancher sur un sujet que Nicolas Sendrier tenait bien au chaud pour d'éventuels amateurs. Je lui sais gré également de son incommensurable sagesse, ainsi que d'avoir aiguillé mes recherches.

Et c'est là, frais émoulu de l'école, que je découvris Nicolas Sendrier autrement que par ouï-dire. Je ne saurais trop le remercier d'avoir dirigé mon stage de DEA qui, petit à petit se métamorphosa en sujet de thèse. Bien qu'ayant longtemps travaillé dessus, son algorithme de séparation des supports, restera pour longtemps encore, pour moi, un des insondables mystères de la science. D'aucuns osèrent même affirmer que ce bout de code pouvait faire le café. Cela me paraît invraisemblable !

Un grand merci à Anne Canteaut et à Daniel Augot. La première pour ses conseils avisés concernant le cryptosystème de McEliece, et le second, également pour ses conseils, mais également pour ses démonstrations émérites d'aïkido. Je le remercie aussi de m'avoir véhiculé de l'INRIA à l'université d'Orsay. De plus je suis certain qu'avec lui, l'industrie de la moutarde a de beaux jours devant elle.

Du projet je remercie tous les autres qui contribuèrent à l'animation du petit bout de table carrée le matin et le midi, à savoir Grégoire Bommier et Matthieu Brunet, qui nous manquent à tous, ainsi que Caroline Fontaine désormais à Lille, Lancelot Pecquet exilé aux USA, Antoine Valembois en short à Hawaï. Merci également à Christelle Guiziou, Christine Pourcelot, Carmen McCallum, Electra, Marion Videau, Hervé Alavoine, Francis Blanchet, Claude Carlet, Guy Chassé, Mathieu Finiasz, Fabien Galand, Frédéric Raynal, et Cédric Tavernier, aux Watchmen, pour avoir égayé le projet ces

matins bien mornes de forte pluviosité.

Je remercie Thierry Berger d'avoir accepté d'être rapporteur de ce document. Je lui suis reconnaissant de ses remarques nombreuses et des moûlts discussions qui conduisirent à l'élaboration de la dernière partie de cette thèse.

Je suis également très honoré que Grisha Kabatiansky ait bien voulu rapporter mon document de thèse, avec sa bonne humeur habituelle et son engouement pour les mots d'esprit. Hormis cela, de nombreuses discussions avec lui m'ont clarifié considérablement les idées sur divers problèmes de théorie des codes.

Je sais gré à Ernst Gabidulin d'avoir bien voulu se déplacer de Moscou afin de participer au jury de thèse. Sa gentillesse, sa patience, et ses recherches aiguisèrent mon appétit pour l'étude de la métrique rang à laquelle est dédiée le dernier chapitre de la thèse.

Merci à François Morain d'avoir accepté de faire partie du jury. Il m'honore en m'intronisant membre du club des amateurs des corps de caractéristique impaire, même si pour l'instant je n'en suis qu'au stade du noviciat.

Merci à Jacques Stern, d'une part de m'avoir enseigné les bases de la cryptographie alors que j'étais encore en DEA, et d'autre part d'avoir accepté de faire partie du jury.

Merci à Sean Coffey pour m'avoir fait découvrir les joies du hurling lors de son court passage au projet.

Je souhaite dire à Bernard Duhamel merci. Merci d'avoir relu le dernier chapitre de ma thèse et découvert un très grand nombre de coquilles, et de fautes de style. Grâce à ses multiples tuyaux, la rédaction grandement facilitée. Je suis certain qu'une équipe comme celle-là ira au fond des choses.

Enfin merci à tous ceux que j'ai oublié de remercier.

Introduction

Bienvenue, bienvenue dans le monde d’Alice et de Bob, ou de la *cryptographie à clé publique*. Alice et Bob furent conçus en 1976 par W. DIFFIE et M. HELLMAN dans [DH76], et naquirent en 1978 dans un article de R. RIVEST, A. SHAMIR, et L. ADLEMAN [RSA78]. Après quelques millénaires de règne sans partage, la cryptographie à *clé secrète* cédait du terrain à ce qui allait devenir la *cryptographie à clé publique*.

Introduction à la cryptographie à clé publique

Depuis que les hommes ont pu communiquer, ils ont eu besoin de confidentialité. Des égyptiens, en passant par les grecs et les arabes, jusqu’à nos jours, ils déployèrent des trésors d’ingéniosité pour développer des systèmes assurant à leurs utilisateurs une parfaite sécurité de communication. Jusqu’à une période récente, la sécurité de ces systèmes reposait sur une information secrète que se partageaient les utilisateurs, et qui permettait de communiquer de manière confidentielle. Pour cette raison, l’ensemble de ces systèmes est appelé *cryptographie à clé secrète*. De nos jours, avec le développement de la théorie de l’information – initiée par Claude SHANNON [Sha49] –, la cryptographie à clé secrète s’est mathématisée puis informatisée. Comme systèmes à *clé secrète*, on ne se sert plus que d’algorithmes utilisant la même clé en chiffrement et en déchiffrement et qui, pour cela, sont appelés algorithmes de *chiffrement symétriques*. Le DES, l’AES, IDEA et GOST en sont les exemples les plus célèbres. Bien que ces algorithmes soient encore beaucoup utilisés pour le chiffrement de messages en raison du très haut débit de chiffrement, ceux-ci ne satisfont plus les besoins nouveaux qui apparaissent parallèlement à l’explosion des moyens de communication très impersonnels – le développement du réseau par exemple. C’est précisément ce que firent remarquer Withfield DIFFIE et Martin HELLMAN dans [DH76] :

« Contemporary cryptography is unable to meet the requirements, in that its use would impose such severe inconveniences on the system users, as to eliminate many of the benefits of teleprocessing. »

La *cryptographie à clé publique* est donc l’ensemble des moyens qui permettent de fournir une solution aux nouveaux problèmes qui sont notamment :

- *L'identification de personnes* : avec l'apparition des moyens de communication impersonnels, comme internet, Alice veut être sûre que la personne avec laquelle elle communique ne triche pas sur son identité, c'est-à-dire que quelqu'un prétendant être «Bob» est bien Bob en réalité. Pour ce faire, elle utilise un *schéma ou protocole d'identification*. Le lecteur intéressé par ces schémas pourra se reporter au livre de Bruce SCHNEIER [Sch96, Ch. 11].
- *L'authentification de documents* : une autorité peut authentifier des documents par l'intermédiaire d'une *signature numérique*. La signature consiste à adjoindre au document un certain nombre de bits qui sont calculés en fonction du document et de l'autorité. De plus, toute personne ayant accès au document, doit pouvoir vérifier que la signature est bien celle de l'autorité. Le lecteur intéressé se reportera à [Sch96, Ch. 10].
- *La confidentialité de messages* : Alice et Bob veulent communiquer, sans qu'une personne qui intercepterait, par hasard ou intentionnellement, leur communication puisse en extraire l'information.

Chaque utilisateur d'un cryptosystème à *clé publique* dispose de :

- *Une clé privée* : c'est une quantité connue par une seule personne. Cette clé sert pour le déchiffrement des messages.
- *Une clé publique* : C'est une quantité qui est publiée, mettons dans un annuaire. Cette clé sert pour le chiffrement des messages.

Supposons qu'Alice souhaite communiquer avec Bob de manière privée. Alice trouve dans l'annuaire la *clé publique* de Bob, puis chiffre le message avec cette clé. Quand Bob reçoit le message chiffré, il utilise sa *clé privée* afin de *déchiffrer* le message et de retrouver le *texte clair*. Les deux clés ont des rôles fondamentalement différents, et c'est la raison pour laquelle on parle encore pour de tels systèmes de *cryptosystèmes asymétriques*, par opposition aux cryptosystèmes à *clé secrète* utilisant la même clé en chiffrement et en déchiffrement et qui sont également appelés *cryptosystèmes symétriques*.

Bien que le rôle de confidentialité fût initialement dévolu aux cryptosystèmes symétriques, la cryptographie asymétrique possède un avantage majeur. En effet, si n utilisateurs souhaitent communiquer par le biais d'un cryptosystème symétrique, chacun d'eux doit disposer d'une clé différente par personne du groupe. Il faut donc pouvoir gérer en tout $n(n-1)$ clés. Sachant que n peut être de l'ordre de plusieurs milliers, il faut gérer des fichiers de plusieurs millions de clés. De plus, ajouter un utilisateur au groupe n'est pas une mince affaire, puisqu'il faut alors engendrer n clés pour que le nouvel utilisateur puisse communiquer avec les autres membres du groupe, puis distribuer les nouvelles clés à tout le groupe. En revanche, dans le cas d'un *cryptosystème asymétrique*, on stocke les n clés publiques des utilisateurs dans un annuaire. Pour rajouter un utilisateur, il suffit qu'il mette sa *clé publique* dans l'annuaire.

Pour plus de détails, je renvoie les lecteurs aux livres de SCHNEIER [Sch96], de STINSON [Sti95], et au «Handbook» de cryptographie [MvOV97]. Le lecteur intéressé par l'aspect historique de la cryptographie pourra consulter le livre de David KAHN [Kah96].

Sécurité des systèmes de chiffrement à clé publique

La cryptographie à *clé publique* intervient dans nombre d'applications quotidiennes, de l'utilisation de cartes à puces en passant par la téléphonie mobile, jusqu'à la connexion d'un utilisateur à un ordinateur. Cependant, la sécurité de ce foisonnement d'applications repose dangereusement sur seulement deux problèmes «réputés» difficiles de théorie des nombres :

- La problème de la *factorisation des nombres entiers* : étant donné un produit n de deux nombres premiers p et q , retrouver les nombres p et q . Ce problème fonde la sécurité du cryptosystème RSA, voir [RSA78].
- Le problème de la *recherche du logarithme discret* : étant donné un corps fini $\mathbf{F}_q$, un élément primitif g de $\mathbf{F}_q$, et un élément y de $\mathbf{F}_q$, retrouver $x \in \mathbf{F}_q$ tel que

$$g^x = y.$$

Bien que ces deux problèmes résistent encore aux cryptanalystes, ils ne sont pas à l'abri d'une percée théorique qui mettrait en danger leur sécurité. La destinée «tragique» des cryptosystèmes dont la sécurité reposait sur le problème dit du *sac-à-dos* en est un exemple flagrant.

Le principe de ces cryptosystèmes fut introduit par MERKLE et HELLMANN en 1978 dans [MH78]. Leur sécurité fondait sa légitimité sur le problème *NP-dur* dit du *sac-à-dos* – celui-ci est décrit par exemple dans [Sch96, pp. 501]. Cependant, il s'avéra que presque toutes ses instances étaient faibles – voir [Odl90] –, et pouvaient être cryptanalysées en se servant de l'algorithme LLL, [LLL82].

Un autre inconvénient des systèmes susmentionnés est leur extrême lenteur en chiffrement et en déchiffrement, comparée à la vitesse de chiffrement et de déchiffrement des algorithmes symétriques. Par exemple, en déchiffrement, le cryptosystème RSA est plusieurs centaines de fois plus lent que l'algorithme DES. Ils ne peuvent donc pas être utilisés pour chiffrer de l'information à haut débit. Dans la pratique, on les utilise pour échanger des clés de sessions entre utilisateurs, qui servent ensuite de clés secrètes pour un algorithme symétrique – voir par exemple PGP qui utilise actuellement un protocole d'échange de clé fondé sur le problème du logarithme discret, puis se sert de l'algorithme à *clé secrète* IDEA en chiffrement.

En 1978, R. J. McELIECE présenta un système de chiffrement à clé publique dont la sécurité reposait sur le problème du *décodage borné* d'un code correcteur. La *clé privée* est constituée d'un code correcteur structuré

pour lequel on dispose d'un algorithme polynomial de décodage, et la *clé publique* est constituée d'une *matrice génératrice* du code, préalablement déstructuré. Tandis que d'autres systèmes furent cryptanalysés avec succès, celui-ci est l'un des rares à encore résister à toute cryptanalyse. Il possède deux avantages majeurs sur les cryptosystèmes fondés sur la théorie des nombres :

1. Les meilleurs algorithmes connus pour résoudre le problème du *décodage borné* d'un code sont de complexité exponentielle [CC98], tandis que les meilleurs algorithmes permettant de *factoriser des entiers* sont sous-exponentiels.
2. Sa complexité en chiffrement et en déchiffrement est équivalente à la complexité des *algorithmes symétriques*.

Évidemment, il y a des inconvénients. La taille de la *clé publique*, notamment, est de l'ordre de plusieurs centaines de milliers de bits, là où la *clé publique* d'un cryptosystème RSA a une taille de quelques centaines de bits seulement. De plus, le *taux de transmission* du cryptosystème n'est pas égal à 1, mais, dans la pratique, cela constitue un inconvénient mineur.

Déroulement de la thèse

Dans cette thèse, je m'intéresse aux systèmes de chiffrement dont la sécurité repose sur des problèmes «réputés» difficiles de théorie des *codes correcteurs*. Mon travail fut motivé, d'une part d'un point de vue théorique par l'intérêt de l'étude de la structure de certaines familles de codes correcteurs, et d'autre part, d'un point de vue pratique visant à utiliser ces propriétés structurelles afin de gommer une partie des inconvénients des systèmes de chiffrement fondés sur les codes correcteurs.

Le premier chapitre est un chapitre d'introduction aux cryptosystèmes de McELIECE et de NIEDERREITER – voir [McE78, Nie86]. La sécurité de ces deux cryptosystèmes repose sur deux problèmes équivalents de théorie des codes [LDW94, Sid94] : le problème du *décodage borné* d'un code, et le problème de la *recherche de mot de poids fixé dans le translaté d'un code*. Je décris également le fonctionnement des meilleurs algorithmes connus qui permettent de résoudre ces problèmes : les algorithmes de décodage par *ensembles d'information*. Puis, je présente deux familles de codes qui peuvent, en théorie, être utilisées dans les cryptosystèmes : les codes de REED–SOLOMON généralisés, mais surtout les codes de GOPPA – introduits dans [Gop70] – qui constituent la matière première des chapitres suivants.

Le second chapitre est dévolu à un travail commun avec Nicolas SENDRIER [LS01], sur la recherche de *clés faibles* du cryptosystème de McELIECE. Dans l'absolu, une clé faible est une *clé publique* de laquelle on extrait des informations sur la *clé privée*, autres que celles fournies par le concepteur. Dans ce chapitre, je décris une famille d'attaques – les *attaques structurelles*

– qui utilisent la structure de la famille des codes de GOPPA afin de reconstruire un décodeur de la *clé publique*. Je montre que l'on peut reconnaître des *clés faibles* en utilisant les propriétés de l'*algorithme de séparation des supports* [Sen00]. Ceci permet en outre de réduire considérablement la complexité des *attaques structurelles*.

Le troisième chapitre est dédié à l'étude des propriétés des *clés faibles* du cryptosystème de McELIECE. Je présente en détail une étude sur leur structure, et je montre dans un second temps que certaines propriétés structurelles peuvent renforcer le cryptosystème de McELIECE contre les *attaques par décodage* décrites dans le chapitre 1. Par conséquent, en faisant un compromis entre la résistance du système aux *attaques par décodage* du chapitre 1, et entre la résistance du système aux *attaques structurelles* du chapitre 2, on parvient à renforcer le cryptosystème, à taille de clé constante. Ce chapitre s'articule autour des travaux publiés dans [Loi, Loi00c].

Enfin, dans le quatrième et dernier chapitre, j'introduis d'abord la théorie de la *métrique rang* présentée par E. M. GABIDULIN dans [Gab85b]. Je présente ensuite le cryptosystème proposé par E. M. GABIDULIN, A. V. PARAMONOV, et O. V. TRETJAKOV dans [GPT91], dont l'originalité réside dans l'utilisation d'un problème de décodage pour la métrique rang. Par conséquent, les *clés publiques* de ce cryptosystème peuvent être de petite taille, de l'ordre de quelques milliers de bits. Je discute finalement de sa résistance aux attaques structurelles construites par GIBSON dans [Gib95b, Gib96]. Ce dernier chapitre décrit en particulier des travaux communs avec T. P. BERGER et E. M. GABIDULIN [BL00, GL00].

Notations

- Concernant les corps finis :
 - $\mathbf{F}_{q^m}$, le corps fini à q^m éléments.
 - σ , l'automorphisme de FROBENIUS d'une extension $\mathbf{F}_{q^m}/\mathbf{F}_q$.
 - $\langle \sigma \rangle$, le groupe engendré par σ .
 - $\gamma_1, \gamma_2, \dots, \gamma_m$, une base de $\mathbf{F}_{q^m}$ considéré comme espace vectoriel sur $\mathbf{F}_q$.
 - $\text{wt}(\mathbf{a})$, le poids de HAMMING du mot $\mathbf{a}$.
- Concernant les codes en général :
 - C , un code.
 - $\widehat{C}$, le code C étendu.
 - n, k, d , respectivement la longueur, la dimension, et la distance minimale – ou la distance rang minimale – d'un code.
 - $\mathbf{G}$, une matrice génératrice d'un code.
 - $\mathbf{H}$, une matrice de parité d'un code.
 - $\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n)$, le vecteur générateur d'un code.
- Concernant les codes de GOPPA :
 - g , le polynôme générateur.
 - t , le degré de g .
 - $\Gamma(\mathbf{L}, g)$, le code de GOPPA de vecteur générateur $\mathbf{L}$, et de polynôme générateur g .
- Concernant les systèmes de chiffrement :
 - $\mathcal{G}$, l'espace des clés.
 - $\mathbf{x}$, le *texte clair*.
 - $\mathbf{y}$, le *texte chiffré*.
 - $\mathbf{e}$, le *vecteur d'erreur*.
 - C_{pub} , le *code public*.

Chapitre 1

Cryptosystèmes à clé publique utilisant des codes correcteurs

Le principe des cryptosystèmes à *clé publique* utilisant la théorie des codes correcteurs fut introduit par McELIECE dans un rapport de recherche en 1978 [McE78], soit peu de temps après la publication de l'article de DIFFIE et de HELLMAN fondant la *cryptographie à clé publique* [DH76]. Dans ce rapport, McELIECE proposa un système de chiffrement qui utilisait comme *espace des clés* la famille des *codes de GOPPA* binaires irréductibles de degré 50 et de longueur 1024. Le principe consiste à publier un code de l'espace des clés qui a été préalablement déstructuré. La sécurité du cryptosystème repose sur le problème réputé difficile du *décodage borné*, [Var97], d'un code linéaire. Ce problème consiste à pouvoir décoder un code linéaire aléatoire jusqu'à une distance fixée. En théorie, on peut généraliser un tel système en remplaçant les *codes de GOPPA* par n'importe quelle famille de codes ayant un algorithme qui permette de résoudre le problème du décodage borné en temps polynomial. En pratique cependant, il est très difficile de déstructurer suffisamment les codes pour parvenir à masquer leur structure. En 1986, NIEDERREITER – voir [Nie86] – proposa également un cryptosystème à *clé publique* reposant sur le problème de la *recherche de mot de poids borné* dans le *translaté* d'un code.

L'objet de ce chapitre est de présenter dans un premier temps les systèmes de chiffrement de McELIECE et de de NIEDERREITER dans leurs versions les plus générales. Je montre que ces cryptosystèmes sont équivalents du point de vue de la sécurité contre les *attaques passives*. De plus, les *attaques par décodage* les plus performantes connues – voir [CC98] – imposent que la taille de la *clé publique* soit de l'ordre de plusieurs centaines de milliers de bits.

Dans une seconde partie, je présente deux familles de codes correcteurs qui peuvent jouer le rôle d'espace des clés pour un cryptosystème de McELIECE ou de NIEDERREITER : les codes de REED–SOLOMON généralisés, et les codes de GOPPA – voir [Gop70]. Je montre cependant, qu'en exploitant la structure de la *clé publique*, un attaquant peut facilement casser un crypto-

système utilisant des codes de REED–SOLOMON généralisés, tandis qu'on ne connaît pas de tel moyen contre les systèmes utilisant les codes de GOPPA.

1 Présentation générale

C'est en 1978 que R. J. McELIECE introduisit le concept de systèmes de chiffrement à clé publique utilisant la théorie des codes correcteurs [McE78]. Dans ces systèmes, la *clé publique* est constituée par la *matrice génératrice* d'un code linéaire. Quoique McELIECE ait préconisé l'utilisation de la famille des codes de GOPPA, cette démarche demeure très générale et on peut l'appliquer à n'importe quelle famille de codes qui dispose d'un algorithme polynomial de décodage. En 1986, NIEDERREITER proposa également un cryptosystème fondé sur des codes correcteurs, utilisant cette fois comme *clé publique*, une *matrice de parité* d'un code linéaire.

Dans cette section, je décris dans un premier temps les systèmes de McELIECE et de NIEDERREITER, et je montre qu'ils sont de sécurité équivalente. Dans une seconde partie, je recense les attaques contre ces cryptosystèmes, qui consistent à décoder un *message chiffré* relativement à la *clé publique*, en utilisant des algorithmes de décodage généraux.

1.1 Cryptosystèmes de McELIECE et de NIEDERREITER

Les cryptosystèmes McELIECE et de NIEDERREITER fondent leur sécurité sur les deux problèmes suivants de théorie des codes :

1. Le problème du *décodage borné* d'un code pour le système de chiffrement de McELIECE :

Problème 1 (Décodage borné par t)

Étant donnés,

- un code linéaire de longueur n , de dimension k donné par une matrice génératrice $\mathbf{G}$,
 - un mot $\mathbf{y}$ de longueur n ,
- trouver s'il existe un couple de mots $(\mathbf{x}, \mathbf{e})$ tel que

$$\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e},$$

où $\mathbf{e}$ est un mot de poids t .

2. Le problème de la *recherche d'un mot de poids borné dans un translaté* donné d'un code pour le système de chiffrement de NIEDERREITER :

Problème 2 (Recherche d'un mot de poids t d'un translaté) *Étant donnés,*

- un code linéaire de longueur n , de dimension k , donné par une matrice de parité $\mathbf{H}$,
- un mot $\mathbf{y}$ de longueur $n - k$,

- *Clé privée* : un code aléatoire Γ choisi dans $\mathcal{G}$ appelé *code caché*, et une permutation aléatoire π des coordonnées de Γ .
- *Clé publique* : une matrice génératrice aléatoire $\mathbf{G}_{\text{pub}}$ du *code public* $C_{\text{pub}} = \pi(\Gamma)$ – le code Γ dont les coordonnées sont permutées par π .

Les procédures de *chiffrement* et de *déchiffrement* sont les suivantes :

- *Chiffrement* : soit $\mathbf{x}$ un mot de longueur k . L'émetteur calcule et envoie

$$\mathbf{y} = \mathbf{x}\mathbf{G}_{\text{pub}} + \mathbf{e},$$

où $\mathbf{e}$ est un vecteur aléatoire de poids t de longueur n . $\mathbf{e}$ est appelé le *vecteur d'erreur*.

- *Déchiffrement* : le receveur calcule

$$\pi^{-1}(\mathbf{y}) = \mathbf{x}\pi^{-1}(\mathbf{G}_{\text{pub}}) + \pi^{-1}(\mathbf{e}).$$

Comme $\pi^{-1}(\mathbf{G}_{\text{pub}})$ est une matrice génératrice du *code caché* Γ , le receveur retrouve le couple $(\mathbf{x}, \pi^{-1}(\mathbf{e}))$ en appliquant au code Γ l'algorithme de résolution du problème du *décodage borné* par t .

TABLEAU 1.1 : Système de chiffrement de McELIECE

trouver s'il existe un mot $\mathbf{x}$ de longueur n et de poids t tel que

$$\mathbf{y} = \mathbf{x}\mathbf{H}^T.$$

Dans cette section, je commence par décrire les instances les plus générales des systèmes de chiffrement de McELIECE et de NIEDERREITER. Puis, je montre que, bien que les problèmes 1 et 2 paraissent différents, leur résolution est équivalente du point de vue de la complexité. Les deux systèmes de chiffrement sont donc de sécurité équivalente.

1.1.1 Cryptosystème de McELIECE

Soit $\mathcal{G}$ une famille de codes linéaires de longueur n sur $\mathbf{F}_{q^m}$ et de dimension k , pour lesquels on dispose d'un algorithme polynomial permettant de résoudre le problème 1 du *décodage borné* par t , *i.e.* pour tout code $C \in \mathcal{G}$, de matrice génératrice $\mathbf{G}$ et pour tout mot $\mathbf{y}$ tel que

$$\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e},$$

où $\mathbf{e}$ est de poids $\leq t$, alors l'algorithme retrouve le couple $(\mathbf{x}, \mathbf{e})$ en temps polynomial. La famille $\mathcal{G}$ sera appelée par la suite *espace des clés* du cryptosystème.

Le fonctionnement du cryptosystème est décrit dans le tableau 1.1.1. Le système que je propose est équivalent au système original donné par

McELIECE dans le rapport de recherche [McE78]. En effet, ce dernier propose comme *clé privée* le triplet

$$(\mathbf{G}, \mathbf{S}, \mathbf{P}),$$

où

- $\mathbf{G}$ est une matrice génératrice du *code caché* $\Gamma \in \mathcal{G}$, de taille $k \times n$,
- $\mathbf{S}$ est une matrice aléatoire inversible de taille $k \times k$,
- $\mathbf{P}$ est une matrice de permutation de taille $n \times n$.

La *clé publique* est la matrice

$$\mathbf{G}_{\text{pub}} = \mathbf{SGP}.$$

La matrice $\mathbf{G}$ constitue un moyen de décrire le *code caché* Γ . La matrice $\mathbf{P}$ est un moyen parmi d'autres de coder la permutation π des coordonnées de Γ . Donc le produit matriciel $\mathbf{GP}$ est une matrice génératrice du *code public* $\pi(\Gamma)$. Prendre une matrice aléatoire $\mathbf{S}$ constitue un moyen de décrire le choix d'une matrice aléatoire du *code public*. En effet, la matrice

$$\mathbf{SGP}$$

est une matrice génératrice du *code public*.

La version du système décrite dans le tableau 1.1.1 est donc équivalente au système original, mais elle offre plus de souplesse. Celle-ci n'impose pas la manière dont le concepteur doit stocker les données concernant la *clé privée*. Il n'est notamment pas tenu de stocker les trois matrices $\mathbf{S}$, $\mathbf{G}$ et $\mathbf{P}$ qui, compte tenu des paramètres proposés, sont de très grande taille.

Donc, la complexité en mémoire du système de chiffrement est :

- *Clé privée* : la taille de la clé privée dépend des ressources disponibles. Elle est constituée au plus des trois matrices $\mathbf{G}$ de taille $n \times k$ à coefficients dans $\mathbf{F}_{q^m}$, $\mathbf{S}$ de taille $k \times k$ à coefficients dans $\mathbf{F}_{q^m}$, et de la matrice de permutation $n \times n$ sur $\mathbf{F}_2$. Comme chaque élément de $\mathbf{F}_{q^m}$ est codé sur $m \log_2(q)$ bits, la taille de la clé privée est au plus de

$$m(nk + k^2) \log_2(q) + n^2 \text{ bits.}$$

- *Clé publique* : c'est la taille d'une matrice $k \times n$ à coefficients dans $\mathbf{F}_{q^m}$. La taille de la *clé publique* est donc égale à

$$nkm \log_2(q) \text{ bits.}$$

Les complexités du cryptosystème en chiffrement et en déchiffrement sont égales à :

- *Complexité en chiffrement* : elle est égale au coût de la multiplication d'un vecteur de longueur k sur $\mathbf{F}_{q^m}$ par une matrice de taille $k \times n$ sur $\mathbf{F}_{q^m}$ – soit nk multiplications dans $\mathbf{F}_{q^m}$ – plus le coût de la génération du *vecteur* d'erreur qui est un vecteur sur $\mathbf{F}_{q^m}$ de longueur n et de poids t . Ce coût est en général négligeable comparé au coût du produit d'un vecteur par une matrice.

- *Complexité en déchiffrement* : elle est égale au coût du calcul de $\pi^{-1}(\mathbf{y})$, plus le coût d'une utilisation de l'algorithme permettant de décoder t erreurs dans le code caché Γ .

Exemple 1 (Les codes de GOPPA)

Dans son article [McE78], McELIECE propose d'utiliser comme *espace des clés* la famille des *codes de GOPPA* binaires de longueur $n = 2^m$, de dimension k et corrigeant t erreurs. Si on prend comme algorithme de décodage, un algorithme d'EUCLIDE étendu, alors la complexité du cryptosystème en chiffrement et en déchiffrement – voir [Can96, p. 20] – est :

- *Complexité en chiffrement* : un mot de longueur n sur $\mathbf{F}_2$ tiré aléatoirement est en moyenne de poids $\lfloor n/2 \rfloor$. La complexité en moyenne d'une opération de chiffrement est donc égale à

$$W_C = nk/2 \text{ opérations binaires.}$$

Négliger le coût de la génération d'un vecteur d'erreur aléatoire de poids t est légitime. En effet, si par exemple on utilise un générateur pseudo-aléatoire utilisant un code de HUFFMAN alors la complexité de la génération d'un *vecteur d'erreur* binaire de longueur n et de poids t – voir [Sen95a] – est égale à

$$n + 3 \text{Log}_2 \binom{n}{t} \text{ opérations binaires.}$$

Cette quantité est négligeable comparée à la quantité $nk/2$.

- *Complexité en déchiffrement* : le coût du décodage d'un *vecteur d'erreur* binaire de poids t dans un code de GOPPA de longueur $n = 2^m$ et de dimension k est égal à

$$3mnt + 4m^2t^2 \text{ opérations binaires}$$

Le coût du déchiffrement d'un message est donc égal à

$$W_D \approx 3mnt + 4m^2t^2 + k^2/2 \text{ opérations binaires.}$$

Pour les paramètres originellement proposés, à savoir

$$m = 10, \quad n = 2^{10} = 1024, \quad k = 524, \quad \text{et} \quad t = 50,$$

- la taille de la *clé publique* est égale à

$$67\,072 \text{ octets,}$$

- le facteur de travail en *chiffrement* du cryptosystème est égal à

$$W_C/k = 512 \text{ opérations binaires par bit d'information,}$$

- le facteur de travail du cryptosystème en *déchiffrement* est égal à

$$W_D/k = 5101,7 \text{ opérations binaires par bit d'information.}$$

- *Clé privée* : un code aléatoire Γ choisi dans $\mathcal{G}$ appelé *code caché* et une permutation aléatoire π des coordonnées de Γ .
- *Clé publique* : une matrice de parité aléatoire $\mathbf{H}_{\text{pub}}$ du *code public* $C_{\text{pub}} = \pi(\Gamma)$ – le code Γ dont les coordonnées sont permutées par π .

Les procédures de *chiffrement* et de *déchiffrement* sont les suivantes

- *Chiffrement* : soit $\mathbf{x}$ un mot de longueur n et de poids t . L'émetteur calcule et envoie

$$\mathbf{y} = \mathbf{x}\mathbf{H}_{\text{pub}}^T.$$

- *Déchiffrement* : le receveur retrouve $\pi(\mathbf{x})$ en appliquant l'algorithme polynomial de *recherche de mots de poids inférieur à t d'un translaté* du *code caché* Γ . Puis, en appliquant la permutation π^{-1} , il retrouve $\mathbf{x}$.

TABLEAU 1.2 : Système de chiffrement de NIEDERREITER

1.1.2 Cryptosystème de NIEDERREITER

Soit $\mathcal{G}$ une famille de codes linéaires de longueur n sur $\mathbf{F}_{q^m}$ et de dimension k pour lesquels on dispose d'un algorithme polynomial permettant de résoudre le problème 2 de la *recherche d'un mot de poids t dans un translaté*, *i.e.* pour tout code $C \in \mathcal{G}$ de matrice de parité $\mathbf{H}$, et pour tout mot $\mathbf{y}$ tel que

$$\mathbf{y} = \mathbf{x}\mathbf{H}^T,$$

l'algorithme retrouve $\mathbf{x}$ de longueur n et de poids t en temps polynomial. La famille $\mathcal{G}$ sera appelée par la suite *espace des clés* du cryptosystème. Le fonctionnement de ce dernier est décrit dans le tableau 1.1.2.

La version «classique» du cryptosystème consiste à prendre comme *clé privée* le triplet

$$(\mathbf{H}, \mathbf{S}, \mathbf{P}),$$

où

- $\mathbf{H}$ est une matrice de parité du *code caché* $\Gamma \in \mathcal{G}$, de taille $(n - k) \times n$,
- $\mathbf{S}$ est une matrice aléatoire inversible de taille $(n - k) \times (n - k)$,
- $\mathbf{P}$ est une matrice de permutation de taille $n \times n$.

La *clé publique* est la matrice

$$\mathbf{H}_{\text{pub}} = \mathbf{SHP}.$$

De la même manière que dans le cas du système de MCELIECE, le produit $\mathbf{HP}$ est une matrice de parité du *code public*, et la matrice aléatoire inversible $\mathbf{S}$ permet d'en choisir une matrice de parité aléatoire.

En utilisant cette formulation, on peut évaluer la complexité en mémoire du cryptosystème de NIEDERREITER.

- *Clé privée* : si les informations sont stockées sous forme matricielle $\mathbf{H}$ de taille $(n - k) \times n$ à coefficients dans $\mathbf{F}_{q^m}$, $\mathbf{S}$ de taille $k \times k$ à

coefficients dans $\mathbf{F}_{q^m}$, et $\mathbf{P}$ de taille $n \times n$ dans $\mathbf{F}_2$, alors la taille de la clé privée est égale à

$$(n(n-k) + (n-k)^2)m \text{Log}_2(q) + n^2 \text{ bits.}$$

- *Clé publique* : la matrice $\mathbf{H}_{\text{pub}}$ est une matrice de taille $(n-k) \times k$ à coefficients dans $\mathbf{F}_{q^m}$. Cependant, le concepteur peut ne publier que la forme systématique

$$\mathbf{H}_{\text{syst}} = (\mathbf{I} \mid \mathbf{R})$$

de $\mathbf{H}_{\text{pub}}$, c'est-à-dire l'unique matrice qui vérifie

$$\mathbf{H}_{\text{syst}} = \mathbf{S}'\mathbf{H}_{\text{pub}},$$

où $\mathbf{S}'$ est une matrice de taille $(n-k) \times (n-k)$, inversible et à coefficients dans $\mathbf{F}_{q^m}$. En effet, quiconque connaît $\mathbf{H}_{\text{pub}}$ peut faire les opérations de réduction afin d'obtenir la forme systématique $\mathbf{H}_{\text{syst}}$. La forme systématique ne fait donc pas partie du secret. Le concepteur peut donc ne publier que la matrice $\mathbf{R}$ qui est à coefficients dans $\mathbf{F}_{q^m}$, de taille $(n-k) \times k$. La taille de la clé publique est alors égale à

$$m(n-k)k \text{Log}_2(q) \text{ bits.}$$

Les complexités du cryptosystème en chiffrement et en déchiffrement du sont égales à :

- *Complexité en chiffrement* : elle est égale au coût de la multiplication d'un vecteur de longueur n sur $\mathbf{F}_{q^m}$ par une matrice de taille $n \times (n-k)$, c'est-à-dire égale à

$$n(n-k) \text{ multiplications dans } \mathbf{F}_{q^m}.$$

- *Complexité en déchiffrement* : elle est égale au coût d'une utilisation de l'algorithme de *recherche d'un mot de poids t dans un translaté du code caché* Γ , plus le coût du calcul de la permutation inverse appliquée au mot retrouvé.

Tandis que le *taux de transmission* du cryptosystème de McELIECE est trivialement égal à k/n , celui du système de NIEDERREITER est un peu plus délicat à évaluer. Le *taux de transmission* correspond au rapport entre le nombre de bits d'information qu'on transmet effectivement et le nombre de bits d'information qu'on pourrait transmettre, si l'opération de chiffrement était surjective.

Dans le cas du cryptosystème de NIEDERREITER, il est donc égal à

$$\tau = \frac{\text{Log}_q \binom{n}{t}}{n-k}.$$

Exemple 2

Considérons la famille des codes de GOPPA binaires de longueur $n = 2^m$, de dimension k et pour lesquels on dispose d'un algorithme *ad hoc* permettant de résoudre le problème 1. Alors, on peut affiner les calculs de complexité en chiffrement et en déchiffrement et on obtient – voir [Can96, p. 26]

- *Complexité en chiffrement* : en utilisant le fait que la clé publique est mise sous forme systématique, la complexité de chiffrement est égale à

$$W_C = \frac{(n-k)kt}{n} \text{ opérations binaires}$$

en négligeant les termes de second ordre.

- *Complexité en déchiffrement* : en négligeant les termes de second ordre, le coût de la recherche d'un mot de poids t dans un translaté d'un code, est égal à

$$W_D = 4m^2t^2 + mn(2t + 1) + \frac{(n-k)^2}{2} \text{ opérations binaires.}$$

Pour les paramètres originellement proposés, à savoir

$$n = 2^{10} = 1024, \quad k = 524, \quad t = 50,$$

on obtient :

- *Taille de la clé publique* : si on choisit de ne publier que la matrice de redondance, la taille de la *clé publique* est égale à 262kbits.
- *Complexité en chiffrement* : 50,1 opérations binaires par bit d'information.
- *Complexité en déchiffrement* : 7863,3 opérations binaires par bit d'information.
- *Taux de transmission* : 56,81%.

1.1.3 Sécurité des cryptosystèmes

Les cryptosystèmes de McELIECE et de NIEDERREITER fondent leur sécurité sur les problèmes 1 et 2, respectivement du *décodage borné* d'un code et de la *recherche d'un mot de poids borné dans un translaté* d'un code. Bien que ces deux problèmes soient de formulation distincte, du point de vue de la complexité, ils sont en fait équivalents, [LDW94]

Proposition 1 (Équivalence des cryptosystèmes)

- *Il existe un algorithme en temps polynomial qui, à partir de la résolution du problème 1 pour un code C , permet de résoudre le problème 2 pour C .*
- *Réciproquement, il existe un algorithme en temps polynomial qui, à partir de la résolution du problème 2 pour un code C , permet de résoudre le problème 1.*

Preuve

La preuve se décompose en deux étapes.

1. Supposons que l'on sache résoudre le problème 1 du *décodage borné* par t d'un code C de longueur n et de dimension k , de matrice de génératrice $\mathbf{G}$, *i.e.* pour tout mot $\mathbf{y}$ à distance t du code C , on retrouve le couple $(\mathbf{x}, \mathbf{e})$ tel que

$$\mathbf{y} = \mathbf{xG} + \mathbf{e}$$

où $\mathbf{e}$ est de poids t .

Soit $\mathbf{H}$ une matrice de parité du code C , *i.e.* $\mathbf{GH}^T = 0$, et soit $\mathbf{y}$ un mot tel que

$$\mathbf{y} = \mathbf{xH}^T, \quad (1.1)$$

où $\mathbf{x}$ est de longueur $(n - k)$ et de poids t .

Supposons, sans perte de généralité, que la forme systématique de la matrice $\mathbf{H}$ est égale à

$$\mathbf{H}_{\text{syst}} = (\mathbf{I} \mid \mathbf{R}),$$

c'est-à-dire qu'il existe une matrice $\mathbf{S}$ inversible de taille $(n - k) \times (n - k)$ telle que

$$\mathbf{H}_{\text{syst}} = \mathbf{SH}_{\text{pub}}.$$

Si tel n'est pas le cas, par des permutations adéquates sur les colonnes, on peut s'y ramener. Alors le mot

$$\mathbf{y}' = (\mathbf{y} (\mathbf{S}^T)^{-1}, \underbrace{0, \dots, 0}_{k \text{ fois}})$$

de longueur n vérifie l'équation

$$\mathbf{y}'\mathbf{H} = \mathbf{y} = \mathbf{xH}^T.$$

Donc les mots $\mathbf{y}'$ et $\mathbf{x}$ sont dans le même translaté du code C , c'est-à-dire qu'il existe un mot $\mathbf{c} \in C$ tel que

$$\mathbf{y}' = \mathbf{c} + \mathbf{x},$$

où $\mathbf{x}$ est de poids t . Comme on sait résoudre le problème du décodage borné par t du code C , on connaît le mot $\mathbf{x}$. On passe ainsi de la solution du problème de la *recherche de mots de poids inférieur à t dans un translaté* de C à la solution du problème du *décodage borné* par t de C en $O((n - k)^3)$ opérations.

2. Supposons que l'on sache résoudre le problème 2 de la *recherche d'un mot de poids t dans un translaté* du code C donné par une matrice de parité $\mathbf{H}$, *i.e.* pour tout mot $\mathbf{y}$ tel que

$$\mathbf{y} = \mathbf{xH}^T,$$

on sait retrouver le mot $\mathbf{x}$.

Soit $\mathbf{y}$ un mot tel que

$$\mathbf{y} = \mathbf{xG} + \mathbf{e},$$

En évaluant le produit de $\mathbf{y}$ à droite par la matrice de parité $\mathbf{H}$ du code C , on a

$$\mathbf{yH}^T = \mathbf{x} \underbrace{\mathbf{GH}^T}_0 + \mathbf{eH}^T = \mathbf{eH}^T,$$

où $\mathbf{e}$ est de poids t . Comme on sait résoudre le problème de la recherche d'un mot de poids borné par t dans un translaté de C , on sait retrouver $\mathbf{e}$. On passe ainsi de la solution du problème du décodage borné par t du code C à la solution du problème de la recherche de mots de poids inférieur à t dans un translaté de C en $O(n(n - k))$ opérations.

■

D'un point de vue de la théorie de la complexité, les systèmes de chiffrement de McELIECE et de NIEDERREITER sont donc de sécurité équivalente. Les problèmes 1 et 2 de théorie des codes sont réputés difficiles bien qu'ils ne soient pas prouvés NP-durs et se rapprochent des problèmes suivants de théorie des codes.

Problème 3 (Décodage complet d'un code linéaire)

Étant donnés,

- un code linéaire C de longueur n ,

- un mot $\mathbf{y}$ de longueur n ,

trouver un mot $\mathbf{c}_0 \in C$ qui tel que

$$\text{wt}(\mathbf{c}_0 - \mathbf{y}) = \text{Min}_{\mathbf{c} \in C} (\text{wt}(\mathbf{c}) - \mathbf{y})$$

où wt désigne le poids de HAMMING.

Problème 4 (Distance minimale) Étant donné un code linéaire C , déterminer la distance minimale de C .

Le problème du *décodage complet* d'un code fut prouvé NP-dur en 1978 par BERLEKAMP, McELIECE et VAN TILBORG dans l'article [BMvT78]. Le problème de la recherche de la distance minimale d'un code fut prouvé NP-dur beaucoup plus récemment par VARDY dans [Var97]. Dans cet article, ce dernier écrit que, selon lui, les problèmes 1 et 2 sont NP-durs. Cependant, l'important est de savoir que ces problèmes sont difficiles en pratique et que les meilleurs algorithmes permettant de les résoudre, dont je présente le fonctionnement dans la section 1.2.1 sont de complexité exponentielle. En effet, l'exemple du *sac-à-dos* est édifiante à ce sujet. Ce cryptosystème fut introduit en 1978 par MERKLE et HELLMAN dans [MH78]. Quoique la sécurité de celui-ci repose sur le problème NP-dur appelé problème du *sac-à-dos* – voir [Sch96, p. 501], presque toutes les instances sont faibles, et peuvent être cassées avec l'algorithme LLL, voir [LLL82].

Bien que la sécurité générale des cryptosystèmes de McELIECE et de NIEDERREITER repose sur des problèmes réputés difficiles, les à côtés des systèmes peuvent être utilisés pour la cryptanalyse.

- *Système de McELIECE* : celui-ci se sert d'un générateur pseudo-aléatoire supposé parfait, qui engendre des *vecteurs d'erreurs* de longueur n et

de poids t . Cependant, concevoir un générateur pseudo-aléatoire sans biais exploitable par un attaquant constitue une gageure.

- *Système de NIEDERREITER* : dans le système de NIEDERREITER on chiffre un vecteur de poids t . Cependant, encoder un message donné en blocs de mots de poids constant égal à t , et d'une manière *cryptologiquement sûre* est difficile. Nicolas SENDRIER proposa une solution dans [Sen95a] qui utilisait les propriétés des codes de HUFFMAN. Cependant, cet encodage n'est pas très satisfaisant. En effet, l'espace des mots encodés, non seulement ne décrit pas l'espace des tous les mots de poids t possibles – réduisant ainsi le taux de transmission du système –, mais les mots encodés ne sont pas tous équiprobables, ce qui offre des biais exploitables par un cryptanalyste.

1.2 Attaques par décodage

On peut concevoir deux types d'attaques contre les cryptosystèmes de McELIECE et de NIEDERREITER.

1. *Attaques structurelles* : une attaque structurelle contre le cryptosystème de McELIECE ou de NIEDERREITER consiste à retrouver un algorithme de décodage du *code public* en exploitant sa structure. Réussir une *attaque structurelle* revient en général à retrouver un code équivalent au *code public* dont on connaît un algorithme polynomial permettant de résoudre le problème du *décodage borné*. Connaissant un tel algorithme, l'attaquant peut décrypter tous les messages interceptés comme s'il était lui-même le concepteur du cryptosystème. Cette attaque dépend exclusivement de la structure de l'*espace des clés* utilisé. Dans la section 2, je montre que certaines familles de codes sont très fortement contre-indiquées [SS92, Sen98]. J'étudierai de manière plus générale les *attaques structurelles* dans le chapitre 2.
2. *Attaques par décodage* : le principe des *attaques par décodage* consiste à décoder le *chiffré* intercepté relativement au *code public*, en utilisant des algorithmes de décodage généraux. La complexité de ces attaques ne dépend que de la longueur, de la dimension et de la *distance minimale* des codes qui constituent l'*espace des clés*.

Parmi les *attaques par décodage*, je distingue deux sous-familles.

- *Les attaques passives* : l'attaquant est passif et se contente de décoder les chiffrés interceptés relativement au *code public*. Dans la section précédente, j'ai montré que les systèmes de McELIECE et de NIEDERREITER étaient équivalents contre ce type d'attaques.
- *Les attaques actives* : l'attaquant est actif et s'autorise à faire des opérations sur plusieurs *chiffrés*, ou même à demander à l'émetteur de renvoyer le même message plusieurs fois. Contre ces attaques, les cryptosystèmes de McELIECE et de NIEDERREITER ne sont pas équivalents.

<p><i>Données :</i></p> <ol style="list-style-type: none"> 1. Une matrice génératrice $\mathbf{G}$ d'un code C sur $\mathbf{F}_{q^m}$. 2. Un paramètre entier t. 3. Un mot $\mathbf{y}$ de longueur n sur $\mathbf{F}_{q^m}$ à distance t du code C. <p><i>Résultat :</i> le couple $(\mathbf{x}, \mathbf{e})$ solution du problème du <i>décodage borné par t</i>, i.e. $(\mathbf{x}, \mathbf{e})$ vérifient</p> $\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e},$ <p>où $\mathbf{e}$ est de poids t.</p> <p><i>Procédure :</i></p> <p>Pour $\mathbf{e}$ décrivant l'ensemble des vecteurs de longueur n et de poids t faire</p> <ol style="list-style-type: none"> 1. Calculer $\mathbf{y} - \mathbf{e}$. 2. Si $\mathbf{y} - \mathbf{e} \in C$, alors sortir de la boucle, <p>Déterminer le mot $\mathbf{x}$ tel que</p> $\mathbf{y} - \mathbf{e} = \mathbf{x}\mathbf{G}.$
--

TABLEAU 1.3: Algorithme trivial de résolution du problème du décodage borné par t d'un code linéaire

Dans cette section, je présente d'abord la famille d'algorithmes la plus efficace permettant de résoudre le problème du *décodage borné*. Ce sont les algorithmes de décodage par *ensembles d'information*. Dans un second temps, je décris une *attaque active* contre le cryptosystème de McELIECE et montre qu'elle ne s'applique pas à la version NIEDERREITER du cryptosystème.

1.2.1 Attaques passives

Dans la section 1.1.3, on a vu que la sécurité des systèmes de chiffrement de McELIECE et de NIEDERREITER reposait sur des problèmes de complexité équivalente. Je ne considérerai donc ici que les algorithmes de résolution du problème du *décodage borné par t* de codes, puisque les algorithmes de *résolution du problème de la recherche de mots de poids t dans un translaté* d'un code s'en déduisent en temps polynomial. Afin d'évaluer la complexité des algorithmes, je considère des codes de longueur n et de dimension k sur un alphabet $\mathbf{F}_{q^m}$, de *vecteur générateur* $\mathbf{L}$.

L'algorithme le plus simple pour résoudre le problème du décodage borné consiste à énumérer les mots de longueur n dans $\mathbf{F}_{q^m}$ et de poids t . Cet algorithme est décrit dans le tableau 1.3.

La complexité de l'algorithme est très fortement exponentielle. En effet,

l'espace des mots de longueur n et de poids t sur $\mathbf{F}_{q^m}$ est de taille

$$(q^m - 1)^t \binom{n}{t}.$$

Si on prend un code binaire de longueur $n = 1024$ avec $t = 50$, l'algorithme doit décrire un ensemble de

$$\binom{1024}{50} \approx 2^{284} \text{ éléments.}$$

Il existe une famille d'algorithmes probabilistes dont l'efficacité moyenne est bien meilleure que celle de l'*algorithme trivial*. Ils constituent la famille des algorithmes de *décodage par ensembles d'information*. Le principe consiste à tirer aléatoirement des positions du *vecteur générateur* du code en supposant que ces positions ne sont pas corrompues par le *vecteur d'erreur*. Actuellement, le meilleur algorithme de décodage par ensemble d'information est une variante de l'algorithme de LEE-BRICKELL proposé par Anne CANTEAUT et Florent CHABAUD et décrit en détail dans l'article [CC98].

Définition 1 (Ensemble d'information) *Soit C un code linéaire de vecteur générateur $\mathbf{L}$, de matrice génératrice $\mathbf{G}$. Un ensemble d'information I est un sous-ensemble de $\mathbf{L}$ tel que :*

La matrice $\mathbf{G}_I$ de taille $k \times k$ formée des colonnes de $\mathbf{G}$ étiquetées par les éléments de I est inversible.

Un *ensemble d'information* d'un code est un paramètre important. Il indique que les positions des mots d'un code prises par paquet de k n'ont pas toutes la même importance. En effet, k positions tirées aléatoirement ne forment pas toujours un ensemble d'information. Cette propriété est intrinsèque au code puisque, quelle que soit $\mathbf{G}'$ une matrice génératrice du code, la matrice $\mathbf{G}'_I$, formées des colonnes de $\mathbf{G}'$ indexées par les éléments de I est inversible.

Désormais I désignera un *ensemble d'information* du code C et J désignera le complémentaire de I dans le vecteur générateur $\mathbf{L}$ d'un code, *i.e.*

$$\mathbf{L} = I \cup J.$$

Pour toute matrice $\mathbf{G}$ et tout sous-ensemble I de $\mathbf{L}$, on note

$$\mathbf{G} = (\mathbf{G}_I \mid \mathbf{G}_J)$$

où $\mathbf{G}_I$ est la sous-matrice de $\mathbf{G}$ constituée des colonnes étiquetées par I . Le principe des algorithmes de *décodage par ensembles d'information* est décrit dans le tableau 1.4.

Théorème 1 ([McE78]) *Soit C un code sur $\mathbf{F}_{q^m}$ de longueur n , de dimension k et de vecteur générateur $\mathbf{L}$. Alors l'algorithme décrit dans le tableau*

Données :

1. Une matrice génératrice $\mathbf{G}$ d'un code C .
2. Un paramètre t .
3. Un mot $\mathbf{y}$ de longueur n sur $\mathbf{F}_{q^m}$ à distance t du code C .

Résultat : le couple $(\mathbf{x}, \mathbf{e})$ solution du problème du *décodage borné par t* , i.e. $(\mathbf{x}, \mathbf{e})$ vérifient

$$\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e},$$

où $\mathbf{e}$ est un mot de poids t .

Procédure :

Répéter

1. Tirer aléatoirement un ensemble d'information I du code engendré par la matrice $\mathbf{G}$. Soit J le complémentaire de I dans le *vecteur générateur* du code.
2. Calculer $\mathbf{R} = \mathbf{G}_I^{-1}\mathbf{G}_J$.
3. Écrire $\mathbf{y} = (\mathbf{y}_I \mid \mathbf{y}_J)$, calculer le poids de

$$\mathbf{y}_J - \mathbf{y}_I\mathbf{R}.$$

Calculer $\mathbf{e}_J = \mathbf{y}_J - \mathbf{y}_I\mathbf{R}$.

Jusqu'à ce que le poids de $\mathbf{e}_J$ soit inférieur ou égal à t .

Retourner $\mathbf{e} = (0 \mid \mathbf{e}_J)$.

Déterminer le mot $\mathbf{x}$ tel que

$$\mathbf{y} - \mathbf{e} = \mathbf{x}\mathbf{G}.$$

TABLEAU 1.4 : Décodage par ensembles d'information

1.4, fournit une solution du problème 1 du décodage borné par t du code C en $O(k^3/P)$ multiplications dans $\mathbf{F}_{q^m}$ en moyenne où

$$P = \frac{\binom{n-t}{k}}{\binom{n}{k}}.$$

Preuve

Soit C un code linéaire de longueur n , de dimension k , de vecteur générateur $\mathbf{L}$. Soit I un ensemble d'information de C . Soit

$$\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e}. \quad (1.2)$$

Si J désigne le complémentaire de I dans le vecteur générateur $\mathbf{L}$, alors on désigne par $\mathbf{y}_I$, respectivement par $\mathbf{y}_J$ les positions du mot $\mathbf{y}$ étiquetées par I , respectivement par J . L'équation (1.2) se réécrit alors

$$\mathbf{y} = (\mathbf{y}_I \mid \mathbf{y}_J) = \mathbf{x}(\mathbf{G}_I \mid \mathbf{G}_J) + (\mathbf{e}_I \mid \mathbf{e}_J)$$

où $\mathbf{G}_I$ désigne la matrice inversible de taille $k \times k$ formée des colonnes étiquetées par I de la matrice $\mathbf{G}$, et $\mathbf{G}_J$ désigne la matrice formée des colonnes de $\mathbf{G}$ étiquetées par J . On obtient le système d'équations suivant :

$$\begin{cases} \mathbf{e}_I = \mathbf{y}_I - \mathbf{x}\mathbf{G}_I, \\ \mathbf{e}_J = \mathbf{y}_J - \mathbf{x}\mathbf{G}_J. \end{cases}$$

Donc, si l'ensemble d'information I ne contient pas de positions d'erreur – i.e. $\mathbf{e}_I = 0$ –, comme $\mathbf{G}_I$ est inversible d'après la définition 1, le système s'écrit

$$\begin{cases} \mathbf{y}_I = \mathbf{x}\mathbf{G}_I, \\ \mathbf{e}_J = \mathbf{y}_J - \mathbf{y}_I\mathbf{G}_I^{-1}\mathbf{G}_J. \end{cases}$$

Donc le vecteur

$$(0 \mid \mathbf{y}_J - \mathbf{y}_I\mathbf{G}_I^{-1}\mathbf{G}_J)$$

est la solution $\mathbf{e}$ cherchée du problème de *décodage borné par t* du code C .

Sans raffiner outre mesure, la probabilité que k éléments de $\mathbf{L}$ tirés aléatoirement parmi n n'étiquettent aucune des t positions du vecteur d'erreur est égale à

$$P = \frac{\binom{n-t}{k}}{\binom{n}{k}}.$$

Partant, le nombre moyen d'itérations effectuées par l'algorithme est égal à $1/P$. Comme, à chaque itération, le coût principal provient d'une inversion de matrice $k \times k$ à coefficients dans $\mathbf{F}_{q^m}$, la complexité de l'algorithme est en moyenne $O(k^3/P)$ multiplications dans $\mathbf{F}_{q^m}$.

■

Exemple 3

Si on prend les paramètres

$$n = 1024, \quad k = 524, \quad t = 50,$$

l'algorithme décrit dans le tableau 1.4 fait en moyenne 2^{53} itérations, ce qui est nettement mieux que l'algorithme de décodage décrit dans le tableau 1.3 qui, pour les mêmes paramètres, fait 2^{283} itérations en moyenne.

Variantes de l'algorithme : La description complète des algorithmes de décodage par *ensembles d'information* dépasse largement le cadre de cette thèse. Le lecteur intéressé pourra se reporter à la thèse d'Anne CANTEAUT [Can96, Ch. 4], qui décrit très précisément plusieurs variantes permettant d'améliorer la complexité de l'algorithme présenté dans le tableau 1.4. Les complexités sont évaluées dans le cas de codes binaires.

Je me contenterai de mentionner les idées mises en jeu. L'attaque brute de fonderie proposée dans le tableau 1.4 n'est pas la plus efficace. En effet, dans le cadre des cryptosystèmes de McELIECE et de NIEDERREITER, on applique ces algorithmes à des codes de grande taille – la dimension vaut quelques centaines. Le coût de l'inversion d'une matrice de taille $k \times k - O(k^3)$ multiplications – intervenant dans l'algorithme 1.4 est alors non négligeable.

Les variantes proposées [LB88, Leo88, Ste89, CC98] s'appuient sur des études probabilistes permettant de réduire la complexité de l'attaque en élaborant un compromis entre le nombre d'*ensembles d'information* à tirer en moyenne et le coût des inversions matricielles. À chaque itération de l'algorithme, on s'autorise à choisir des positions erronées dans l'ensemble d'information. Cela revient à introduire un certain nombre d'*effacements* paramétrés par un entier p et permet d'optimiser la complexité de l'algorithme en fonction de la longueur du code, de sa dimension, et du nombre d'erreurs t .

Afin de prévenir les attaques par décodage, la taille des clés des cryptosystèmes de McELIECE et NIEDERREITER doit être importante. Si on prend les paramètres

$$n = 2^{10} = 1024, \quad k = 512, \quad t = 50,$$

le meilleur algorithme de *décodage par ensembles d'information* décrit dans [CC98] donne une complexité de 2^{64} opérations binaires. En comparant cette complexité aux puissances de calcul disponibles à l'heure actuelle, cela commence à devenir un peu juste. Pourtant, la taille de la *clé publique* du cryptosystème de McELIECE est déjà de 524kbits, tandis que la taille de la clé dans le cryptosystème de NIEDERREITER pour ces mêmes paramètres est de 250kbits.

La taille des clés nécessaires pour que les systèmes soient sûrs contre les algorithmes de décodage les plus performants constitue donc un inconvénient majeur des systèmes de chiffrement utilisant des codes correcteurs comme *espace des clés*.

1.2.2 Attaques actives

Les *attaques actives* sont particulières. Elles brisent en effet la symétrie entre les systèmes de chiffrement de McELIECE et de NIEDERREITER, ne s'appliquant qu'au premier. Elles se servent du fait que le cryptosystème de McELIECE n'est pas déterministe, c'est-à-dire qu'un même *texte clair* peut être chiffré de deux manières différentes, à cause du générateur pseudo-aléatoire engendrant les *vecteurs d'erreurs*. Du point de vue de la sécurité, cette propriété est problématique.

En effet, supposons que le *texte clair* $\mathbf{x}$ ait été chiffré de deux manières différentes,

$$\begin{cases} \mathbf{y}_1 = \mathbf{x}\mathbf{G}_{\text{pub}} + \mathbf{e}_1, \\ \mathbf{y}_2 = \mathbf{x}\mathbf{G}_{\text{pub}} + \mathbf{e}_2, \end{cases}$$

où $\mathbf{e}_1$ et $\mathbf{e}_2$ sont deux *vecteurs d'erreurs* distincts de poids t . Si on évalue la différence des deux *chiffrés*, on obtient le mot

$$\mathbf{y}_1 - \mathbf{y}_2 = \mathbf{e}_1 - \mathbf{e}_2$$

qui est de poids $\leq 2t$. On peut estimer la probabilité que deux mots de longueur n pris dans un alphabet à q^m éléments diffèrent d'un poids inférieur ou égal à $2t$. La probabilité qu'un mot $\mathbf{c}$ de longueur n sur $\mathbf{F}_{q^m}$ soit de poids i est égale à

$$P(\text{wt}(\mathbf{c}) = i) = \frac{(q^m - 1)^i \binom{n}{i}}{q^{mn}}.$$

Soit $\mathbf{c}_1$ et $\mathbf{c}_2$ deux mots de longueur n à coefficients dans $\mathbf{F}_{q^m}$ tirés aléatoirement et indépendamment. Alors la probabilité que le poids de HAMMING de $\mathbf{c}_1 - \mathbf{c}_2$ soit inférieur à $2t$ où t est un paramètre entier donné est égale à

$$P(\text{wt}(\mathbf{c}_1 - \mathbf{c}_2) \leq 2t) \leq \frac{\sum_{i=2t+1}^{4t} P(\text{wt}(\mathbf{c}) = i)}{q^{mn}},$$

soit encore

$$P(\text{wt}(\mathbf{c}_1 - \mathbf{c}_2) \leq 2t) \leq \frac{\sum_{i=0}^{4t} (q^m - 1)^i \binom{n}{i}}{q^{mn}}. \quad (1.3)$$

Si l'on suppose que deux chiffrés $\mathbf{y}_1$ et $\mathbf{y}_2$ sont idéalement représentés par deux variables aléatoires indépendantes, alors la probabilité que le poids de la différence $\mathbf{y}_1 - \mathbf{y}_2$ soit inférieur à $2t$ vérifie également l'inégalité (1.3). Donc, la probabilité que les deux *chiffrés* interceptés $\mathbf{y}_1$ et $\mathbf{y}_2$ correspondent au même *clair* est supérieure à

$$1 - \frac{\sum_{i=0}^{4t} (q^m - 1)^i \binom{n}{i}}{q^{mn}}.$$

Exemple 4

Si je reprends les paramètres de l'exemple 1, à savoir,

$$\mathbf{F}_{q^m} = \mathbf{F}_2, \quad n = 2^{10} = 1024, \quad t = 50,$$

alors la probabilité que deux chiffrés $\mathbf{y}_1$ et $\mathbf{y}_2$ dont la différence des poids est inférieure à $2t$, correspondent au même *texte clair* est supérieure à $1 - 10^{-89}$.

Données :

1. La *clé publique* $\mathbf{G}_{\text{pub}}$ de taille $k \times n$.
2. Deux mots $\mathbf{y}_1$ et $\mathbf{y}_2$ tels que

$$\begin{cases} \mathbf{y}_1 = \mathbf{x}\mathbf{G}_{\text{pub}} + \mathbf{e}_1, \\ \mathbf{y}_2 = \mathbf{x}\mathbf{G}_{\text{pub}} + \mathbf{e}_2, \end{cases}$$

où $\mathbf{e}_1$ et $\mathbf{e}_2$ sont de longueur n et de poids t .

Résultat : le texte clair $\mathbf{x}$.

Procédure :

Répéter

1. Calculer $\mathbf{y}_1 - \mathbf{y}_2$.
2. Tirer aléatoirement un ensemble d'information I du *code public* parmi les éléments du *vecteur générateur* étiquetant les positions nulles de $\mathbf{y}_1 - \mathbf{y}_2$. Soit J le complémentaire de I dans le *vecteur générateur*.
3. Calculer

$$\mathbf{e} = \mathbf{y}_1 + \mathbf{y}_I \mathbf{G}_I \mathbf{G},$$

où $\mathbf{y}_I = (\mathbf{y}_I \mid \mathbf{y}_J)$.

Jusqu'à ce que le poids de $\mathbf{e}$ soit inférieur ou égal à t .

Retourner $\mathbf{x} = \mathbf{y}_I \mathbf{G}_I$.

TABLEAU 1.5: Algorithme de recouvrement d'un texte clair, chiffré deux fois par le même cryptosystème de McELIECE

Une fois qu'un attaquant a établi que les deux *chiffrés* $\mathbf{y}_1$ et $\mathbf{y}_2$ correspondent au même texte *clair*, celui-ci récupère un supplément d'information sur les positions corrompues de $\mathbf{y}_1$. Cette information permet de réduire le nombre d'itérations de l'algorithme de *décodage par ensembles d'information*. La procédure à suivre est décrite dans le tableau 1.5.

Proposition 2 *Si l'on connaît deux chiffrés différents d'un même texte clair par un système de McELIECE dont le code public est de longueur n , de dimension k corrigeant t erreurs, l'algorithme (1.5) retrouve le texte clair en effectuant en moyenne*

$$\frac{\binom{n-2t+2(t^2/n)}{k}}{\binom{n-2t+(t^2/n)}{k}} \text{ itérations.}$$

Preuve

Le nombre moyen de tirages aléatoires à effectuer est égal à $1/P$, où P désigne la probabilité qu'un ensemble de k éléments choisis parmi les

éléments du vecteur générateur étiquetant les positions non nulles de $\mathbf{y}_1 - \mathbf{y}_2$ corresponde à des positions non nulles du vecteur d'erreur $\mathbf{e}_1$. Cette probabilité P est égale à

$$P = \frac{\binom{n-2t+(t^2/n)}{k}}{\binom{n-2t+2(t^2/n)}{k}}$$

■

Exemple 5

En reprenant les paramètres de l'exemple 1,

$$n = 1024, \quad k = 524, \quad t = 50,$$

il suffit de faire en moyenne 3 itérations pour retrouver le texte clair.

Il existe d'autres familles d'*attaques actives* – voir [KI01]–, qui toutes reposent sur le fait que le chiffrement avec un cryptosystème de MCELIECE n'est pas déterministe. Cependant toutes peuvent être évitées en utilisant plutôt la version NIEDERREITER du cryptosystème dont le chiffrement est déterministe. Le cryptosystème de NIEDERREITER présente donc de fait un avantage certain comparé au cryptosystème de MCELIECE.

2 Quels codes utiliser ?

Dans la section 1, j'ai présenté le principe de fonctionnement des systèmes de chiffrement de MCELIECE et de NIEDERREITER. Leur sécurité repose sur un problème réputé difficile de théorie des codes, le problème du *décodage borné* d'un code aléatoire. Cependant, j'ai supposé, que la famille de codes constituant l'*espace des clés* était «parfaite», c'est-à-dire qu'elle avait toutes les caractéristiques d'une famille de codes aléatoires. En pratique cela signifie que les *attaques structurelles* sont impossibles, et que l'attaquant doit se contenter de décrypter les messages interceptés en utilisant des algorithmes de décodages généraux.

Cette hypothèse est néanmoins utopique en général. En effet, on ne connaît d'une part, qu'un petit nombre de familles de codes ayant des algorithmes de décodage en temps polynomial, et d'autre part l'existence même d'un tel algorithme impose que les codes en questions soient structurés. Tout le problème revient donc à savoir si les opérations de déstructuration du *code caché* opérées dans le cadre des cryptosystèmes sont suffisamment inexploitable pour un attaquant. Parfois, celui-ci parvient à utiliser cette structure pour reconstruire un décodeur du *code public*, cassant ainsi le système.

Dans cette section, je décris deux familles de codes disposant d'algorithmes de décodage en temps polynomial. Néanmoins, quoique très proches dans leur construction, l'une est à proscrire dans la conception d'un cryptosystème tandis que l'autre semble résistante. Dans une première partie,

je présente la famille des codes de REED–SOLOMON généralisés. Comme ils sont *MDS*, ils ont, à longueur et à dimension donnée, la plus grande distance minimale atteignable. En regard des *attaques passives* utilisant des algorithmes de décodage généraux, ils offrent la meilleure sécurité théorique. Mais SIDEL’NIKOV et SHESTAKOV ont montré qu’un attaquant pouvait retrouver un décodeur du *code public* en temps polynomial – voir [SS92]. Dans un second temps, je présente la famille des codes de GOPPA. MCELIECE suggéra son utilisation pour le cryptosystème. Quoiqu’ils dérivent des codes de REED–SOLOMON généralisés, ils sont demeurés jusqu’à présent inviolés.

Dans la suite et sauf mention contraire, je considère des codes de longueur n pris sur un corps $\mathbf{F}_{q^m}$. Soit

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n)$$

où $\alpha_i \in \mathbf{F}_{q^m}$ un étiquetage d’un certain nombre d’éléments de $\mathbf{F}_{q^m}$. Tout mot $\mathbf{a}$ de longueur n sur $\mathbf{F}_{q^m}$ sera étiqueté par le vecteur $\mathbf{L}$, appelé *vecteur générateur* du mot $\mathbf{a}$, *i.e.*

$$\mathbf{a} = (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n})$$

Définition 2 (Polynôme localisateur) Soit $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n})$ un mot de longueur n sur $\mathbf{F}_{q^m}$ de vecteur générateur $\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n)$. Le polynôme localisateur de $\mathbf{a}$ est le polynôme $f_{\mathbf{a}}$ à coefficients dans $\mathbf{F}_{q^m}$ tel que

$$f_{\mathbf{a}}(x) = \prod_{i=1, a_{\alpha_i} \neq 0}^n (x - \alpha_i). \quad (1.4)$$

2.1 Codes de REED-SOLOMON généralisés

Les *codes de REED-SOLOMON généralisés* – dits *codes GRS* – sont *MDS*. Ce sont des codes linéaires de longueur n , de dimension k et de distance minimale d dont les paramètres atteignent la borne de SINGLETON, *i.e.*

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

À dimension et longueur donnée ce sont les codes qui ont la plus grande *distance minimale* possible. À taille de clé donnée, ce sont donc les codes qui offrent la sécurité optimale contre les *attaques par décodage* pour le cryptosystème de MCELIECE ou de NIEDERREITER. Les codes GRS disposent de plus d’un algorithme de décodage en temps polynomial utilisant l’algorithme d’EUCLIDE étendu. Ils peuvent donc être utilisés comme *espace des clés* d’un cryptosystème de MCELIECE et de NIEDERREITER. SIDEL’NIKOV et SHESTAKOV ont démontré cependant que ces codes étaient si structurés, qu’on pouvait retrouver un décodeur du *code public* en temps polynomial.

Dans une première partie, je présente la définition des codes GRS, ainsi que certaines de leur propriétés, qui me permettront de décrire ensuite le déroulement de l’*attaque de SIDEL’NIKOV–SHESTAKOV*.

2.1.1 Définition et propriétés

Définition 3 (Codes GRS) Soit $\mathbf{v} = (v_1, v_2, \dots, v_n)$ un vecteur de longueur n dans $\mathbf{F}_{q^m}^*$ et soit $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ un vecteur de longueur n dans $\mathbf{F}_{q^m}$ dont les α_i sont distincts deux à deux. Le code de longueur n , de vecteur générateur α et de matrice génératrice

$$\mathbf{G} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ v_1\alpha_1 & v_2\alpha_2 & \cdots & v_n\alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ v_1\alpha_1^{k-1} & v_2\alpha_2^{k-1} & \cdots & v_n\alpha_n^{k-1} \end{pmatrix} \quad (1.5)$$

est noté $\text{GRS}_k(\alpha, \mathbf{v})$. L'ensemble des codes $\text{GRS}_k(\alpha, \mathbf{v})$ est appelée famille des codes de REED–SOLOMON généralisés.

Je ne démontrerai pas les propriétés suivantes qui découlent de la définition des codes. Le lecteur intéressé voudra bien se reporter à [PH98, p. 72].

Propriétés 1 Soit $\text{GRS}_k(\alpha, \mathbf{v})$, un code de REED–SOLOMON généralisé de longueur n , de dimension k et de distance minimale d .

1. $\text{GRS}_k(\alpha, \mathbf{v})$ est un code MDS, i.e.

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

2. Il existe un algorithme de décodage en temps polynomial de $\text{GRS}_k(\alpha, \mathbf{v})$ permettant de corriger

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor \text{ erreurs.}$$

Cet algorithme est une version plus générale de l'algorithme de décodage des codes BCH décrit dans [PH98, p. 70]. Il utilise la forme (1.5) de la matrice génératrice du code GRS.

3. le dual de $\text{GRS}_k(\alpha, \mathbf{v})$ est le code $\text{GRS}_{n-k}(\alpha, \mathbf{v}')$ pour un certain vecteur $\mathbf{v}'$ déterminé en fonction de $\mathbf{v}$.
4. La famille des codes GRS est stable par permutation, i.e. pour toute permutation π du vecteur générateur α , le code

$$\pi(\text{GRS}_k(\alpha, \mathbf{v})) = \text{GRS}_k(\pi(\alpha), \mathbf{v})$$

est un code GRS.

Les codes de REED–SOLOMON généralisés sont très structurés. En effet comme ce sont des codes MDS, on connaît leur distribution des poids – voir [MS77, Ch. 11]. De plus, la forme systématique de la matrice génératrice d'un code GRS, peut être obtenue à partir de la proposition suivante

Proposition 3 *Soit*

$$\mathbf{G} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ v_1\alpha_1 & v_2\alpha_2 & \cdots & v_n\alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ v_1\alpha_1^{k-1} & v_2\alpha_2^{k-1} & \cdots & v_n\alpha_n^{k-1} \end{pmatrix}, \quad (1.6)$$

une matrice génératrice d'un code de REED–SOLOMON généralisé sur $\mathbf{F}_q^m$. Alors, il existe une matrice $k \times k$ inversible $\mathbf{S}$ à coefficients dans $\mathbf{F}_q^m$ et une matrice $\mathbf{R} = (R_{i,j})_{i=1,j=k+1}^{k,n}$ telles que

$$\begin{cases} (\mathbf{I} \mid \mathbf{R}) = \mathbf{S}\mathbf{G}, \\ R_{i,j} = \frac{v_j}{v_i} \prod_{s=1, s \neq i}^k \frac{\alpha_j - \alpha_s}{\alpha_i - \alpha_s}, \end{cases}$$

où $\mathbf{I}$ désigne la matrice identité d'ordre k .

Preuve

Pour $i = 1, 2, \dots, k$, je définis le polynôme d'interpolation suivant

$$f_i(z) = \prod_{s=1, s \neq i}^k \frac{z - \alpha_s}{\alpha_i - \alpha_s} = \sum_{j=1}^{k-1} f_{i,j} z^{j-1},$$

qui est l'unique polynôme unitaire de degré $k - 1$ tel que

$$\begin{cases} f_i(\alpha_i) = 1, \\ f_i(\alpha_j) = 0, \quad \text{pour } j = 1, 2, \dots, k. \end{cases}$$

Notons $\mathbf{S} = \left(\frac{f_{i,j}}{v_i} \right)_{i=1,j=1}^{k,k}$. La i ème ligne de la matrice produit $\mathbf{S}\mathbf{G}$ est

$$\left(f_i(\alpha_1) \frac{v_1}{v_i}, f_i(\alpha_2) \frac{v_2}{v_i}, \dots, f_i(\alpha_n) \frac{v_n}{v_i} \right).$$

Par construction des polynômes f_i , les k premières colonnes de la matrice $\mathbf{S}\mathbf{G}$ forment la matrice identité. Donc $\mathbf{S}$ est inversible et

$$\mathbf{S}\mathbf{G} = (\mathbf{I} \mid \mathbf{R}),$$

où $\mathbf{R} = (R_{i,j})$, et

$$R_{i,j} = f_i(\alpha_j) \frac{v_j}{v_i}.$$

■

La proposition 3 donne explicitement les coefficients de la matrice génératrice sous forme systématique d'un code de REED–SOLOMON généralisé donné.

Corollaire 1 (Forme systématique) Soit $\mathbf{I}$ la matrice d'identité d'ordre k , et soit $\mathbf{R} = (R_{i,j})_{i=1,j=k+1}^{k,n}$ où

$$R_{i,j} = \frac{v_j}{v_i} \prod_{s=1, s \neq i}^k \frac{\alpha_j - \alpha_s}{\alpha_i - \alpha_s}.$$

Alors la matrice $(\mathbf{I} \mid \mathbf{R})$ est la matrice génératrice sous forme systématique du code $\text{GRS}_k(\alpha, \mathbf{v})$.

Preuve

D'après la définition 3, il existe une matrice génératrice $\mathbf{G}$ du code $\text{GRS}_k(\alpha, \mathbf{v})$ satisfaisant les hypothèses de la proposition 3.

Soit $\mathbf{S}$ et $\mathbf{R}$ les matrices telles que

$$(\mathbf{I} \mid \mathbf{R}) = \mathbf{S}\mathbf{G}.$$

D'après la proposition 3, $\mathbf{R} = (R_{i,j})$ vérifie

$$R_{i,j} = \frac{v_j}{v_i} \prod_{s=1, s \neq i}^k \frac{\alpha_j - \alpha_s}{\alpha_i - \alpha_s}.$$

■

2.1.2 Attaque de SIDEL'NIKOV–SHESTAKOV

L'attaque de SIDEL'NIKOV–SHESTAKOV est décrite dans l'article [SS92]. Elle s'appuie sur la structure particulière des *codes GRS* et sur la forme de leur matrice génératrice sous forme systématique. Cette attaque permet de retrouver, en temps polynomial, un décodeur du *code public* d'un système de McELIECE et de NIEDERREITER.

Proposition 4 (Noyau de l'attaque) Étant donnée une matrice génératrice $\mathbf{G}$, aléatoire de taille $k \times n$, d'un code de REED–SOLOMON généralisé sur $\mathbf{F}_{q^m}$, on peut retrouver une matrice $k \times k$ inversible $\mathbf{S}$ et une matrice

$$\mathbf{G}' = \begin{pmatrix} v_1 & \cdots & v_n \\ \vdots & \ddots & \vdots \\ v_1 \alpha_1^k & \cdots & v_n \alpha_n^k \end{pmatrix}$$

à coefficients dans $\mathbf{F}_{q^m}$ telles que

$$\mathbf{G} = \mathbf{S}\mathbf{G}'$$

en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.

Preuve

On commence par mettre la matrice $\mathbf{G}$ sous la forme échelonnée réduite décrite dans le corollaire 1. Cette opération coûte $O(k^3)$ multiplication dans $\mathbf{F}_{q^m}$ et fournit la matrice

$$(\mathbf{I} \mid \mathbf{R})$$

qui est la matrice génératrice d'un *code GRS* sous *forme systématique*. D'après la proposition 3, les coefficients $R_{i,j}$ de la matrice $\mathbf{R}$ vérifient

$$R_{i,j} = \frac{v_j}{v_i} \prod_{s=1, s \neq i}^k \frac{\alpha_j - \alpha_s}{\alpha_i - \alpha_s}, \quad (1.7)$$

Pour des éléments α_i et v_i de $\mathbf{F}_{q^m}$.

Comme le *groupe d'automorphismes* des codes GRS est trois fois transitif, on peut supposer sans perte de généralité que $\alpha_1, \alpha_2, \alpha_{k+1}$ et v_1 sont connus.

1. *On détermine v_2/v_1* : comme $\alpha_1, \alpha_2, \alpha_{k+1}$ et v_1 sont supposés connus, en calculant le rapport

$$\frac{R_{1,k+1}}{R_{2,k+1}} = c \frac{\alpha_{k+1} - \alpha_2}{\alpha_{k+1} - \alpha_1}$$

où $c = v_2/v_1$, on détermine c et donc v_2 en 3 multiplications dans $\mathbf{F}_{q^m}$.

2. *On détermine α_j pour $j = k + 2, \dots, n$* : on a

$$\frac{R_{1,j}}{R_{2,j}} = c \frac{\alpha_j - \alpha_2}{\alpha_j - \alpha_1}.$$

Comme c a été déterminé à l'étape précédente, l'unique inconnue de chacune des équations est la valeur de α_j , que l'on détermine, en résolvant l'équation

$$\alpha_j \left(\frac{R_{1,j}}{R_{2,j}} - c \right) = \alpha_1 \frac{R_{1,j}}{R_{2,j}} - c\alpha_2.$$

Les α_j étant deux à deux distincts, l'équation n'est jamais dégénérée. Cette opération coûte $4(n - k - 2)$ multiplications dans $\mathbf{F}_{q^m}$.

3. *On détermine α_j pour $j = 3, \dots, k$* : on a

$$\frac{R_{1,k+1}R_{j,k+2}}{R_{j,k+1}R_{1,k+2}} = \frac{(\alpha_{k+1} - \alpha_j)(\alpha_{k+2} - \alpha_2)}{(\alpha_{k+2} - \alpha_j)(\alpha_{k+1} - \alpha_1)}.$$

La seule inconnue de chacune des équations étant la valeur α_j , on détermine α_j en 6 multiplications dans $\mathbf{F}_{q^m}$.

Les trois étapes permettent le calcul des éléments α_j , pour $j = k + 1, \dots, n$, de manière unique connaissant $\alpha_1, \alpha_2, \alpha_{k+1}$ et v_1 . À partir de là, on détermine les v_i en utilisant la formule (1.7).

En effet, pour $i = k + 1, \dots, n$ on a

$$v_i = v_1 R_{1,i} \prod_{s=2}^k \frac{(\alpha_i - \alpha_s)}{(\alpha_1 - \alpha_s)},$$

et pour $i = 2, \dots, k$, on a

$$v_i = \frac{v_{k+1}}{R_{i,k+1}} \prod_{s=1, s \neq i}^k \frac{(\alpha_{k+1} - \alpha_s)}{(\alpha_i - \alpha_s)}$$

Ces étapes coûtent $O(k^2)$ multiplications dans $\mathbf{F}_{q^m}$.

La matrice $\mathbf{G}' = (v_j \alpha_j^i)$ candidate a donc été déterminée en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$, l'essentiel du coût étant la réduction de la matrice $\mathbf{G}$ donnée sous sa *forme systématique*. De plus, retrouver la matrice $\mathbf{S}$ vérifiant

$$\mathbf{G} = \mathbf{S}\mathbf{G}'$$

coûte une inversion de matrice $k \times k$ soit $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.

■

Corollaire 2 *Étant donné un système de chiffrement de McELIECE ou de NIEDERREITER utilisant comme code caché un code de REED–SOLOMON généralisé de longueur n et de dimension k sur $\mathbf{F}_{q^m}$, un attaquant retrouve un décodeur pour le code public en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.*

Preuve

On peut ramener l'attaque *structurelle* contre les deux systèmes au problème résolu dans la proposition précédente.

1. *Cryptosystème de McELIECE* : la *clé publique* est la matrice

$$\mathbf{G}_{\text{pub}} = \mathbf{S}\mathbf{G}\mathbf{P},$$

où $\mathbf{G}$ est une matrice génératrice d'un code GRS, où $\mathbf{P}$ est une matrice de permutation, et où $\mathbf{S}$ est une matrice inversible. Or, comme les codes GRS sont stables par permutation, la matrice $\mathbf{G}\mathbf{P}$ est une matrice génératrice d'un code GRS. $\mathbf{G}_{\text{pub}}$ est donc une matrice génératrice d'un code GRS.

2. *Cryptosystème de NIEDERREITER* : la *clé publique* est la matrice

$$\mathbf{H}_{\text{pub}} = \mathbf{S}\mathbf{H}\mathbf{P},$$

où $\mathbf{H}$ est une matrice de parité d'un code GRS, où $\mathbf{P}$ est une matrice de permutation, et où $\mathbf{S}$ est une matrice inversible. Or d'après la propriété 1, le dual d'un code GRS est un code GRS. Donc $\mathbf{H}$ est également matrice génératrice d'un code GRS. Par conséquent, $\mathbf{H}_{\text{pub}}$ est une matrice génératrice d'un code GRS.

Pour attaquer les deux cryptosystèmes on est ramené au problème résolu dans la proposition 3. Prenons par exemple le cas du système de McELIECE. D'après la proposition 3, à partir de $\mathbf{H}_{\text{pub}}$, un attaquant retrouve deux matrices $\mathbf{S}$ et $\mathbf{G}$ telles que

$$\mathbf{G}_{\text{pub}} = \mathbf{S}\mathbf{G},$$

en $O(k^3)$ multiplications, où $\mathbf{S}$ est une matrice inversible $k \times k$ et $\mathbf{G}$ est une matrice génératrice d'un code GRS permettant le décodage.

■

La structure générale de l'attaque de SIDEL'NIKOV-SHESTAKOV est décrite dans le tableau 1.6.

2.2 Codes de GOPPA classiques

Les codes de GOPPA furent introduits en 1970 par V. D. GOPPA comme généralisation des codes BCH – voir [Gop70]. Ils peuvent également être vus comme une sous-famille des *codes alternants*, qui constituent les *sous-codes trace* des codes de REED-SOLOMON généralisés. Ils disposent d'algorithmes polynomiaux de décodage et peuvent donc être utilisés comme *espace des clés* dans un cryptosystème de McELIECE ou de NIEDERREITER, tel que le suggéra McELIECE dans [McE78].

Dans un premier temps, je présente les définitions et les propriétés des codes de GOPPA. Dans un deuxième temps, je montre que les codes de GOPPA binaires ont des propriétés propres à la caractéristique 2. À paramètres égaux, ils peuvent être décodés plus loin que des codes de GOPPA sur des corps de caractéristiques impaires. Enfin, dans une troisième partie, j'ébauche les prémisses d'une attaque de type SIDEL'NIKOV-SHESTAKOV contre un cryptosystème utilisant des codes de GOPPA. Quoique ceux-ci dérivent des codes GRS, je montre qu'une telle attaque n'a pas de sens.

2.2.1 Définitions

Considérons un polynôme $g(x) \in \mathbf{F}_{q^m}[x]$, de degré t . Soit

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n)$$

Données :

- Une famille $\mathcal{G}$ de codes de REED–SOLOMON généralisés de longueur n , de dimension k constituant l'espace des clés du cryptosystème.
- La *clé publique* $\mathbf{G}_{\text{pub}}$

Résultat : les matrices

$$\begin{cases} \mathbf{G} = \left(v_j \alpha_j^i \right)_{i=0, j=1}^{k-1, n}, \\ \mathbf{S} \text{ matrice inversible de taille } k \times k. \end{cases}$$

telles que

$$\mathbf{G}_{\text{pub}} = \mathbf{S}\mathbf{G}.$$

Procédure :

- Mettre la matrice $\mathbf{G}_{\text{pub}}$ sous la forme échelonnée réduite

$$(\mathbf{I} \mid \mathbf{R}).$$

- Déterminer la matrice

$$\mathbf{G} = \left(v_j \alpha_j^i \right)_{i=0, j=1}^{k-1, n}$$

telle que $\alpha_1, \dots, \alpha_n$ et $v_1, \dots, v_n$ de $\mathbf{F}_{q^m}$ vérifient les équations

$$R_{i,j} = \frac{v_j}{v_i} \prod_{s=1, s \neq i}^k \frac{(\alpha_j - \alpha_s)}{(\alpha_i - \alpha_s)}.$$

- Déterminer la matrice $\mathbf{S}$ telle que

$$\mathbf{G}_{\text{pub}} = \mathbf{S}\mathbf{G}.$$

TABLEAU 1.6: Attaque de SIDEL'NIKOV–SHESTAKOV contre un cryptosystème de MCELIECE

un étiquetage des éléments de $\mathbf{F}_{q^m}$ qui ne sont pas racines de g . Soit alors la matrice

$$\mathbf{H} = \begin{pmatrix} 1/g(\alpha_1) & 1/g(\alpha_2) & \cdots & 1/g(\alpha_n) \\ \alpha_1/g(\alpha_1) & \alpha_2/g(\alpha_2) & \cdots & \alpha_n/g(\alpha_n) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1}/g(\alpha_1) & \alpha_2^{t-1}/g(\alpha_2) & \cdots & \alpha_n^{t-1}/g(\alpha_n) \end{pmatrix}. \quad (1.8)$$

On a la proposition suivante :

Proposition 5 ([MS77]) *Soit $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n})$ un mot de longueur n sur $\mathbf{F}_{q^m}$ de vecteur générateur $\mathbf{L}$, alors les deux assertions suivantes sont équivalentes :*

- i) $\sum_{i=1}^n \frac{a_{\alpha_i}}{(x-\alpha_i)} \equiv 0 \pmod{g(x)}$.*
- ii) $\mathbf{aH}^T = 0$.*

Preuve de la proposition

1. *i) $\implies$ ii) :* dans l'anneau des polynômes à coefficients dans $\mathbf{F}_{q^m}$ modulo $g(x)$, les polynômes $x - \alpha_i$ sont inversibles puisque les α_i ne sont pas racine de g . De plus

$$(x - \alpha_i) \frac{g(x) - g(\alpha_i)}{g(\alpha_i)(x - \alpha_i)} \equiv -1 \pmod{g(x)}.$$

On en déduit que, dans l'anneau des polynômes à coefficients dans $\mathbf{F}_{q^m}$ modulo $g(x)$, on a

$$\frac{1}{x - \alpha_i} = -\frac{g(x) - g(\alpha_i)}{g(\alpha_i)(x - \alpha_i)}. \quad (1.9)$$

Donc $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n})$ vérifie

$$\sum_{i=1}^n \frac{a_{\alpha_i}}{(x - \alpha_i)} \equiv 0 \pmod{g(x)}.$$

Puis, en remplaçant les $1/(x - \alpha_i) \pmod{g(x)}$ par leurs valeurs données dans (1.9), on obtient

$$\sum_{i=1}^n a_{\alpha_i} \frac{g(x) - g(\alpha_i)}{g(\alpha_i)(x - \alpha_i)} = 0. \quad (1.10)$$

Si le polynôme g s'écrit

$$g(x) = \sum_{i=0}^t g_i x^i,$$

où $g_t \neq 0$, alors

$$\frac{g(x) - g(\alpha_i)}{x - \alpha_i} = \sum_{j=1}^t g_j \sum_{k=0}^{i-1} x^k \alpha_i^{i-1-k}.$$

Par une permutation sous le signe Σ et un changement de variables *ad hoc*, on obtient

$$\frac{g(x) - g(\alpha_i)}{x - \alpha_i} = \sum_{k=0}^{t-1} \left(\sum_{j=0}^{t-k} g_{t-j} \alpha_i^{t-j-1} \right) x^k.$$

En annulant terme à terme les membres de l'équation (1.10) pour chacune des puissances x^k , où $k = 0, \dots, t-1$, le mot $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n})$ vérifie l'équation $\mathbf{a}\mathbf{H}^T = 0$ où

$$\mathbf{H} = \begin{pmatrix} g_t/g(\alpha_1) & \cdots & g_t/g(\alpha_n) \\ \vdots & \ddots & \vdots \\ \sum_{s=1}^t g_s \alpha_1^{s-1}/g(\alpha_1) & \cdots & \sum_{s=1}^t g_s \alpha_n^{s-1}/g(\alpha_n) \end{pmatrix}.$$

Bien que la forme de cette matrice paraisse un rien complexe, on remarque que

$$\mathbf{H} = \mathbf{S}\mathbf{G}'$$

où $\mathbf{S}$ est une matrice $k \times k$ triangulaire supérieure dont la valeur sur la diagonale est égale à g_t et où

$$\mathbf{G}' = \begin{pmatrix} 1/g(\alpha_1) & 1/g(\alpha_2) & \cdots & 1/g(\alpha_n) \\ \alpha_1/g(\alpha_1) & \alpha_2/g(\alpha_2) & \cdots & \alpha_n/g(\alpha_n) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1}/g(\alpha_1) & \alpha_2^{t-1}/g(\alpha_2) & \cdots & \alpha_n^{t-1}/g(\alpha_n) \end{pmatrix}.$$

Comme le coefficient g_t du terme de plus haut degré du polynôme $g(x)$ n'est pas nul, $\mathbf{S}$ est inversible et l'équation $\mathbf{a}\mathbf{H}^T = 0$ équivaut à

$$\mathbf{a}\mathbf{G}'^T = 0.$$

2. *ii) $\implies$ i)* : toutes les opérations accomplies dans l'étape précédente sont bijectives.

■

À présent, je donne la définition des codes de GOPPA. Ceux-ci sont en fait les mots de longueur n sur $\mathbf{F}_q \subset \mathbf{F}_{q^m}$ vérifiant une des assertions équivalentes de la proposition 5.

Définition 4 (Codes de GOPPA) Soit $g(x) \in \mathbf{F}_{q^m}[x]$ et soit

$$\mathbf{L} = (\alpha_1, \dots, \alpha_n)$$

un étiquetage des éléments de $\mathbf{F}_{q^m}$ qui ne sont pas racines de g . Alors l'ensemble des mots $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n}) \in \mathbf{F}_q^n$ de vecteur générateur $\mathbf{L}$ vérifiant l'une des assertions équivalentes de la proposition 5,

$$\begin{aligned} & - \sum_{i=1}^n \frac{a_{\alpha_i}}{(x-\alpha_i)} \equiv 0 \pmod{g(x)}, \\ & - \mathbf{aH}^T = 0, \end{aligned}$$

est appelé code de GOPPA de corps générateur $\mathbf{F}_{q^m}$, de polynôme générateur g et de vecteur générateur $\mathbf{L}$. Il est noté $\Gamma(\mathbf{L}, g)$.

Quand le polynôme générateur $g(x) \in \mathbf{F}_{q^m}[x]$ d'un code de GOPPA est irréductible dans $\mathbf{F}_{q^m}$, le vecteur générateur $\mathbf{L}$ du code est un étiquetage de $\mathbf{F}_{q^m}$ tout entier.

Définition 5 (Codes de GOPPA irréductibles) Si le polynôme générateur $g(x) \in \mathbf{F}_{q^m}[x]$ de $\Gamma(\mathbf{L}, g)$ est irréductible dans le corps générateur $\mathbf{F}_{q^m}$, alors $\Gamma(\mathbf{L}, g)$ est appelé code de GOPPA irréductible.

Les codes de GOPPA irréductibles constituent une sous-famille importante de la famille des codes de GOPPA. En effet, c'est la famille de codes utilisés originellement dans la conception du cryptosystème de McELIECE.

Chacune des assertions équivalentes de la proposition 5 donne un point de vue différent sur la structure des codes de GOPPA.

- *Généralisation des codes BCH* : c'est de cette façon que GOPPA les a construits dans son article de 1970 – voir [Gop70]. La construction des codes de GOPPA repose sur une extension de la construction des codes BCH – codes cycliques très étudiés en théorie des codes correcteurs. Le lecteur intéressé par l'étude des codes BCH pourra se reporter à [MS77, Ch. 8] ou bien à [PH98, p. 65].

Soit β une racine n ième de l'unité dans un corps $\mathbf{F}_{q^m}$. Considérons le code BCH sur $\mathbf{F}_q$ au sens strict, de distance construite δ . Ce code BCH est par définition l'ensemble des polynômes f à coefficients dans $\mathbf{F}_q$ et de degré $\leq q^m - 1$ tels que

$$f(\beta) = f(\beta^2) = \dots = f(\beta^{\delta-1}) = 0.$$

On peut démontrer que le code BCH ainsi construit

1. a pour dimension $k \geq n - m(d - 1)$,
2. a pour distance minimale $d \geq \delta$.

Comme β est une racine primitive n ième de $\mathbf{F}_{q^m}$, le vecteur

$$\mathbf{L} = (\beta^{-1}, \beta^{-2}, \dots, \beta^{-n} = 1)$$

est un étiquetage de $\mathbf{F}_{q^m}^*$.

On peut montrer – voir [Gop70] – que le code BCH sur $\mathbf{F}_q$ de distance construite δ est également l'ensemble des mots $\mathbf{a} = (a_{\beta^{-1}}, \dots, a_{\beta^{-n}}) \in \mathbf{F}_q^n$ tels que

$$\sum_{i=1}^n \frac{a_{\alpha_i}}{x - \beta^{-i}} \equiv 0 \pmod{x^{\delta-1}}.$$

Donc, d'après la définition 5, le code BCH sur $\mathbf{F}_q$ de distance construite δ est aussi un code de GOPPA de *polynôme générateur* $x^{\delta-1}$, et de *vecteur générateur* $\mathbf{L}$.

– Sous-codes trace *des codes de REED–SOLOMON généralisés* : la matrice

$$\mathbf{H} = \begin{pmatrix} 1/g(\alpha_1) & 1/g(\alpha_2) & \cdots & 1/g(\alpha_n) \\ \alpha_1/g(\alpha_1) & \alpha_2/g(\alpha_2) & \cdots & \alpha_n/g(\alpha_n) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1}/g(\alpha_1) & \alpha_2^{t-1}/g(\alpha_2) & \cdots & \alpha_n^{t-1}/g(\alpha_n) \end{pmatrix}$$

à coefficients dans $\mathbf{F}_{q^m}$ est, par construction, une matrice génératrice du code $\text{GRS}_t(\mathbf{L}, \mathbf{v})$ où

$$\begin{cases} \mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n), \\ \mathbf{v} = \left(\frac{1}{g(\alpha_1)}, \frac{1}{g(\alpha_2)}, \dots, \frac{1}{g(\alpha_n)} \right). \end{cases}$$

Comme les codes de REED–SOLOMON généralisés sont stables par dualité – voir les propriétés 1 –, $\mathbf{H}$ est également une matrice de parité d'un code de REED–SOLOMON généralisé. Or, par définition, $\Gamma(\mathbf{L}, g)$ est l'ensemble des mots

$$\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n}) \in \mathbf{F}_q^n$$

tels que $\mathbf{a}\mathbf{H}^T = 0$. $\mathbf{H}$ donne donc des équations de parité vérifiées par les mots de $\Gamma(\mathbf{L}, g)$. Par conséquent, $\Gamma(\mathbf{L}, g)$ est le *sous-code trace* sur $\mathbf{F}_q$ du code de REED–SOLOMON généralisé de matrice de parité $\mathbf{H}$.

2.2.2 Propriétés générales

On peut transmettre certaines propriétés de la famille des codes de REED–SOLOMON généralisés sur $\mathbf{F}_{q^m}$ aux codes de GOPPA par une simple projection sur $\mathbf{F}_q$. Ces propriétés sont donc vérifiées, non seulement par les codes de GOPPA, mais également par les *codes alternants* qui sont par définition les *sous-codes trace* des codes de REED–SOLOMON généralisés.

Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA dérivant du *code GRS* sur $\mathbf{F}_{q^m}$ de matrice de parité

$$\mathbf{H} = \begin{pmatrix} 1/g(\alpha_1) & 1/g(\alpha_2) & \cdots & 1/g(\alpha_n) \\ \alpha_1/g(\alpha_1) & \alpha_2/g(\alpha_2) & \cdots & \alpha_n/g(\alpha_n) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{t-1}/g(\alpha_1) & \alpha_2^{t-1}/g(\alpha_2) & \cdots & \alpha_n^{t-1}/g(\alpha_n) \end{pmatrix}.$$

À partir de $\mathbf{H}$, on déduit une matrice de parité de $\Gamma(\mathbf{L}, g)$ en développant chacune des lignes de $\mathbf{H}$ sur une base de $\mathbf{F}_{q^m}/\mathbf{F}_q$. La matrice ainsi obtenue est une matrice à coefficients dans $\mathbf{F}_q$ de taille $mt \times n$. Il est facile de vérifier que c'est effectivement une matrice de parité de $\Gamma(\mathbf{L}, g)$. Donc la dimension k de $\Gamma(\mathbf{L}, g)$ vérifie l'inégalité

$$k \geq n - mt.$$

Or les codes GRS sont également MDS. La distance minimale du code de matrice de parité $\mathbf{H}$ est donc égale à $t+1$. Comme $\Gamma(\mathbf{L}, g)$ est un sous-code du code de matrice de parité $\mathbf{H}$, sa *distance minimale* d est également minorée par $t+1$. Finalement, à partir de l'algorithme de décodage des codes BCH, on déduit l'existence d'un algorithme polynomial de décodage permettant de décoder $\Gamma(\mathbf{L}, g)$ jusqu'à la distance $\lfloor t/2 \rfloor$ – voir [Pat75]. En résumé,

Propriétés 2 Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA de longueur n , de corps générateur $\mathbf{F}_{q^m}$, et dont le polynôme générateur g est de degré t .

1. La dimension k de $\Gamma(\mathbf{L}, g)$ vérifie l'inégalité

$$k \geq n - mt.$$

2. La distance minimale d de $\Gamma(\mathbf{L}, g)$ vérifie

$$d \geq t + 1.$$

3. Il existe un algorithme polynomial permettant de décoder

$$\left\lfloor \frac{t}{2} \right\rfloor \text{ erreurs.}$$

Un résultat général sur la structure du dual des sous-codes trace prouvé par DELSARTE [Del75b] permet de déduire une formule explicite du dual d'un code de GOPPA :

Proposition 6 Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA de corps générateur $\mathbf{F}_{q^m}$, où g est de degré t , et où

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n).$$

Alors le dual de $\Gamma(\mathbf{L}, g)$ est l'ensemble

$$\Gamma(\mathbf{L}, g)^\perp = \left\{ \left(\text{Tr} \left[\frac{f(\alpha_1)}{g(\alpha_1)} \right], \dots, \text{Tr} \left[\frac{f(\alpha_n)}{g(\alpha_n)} \right] \right) \mid \begin{array}{l} f \in \mathbf{F}_{q^m}[x], \\ \deg(f) \leq t - 1. \end{array} \right\}$$

où Tr désigne l'opérateur trace de $\mathbf{F}_{q^m}$ sur $\mathbf{F}_q$, i.e.

$$\forall z \in \mathbf{F}_{q^m}, \quad \text{Tr}(z) = \sum_{i=1}^m z^{q^i}.$$

2.2.3 Propriétés des codes de GOPPA binaires

J'ai choisi de faire une section à part pour parler des codes de GOPPA binaires. En effet, ceux-ci font d'une part l'objet d'une littérature plus abondante que les codes de GOPPA sur des corps de caractéristique impaire, et d'autre part ils ont des propriétés spécifiques qui les distinguent des *codes alternants* – voir par exemple [Ber73].

Ici, je montre que les codes de GOPPA binaires peuvent être caractérisés par des propriétés simples d'algèbres de polynômes sur un corps de caractéristique 2. Cette caractérisation permet d'améliorer la borne inférieure sur la *distance minimale*. Elle montre en outre que l'algorithme polynomial de décodage corrige plus d'erreurs dans ce cas.

Proposition 7 Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA binaire de corps générateur $\mathbf{F}_{2^m}$ et de vecteur générateur

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \mathbf{F}_{2^m}^n.$$

Alors $\Gamma(\mathbf{L}, g)$ est l'ensemble de tous les mots binaires $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n})$ dont la dérivée du polynôme localisateur

$$f_{\mathbf{a}}(x) = \prod_{i=1, a_{\alpha_i} \neq 0}^n (x - \alpha_i)$$

est multiple du polynôme générateur g , i.e.

$$\mathbf{a} \in \Gamma(\mathbf{L}, g) \iff f'_{\mathbf{a}}(x) \equiv 0 \pmod{g(x)}.$$

Preuve

Soit $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n}) \in \Gamma(\mathbf{L}, g)$. Par définition, $\mathbf{a}$ vérifie

$$\sum_{i=1}^n \frac{a_{\alpha_i}}{x - \alpha_i} \equiv 0 \pmod{g(x)}.$$

Comme les a_{α_i} sont des éléments de $\mathbf{F}_2$, ils sont égaux soit à 0 soit à 1. Donc

$$\sum_{i=1}^n \frac{a_{\alpha_i}}{x - \alpha_i} = \sum_{i=1, a_{\alpha_i} \neq 0}^n \frac{1}{x - \alpha_i} = \frac{f'_{\mathbf{a}}(x)}{f_{\mathbf{a}}(x)}.$$

On en déduit

$$\mathbf{a} \in \Gamma(\mathbf{L}, g) \iff \frac{f'_{\mathbf{a}}(x)}{f_{\mathbf{a}}(x)} \equiv 0 \pmod{g(x)}.$$

De plus, les racines du *polynôme localisateur* $f_{\mathbf{a}}$ sont, par construction, des éléments α_i du *vecteur générateur* $\mathbf{L}$. Le *vecteur générateur* étant un étiquetage des éléments de $\mathbf{F}_{2^m}$ qui ne sont pas racines de g , les polynômes $f_{\mathbf{a}}$ et g sont premiers entre eux. Donc

$$\mathbf{a} \in \Gamma(\mathbf{L}, g) \iff f'_{\mathbf{a}}(x) \equiv 0 \pmod{g(x)}.$$

■

Outre les caractérisations données par la définition 4, la proposition 7 fournit donc une autre caractérisation des codes de GOPPA binaires. À savoir, un code de GOPPA binaire est un ensemble de polynômes à coefficients dans $\mathbf{F}_{2^m}$, scindés et sans racines multiples, et dont les dérivées ont comme *pgcd* commun le *polynôme générateur* g .

Cette nouvelle caractérisation permet entre autre d'améliorer la borne inférieure sur la distance minimale.

Proposition 8 *Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA binaire, où $g(x) \in \mathbf{F}_{2^m}[x]$ tel que*

$$g = f^2h,$$

où $f(x) \in \mathbf{F}_{2^m}[x]$ et $h(x) \in \mathbf{F}_{2^m}[x]$ sont sans facteurs carrés et de degrés respectifs t_1 et t_2 . Alors :

1. *La distance minimale d de $\Gamma(\mathbf{L}, g)$ vérifie*

$$d \geq 2(t_1 + t_2) + 1.$$

2. *Il existe un algorithme polynomial de décodage de $\Gamma(\mathbf{L}, g)$ jusqu'à la capacité de correction construite $t_1 + t_2$.*

Preuve

- La preuve de la première assertion de la proposition repose sur le fait que, en caractéristique 2, la dérivée de tout polynôme est un carré. Soit $g = f^2h$ le *polynôme générateur* d'un code de GOPPA binaire $\Gamma(\mathbf{L}, g)$. Soit $\mathbf{a} \in \Gamma(\mathbf{L}, g)$ un mot dont le *polynôme localisateur* est $f_{\mathbf{a}}$. D'après la proposition 7, on a

$$g = f^2h \mid f'_{\mathbf{a}}.$$

Comme $f'_{\mathbf{a}}$ est un carré on a également

$$(fh)^2 \mid f'_{\mathbf{a}}$$

Partant le degré de la dérivée du polynôme localisateur de $\mathbf{a}$ est au moins égal à $2(t_1 + t_2)$ où t_1 est le degré de f et t_2 est le degré de h . Donc le degré de $f_{\mathbf{a}}$ est supérieur à $2(t_1 + t_2) + 1$. Donc $\mathbf{a}$ est de poids supérieur ou égal à $2(t_1 + t_2) + 1$.

- La description de l'algorithme polynomial permettant de décoder $t_1 + t_2$ erreurs sort du cadre de cette thèse. Le lecteur intéressé pourra se reporter aux articles [CL87, Sen91]. Et le lecteur vraiment très courageux pourra tenter de décortiquer le papier de PATTERSON [Pat75].

■

En caractéristique impaire, l'algorithme de décodage en temps polynomial décode

$$\left\lfloor \frac{t_1 + t_2}{2} \right\rfloor = \left\lfloor \frac{t}{2} \right\rfloor \text{ erreurs.}$$

Donc, si le polynôme générateur d'un code de GOPPA binaire est de degré t , sans facteurs carrés, on peut corriger t erreurs, soit deux fois plus qu'en caractéristique impaire sous les mêmes conditions.

En caractéristique 2, quand le degré du polynôme générateur du code de GOPPA considéré est suffisamment petit, et que ce dernier est sans facteur carré, alors la dimension du code atteint sa borne inférieure, à savoir,

$$k = n - mt,$$

où k est la dimension du code, n sa longueur, m le degré du *corps générateur* $\mathbf{F}_{2^m}$, et où t est le degré du *polynôme générateur*.

Théorème 2 ([vdV90]) *Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA binaire de corps générateur $\mathbf{F}_{2^m}$, où g est sans racines dans $\mathbf{F}_{2^m}$, et tel que*

$$g = f^2 h,$$

où f et h sont sans facteurs carrés, de degrés respectifs t_1 et t_2 . Si le degré $t = 2t_1 + t_2$ de g vérifie

$$t < 2^{m/2-1} + 1,$$

alors la dimension k de $\Gamma(\mathbf{L}, g)$ est égale à

$$k = n - m(t_1 + t_2).$$

Corollaire 3 *Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA binaire de corps générateur $\mathbf{F}_{2^m}$, et dont le polynôme générateur g de degré t est sans racines dans $\mathbf{F}_{2^m}$ et sans facteurs carrés. Si*

$$t < 2^{m/2-1} + 1,$$

alors la dimension k de $\Gamma(\mathbf{L}, g)$ est égale à

$$k = n - mt.$$

Ce corollaire s'applique en particulier aux codes de GOPPA binaires irréductibles.

Dans la pratique, le théorème 2 reste vrai, même quand le degré du *polynôme générateur* est supérieur à $2^{m/2-1} + 1$.

2.2.4 Une approche de la cryptanalyse structurelle des systèmes basés sur les codes de GOPPA

Comme les codes de GOPPA disposent d'algorithmes polynômiaux de décodage jusqu'à une distance bornée, on peut les utiliser comme *espace des clés* dans la conception d'un cryptosystème de McELIECE ou de NIEDERREITER. À taille de clé constante, les codes de GOPPA offrant la meilleure résistance contre les *attaques par décodage* sont les codes de GOPPA binaires dont le polynôme générateur est sans facteurs carrés. En effet, comme ceux-ci sont décodables plus loin que les codes de GOPPA sur des corps de caractéristique impaire – voir la proposition 8 –, le *vecteur d'erreur* que l'on peut ajouter est de poids plus grand. Par conséquent, d'après le théorème 1, les *attaques par décodage* contre les cryptosystèmes sont plus coûteuses. C'est sans doute la raison pour laquelle McELIECE suggéra l'utilisation des codes de GOPPA binaires comme *espace des clés* de son cryptosystème.

En fait, il proposa l'utilisation des codes de GOPPA irréductibles, de corps générateur $\mathbf{F}_{2^{10}}$ et de polynôme générateur de degré $t = 50$. Comme il y a autant de codes de GOPPA irréductibles de *corps générateur* $\mathbf{F}_{2^{10}}$ et de *polynôme générateur* de degré t , que de polynômes irréductibles unitaires de degré t à coefficients dans $\mathbf{F}_{2^{10}}$, la taille de l'espace des clés est égale à

$$2^{498,55} \text{ éléments.}$$

Bien que les conditions du théorème 2 ne soient pas satisfaites, en effet

$$50 > 2^{m/2-1} + 1 = 17,$$

en pratique néanmoins, la dimension des codes satisfait l'égalité

$$k = n - mt = 1024 - 50 \times 10 = 524.$$

Donc, la taille de la *clé publique* d'un système de chiffrement de McELIECE utilisant cette famille de codes vaut

$$1024 \times 524 \text{ bits.}$$

Comme g est irréductible, donc par voie de conséquence sans facteurs carrés, l'algorithme polynomial de décodage permet de corriger $t = 50$ erreurs. Donc, en utilisant l'algorithme de décodage par *ensembles d'information* décrit dans [CC98], le facteur de travail de l'algorithme consistant à retrouver le texte *clair* à partir du texte *chiffré* par un tel système est égal à

$$2^{64} \text{ opérations binaires.}$$

Les codes de GOPPA sont très liés aux codes GRS. Dans la section 2.1.2 nous avons vu que les *codes GRS* n'offraient aucune sécurité contre les *attaques structurelles*. En effet, un algorithme permettant de décoder le *code public* peut-être reconstruit en temps polynomial à partir de la *clé publique*.

Malgré tout, les codes de GOPPA sont suffisamment peu structurés pour qu'une telle attaque soit hautement improbable.

Considérons un cryptosystème de NIEDERREITER – par commodité – dont le *code caché* est un code de GOPPA $\Gamma(\mathbf{L}, g)$ de *corps générateur* $\mathbf{F}_{q^m}$. Grâce à la proposition 6 nous savons qu'une matrice de parité de $\Gamma(\mathbf{L}, g)$ est de la forme

$$\mathbf{H} = \begin{pmatrix} \text{Tr}\left(\frac{f_1(\alpha_1)}{g(\alpha_1)}\right) & \cdots & \text{Tr}\left(\frac{f_1(\alpha_n)}{g(\alpha_n)}\right) \\ \vdots & \ddots & \vdots \\ \text{Tr}\left(\frac{f_{n-k}(\alpha_1)}{g(\alpha_1)}\right) & \cdots & \text{Tr}\left(\frac{f_{n-k}(\alpha_n)}{g(\alpha_n)}\right) \end{pmatrix}, \quad (1.11)$$

pour des polynômes $f_1, f_2, \dots, f_{n-k}$ à coefficients dans $\mathbf{F}_{q^m}$, de degrés inférieurs à $t-1$, Tr désignant l'*opérateur Trace* de $\mathbf{F}_{q^m}/\mathbf{F}_q$. Par construction, la *clé publique* du cryptosystème de NIEDERREITER est égale à

$$\mathbf{H}_{\text{pub}} = \mathbf{SHP},$$

où $\mathbf{S}$ est une matrice inversible et où $\mathbf{P}$ est une matrice de permutation. Si une attaque de type SIDEL'NIKOV–SHESTAKOV était possible, on retrouverait en particulier la matrice $\mathbf{H}$. Le problème étant que, pour reconstruire un décodeur à partir de $\mathbf{H}$, il faut connaître le polynôme g . Soit $\mathbf{H} = (h_{ij})_{i=1, j=1}^{n-k, n}$, on est amené à résoudre le système d'équations suivants aux inconnues $f_i(x) \in \mathbf{F}_{q^m}[x]$ de degré inférieur ou égal à $t-1$, et $g(x) \in \mathbf{F}_{q^m}[x]$ de degré t :

$$\text{Tr}\left(\frac{f_i(\alpha_j)}{g(\alpha_j)}\right) = h_{i,j},$$

pour $i = 1, \dots, n-k$ et $j = 1, \dots, n$. Or, chaque élément de $\mathbf{F}_q$ a exactement q^m/q antécédents dans $\mathbf{F}_{q^m}$ par l'opérateur trace. Donc, le simple système d'équations

$$\text{Tr}(\beta_{i,j}) = h_{i,j}$$

pour $i = 1, \dots, n-k$, et $j = 1, \dots, n$, aux inconnues $\beta_{i,j}$ a déjà

$$\left(\frac{q^m}{q}\right)^{(n-k)n} \text{ solutions.}$$

Même en supposant que les polynômes f_i sont publics, je ne connais aucun moyen de discriminer ces familles de solutions. Ceci est dû au fait que l'*opérateur trace* déstructure la multiplication dans $\mathbf{F}_{q^m}$. En effet, il n'existe aucun moyen permettant d'exprimer simplement la trace du produit de deux éléments

$$\text{Tr}(\alpha\beta), \quad \alpha, \beta \in \mathbf{F}_{q^m}$$

en fonction de la trace des éléments $\text{Tr}(\alpha)$ et $\text{Tr}(\beta)$.

L'attaque de SIDEL'NIKOV–SHESTAKOV est donc impraticable contre la famille des codes de GOPPA, en dépit de la structure qu'ils héritent des codes

de REED–SOLOMON généralisés. Cette remarque peut se généraliser à toute famille de *sous-codes trace* quels qu'ils soient. Les *sous-codes trace* semblent donc constituer des familles de codes résistantes aux *attaques structurelles*. On peut même renchérir en disant qu'il n'existe pas d'invariant connus des codes de GOPPA – tels que la distance minimale, ou bien la distribution des poids – qui permette de distinguer un code de GOPPA tiré aléatoirement d'un *code aléatoire*, mais ceci est une autre histoire et sera développé plus en profondeur dans le prochain chapitre.

3 Conclusion

Dans ce chapitre, j'ai présenté les cryptosystèmes de McELIECE et de NIEDERREITER, dont la sécurité repose sur le problème difficile du *décodage borné* d'un code pour la métrique de HAMMING. Nous avons vu qu'il fallait choisir soigneusement la famille de codes jouant le rôle de *l'espace des clés*. Hormis les codes GRS, Nicolas SENDRIER a montré qu'il fallait également éviter d'utiliser les *codes concaténés* [Sen98]. Ces études conduisent à la prudence en ce qui concerne les cryptosystèmes utilisant les codes de GOPPA. Bien que ces derniers résistent depuis plus de 20 ans à toute tentative de *cryptanalyse structurelle*, des travaux récents montrèrent que, lorsque un code de GOPPA était très structuré, son *polynôme générateur* était également très structuré. Pour l'instant, un certain nombre de résultats ont été obtenus dans le cas binaire – voir [Can96, Loi97, Vér98, LS01, Loi]. Par contre, il y a très peu de résultats concernant les codes de GOPPA sur des corps de caractéristique impaire. Ceci m'a notamment conduit à étudier des propriétés de primalité et de primitivité des trinômes sur des corps de caractéristiques impaires. Ce travail est présenté en dans l'annexe A.

D'autres systèmes de chiffrement utilisant la théorie des codes correcteurs et basés sur le problème du décodage borné furent proposés ultérieurement. SIDEL'NIKOV en a notamment proposé un qui utilise comme clé publique, un code de REED–MULLER permuté, et un algorithme de décodage probabiliste – [SP92, Sid94]. D'autre part un cryptosystème reposant sur des problèmes de décodages dans des métriques différentes de la métriques de HAMMING, fut présenté par GABIDULIN, PARAMONOV et TRETJAKOV et fera l'objet du chapitre 4.

Chapitre 2

Clés faibles du cryptosystème de McELIECE

Dans le chapitre précédent, j'ai présenté les systèmes de chiffrement de McELIECE et de NIEDERREITER dont la sécurité repose sur la difficulté de décoder un *code aléatoire* jusqu'à une distance de HAMMING fixée. Cependant, quand on utilise certaines familles de codes comme espace des clés, il existe des biais structurels par lesquels un attaquant peut retrouver un décodeur du *code public*. J'ai présenté l'attaque de SIDEL'NIKOV-SHESTAKOV contre les codes de REED-SOLOMON généralisés, et Nicolas SENDRIER a montré que les codes concaténés étaient également à proscrire [Sen98]. En revanche, pour la famille des codes de GOPPA, on ne connaît pas de tel biais. Partant, le seul moyen de reconstruire un décodeur du *code public* consiste à énumérer les codes qui constituent l'*espace des clés* jusqu'à en trouver un qui soit équivalent au *code public*. C'est le principe d'une *attaque structurelle*.

Dans ce chapitre, je présente d'abord deux types d'*attaques structurelles* contre le système de chiffrement de McELIECE. D'une part une attaque consistant à énumérer l'*espace des clés* en testant l'équivalence de chaque élément avec le *code public*, et d'autre part une attaque consistant à énumérer l'*espace des clés étendu* en testant l'équivalence de chaque élément avec le *code public étendu*. Les outils développés dans la première partie me servent dans un second temps pour concevoir une *attaque structurelle* réaliste sur un ensemble de *clés faibles* du cryptosystème. Ce travail est issu d'une collaboration avec N. SENDRIER, dans laquelle nous montrons que les codes de GOPPA dont le *polynôme générateur* est binaire constituent une famille de *clés faibles* du cryptosystème de McELIECE. Alors, on réduit considérablement le nombre de tests à effectuer, mais on réduit également le coût de chaque test grâce aux propriétés d'une famille de codes dérivés appelés *sous-codes idempotents projetés*.

1 Principe général des attaques structurelles

Le but d'une *attaque structurelle* contre le cryptosystème de McELIECE ou de NIEDERREITER consiste à retrouver un décodeur polynomial du *code public*. Dans la section 2.1 du chapitre 1, nous avons vu que, en exploitant des propriétés de structure des codes de REED–SOLOMON généralisés, retrouver un décodeur prenait $O(k^3)$ multiplications, où k est la dimension des codes. Idéalement cependant, l'*espace des clés* ne doit pas posséder de tel point faible. On doit donc procéder différemment.

Par construction, le *code public* du cryptosystème de McELIECE ou de NIEDERREITER est équivalent au *code caché*. Donc, tout code de la *classe d'équivalence* du *code caché* est équivalent au *code public*. Le principe général de l'*attaque structurelle* consiste alors à

1. énumérer les éléments de l'*espace des clés* en testant leur équivalence avec le *code public*,
2. retrouver la permutation entre les deux codes équivalents.

Comme les cryptosystèmes de McELIECE et de NIEDERREITER sont de sécurité équivalente – voir la section 1.1.3 du chapitre 1 –, je ne considérerai dans la suite que les attaques contre le système de chiffrement de McELIECE. Dans cette section, j'introduis d'abord la terminologie nécessaire concernant l'*équivalence de codes* et les *groupes d'automorphismes* de codes. Dans un second temps, je décris comment concevoir deux types d'*attaques structurelles* contre le cryptosystème. Enfin, je présente un outil efficace qui permet, d'une part de tester l'équivalence entre deux codes en temps polynomial, et d'autre part de déterminer la permutation entre deux codes équivalents : l'*algorithme de séparation des supports*, noté *SSA*. Sa description complète se trouve dans l'article de Nicolas SENDRIER [Sen00].

Dans la suite, et sauf mention contraire, les mots $\mathbf{a}$ de longueur n sont étiquetés par le vecteur

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n),$$

i.e. tout mot $\mathbf{a}$ de longueur n s'écrit

$$\mathbf{a} = (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n}).$$

Le vecteur $\mathbf{L}$ est appelé *vecteur générateur* du mot $\mathbf{a}$, conformément aux notations introduites par GIBSON dans [Gib95a, Gib91].

1.1 Classes d'équivalence de codes et groupes d'automorphismes

Dans cette section j'introduis en premier lieu les notions d'équivalence de codes, de groupe d'automorphismes, de *code étendu*, de *code poinçonné*, et dans un second temps, je caractérise les classes d'équivalence de codes de GOPPA et de codes de GOPPA étendus.

1.1.1 Généralités sur les codes

Il existe plusieurs notions d'équivalences entre codes. Je ne mentionne ici que celle qui me sera utile afin de construire les *attaques structurelles* : l'équivalence par permutation du *vecteur générateur*.

Définition 6 (Équivalence de codes) Deux codes C et C' de vecteur générateur

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n)$$

sont équivalents s'il existe une permutation π de $\mathbf{L}$ telle que

$$\forall \mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n}) \in C, \quad \pi(\mathbf{a}) = (b_{\alpha_1}, \dots, b_{\alpha_n}) \in C',$$

où $b_{\alpha_i} = a_{\pi^{-1}(\alpha_i)}$. On note alors $C' = \pi(C)$.

L'équivalence de codes définie dans la définition 6 est une relation d'équivalence au sens usuel du terme. Partant, l'ensemble des *classes d'équivalence* modulo cette relation constitue une partition de l'ensemble des codes de *vecteur générateur* $\mathbf{L}$. Parallèlement, j'introduis le concept de *groupe d'automorphismes* de codes dont le rôle s'avère crucial dans le processus de reconnaissance des *clés faibles* et de réduction de la complexité des tests d'équivalence.

Définition 7 (Groupe d'automorphismes d'un code) Soit C un code de vecteur générateur $\mathbf{L}$. Le groupe d'automorphismes de C – noté $\text{Aut}(C)$ – est l'ensemble des permutations de $\mathbf{L}$ qui laissent le code C invariant, i.e.

$$\pi \in \text{Aut}(C) \iff \pi(C) = C.$$

Étant donné un code linéaire C de longueur n , étendre C consiste à le plonger dans un code de longueur $n+1$ en prenant la position à l'infini, voir [MS77, p. 346].

Définition 8 (Code étendu) Soit C un code de dimension k et de vecteur générateur

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n),$$

alors le code étendu, noté $\widehat{C}$, de C est le code de longueur $n+1$, de dimension k et de vecteur générateur

$$\widehat{\mathbf{L}} = (\alpha_1, \alpha_2, \dots, \alpha_n, \infty),$$

défini par

$$\widehat{C} = \left\{ (a_{\alpha_1}, \dots, a_{\alpha_n}, a_{\infty}) \mid \begin{array}{l} (a_{\alpha_1}, \dots, a_{\alpha_n}) \in C \\ a_{\infty} = -\sum_{\alpha \in \mathbf{L}} a_{\alpha} \end{array} \right\}.$$

Comme le code C peut être naturellement plongé dans son *code étendu* $\widehat{C}$ en ajoutant la coordonnée à l'infini, on déduit une relation entre le *groupe d'automorphismes* de C et le *groupe d'automorphismes* de $\widehat{C}$.

Proposition 9 *Soit C un code et soit $\widehat{C}$ le code étendu correspondant, alors on a*

$$\text{Aut}(C) \subset \text{Aut}(\widehat{C}),$$

Remarque 1 *La proposition 9 n'est pas tout à fait exacte. Par définition, les éléments du groupe d'automorphismes de C agissent sur le vecteur générateur $\mathbf{L}$ de C tandis que les éléments du groupe d'automorphismes de $\widehat{C}$ agissent sur le vecteur générateur étendu*

$$\widehat{\mathbf{L}} = \mathbf{L} \cup (\infty).$$

Si on considère que les permutations de $\mathbf{L}$ sont exactement les permutations de $\widehat{\mathbf{L}}$ qui laissent invariant l'élément ∞ , alors la proposition est vraie. Néanmoins, afin de ne pas surcharger le texte de notations, je tiens le plongement pour implicite.

Si deux codes sont équivalents, alors leurs étendus sont également équivalents.

Proposition 10 (Équivalence de codes étendus) *Soient C et C' deux codes équivalents. Alors, les codes étendus correspondants $\widehat{C}$ et $\widehat{C}'$ sont équivalents.*

Preuve

Soient C et C' deux codes de vecteur générateur

$$\mathbf{L} = (\alpha_1, \dots, \alpha_n).$$

Soit π la permutation de $\mathbf{L}$ telle que

$$C' = \pi(C).$$

Soit alors $\widehat{\pi}$, la permutation de $\widehat{\mathbf{L}} = \mathbf{L} \cup (\infty)$, telle que

$$\begin{cases} \widehat{\pi}(\alpha_i) = \pi(\alpha_i), & \forall \alpha_i \in \mathbf{L}, \\ \widehat{\pi}(\infty) = \infty. \end{cases}$$

Alors on a $\widehat{C}' = \widehat{\pi}(\widehat{C})$.

■

Définition 9 (Code poinçonné) *Soit C un code de longueur n , de vecteur générateur $\mathbf{L}$. Soit $\alpha \in \mathbf{L}$. Alors le code poinçonné de C en α est le code, noté C_α , de longueur $n - 1$ constitué des mots de C auxquels on a supprimé la position étiquetée par α .*

J'introduis également la notion de *hull* d'un code

Définition 10 (Hull d'un code) *Étant donné un code linéaire C , le hull de C est le code linéaire $\mathcal{H}(C)$ défini par*

$$\mathcal{H}(C) = C \cap C^\perp,$$

où $C^\perp$ désigne le dual de C .

Si on se place sur un alphabet de caractéristique 0, le *hull* d'un code est toujours réduit au mot nul

$$\mathbf{0} = \underbrace{(0, 0, \dots, 0)}_{|\mathbf{L}| \text{ fois}}$$

Cependant, si on se place en caractéristique non nulle il peut être non trivial – la répartition des dimension des *hulls* de codes binaires aléatoires se trouve dans l'article de SENDRIER [Sen97].

1.1.2 Classes d'équivalence des codes de GOPPA

Un résultat important pour les prochaines sections, concerne les transformations du *polynôme générateur* qui changent un code de GOPPA en code de GOPPA équivalent, *i.e.* étant donné un code de GOPPA $\Gamma(\mathbf{L}, g)$ où $g(x) \in \mathbf{F}_{q^m}[x]$, quels sont les polynômes $h(x) \in \mathbf{F}_{q^m}[x]$ tels que $\Gamma(\mathbf{L}, g)$ et $\Gamma(\mathbf{L}, h)$ sont équivalents ?

GIBSON apporta des éléments de réponse dans l'article [Gib91] pour les codes de GOPPA binaires. Il montre que si le *polynôme générateur* g , de degré t , a pour racines les éléments

$$\gamma_1, \dots, \gamma_t,$$

dans une extension de $\mathbf{F}_2$, alors le code de GOPPA dont le polynôme générateur a pour racines

$$a\gamma_1^{2^i} + b, \dots, a\gamma_t^{2^i} + b$$

où $a, b \in \mathbf{F}_{2^m}$, non nuls tous les deux, est équivalent à $\Gamma(\mathbf{L}, g)$. Comme le nombre de telles transformations distinctes est égal à

$$m(2^{2^m} - 1),$$

la taille des classes d'équivalence de codes de GOPPA binaire est au moins égale à $m2^{2^m}$.

On peut généraliser cette propriété aux codes de GOPPA de *corps générateur* $\mathbf{F}_{q^m}$. Donc, pour un code de GOPPA donné de *corps générateur* $\mathbf{F}_{q^m}$, il y a au moins

$$m(q^{2^m} - 1)$$

codes de GOPPA qui lui sont équivalents.

Plus récemment, RYAN et FITZPATRICK ont calculé pour certains paramètres le nombre de codes de GOPPA irréductibles qui ne sont pas équivalents, voir [RF01]. De leur côté, ELIA, TARICCO et VITERBO ont dénombré les codes de GOPPA à équivalence près, voir [ETV00]. Il s'avère que les seules transformations qui laissent invariante par permutation la famille des codes de GOPPA sont celles données par GIBSON. La borne inférieure mq^{2m} sur la taille des classes d'équivalence de codes de GOPPA ne peut donc pas être améliorée.

Pour la suite, j'ai besoin du nombre de transformations du *polynôme générateur* d'un code de GOPPA qui changent un *code de GOPPA étendu* en un *code de GOPPA étendu* équivalent. D'après la proposition 10, le nombre de telles transformations est au moins égal au nombre de transformations qui changent un code de GOPPA en un code de GOPPA équivalent.

Définition 11 (Code de GOPPA étendu) *Étant donné $\Gamma(\mathbf{L}, g)$, un code de GOPPA, de corps générateur $\mathbf{F}_{q^m}$, de vecteur générateur*

$$\mathbf{L} = (\alpha_1, \dots, \alpha_n),$$

on appelle code de GOPPA étendu, de corps générateur $\mathbf{F}_{q^m}$, de vecteur générateur $\mathbf{L}$, et de polynôme générateur g le code, noté $\widehat{\Gamma}(\mathbf{L}, g)$, défini par

$$\widehat{\Gamma}(\mathbf{L}, g) = \left\{ (\mathbf{a}, a_\infty) \mid \begin{array}{l} \mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n}) \in \Gamma(\mathbf{L}, g), \\ a_\infty = -\sum_{\alpha \in \mathbf{L}} a_\alpha \end{array} \right\}.$$

Avec cette définition, on a la proposition suivante :

Proposition 11 *Soit l'homographie*

$$\pi(x) = \frac{ax + b}{cx + d},$$

où $a, b, c, d \in \mathbf{F}_{q^m}$, et $ad - bc \neq 0$. Soit g un polynôme à coefficients dans $\mathbf{F}_{q^m}$, et soit

$$g_1 = (cx + d)^t g(\pi(x)).$$

Alors les codes de GOPPA étendus $\widehat{\Gamma}(\mathbf{L}, g)$ et $\widehat{\Gamma}(\mathbf{L}, g_1)$ sont équivalents.

La preuve de cette proposition se trouve dans [MS77, p. 347]. La proposition 11 implique en particulier que le nombre de codes de GOPPA étendus équivalents à un code de GOPPA étendu donné de *corps générateur $\mathbf{F}_{q^m}$* est au moins égal à

$$mq^m(q^{2m} - 1).$$

Cette quantité correspond au nombre d'homographies de $\mathbf{F}_{q^m}$.

1.2 Attaques structurelles – généralités

Dans cette section je présente l'attaque générale permettant de recouvrer un décodeur de la *clé publique* du cryptosystème de McELIECE en utilisant la propriété de l'équivalence du *code public* avec le *code caché*. Les attaques structurelles consistent à énumérer les codes de l'*espace des clés* en testant

- ou bien leur équivalence avec le *code public*,
- ou bien l'équivalence de leurs *codes étendus* avec le *code public étendu*.

Considérons le cryptosystème de McELIECE dont l'*espace des clés* est une famille $\mathcal{G}$ de codes de *vecteur générateur*

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n),$$

pour lesquels on dispose d'un algorithme polynomial permettant de décoder t erreurs.

- *Clé privée* :
 - Un code aléatoire $\Gamma \in \mathcal{G}$. Γ est appelé *code caché*.
 - Une permutation π de $\mathbf{L}$.
- *Clé publique* : une matrice génératrice prise aléatoirement du *code public* $C_{\text{pub}} = \pi(\Gamma)$.

Par construction, le *code public* est équivalent au *code caché* au sens de la définition 6. D'après la proposition 10, le *code public étendu* est donc équivalent au *code caché étendu*. Pour plus de détails sur la construction du cryptosystème, le lecteur se reportera à la section 1.1 du chapitre 1.

Dans cette section, je décris d'abord le principe de l'*attaque structurelle* contre les codes de $\mathcal{G}$, puis je montre comment la généraliser aux *codes étendus*. Dans une troisième partie, je montre que, dans le cas des codes de GOPPA irréductibles, il est préférable au niveau de la complexité d'utiliser l'*attaque structurelle* sur les *codes étendus*.

1.2.1 Description de l'attaque structurelle

Dans le chapitre précédent, nous avons vu que le principe de l'*attaque structurelle* contre le système de chiffrement de McELIECE ou de manière équivalente contre le système de chiffrement de NIEDERREITER consistait à retrouver un décodeur du *code public*. Quand on utilise des familles de codes telles que les codes de REED–SOLOMON généralisés, l'attaque de SIDEL'NIKOV–SHESTAKOV y parvient en temps polynomial. Cette attaque utilise la structure des codes ainsi que les propriétés de transitivité de leur groupe d'automorphismes. Sa complexité est égale à $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$ où k est la dimension des codes. De même SENDRIER a montré que les *codes concaténés* étaient également à proscrire dans la conception de tels cryptosystèmes [Sen98]. Cependant, pour les codes de GOPPA, on ne connaît pas de tels raccourcis permettant de construire un algorithme polynomial de décodage du *code public*.

Donc, de façon générale, retrouver un algorithme de décodage du *code public* consiste à énumérer l'*espace des clés* et à trouver un code pour lequel on sache résoudre le problème suivant :

Problème 5 (Attaque structurelle) *Étant donné C_{pub} , le code public, de vecteur générateur $\mathbf{L}$, d'un cryptosystème de McELIECE, et étant donné l'espace des clés $\mathcal{G}$,*

1. *trouver un code $C \in \mathcal{G}$ de vecteur générateur $\mathbf{L}$ tel que C et C_{pub} soient équivalents,*
2. *trouver la permutation π de $\mathbf{L}$ telle que*

$$C = \pi(C_{\text{pub}}).$$

Si un attaquant résout le problème 5, il est en mesure de construire un décodeur permettant de déchiffrer n'importe quel message, préalablement chiffré par le cryptosystème de McELIECE de code public C_{pub} . Le code C joue le même rôle que le *code caché* puisque, comme on connaît la permutation π entre les deux codes, c'est l'algorithme de décodage de C qui permet le déchiffrement des messages.

En effet, soit $\mathbf{G}_{\text{pub}}$ la *clé publique* du cryptosystème de McELIECE, et soit

$$\mathbf{y} = \mathbf{x}\mathbf{G}_{\text{pub}} + \mathbf{e},$$

un mot chiffré. Si l'attaquant résout le problème 5 pour le code C_{pub} , alors il peut calculer

$$\pi^{-1}(\mathbf{y}) = \mathbf{x}\pi^{-1}(\mathbf{G}_{\text{pub}}) + \pi^{-1}(\mathbf{e}).$$

Comme $\mathbf{G}_{\text{pub}}$ est, par construction, une matrice génératrice du *code public* C_{pub} , alors $\pi^{-1}(\mathbf{G}_{\text{pub}})$ est une matrice génératrice de C . Or $C \in \mathcal{G}$, donc il existe un algorithme polynomial de décodage de C , permettant de décoder C jusqu'à la distance t . L'attaquant retrouve alors couple $(\mathbf{x}, \pi^{-1}(\mathbf{e}))$.

Le problème de la recherche d'un décodeur du *code public* est donc réduit au problème 5. Pour résoudre l'instance générale de ce problème, il n'existe pas d'autre moyen que d'énumérer l'*espace des clés* en testant l'équivalence de chaque élément avec le *code public* jusqu'à trouver un code équivalent. La procédure à suivre est détaillée dans le tableau 2.1.

Soit Γ le *code caché* du cryptosystème de McELIECE, et soit $\text{Cl}_{\mathcal{G}}(\Gamma)$ la *classe d'équivalence* de Γ dans la famille $\mathcal{G}$. Comme C_{pub} et Γ sont équivalents par construction, tout code $C \in \text{Cl}_{\mathcal{G}}(\Gamma)$ est également équivalent à C_{pub} . L'algorithme décrit dans le tableau 2.1 effectue donc en moyenne

$$\frac{|\mathcal{G}|}{|\text{Cl}(\Gamma)|} \text{ itérations.}$$

Données :

1. L'espace des clés $\mathcal{G}$ d'un cryptosystème de McELIECE, de vecteur générateur $\mathbf{L}$,
2. Le code public C_{pub} de vecteur générateur $\mathbf{L}$.

Résultats :

- Un code $C \in \mathcal{G}$ équivalent à C_{pub} .
- La permutation π de $\mathbf{L}$, telle que

$$C = \pi(C_{\text{pub}}).$$

Procédure :

Pour $C \in \mathcal{G}$, faire

- Tester si C et C_{pub} sont équivalents.
- Si C et C_{pub} sont équivalents, alors *sortir de la boucle*.

Déterminer la permutation π de $\mathbf{L}$ telle que

$$C = \pi(C_{\text{pub}}).$$

TABLEAU 2.1 : Attaque structurelle par énumération de l'espace des clés

1.2.2 Attaque utilisant les codes étendus

En utilisant les propriétés des *codes étendus*, on peut améliorer l'efficacité de l'*attaque structurelle*. Par construction, le *code public* C_{pub} et le *code caché* Γ d'un cryptosystème de McELIECE sont équivalents. De la proposition 10 on infère que les codes étendus correspondants $\widehat{C}_{\text{pub}}$ et $\widehat{\Gamma}$ sont également équivalents. On passe ainsi de la résolution du problème 5 à la résolution du problème suivant :

Problème 6 (Attaque structurelle sur les codes étendus) *Étant donné C_{pub} , le code public de vecteur générateur $\mathbf{L}$ d'un cryptosystème de McELIECE, et étant donné l'espace des clés $\mathcal{G}$ du cryptosystème,*

1. *trouver un code $C \in \mathcal{G}$ de vecteur générateur $\mathbf{L}$ tel que $\widehat{C}$ et $\widehat{C}_{\text{pub}}$ soient équivalents,*
2. *trouver la permutation π de $\widehat{\mathbf{L}} = \mathbf{L} \cup \infty$ telle que*

$$\widehat{C} = \pi(\widehat{C}_{\text{pub}}).$$

Si un attaquant résout ce problème, il est en mesure de construire un décodeur polynomial permettant de déchiffrer tous les messages, préalablement chiffrés par le cryptosystème de McELIECE de *code public* C_{pub} . En effet soit $\mathbf{G}_{\text{pub}}$ la *clé publique* du cryptosystème, et soit

$$\mathbf{y} = \mathbf{x}\mathbf{G}_{\text{pub}} + \mathbf{e},$$

un chiffré. L'attaquant calcule le *chiffré étendu*

$$\hat{\mathbf{y}} = \mathbf{x}\hat{\mathbf{G}}_{\text{pub}} + \hat{\mathbf{e}}$$

où $\hat{\mathbf{G}}_{\text{pub}}$ désigne la matrice génératrice de $\hat{C}_{\text{pub}}$ obtenue en étendant chaque ligne de $\mathbf{G}_{\text{pub}}$ avec la coordonnée à l'infini, comme présenté dans la définition 8. Ensuite, il calcule

$$\pi^{-1}(\hat{\mathbf{y}}) = \mathbf{x}\pi^{-1}(\hat{\mathbf{G}}_{\text{pub}}) + \pi^{-1}(\hat{\mathbf{e}}).$$

Comme $\hat{C} = \pi(\hat{C}_{\text{pub}})$, $\pi^{-1}(\hat{\mathbf{G}}_{\text{pub}})$ est donc une matrice génératrice de $\hat{C}$. En *poinçonnant* – voir la définition 9 – le vecteur $\mathbf{y}' = \pi^{-1}(\hat{\mathbf{y}})$ en la position $\pi^{-1}(\infty)$, il obtient l'équation

$$\mathbf{y}' = \mathbf{x}\mathbf{G}' + \mathbf{e}',$$

où $\mathbf{G}'$ est une matrice génératrice de C , et où $\mathbf{e}'$ désigne le *vecteur d'erreur* $\pi^{-1}(\hat{\mathbf{e}})$ poinçonné en $\pi^{-1}(\infty)$. Or $C \in \mathcal{G}$, donc il existe un algorithme polynomial de décodage de C jusqu'à la distance t . L'attaquant retrouve alors le couple $(\mathbf{x}, \mathbf{e}')$.

Le problème de la recherche du décodeur du *code public* est donc réduit au problème 6. Pour résoudre l'instance générale de ce problème, je ne connais pas d'autre moyen que d'énumérer l'*espace des clés* en testant, pour chaque élément, l'équivalence de son *code étendu* avec le *code public étendu*. La procédure à suivre est détaillée dans le tableau 2.2.

Soit Γ , le *code caché* du cryptosystème de McELIECE. Considérons la famille

$$\hat{\mathcal{G}} = \{\hat{C} \mid C \in \mathcal{G}\},$$

des *codes étendus* de $\mathcal{G}$. Soit $\text{Cl}_{\hat{\mathcal{G}}}(\Gamma)$ la classe d'équivalence de $\hat{\Gamma}$ dans $\hat{\mathcal{G}}$. Par construction, C_{pub} est équivalent au *code caché* Γ . D'après la proposition 10, les *codes étendus* $\hat{C}_{\text{pub}}$ et $\hat{\Gamma}$ sont équivalents. Donc, tout code de $\text{Cl}_{\hat{\mathcal{G}}}(\Gamma)$ est équivalent à $\hat{C}_{\text{pub}}$. Par conséquent, l'algorithme du tableau 2.2 doit effectuer en moyenne

$$\frac{|\hat{\mathcal{G}}|}{|\text{Cl}_{\hat{\mathcal{G}}}(\Gamma)|} = \frac{|\mathcal{G}|}{|\text{Cl}_{\mathcal{G}}(\Gamma)|} \text{ itérations.}$$

Comme $|\text{Cl}_{\hat{\mathcal{G}}}(\Gamma)| \geq |\text{Cl}_{\mathcal{G}}(\Gamma)|$, le nombre moyen d'itérations de l'algorithme 2.2 est moindre que le nombre moyen d'itérations de l'algorithme 2.1.

Données :

1. L'espace des clés $\mathcal{G}$ d'un cryptosystème de McELIECE, de vecteur générateur $\mathbf{L}$.
2. Le code public C_{pub} de vecteur générateur $\mathbf{L}$.

Résultat :

- Un code $C \in \mathcal{G}$ tel que $\widehat{C}$ soit équivalent à $\widehat{C}_{\text{pub}}$.
- La permutation π de $\widehat{\mathbf{L}} = \mathbf{L} \cup \infty$ telle que

$$\widehat{C} = \pi \left(\widehat{C}_{\text{pub}} \right).$$

Procédure :

Déterminer $\widehat{C}_{\text{pub}}$.

Pour $C \in \mathcal{G}$, faire

- Déterminer $\widehat{C}$.
- Tester si $\widehat{C}$ et $\widehat{C}_{\text{pub}}$ sont équivalents.
- Si $\widehat{C}$ et $\widehat{C}_{\text{pub}}$ sont équivalents, alors *sortir de la boucle*.

Trouver la permutation π de $\widehat{\mathbf{L}}$ telle que

$$\widehat{C} = \pi \left(\widehat{C}_{\text{pub}} \right).$$

TABLEAU 2.2 : Attaque structurelle sur les codes étendus

1.2.3 Application à la famille des codes de GOPPA irréductibles

Le codes de GOPPA sont présentés dans la section 2.2 du chapitre 1. Je rappelle qu'un code de GOPPA irréductible est un code de GOPPA dont le *polynôme générateur* est irréductible. Soit

$$\mathcal{G} = \{\Gamma(\mathbf{L}, g) \mid g(x) \in \mathbf{F}_{q^m}[x] \text{ est irréductible de degré } t\},$$

où $\Gamma(\mathbf{L}, g)$ désigne le code de GOPPA de corps générateur $\mathbf{F}_{q^m}$, de vecteur générateur $\mathbf{L}$, et de polynôme générateur g . Le cardinal de $\mathcal{G}$ est égal au nombre de polynômes irréductibles de degré t sur $\mathbf{F}_{q^m}$, soit

$$N = \frac{1}{t} \sum_{d|t} \mu(d) q^{mt/d},$$

où μ est la fonction de MOEBIUS – voir [LN97, p. 93] – définie par

$$\mu(d) = \begin{cases} 1 & \text{si } d = 1, \\ (-1)^k & \text{si } d \text{ est le produit de } k \text{ nombres premiers distincts,} \\ 0 & \text{si } d \text{ est divisible par le carré d'un nombre premier.} \end{cases}$$

Partant, pour les paramètres

$$q = 2, \quad t = 50, \quad \text{et} \quad m = 10,$$

la taille de $\mathcal{G}$ est approximativement égale à $|\mathcal{G}| \approx 2^{496}$.

Afin d'évaluer le nombre maximum moyen d'itérations des algorithmes 2.1 et 2.2, je minore la taille des classes d'équivalence de codes de GOPPA et de codes de GOPPA étendus.

- *Codes de GOPPA* : dans la section 1.1.2 nous avons vu que la taille des classes d'équivalence de codes de GOPPA de *corps générateur* $\mathbf{F}_{q^m}$ est égale à $m q^{2m}$. Donc, l'algorithme 2.1 effectue en moyenne

$$\frac{N}{m q^m} \approx \frac{q^{m(t-2)}}{m t} \text{ itérations.}$$

Pour les paramètres

$$q = 2, \quad t = 50, \quad \text{et} \quad m = 10,$$

on a $m q^m \approx 2^{23}$, le nombre d'itérations moyen est égal à 2^{473} .

- *Codes étendus* :

la proposition 10 de la section 1.1.2 exprime que l'action des homographies de $\mathbf{F}_{q^m}$ sur le *polynôme générateur* d'un *code de GOPPA étendu* donne des codes de GOPPA étendus équivalents. Les classes d'équivalence de codes de GOPPA étendus sont donc de tailles égales au nombre d'homographies de $\mathbf{F}_{q^m}$, c'est-à-dire

$$m(q^{3m} - q^m) \approx m q^{3m}.$$

Donc l'algorithme 2.2 effectue en moyenne

$$\frac{N}{m(q^{3m} - q^m)} \approx \frac{q^{m(t-3)}}{mt} \text{ itérations.}$$

Pour les paramètres

$$q = 2, \quad t = 50, \quad \text{et} \quad m = 10,$$

on a $m(q^{3m} - q^m) \approx 2^{33}$. Le nombre d'itérations moyen est égal 2^{463} .

L'attaque sur les *codes étendus* gagne un facteur q^m en moyenne. La *morale de l'histoire* est d'une part que les paramètres proposés par McELIECE sont extrêmement résistants aux *attaques structurelles*, et d'autre part qu'il est préférable d'utiliser les attaques structurelles sur les *codes étendus* en raison du gain appréciable en complexité.

1.3 Algorithme de séparation des supports

Pour évaluer la complexité des attaques structurelles permettant de résoudre les problèmes 5 et 6, j'ai négligé la difficulté de la résolution des problèmes de décision et de recherche suivants :

Problème 7 (Équivalence de codes) *Étant donnés deux codes C et C' de vecteur générateur $\mathbf{L}$, décider si C et C' sont équivalents.*

Problème 8 (Retrouver la permutation) *Étant donnés deux codes C et C' de vecteur générateur $\mathbf{L}$ qui sont équivalents, retrouver la permutation π de $\mathbf{L}$ telle que*

$$C' = \pi(C).$$

L'instance générale du problème 7 décidant de l'équivalence entre deux codes, est déjà un problème difficile. PETRANK et ROTH ont prouvé que l'instance générale de ce problème était au moins aussi difficile à résoudre que l'instance générale du problème de *l'isomorphisme de graphes*, voir [PR97].

Dans la plupart des cas cependant, l'*algorithme de séparation des supports* de Nicolas SENDRIER fournit un outil efficace permettant de résoudre d'abord le problème 7 et ensuite le problème 8 en temps polynomial. La description complète de cet algorithme se trouve dans l'article [Sen00].

Dans cette section, j'introduis d'abord les notions d'*invariant de code* et de *discriminante d'un invariant*, qui sous-tendent la construction de l'*algorithme de séparation des supports* – désigné par $\mathcal{SSA}$. Dans un deuxième temps, j'ébauche brièvement la structure de $\mathcal{SSA}$ qui permet de déterminer l'*invariant de code* le plus *discriminant* possible, et permet de résoudre le problème de décision 7 en temps polynomial sous certaines conditions. Je montre également que ces conditions sont remplies en ce qui concerne la famille des codes de GOPPA. Dans une troisième partie, je montre comment résoudre le problème de recherche 8 en utilisant $\mathcal{SSA}$.

1.3.1 Invariant de codes

Formellement, un *invariant de code* est une application qui prend en entrée un code et renvoie une valeur dans un espace où la notion d'égalité est définie, et qui prend la même valeur sur les *classes d'équivalence* de codes.

Définition 12 (Invariant de codes) Soit $\mathcal{G}$ une famille de codes de vecteur générateur $\mathbf{L}$. Soit $\mathcal{E}$ un ensemble sur lequel la notion d'égalité est définie. Alors toute application

$$\mathcal{I} : \mathcal{G} \mapsto \mathcal{E}$$

telle que pour tout code $C \in \mathcal{G}$, et pour toute permutation π de $\mathbf{L}$, on ait

$$\mathcal{I}(\pi(C)) = \mathcal{I}(C)$$

est appelée *invariant de la famille $\mathcal{G}$* .

Par définition, l'invariant $\mathcal{I}$ prend les mêmes valeurs sur les classes d'équivalence de codes. Mais la réciproque est fautive. En effet, le fait que, pour deux codes C et C' de $\mathcal{G}$ on ait

$$\mathcal{I}(C) = \mathcal{I}(C'),$$

n'implique pas en général que C et C' soient équivalents. La notion de discriminance d'un invariant introduit une relation d'ordre sur les *invariants* d'une famille $\mathcal{G}$.

Définition 13 (Discriminance d'un invariant) Soit $\mathcal{G}$ une famille de codes, et soit $\mathcal{I}$ et $\mathcal{J}$ deux invariants de $\mathcal{G}$. Soit $\mathcal{I}(\mathcal{G})$, respectivement $\mathcal{J}(\mathcal{G})$, l'ensemble des valeurs prises par $\mathcal{I}$ sur $\mathcal{G}$, respectivement l'ensemble des valeurs prises par $\mathcal{J}$ sur $\mathcal{G}$. Alors, $\mathcal{I}$ est plus discriminant que $\mathcal{J}$ si

$$|\mathcal{J}(\mathcal{G})| \leq |\mathcal{I}(\mathcal{G})|.$$

La relation «être plus discriminant que» est clairement une relation d'ordre de l'ensemble des invariants de $\mathcal{G}$. Partant, d'après la définition des invariants de codes, l'invariant le «plus discriminant possible» est celui qui permet de discriminer deux codes équivalents, *i.e.*

$$\mathcal{I}(C) = \mathcal{I}(C') \implies C \text{ et } C' \text{ sont équivalents,}$$

Mon objectif pour résoudre le problème 7 en temps raisonnable est donc de déterminer un *invariant de codes* $\mathcal{I}$ qui soit

1. le plus discriminant possible, l'idéal étant

$$\mathcal{I}(C) = \mathcal{I}(C') \implies C \text{ et } C' \text{ sont équivalents,}$$

2. de complexité calculatoire suffisamment petite.

Malheureusement, ces deux conditions sont en général incompatibles.

Exemple 6

Il existe un grand nombre d'invariants de codes. Par exemple :

- *Invariants fondamentaux* : la longueur et la dimension d'un code sont des invariants de l'ensemble des codes. Leur calcul est de complexité linéaire. Malheureusement, ils ne sont pas très discriminants. En effet, deux codes de même longueur ne sont en général pas équivalents.

La *distance minimale* d'un code est également un invariant de la famille des codes. En revanche, elle est, non seulement difficile à calculer – VARDY a prouvé que le problème du calcul de la distance minimale d'un code était NP-dur, voir [Var97] –, mais elle n'est pas non plus discriminante, puisque deux codes de même distance minimale ne sont pas équivalents en général.

- *Distribution des poids d'un code* : bien que deux codes ayant la même *distribution des poids* ne soient pas forcément équivalents, on considère que la *distribution des poids* d'un code est un invariant très *discriminant*. Néanmoins, déterminer la distribution des poids d'un code est un problème NP-dur. Étant donné que les codes utilisés habituellement dans le système de chiffrement de McELIECE sont de dimension au moins égale à 500, calculer leur *distribution des poids* est impraticable.
- *Distribution des poids du hull d'un code* : le *hull* d'un code est l'intersection d'un code avec son *dual*. En moyenne, la dimension du hull d'un code dépasse rarement quelques unités – voir tableau 2.3. Cet invariant est donc facile à calculer. De plus, il est assez discriminant. C'est l'invariant que Nicolas SENDRIER a retenu dans pour concevoir *SSA*.

1.3.2 Description et complexité de *SSA*

L'*algorithme de séparation des supports* dépassant très largement le cadre de cette thèse, je n'en donnerai ici qu'un bref aperçu, en soulignant les propriétés qui me serviront dans la suite.

Considérons tout d'abord le vecteur

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n)$$

qui étiquettera tous les mots de longueur n et jouera par conséquent le rôle du *vecteur générateur*. Soit $\mathcal{P}$ une partition étiquetée des éléments de $\mathbf{L}$, *i.e.* $\mathcal{P}$ est une partition de $\mathbf{L}$ telle que

$$\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_\ell\},$$

où chaque $\mathcal{P}_i$ est un ensemble d'éléments de $\mathbf{L}$ appelé *cellule* de la partition $\mathcal{P}$.

Définition 14 (Équivalence de partitions)

Deux partitions $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_\ell\}$ et $\mathcal{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \dots, \mathcal{Q}_\ell\}$ du même vecteur $\mathbf{L}$ sont dites *équivalentes* s'il existe une permutation ω de $\{1, \dots, \ell\}$ qui préserve la cardinalité des cellules des partitions, *i.e.*

$$\exists \omega, \text{ telle que, } \forall i \in \{1, \dots, \ell\}, \quad |\mathcal{P}_i| = |\mathcal{Q}_{\omega(i)}|.$$

On note alors

$$\mathcal{P} \sim \mathcal{Q}.$$

Exemple 7

Soit α , un élément primitif du corps $\mathbf{F}_{2^4}$, et soit

$$\mathbf{L} = (0, 1, \alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{12}, \alpha^3, \alpha^6, \alpha^9, \alpha^5, \alpha^{10}, \alpha^{11}, \alpha^{13}, \alpha^{14}, \alpha^7)$$

un étiquetage de $\mathbf{F}_{2^4}$. Considérons les deux partitions de $\mathbf{L}$:

$$\begin{aligned} \mathcal{P} &= (\underbrace{\{\alpha^2\}}_{\mathcal{P}_1}, \underbrace{\{\alpha^6\}}_{\mathcal{P}_2}, \underbrace{\{0, \alpha^5\}}_{\mathcal{P}_3}, \underbrace{\{1, \alpha^8, \alpha^{11}, \alpha^{12}\}}_{\mathcal{P}_4}, \underbrace{\{\alpha^3, \alpha^4, \alpha^{13}, \alpha^{15}\}}_{\mathcal{P}_5}, \underbrace{\{\alpha^7, \alpha^9, \alpha^{10}, \alpha^{14}\}}_{\mathcal{P}_6}), \\ \mathcal{Q} &= (\underbrace{\{0\}}_{\mathcal{Q}_1}, \underbrace{\{1\}}_{\mathcal{Q}_2}, \underbrace{\{\alpha^5, \alpha^{10}\}}_{\mathcal{Q}_3}, \underbrace{\{\alpha, \alpha^2, \alpha^4, \alpha^8\}}_{\mathcal{Q}_4}, \underbrace{\{\alpha^{12}, \alpha^3, \alpha^6, \alpha^9\}}_{\mathcal{Q}_5}, \underbrace{\{\alpha^{11}, \alpha^{13}, \alpha^{14}, \alpha^7\}}_{\mathcal{Q}_6}). \end{aligned}$$

$\mathcal{P}$ et $\mathcal{Q}$ sont équivalentes au sens de la définition 14. En effet, on a

$$\forall i \in (1, 2, \dots, 6), \quad |\mathcal{P}_i| = |\mathcal{Q}_i|.$$

Avec cette définition, le fonctionnement de $\mathcal{SSA}$ est le suivant :

- *Données* : un code C binaire de vecteur générateur $\mathbf{L}$.
- *Résultat* : un entier ℓ et une *partition étiquetée*

$$\mathcal{SSA}(C) = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_\ell\}$$

du vecteur générateur $\mathbf{L}$.

Je ne décrirai pas le fonctionnement de l'algorithme $\mathcal{SSA}$. Le lecteur intéressé pourra se reporter à l'article de Nicolas SENDRIER [Sen00]. Je considère $\mathcal{SSA}$ comme une boîte noire qui satisfait les propriétés suivantes :

Propriétés 3 (Résultats fournis par $\mathcal{SSA}$) Soit C et C' deux codes binaires de vecteur générateur $\mathbf{L}$.

1. Si C et C' sont équivalents, alors

$$\mathcal{SSA}(C) \sim \mathcal{SSA}(C').$$

2. Si $\mathcal{SSA}(C) \sim \mathcal{SSA}(C')$, alors il existe une permutation π de $\mathbf{L}$ telle que

$$C' = \pi(C).$$

3. Si

$$\mathcal{SSA}(C) = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_\ell\},$$

alors, les cellules $\mathcal{P}_i$ sont, modulo une permutation, les orbites des éléments de $\mathbf{L}$ sous l'action du groupe d'automorphismes de C , c'est-à-dire, si

$$\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_\ell\}$$

est une partition étiquetée de $\mathbf{L}$ dont les cellules $\mathcal{O}_i$ correspondent aux orbites des éléments de $\mathbf{L}$ sous l'action du groupe $\text{Aut}(C)$, alors

- (a) $\ell = \ell'$,
- (b) $\mathcal{O} \sim \mathcal{SSA}(C)$.

Dans l'absolu, toutes les propriétés de $\mathcal{SSA}$ ne sont pas vraies. On peut construire des codes «exotiques» qui ne satisfont ni la seconde ni la troisième propriété. Cependant, pour les paramètres habituellement utilisés dans le cryptosystème de McELIECE, les propriétés sont vraies «presque toujours».

1. Par construction, $\mathcal{SSA}$ détermine un *invariant de code* en utilisant l'invariant : *distribution des poids du hull* d'un code. La première propriété est donc vraie quels que soient les codes utilisés.
2. La deuxième propriété exprime que $\mathcal{SSA}$ détermine l'invariant le plus *discriminant* possible d'une famille de codes.
3. La troisième propriété montre que $\mathcal{SSA}(C)$ détermine en outre la structure du *groupe d'automorphismes* du code C modulo une permutation. Soit, par exemple, C un code de vecteur générateur

$$\mathbf{L} = \{\alpha_1, \alpha_2, \dots, \alpha_n\}.$$

Si le groupe d'automorphismes de C est trivial, alors la partition $\mathcal{O}$ du vecteur générateur $\mathbf{L}$ sous l'action de $\text{Aut}(C)$ est égale à

$$\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_n\}$$

où $\mathcal{O}_i = \alpha_i$. Donc les cellules de la partition étiquetée $\mathcal{SSA}(C)$ sont des singletons.

Proposition 12 (Complexité de $\mathcal{SSA}$) *La complexité algorithmique de $\mathcal{SSA}$ appliqué à un code C est égale à*

$$O(n^3 + 2^h n^2 \text{Log}_2(n))$$

opérations binaires où

- n est la longueur de C ,
- h est la dimension du hull de C ,

Comme $\mathcal{SSA}$ calcule des *énumérateurs des poids de hulls* de codes, il est normal que sa complexité soit exponentielle en fonction de la dimension du *hull* du code.

En pratique, cela constituerait un problème si la dimension du *hull* de C était élevée. Cependant SENDRIER a montré que la proportion de *codes binaires aléatoires* ayant un *hull* de taille donnée décroît exponentiellement avec la dimension du hull. Ce résultat est démontré dans [Sen97]. Le tableau 2.3 en donne un récapitulatif, et montre ainsi que la proportion de codes dont le hull est de dimension supérieure à 5 est négligeable.

Or le cryptosystème de McELIECE n'utilise pas de *codes binaires aléatoires*, mais des codes de GOPPA binaires aléatoires. Expérimentalement, la

	Dimension du hull				
	0	1	2	3	4
Taux	41.94%	41.94%	13.98%	2.00%	0.13%
Dimension moyenne	0.76				

TABLEAU 2.3 : Dimension des hulls de codes linéaires binaires.

répartition des dimensions des *hulls* des codes de GOPPA aléatoires est identique à celle des *codes binaires aléatoires*.

Afin de prévenir les *attaques par décodage*, le concepteur du cryptosystème utilise des codes dont la longueur est de l'ordre de plusieurs milliers. L'apport du facteur $2^h n^2 \log(n)$ est donc négligeable par rapport à n^3 dans la complexité de $\mathcal{SSA}$. Le temps de calcul de $\mathcal{SSA}(C)$ où C est un code de GOPPA binaire de longueur 1024, et de dimension 524 est en moyenne égal à 0,15s sur une station de travail Digital Alpha Processor à 500Mhz.

1.3.3 Recouvrement de la permutation entre codes équivalents

Par construction, l'algorithme $\mathcal{SSA}$ permet de décider de l'équivalence de deux codes C et C' de longueur n en temps $O(n^3)$. Pour finaliser les *attaques structurelles* décrites dans les tableaux 2.1 et 2.2, l'attaquant doit également recouvrir la permutation entre les deux codes équivalents C et C' .

Bien qu'il eut fallu rentrer dans des détails fastidieux de sa construction, $\mathcal{SSA}$ permet de retrouver la permutation en temps polynomial. Le principe est le suivant :

soit C et C' deux codes équivalents de *vecteur générateur*

$$\mathbf{L} = (\alpha_1, \dots, \alpha_n).$$

On cherche la permutation π de $\mathbf{L}$ telle que

$$C' = \pi(C).$$

La procédure à suivre pour l'attaquant est décrite dans le tableau 2.4. La complexité de la procédure est en gros égale à $O(n^4)$ opérations binaires.

2 Clés faibles du système

Dans la section 1.2.3, nous avons vu que les *attaques structurelles* contre le système de chiffrement de McELIECE sont impraticables, notamment lorsque l'*espace des clés* est constitué des codes de GOPPA irréductibles de longueur 1024 et de dimension 524. En effet, quelle que soit la rapidité avec laquelle on teste l'équivalence de codes, le nombre d'itérations des algorithmes des tableaux 2.1 et 2.2 reste gigantesque $> 2^{460}$. On considère

Données : deux codes binaires équivalents C et C' de vecteur générateur $\mathbf{L}$.
Résultat : la permutation π de $\mathbf{L}$ telle que

$$C' = \pi(C).$$

Pour tout $\alpha_i \in \mathbf{L}$,

1. l'attaquant détermine le code C_{α_i} qui est le code C poinçonné en la position α_i , *i.e.*

$$C_{\alpha_i} = \{(a_{\alpha_1}, \dots, a_{\alpha_{i-1}}, a_{\alpha_{i+1}}, \dots, a_{\alpha_n}) \mid (a_{\alpha_1}, \dots, a_{\alpha_n}) \in C\}.$$

Il calcule ensuite $\mathcal{SSA}(C_{\alpha_i})$,

2. *Répéter*

- Choisir $\beta \in \mathbf{L}$.
- Déterminer le code C'_{β} qui est le code C' poinçonné en la position β . Il calcule ensuite $\mathcal{SSA}(C'_{\beta})$.
- Jusqu'à ce que

$$\mathcal{SSA}(C_{\alpha_i}) \sim \mathcal{SSA}(C'_{\beta}).$$

3. On pose $\pi(\alpha_i) = \beta$.

TABLEAU 2.4: Recouvrement de la permutation entre deux codes équivalents à l'aide de $\mathcal{SSA}$

actuellement qu'une attaque qui prend entre 2^{55} et 2^{65} opérations binaires est d'une complexité conséquente mais raisonnable, compte tenu de la possible parallélisation des algorithmes.

Le but de cette section est de montrer que les codes de GOPPA dont le polynôme générateur est binaire constituent une famille de *clés faibles* du système de chiffrement de McELIECE, permettant de réduire considérablement la complexité des *attaques structurelles*.

En effet, en étudiant la structure du *code public* un attaquant peut

1. détecter si le polynôme générateur du code caché est binaire, restreignant ainsi le nombre d'itérations des *attaques structurelles*.
2. réduire la complexité des tests d'équivalence en utilisant les propriétés du *sous-code idempotent projeté* d'un code de GOPPA.

Je présente dans un premier temps des éléments de structure des codes de GOPPA de *polynôme générateur* binaires. Ceux-ci me servent à construire le *sous-code idempotent projeté*. Dans un second temps je montre comment, en appliquant *SSA* au *code public*, on peut détecter les *clés faibles*. Enfin, dans un troisième temps, je montre qu'on peut ramener le test de l'équivalence entre deux codes, au test de l'équivalence entre deux codes de longueur et de dimension plus petites.

2.1 Propriétés des codes de GOPPA de polynôme générateur binaire

Dans la suite et sans perte de généralité, je considère des codes de GOPPA $\Gamma(\mathbf{L}, g)$ de *corps générateur* $\mathbf{F}_{2^m}$, de polynôme générateur $g(x) \in \mathbf{F}_2[x]$, sans racines dans $\mathbf{F}_{2^m}$. Le *vecteur générateur* $\mathbf{L}$ est donc un étiquetage de $\mathbf{F}_{2^m}$. σ désigne l'*automorphisme de FROBENIUS* de l'extension $\mathbf{F}_{2^m}/\mathbf{F}_2$, c'est-à-dire

$$\forall z \in \mathbf{F}_{2^m}, \quad \sigma(z) = z^2.$$

Dans cette section, je montre que le *groupe d'automorphismes* de $\Gamma(\mathbf{L}, g)$, où $g(x) \in \mathbf{F}_2[x]$, contient le groupe

$$\langle \sigma \rangle = \{\mathbf{I}, \sigma, \dots, \sigma^{m-1}\}$$

d'ordre m . Ensuite, je montre que l'ensemble des mots de $\Gamma(\mathbf{L}, g)$ qui sont invariants sous l'action de σ – i.e. $\sigma(\mathbf{a}) = \mathbf{a}$ – est un sous-code non-trivial de $\Gamma(\mathbf{L}, g)$ appelé *sous-code idempotent*. Celui-ci est en bijection avec un code de longueur plus petite appelé *sous-code idempotent projeté*. Dans une troisième partie, je montre que la répartition des dimensions des hulls des codes de GOPPA, dont le polynôme générateur est binaire décroît exponentiellement avec la dimension, permettant l'utilisation de *SSA* en temps polynomial.

Les résultats des sections 2.1.1 et 2.1.2 sont généralisés et approfondis dans la section 1 du chapitre 3.

2.1.1 Groupe d'automorphismes

Proposition 13 Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA de corps générateur $\mathbf{F}_{2^m}$, et de polynôme générateur $g \in \mathbf{F}_2[x]$. Alors, le groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$ contient le groupe $\langle \sigma \rangle$ engendré par l'automorphisme de FROBENIUS de l'extension $\mathbf{F}_{2^m}/\mathbf{F}_2$, i.e.

$$\{\mathbf{I}, \sigma, \sigma^2, \dots, \sigma^{m-1}\} \subset \text{Aut}(\Gamma(\mathbf{L}, g)).$$

Preuve

Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA de corps générateur $\mathbf{F}_{2^m}$, de polynôme générateur $g \in \mathbf{F}_2[x]$, et de vecteur générateur

$$\mathbf{L} = (\alpha_1, \dots, \alpha_n).$$

Le mot binaire $\mathbf{a} = (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n})$ de vecteur générateur $\mathbf{L}$ appartient au code de GOPPA $\Gamma(\mathbf{L}, g)$ si et seulement si ses coordonnées a_{α_i} vérifient

$$\sum_{i=1}^n a_{\alpha_i} \frac{\alpha_i^j}{g(\alpha_i)} = 0,$$

pour $j = 0, \dots, t-1$. L'action de σ sur le vecteur $\mathbf{a}$ s'écrit

$$\sigma(\mathbf{a}) = (a_{\beta_1}, a_{\beta_2}, \dots, a_{\beta_n}),$$

où $\beta_i^2 = \alpha_i$. Comme, par hypothèse, le polynôme g est à coefficients dans $\mathbf{F}_2$, on a $g(\beta_i^2) = g(\alpha_i)^2$. De plus, comme $a_{\alpha_i} \in \mathbf{F}_2$, on a $a_{\alpha_i}^2 = a_{\alpha_i}$. Donc

$$\sum_{i=1}^n a_{\beta_i} \frac{\alpha_i^j}{g(\alpha_i)} = \sum_{i=1}^n a_{\beta_i} \frac{(\beta_i^j)^2}{g(\alpha_i)^2} = \left(\sum_{i=1}^n a_{\beta_i} \frac{\beta_i^j}{g(\beta_i)} \right)^2 = 0$$

pour $j = 0, \dots, t-1$. Donc $\sigma(\mathbf{a}) \in \Gamma(\mathbf{L}, g)$.

■

Au cours de simulations numériques effectuées durant mon stage de DEA – voir [Loi97] –, j'ai constaté que le groupe d'automorphismes des codes de GOPPA, de polynôme générateur binaire était presque toujours égal à $\langle \sigma \rangle$. Dans les quelques cas où ce n'était pas vrai, le polynôme générateur avait une structure particulière, des symétries en général.

2.1.2 Sous-code idempotent d'un code de GOPPA

Soit α un élément de $\mathbf{F}_{2^m}$. Soit

$$\mathcal{L}_\alpha = \{\alpha, \sigma(\alpha), \dots, \sigma^{m-1}(\alpha)\},$$

l'orbite de α sous l'action de $\langle \sigma \rangle$. Le cardinal $\mathcal{L}_\alpha$ est égal au degré de l'extension de $\mathbf{F}_2(\alpha)/\mathbf{F}_2$, où $\mathbf{F}_2(\alpha)$ désigne le plus petit sous corps de $\mathbf{F}_{2^m}$ contenant α .

Exemple 8

Soit α un élément primitif du corps $\mathbf{F}_{2^4}$. Les éléments de $\mathbf{F}_{2^4}$ sont alors répartis de la manière suivante

$$\mathbf{F}_{2^4} = \left\{ \underbrace{0}_{\mathcal{L}_0}, \underbrace{1}_{\mathcal{L}_1}, \underbrace{\alpha, \alpha^2, \alpha^4, \alpha^8}_{\mathcal{L}_\alpha}, \underbrace{\alpha^{12}, \alpha^3, \alpha^6, \alpha^9}_{\mathcal{L}_{\alpha^3}}, \underbrace{\alpha^5, \alpha^{10}}_{\mathcal{L}_{\alpha^5}}, \underbrace{\alpha^{11}, \alpha^{13}, \alpha^{14}, \alpha^7}_{\mathcal{L}_{\alpha^7}} \right\},$$

où chaque $\mathcal{L}_\beta$ désigne l'orbite de l'élément β sous l'action de σ .

Définition 15 (Sous-code idempotent) Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA où $g(x) \in \mathbf{F}_2[x]$, de vecteur générateur

$$\mathbf{L} = (\alpha_1, \dots, \alpha_n).$$

Un mot binaire $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n})$ est appelé idempotent si

$$\sigma(\mathbf{a}) = \mathbf{a}.$$

L'ensemble de tous les mots idempotents de $\Gamma(\mathbf{L}, g)$ est un sous-code de $\Gamma(\mathbf{L}, g)$ noté $I(\mathbf{L}, g)$ et appelé sous-code idempotent de $\Gamma(\mathbf{L}, g)$.

Le support d'un mot de vecteur générateur $\mathbf{L}$ est un sous-ensemble de $\mathbf{L}$ qui étiquette les coordonnées non nulles du mot. Dans le cas d'un mot binaire idempotent, le support du mot a une forme particulière.

Proposition 14 Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA où $g(x) \in \mathbf{F}_2[x]$, de vecteur générateur

$$\mathbf{L} = (\alpha_1, \dots, \alpha_n).$$

Soit $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n})$, un mot binaire de vecteur générateur $\mathbf{L}$. Alors, les deux assertions suivantes sont équivalentes :

1. $\mathbf{a} = \sigma(\mathbf{a})$, où σ désigne l'automorphisme de FROBENIUS de $\mathbf{F}_{2^m}/\mathbf{F}_2$,
2. le support de $\mathbf{a}$ est une réunion d'orbites

$$\mathcal{L}_\beta = \{\beta, \sigma(\beta), \dots, \sigma^{m-1}(\beta)\},$$

pour certains $\beta \in \mathbf{F}_{2^m}$.

Preuve

Soit $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n})$ un mot binaire de vecteur générateur $\mathbf{L}$. Alors, d'après la définition 6, on a

$$\sigma(\mathbf{a}) = (a_{\beta_1}, \dots, a_{\beta_n}),$$

où $\beta_i = \sigma^{-1}(\alpha_i)$. Si $\mathbf{a} = \sigma(\mathbf{a})$, alors $a_{\beta_i} = a_{\alpha_i}$, donc, par transitivité, on obtient

$$a_{\alpha_i} = a_{\sigma(\alpha_i)} = \dots = a_{\sigma^{m-1}(\alpha_i)}.$$

Par conséquent, le mot binaire $\mathbf{a}$ prend la même valeur – 1 ou 0 – sur les positions étiquetées par des éléments de la même orbite.

La réciproque est immédiate.

■

Soit $\mathbf{R} = (o_1, o_2, \dots, o_N)$ un ensemble de représentants des orbites qui constituent $\mathbf{L}$, c'est-à-dire

$$\mathbf{L} = \cup_{i=1}^N \mathcal{L}_{o_i}, \quad \text{où } \mathcal{L}_{o_i} = \{o_i, \sigma(o_i), \dots, \sigma^{m-1}(o_i)\}.$$

Soit $\kappa_{\mathbf{R}}$ la projection de $\mathbf{F}_2^n$ dans $\mathbf{F}_2^N$ qui, à partir d'un mot $\mathbf{a}$ de *vecteur générateur* $\mathbf{L}$, construit le mot de *vecteur générateur* $\mathbf{R}$:

$$\begin{aligned} \kappa_{\mathbf{R}} : \quad \mathbf{F}_2^n &\rightarrow \mathbf{F}_2^N \\ \mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n}) &\mapsto \tilde{\mathbf{a}} = (a_{o_1}, \dots, a_{o_N}). \end{aligned} \quad (2.1)$$

Le mot $\tilde{\mathbf{a}} = \kappa_{\mathbf{R}}(\mathbf{a})$ s'obtient à partir de $\mathbf{a}$ en gardant les coordonnées étiquetées par $\mathbf{R}$. Pour tout mot $\mathbf{b}$ dans $\mathbf{F}_2^N$, on note $\kappa_{\mathbf{R}}^{-1}(\mathbf{b})$ l'unique mot *idempotent* de *vecteur générateur* $\mathbf{L}$, tel que

$$\kappa_{\mathbf{R}}(\kappa_{\mathbf{R}}^{-1}(\mathbf{b})) = \mathbf{b}.$$

$\kappa_{\mathbf{R}}^{-1}(\mathbf{b})$ s'obtient de $\mathbf{b} = (b_{o_1}, \dots, b_{o_N})$ en dupliquant chaque position b_{o_i} exactement $|\mathcal{L}_{o_i}|$ fois. Si on applique $\kappa_{\mathbf{R}}$ à tous les mots du *sous-code idempotent* $I(\mathbf{L}, g)$ défini dans la définition 15, on obtient le code linéaire

$$I(\mathbf{R}, g) = \{\kappa_{\mathbf{R}}(\mathbf{a}) \mid \mathbf{a} \in I(\mathbf{L}, g)\},$$

qui a la même dimension que $I(\mathbf{L}, g)$ mais dont la longueur est égale à $N \approx n/m$.

Définition 16 *Le code $I(\mathbf{R}, g)$ ainsi obtenu est appelé sous-code idempotent projeté dérivé de $\Gamma(\mathbf{L}, g)$.*

Lorsque le polynôme générateur g d'un code de GOPPA est binaire, le *sous-code idempotent* dérivé n'est pas trivial. Une matrice de parité de sa restriction à un vecteur $\mathbf{R}$ s'exprime en fonction de g et de $\mathbf{R}$. Cette propriété donne un moyen de construire le *sous-code idempotent projeté*, et fournit également une borne inférieure sur sa dimension.

Pour tout $\alpha \in \mathbf{F}_{2^m}$, $\mathbf{F}_2(\alpha)$ désigne le plus petit sous-corps de $\mathbf{F}_{2^m}$ qui contient α . Je désigne également par Tr_{α} l'*opérateur trace* de l'extension $\mathbf{F}_2(\alpha)/\mathbf{F}_2$ et par $\mathcal{L}_{\alpha}$, l'orbite de α sous l'action de σ . Avec ces notations, l'*opérateur trace* s'écrit

$$\begin{aligned} \text{Tr}_{\alpha} : \quad \mathbf{F}_2(\alpha) &\rightarrow \mathbf{F}_2 \\ \beta &\mapsto \sum_{i=0}^{|\mathcal{L}_{\alpha}|-1} \sigma^i(\beta). \end{aligned}$$

Afin de simplifier les notations, je suppose sans perte de généralité que les N premiers éléments du *vecteur générateur*

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n)$$

forment un ensemble de représentants des orbites de $\mathbf{F}_{2^m}$ sous l'action de l'automorphisme de FROBENIUS, c'est-à-dire que le vecteur $\mathbf{R}$ est égal à

$$\mathbf{R} = (\alpha_1, \dots, \alpha_N).$$

Proposition 15 Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA, où $g(x) \in \mathbf{F}_2[x]$, et est de degré t . Alors, la matrice binaire

$$\mathbf{H}_I = \begin{pmatrix} \text{Tr}_{\alpha_1} \left(\frac{1}{g(\alpha_1)} \right) & \cdots & \text{Tr}_{\alpha_N} \left(\frac{1}{g(\alpha_N)} \right) \\ \text{Tr}_{\alpha_1} \left(\frac{\alpha_1}{g(\alpha_1)} \right) & \cdots & \text{Tr}_{\alpha_N} \left(\frac{\alpha_N}{g(\alpha_N)} \right) \\ \vdots & \ddots & \vdots \\ \text{Tr}_{\alpha_1} \left(\frac{\alpha_1^{t-1}}{g(\alpha_1)} \right) & \cdots & \text{Tr}_{\alpha_N} \left(\frac{\alpha_N^{t-1}}{g(\alpha_N)} \right) \end{pmatrix}$$

est une matrice de parité de $I(\mathbf{R}, g)$ – le sous-code idempotent projeté dérivé de $\Gamma(\mathbf{L}, g)$.

Preuve

Soit $\tilde{\mathbf{a}} = (a_\alpha)_{\alpha \in \mathbf{R}} \in I(\mathbf{R}, g)$ et soit $\mathbf{a} = (a_\alpha)_{\alpha \in \mathbf{L}} = \kappa_{\mathbf{R}}^{-1}(\tilde{\mathbf{a}})$. Par construction de $\kappa_{\mathbf{R}}$, $\mathbf{a} \in \Gamma(\mathbf{L}, g)$ et vérifie par conséquent les équations de parité suivantes – voir la section 2.2 du chapitre 1 :

$$\forall j = 0, \dots, t-1, \quad \sum_{\alpha \in \mathbf{L}} a_\alpha \frac{\alpha^j}{g(\alpha)} = 0.$$

De plus, comme $g(x) \in \mathbf{F}_2[x]$, on a $g(\alpha^{2^i}) = g(\alpha)^{2^i}$. Par définition de l'opérateur Tr_α , et comme $\mathbf{a}$ est un mot *idempotent*, on a en outre

$$\forall j = 0, \dots, t-1, \quad \sum_{\alpha \in \mathbf{L}} a_\alpha \frac{\alpha^j}{g(\alpha)} = \sum_{\alpha \in \mathbf{R}} a_\alpha \text{Tr}_\alpha \left(\frac{\alpha^j}{g(\alpha)} \right) = 0.$$

De ceci, on déduit un ensemble d'équations de parité satisfaites par le mot $\tilde{\mathbf{a}}$:

$$\tilde{\mathbf{a}} \in I(\mathbf{R}, g) \iff \mathbf{H}_I \tilde{\mathbf{a}}^T = 0.$$

Comme g a tous ses coefficients dans $\mathbf{F}_2$, $\frac{\alpha^j}{g(\alpha)} \in \mathbf{F}_2(\alpha)$ pour tout j . Les coefficients de la matrice $\mathbf{H}_I$ sont donc binaires. En conséquence, $\mathbf{H}_I$ est une matrice de parité de $I(\mathbf{R}, g)$.

■

De la proposition 15 on déduit une borne inférieure sur la dimension des sous-codes idempotents projetés.

Corollaire 4 Si $g(x) \in \mathbf{F}_2[x]$ est de degré t , alors la dimension de $I(\mathbf{R}, g)$ est supérieure ou égale à $N - t$.

Exemple 9

Je reprends l'exemple 8, où α est un élément primitif du corps $\mathbf{F}_{2^4}$. Le polynôme $g(x) = x^3 + x + 1$ n'a pas de racines dans $\mathbf{F}_{2^4}$. Soit

$$\mathbf{L} = \left(\underbrace{0}_{\mathcal{L}_0}, \underbrace{1}_{\mathcal{L}_1}, \underbrace{\alpha, \alpha^2, \alpha^4, \alpha^8}_{\mathcal{L}_\alpha}, \underbrace{\alpha^{12}, \alpha^3, \alpha^6, \alpha^9}_{\mathcal{L}_{\alpha^3}}, \underbrace{\alpha^5, \alpha^{10}}_{\mathcal{L}_{\alpha^5}}, \underbrace{\alpha^{11}, \alpha^{13}, \alpha^{14}, \alpha^7}_{\mathcal{L}_{\alpha^7}} \right),$$

un étiquetage de $\mathbf{F}_{2^4}$. Alors, le code de GOPPA $\Gamma(\mathbf{L}, g)$ est un code de longueur 16, de matrice génératrice

$$\mathbf{G}_\Gamma = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array} \right).$$

Le sous-code idempotent $I(\mathbf{L}, g)$ de $\Gamma(\mathbf{L}, g)$ est de longueur $n = 16$, de dimension 3, et de matrice génératrice

$$\mathbf{G}_I = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right).$$

Soit

$$\mathbf{R} = (0, 1, \alpha, \alpha^3, \alpha^5, \alpha^{11}),$$

un étiquetage d'un ensemble de représentants des orbites constituant $\mathbf{L}$. Alors, le sous-code idempotent projeté dérivé de $\Gamma(\mathbf{L}, g)$ est un code de longueur $|\mathbf{R}| = 6$, dont la matrice génératrice est obtenue en prenant les colonnes de la matrice $\mathbf{G}_I$ qui sont étiquetées par les éléments de $\mathbf{R}$:

$$\mathbf{R} = (0, 1, \alpha, \alpha^{12}, \alpha^5, \alpha^{11})$$

$$\mathbf{G}_{\tilde{I}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

2.1.3 Répartition des dimensions des hulls

Dans la section 1.3, nous avons vu que la complexité de $\mathcal{SSA}$ était polynomiale, pourvu que la dimension des hulls des codes considérés soit petite. Nous avons vu également que cette hypothèse était légitime pour les codes de GOPPA binaires. Cependant, j'étudie dans la suite l'ensemble des *clés faibles*, c'est-à-dire les codes de GOPPA de polynôme générateur binaire. La question qu'on peut alors se poser devient :

peut-on encore considérer que l'hypothèse sur la petitesse de la dimension du hull d'une *clé faible* est légitime ?

SKERSYS a montré dans [Ske01] que la taille du groupe d'automorphismes d'un code influait sur la taille de son hull. Si la dimension des hulls des codes considérés dépasse régulièrement 12 ou 13, $\mathcal{SSA}$ ne peut plus être considéré comme un algorithme de complexité polynomiale.

Lors de mon stage de DEA – voir [Loi97] –, j'ai fait des simulations sur la répartition des dimensions des hulls de codes de GOPPA de polynôme générateur binaire. Les résultats sont présentés dans les tableaux 2.5 et 2.6.

Au final, il s'avère que la proportion de codes ayant des hulls de dimension importante est négligeable. On peut donc légitimement tenir la complexité de $\mathcal{SSA}$ pour polynomiale quand il est utilisé avec des codes de GOPPA de *polynôme générateur* binaire.

Dimension du Hull	Pourcentage de Polynômes tirés de Degré 16	Pourcentage de Polynômes tirés de Degré 32	Pourcentage de Polynômes tirés de Degré 50
0	32.3	34.8	33
1	16.8	16.8	16.9
2	21.9	21.8	22
3	3.5	4.9	5
4	8	8	9
5	3.6	3.4	3.3
6	4.5	4.7	4.2
7	1.15	1.2	0.7
8	3.6	2.7	2.6
9	1.8	1.4	1
10	1.6	1.7	1.5
11	0.6	0.4	0.4
12	0.3	0.2	0.2
Échantillon	4000	4000	7053

TABLEAU 2.5: Répartition des dimensions des hulls des codes de GOPPA de longueur 1024, de polynôme générateur binaire

2.2 Détection des clés faibles

Soit le cryptosystème de McELIECE dont

- le *code caché* est un code de GOPPA $\Gamma(\mathbf{L}, g)$, de *polynôme générateur* g sans racines dans $\mathbf{F}_{2^m}$, et de *vecteur générateur* $\mathbf{L}$, un étiquetage de $\mathbf{F}_{2^m}$.
- Le *code public* est le code C_{pub} , équivalent à $\Gamma(\mathbf{L}, g)$.

Soit la partition étiquetée de $\mathbf{L}$ suivante :

$$\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_N\},$$

où les cellules $\mathcal{O}_i$ sont les orbites des éléments de $\mathbf{F}_{2^m}$ sous l'action de l'*automorphisme de FROBENIUS* σ de $\mathbf{F}_{2^m}/\mathbf{F}_2$, *i.e.*

$$\mathcal{O}_i = \{o_i, \sigma(o_i), \dots, \sigma^{m-1}(o_i)\}.$$

Supposons que $g(x) \in \mathbf{F}_2[x]$. Dans la section 2.1.1 on a montré que le groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$ était égal au groupe engendré par σ . Donc d'après la troisième assertion des propriétés 3,

$$\mathcal{SSA}(\Gamma(\mathbf{L}, g)) \sim \mathcal{O}.$$

Or, par construction, $\Gamma(\mathbf{L}, g)$ et C_{pub} sont équivalents. Donc, d'après la première assertion des propriétés 3, $\mathcal{SSA}(\Gamma(\mathbf{L}, g))$ et $\mathcal{SSA}(C_{\text{pub}})$ sont équivalentes au sens de la définition 14. La relation d'équivalence étant transitive, on a aussi

$$\mathcal{SSA}(C_{\text{pub}}) \sim \mathcal{O}.$$

Dimension du Hull	Pourcentage de Polynômes tirés de Degré 16	Pourcentage d'irréductibles de Degré 16
0	22	24.2
1	22	20.6
2	21.5	20.5
3	14.6	16.1
4	7.6	7.6
5	2.7	2.2
6	2.9	2
7	2.5	2.1
8	2.1	2.8
9	1.2	1
10	0.7	0.8
échantillon	6000	2396

TABLEAU 2.6: Répartition des dimensions des hulls des codes de GOPPA de longueur 512, de polynôme générateur binaire

Pour reconnaître si le *polynôme générateur* du code caché est binaire, on suppose que la réciproque est vraie, *i.e.* si

$$\mathcal{SSA}(C_{\text{pub}}) \sim \mathcal{O},$$

alors $\Gamma(\mathbf{L}, g)$ est un code de GOPPA où $g(x) \in \mathbf{F}_2[x]$.

Un attaquant ne dispose que du *code public* pour détecter une *clé faible*. Pour ce faire :

1. Il détermine la partition du *vecteur générateur* $\mathbf{L}$

$$\mathcal{SSA}(C_{\text{pub}}) = \{\mathcal{P}_1, \dots, \mathcal{P}_\ell\}.$$

Cette étape coûte $\mathcal{O}(n^3)$ opérations binaires, d'après les résultats obtenus dans la section 2.1.3.

2. Ensuite il teste l'égalité $N = \ell$. Si on a bien l'égalité, alors il cherche une permutation ω de $\{1, 2, \dots, N\}$ telle que

$$\forall i \in \{1, 2, \dots, N\}, \quad |\mathcal{P}_{\omega(i)}| = |\mathcal{O}_i|.$$

S'il trouve une telle permutation, alors par définition

$$\mathcal{SSA}(C_{\text{pub}}) \sim \mathcal{O}.$$

Donc le *polynôme générateur* g du *code caché* est binaire. Cette étape est de faible complexité comparée au coût du calcul de $\mathcal{SSA}(C_{\text{pub}})$.

Exemple 10

Soit Γ , un code de GOPPA de longueur $n = 2^4 = 16$ dont le polynôme générateur est de degré $t = 3$. Soit α un élément primitif de $\mathbf{F}_{2^4}$. Écrivons le vecteur générateur $\mathbf{L}$ sous la forme

$$\mathbf{L} = (0, 1, \alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{12}, \alpha^3, \alpha^6, \alpha^9, \alpha^5, \alpha^{10}, \alpha^{11}, \alpha^{13}, \alpha^{14}, \alpha^7).$$

Soit une partition $\mathcal{O}$ des éléments de $\mathbf{L}$ définie par

$$\mathcal{O} = (\underbrace{\{0\}}_{\mathcal{O}_0}, \underbrace{\{1\}}_{\mathcal{O}_1}, \underbrace{\{\alpha, \alpha^2, \alpha^4, \alpha^8\}}_{\mathcal{O}_\alpha}, \underbrace{\{\alpha^{12}, \alpha^3, \alpha^6, \alpha^9\}}_{\mathcal{O}_{\alpha^3}}, \underbrace{\{\alpha^5, \alpha^{10}\}}_{\mathcal{O}_{\alpha^5}}, \underbrace{\{\alpha^{11}, \alpha^{13}, \alpha^{14}, \alpha^7\}}_{\mathcal{O}_{\alpha^7}})$$

Si l'algorithme $\mathcal{SSA}$ appliqué au code Γ renvoie la partition

$$(\underbrace{\{\alpha^2\}}_{\mathcal{P}_1}, \underbrace{\{\alpha^6\}}_{\mathcal{P}_2}, \underbrace{\{0, \alpha^5\}}_{\mathcal{P}_3}, \underbrace{\{1, \alpha^8, \alpha^{11}, \alpha^{12}\}}_{\mathcal{P}_4}, \underbrace{\{\alpha^3, \alpha^4, \alpha^{13}, \alpha^{15}\}}_{\mathcal{P}_5}, \underbrace{\{\alpha^7, \alpha^9, \alpha^{10}, \alpha^{14}\}}_{\mathcal{P}_6}),$$

alors les partitions $\mathcal{SSA}(\Gamma)$ et $\mathcal{O}$ sont équivalentes et le code Γ est une *clé faible*.

2.3 Attaques structurelles contre les clés faibles

Soit le cryptosystème de McELIECE dont

- le *code caché* est un code de GOPPA $\Gamma(\mathbf{L}, g)$, où $g(x) \in \mathbf{F}_2[x]$,
- le *code public* est le code C_{pub} , équivalent à $\Gamma(\mathbf{L}, g)$.

Quand l'attaquant a détecté l'utilisation d'une *clé faible*, les *attaques structurelles* des section 1.2.1 et 1.2.2 reviennent à énumérer les codes de GOPPA de *polynôme générateur* binaire et à tester leur équivalence avec le *code public*.

Dans un premier temps, je décris brièvement l'algorithme permettant de retrouver un code de GOPPA, de *polynôme générateur* binaire, équivalent au *code public*. J'utilise pour cela les propriétés de l'algorithme $\mathcal{SSA}$. Dans un second temps, je montre que grâce aux propriétés du *sous-code idempotent projeté*, on réduit le coût d'une itération de $\mathcal{SSA}$ de $O(n^3)$ – où n est la longueur des codes – à $O(\ell^3)$ – où ℓ est la longueur des *sous-codes idempotents projetés* dérivés. Par souci de commodité, je ne considère dans la suite que la famille des *codes de GOPPA irréductibles*.

2.3.1 Attaque générale

Le problème 5 de l'*attaque structurelle* devient :

Problème 9 (Problème général)

1. Trouver un polynôme irréductible $g(x) \in \mathbf{F}_2[x]$ tel que $\Gamma(\mathbf{L}, g)$ soit équivalent au code public C_{pub} .
2. Trouver la permutation π de $\mathbf{L}$, telle que

$$\Gamma(\mathbf{L}, g) = \pi(C_{\text{pub}}).$$

D'après la deuxième assertion des propriétés 3, $\mathcal{SSA}$ permet de décider si deux codes binaires sont équivalents, en $O(n^3)$ opérations binaires. Pour trouver un code de GOPPA équivalent au *code public*, on énumère la famille des codes de GOPPA de polynôme générateur binaire, puis on teste leur équivalence avec C_{pub} . L'algorithme est décrit dans le tableau 2.7.

Une fois qu'on a trouvé un code de GOPPA satisfaisant, on utilise $\mathcal{SSA}$ pour recouvrer la permutation en $O(n^4)$ opérations binaires – voir la section 1.3.3.

Données : le *code public* C_{pub} issu d'une *clé faible*.

Résultats : un polynôme irréductible $g(x) \in \mathbf{F}_2[x]$, de degré t tel que $\Gamma(\mathbf{L}, g)$ et C_{pub} soient équivalents.

Calculer $\mathcal{SSA}(C_{\text{pub}}) = \mathcal{P}$.

Pour tout $g(x) \in \mathbf{F}_2[x]$ irréductible de degré t , faire

- Déterminer le code $\Gamma(\mathbf{L}, g)$.
- Si $\mathcal{SSA}(\Gamma(\mathbf{L}, g)) \sim \mathcal{P}$, alors retourner g .

TABLEAU 2.7: Attaque structurelle contre les *clés faibles* lorsque l'espace des clés est la famille des codes de GOPPA irréductibles de degré t

Complexité de l'algorithme :

soit I_t le nombre de codes de GOPPA irréductibles dont le polynôme générateur est binaire, de degré t . Le coût d'une itération de $\mathcal{SSA}$ est égale à λn^3 opérations binaires, où λ désigne un paramètre qui dépend de l'implémentation. Le coût de l'algorithme du tableau 2.7 est alors égal à $\lambda n^3 I_t$ opérations binaires, en moyenne.

- Avec les *paramètres de McELIECE* : le nombre I_t correspond au nombre de polynômes binaires de degré 50 irréductibles sur $\mathbf{F}_{2^{10}}$ soit,

$$I_t \approx 2^{44,3}.$$

De plus, un test avec $\mathcal{SSA}$, de coût λn^3 , peut être effectué en 0,15 secondes sur une station de travail Digital Alpha Processor à 500Mhz. Sur cette station, le coût total de l'algorithme est supérieur à 10^5 années.

- Avec d'autres *paramètres* : si on prend

$$m = 9, \quad n = 512 = 2^9, \quad t = 28,$$

alors

$$I_t \approx 2^{25},$$

et le temps de calcul total n'excède par quelques jours.

2.3.2 Utilisation du sous-code idempotent

Une itération de $\mathcal{SSA}$ coûte λn^3 opérations binaires, où n désigne la longueur des codes. Ici, se révèle l'intérêt pratique du *sous-code idempotent projeté* dérivé d'un code de GOPPA, et introduit dans la section 2.1.2. En effet, si $\mathbf{F}_{2^m}$ est le *corps générateur* des codes de GOPPA, alors la complexité de l'attaque structurelle décrite dans le tableau 2.7 est réduite par un facteur m^3 . Comme les paramètres pratiques sont $m = 10$, le coût d'une itération de $\mathcal{SSA}$ est réduite d'un facteur 1000.

Soit C_{pub} , le code public d'un cryptosystème de McELIECE, de vecteur générateur

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n).$$

Soit

$$\mathcal{SSA}(C_{\text{pub}}) = \{\mathcal{P}_1, \dots, \mathcal{P}_\ell\},$$

et soit $E_{\mathcal{P}}$ l'ensemble des mots $\mathbf{a} = (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n}) \in \mathbf{F}_2^n$, de vecteur générateur $\mathbf{L}$ tels que, pour toute cellule $\mathcal{P}_i$ on ait,

$$\forall \beta, \beta' \in \mathcal{P}_i, \quad a_\beta = a_{\beta'}.$$

$E_{\mathcal{P}}$ est un code binaire de vecteur générateur $\mathbf{L}$. Soit alors

$$I_{\text{pub}} = E_{\mathcal{P}} \cap C_{\text{pub}}.$$

Par construction, I_{pub} est un code binaire dont le *support* est une réunion de cellules $\mathcal{P}_i$ – toutes les coordonnées étiquetées par des éléments de $\mathbf{L}$ appartenant à $\mathcal{P}_i$ sont identiques. Soit $\mathbf{T} = (p_1, \dots, p_\ell)$, où $p_i \in \mathcal{P}_i$, un sous-vecteur de $\mathbf{L}$ contenant exactement un élément dans chaque cellule $\mathcal{P}_i$. Soit $\kappa_{\mathbf{T}}$ la projection sur les positions étiquetées par $\mathbf{T}$. Elle est définie par

$$\begin{aligned} \kappa_{\mathbf{T}} : \quad \mathbf{F}_2^n &\rightarrow \mathbf{F}_2^\ell \\ (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n}) &\mapsto (a_{p_1}, \dots, a_{p_\ell}). \end{aligned} \quad (2.2)$$

Alors, le code défini par

$$\tilde{I}_{\text{pub}} = \{\kappa_{\mathbf{T}}(\mathbf{a}) \mid \mathbf{a} \in I_{\text{pub}}\}$$

est un code linéaire de longueur ℓ = nombre de cellules constituant la partition $\mathcal{SSA}(C_{\text{pub}})$. Le théorème suivant est le théorème fondamental qui relie $\tilde{I}_{\text{pub}}$ au *sous-code idempotent projeté* du code caché.

Théorème 3 *Soit C_{pub} le code public d'un cryptosystème de McELIECE issu d'une clé faible. Si $\Gamma(\mathbf{L}, g)$ où $g(x) \in \mathbf{F}_2[x]$ et C_{pub} sont équivalents. Alors il existe un sous-vecteur $\mathbf{R}$ de $\mathbf{L}$ tel que $I(\mathbf{R}, g)$ et $\tilde{I}_{\text{pub}}$ sont équivalents.*

Preuve

D'après les propriétés 3, les cellules $\mathcal{P}_i$ de la partition $\mathcal{SSA}(C_{\text{pub}})$ sont

les orbites des éléments de $\mathbf{L}$ sous l'action du groupe d'automorphismes de C_{pub} . Soit π la permutation de $\mathbf{L}$ telle que

$$C_{\text{pub}} = \pi(\Gamma(\mathbf{L}, g)).$$

Selon la proposition 13, le groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$ est engendré par l'automorphisme de FROBENIUS σ . Le groupe d'automorphismes de C_{pub} est donc engendré par $\pi \circ \sigma \circ \pi^{-1}$.

$\mathbf{T} = (p_1, \dots, p_\ell)$ est un ensemble de représentant des cellules $\mathcal{P}_i$. Donc le vecteur

$$\mathbf{R} = (\pi^{-1}(p_1), \dots, \pi^{-1}(p_\ell)),$$

est un ensemble de représentants des orbites des éléments de $\mathbf{L}$ sous l'action de σ . Par conséquent, d'après la construction du *sous-code idempotent projeté* de $\Gamma(\mathbf{L}, g)$ décrite dans la section 2.1.2, on a

$$\tilde{I}_{\text{pub}} = \tilde{\pi}(I(\mathbf{R}, g)),$$

où $\tilde{\pi}$ désigne la restriction de la permutation π de $\mathbf{L}$ au sous-vecteur $\mathbf{R}$.

■

La réciproque du théorème 3 est vraie expérimentalement. Si on trouve un polynôme binaire g tel que $\tilde{I}_{\text{pub}}$ et $I(\mathbf{R}, g)$ sont équivalents, alors les codes C_{pub} et $\Gamma(\mathbf{L}, g)$ dont ils dérivent sont équivalents. Résoudre le problème 9 revient alors à résoudre le problème réduit suivant :

Problème 10 (Attaque réduite)

1. Trouver un polynôme irréductible $g(x) \in \mathbf{F}_2[x]$ tel que $I(\mathbf{R}, g)$ soit équivalent à $\tilde{I}_{\text{pub}}$.
2. Trouver la permutation π de $\mathbf{L}$, telle que

$$\Gamma(\mathbf{L}, g) = \pi(C_{\text{pub}}).$$

Pour trouver un sous-code idempotent projeté équivalent à $\tilde{I}_{\text{pub}}$, on énumère les codes de GOPPA de *polynôme générateur* binaire. Puis on teste l'équivalence entre le *sous-code idempotent projeté* et $\tilde{I}_{\text{pub}}$. La procédure à suivre se trouve décrite dans le tableau 2.8.

Complexité et implémentation de la meilleure attaque :

le nombre d'itérations de l'algorithme 2.8 est le même que le nombre d'itérations de l'algorithme 2.7. En revanche, le coût d'une itération de $\mathcal{SSA}$ est réduit. Comme on applique $\mathcal{SSA}$ à des *sous-codes idempotents projetés* dont la longueur est égale à $\ell \approx n/m$, une itération de $\mathcal{SSA}$ coûte $\lambda \ell^3$ opérations binaires, où λ est un paramètre dépendant de l'implémentation. Le coût total de l'attaque 2.8 est donc égal à $\lambda \ell^3 I_t$.

<p><i>Données :</i></p> <ul style="list-style-type: none"> – Le code $\tilde{I}_{\text{pub}}$ issu du <i>code public</i> C_{pub}, – Un sous-vecteur $\mathbf{R}$ de $\mathbf{L}$ formé des représentants des orbites de $\mathbf{L}$ sous l'action de l'automorphisme de FROBENIUS. <p><i>Résultat :</i> un polynôme $g(x) \in \mathbf{F}_2[x]$ irréductible de degré t tel que $I(\mathbf{R}, g)$ et $\tilde{I}_{\text{pub}}$ soient équivalents.</p> <p>Calculer $\mathcal{SSA}(\tilde{I}_{\text{pub}}) = \mathcal{P}$</p> <p><i>Pour tout</i> $g(x) \in \mathbf{F}_2[x]$ de degré t faire</p> <ul style="list-style-type: none"> – Déterminer le sous-code projeté $I(\mathbf{R}, g)$. – Si $\mathcal{SSA}(I(\mathbf{R}, g)) \sim \mathcal{P}$, alors <i>retourner</i> g.
--

TABLEAU 2.8 : Attaque réduite contre les clés faibles

Cette attaque peut être généralisée en utilisant les *codes étendus* dont les propriétés sont décrites dans la section 1.2.2.

On réduit alors la taille de l'espace à énumérer d'un facteur 6 environ. En effet, les transformations suivantes

$$x \mapsto x + 1, \quad x \mapsto \frac{1}{x}, \quad x \mapsto \frac{1}{x+1}, \quad x \mapsto \frac{x}{x+1}, \quad x \mapsto \frac{x+1}{x},$$

sont des homographies de $\mathbf{F}_{2^m}$. Partant si on fait agir chacune d'elle sur le *polynôme générateur binaire* d'un code de GOPPA irréductible, on obtient 6 codes de GOPPA irréductibles équivalents – voir la proposition 11. La taille des classes d'équivalence des codes de GOPPA dans la famille des *clés faibles* est donc au moins égale à 6. La description de l'attaque structurelle sur les *codes étendus*, décrite dans le tableau 2.2 montre que le nombre moyen d'itération est égal à

$$\frac{I_t}{6}.$$

- *Paramètres de MCELIECE* : le nombre de polynômes binaires, irréductibles de degré 50 sur $\mathbf{F}_{2^{10}}$ est égal à

$$I_t \approx 2^{44,3}.$$

Les sous-codes idempotents projetés sont de longueur $\ell = 108$ = nombre d'orbites de $\mathbf{F}_{2^{10}}$ sous l'action de l'automorphisme de FROBENIUS de $\mathbf{F}_{2^{10}}/\mathbf{F}_2$. La complexité de l'attaque est donc réduite par un facteur $(1024/108)^3$. Sur une station de travail Digital Alpha Processor à 500Mhz, l'attaque prendrait quelques centaines d'années.

- *Avec d'autres paramètres* : si on prend

$$n = 2^9 = 512, \quad t = 28,$$

la longueur des codes projetés est égale à $\ell = 61$. La complexité de l'attaque est réduite par un facteur $(512/61)^3$. L'attaque fut implémentée et elle prend 15 minutes sur une station de travail Digital Alpha Processor à 500Mhz.

3 Conclusion

Dans ce chapitre j'ai décrit les schémas généraux des *attaques structurelles* contre le cryptosystème de McELIECE, consistant à déterminer d'abord un code équivalent au *code public*, puis à retrouver la permutation correspondante. L'outil utilisé pour y parvenir est l'*algorithme de séparation des supports* construit par Nicolas SENDRIER [Sen00]. Grâce aux propriétés de cet algorithme, on peut reconnaître l'utilisation de *clés faibles* et considérablement réduire la complexité des *attaques structurelles*, jusqu'au point où ces attaques deviennent «presque réalistes», quand les paramètres de McELIECE sont utilisés. Dans la pratique, ces problèmes peuvent être facilement évités par le concepteur, puisque la proportion de *clés faibles* dans l'ensemble des codes de GOPPA est infime.

La démarche qui ramène l'étude de la structure d'une *clé faible*, à l'étude de la structure du *sous-code idempotent projeté* dérivé, s'inscrit dans une démarche plus générale développée dans le prochain chapitre, sous le thème de l'étude des *codes s-projetés*.

Chapitre 3

Propriétés de codes de GOPPA particuliers et application au renforcement du cryptosystème de MCELIECE

Dans le chapitre précédent, j'ai montré que les codes de GOPPA de polynôme générateur binaire constituent une famille de clés faibles des cryptosystèmes de MCELIECE et de NIEDERREITER. Leur utilisation permet d'une part de réduire le nombre de codes candidats, et d'autre part de réduire la complexité des tests en utilisant un code dérivé du *code public* de plus petite longueur et de plus petite dimension. En réalité, cette propriété est bien plus générale. En effet, dès que le *polynôme générateur* d'un code de GOPPA est à coefficients dans un sous-corps strict du *corps générateur*, alors le groupe d'automorphismes du code n'est pas trivial et contient le groupe engendré par l'*automorphisme de FROBENIUS* de l'extension. Ces codes constituent donc des familles de clés faibles pour les cryptosystèmes de MCELIECE et de NIEDERREITER. Cependant, lorsque les polynômes générateurs ne sont pas binaires, le nombre de tels codes est très élevé et, malgré les réductions de complexité considérées précédemment, cela ne permet pas de construire d'*attaques structurelles* viables. En revanche, on peut utiliser leurs propriétés pour renforcer les cryptosystèmes contre les *attaques par décodage*. On élabore un compromis entre la résistance des cryptosystèmes de MCELIECE et de NIEDERREITER aux *attaques structurelles* et leur résistance aux *attaques par décodage*.

Dans ce chapitre, j'étudie dans un premier temps les propriétés des codes de GOPPA dont le *polynôme générateur* est à coefficients dans un sous-corps du *corps générateur*. Je montre qu'on peut en dériver des codes appelés *s-projetés*, qui constituent une généralisation du sous-code idempotent projeté introduit dans la section 2.1.2 du chapitre 2, et j'étudie leurs propriétés. Dans une seconde partie, je montre qu'on peut utiliser les propriétés de non

trivialité du *groupe d'automorphismes*, afin de construire des ensembles *t-décodables par tour* qui permettent de décoder des mots de poids supérieur à la capacité de correction des codes. Je montre enfin comment utiliser ces propriétés pour renforcer le cryptosystème de McELIECE contre les *attaques par décodage*. Ce chapitre met en forme et en français, les articles [Loi, Loi00c].

1 Codes dérivés de codes de GOPPA binaires

Dans cette section, je considère des codes de GOPPA binaires $\Gamma(\mathbf{L}, g)$ dont le *polynôme générateur* g est à coefficients dans un sous corps $\mathbf{F}_{2^s}$ du *corps générateur* $\mathbf{F}_{2^m}$. Je montre que le groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$ contient le groupe engendré par l'*automorphisme de FROBENIUS* de l'extension $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$. Le sous-code des mots de $\Gamma(\mathbf{L}, g)$ qui sont invariants sous l'action de l'automorphisme de FROBENIUS peut-être projeté d'une certaine façon, constituant ainsi ce que j'appelle un *code s-projeté*. Dans une seconde partie, je montre qu'on peut dériver toute une série de propriétés des *codes s-projetés* à partir de leur construction.

1.1 Construction des codes s-projetés

Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA binaire. Lorsque le *polynôme générateur* g du code est à coefficients dans un sous-corps $\mathbf{F}_{2^s}$ de $\mathbf{F}_{2^m}$, alors le groupe d'automorphismes du code contient le groupe engendré par l'automorphisme de FROBENIUS

$$\sigma : x \mapsto x^{2^s}$$

de l'extension $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$.

1.1.1 Groupe d'automorphismes des codes de GOPPA

Soit $\mathbf{a} = (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n})$ un mot de $\mathbf{F}_{q^m}$ de *vecteur générateur*

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n).$$

Soit π une permutation des éléments de $\mathbf{L}$, alors on définit l'action de π sur le mot $\mathbf{a}$ par

$$\pi(\mathbf{a}) = (a_{\pi^{-1}(\alpha_1)}, a_{\pi^{-1}(\alpha_2)}, \dots, a_{\pi^{-1}(\alpha_n)})$$

avec ces notations, on définit l'ordre du mot $\mathbf{a}$

Définition 17 (Ordre d'un mot et invariant) *On dit qu'un mot $\mathbf{a}$ est d'ordre $k \geq 1$ sous l'action de π si*

1. $\pi^j(\mathbf{a}) \neq \mathbf{a}$, pour $j = 1, \dots, k - 1$,
2. $\pi^k(\mathbf{a}) = \mathbf{a}$.

Si $\pi(\mathbf{a}) = \mathbf{a}$, alors on dit que $\mathbf{a}$ est invariant sous l'action de π .

Proposition 16 (Groupe d'automorphismes) Soit σ l'automorphisme de FROBENIUS de l'extension $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$. Soit $g \in \mathbf{F}_{2^s}[x]$, et soit $\mathbf{L}$ un étiquetage des éléments de $\mathbf{F}_{q^m}$ qui ne sont pas racines de g . Alors le groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$ contient le groupe $\langle \sigma \rangle$ engendré par σ , i.e.

$$\forall \mathbf{a} \in \Gamma(\mathbf{L}, g), \quad \sigma(\mathbf{a}) \in \Gamma(\mathbf{L}, g).$$

Preuve

Le mot $\mathbf{a} = (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n})$ appartient au code de GOPPA $\Gamma(\mathbf{L}, g)$ si et seulement si ses coordonnées a_{α_i} vérifient

$$\sum_{i=1}^n a_{\alpha_i} \frac{\alpha_i^j}{g(\alpha_i)} = 0,$$

pour $j = 0, \dots, t-1$. L'action de σ sur le vecteur $\mathbf{a}$ s'écrit

$$\sigma(\mathbf{a}) = (a_{\beta_1}, a_{\beta_2}, \dots, a_{\beta_n})$$

où $\beta_i^{2^s} = \alpha_i$. Comme par hypothèse le polynôme g est à coefficients dans $\mathbf{F}_{2^s}$, on a $g(\beta_i^{2^s}) = g(\beta_i)^{2^s}$. De plus, comme $a_{\alpha_i} \in \mathbf{F}_2$, $a_{\alpha_i}^{2^s} = a_{\alpha_i}$. Donc

$$\sum_{i=1}^n a_{\beta_i} \frac{\alpha_i^j}{g(\alpha_i)} = \sum_{i=1}^n a_{\beta_i} \frac{(\beta_i^j)^{2^s}}{g(\alpha_i)^{2^s}} = \left(\sum_{i=1}^n a_{\beta_i} \frac{\beta_i^j}{g(\beta_i)} \right)^{2^s} = 0$$

pour $j = 0, \dots, t-1$. Donc $\sigma(\mathbf{a}) \in \Gamma(\mathbf{L}, g)$.

■

Comme le groupe $\langle \sigma \rangle$ est d'ordre m/s , la proposition 16 implique que, chaque fois que le polynôme générateur g d'un code de GOPPA est à coefficients dans un sous-corps strict $\mathbf{F}_{2^s}$ de $\mathbf{F}_{2^m}$, alors le groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$ n'est pas trivial et est d'ordre au moins égal à m/s . Expérimentalement, le groupe d'automorphismes de presque tous les codes de GOPPA dont le polynôme générateur est à coefficients dans $\mathbf{F}_{2^s}$ est égal au groupe engendré par l'automorphisme de FROBENIUS de $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$, i.e.

$$\text{Aut}(\Gamma(\mathbf{L}, g)) = \langle \sigma \rangle,$$

et donc est d'ordre exactement m/s . Par commodité, je ne considère que ce cas dans la suite.

1.1.2 Sous-code des mots invariants

Soit $g(x) \in \mathbf{F}_{2^s}[x]$ et soit

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n),$$

un étiquetage des éléments de $\mathbf{F}_{2^m}$ qui ne sont pas racines de g . Comme g est à coefficients dans $\mathbf{F}_{2^s}$, un sous corps de $\mathbf{F}_{2^m}$, l'ensemble des racines de

g dans $\mathbf{F}_{2^m}$ est invariant sous l'action de l'automorphisme de FROBENIUS σ . Partant, le vecteur générateur $\mathbf{L}$ s'écrit formellement

$$\mathbf{L} = \cup_{i=1}^N \mathcal{O}_i,$$

où les $\mathcal{O}_i$ sont les orbites des éléments de $\mathbf{F}_{2^m}$, qui ne sont pas racines de g , sous l'action de $\langle \sigma \rangle$. Soit

$$I(\mathbf{L}, g) = \{ \mathbf{a} \in \Gamma(\mathbf{L}, g) \mid \sigma(\mathbf{a}) = \mathbf{a} \}$$

l'ensemble des mots du code de GOPPA invariants sous l'action de σ . De cette définition, on déduit immédiatement les propriétés suivantes :

Propriétés 4

1. $I(\mathbf{L}, g)$ est un sous-code de $\Gamma(\mathbf{L}, g)$,
2. le sous-ensemble de $\mathbf{L}$ étiquetant les positions non nulles d'un mot $\mathbf{a} \in I(\mathbf{L}, g)$ est une réunion d'un certain nombre d'orbites $\mathcal{O}_i$.

Pour ces raisons, le sous-code $I(\mathbf{L}, g)$ est appelé *sous-code invariant* du code $\Gamma(\mathbf{L}, g)$.

À partir de là, on construit les *codes s-projetés*. Soit $g(x) \in \mathbf{F}_{2^s}[x]$, et soit $\mathbf{L}$ un étiquetage des éléments de $\mathbf{F}_{2^m}$ qui ne sont pas racines de g . Le processus de construction est le suivant :

1. On détermine le *sous-code invariant* $I(\mathbf{L}, g)$ de $\Gamma(\mathbf{L}, g)$.
2. *Construction du code s-projeté* : soit $\mathcal{O}_1, \mathcal{O}_2, \dots, \mathcal{O}_N$, les orbites des éléments de $\mathbf{L}$ sous l'action du groupe $\langle \sigma \rangle$, *i.e.*

$$\mathbf{L} = \cup_{i=1}^N \mathcal{O}_i.$$

Dans chaque orbite $\mathcal{O}_i$, on choisit un représentant o_i , *i.e.*

$$\mathcal{O}_i = \{ \sigma^j(o_i) \mid j = 1, \dots, m/s \}.$$

Soit alors le vecteur

$$\mathbf{R} = (o_1, o_2, \dots, o_N),$$

et soit $\mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n})$, un mot du *sous-code invariant* $I(\mathbf{L}, g)$. On projette le mot $\mathbf{a}$ de vecteur générateur $\mathbf{L}$ en un mot $\tilde{\mathbf{a}}$ de vecteur générateur $\mathbf{R}$ en prenant pour chaque orbite $\mathcal{O}_i$ la coordonnée a_{o_i} .

$$\begin{aligned} I(\mathbf{L}, g) &\rightarrow \tilde{I}(\mathbf{L}, g) \\ \mathbf{a} = (a_{\alpha_1}, \dots, a_{\alpha_n}) &\mapsto \tilde{\mathbf{a}} = (a_{o_1}, \dots, a_{o_N}) \end{aligned}$$

Construit de la sorte, l'ensemble $\tilde{I}(\mathbf{R}, g)$ des projetés des mot du *sous-code invariant* $I(\mathbf{L}, g)$ est un code linéaire de longueur $N \leq n$, et de même dimension k_I que $I(\mathbf{L}, g)$.

Tout mot $\mathbf{a} = (a_{o_1}, \dots, a_{o_N})$ de $\tilde{I}(\mathbf{R}, g)$ peut être relevé de manière unique en un mot $\bar{\mathbf{a}} \in I(\mathbf{L}, g)$, en répétant chaque coordonnée a_{o_i} exactement $|\mathcal{O}_i|$ fois. Les codes $\tilde{I}(\mathbf{R}, g)$ et $I(\mathbf{L}, g) \subset \Gamma(\mathbf{L}, g)$ sont donc en bijection.

Définition 18 *Le code $\tilde{I}(\mathbf{R}, g)$ ainsi construit est appelé code s -projeté dérivé de $\Gamma(\mathbf{L}, g)$, de vecteur générateur $\mathbf{R}$.*

Par construction, $\tilde{I}(\mathbf{R}, g)$ est indépendant du vecteur $\mathbf{R}$ choisi. En effet, si on prend un ensemble de représentants des orbites

$$\mathbf{R}' = (o'_1, o'_2, \dots, o'_N),$$

où o'_i et o_i sont dans la même orbite $\mathcal{O}_i$, il est facile de voir que

$$\tilde{I}(\mathbf{R}', g) = \tilde{I}(\mathbf{R}, g).$$

Remarque 2 *Le lecteur perspicace aura sans doute remarqué que les codes 1-projetés définis ici quand g est binaire, correspondent aux sous-codes idempotents projetés de codes de GOPPA, introduits dans la section 2.1.2 du chapitre 2.*

Exemple 11

Considérons le polynôme $g(x) = x^3 + x + 1$ à coefficients dans le corps $\mathbf{F}_2$. Comme g est sans racines dans $\mathbf{F}_{2^4}$, on prend comme *vecteur générateur* $\mathbf{L}$ un étiquetage de $\mathbf{F}_{2^4}$. Soit α un élément primitif de $\mathbf{F}_{2^4}$ et soit

$$\mathbf{L} = \left(\underbrace{0}_{\mathcal{O}_1}, \underbrace{1}_{\mathcal{O}_2}, \underbrace{\alpha, \alpha^2, \alpha^4, \alpha^8}_{\mathcal{O}_3}, \underbrace{\alpha^{12}, \alpha^3, \alpha^6, \alpha^9}_{\mathcal{O}_4}, \underbrace{\alpha^5, \alpha^{10}}_{\mathcal{O}_5}, \underbrace{\alpha^{11}, \alpha^{13}, \alpha^{14}, \alpha^7}_{\mathcal{O}_6} \right).$$

Une matrice génératrice du code $\Gamma(\mathbf{L}, g)$ s'écrit :

$$\mathbf{G}_\Gamma = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{array} \right).$$

Les séparations $|$ entre les différentes colonnes correspondent à des positions du vecteur générateur $\mathbf{L}$ étiquetées par différentes orbites. Il y a deux orbites de taille 1 – $\mathcal{O}_1$, et $\mathcal{O}_2$ –, une orbite de taille 2 – $\mathcal{O}_3$ – et trois orbites de taille 4 – $\mathcal{O}_4$, $\mathcal{O}_5$, et $\mathcal{O}_6$. Une matrice génératrice du *sous-code invariant* $I(\mathbf{L}, g)$ a la forme suivante :

$$\mathbf{G}_I = \left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right).$$

Prenons $\mathbf{R} = (0, 1, \alpha, \alpha^{12}, \alpha^5, \alpha^{11})$, un ensemble de représentants de chaque orbite $\mathcal{O}_i$. Alors la matrice génératrice sous forme échelonnée réduite du code 1-projeté $\tilde{I}(\mathbf{R}, g)$ s'obtient en prenant les colonnes de la matrice $\mathbf{G}_I$ étiquetées par les éléments de $\mathbf{R}$, soit

$$\mathbf{R} = (0 \quad 1 \quad \alpha \quad \alpha^{12} \quad \alpha^5 \quad \alpha^{11})$$

$$\mathbf{G}_{\tilde{I}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

1.2 Propriétés des codes s -projetés

Je montre que les *codes s -projetés* sont les *sous-codes traces* sur $\mathbf{F}_2$ de codes à coefficients dans $\mathbf{F}_{2^s}$, pour lesquels on connaît une matrice de parité. J'en déduis une forme explicite des équations de parité des *codes s -projetés*, ainsi que les propriétés suivantes :

Propriétés 5 Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA, de longueur n , de dimension k et de distance minimale d , où g est de degré t , à coefficients dans $\mathbf{F}_{2^s} \subset \mathbf{F}_{2^m}$. Soit $\tilde{\Gamma}(\mathbf{R}, g)$ le code s -projeté correspondant de longueur $N = |\mathbf{R}|$, de dimension k_I et de distance minimale d_I . Alors, on a les propriétés suivantes :

1. $k_I \geq N - st$,
2. si le polynôme générateur g est sans facteurs carrés, et sans racines dans $\mathbf{F}_{2^m}$, alors

$$\text{Si } t < 2^{m/2} - 1 \text{ alors } k_I = N - st,$$

3. si le degré m/s de l'extension est une puissance d'un nombre premier p , alors la dimension k du code de GOPPA et la dimension k_I du code s -projeté sont congrues modulo $p - 1$, i.e.

$$k \equiv k_I \pmod{p - 1},$$

4. la distance minimale d_I du code s -projeté vérifie

$$d_I \geq d \frac{s}{m},$$

5. un algorithme de décodage des codes de GOPPA permet de décoder

$$\deg(\bar{g}) d \frac{s}{m}$$

erreurs dans le code s -projeté, où $\bar{g}$ désigne le plus petit polynôme sans facteurs carrés, divisant g ,

Le lecteur aura remarqué que ces propriétés sont similaires à celles que l'on peut dériver pour les codes de GOPPA – voir [MS77, p. 339].

1.2.1 Dimension des codes s -projetés

Je commence par traiter des propriétés concernant la dimension des *codes s -projetés*. Je montre qu'un code s -projeté est le *sous-code trace* sur $\mathbf{F}_2$ d'un code sur $\mathbf{F}_{2^s}$ dont on dérive une matrice de parité à partir de la matrice de parité du code de GOPPA correspondant.

Soit $\Gamma(\mathbf{L}, g)$ le code de GOPPA, de polynôme générateur $g(x) \in \mathbf{F}_{2^s}[x]$, où $\mathbf{F}_{2^s}$ est un sous-corps de $\mathbf{F}_{2^m}$, et de degré t . Soit $\tilde{\Gamma}(\mathbf{R}, g)$ le code s -projeté dérivé, où $\mathbf{R} = (o_1, o_2, \dots, o_N)$. Considérons la matrice

$$\mathbf{H}_I = \begin{pmatrix} \text{Tr}_1 \left(\frac{1}{g(o_1)} \right) & \cdots & \text{Tr}_N \left(\frac{1}{g(o_N)} \right) \\ \vdots & \ddots & \vdots \\ \text{Tr}_1 \left(\frac{o_1^{t-1}}{g(o_1)} \right) & \cdots & \text{Tr}_N \left(\frac{o_N^{t-1}}{g(o_N)} \right) \end{pmatrix},$$

où $\mathbf{F}_{2^s}(o_i)$ est le plus petit corps contenant $\mathbf{F}_{2^s}$ et o_i et où

$$\text{Tr}_i = \text{Tr}_{\mathbf{F}_{2^s}(o_i)/\mathbf{F}_{2^s}}$$

désigne l'opérateur trace de l'extension $\mathbf{F}_{2^s}(o_i)/\mathbf{F}_{2^s}$, i.e. si $\mathcal{O}_i$ désigne l'orbite de l'élément o_i sous l'action de l'automorphisme de FROBENIUS de $\mathbf{F}_{2^s}(o_i)/\mathbf{F}_{2^s}$, alors

$$\text{Tr}_i(x) = \sum_{j=1}^{|\mathcal{O}_i|} x^{2^{sj}}.$$

$\mathbf{H}_I$ est donc une matrice à coefficients dans $\mathbf{F}_{2^s}$, et

Théorème 4 *Le code s -projeté $\tilde{I}(\mathbf{R}, g)$ est l'ensemble*

$$\tilde{I}(\mathbf{R}, g) = \{ \tilde{\mathbf{a}} = (a_{o_1}, \dots, a_{o_N}) \in \mathbf{F}_2^N \mid \mathbf{H}_I \tilde{\mathbf{a}}^T = 0 \}.$$

$\tilde{I}(\mathbf{R}, g)$ est donc le sous-code trace du code de matrice de parité $\mathbf{H}_I$.

Preuve

Soit $\tilde{\mathbf{a}} = (a_{o_1}, \dots, a_{o_N}) \in \tilde{I}(\mathbf{R}, g)$. Par construction de $\tilde{I}(\mathbf{R}, g)$, le mot $\tilde{\mathbf{a}}$ peut être relevé en un mot

$$\mathbf{a} = (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n}) \in \Gamma(\mathbf{L}, g),$$

en dupliquant chaque coordonnée a_{o_i} de $\tilde{\mathbf{a}}$, un nombre de fois égal au cardinal de l'orbite de l'élément o_i . Le mot $\mathbf{a}$ vérifie donc les équations de parité du code de GOPPA $\Gamma(\mathbf{L}, g)$

$$\sum_{j=1}^n a_{\alpha_j} \frac{\alpha_j^i}{g(\alpha_j)} = 0 \quad (3.1)$$

pour $i = 0, \dots, t-1$. Comme les coordonnées de $\mathbf{a}$ étiquetées par la même orbite sont identiques, si le vecteur générateur $\mathbf{L}$ s'écrit

$$\mathbf{L} = \cup_{j=1}^N \mathcal{O}_j,$$

où $\mathcal{O}_j$ est l'orbite de l'élément o_j , alors l'équation (3.1) se réécrit

$$\sum_{j=1}^N a_{o_j} \sum_{\beta \in \mathcal{O}_j} \frac{\beta^i}{g(\beta)} = 0 \quad (3.2)$$

pour $i = 0, \dots, t-1$. Comme g est à coefficients dans le sous-corps $\mathbf{F}_{2^s}$ de $\mathbf{F}_{2^m}$, g et l'automorphisme de FROBENIUS de $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$ commutent. Or, par définition de Tr_i , on a

$$\sum_{\beta \in \mathcal{O}_j} \frac{\beta^i}{g(\beta)} = \text{Tr}_i \left(\frac{o_j^i}{g(o_j)} \right).$$

En remplaçant cette expression dans l'équation (3.2), il s'ensuit

$$\sum_{j=1}^N a_{o_j} \text{Tr}_i \left(\frac{o_j^i}{g(o_j)} \right) = 0$$

pour $i = 0, \dots, t-1$. D'où $\mathbf{H}_I \tilde{\mathbf{a}}^T = 0$.

■

Le théorème 4 fournit une matrice de parité du *code s-projeté* $\tilde{I}(\mathbf{R}, g)$. En effet, étant donnée une base quelconque de $\mathbf{F}_{2^s}/\mathbf{F}_2$, la matrice obtenue en développant chaque ligne de la matrice $\mathbf{H}_I$ sur cette base est une matrice $st \times n$ à coefficients dans $\mathbf{F}_2$. Celle-ci est une matrice de parité de $\tilde{I}(\mathbf{R}, g)$. J'en déduis le corollaire suivant qui donne une borne inférieure sur la dimension des *codes s-projetés*.

Corollaire 5 (Minoration de la dimension)

Soit un code de GOPPA $\Gamma(\mathbf{L}, g)$, de polynôme générateur g à coefficients dans le sous-corps $\mathbf{F}_{2^s}$ de $\mathbf{F}_{2^m}$, et de degré t . Soit $\tilde{I}(\mathbf{R}, g)$ le code *s-projeté dérivé*, de longueur $N = |\mathbf{R}|$ et de dimension k_I , alors

$$k_I \geq N - st.$$

Voici un corollaire du théorème 4, reliant une première fois la dimension d'un code de GOPPA, à la dimension du *code s-projeté dérivé* :

Corollaire 6 Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA de polynôme générateur g à coefficients dans le sous-corps $\mathbf{F}_{2^s}$ de $\mathbf{F}_{2^m}$, et de degré t . Si $n - mt > 0$, alors le code *s-projeté dérivé* $\tilde{I}(\mathbf{R}, g)$ n'est pas trivial.

Preuve

Comme le groupe engendré par l'automorphisme de FROBENIUS de $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$ est d'ordre m/s , les orbites $\mathcal{O}_j$ qui constituent le *vecteur générateur* $\mathbf{L}$ sont de taille au plus m/s . De plus, la longueur n de $\Gamma(\mathbf{L}, g)$ vérifie $n = |\mathbf{L}| = \sum_{j=1}^N |\mathcal{O}_j|$. Donc la longueur N du *code s-projeté* $\tilde{I}(\mathbf{R}, g)$ vérifie

$$N \geq n \frac{s}{m}.$$

Donc

$$N - st \geq n \frac{s}{m} - st = \frac{s}{m}(n - mt) > 0$$

et comme, d'après le corollaire 5, la dimension k_I de $\tilde{I}(\mathbf{R}, g)$ vérifie $k_I \geq N - st$, on a $k_I > 0$.

■

Le corollaire 5 permet d'affiner la borne inférieure sur la dimension d'un code de GOPPA. Le lecteur se rappelle – voir la section 2.2 du chapitre 1 – que la dimension k d'un code de GOPPA de longueur n dont le *polynôme générateur* est de degré t , à coefficients dans $\mathbf{F}_{2^m}$ vérifie

$$k \geq n - mt.$$

Supposons désormais que les coefficients de g soient dans un sous-corps $\mathbf{F}_{2^s}$ de $\mathbf{F}_{2^m}$, et soit k_I la dimension du code *s-projeté dérivé*. Comme ce code est en bijection avec le *sous-code invariant* du code de GOPPA, on a $k_I \leq k$. De plus, d'après le corollaire 5, on a $k_I \geq N - st$. Donc chaque fois que $N - st > n - mt$, alors $k > N - st$. Par conséquent, si le degré t de g vérifie

$$\frac{N}{s} > t > \frac{n - N}{m - s},$$

alors la borne inférieure sur la dimension est affinée.

Exemple 12

Considérons le vecteur $\mathbf{L}$ comme un étiquetage du corps à $2^{10} = 1024$ éléments. Alors, $m = 10$ et la borne minimale

$$k \geq n - mt$$

sur la dimension des codes de GOPPA nous assure que les codes sont non triviaux pour un degré du polynôme générateur t sans facteur carré vérifiant

$$t \leq 102.$$

Pour les différentes valeurs de s divisant $m = 10$ ($s = 1, 2, 5$), la borne inférieure $N - st$ est meilleure que la borne inférieure $n - mt$ pour les valeurs suivantes de t :

- Pour $s = 1$, il y a $N = 108$ orbites dans $\mathbf{F}_{2^{10}}/\mathbf{F}_2$ et donc si $108 > t \geq 102$, alors la nouvelle borne est meilleure.
- Pour $s = 2$, il y a $N = 208$ orbites dans $\mathbf{F}_{2^{10}}/\mathbf{F}_{2^2}$ et donc si $t = 103$, alors la nouvelle borne est meilleure.
- Pour $s = 5$, il y a $N = 528$ orbites dans $\mathbf{F}_{2^{10}}/\mathbf{F}_{2^5}$ et donc si $105 > t \geq 100$, alors la nouvelle borne est meilleure.

Dans l'article [vdV90], VAN DER VLUGT a montré que, lorsque le degré du *polynôme générateur* d'un code de GOPPA était suffisamment petit, alors la borne sur la dimension $k \geq n - mt$ était atteinte. Je montre à présent comment étendre ce résultat à la famille des *codes s-projetés*.

Théorème 5 (Dimension des codes s-projetés)

Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA dont $g(x) \in \mathbf{F}_{2^s}[x]$ est de degré t , sans facteurs carrés, et sans racines dans $\mathbf{F}_{2^m}$. Soit $\tilde{I}(\mathbf{R}, g)$, le code *s-projeté dérivé* de longueur N et de dimension k_I .

Si $t < 2^{m/2-1}$, alors $k_I = N - st$.

Afin de prouver le théorème j'ai recours aux deux lemmes suivants

Lemme 1 Soit $\tilde{I}(\mathbf{R}, g)^\perp$, le dual du code $\tilde{I}(\mathbf{R}, g)$, alors

$$\tilde{I}(\mathbf{R}, g)^\perp = \left\{ \left(\text{Tr}^{(1)} \left[\frac{f(o_1)}{g(o_1)} \right], \dots, \text{Tr}^{(N)} \left[\frac{f(o_N)}{g(o_N)} \right] \right)_{j=1}^N \mid \begin{array}{l} f(x) \in \mathbf{F}_{2^m}[x], \\ \deg(f) \leq t-1 \end{array} \right\}$$

où $\text{Tr}^{(i)}$ désigne l'opérateur trace de $\mathbf{F}_{2^s}(o_i)/\mathbf{F}_2$.

Preuve du lemme

D'après la proposition 4, le code $\tilde{I}(\mathbf{R}, g)$ où $\mathbf{R} = (o_1, \dots, o_N)$ est le sous-code trace du code de matrice de parité $\mathbf{H}_I$, où

$$\mathbf{H}_I = \begin{pmatrix} \text{Tr}_1 \left(\frac{1}{g(o_1)} \right) & \cdots & \text{Tr}_N \left(\frac{1}{g(o_N)} \right) \\ \vdots & \ddots & \vdots \\ \text{Tr}_1 \left(\frac{o_1^{t-1}}{g(o_1)} \right) & \cdots & \text{Tr}_N \left(\frac{o_N^{t-1}}{g(o_N)} \right) \end{pmatrix},$$

Tr_i désignant l'opérateur trace de $\mathbf{F}_{2^s}(o_i)/\mathbf{F}_{2^s}$. DELSARTE a démontré dans l'article [Del75b] que, pour tout code linéaire C sur une extension F d'un corps F_0 , alors, le dual du sous-code trace de C sur F_0 , noté $(C | F_0)^\perp$ vérifie

$$(C | F_0)^\perp = \text{Tr}_{F/F_0}(C).$$

En remplaçant le corps F par $\mathbf{F}_{2^s}$ et le corps F_0 par $\mathbf{F}_2$, il s'ensuit que

$$\tilde{I}(\mathbf{R}, g)^\perp = \left\{ \left(\text{Tr}_{\mathbf{F}_{2^s}/\mathbf{F}_2} \left[\text{Tr}_i \left(\frac{f(o_i)}{g(o_i)} \right) \right] \right)_{j=1}^N \mid \begin{array}{l} f(x) \in \mathbf{F}_{2^s}[x], \\ \deg(f) \leq t-1 \end{array} \right\}.$$

La preuve du lemme découle ensuite directement de la propriété suivante de l'opérateur trace, que le lecteur pourra trouver dans [LN97, p. 55] :

$$\underbrace{\text{Tr}_{\mathbf{F}_{2^s}(o_i)/\mathbf{F}_2}(z)}_{\text{Tr}^{(i)}(z)} = \text{Tr}_{\mathbf{F}_{2^s}/\mathbf{F}_2}(\text{Tr}_i(z)).$$

■

Lemme 2 Soit $g(x) \in \mathbf{F}_{2^m}[x]$ un polynôme de degré t sans racines dans $\mathbf{F}_{2^m}$, et soit $f(x) \in \mathbf{F}_{2^m}[x]$ un polynôme non nul de degré inférieur ou égal à $t-1$. Si

$$\left| \sum_{\alpha \in \mathbf{F}_{2^m}} (-1)^{\text{Tr}_{\mathbf{F}_{2^m}/\mathbf{F}_2} \left(\frac{f(\alpha)}{g(\alpha)} \right)} \right| > (2t-2)2^{m/2},$$

alors l'équation

$$Y^2 + Y = \frac{f}{g}$$

a une solution dans le corps $\overline{\mathbf{F}_2}(x)$, où $\overline{\mathbf{F}_2}(x)$ désigne le corps des fractions de $\overline{\mathbf{F}_2}$ – la clôture algébrique de $\mathbf{F}_2$.

La preuve de ce lemme se trouve dans l'article de VAN DER VLUGT [vdV90] et reprend un résultat de BOMBIERI [Bom66].

Preuve du théorème 5

Je vais montrer que, sous les hypothèses du théorème, tout polynôme non nul $f(x) \in \mathbf{F}_{2^s}[x]$ de degré inférieur ou égal à $t - 1$ correspond à un mot non nul de $\tilde{I}(\mathbf{R}, g)^\perp$, suivant la construction donnée dans le lemme 1. Il s'ensuit alors que deux polynômes distincts de $\mathbf{F}_{2^s}[x]$ de degré inférieur ou égal à $t - 1$ donnent deux mots distincts, toujours par la construction de $\tilde{I}(\mathbf{R}, g)^\perp$ donnée par le lemme 1. Donc $\tilde{I}(\mathbf{R}, g)^\perp$ est de dimension supérieure à st . Comme, d'après le corollaire 5, la dimension de $\tilde{I}(\mathbf{R}, g)^\perp$ est inférieure ou égale à st , on en déduit que sa dimension est égale à st . Donc la dimension de $\tilde{I}(\mathbf{R}, g)$ est égale à $N - st$.

Je raisonne par contradiction. Supposons qu'il existe $f(x) \in \mathbf{F}_{2^s}[x]$ non nul de degré inférieur ou égal à $t - 1$, correspondant au mot nul du code $\tilde{I}(\mathbf{R}, g)$, i.e.

$$\mathrm{Tr}^{(j)} \left(\frac{f(o_j)}{g(o_j)} \right) = 0$$

pour $j = 1, \dots, N$, où $\mathrm{Tr}^{(j)}$ désigne l'opérateur trace de $\mathbf{F}_{2^s}(o_j)/\mathbf{F}_2$. Comme $\mathrm{Tr}_{\mathbf{F}_{2^m}/\mathbf{F}_2}(z) = \mathrm{Tr}_{\mathbf{F}_{2^m}/\mathbf{F}_{2^s}(o_j)}(1) \mathrm{Tr}^{(j)}(z)$, on a également

$$\mathrm{Tr}_{\mathbf{F}_{2^m}/\mathbf{F}_2} \left(\frac{f(o_j)}{g(o_j)} \right) = 0$$

pour $j = 1, \dots, N$. Comme les polynômes f et g sont à coefficients dans le corps $\mathbf{F}_{2^s}$, ils commutent avec l'automorphisme de FROBENIUS de $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$, $\sigma : x \mapsto x^{2^s}$. De plus, l'opérateur trace commute également avec σ . Comme g est sans racines dans $\mathbf{F}_{2^m}$, le vecteur générateur de $\Gamma(\mathbf{L}, g)$ étiquette le corps $\mathbf{F}_{2^m}$ tout entier. Donc les o_j décrivent toutes les orbites de $\mathbf{F}_{2^m}$. Partant, pour tout élément $\alpha \in \mathbf{F}_{2^m}$, on a

$$\mathrm{Tr}_{\mathbf{F}_{2^m}/\mathbf{F}_2} \left(\frac{f(\alpha)}{g(\alpha)} \right) = 0.$$

Il s'ensuit que

$$\left| \sum_{\alpha \in \mathbf{F}_{2^m}} (-1)^{\mathrm{Tr}_{\mathbf{F}_{2^m}/\mathbf{F}_2} \left(\frac{f(\alpha)}{g(\alpha)} \right)} \right| = 2^m.$$

Comme, d'après les hypothèses du théorème, le degré t de g vérifie $t < 2^{m/2} - 1$, alors $2^m > (2t - 2)2^{m/2}$, et

$$\left| \sum_{\alpha \in \mathbf{F}_{2^m}} (-1)^{\mathrm{Tr}_{\mathbf{F}_{2^m}/\mathbf{F}_2} \left(\frac{f(\alpha)}{g(\alpha)} \right)} \right| > (2t - 2)2^{m/2}.$$

En appliquant les résultats du lemme 2, on déduit qu'il existe une fraction rationnelle $h(x) = u(x)/v(x)$ à coefficients dans $\overline{\mathbf{F}_2}$ telle que

$$h^2 + h = \frac{f}{g},$$

avec u et v premiers entre eux. Considérons alors les polynômes

$$G(x) = \frac{g(x)}{d(x)}, \quad F(x) = \frac{f(x)}{d(x)}$$

où $d(x)$ est le *pgcd* de f et de g . F et G vérifient alors l'équation suivante

$$G(v^2 + vu) = Fu^2.$$

Comme on a construit F et G de telle sorte qu'ils soient premiers entre eux, G divise u^2 . De plus, par hypothèse, le polynôme g est sans facteurs carrés, donc G est sans facteurs carrés. Donc G divise u . Soit $U = u/G$. U est un polynôme qui vérifie l'équation

$$v^2 + vUG = FU^2G.$$

Donc G divise v^2 et partant, G divise v . Comme u et v sont premiers entre eux par construction $G = 1$. Donc $g(x) = d(x)$, donc g divise f . Il s'ensuit que le degré de f est nécessairement supérieur ou égal à t , ce qui contredit l'hypothèse donc f est le polynôme nul. Ce qui prouve le théorème.

■

En fait, l'inégalité donnée par le théorème 5 n'est pas optimale. Par exemple si on prend $n = 1024$, c'est-à-dire $m = 10$, le théorème donne $k_I = N - st$ pour $t < 16$. Cependant, les simulations montrent que cette égalité est satisfaite pour des valeurs de t bien supérieures – pour $t > 50$ notamment.

Dans cette dernière partie, je démontre que, dans certains cas, il existe une relation explicite entre la dimension d'un code de GOPPA et la dimension du *code s-projeté* dérivé, et ce, quel que soit le degré du polynôme générateur.

Proposition 17 *Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA de dimension k , dont le vecteur générateur g est à coefficients dans $\mathbf{F}_{2^s}$, un sous-corps de $\mathbf{F}_{2^m}$. Soit $\tilde{I}(\mathbf{R}, g)$ le code *s-projeté dérivé* de dimension k_I . Si $\frac{m}{s} = p^r$ où p est un nombre premier et r un entier positif quelconque, alors*

$$k_I \equiv k \pmod{p-1}.$$

Preuve

Soit $A_\Gamma = 2^k$, le nombre de mots dans le code $\Gamma(\mathbf{L}, g)$. Pour tout i divisant $\frac{m}{s} = p^r$, soit A_i le nombre de mots d'ordre i sous l'action du groupe engendré par l'*automorphisme* de FROBENIUS de $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$. On a

1. $A_\Gamma = A_1 + \sum_{i/p^r, i>1} A_i$,
 2. $A_i \equiv 0 \pmod{i}$,
 3. $A_1 = 2^{k_I}$, par construction du *code s-projeté* $\tilde{I}(\mathbf{R}, g)$.
- Comme $\frac{m}{s} = p^r$ par hypothèse, les diviseurs de $\frac{m}{s}$ sont les puissances $i = p^j$ où $j = 1, \dots, r$. Donc $A_i \equiv 0 \pmod{p}$, d'où

$$A_\Gamma \equiv A_1 \pmod{p},$$

et $2^k \equiv 2^{k_I} \pmod{p}$. Comme p est un nombre premier, en appliquant le *petit théorème de FERMAT*, on obtient

$$k_I \equiv k \pmod{p-1}.$$

■

1.2.2 Distance minimale et algorithme de décodage

Une borne inférieure sur la distance minimale des *codes s-projetés* découle de la manière dont ils furent construits à la section 1.1.2.

Proposition 18 (Distance minimale) *Soit $\tilde{I}(\mathbf{R}, g)$ le code s-projeté de distance minimale d_I dérivé du code de GOPPA $\Gamma(\mathbf{L}, g)$, de corps générateur $\mathbf{F}_{2^m}$, et de distance minimale d . Alors on a*

$$d_I \geq d \frac{s}{m}.$$

Preuve

Soit $\mathbf{a}$ un mot de $\tilde{I}(\mathbf{R}, g)$. Le mot $\mathbf{a}$ peut être relevé en un mot $\bar{\mathbf{a}} \in I(\mathbf{L}, g)$ en dupliquant les coordonnées de $\mathbf{a}$ relativement à chaque orbite. Soit $\text{wt}(\mathbf{a})$ le *poids* de $\mathbf{a}$. Étant donné que la taille de chaque orbite est au plus égale à m/s , on a

$$\text{wt}(\mathbf{a}) > \text{wt}(\bar{\mathbf{a}}) \frac{s}{m}.$$

Comme $\bar{\mathbf{a}}$ appartient également au code de GOPPA, alors $\text{wt}(\bar{\mathbf{a}}) \geq d$ et donc $d_I \geq d \frac{s}{m}$.

■

Le principe de l'algorithme de décodage des *codes s-projetés* réside dans le principe du relèvement des mots du code dans le *sous-code invariant* du code de GOPPA dont il dérive. On utilise ensuite un algorithme de décodage du code de GOPPA, par exemple celui décrit dans [Pat75]. Enfin, on reprojette le mot obtenu sur le *vecteur générateur* $\mathbf{R}$ du *code s-projeté* correspondant. Partant, la complexité de l'algorithme de décodage est équivalente à celle du décodage du code de GOPPA. Cependant, comme les calculs sont effectués

1. Soit $\mathbf{y} = \mathbf{x} + \mathbf{e}$ le mot reçu, où $\mathbf{y} = (y_{o_j})_{j=1}^N$, $\mathbf{x} = (x_{o_j})_{j=1}^N \in \tilde{I}(\mathbf{R}, g)$ – est le mot que l'on cherche à déterminer, et $\mathbf{e} = (e_{o_j})_{j=1}^N$ est un *vecteur d'erreur* de poids inférieur ou égal à $t \frac{s}{m}$.

2. Le receveur relève le mot $\mathbf{y}$ en un mot $\bar{\mathbf{y}}$ en dupliquant les coordonnées y_{o_i} un nombre $|\mathcal{O}_i|$ fois,

$$\bar{\mathbf{y}} = (\mathbf{y}_{\mathcal{O}_1}, \dots, \mathbf{y}_{\mathcal{O}_N}),$$

où

$$\mathbf{y}_{\mathcal{O}_i} = \underbrace{(y_{o_i}, \dots, y_{o_i})}_{|\mathcal{O}_i| \text{ fois}}.$$

$\bar{\mathbf{y}}$ est un mot de longueur n qui s'écrit donc

$$\bar{\mathbf{y}} = \bar{\mathbf{x}} + \bar{\mathbf{e}},$$

où $\bar{\mathbf{x}} = (\mathbf{x}_{\mathcal{O}_j})_{j=1}^N \in \Gamma(\mathbf{L}, g)$ et $\bar{\mathbf{e}} = (\mathbf{e}_{\mathcal{O}_j})_{j=1}^N$ sont construits de la même manière que $\bar{\mathbf{y}}$. Comme les $\mathcal{O}_i$ sont de taille au plus $\frac{m}{s}$, le poids de $\bar{\mathbf{e}}$ vérifie alors

$$\text{wt}(\bar{\mathbf{e}}) \leq \text{wt}(\mathbf{e}) \frac{m}{s} \leq t.$$

3. Le receveur applique l'algorithme de décodage du code de GOPPA au mot $\bar{\mathbf{y}}$. Comme $\bar{\mathbf{e}}$ est de poids inférieur ou égal à t , il retrouve le mot $\bar{\mathbf{x}}$, qui est un mot du *sous-code invariant* de $\Gamma(\mathbf{L}, g)$.

4. Le receveur récupère le mot $\mathbf{x}$ en prenant les coordonnées du mot $\bar{\mathbf{x}} \in \Gamma(\mathbf{L}, g)$ étiquetées par les éléments de $\mathbf{R}$.

TABLEAU 3.1 : Procédure de décodage d'un code s -projeté

dans un sous-corps $\mathbf{F}_{2^s}$ de $\mathbf{F}_{2^m}$, la complexité du décodage est réduite du facteur constant m/s .

Par souci de simplicité, je suppose que le *polynôme générateur* $g(x) \in \mathbf{F}_{2^s}[x]$ de $\Gamma(\mathbf{L}, g)$ est de degré t , sans facteurs carrés. La distance minimale de $\Gamma(\mathbf{L}, g)$ est alors minorée par $2t + 1$. Les algorithmes de décodage connus permettent alors de corriger t erreurs – voir [Pat75].

D'après la proposition 18, la distance minimale du *code s -projeté* dérivé de $\Gamma(\mathbf{L}, g)$ vérifie l'inégalité

$$d_I \geq (2t + 1) \frac{s}{m}.$$

La procédure décrite dans le tableau 3.1 permet donc de corriger $t \frac{s}{m}$ erreurs dans le code s -projeté, en utilisant un algorithme corrigeant t erreurs dans le code de GOPPA.

Code	code de GOPPA $\Gamma(\mathbf{L}, g)$	code 1-projeté $\tilde{I}(\mathbf{R}, g)$
Longueur	$n = 128$	$N = 20 =$ nombre d'orbites
Dimension réelle	$k = 30$	$k_I = 6, N - t = 6$
Borne sur la dimension	$n - mt = 30$	$N - t = 6$
Taux de transmission	0, 23	0, 33
Correction	14 erreurs	$2 = \frac{14}{7}$ erreurs

TABLEAU 3.2: Propriétés comparées de $\Gamma(\mathbf{L}, g)$ et du code 1-projeté $\tilde{I}(\mathbf{R}, g)$ pour la longueur $n = 128$

Code	code de GOPPA	code s -projeté
Longueur	n	$N \approx n \frac{s}{m}$ quand n croît
Dimension	$k \geq n - mt$	$k_I \geq N - st$
Taux de transmission	$R = \frac{k}{n}$	$R_I = \frac{k_I}{N} \approx R$ quand n croît
Distance minimale	$d = 2t + 1$	$d_I \geq (2t + 1) \frac{s}{m}$

TABLEAU 3.3: Comparaison asymptotique des paramètres des codes de GOPPA et des codes s -projetés.

1.2.3 Codes de GOPPA et codes s -projetés

Soit $g(x) = x^{14} + x^3 + 1$ un polynôme sans racines dans $\mathbf{F}_{2^7}$. Soit $\mathbf{L}$ un étiquetage de $\mathbf{F}_{2^7}$. Les propriétés de $\Gamma(\mathbf{L}, g)$ et du *code 1-projeté* dérivé sont résumées dans le tableau 3.2. Cet exemple montre que, pour de petits paramètres, les codes de GOPPA et les codes s -projetés se comportent différemment – les codes de GOPPA ayant de meilleurs paramètres. Cette différence tend à disparaître quand la longueur et la dimension augmentent. Prenons par exemple le cas où $\frac{m}{s}$ est un nombre premier. Le nombre d'orbites de $\mathbf{F}_{2^m}$ sous l'action de l'automorphisme de FROBENIUS de $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$ est égal à

$$N = n \frac{s}{m} \left(1 + 2^{s-m} \left(\frac{m}{s} - 1 \right) \right),$$

où $n = 2^m$. Ainsi, asymptotiquement, quand $\frac{m}{s}$ reste constant et que la longueur n augmente, on a $N \approx n \frac{s}{m}$. Les résultats concernant les valeurs asymptotiques sont donnés dans le tableau 3.3

2 Décodage de motifs de poids élevés

Dans cette section, j'introduis le principe de *t-décodabilité par tour*. Cela consiste à utiliser les propriétés du groupe d'automorphismes des codes de

GOPPA afin de construire des ensembles de mots décodables de poids supérieur à la capacité de correction des codes.

Dans un premier temps je présente le principe de la *t-décodabilité par tour*. Dans un second temps, je montre comment construire des ensembles *t-décodables par tour*.

2.1 *t-décodabilité par tour*

Dans la suite, je considère un code de GOPPA $\Gamma(\mathbf{L}, g)$, où $g(x) \in \mathbf{F}_{2^s}[x]$ est sans racines dans $\mathbf{F}_{2^m}$. Le *vecteur générateur* $\mathbf{L}$ constitue donc un étiquetage de $\mathbf{F}_{2^m}$ tout entier. Soit σ l'*automorphisme de FROBENIUS* de $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$. D'après la proposition 16, le groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$ contient le groupe

$$\langle \sigma \rangle = \left\{ \sigma, \sigma^2, \dots, \sigma^{m/s} = \mathbf{I} \right\}.$$

Supposons qu'on ait reçu le mot suivant :

$$\mathbf{y} = \mathbf{c} + \mathbf{e}, \tag{3.3}$$

où $\mathbf{c}$ est un mot de $\Gamma(\mathbf{L}, g)$, et où $\mathbf{e}$ est un *vecteur d'erreur*. Si on fait agir le groupe $\langle \sigma \rangle$ sur le mot $\mathbf{y}$, on obtient un ensemble de m/s relations

$$\sigma^i(\mathbf{y}) = \sigma^i(\mathbf{c}) + \sigma^i(\mathbf{e}),$$

pour $i = 0, \dots, m/s - 1$. De plus, comme $\langle \sigma \rangle$ est un sous-groupe du groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$, les mots $\sigma^i(\mathbf{c})$ sont également des mots de $\Gamma(\mathbf{L}, g)$. Donc, toute combinaison linéaire sur $\mathbf{F}_2$ des $\sigma^i(\mathbf{c})$ est un mot de $\Gamma(\mathbf{L}, g)$, *i.e.* pour tout vecteur $(\epsilon_1, \dots, \epsilon_{m/s})$ binaire, et pour tout mot $\mathbf{c} \in \Gamma(\mathbf{L}, g)$,

$$\sum_{i=1}^{m/s} \epsilon_i \sigma^i(\mathbf{c}) \in \Gamma(\mathbf{L}, g).$$

De l'équation (3.3), on infère alors les $2^{m/s}$ équations suivantes :

$$\sum_{i=1}^{m/s} \epsilon_i \sigma^i(\mathbf{y}) = \sum_{i=1}^{m/s} \epsilon_i \sigma^i(\mathbf{c}) + \sum_{i=1}^{m/s} \epsilon_i \sigma^i(\mathbf{e}),$$

où $(\epsilon_1, \dots, \epsilon_{m/s})$ décrit $\mathbf{F}_2^{m/s}$. Donc, s'il existe une combinaison linéaire $(\epsilon_1^{(0)}, \dots, \epsilon_{m/s}^{(0)}) \in \mathbf{F}_2^{m/s}$ des puissances de σ

$$\mathbf{E} = \sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i(\mathbf{e})$$

telle que le poids de $\mathbf{E}$ soit inférieur à la capacité de correction de $\Gamma(\mathbf{L}, g)$, alors si on note

$$\mathbf{Y} = \sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i(\mathbf{y}),$$

alors on a

$$\mathbf{Y} = \underbrace{\sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i(\mathbf{c})}_{\in \Gamma(\mathbf{L}, g)} + \mathbf{E}.$$

On peut donc décoder $\mathbf{Y}$ en utilisant l'algorithme de décodage du code $\Gamma(\mathbf{L}, g)$. On retrouve alors $\mathbf{E}$, le problème étant que, en général, on veut retrouver le *vecteur d'erreur* $\mathbf{e}$. Ce sont ces remarques qui m'amènent à introduire la définition suivante :

Définition 19 (*t*-décodabilité par tour) Soit $\mathcal{E}$ un ensemble de mots de vecteur générateur $\mathbf{L}$. Alors, on dit que $\mathcal{E}$ est *t*-décodable par tour si : pour tout mot $\mathbf{e} \in \mathcal{E}$,

1. il existe une combinaison linéaire $(\epsilon_1, \dots, \epsilon_{m/s})$ sur $\mathbf{F}_2$ telle que

$$\begin{cases} \mathbf{E} = \sum_{i=1}^{m/s} \epsilon_i \sigma^i(\mathbf{e}), \\ \text{wt}(\mathbf{E}) \leq t, \end{cases}$$

2. la connaissance de $\mathbf{E}$ permet de recouvrer $\mathbf{e} \in \mathcal{E}$.

En d'autres termes, étant donné un code C que l'on sait décoder jusqu'à la distance t , et dont le groupe d'automorphismes contient $\langle \sigma \rangle$, la *t*-décodabilité par tour de $\mathcal{E}$ assure que le mot

$$\mathbf{y} = \mathbf{c} + \mathbf{e}$$

où $\mathbf{c} \in C$ et $\mathbf{e} \in \mathcal{E}$, est décodable dans C .

Proposition 19 Soit $\Gamma(\mathbf{L}, g)$ un code de GOPPA, où $g(x) \in \mathbf{F}_{2^s}[x]$ est de degré t et sans facteurs carrés. Alors, tout mot d'un ensemble *t*-décodable par tour est décodable dans $\Gamma(\mathbf{L}, g)$.

Preuve

Soit $\mathcal{E}$ un ensemble *t*-décodable par tour, et soit

$$\mathbf{y} = \mathbf{c} + \mathbf{e},$$

où $\mathbf{c} \in \Gamma(\mathbf{L}, g)$ et $\mathbf{e} \in \mathcal{E}$. D'après la définition 18, il existe une combinaison linéaire

$$\mathbf{E} = \sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i(\mathbf{e})$$

de poids inférieur à t . Comme le *polynôme générateur* g de $\Gamma(\mathbf{L}, g)$ est de degré t et sans facteurs carrés, il existe un algorithme de décodage permettant de corriger t erreurs dans $\Gamma(\mathbf{L}, g)$. Donc, en appliquant cet algorithme au mot

$$\sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i(\mathbf{y}).$$

On recouvre $\mathbf{E}$. Comme $\mathcal{E}$ est t -décodable par tour, on retrouve ensuite le *vecteur d'erreur* $\mathbf{e}$.

■

2.2 Construction d'ensembles t -décodables par tour

Construire des ensembles t -décodables par tour n'est pas chose aisée. En effet, bien que la première condition de la définition 19 soit en général facile à réaliser, la seconde est en revanche bien plus difficile à réaliser.

2.2.1 Extensions de degré 5

Soit s un entier strictement positif. Considérons l'extension $\mathbf{F}_{2^{5s}}/\mathbf{F}_{2^s}$ dont l'*automorphisme de FROBENIUS* est $\sigma : x \mapsto x^{2^s}$. Le groupe $\langle \sigma \rangle$ est d'ordre 5. Comme 5 est un nombre premier, les orbites des éléments de $\mathbf{F}_{2^{5s}}$ sous l'action de σ sont toutes de taille 5, excepté les orbites des éléments de $\mathbf{F}_{2^s}$ qui, elles, sont de taille 1. Il y a donc exactement

$$N = \frac{2^{5s} - 2^s}{5}$$

orbites de taille 5. Soit $o_1, \dots, o_N$ des représentants des N orbites de taille 5 de $\mathbf{F}_{2^{5s}}/\mathbf{F}_{2^s}$, et soit $\alpha_1, \dots, \alpha_{2^s}$ un étiquetage quelconque des éléments de $\mathbf{F}_{2^s}$. On considère $\mathbf{L}$ l'étiquetage de $\mathbf{F}_{2^{5s}}$ suivant :

$$\mathbf{L} = (\mathcal{O}_1, \dots, \mathcal{O}_N, \alpha_1, \dots, \alpha_{2^s}),$$

où $\mathcal{O}_i = (o_i, \sigma(o_i), \dots, \sigma^4(o_i))$ désigne l'orbite de o_i sous l'action de $\langle \sigma \rangle$. Soit $\mathbf{a}$ un mot de *vecteur générateur* $\mathbf{L}$. $\mathbf{a}$ s'écrit alors

$$\mathbf{a} = (\mathbf{a}_1, \dots, \mathbf{a}_N, a_{\alpha_1}, \dots, a_{\alpha_{2^s}}),$$

où $\mathbf{a}_i = (a_{o_i}, a_{\sigma(o_i)}, \dots, a_{\sigma^4(o_i)})$ désigne le vecteur formé des coordonnées de $\mathbf{a}$ étiquetées par $\mathcal{O}_i$. Considérons alors l'ensemble des mots de vecteur générateur $\mathbf{L}$ construit de la manière suivante :

Construction 1 Soit p un entier positif et soit $\mathcal{E}$ l'ensemble de tous les mots binaires de vecteur générateur $\mathbf{L}$

$$\mathbf{e} = (\mathbf{e}_1, \dots, \mathbf{e}_N, \underbrace{0, \dots, 0}_{2^s \text{ fois}}),$$

tels que :

1. Il existe un sous ensemble $I \subset \{1, 2, \dots, N\}$ de taille p tel que pour tout $i \in I$, $\mathbf{e}_i$ est de poids 3,
2. Pour tout j n'appartenant pas à I ,

$$\mathbf{e}_j = (0, 0, 0, 0, 0).$$

De la construction de $\mathcal{E}$, on déduit la proposition suivante

Proposition 20 Soit $\mathcal{E}$ l'ensemble des mots ainsi construits, alors

1. $\mathcal{E}$ est un ensemble de mots de poids constant égal à $3p$.
2. Le cardinal de $\mathcal{E}$ est égal à $10^p \binom{N}{p}$.
3. $\mathcal{E}$ est $2p$ -décodable par tour.

Preuve

- *Première assertion* : d'après la construction 1, les mots de $\mathcal{E}$ sont fabriqués en choisissant p motifs de poids 3. Tous les mots de $\mathcal{E}$ ont donc un poids égal à $3p$.
- *Deuxième assertion* : on peut choisir de $\binom{N}{p}$ manière différente p orbites parmi les N orbites de taille 5. Pour chacune d'entre elle, il y a $\binom{5}{3} = 10$ motifs de poids 3 différents. Donc

$$|\mathcal{E}| = \binom{N}{p} \binom{5}{3}^p = \binom{N}{p} 10^p.$$

- *Troisième assertion* : soit

$$\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N, 0, \dots, 0) \in \mathcal{E}$$

où $\mathbf{e}_i = (e_{o_i}, e_{\sigma(o_i)}, e_{\sigma^2(o_i)}, e_{\sigma^3(o_i)}, e_{\sigma^4(o_i)})$. Par construction de $\mathcal{E}$, chaque sous-vecteur $\mathbf{e}_i$ est de poids exactement égal à 0 ou à 3.

L'action de l'automorphisme de FROBENIUS sur le mot $\mathbf{e}$ se décompose donc en produit de *permutations circulaires* sur les sous-vecteurs $\mathbf{e}_i$. Les $\binom{5}{3} = 10$ motifs de longueur 5 et de poids 3 peuvent ainsi être divisés en deux classes $\mathbf{f}_1$ et $\mathbf{f}_2$, modulo l'action du groupe $\langle \sigma \rangle$ engendré par l'automorphisme de FROBENIUS.

$$\begin{array}{l} \mathbf{f}_1 \\ \sigma(\mathbf{f}_1) \\ \sigma^2(\mathbf{f}_1) \\ \sigma^3(\mathbf{f}_1) \\ \sigma^4(\mathbf{f}_1) \end{array} = \begin{array}{l} \text{Type 1} \\ (11100) \\ (01110) \\ (00111) \\ (10011) \\ (11001) \end{array} \parallel \begin{array}{l} \mathbf{f}_2 \\ \sigma(\mathbf{f}_2) \\ \sigma^2(\mathbf{f}_2) \\ \sigma^3(\mathbf{f}_2) \\ \sigma^4(\mathbf{f}_2) \end{array} = \begin{array}{l} \text{Type 2} \\ (11010) \\ (01101) \\ (10110) \\ (01011) \\ (10101) \end{array}$$

Les motifs $\mathbf{f}_1$ et $\mathbf{f}_2$ jouent des rôles symétriques. Il existe des combinaisons linéaires des puissances de l'automorphisme de FROBENIUS

permettant de réduire les motifs de poids 3 à certains motifs de poids 1, tout en conservant le poids de l'autre. En effet, on a

$$\begin{cases} \mathbf{f}_1 + \sigma(\mathbf{f}_1) + \sigma^2(\mathbf{f}_1) = (10101), \\ \mathbf{f}_2 + \sigma(\mathbf{f}_2) + \sigma^2(\mathbf{f}_2) = (00001), \end{cases}$$

et

$$\begin{cases} \mathbf{f}_1 + \sigma^2(\mathbf{f}_1) + \sigma^3(\mathbf{f}_1) = (01000), \\ \mathbf{f}_2 + \sigma^2(\mathbf{f}_2) + \sigma^3(\mathbf{f}_2) = (00111). \end{cases}$$

De plus, dès qu'on connaît l'image d'un motif $\mathbf{f}$ de longueur 5 et de poids 3 par $\mathbf{I} + \sigma + \sigma^2$ ou bien par $\mathbf{I} + \sigma^2 + \sigma^3$, on retrouve le motif $\mathbf{f}$ de manière unique. En effet, soit $\mathbf{f}$ un motif tel que

$$\mathbf{f} + \sigma(\mathbf{f}) + \sigma^2(\mathbf{f}) = (01101) = \sigma(\mathbf{f}_2).$$

$\mathbf{f}$ est nécessairement un motif de type 1. En conséquence, il existe un entier $i \in \{0, 1, 2, 3, 4\}$ tel que

$$\mathbf{f} = \sigma^i(\mathbf{f}_1),$$

où $\mathbf{f}_1 = (11100)$. L'unique motif de type 1 solution de cette équation est

$$\mathbf{f} = \sigma^2(\mathbf{f}_1) = (00111).$$

Les autres cas sont identiques.

Supposons que $\mathbf{e} \in \mathcal{E}$ soit formé de p_1 motifs de type 1, et de p_2 motifs de type 2. Alors

$$\begin{cases} \mathbf{e} + \sigma(\mathbf{e}) + \sigma^2(\mathbf{e}) \text{ est de poids } 3p_1 + p_2, \\ \mathbf{e} + \sigma^2(\mathbf{e}) + \sigma^3(\mathbf{e}) \text{ est de poids } p_1 + 3p_2. \end{cases}$$

Comme

$$p = p_1 + p_2,$$

alors, ou bien $3p_1 + p_2 \leq 2p$, ou bien $p_1 + 3p_2 \leq 2p$. Donc il existe une combinaison linéaire sur $\mathbf{F}_2$ des puissances de l'automorphisme de FROBENIUS qui donne un mot de poids inférieur à $2p$. $\mathcal{E}$ remplit donc la première condition de la définition 19.

De plus, le mot $\mathbf{E} = \mathbf{e} + \sigma(\mathbf{e}) + \sigma^2(\mathbf{e})$ s'écrit

$$\mathbf{E} = (\mathbf{e}_1 + \sigma(\mathbf{e}_1) + \sigma^2(\mathbf{e}_1), \dots, \mathbf{e}_N + \sigma(\mathbf{e}_N) + \sigma^2(\mathbf{e}_N), 0, \dots, 0).$$

À partir des $\mathbf{e}_i + \sigma(\mathbf{e}_i) + \sigma^2(\mathbf{e}_i)$ on a vu que l'on pouvait recouvrir les motifs $\mathbf{e}_i$ de manière unique. Donc on recouvre également $\mathbf{e}$ de manière unique. $\mathcal{E}$ remplit la seconde condition de la définition 19.

$\mathcal{E}$ est donc $2p$ -décodable par tour.

■

Exemple 13

Considérons l'extension $\mathbf{F}_{2^{10}}/\mathbf{F}_{2^2}$, et le paramètre $t = 50$. Soit $\mathcal{E}$ l'ensemble 50-décodable par tour construit suivant la procédure décrite précédemment.

Le nombre d'orbites de taille 5 de $\mathbf{F}_{2^{10}}/\mathbf{F}_2$ est égal à 204. On a alors :

- $|\mathcal{E}| \approx 2^{188}$.
- Les vecteurs de $\mathcal{E}$ ont un poids de HAMMING égal à 75.

3 Renforcement du système de MCELIECE

Les ensembles *t-décodables par tour* introduits dans la section précédente ont une application en cryptographie. Ils permettent de renforcer le cryptosystème de MCELIECE contre les *attaques par décodage*. En effet, la proposition 19 montre que les ensembles *t-décodables par tour* sont décodables relativement à des codes de GOPPA de capacité de correction t . Je me sers de cette propriété pour modifier le cryptosystème de MCELIECE en publiant des ensembles *t-décodables par tour* pour le *code public*. Comme ces ensembles contiennent des mots de poids supérieur à t , la complexité des *attaques par décodage* utilisant les algorithmes décrits dans la section 1.2 du chapitre 1, est considérablement accrue.

Dans un premier temps, je présente le cryptosystème de MCELIECE modifié par l'utilisation des propriétés du *groupe d'automorphismes* de codes de GOPPA spécifiques. Dans une seconde partie, la discussion concerne les failles de sécurité introduites par ces modifications. Dans un troisième temps, je donne un exemple concret en utilisant l'ensemble *t-décodable par tour* construit dans la section 2.2.1.

3.1 Description du schéma

Soit $\mathbf{L}$ un étiquetage de $\mathbf{F}_{2^m}$, et soit $\mathbf{F}_{2^s}$ un sous-corps de $\mathbf{F}_{2^m}$. Je désigne par σ l'*automorphisme de FROBENIUS* de $\mathbf{F}_{2^m}/\mathbf{F}_{2^s}$. L'*espace des clés* $\mathcal{G}$ est formée par la famille des codes de GOPPA binaires $\Gamma(\mathbf{L}, g_1g)$ tels que :

- $g_1(x) \in \mathbf{F}_{2^m}[x]$ est un polynôme fixé sans racine dans $\mathbf{F}_{2^m}$. g_1 est appelé *polynôme masquant*.
- g décrit l'ensemble des polynômes de degré t , à coefficients dans $\mathbf{F}_{2^s}$, un sous corps de $\mathbf{F}_{2^m}$, sans facteurs carrés et sans racine dans $\mathbf{F}_{2^m}$.

Comme, pour tout polynôme g , on a

$$\Gamma(\mathbf{L}, gg_1) \subset \Gamma(\mathbf{L}, g),$$

l'algorithme de décodage du code $\Gamma(\mathbf{L}, g)$ permet également de décoder $\Gamma(\mathbf{L}, g_1g)$ jusqu'à la distance t . La modification du système de chiffrement de MCELIECE est décrite dans le tableau 3.4.

La complexité en temps du cryptosystème s'évalue de la manière suivante :

- *Clé privée* :
 - Un code aléatoire $\Gamma(\mathbf{L}, g_1g) \in \mathcal{G}$ appelé *code caché*.
 - Une permutation π aléatoire du vecteur générateur $\mathbf{L}$.
- *Clé publique* :
 - Une matrice génératrice aléatoire $\mathbf{G}_{\text{pub}}$ du code $\pi(\Gamma(\mathbf{L}, g_1g))$.
 - Une façon d'engendrer $\pi(\mathcal{E})$, où $\mathcal{E}$ désigne un ensemble *t-décodable par tour*.

Les procédures de *chiffrement* et de *déchiffrement* sont les suivantes.

- *Chiffrement* : soit $\mathbf{x}$ le message binaire de longueur k que l'on doit chiffrer. L'émetteur calcule et envoie

$$\mathbf{y} = \mathbf{x}\mathbf{G}_{\text{pub}} + \mathbf{e},$$

où $\mathbf{e}$ est choisi aléatoirement dans $\pi(\mathcal{E})$.

- *Déchiffrement* : le receveur calcule

$$\pi^{-1}(\mathbf{y}) = \mathbf{x}\pi^{-1}(\mathbf{G}_{\text{pub}}) + \pi^{-1}(\mathbf{e}). \quad (3.4)$$

$\pi^{-1}(\mathbf{G}_{\text{pub}})$ est une matrice génératrice de $\Gamma(\mathbf{L}, g_1g)$. Donc

$$\mathbf{c} = \mathbf{x}\pi^{-1}(\mathbf{G}_{\text{pub}}) \in \Gamma(\mathbf{L}, g_1g).$$

Comme $\pi^{-1}(\mathbf{e}) \in \mathcal{E}$, il existe une combinaison linéaire $\mathbf{E}$ des puissances de σ , soit $(\epsilon_1^{(0)}, \dots, \epsilon_{m/s}^{(0)})$, telle que

$$\mathbf{E} = \sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i(\pi^{-1}(\mathbf{e})), \quad \text{et } \text{wt}(\mathbf{E}) \leq t.$$

L'équation (3.4) donne

$$\sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i(\pi^{-1}(\mathbf{y})) = \sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i(\mathbf{c}) + \mathbf{E}.$$

De plus comme $\Gamma(\mathbf{L}, g_1g) \subset \Gamma(\mathbf{L}, g)$, et que $\langle \sigma \rangle$ est un sous-groupe du groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$, $\sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i(\mathbf{c})$ appartient à $\Gamma(\mathbf{L}, g)$. On peut donc décoder $\mathbf{E}$ dans $\Gamma(\mathbf{L}, g)$. À partir de là, on recouvre $\mathbf{e}$, en utilisant les propriétés de l'ensemble $\mathcal{E}$.

TABLEAU 3.4 : Cryptosystème de McELIECE modifié

- *En chiffrement* : le coût est identique au coût du système de McELIECE, à savoir $nk/2$ opérations binaires en moyenne pour une opération de chiffrement. Cela correspond exactement au calcul de

$$\mathbf{xG}_{\text{pub}},$$

qui est la multiplication d'un mot binaire de longueur k par une matrice binaire de taille $k \times n$. On suppose négligeable le coût du tirage aléatoire d'un vecteur d'erreur dans l'ensemble $\mathcal{E}$, *t-décodable par tour*.

- *En déchiffrement* : le coût du déchiffrement dépend non seulement de l'algorithme de décodage des codes de GOPPA, mais également de la structure de l'ensemble *t-décodable par tour* $\mathcal{E}$. En effet, dans la phase de *déchiffrement*, on détermine le vecteur d'erreur initial $\mathbf{e}$ d'un chiffré

$$\mathbf{y} = \mathbf{xG}_{\text{pub}} + \mathbf{e}$$

à partir de la combinaison linéaire $\mathbf{E}$ des puissances de σ . Retrouver $\mathbf{e}$ à partir de $\mathbf{E}$ ne dépend donc que de la manière dont l'ensemble $\mathcal{E}$ est construit.

3.2 Sécurité du schéma

Par rapport au système de McELIECE décrit dans la section 1.1.1 du chapitre 1, les modifications interviennent au niveau de la *clé publique*. Ici, outre le *code public*, il faut publier l'information nécessaire permettant d'engendrer un ensemble *t-décodable par tour*. C'est pourquoi, il faut s'assurer également que la façon dont l'ensemble $\mathcal{E}$ est construit n'affaiblit pas le cryptosystème. À ces fins, l'ensemble $\mathcal{E}$ doit au moins vérifier certaines conditions «cryptographiques» qui sont :

Conditions 1

1. $\mathcal{E}$ doit être constitué de mots de poids supérieur à t . Si tel n'est pas le cas, le cryptosystème décrit dans le tableau 3.4 n'offre aucun avantage contre les attaques par décodage par rapport au système de chiffrement classique. En effet, ce schéma a été construit pour renforcer le cryptosystème de McELIECE. Or, si le poids des vecteurs d'erreurs est plus petit que dans le système original, on affaiblit le cryptosystème.
2. $\mathcal{E}$ doit être de grande taille. Si un attaquant peut énumérer $\mathcal{E}$, il peut retrouver le texte clair sans coup férir.
3. La façon d'engendrer $\mathcal{E}$ doit être publique et ne doit pas révéler d'information sur le cryptosystème autre que l'information fournie par la clé publique.

Le lecteur perspicace se demandera dans quel but j'ai introduit le *polynôme masquant* g_1 . On peut tout à fait remplacer la famille $\Gamma(\mathbf{L}, g_1g)$ des

codes constituant l'espace des clés par la famille $\Gamma(\mathbf{L}, g)$ où g décrit l'ensemble des polynômes de degré t , à coefficients dans $\mathbf{F}_{2^s}$, sans facteurs carrés et sans racines dans $\mathbf{F}_{2^m}$.

Cependant, il y a une faille de taille. En effet, l'algorithme de séparation des supports – voir la section 1.3 du chapitre 2 et [Sen00] – appliqué à un code C détermine entre autres le groupe d'automorphismes de C . Soit $\Gamma(\mathbf{L}, g)$, un code de GOPPA, dont le polynôme générateur est à coefficients dans le sous-corps $\mathbf{F}_{2^s}$ de $\mathbf{F}_{2^m}$. D'après la proposition 16, le groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$ contient le groupe engendré par l'automorphisme de FROBENIUS σ :

$$\langle \sigma \rangle = \{\sigma, \dots, \sigma^{m/s}\}.$$

Supposons que le code public d'un cryptosystème de McELIECE soit

$$C_{\text{pub}} = \pi(\Gamma(\mathbf{L}, g)),$$

où π est une permutation du vecteur générateur $\mathbf{L}$. Alors le groupe d'automorphismes de C_{pub} contient le groupe

$$\mathcal{A} = \{\pi \circ \sigma \circ \pi^{-1}, \dots, \pi \circ \sigma^{m/s} \circ \pi^{-1}\}.$$

Appliquer SSA au code public C_{pub} permet de retrouver le groupe $\mathcal{A}$. À partir de là tout devient naturel. Soit un chiffré

$$\mathbf{y} = \mathbf{c} + \mathbf{e}$$

où $\mathbf{c}$ est un mot de C_{pub} et où $\mathbf{e}$ est un mot de $\pi(\mathcal{E})$, l'image par π d'un ensemble t -décodable par tour. Connaissant $\mathcal{A}$, on peut calculer toutes les combinaisons linéaires

$$\sum_{i=1}^{m/s} \epsilon_i (\pi \circ \sigma^i \circ \pi^{-1})(\mathbf{y}) = \sum_{i=1}^{m/s} \epsilon_i (\pi \circ \sigma^i \circ \pi^{-1})(\mathbf{c}) + \sum_{i=1}^{m/s} \epsilon_i \pi \circ \sigma^i (\pi^{-1}(\mathbf{e})),$$

pour $(\epsilon_1, \dots, \epsilon_{m/s}) \in \mathbf{F}_2^{m/s}$. Comme, par construction $\pi^{-1}(\mathbf{e}) \in \mathcal{E}$, il existe une combinaison linéaire $(\epsilon_1^{(0)}, \dots, \epsilon_{m/s}^{(0)})$, telle que

$$\begin{cases} \mathbf{E} = \sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i (\pi^{-1}(\mathbf{e})), \\ \text{wt}(\mathbf{E}) \leq t. \end{cases}$$

Donc la combinaison linéaire

$$\mathbf{E}' = \sum_{i=1}^{m/s} \epsilon_i^{(0)} \pi \circ \sigma^i (\pi^{-1}(\mathbf{e})) = \pi \left(\sum_{i=1}^{m/s} \epsilon_i^{(0)} \sigma^i (\pi^{-1}(\mathbf{e})) \right) = \pi(\mathbf{E})$$

est également de poids inférieur ou égal à t . Partant, la connaissance de $\mathcal{A}$ permet à un attaquant de ramener le problème du décodage dans un ensemble t -décodable par tour à celui du décodage borné par la distance t .

L'intérêt du *polynôme masquant* est précisément d'éviter ce problème. Bien que $\Gamma(\mathbf{L}, g_1g)$ soit un sous-code de $\Gamma(\mathbf{L}, g)$, dont le *groupe d'automorphismes* contient $\langle \sigma \rangle$, son groupe d'automorphismes est néanmoins trivial. $\mathcal{SSA}$ appliqué au *code public* ne donne donc aucune information sur la structure du groupe d'automorphismes de $\Gamma(\mathbf{L}, g)$.

3.3 Application aux extensions de degré 5

Je prends comme *polynôme masquant* un polynôme g_1 de degré 2 à coefficients dans le corps $\mathbf{F}_{2^{5s}}$ et sans racines. Soit σ l'automorphisme de FROBENIUS de $\mathbf{F}_{2^{5s}}/\mathbf{F}_{2^s}$. Soit $\mathbf{L}$ l'étiquetage suivant de $\mathbf{F}_{2^{5s}}$.

$$\mathbf{L} = (\mathcal{O}_1, \dots, \mathcal{O}_N, \alpha_1, \dots, \alpha_{2^s}),$$

où $\mathcal{O}_i = (o_i, \sigma(o_i), \dots, \sigma^4(o_i))$ désigne l'orbite de o_i sous l'action de $\langle \sigma \rangle$.

L'*espace des clés* est la famille

$$\mathcal{G} = \left\{ \Gamma(\mathbf{L}, g_1g) \left| \begin{array}{l} g \in \mathbf{F}_{2^s}[x] \\ g \text{ n'a pas de racines dans } \mathbf{F}_{2^{5s}}, \\ g \text{ sans facteurs carrés,} \end{array} \right. \right\}.$$

La *clé privée* du système est constituée d'un code $\Gamma(\mathbf{L}, g_1g)$ tiré aléatoirement dans $\mathcal{G}$ et d'une permutation π de $\mathbf{L}$.

La *clé publique* est cette fois constituée de :

- Une matrice génératrice aléatoire du *code public* $\pi(\Gamma(\mathbf{L}, g_1g))$,
- Les positions des orbites constituant $\mathbf{L}$, *i.e.* le concepteur public

$$\pi(\mathcal{O}_1), \dots, \pi(\mathcal{O}_N).$$

où $\pi(\mathcal{O}_i) = (\pi(o_i), \pi \circ \sigma(o_i), \dots, \pi \circ \sigma^4(o_i))$. Cette information est suffisante pour construire l'ensemble $\pi(\mathcal{E})$, image par la permutation π de l'ensemble t -décodable par tour $\mathcal{E}$, suivant la construction 1. En effet, pour un utilisateur du système, choisir un mot aléatoire dans $\pi(\mathcal{E})$ revient à construire le mot

$$\mathbf{e} = (\mathbf{e}_1, \dots, \mathbf{e}_N, 0, \dots, 0),$$

où $\mathbf{e}_i = \pi(\mathbf{e}_{\mathcal{O}_i}) = (e_{\pi^{-1}(o_i)}, \dots, e_{\pi^{-1} \circ \sigma^4(o_i)})$, en

1. choisissant aléatoirement $t/2$ orbites de taille 5 parmi N ,
2. en plaçant aléatoirement trois 1 sur chacune des $t/2$ orbites choisies.

Dans ce cas, je peux évaluer le coût de la procédure de déchiffrement. En effet, pour déchiffrer le message

$$\mathbf{y} = \mathbf{x}\mathbf{G}_{\text{pub}} + \mathbf{e},$$

où $\mathbf{e} \in \mathcal{E}$, le receveur procède en quatre phases :

1. Il calcule $\mathbf{y}' = \pi^{-1}(\mathbf{y})$. On peut supposer que le coût de cette opération est négligeable par rapport aux coûts des autres phases.
2. Il calcule les combinaisons linéaires des puissances de σ :

$$\begin{cases} \mathbf{y}'_1 = \mathbf{y}' + \sigma(\mathbf{y}') + \sigma^2(\mathbf{y}'), \\ \mathbf{y}'_2 = \mathbf{y}' + \sigma^2(\mathbf{y}') + \sigma^3(\mathbf{y}'). \end{cases}$$

Soit s_n le coût du calcul de $\sigma(\mathbf{y}')$. Calculer $\sigma^2(\mathbf{y}')$ à partir de $\sigma(\mathbf{y}')$ coûte s_n . De même, le calcul de $\sigma^3(\mathbf{y}')$ à partir de $\sigma^2(\mathbf{y}')$ coûte également s_n . Soit a_n le coût de la somme de deux mots binaires de longueur n . Alors, le coût du calcul de $\mathbf{y}'_1$ et $\mathbf{y}'_2$ est au plus égal à $3(s_n + a_n)$. Comme l'automorphisme σ peut être vu comme un produit de permutations circulaires – voir la construction 1 –, le coût de cette phase est négligeable en regard du coût de la phase de décodage.

3. *Décodage* : il applique l'algorithme de décodage du *code caché* au vecteur $\mathbf{y}'_1$. Si l'algorithme termine normalement, alors il recouvre

$$\mathbf{E}' = \mathbf{e}' + \sigma(\mathbf{e}') + \sigma^2(\mathbf{e}')$$

où $\mathbf{e}' = \pi^{-1}(\mathbf{e}) \in \mathcal{E}$, est de poids inférieur ou égal à t .

Si l'algorithme échoue, alors $\mathbf{E}'$ est de poids supérieur à t . Donc, d'après la construction 1,

$$\mathbf{E}'' = \mathbf{e}' + \sigma^2(\mathbf{e}') + \sigma^3(\mathbf{e}')$$

est de poids inférieur ou égal à t . Le receveur applique alors l'algorithme de décodage au vecteur $\mathbf{y}'_2$, et recouvre $\mathbf{E}''$.

Au final, le receveur utilise l'algorithme de décodage 1,5 fois en moyenne. Soit A_t le coût d'une utilisation de l'algorithme. La complexité moyenne de l'étape de décodage est donc égale à $\frac{3}{2}A_t$, tandis que pour le cryptosystème de McELIECE, le coût de l'étape de décodage est égal à A_t .

En reprenant la complexité donnée dans [Can96, p. 20] pour l'algorithme de décodage des codes de GOPPA, la complexité de cette étape devient

$$A_t \approx 15snt + 100s^2t^2 \text{ opérations binaires.}$$

4. Connaissant $\mathbf{E}'$, les opérations de recouvrement du vecteur d'erreur $\mathbf{e}$ ont un coût négligeable par rapport au coût de l'étape de décodage.

Finalement, la complexité totale de l'étape de déchiffrement est égale à

$$\frac{3}{2}A_t \approx 22,5snt + 150s^2t^2 \text{ opérations binaires.}$$

Du point de vue de la sécurité, l'ensemble *t-décodable par tour* $\mathcal{E}$ satisfait deux conditions pour être cryptologiquement sûr, à savoir :

- $\mathcal{E}$ est constitué de mots de poids $\frac{3t}{2} > t$.

– D’après la proposition 20, on a

$$|\mathcal{E}| = 10^{\lfloor t/2 \rfloor} \binom{N}{\lfloor t/2 \rfloor},$$

et peut donc être choisi de manière à n’être pas énumérable.

La troisième condition est plus problématique. En effet, le concepteur fournit des informations sur les images des orbites de taille 5 par la permutation π , et donc donne les positions orbites des éléments de $\mathbf{F}_{2^{5s}}$ sous l’action de $\langle \pi \circ \sigma \circ \pi^{-1} \rangle$. Comme le *groupe d’automorphismes* du *code public* est trivial, le concepteur donne plus d’information que n’en pourrait récupérer un attaquant en étudiant seulement la structure du *code public*. Mais fort heureusement, connaître la position des orbites d’un groupe, ne suffit pas pour déterminer le groupe en question. Cette information semble donc inexploitable par un attaquant potentiel.

Exemple 14

Considérons l’extension $\mathbf{F}_{2^{10}}/\mathbf{F}_{2^2}$, et l’*espace des clés* $\mathcal{G}$ constitué des codes de GOPPA $\Gamma(\mathbf{L}, g_1g)$ où g est un polynôme de degré 50, sans racines dans $\mathbf{F}_{2^{10}}$ et sans facteurs carrés. Alors

- la taille de l’espace des clés est $|\mathcal{G}| \approx 2^{95}$,
- Le coût d’un déchiffrement est environ égal à 7500 opérations binaires par bit d’information.
- $\mathcal{E}$ est un ensemble de vecteurs de poids égal à $75 > 50$. Pour ces paramètres, le coût du meilleur algorithme de décodage par ensembles d’information fournit un facteur de travail de

$$2^{91} \text{ opérations binaires.}$$

Comparés aux 2^{64} opérations binaires nécessaires pour décrypter un *chiffré* de l’instance générale du cryptosystème de MCELIECE c’est un gain très appréciable.

- $|\mathcal{E}| = 2^{188}$, donc $\mathcal{E}$ n’est pas énumérable.

Pour ces paramètres, $\mathcal{E}$ satisfait les conditions «cryptographiques» requises.

4 Conclusion

Dans ce chapitre, j’ai montré comment renforcer la sécurité du cryptosystème de MCELIECE, à taille de clé constante, en augmentant la sécurité contre les *attaques par décodage* dont la complexité est la plus critique – voir le chapitre 1 –, sans pour autant trop affaiblir le cryptosystème vis à vis des *attaques structurelles*. En effet avec les paramètres que j’ai considérés, la taille seule de l’espace des clés empêche toute énumération.

Pour parvenir à ces fins, un concepteur doit être en mesure de construire des ensembles t -décodables par tour, cryptologiquement sûrs, c’est-à-dire qui vérifient toutes les conditions 1. Hormis les ensembles t -décodables construits à partir d’extensions de degré 5, je n’ai pas trouvé d’ensemble qui satisfasse les critères de sécurité.

Chapitre 4

Cryptosystèmes fondés sur la métrique rang

Dans les chapitres précédents, j'ai étudié les systèmes de chiffrement de McELIECE et de NIEDERREITER. La sécurité de ces systèmes repose d'une part sur la famille de codes utilisée comme *espace des clés* et d'autre part sur le problème plus général du *décodage borné* d'un code pour la *métrique de HAMMING*. Ce problème est réputé difficile puisqu'il n'existe pas d'algorithme connu de complexité polynomiale permettant d'en résoudre l'instance générale. Cependant, les algorithmes de décodage par *ensembles d'information* permettent d'attaquer le système si le concepteur utilise des codes de petites tailles. Pour une bonne sécurité, il faut donc utiliser des tailles de clés de plusieurs centaines de milliers de bits, – voir [CC98] et [CS98]. Cela peut poser un problème si on souhaite implanter de tels cryptosystèmes sur des supports à ressources limitées, comme des puces électroniques.

Cependant, GABIDULIN, PARAMONOV et TRETJAKOV remédièrent à ce problème en élaborant en 1991 – voir [GPT91] – un cryptosystème dont la sécurité repose sur le problème du décodage borné d'un code, non plus pour la *métrique de HAMMING*, mais pour la *métrique rang*. Celle-ci fut introduite par GABIDULIN en 1985 dans [Gab85b]. Puisque les *attaques par décodage*, présentées dans la section 1.2 du chapitre 1, dépendent exclusivement de la métrique dans laquelle on se place, le cryptosystème permet alors l'utilisation de clés de quelques milliers de bits seulement.

Dans ce chapitre, je présente dans un premier temps la théorie puis les propriétés de la *métrique rang*. Ensuite je montre qu'on peut construire des codes optimaux pour cette métrique appelés *codes de GABIDULIN*. Ceux-ci sont les analogues pour la métrique rang des codes de REED–SOLOMON généralisés. En conséquence, on peut dériver un algorithme de décodage en temps polynomial pour les codes de GABIDULIN, calqué sur l'algorithme de décodage des codes de REED–SOLOMON généralisés. Je montre enfin que les *sous-codes traces* de codes de GABIDULIN sont équivalents à des produits directs de *codes de GABIDULIN*. Ce sont donc des codes très structurés. Dans

une seconde partie je décris les forces et les faiblesses du cryptosystème GPT qui utilise comme *espace des clés* la famille des codes de GABIDULIN. Je montre également qu'il est possible de pallier simplement les carences du système original en utilisant des sous-codes de codes de GABIDULIN dont on sait contrôler la structure.

1 Théorie de la métrique rang et codes optimaux

La théorie de la *métrique rang* fut introduite en 1985 par GABIDULIN dans l'article [Gab85b]. Cette théorie constitue une généralisation des travaux de l'auteur sur les codes qui corrigent des *tableaux d'erreur* – voir [GK72]. GABIDULIN exhibe des similarités et des différences entre la *métrique rang* et la *métrique de HAMMING*. Il parvient notamment à généraliser la *borne de SINGLETON* et fournit ainsi des conditions d'existence de codes de paramètres donnés. Il présenta également le moyen de construire une famille de codes atteignant la borne de SINGLETON. Ces codes sont de fait les analogues pour la *métrique rang* des codes de REED–SOLOMON généralisés, présentés à la section 2.1 du chapitre 1. De cette façon, il en déduit un algorithme de décodage en temps polynomial jusqu'à leur capacité de correction construite. Cependant, cette famille de codes optimaux et décodables fit indépendamment l'objet d'un article de ROTH en 1991 – voir [Rot91]. Ce dernier montra que de tels codes pouvaient avoir des applications pratiques, notamment concernant le stockage de données sur bandes magnétiques. Quoiqu'il ne mentionnât pas les travaux de GABIDULIN, l'algorithme de décodage qu'il proposa est similaire à celui donné dans l'article [Gab91]. Le point de vue de ROTH et ses orientations sont néanmoins radicalement différentes, notamment en ce qui concerne la généralisation des codes corrigeant des *tableaux d'erreurs* [Rot96, Rot97], et ne sera pas développé dans la suite du chapitre.

Le lecteur voudra bien m'excuser d'avoir étalé dans cette partie un arsenal théorique matriciel, assez technique mais indispensable à une bonne compréhension des problèmes gravitant autour de la *métrique rang*. J'ai tenté de suivre un fil conducteur en mettant en exergue, les similarités et les différences entre la *métrique de HAMMING* et la *métrique rang*.

Dans un premier temps, je présente les définitions fondamentales liées à la métrique rang, puis les propriétés immédiates qui en découlent, comme l'existence d'une borne de SINGLETON. Dans une seconde partie, je présente une famille de codes optimaux, atteignant la *borne de SINGLETON* appelés *codes de GABIDULIN*. Je décris leur algorithme de décodage en temps polynomial qui est calqué sur l'algorithme de décodage des *codes de REED–SOLOMON généralisés*. Tandis que les deux premières parties sont des reformulations de plusieurs articles, j'introduis dans une troisième partie un travail que j'ai réalisé avec E. M. GABIDULIN et qui concerne la structure des *sous-codes traces* de *codes de GABIDULIN*. Je montre que, contrairement au cas des *sous-codes traces* de codes GRS – tels les *codes de GOPPA*, les *sous-codes*

traces de codes de GABIDULIN ont la même structure que les codes dont ils dérivent. En effet, ils sont équivalents d'une certaine manière à des produits directs de *codes de GABIDULIN*.

1.1 Métrique rang : définitions et propriétés

On se place dorénavant dans l'espace des mots de longueur n sur le corps fini à q^m éléments noté $\mathbf{F}_{q^m}$. Le corps $\mathbf{F}_q$ où q est une puissance d'un nombre premier est appelé *corps de base*. On fixe pour la suite une base quelconque $\gamma_1, \gamma_2, \dots, \gamma_m$ de $\mathbf{F}_{q^m}$ sur $\mathbf{F}_q$. Tout élément b de $\mathbf{F}_{q^m}$ s'écrit alors de manière unique sous la forme

$$b = \sum_{i=1}^m b_i \gamma_i,$$

où les b_j sont des éléments du *corps de base* $\mathbf{F}_q$. Un vecteur $\mathbf{b}$ de longueur n sur $\mathbf{F}_{q^m}$ est donc en bijection avec l'ensemble des matrices $m \times n$ modulo cette représentation. On peut donc définir une norme sur $\underbrace{\mathbf{F}_{q^m} \times \mathbf{F}_{q^m} \times \dots \times \mathbf{F}_{q^m}}_{n \text{ fois}}$

de la façon suivante :

Définition 20 (Norme rang) Soit $\mathbf{a} = (a_1, a_2, \dots, a_n)$ un mot de longueur n sur $\mathbf{F}_{q^m}$. Soit $\mathbf{A}$ la matrice à m lignes et n colonnes de l'extension des coordonnées a_i de $\mathbf{a}$ sur la base $\gamma_1, \gamma_2, \dots, \gamma_m$. $\mathbf{A}$ est donnée par

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1m} & a_{2m} & \cdots & a_{nm} \end{pmatrix}$$

où $a_j = \sum_{i=1}^m a_{ij} \gamma_i$. On définit la norme rang - ou plus simplement le rang - du mot $\mathbf{a}$ - notée $\text{Rg}(\mathbf{a})$ - comme étant le rang de la matrice $\mathbf{A}$, i.e.

$$\text{Rg}(\mathbf{a}) \equiv \text{Rang}(\mathbf{A}).$$

Cette définition satisfait toutes les conditions requises pour être une norme de l'espace $\mathbf{F}_{q^m}^n$. En effet, comme l'application qui associe à une matrice $m \times n$ son rang est une norme de l'espace des matrices $m \times n$, on en déduit que la *norme rang* est une norme de l'espace $\mathbf{F}_{q^m}^n$ au sens usuel du terme. En conséquence on peut lui associer une distance appelée *distance rang*.

Définition 21 (Distance rang) Étant donnés deux mots $\mathbf{a}$ et $\mathbf{b}$ de longueur n sur $\mathbf{F}_{q^m}$, l'application d_{rg} définie par

$$d_{\text{rg}}(\mathbf{a}, \mathbf{b}) = \text{Rg}(\mathbf{a} - \mathbf{b})$$

est une distance sur $\mathbf{F}_{q^m}^n$. Elle définit la métrique rang et s'appelle distance rang.

La *métrique rang* d'un mot est reliée à la *métrique de HAMMING* de ce même mot par la proposition suivante :

Proposition 21 (Finesse de la métrique rang) *Soit $\mathbf{a}$ un mot de longueur n sur $\mathbf{F}_{q^m}$. Alors si $\text{wt}(\mathbf{a})$ désigne le poids de HAMMING du mot $\mathbf{a}$, on a*

$$\text{Rg}(\mathbf{a}) \leq \text{wt}(\mathbf{a}).$$

Preuve

Le *poids* du mot $\mathbf{a}$ noté $\text{wt}(\mathbf{a})$ où $\mathbf{a} = (a_1, a_2, \dots, a_n)$ désigne par définition le nombre de coordonnées non nulles a_i . En conséquence, la matrice $\mathbf{A}$ du développement de $\mathbf{a}$ sur une base $\gamma_1, \gamma_2, \dots, \gamma_m$ de $\mathbf{F}_{q^m}$ sur $\mathbf{F}_q$ a au plus $\text{wt}(\mathbf{a})$ colonnes non nulles. Le rang de $\mathbf{a}$ est donc inférieur à $\text{wt}(\mathbf{a})$.

■

La *métrique rang* est donc plus grossière que la *métrique de HAMMING*. En effet, d'après la proposition 21, à taille donnée, les boules de la *métrique rang* contiennent plus de mots que les boules de la *métrique de HAMMING*.

1.1.1 Distance rang minimale d'un code

Un code sur $\mathbf{F}_{q^m}$ étant par définition un ensemble de mots de longueur n sur $\mathbf{F}_{q^m}$, on peut lui associer sa *distance rang minimale* comme étant la valeur minimale des rangs des mots qui le constituent.

Définition 22 (Distance rang minimale d'un code)

Étant donné C , un code de longueur n , la distance rang minimale de C est la quantité d_{rg} définie par

$$d_{\text{rg}} = \text{Min}_{\mathbf{a} \in C} (\text{Rg}(\mathbf{a})).$$

Dans le cas des codes linéaires, on dispose d'un théorème donnant une condition nécessaire et suffisante sur un code linéaire pour que sa distance minimale soit égale à une valeur donnée d ,

Théorème 6 ([MS77]) *Soit C un code linéaire sur $\mathbf{F}_{q^m}$, de longueur n , de dimension k , de distance minimale d . Soit $\mathbf{H}$ une matrice de parité de C . Alors la distance minimale de C est égale à d si et seulement si*

1. pour tout mot $\mathbf{y}$ de longueur n sur $\mathbf{F}_{q^m}$ et de poids $\leq d - 1$

$$\mathbf{y}\mathbf{H}^T \neq 0,$$

2. il existe un mot $\mathbf{y}_0$ de poids d de longueur n sur $\mathbf{F}_{q^m}$ tel que

$$\mathbf{y}_0\mathbf{H}^T = 0.$$

Dans le cas de la *métrique rang*, on a un théorème similaire

Théorème 7 ([Gab85b]) Soit C un code linéaire sur $\mathbf{F}_{q^m}$, de longueur n , de dimension k , de distance minimale d . Soit $\mathbf{H}$ une matrice de parité de C . Alors la distance rang minimale de C est égale à d si et seulement si

1. pour toute matrice $\mathbf{Y}$ de taille $(d-1) \times n$, à coefficients $\mathbf{F}_q$ et de rang $d-1$, on a

$$\text{Rang}(\mathbf{YH}^T) = d-1, \quad (4.1)$$

2. il existe une matrice $\mathbf{Y}_0$ de taille $d \times n$, à coefficients dans $\mathbf{F}_q$ et de rang d qui vérifie

$$\text{Rang}(\mathbf{Y}_0\mathbf{H}^T) < d. \quad (4.2)$$

La propriété (4.1) indique que la *distance rang minimale* du code est supérieure ou égale à d , tandis que la propriété (4.2) indique qu'il existe un mot de rang d dans le code.

Preuve

Condition nécessaire : La preuve est scindée en deux étapes

1. Soit C un code sur $\mathbf{F}_{q^m}$, de matrice de parité $\mathbf{H}$, de *distance rang minimale* d . Donc, il existe au moins un mot $\mathbf{a}_0 \in C$, de rang d . $\mathbf{a}_0$ est représentable sous la forme

$$\mathbf{a}_0 = \mathbf{zY}_0$$

où $\mathbf{z} = (z_1, z_2, \dots, z_d)$ est un vecteur de longueur d à coefficients dans $\mathbf{F}_{q^m}$, tandis que $\mathbf{Y}_0$ est une matrice de taille $d \times n$ à coefficients dans $\mathbf{F}_q$ et de rang d . Comme $\mathbf{H}$ est une matrice de parité de C ,

$$\mathbf{a}_0\mathbf{H}^T = \mathbf{zY}_0\mathbf{H}^T = 0.$$

Par conséquent, le rang de la matrice $\mathbf{Y}_0\mathbf{H}^T$ est strictement inférieur à d , ce qui prouve (4.2).

2. Comme C ne contient pas de mot de rang inférieur à d , pour toute matrice $\mathbf{Y}$ de taille $(d-1) \times n$ et de rang $d-1$, l'équation

$$(z_1, z_2, \dots, z_{d-1})\mathbf{YH}^T = 0$$

aux inconnues $z_1, z_2, \dots, z_{d-1}$ n'a comme solution que le vecteur nul $(\underbrace{0, 0, \dots, 0}_{d-1 \text{ fois}})$ comme solution. Donc $\text{Rang}(\mathbf{YH}^T) \geq d-1$ et

comme $\text{Rang}(\mathbf{Y}) = d-1$, on a $\text{Rang}(\mathbf{YH}^T) = d-1$, ce qui prouve (4.1).

Condition suffisante : L'équation (4.1) montre qu'il n'y a pas de mot de rang inférieur à $d-1$ dans le code C , tandis que l'équation (4.2) montre qu'il existe un mot de rang d dans C . Donc la distance rang minimale du code est égale à d .

■

Ce théorème ne permet cependant pas d'obtenir facilement une borne de type GILBERT-VARSHAMOV contrairement au cas de la *métrique de HAMMING* – voir [MS77, p. 33]. Il semble en effet difficile de construire des *matrices de parité* de codes qui satisfont la première condition du théorème 7.

1.1.2 Borne de SINGLETON

La *distance rang minimale* d'un code présente beaucoup d'analogies avec la *distance minimale* d'un code. Par des considérations simples, le fait que la *métrique rang* soit plus grossière que la *métrique de HAMMING*, comme montré dans la proposition 21, donne une borne immédiate. En effet, étant donné C un code linéaire de *distance minimale* d et de *distance rang minimale* d_{rg} , comme $d \leq n - k + 1$ et comme, d'après la proposition 21, $d_{\text{rg}} \leq d$ on a également

$$d_{\text{rg}} \leq n - k + 1. \quad (4.3)$$

Dans le cas où les codes considérés sont des *codes groupes*, la borne peut être affinée. Cette nouvelle borne fait intervenir en particulier le degré m de l'extension $\mathbf{F}_{q^m}/\mathbf{F}_q$.

J'introduis ici la notion de *code groupe* qui généralise la notion de code linéaire.

Définition 23 (Code groupe) *Un code groupe de longueur n et de taille $\mathcal{N}$ sur un alphabet $\mathbf{F}_{q^m}$ est un ensemble de $\mathcal{N}$ mots*

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

de longueur n sur $\mathbf{F}_{q^m}$ qui forme un groupe additif.

Les *codes groupes* sont des ensembles de mots sur $\mathbf{F}_{q^m}$ stables par addition, mais pas forcément stables par multiplication par un élément de $\mathbf{F}_{q^m}$. De la même façon que pour les codes linéaires on peut définir la *distance rang minimale* d'un code groupe. Alors on a

Théorème 8 (Borne de SINGLETON) *Soit C un code groupe de longueur n sur $\mathbf{F}_{q^m}$ de taille $\mathcal{N}$, et de distance rang minimale d_{rg} , alors on a*

$$\mathcal{N} \leq q^{(m-d_{\text{rg}}+1)n}.$$

Preuve

Soit C un *code groupe* sur $\mathbf{F}_{q^m}$, de longueur n , de taille $\mathcal{N}$. Considérons deux mots $\mathbf{a} = (a_1, a_2, \dots, a_n)$ et $\mathbf{b} = (b_1, b_2, \dots, b_n)$ de C , ainsi que leurs représentation matricielle définie dans la définition 20 qui sont les matrices $\mathbf{A}$ et $\mathbf{B}$ de taille $m \times n$.

Comme C est un groupe, le mot $\mathbf{a} - \mathbf{b}$ de représentation matricielle $\mathbf{A} - \mathbf{B}$ appartient à C et est donc de rang supérieur ou égal à d_{rg} .

Soit $\mathbf{A}_1$ et $\mathbf{B}_1$ les matrices $(m - d_{\text{rg}} + 1) \times n$ constituées respectivement des $m - d_{\text{rg}} + 1$ premières lignes des matrices $\mathbf{A}$ et $\mathbf{B}$ respectivement *i.e.*

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \end{pmatrix}.$$

On montre que $\mathbf{A}_1 \neq \mathbf{B}_1$. Si on avait $\mathbf{A}_1 = \mathbf{B}_1$, alors on aurait également

$$\mathbf{A} - \mathbf{B} = \begin{pmatrix} 0 \\ \mathbf{A}_2 - \mathbf{B}_2 \end{pmatrix}$$

qui est de rang $\leq d_{\text{rg}} - 1$. Donc $\mathbf{A}_1 \neq \mathbf{B}_1$.

De ceci on déduit que deux mots d'un code C de longueur n et de distance rang minimale d_{rg} ont nécessairement les $m - d_{\text{rg}} + 1$ premières lignes de leurs représentations matricielles différentes. Le nombre $\mathcal{N}$ de mots du code C est donc au plus égal au nombre de matrices de taille $(m - d_{\text{rg}} + 1) \times n$ à coefficients dans $\mathbf{F}_q$, soit

$$\mathcal{N} \leq q^{(m - d_{\text{rg}} + 1)n}.$$

■

Pour le cas des codes linéaires, comme $\mathcal{N} = q^{km}$ où k est la dimension on obtient :

Corollaire 7 *Soit C un code linéaire sur $\mathbf{F}_{q^m}$ de longueur n , de dimension k de distance rang minimale d_{rg} . Alors on a*

$$km \leq (m - d_{\text{rg}} + 1)n. \quad (4.4)$$

L'inégalité du théorème 7 permet une généralisation de la borne de SINGLETON classique. En effet, si on pose $m = \lambda n$, l'inégalité (4.4) devient alors

$$d_{\text{rg}} \leq \lambda(n - k) + 1. \quad (4.5)$$

On constate que si $\lambda \geq 1$, c'est-à-dire si le degré m de l'extension de corps est supérieur à la longueur n du code, la borne de SINGLETON classique (4.3) est meilleure, tandis que si la longueur du code est supérieure au degré de l'extension, alors la borne (4.4) est meilleure. Ce qui nous amène naturellement à scinder en deux les cas où l'on recherche des codes linéaires qui sont optimaux pour la *distance rang* que l'on appelle *codes MRD* pour *Maximum Rank Distance*.

Définition 24 (Codes MRD) *Un code linéaire C de longueur n sur $\mathbf{F}_{q^m}$, de dimension k et de distance rang minimale d_{rg} est appelé code MRD - pour Maximum Rank Distance, s'il vérifie une des deux conditions suivantes :*

1. Si $n \leq m$, alors $d_{\text{rg}} = n - k + 1$.

2. Si $n > m$, alors $d_{\text{rg}} = \lfloor m(1 - k/n) + 1 \rfloor$.

Comme les paramètres d'un code sont des entiers, il est clair que si la quantité $k(m/n)$ n'est pas entière, la quantité $m(1 - k/n) + 1$ n'est pas entière non plus. Chercher un code dont les paramètres vérifient l'inégalité (4.4) revient par conséquent à chercher un code vérifiant $d_{\text{rg}} = \lfloor m(1 - k/n) + 1 \rfloor$.

Corollaire 8 (Codes MRD et codes MDS) *On se place dans l'espace des mots de longueur n sur le corps $\mathbf{F}_{q^m}$.*

- Si $n \leq m$, alors tout code MRD est un code MDS.

Preuve

Soit C un code MRD sur $\mathbf{F}_{q^m}$ de longueur n où $n \leq m$, de dimension k , et de distance rang minimale d_{rg} . Par définition des codes MRD on a

$$d_{\text{rg}} = n - k + 1.$$

De plus, d'après la proposition 21, la *métrique rang* est plus grossière que la *métrique de HAMMING*, donc $d_{\text{rg}} \leq d$ où d est la *distance minimale* de C . Comme $d \leq n - k + 1$, on en déduit

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

■

Les bornes données par la définition 24 des codes MRD sont les meilleures possibles. Dans la prochaine section je montre que pour tout couple de paramètres (n, d_{rg}) où $n \leq m$, il existe au moins un *code MRD* sur $\mathbf{F}_{q^m}$, de longueur n et de distance rang minimale d_{rg} .

Cependant, quand la longueur n du code dépasse le degré m de l'extension, la question générale de l'existence de *codes MRD* demeure ouverte. Dans certains cas on peut tout de même y répondre. Pour cela on construit des codes MRD comme des produits directs de codes MRD de longueur $n \leq m$.

Proposition 22 *Pour tout $n = mr$, et pour tout entier $1 \leq d_{\text{rg}} \leq m$, il existe un code MRD sur $\mathbf{F}_{q^m}$ de longueur n et de dimension $r(m - d_{\text{rg}} + 1)$.*

Preuve

Soit C_1 un code MRD de longueur $n = m$, et de *distance rang minimale* d_{rg} . Le code

$$C = \underbrace{C_1 \times C_1 \times \cdots \times C_1}_{r \text{ fois}}$$

est un code de longueur $n = mr$, de *distance rang* minimale d_{rg} et de dimension $k = r(m - d_{\text{rg}} + 1)$. Donc C est un code MRD.

■

Dans la *métrique de HAMMING*, la contrainte de l'égalité mise sur les codes MDS conditionne de multiples propriétés – voir [MS77, Ch. 11]. Tous les *codes MDS* ont par exemple la même distribution des poids, qui est parfaitement connue. De même, tous les *codes MRD* de longueur $n \leq m$ ont la même *distribution des rangs*.

Proposition 23 (Distribution des rangs) *Soit $A_s(n, d_{\text{rg}})$ le nombre de mots de rang s dans un code MRD sur $\mathbf{F}_{q^m}$ de longueur $n \leq m$ et de distance rang minimale d_{rg} , alors*

$$A_{d_{\text{rg}}+i}(n, d_{\text{rg}}) = \begin{bmatrix} n \\ d_{\text{rg}}+i \end{bmatrix} \sum_{j=0}^i (-1)^{i+j} \begin{bmatrix} d_{\text{rg}}+i \\ d_{\text{rg}}+j \end{bmatrix} q^{(i-j)(i-j-1)/2} (q^{m(j+1)} - 1),$$

où $\begin{bmatrix} t \\ s \end{bmatrix}$ désigne le nombre de sous-espaces vectoriels sur $\mathbf{F}_q$ de dimension s d'un espace vectoriel sur $\mathbf{F}_q$ de dimension $t \geq s$, i.e.

$$\begin{bmatrix} t \\ s \end{bmatrix} = \frac{(q^t - 1)(q^t - q^{t-1}) \cdots (q^t - q^{s-1})}{(q^s - 1)(q^s - q) \cdots (q^s - q^{s-1})}.$$

La preuve très technique est détaillée dans [Gab85b].

1.1.3 Décodage de la métrique rang

À l'instar de la métrique de HAMMING pour laquelle on cherche des familles de codes permettant de résoudre en temps polynomial le problème du *décodage borné* d'un code – le lecteur se reportera à la section 1.1 du chapitre 1 –, l'intérêt de la *métrique rang* consiste à trouver des familles de codes qui permettent de résoudre en temps polynomial le problème du *décodage rang borné*.

Problème 11 (Problème du décodage rang borné par t)

Étant donné

- un code linéaire de longueur n , de dimension k donné par une matrice génératrice $\mathbf{G}$,
- un mot $\mathbf{y}$ de longueur n ,

trouver s'il existe un couple $(\mathbf{x}, \mathbf{e})$ tel que

$$\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e},$$

où $\mathbf{e}$ est un vecteur de rang t .

Considérons le vecteur

$$\mathbf{y} = \mathbf{x}\mathbf{G} + \mathbf{e},$$

où $\mathbf{e}$ est un mot de poids de HAMMING t , et où $\mathbf{G}$ est une matrice génératrice d'un code C . Comme le poids de $\mathbf{e}$ est inférieur à t , le rang de $\mathbf{e}$ est également

inférieur à t . Donc, si on sait résoudre le problème 11 pour C , alors on sait résoudre le problème du *décodage rang borné par t* de C . Le décodage d'erreurs rangs est donc un problème plus difficile que le décodage classique – présenté dans la section 1.1 du chapitre 1.

Pour une métrique donnée sur la famille des codes, je rappelle la notion d'algorithme de décodage à vraisemblance maximale :

Définition 25 (Décodage à vraisemblance maximale) *Étant donné un code C de longueur n sur $\mathbf{F}_q^m$, un algorithme de décodage à vraisemblance maximale de C est un algorithme qui prend en entrée un mot $\mathbf{y}$ de longueur n sur $\mathbf{F}_q^m$ et retourne un mot du code C , le plus proche possible de $\mathbf{y}$ au sens de la métrique considérée.*

Définition 26 (Capacité de correction) *Soit C un code de longueur n , de distance rang minimale d_{rg} . Alors la quantité*

$$\left\lfloor \frac{d_{\text{rg}} - 1}{2} \right\rfloor$$

est appelée capacité de correction de C .

Dans le cas de la métrique rang, on a le résultat suivant identique à celui de la métrique de HAMMING, expliquant pourquoi on emploie le terme de *capacité de correction* d'un code.

Théorème 9 (Capacité de correction d'un code) *Un algorithme de décodage à vraisemblance maximale d'un code C de longueur n , de distance rang minimale d_{rg} peut corriger de manière unique jusqu'à $\lfloor (d_{\text{rg}} - 1)/2 \rfloor$ erreurs.*

Preuve

L'ensemble des boules de rayon $\lfloor (d_{\text{rg}} - 1)/2 \rfloor$ et de centre les mots de C sont deux à deux disjointes, et $\lfloor (d_{\text{rg}} - 1)/2 \rfloor$ est le plus grand rayon possible pour lequel on peut en être sûr.

■

1.2 Codes de GABIDULIN

L'étude des propriétés de la métrique rang a démontré que, quand la longueur n d'un code sur $\mathbf{F}_q^m$ est inférieure au degré de l'extension m , alors il existe une *borne de SINGLETON* que vérifient les paramètres du code. À savoir, la distance rang minimale d_{rg} et la dimension k d'un code, satisfont l'inégalité

$$d_{\text{rg}} - 1 \leq n - k.$$

Les codes vérifiant l'égalité $d_{\text{rg}} - 1 = n - k$ sont appelés *codes MRD* pour *Maximum Rank Distance*. En 1985, GABIDULIN a montré comment construire

une famille de codes MRD sur $\mathbf{F}_{q^m}$ de longueur $n \leq m$ et qui possèdent un algorithme polynomial de décodage rang borné jusqu'à la *capacité de correction* des codes – voir [Gab85b]. Cet algorithme est calqué sur l'algorithme de décodage des *codes de REED–SOLOMON généralisés*.

1.2.1 Définition et propriétés

Afin d'éviter une surcharge de notations je pose $[i] = q^i$. Ainsi, l'application $z \mapsto z^{[i]}$ est un automorphisme de $\mathbf{F}_{q^m}$ dans lui-même qui laisse $\mathbf{F}_q$ invariant. J'utilise également la notation d plutôt que d_{rg} pour désigner la distance rang minimale d'un code.

Soit $\mathbf{h} = (h_1, h_2, \dots, h_n)$ une suite d'éléments de $\mathbf{F}_{q^m}$ linéairement indépendants sur $\mathbf{F}_q$ – ce qui implique entre autres que $n \leq m$ – et soit un entier $2 \leq d \leq n$, considérons le code C sur $\mathbf{F}_{q^m}$ de longueur n et de matrice de parité

$$\mathbf{H} = \begin{pmatrix} h_1 & h_2 & \cdots & h_n \\ h_1^{[1]} & h_2^{[1]} & \cdots & h_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{[d-2]} & h_2^{[d-2]} & \cdots & h_n^{[d-2]} \end{pmatrix} \quad (4.6)$$

où $[i] = q^i$, *i.e.*

$$C = \{\mathbf{a} \in \mathbf{F}_{q^m}^n \mid \mathbf{a}\mathbf{H}^T = 0\}$$

Le paramètre d introduit ne doit rien au hasard. On va montrer qu'il correspond à la distance rang minimale du code C . Ceci prouve ainsi l'existence de *codes MRD* pour une longueur n donnée, inférieure à m , et pour une distance rang minimale d donnée.

Proposition 24 *Le code C de matrice de parité $\mathbf{H}$ est un code MRD de longueur n , et de distance rang minimale d .*

Preuve

Je me sers du théorème 7. Soit $\mathbf{Y}$ une matrice de taille $(d-1) \times n$, à coefficients dans $\mathbf{F}_q$ un sous-corps de $\mathbf{F}_{q^m}$, et de rang $d-1$. On a

$$\mathbf{H}\mathbf{Y}^T = \begin{pmatrix} f_1 & f_2 & \cdots & f_{d-1} \\ f_1^{[1]} & f_2^{[1]} & \cdots & f_{d-1}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ f_1^{[d-2]} & f_2^{[d-2]} & \cdots & f_{d-1}^{[d-2]} \end{pmatrix}$$

où $(f_1, f_2, \dots, f_{d-1}) = (h_1, h_2, \dots, h_n)\mathbf{Y}^T$.

Comme les h_i sont par hypothèse linéairement indépendants sur $\mathbf{F}_q$ et comme $\mathbf{Y}$ est de rang maximum sur $\mathbf{F}_q$, les f_i sont également linéairement indépendants sur $\mathbf{F}_q$. La matrice $\mathbf{H}\mathbf{Y}^T$ est donc de rang $d-1$.

Ceci prouve la condition (4.1) du théorème 7, stipulant que la distance rang minimale du code est au moins égale à $d = n - k + 1$.

Or, l'inégalité de SINGLETON (4.3) affirme que la distance rang minimale est inférieure ou égale à $n - k + 1$. Donc d correspond bien à la distance rang minimale de C . Partant, C est un code MRD.

■

Théorème 10 *Le dual du code C de matrice de parité $\mathbf{H} = \left(h_j^{[i]} \right)_{i=0, j=1}^{d-2, n}$ est le code de longueur n , de dimension $k = n - d + 1$ et de matrice de parité*

$$\mathbf{G} = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^{[1]} & g_2^{[1]} & \cdots & g_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{[k-1]} & g_2^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}$$

où $g_i = \lambda_i^{[-k+1]}$, pour des éléments $(\lambda_i)_{i=1}^n$ de $\mathbf{F}_{q^m}$ solutions des équations

$$\sum_{i=1}^n \lambda_i h_i^{[j]} = 0$$

pour $j = 0, 1, \dots, n - 2$.

La preuve de ce théorème quoique simple fait appel à une jolie petite astuce.

Preuve

Soit C' , le code sur $\mathbf{F}_{q^m}$ de matrice de parité

$$\mathbf{H}' = \begin{pmatrix} h_1 & h_2 & \cdots & h_n \\ h_1^{[1]} & h_2^{[1]} & \cdots & h_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{[n-2]} & h_2^{[n-2]} & \cdots & h_n^{[n-2]} \end{pmatrix}.$$

Soit $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n) \in C'$, non nul. λ vérifie les équations de parité suivantes

$$\sum_{i=1}^n \lambda_i h_i^{[j]} = 0, \quad (4.7)$$

pour $j = 0, 1, \dots, n - 2$. D'après la proposition 24, C' est un code MRD. Sa distance rang minimale est donc égale à n . Par conséquent, C' est un code dont tous les mots ont le même rang égal à n . Donc

$$\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n) \in C'$$

est de rang n . Donc, les éléments $\lambda_1, \lambda_2, \dots, \lambda_n$ forment une famille libre sur $\mathbf{F}_q$. Posons alors $\mathbf{g} = (g_1, g_2, \dots, g_n)$ avec $g_i = \lambda_i^{[-k+1]}$. Comme

l'élevation à la puissance $[i] = q^i$ est un automorphisme de $\mathbf{F}_{q^m}$ laissant $\mathbf{F}_q$ invariant, les éléments $g_1, g_2, \dots, g_n$ forment également une famille libre sur $\mathbf{F}_q$. Soit $\mathbf{G}$ la matrice

$$\mathbf{G} = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^{[1]} & g_2^{[1]} & \cdots & g_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{[k-1]} & g_2^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}.$$

En utilisant les équations de parité (4.7) on déduit que

$$\mathbf{G}\mathbf{H}^T = 0.$$

Ce qui implique que $\mathbf{G}$ est une matrice génératrice du code C .

■

Si on considère une matrice de parité d'un code donnée sous la forme très particulière (4.6), alors le théorème implique qu'il existe une matrice génératrice de ce code ayant la même forme. Cette remarque m'amène à définir la famille des codes de GABIDULIN.

Définition 27 (Codes de GABIDULIN) Soit $\mathbf{g} = (g_1, g_2, \dots, g_n)$ des éléments de $\mathbf{F}_{q^m}$ linéairement indépendants sur $\mathbf{F}_q$ – ce qui implique entre autres que $n \leq m$. Alors le code de matrice génératrice

$$\mathbf{G} = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^{[1]} & g_2^{[1]} & \cdots & g_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{[k-1]} & g_2^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}$$

où $[i] = q^i$ est un code MRD appelé code de GABIDULIN, de dimension k , engendré par $\mathbf{g} = (g_1, g_2, \dots, g_n)$. Il est noté $\text{Gab}(\mathbf{g}, k)$.

Codes de GABIDULIN et polynômes linéarisés :

Le lecteur averti aura sans doute été frappé par l'étonnante similarité de construction entre certains codes GRS présentés dans le chapitre 1 section 2.1, qui forment une sous-famille des codes MDS et les codes de GABIDULIN qui constituent une sous-famille de codes MRD. En effet, étant donné $\mathbf{g} = (g_1, g_2, \dots, g_n)$, le code de matrice génératrice

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ g_1 & g_2 & \cdots & g_n \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{k-1} & g_2^{k-1} & \cdots & g_n^{k-1} \end{pmatrix}$$

est un code GRS et est de plus égal à l'ensemble des mots

$$\mathbf{a} = (P(g_1), P(g_2), \dots, P(g_n)),$$

où P décrit l'ensemble des polynômes à coefficients dans $\mathbf{F}_{q^m}$ de degré strictement inférieur à k .

La relation entre les codes de GABIDULIN et ces codes GRS tient aux propriétés communes aux algèbres «classiques» de polynômes et aux algèbres de *polynômes linéarisés*.

Définition 28 (Polynômes linéarisés) *Étant donné un corps $\mathbf{F}_{q^m}$, on appelle polynôme linéarisé de degré t sur $\mathbf{F}_{q^m}$, un polynôme*

$$P(x) = \sum_{i=0}^t p_i x^{[i]}$$

où les $p_i \in \mathbf{F}_{q^m}$, et où $[i] = q^i$.

Avec cette définition, un *code de GABIDULIN* de longueur n et de dimension k , engendré par $\mathbf{g} = (g_1, g_2, \dots, g_n)$, est l'ensemble des mots

$$\mathbf{a} = (P(g_1), P(g_2), \dots, P(g_n)),$$

où P décrit l'ensemble des *polynômes linéarisés* sur $\mathbf{F}_{q^m}$, de degrés strictement inférieurs à k .

La théorie des *polynômes linéarisés* est détaillée dans [Ore33, Ore34] et dans le livre de BERLEKAMP [Ber84, Ch. 11]. Les quelques propriétés suivantes sont utiles pour la suite et permettent d'appréhender les différences et les similitudes entre les deux algèbres de polynômes.

Propriétés 6

1. *Toute application linéaire de $\mathbf{F}_{q^m}$ dans lui-même s'exprime de manière unique sous la forme d'un polynôme linéarisé sur $\mathbf{F}_{q^m}$ de degré $t < m$.*
2. *L'ensemble des polynômes linéarisés sur $\mathbf{F}_{q^m}$ est une algèbre non-commutative lorsqu'on le munit de l'addition terme à terme et de la composition classique de polynômes i.e. si P et Q sont des polynômes linéarisés sur $\mathbf{F}_{q^m}$, alors*
 - $(P + Q)(x) = P(x) + Q(x)$ *est un polynôme linéarisé sur $\mathbf{F}_{q^m}$,*
 - $(P \circ Q)(x) = P(Q(x))$ *est un polynôme linéarisé sur $\mathbf{F}_{q^m}$.*
3. *L'ensemble des racines d'un polynôme linéarisé sur $\mathbf{F}_{q^m}$ de degré t est un espace vectoriel sur $\mathbf{F}_q$.*
4. *Il existe un algorithme d'EUCLIDE dans l'algèbre des polynômes linéarisés, i.e. étant donnés P et S , deux polynômes linéarisés sur $\mathbf{F}_{q^m}$, il existe un unique couple (Q, R) de polynômes linéarisés sur $\mathbf{F}_{q^m}$ tel que*

$$\begin{cases} P(x) = (Q \circ S)(x) + R(x), \\ \deg(R) < \deg(S). \end{cases}$$

Le décodage en temps polynomial des *codes de REED–SOLOMON généralisés* repose sur l'existence d'un algorithme d'EUCLIDE étendu dans l'algèbre des polynômes. Partant, l'existence d'un algorithme d'EUCLIDE dans l'algèbre des polynômes linéarisés laisse entrevoir l'existence d'un algorithme de décodage en temps polynomial des *codes de GABIDULIN*.

1.2.2 Algorithme de décodage

GABIDULIN a présenté plusieurs versions d'algorithmes de décodage dans les articles [Gab85b, Gab91]. La première version de l'algorithme ressemble trait pour trait à l'algorithme d'EUCLIDE étendu qui permet de décoder les *codes de REED–SOLOMON généralisés*. Il suffit de remplacer l'algèbre classique des polynômes par l'algèbre des *polynômes linéarisés*, puis d'utiliser le fait qu'il existe un algorithme d'EUCLIDE pour cette algèbre, et le tour est joué !

Dans cette partie, je ne décris pas la version de la plus efficace en termes de complexité en temps. Je présente celle qui, à mon avis, permet le mieux de cerner le problème du décodage des codes de GABIDULIN. Il s'agit d'une version duale de l'algorithme qui est présentée dans [Gab91], et fait appel à une méthode qui ressemble à celle utilisée dans l'algorithme de BERLEKAMP–MASSEY sans toutefois en avoir l'efficacité calculatoire.

Étant donné des éléments $(h_1, h_2, \dots, h_n) \in \mathbf{F}_{q^m}^n$ linéairement indépendants sur $\mathbf{F}_q$, on considère le code de GABIDULIN C , de matrice de parité

$$\mathbf{H} = \left(h_j^{[i]} \right)_{i=0, j=1}^{d-2, n}.$$

C est un code de longueur n et de distance rang minimale d , d'après la proposition 24. Supposons que, par un canal de transmission, on reçoive le vecteur $\mathbf{y}$ de longueur n sur $\mathbf{F}_{q^m}$ tel que

$$\mathbf{y} = \mathbf{c} + \mathbf{e},$$

où $\mathbf{c} \in C$ et où $\mathbf{e}$ est un vecteur d'erreur de rang $t \leq \lfloor (d-1)/2 \rfloor$.

D'un point de vue théorique, on peut décoder le vecteur $\mathbf{y}$ reçu en utilisant un algorithme de décodage à vraisemblance maximale, puisque le rang de $\mathbf{e}$ est inférieur à la capacité de correction du code C – voir la section 1.1.3. L'algorithme de décodage proposé par GABIDULIN permet de résoudre le problème suivant en temps polynomial :

Problème 12 (Décodage d'un code de GABIDULIN) *Étant donné*

$$\mathbf{s} = (s_0, s_1, \dots, s_{d-2}) = \mathbf{y}\mathbf{H}^T,$$

déterminer l'unique vecteur $\mathbf{e} \in \mathbf{F}_{q^m}^n$ tel que

$$\begin{cases} \mathbf{e}\mathbf{H}^T = \mathbf{s}, \\ \text{Rg}(\mathbf{e}) \leq \lfloor \frac{d-1}{2} \rfloor. \end{cases} \quad (4.8)$$

Je commence par découper le problème en trois étapes correspondant chacune à une réduction du problème 12. Le décodage s'effectuera en commençant par la dernière étape puis en remontant. Le lecteur constatera que l'algorithme de décodage proposé est très similaire à celui donné par CONAN et LOELOELIAN dans le contexte des *codes de GOPPA* – voir [CL87].

Réduction 1 :

La première étape consiste à transformer l'équation (4.8) en un système équivalent. Étant donné que le vecteur d'erreur $\mathbf{e}$ est de longueur n et de rang t , il existe un vecteur $\epsilon = (\epsilon_1, \epsilon_2, \dots, \epsilon_t)$ dans $\mathbf{F}_{q^m}$ de rang t et une matrice $\mathbf{Y} = (Y_{ij})$ de taille $t \times n$, à coefficients dans $\mathbf{F}_q$ et de rang t tels que

$$\mathbf{e} = \epsilon \mathbf{Y}. \quad (4.9)$$

Remarque 3 (Famille de solutions) *On peut compter le nombre de solutions $(\epsilon, \mathbf{Y})$ du système $\mathbf{e} = \epsilon \mathbf{Y}$. En effet, comme $\mathbf{Y}$ et ϵ sont de rang maximal, si on a une solution $(\epsilon_0, \mathbf{Y}_0)$ de l'équation (4.8), alors l'ensemble des solutions est égal à*

$$(\epsilon_0 \mathbf{S}, \mathbf{S}^{-1} \mathbf{Y}_0)$$

où $\mathbf{S}$ décrit l'ensemble des matrices inversibles de taille $t \times t$, à coefficients dans $\mathbf{F}_q$. Le nombre de solutions $(\epsilon, \mathbf{Y})$ de l'équation

$$\mathbf{e} = \epsilon \mathbf{Y}$$

est donc égal au nombre de matrices $t \times t$ inversibles à coefficient dans $\mathbf{F}_q$, c'est-à-dire égal à

$$(q^t - 1)(q^t - q) \cdots (q^t - q^{t-1}).$$

Dans les faits, je montrerai que cette remarque est prise en compte dans les réductions successives.

Résoudre (4.8) est alors équivalent à trouver une solution $(\epsilon, \mathbf{Y})$ de l'équation

$$\mathbf{s} = \epsilon \mathbf{Y} \mathbf{H}^T. \quad (4.10)$$

Soit $\mathbf{X} = \mathbf{Y} \mathbf{H}^T$. Comme $\mathbf{H} = (h_j^{[i]})$, et comme la matrice $\mathbf{Y} = (Y_{ij})$ est à coefficients dans $\mathbf{F}_q$, on a

$$\mathbf{X} = \begin{pmatrix} x_1 & x_1^{[1]} & \cdots & x_1^{[d-2]} \\ x_2 & x_2^{[1]} & \cdots & x_2^{[d-2]} \\ \vdots & \vdots & \ddots & \vdots \\ x_t & x_t^{[1]} & \cdots & x_t^{[d-2]} \end{pmatrix},$$

où

$$x_i = \sum_{j=1}^n Y_{ij} h_j. \quad (4.11)$$

Comme $\mathbf{Y}$ est de rang t sur $\mathbf{F}_q$, les x_i sont linéairement indépendants sur $\mathbf{F}_q$. Résoudre l'équation (4.10) est donc équivalent à trouver une solution du système d'équations suivant dont les inconnues sont les éléments $\epsilon_1, \epsilon_2, \dots, \epsilon_t$, et $x_1, x_2, \dots, x_t$ de $\mathbf{F}_{q^m}$:

$$\sum_{i=1}^t \epsilon_i x_i^{[p]} = s_p, \quad (4.12)$$

pour $p = 1, 2, \dots, d-1$. Considérons alors le problème suivant

Problème 13 (Décodage réduit) *Étant donnée $(h_1, h_2, \dots, h_n) \in \mathbf{F}_{q^m}^n$ une famille libre sur $\mathbf{F}_q$, et un vecteur $\mathbf{s} = (s_1, s_2, \dots, s_{d-1}) \in \mathbf{F}_{q^m}^{d-1}$, déterminer une famille $(x_i)_{i=1}^t \in \mathbf{F}_{q^m}^t$ libre sur $\mathbf{F}_q$, telle que, si on note $\mathbf{X} = (x_i^{[j]})$, et $\mathbf{H} = (h_i^{[j]})$, alors*

$$\begin{cases} \mathbf{X} = \mathbf{YH}^T, \text{ où } \mathbf{Y} \in \mathbf{F}_q^{t \times n}, \\ \sum_{i=1}^t \epsilon_i x_i^{[u]} = s_u, \text{ pour } u = 1, 2, \dots, d-1. \end{cases}$$

À partir de la solution du problème 13, on résout le problème 12 du décodage en temps polynomial. En effet, considérons $x_1, x_2, \dots, x_t$, des éléments solution du problème 13.

1. Comme les x_i sont libres sur $\mathbf{F}_q$, la matrice $\mathbf{X}$ est de rang maximal. On détermine alors l'unique vecteur $\epsilon = (\epsilon_1, \dots, \epsilon_t) \in \mathbf{F}_{q^m}^t$ solution de l'équation

$$\mathbf{s} = \epsilon \mathbf{X},$$

où $\mathbf{X} = (x_i^{[j]})$, et où $\mathbf{s} = (s_1, \dots, s_{d-1})$ est connu.

2. De plus comme les éléments h_j forment une famille libre, l'équation $\mathbf{X} = \mathbf{YH}^T$ a une solution unique $\mathbf{Y}$ qu'on peut déterminer en temps polynomial.
3. Connaissant ϵ et $\mathbf{Y}$, on détermine le mot

$$\mathbf{e} = \epsilon \mathbf{Y},$$

solution du problème 12 de décodage.

Le but des autres étapes de réduction va consister à trouver des éléments $x_1, x_2, \dots, x_t$ solution du problème 13.

Réduction 2 – Détermination des x_i solutions du problème 13 :

Les éléments $x_1, x_2, \dots, x_t$, forment une famille libre sur $\mathbf{F}_q$. En conséquence, le polynôme linéarisé de degré t sur $\mathbf{F}_{q^m}$, dont l'espace des racines est l'espace vectoriel sur $\mathbf{F}_q$ engendré par $x_1, x_2, \dots, x_t$ s'écrit :

$$\sigma(z) = \sum_{i=0}^t \sigma_i z^{[i]}, \quad \sigma_t = 1.$$

Si on connaît σ , déterminer une base de l'espace vectoriel des solutions s'effectue par de simples opérations matricielles dans $\mathbf{F}_q$.

En effet, soit $\gamma_1, \gamma_2, \dots, \gamma_m$ une base de $\mathbf{F}_{q^m}$ sur $\mathbf{F}_q$, et soit b une racine de σ qui se décompose en $b = \sum_{j=1}^m b_j \gamma_j$. Comme $z \mapsto z^{[i]}$ est un automorphisme de $\mathbf{F}_{q^m}$ considéré comme espace vectoriel sur $\mathbf{F}_q$, on a aussi $b^{[i]} = \sum_{j=1}^m b_j \gamma_j^{[i]}$. D'où

$$\begin{aligned} \sigma(b) = 0 &\iff \sum_{i=0}^t \sigma_i \sum_{j=1}^m b_j \gamma_j^{[i]} = 0, \\ &\iff \sum_{j=1}^m b_j \sigma(\gamma_j) = 0. \end{aligned}$$

Donc, les coordonnées b_i dans $\mathbf{F}_q$ d'une racine b de σ vérifient l'équation

$$(b_1, b_2, \dots, b_m) \begin{pmatrix} \sigma(\gamma_1) \\ \sigma(\gamma_2) \\ \vdots \\ \sigma(\gamma_m) \end{pmatrix} = 0. \quad (4.13)$$

Trouver une base $x_1, x_2, \dots, x_t$ des racines de σ s'effectue donc en $O(m^2)$ multiplications dans $\mathbf{F}_q$. Comme l'espace des solutions que l'on recherche est stable par changement de base, n'importe quel ensemble de solutions du système est une solution du problème 13.

Réduction 3 – Évaluation des coefficients de σ

Ici, je montre qu'on peut trouver un système d'équations permettant de déterminer le polynôme σ de la réduction précédente.

Théorème 11 *Soit σ le polynôme linéarisé unitaire de degré t sur $\mathbf{F}_{q^m}$ dont l'ensemble des racines est l'espace vectoriel sur $\mathbf{F}_q$ engendré par les $(x_i)_{i=1}^t$ solutions de l'équation (4.11). Si*

$$\sigma(z) = \sum_{i=0}^t \sigma_i z^{[i]}, \quad \sigma_t = 1,$$

alors

$$s_{j+t}^{[-j]} + \sigma_{t-1} s_{j+t-1}^{[-j]} + \dots + \sigma_0 s_j^{[-j]} = 0, \quad (4.14)$$

pour $j = 0, 1, \dots, d - 2 - t$.

Preuve

La preuve découle directement d'une réécriture du système (4.12). En effet, si on multiplie la j ème équation par $\sigma_0^{[j]}$, la $(j+1)$ ème par $\sigma_1^{[j]}$, ...,

la $(j+t)$ ème par $\sigma_t^{[j]} = 1$, en faisant la somme des équations obtenues, on obtient

$$\sum_{i=0}^t \epsilon_i \sum_{u=0}^t \sigma_k^{[j]} x_i^{[u+j]} = s_{j+t} + \sigma_{t-1}^{[j]} s_{j+t-1} + \cdots + \sigma_0^{[j]} s_j. \quad (4.15)$$

Comme, par construction, les x_i sont des racines de σ , pour tout j entier, on a

$$\sum_{u=0}^t \sigma_k^{[j]} x_i^{[u+j]} = \sigma(x_i)^{[j]} = 0.$$

Le membre de gauche de l'équation (4.15) est nul et l'équation s'écrit

$$s_{j+t} + \sigma_{t-1}^{[j]} s_{j+t-1} + \cdots + \sigma_0^{[j]} s_j = 0.$$

En élevant chacune des équations à la puissance $[-j]$, on obtient finalement

$$s_{j+t}^{[-j]} + \sigma_{t-1}^{[-j]} s_{j+t-1}^{[-j]} + \cdots + \sigma_0 s_j^{[-j]} = 0,$$

pour $j = 0, 1, \dots, d-2-t$. Ce qui achève la preuve et peut être le lecteur ?

■

Soit $\mathbf{M}_i$, les matrices carrées de taille $i \times i$ définies par

$$\mathbf{M}_i = \begin{pmatrix} s_0 & s_1^{[-1]} & \cdots & s_{i-1}^{[-i+1]} \\ s_1 & s_2^{[-1]} & \cdots & s_i^{[-i+1]} \\ \vdots & \vdots & \ddots & \vdots \\ s_{i-1} & s_i^{[-1]} & \cdots & s_{2i-2}^{[-i+1]} \end{pmatrix},$$

pour $i = 1, \dots, t$. Le système d'équations (4.14) vérifié par les coefficients σ_i de σ est équivalent au système d'équations

$$(\sigma_0, \sigma_1, \dots, \sigma_{t-1}) \mathbf{M}_t = - (s_t, s_{t+1}^{[-1]}, \dots, s_{2t-1}^{[-t+1]}). \quad (4.16)$$

Si on connaît *a priori* le rang t du vecteur d'erreur $\mathbf{e}$, résoudre le système d'équations (4.14) revient à retrouver $\mathbf{e}$ et donc à décoder. En effet, le système (4.16) donne les coefficients de σ . Puis la réduction 2 donne les éléments x_i , solutions du problème 14, en fonction des σ_i .

Cependant, dans le cas général où les vecteurs d'erreur sont aléatoires, le décodeur ne connaît pas *a priori* leur rang. La seule information exploitable est que ce rang est borné par la capacité de correction du code, soit par $\lfloor (d-1)/2 \rfloor$. La proposition suivante permet de déterminer le rang de l'erreur si celui-ci est inférieur à la capacité de correction du code.

Proposition 25 (Détermination du rang de l'erreur) *Si le rang du vecteur d'erreur $\mathbf{e}$ est $t \leq \lfloor (d-1)/2 \rfloor$ alors la matrice $\mathbf{M}_i$ est inversible si $i = t$ et est non inversible si $i > t$.*

Preuve

En combinant le système d'équation (4.12) et l'expression de la matrice $\mathbf{M}_i$ en fonction du syndrome, pour $i \leq \lfloor (d-1)/2 \rfloor$, les x_j et les ϵ_j que l'on cherche vérifient le système d'équations suivant

$$\mathbf{M}_i = \begin{pmatrix} x_1 & x_2 & \cdots & x_i \\ x_1^{[1]} & x_2^{[1]} & \cdots & x_i^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{[i-1]} & x_2^{[i-1]} & \cdots & x_i^{[i-1]} \end{pmatrix} \begin{pmatrix} \epsilon_1 & \epsilon_1^{[-1]} & \cdots & \epsilon_1^{[-i+1]} \\ \epsilon_2 & \epsilon_2^{[-1]} & \cdots & \epsilon_2^{[-i+1]} \\ \vdots & \vdots & \ddots & \vdots \\ \epsilon_i & \epsilon_i^{[-1]} & \cdots & \epsilon_i^{[-i+1]} \end{pmatrix}. \quad (4.17)$$

Désignons par t le rang du mot $\mathbf{e}$ solution du problème de décodage

- Si $i = t$: les deux matrices du système (4.17) sont inversibles. En effet, comme par définition le rang de $\epsilon = (\epsilon_1, \dots, \epsilon_t)$ est égal à t , les ϵ_i sont linéairement indépendants sur $\mathbf{F}_q$. De la même façon, les x_i sont des combinaisons linéaires d'éléments linéairement indépendants sur $\mathbf{F}_q$, donc les x_i sont linéairement indépendants. Partant, $\mathbf{M}_i$ est inversible
- Si $i > t$: les deux matrices du système ne sont pas inversibles car elles contiennent une ligne et une colonne de 0 respectivement. Donc $\mathbf{M}_i$ n'est pas inversible.

■

La proposition 25 donne naturellement la marche à suivre pour décoder. On suppose d'abord que le rang du vecteur d'erreur est égal à $t = \lfloor (d-1)/2 \rfloor$. Si la matrice $\mathbf{M}_t$ est non inversible, alors on diminue t de façon incrémentale jusqu'à ce que $\mathbf{M}_t$ soit inversible. La procédure à suivre est résumée Tableau 4.1.

Proposition 26 *L'algorithme du tableau 4.1 permet de corriger jusqu'à*

$$t_{\max} = \lfloor (d-1)/2 \rfloor$$

erreurs dans un code de GABIDULIN de distance rang minimale d donné par une matrice de parité $\mathbf{H} = (h_i^{[j]})$.

Sa complexité est égale à $O(mt_{\max}^3 + mnt_{\max})$ multiplications dans $\mathbf{F}_q$.

Preuve

Par construction, l'algorithme résout le problème 13 dont la résolution donne une solution du problème 12 en temps polynomial.

En ce qui concerne la complexité de l'algorithme, celui-ci n'effectue que des résolutions de systèmes linéaires dans le corps $\mathbf{F}_{q^m}$. Or, l'efficacité du calcul dans les corps finis – surtout de la multiplication – dépend beaucoup de l'implémentation [Men93, Shp92]. Je donne donc

l'évaluation en nombres de multiplications dans le corps $\mathbf{F}_q$ en considérant qu'une multiplication dans $\mathbf{F}_{q^m}$ coûte $O(m)$ multiplications dans $\mathbf{F}_q$. C'est le cas lors d'une implantation matérielle de la multiplication [Men93, Ch. 4].

1. Calcul de $\mathbf{s} = \mathbf{y}\mathbf{H}^T$: $O(mnt_{\max})$ multiplications dans $\mathbf{F}_q$.
2. Trouver t tel que $\mathbf{M}_t$ soit inversible, puis inverser la matrice $\mathbf{M}_t$ pour déterminer le polynôme linéarisé σ : $O(mt_{\max}^3)$ multiplications dans $\mathbf{F}_q$.

Cependant, en considérant la structure de la matrice $\mathbf{M}_t$, il devrait être possible d'éviter la complexité cubique d'inversion d'une matrice en copiant un algorithme du type BERLEKAMP–MASSEY comme le fait remarquer ROTH dans [Rot91]. En effet, l'algorithme de BERLEKAMP–MASSEY permet en particulier d'inverser les matrices du type

$$\begin{pmatrix} s_0 & s_1 & \cdots & s_{i-1} \\ s_1 & s_2 & \cdots & s_i \\ \vdots & \vdots & \ddots & \vdots \\ s_{i-1} & s_i & \cdots & s_{2i-2} \end{pmatrix}$$

avec une complexité de $O(i^2)$ multiplication dans le corps contenant les s_i . Mais je ne connais pas d'algorithme permettant d'inverser «intelligemment» les matrices $\mathbf{M}_i$. C'est pourquoi, je considère que la complexité de l'inversion de la matrice $\mathbf{M}_i$ demeure égale à i^3 multiplications dans $\mathbf{F}_{q^m}$.

3. Calcul d'une base de l'espace des racines de σ : cette étape correspond à la réduction 2. La résolution du système (4.13) se fait en $O(m^2)$ multiplications dans $\mathbf{F}_q$. C'est en général négligeable comparé aux complexités des autres étapes.
4. Déterminer le vecteur ϵ tel que

$$\mathbf{s} = \epsilon\mathbf{X},$$

où $\mathbf{X}$ est une matrice $t \times (d-1)$ à coefficients dans $\mathbf{F}_{q^m}$. Cela coûte $O(mt^2)$ multiplications dans $\mathbf{F}_q$.

La complexité de l'algorithme est donc $O(mt_{\max}^3 + nt_{\max})$ multiplications dans $\mathbf{F}_q$.

■

1.3 Sous-codes traces des codes de GABIDULIN

L'étude de la structure des *sous-codes traces* est le résultat d'un travail commun avec E. M. GABIDULIN – voir [GL00]. Cette étude fut initiée par

Données :

1. La matrice de parité $\mathbf{H} = \left(h_j^{[i]} \right)_{i=0, j=1}^{d-2, n}$ d'un code de GABIDULIN C .
2. Un mot $\mathbf{y} = (y_1, y_2, \dots, y_n)$ tel que $\mathbf{y} = \mathbf{c} + \mathbf{e}$, où $\mathbf{c} \in C$, et où $\mathbf{e} = (e_1, e_2, \dots, e_n)$ est un vecteur d'erreur de longueur n sur $\mathbf{F}_{q^m}$ de rang $t \leq \lfloor (d-1)/2 \rfloor$.

Résultat : le vecteur $\mathbf{e}$.

Procédure :

1. Calculer $\mathbf{s} = (s_0, s_1, \dots, s_{d-2})$ en utilisant la formule :

$$\mathbf{s} = \mathbf{y}\mathbf{H}^T.$$

2. Répéter

$$\mathbf{M}_t = \begin{pmatrix} s_0 & s_1^{[-1]} & \cdots & s_{t-1}^{[-t+1]} \\ s_1 & s_2^{[-1]} & \cdots & s_t^{[-t+1]} \\ \vdots & \vdots & \ddots & \vdots \\ s_{t-1} & s_t^{[-1]} & \cdots & s_{2t-2}^{[-t+1]} \end{pmatrix}.$$

Si $\mathbf{M}_t$ est non inversible alors $t = t - 1$,
jusqu'à ce que $\mathbf{M}_t$ soit inversible.

3. Trouver l'unique vecteur $\sigma = (\sigma_0, \sigma_1, \dots, \sigma_t)$ tel que

$$\sigma\mathbf{M}_t = - \left(s_t, s_{t+1}^{[-1]}, \dots, s_{2t+1}^{[-t+1]} \right).$$

4. Déterminer une base des solutions $x_1, x_2, \dots, x_t$ de l'équation

$$\sum_{i=0}^t \sigma_i z^{[i]} = 0.$$

5. Déterminer un vecteur $\epsilon = (\epsilon_1, \epsilon_2, \dots, \epsilon_t)$ tel que

$$\mathbf{s} = \epsilon\mathbf{X}, \text{ avec } \mathbf{X} = \left(x_i^{[j]} \right).$$

6. Trouver la matrice $\mathbf{Y}$ telle que $\mathbf{X} = \mathbf{Y}\mathbf{H}^T$.
7. Retourner $\mathbf{e} = \epsilon\mathbf{Y}$.

TABLEAU 4.1 : Algorithme de décodage des codes de GABIDULIN

des considérations cryptologiques en regard du système de chiffrement à *clé publique* GPT, qui utilise l'algorithme de décodage des *codes de GABIDULIN*. On a vu dans la section 2.1.2 du chapitre 1 que le cryptosystème de MCELIECE utilisant des *codes de REED–SOLOMON généralisés* comme espace des clés pouvait être cryptanalysé, tandis que les systèmes utilisant des codes de GOPPA résistent aux *attaques structurelles*. Cette propriété semble due au fait que les codes de GOPPA sont des *sous-codes traces* de *codes de REED–SOLOMON généralisés*. Cependant, comme ils sont des sous-codes des codes GRS, le même algorithme polynomial de décodage peut leur être appliqué. Partant, il semble naturel de se demander comment agit l'*opérateur trace* sur les *codes de GABIDULIN* qui sont les équivalents des *codes GRS* pour la métrique rang.

Je montre que les *sous-codes traces* de codes de GABIDULIN restent très structurés. Ils sont équivalents à des produits directs de *codes de GABIDULIN* pris sur des sous-corps. Ce sont donc des *codes MRD*.

Dans toute la section, le degré de l'extension m s'écrit

$$m = s\bar{s},$$

i.e. $\mathbf{F}_{q^s}$ et $\mathbf{F}_{q^{\bar{s}}}$ sont des sous-corps de $\mathbf{F}_{q^m}$. On se fixe également une base quelconque $\gamma_1, \gamma_2, \dots, \gamma_{\bar{s}}$ de $\mathbf{F}_{q^m}/\mathbf{F}_{q^s}$.

Je considère l'application $\mathcal{P}$ qui à tout élément $a \in \mathbf{F}_{q^m}$ associe sa décomposition sur la base choisie,

$$\mathcal{P} : a = \sum_{i=1}^{\bar{s}} a_i \gamma_i \mapsto \mathcal{P}(a) = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{\bar{s}} \end{pmatrix},$$

où les a_i sont des éléments de $\mathbf{F}_{q^s}$.

Cette projection s'étend naturellement aux vecteurs dont les coordonnées sont dans $\mathbf{F}_{q^m}$ en appliquant la décomposition coordonnée par coordonnée. Si $\mathbf{h} = (h_1, h_2, \dots, h_n)$ est un vecteur de longueur n dans $\mathbf{F}_{q^m}$, on écrit

$$\mathcal{P}(\mathbf{h}) = (\mathcal{P}(h_1), \mathcal{P}(h_2), \dots, \mathcal{P}(h_n)).$$

Soit C le *code de GABIDULIN* de distance rang minimale d , et dont une matrice de parité est

$$\mathbf{H} = \begin{pmatrix} h_1 & h_2 & \cdots & h_n \\ h_1^{[1]} & h_2^{[1]} & \cdots & h_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{[d-2]} & h_2^{[d-2]} & \cdots & h_n^{[d-2]} \end{pmatrix}. \quad (4.18)$$

Le sous-code trace de C sur le sous-corps $\mathbf{F}_{q^s}$ est l'ensemble des mots $\mathbf{a} = (a_1, a_2, \dots, a_n) \in C$ dont toutes les coordonnées a_i sont dans $\mathbf{F}_{q^s}$. Pour

obtenir une matrice de parité du sous-code trace, il suffit donc de développer chaque ligne de $\mathbf{H}$ relativement à la base $\gamma_1, \dots, \gamma_{\bar{s}}$, ce qui nous donne

$$\mathbf{H}_{\mathbf{F}_{q^s}} = \begin{pmatrix} \mathcal{P}(h_1) & \mathcal{P}(h_2) & \cdots & \mathcal{P}(h_n) \\ \mathcal{P}(h_1^{[1]}) & \mathcal{P}(h_2^{[1]}) & \cdots & \mathcal{P}(h_n^{[1]}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{P}(h_1^{[d-2]}) & \mathcal{P}(h_2^{[d-2]}) & \cdots & \mathcal{P}(h_n^{[d-2]}) \end{pmatrix} \quad (4.19)$$

qui est la matrice de taille $\bar{s}(d-1) \times n$, à coefficients dans $\mathbf{F}_{q^s}$ des projections de chacun des vecteurs constituant la matrice $\mathbf{H}$ sur le sous-corps $\mathbf{F}_{q^s}$. Le code de matrice de parité $\mathbf{H}_{\mathbf{F}_{q^s}}$ est noté $(C \mid \mathbf{F}_{q^s})$. On en déduit

– Si k désigne la dimension de $(C \mid \mathbf{F}_{q^s})$, alors

$$k \geq n - \bar{s}(d-1).$$

– Comme $(C \mid \mathbf{F}_{q^s})$ est un sous-code de C , sa *distance rang minimale* est inférieure ou égale à d .

Hormis ces propriétés évidentes, la structure du *dual* des sous-codes traces fut déterminée entièrement par DELSARTE dans [Del75b]. Cependant, en raison de la construction particulière des *codes de GABIDULIN*, on peut déduire bien plus d'information sur leurs sous-codes traces.

Théorème 12 (Structure des sous-codes trace) *Soit C un code de GABIDULIN sur $\mathbf{F}_{q^m}$, dont une matrice de parité est $\mathbf{H} = (h_j^{[i]})_{i=0, j=1}^{d-2, n}$. Si la projection des coordonnées de $\mathbf{h} = (h_1, h_2, \dots, h_n)$ sur la base $\gamma_1, \dots, \gamma_{\bar{s}}$ s'écrit*

$$\mathcal{P}(\mathbf{h}) = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\bar{s}1} & h_{\bar{s}2} & \cdots & h_{\bar{s}n} \end{pmatrix},$$

alors la matrice

$$\mathbf{H}'_{\mathbf{F}_{q^s}} = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\bar{s}1} & h_{\bar{s}2} & \cdots & h_{\bar{s}n} \\ h_{11}^{[1]} & h_{12}^{[1]} & \cdots & h_{1n}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\bar{s}1}^{[1]} & h_{\bar{s}2}^{[1]} & \cdots & h_{\bar{s}n}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_{11}^{[d-2]} & h_{12}^{[d-2]} & \cdots & h_{1n}^{[d-2]} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\bar{s}1}^{[d-2]} & h_{\bar{s}2}^{[d-2]} & \cdots & h_{\bar{s}n}^{[d-2]} \end{pmatrix} \quad (4.20)$$

est une matrice de parité de $(C \mid \mathbf{F}_{q^s})$.

Preuve

Soit C un code de matrice de parité $\mathbf{H} = \left(h_j^{[i]} \right)_{i=0, j=1}^{d-2, n}$. Soit

$$\mathcal{P}(\mathbf{h}) = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\bar{s}1} & h_{\bar{s}2} & \cdots & h_{\bar{s}n} \end{pmatrix},$$

la matrice de décomposition du mot $\mathbf{h} = (h_1, h_2, \dots, h_n)$ sur la base de $\mathbf{F}_{q^m}/\mathbf{F}_q$, $\gamma_1, \gamma_2, \dots, \gamma_{\bar{s}}$.

Prouver le théorème équivalent à démontrer que la matrice

$$\mathcal{P}(\mathbf{h}^{[i]}) = \left(\mathcal{P}(h_1^{[i]}), \mathcal{P}(h_2^{[i]}), \dots, \mathcal{P}(h_n^{[i]}) \right)$$

qui constitue l'extension sur la base $\gamma_1, \gamma_2, \dots, \gamma_{\bar{s}}$ de la i ème ligne de $\mathbf{H}$, et la matrice

$$\mathbf{H}_i = \begin{pmatrix} h_{11}^{[i]} & h_{12}^{[i]} & \cdots & h_{1n}^{[i]} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\bar{s}1}^{[i]} & h_{\bar{s}2}^{[i]} & \cdots & h_{\bar{s}n}^{[i]} \end{pmatrix}$$

engendrent le même espace vectoriel sur $\mathbf{F}_{q^s}$, puisque

$$\mathbf{H}'_{\mathbf{F}_{q^s}} = \begin{pmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_{\bar{s}} \end{pmatrix}, \quad \text{et} \quad \mathbf{H}_s = \begin{pmatrix} \mathcal{P}(h_1) & \cdots & \mathcal{P}(h_n) \\ \vdots & \ddots & \vdots \\ \mathcal{P}(h_1^{[d-2]}) & \cdots & \mathcal{P}(h_n^{[d-2]}) \end{pmatrix}.$$

Comme l'application $z \mapsto z^{[i]}$ est un automorphisme de $\mathbf{F}_{q^m}$ laissant le sous-corps $\mathbf{F}_{q^s}$ invariant, la famille d'éléments $\gamma_1^{[i]}, \gamma_2^{[i]}, \dots, \gamma_{\bar{s}}^{[i]}$ est une base de $\mathbf{F}_{q^m}/\mathbf{F}_{q^s}$ pour tout i . Il existe donc une matrice inversible de taille $\bar{s} \times \bar{s}$ de changement de base à coefficients dans $\mathbf{F}_{q^s}$, notée $\mathbf{Q}_i$, telle que

$$\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{\bar{s}} \end{pmatrix} = \mathbf{Q}_i \begin{pmatrix} \gamma_1^{[i]} \\ \gamma_2^{[i]} \\ \vdots \\ \gamma_{\bar{s}}^{[i]} \end{pmatrix}.$$

La j ème coordonnée h_j du vecteur $\mathbf{h}$ s'écrivant $h_j = \sum_{r=1}^{\bar{s}} h_{jr} \gamma_r$, on a également

$$h_j^{[i]} = \sum_{r=1}^{\bar{s}} h_{jr}^{[i]} \gamma_r^{[i]}.$$

Comme $\mathbf{Q}_i$ est la matrice de changement de base, on en déduit que les coordonnées de $h_j^{[i]}$ sur la base $\gamma_1, \gamma_2, \dots, \gamma_{\bar{s}}$ s'écrivent

$$\mathcal{P} \left(h_j^{[i]} \right) = \mathbf{Q}_i \begin{pmatrix} h_{1j}^{[i]} \\ h_{2j}^{[i]} \\ \vdots \\ h_{\bar{s}j}^{[i]} \end{pmatrix}.$$

La j ème ligne de la matrice de parité (4.19) devient

$$\mathcal{P} \left(\mathbf{h}^{[i]} \right) = \left(\mathcal{P} \left(h_1^{[i]} \right), \mathcal{P} \left(h_2^{[i]} \right), \dots, \mathcal{P} \left(h_n^{[i]} \right) \right) = \mathbf{Q}_i \mathbf{H}_i. \quad (4.21)$$

Comme $\mathbf{Q}_i$ est une matrice inversible à coefficients dans $\mathbf{F}_{q^s}$, les matrices $\mathcal{P} \left(\mathbf{h}^{[i]} \right)$ et $\mathbf{H}_i$ sont des matrices de parité du même code sur $\mathbf{F}_{q^s}$.
■

Corollaire 9 Soit C un code de GABIDULIN sur $\mathbf{F}_{q^m}$, de matrice de parité $\mathbf{H} = \left(h_j^{[i]} \right)_{i=0, j=1}^{d-2, n}$. Le sous-code trace $(C \mid \mathbf{F}_{q^s})$ est l'intersection des codes C_i , $i = 1, \dots, \bar{s}$ sur $\mathbf{F}_{q^s}$ de matrices de parité sont

$$\mathbf{H}'_i = \begin{pmatrix} h_{i1} & h_{i2} & \dots & h_{in} \\ h_{i1}^{[1]} & h_{i2}^{[1]} & \dots & h_{in}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_{i1}^{[d-2]} & h_{i2}^{[d-2]} & \dots & h_{in}^{[d-2]} \end{pmatrix}$$

pour $i = 1, \dots, \bar{s}$.

Permuter les lignes de la matrice (4.20) donne une autre matrice de parité du sous-code trace qui a la structure demandée. On pourrait être tenté de croire que le sous-code trace est l'intersection de $\bar{s}$ codes de GABIDULIN sur $\mathbf{F}_{q^s}$, étant donnée la structure des matrices $\mathbf{H}'_i$. Cependant, bien que $h_1, h_2, \dots, h_n$ soient linéairement indépendants sur $\mathbf{F}_q$, cela n'implique nullement que les éléments $h_{i1}, h_{i2}, \dots, h_{in}$ pour i variant de 1 à $\bar{s}$ soient indépendants. Les matrices $\mathbf{H}'_i$ n'ont donc aucune raison d'être des matrices de parité de codes de GABIDULIN.

Dans le cas particulier où $m = n$, on peut montrer beaucoup plus. En effet, les sous-codes trace ont alors la même *distribution des rang* que des produits directs de codes de GABIDULIN. D'après la proposition 22, ce sont donc des codes MRD.

Proposition 27 Soit C un code de GABIDULIN sur $\mathbf{F}_{q^m}$, de longueur m , de distance rang minimale d . Si $\mathbf{F}_{q^s}$ est un sous-corps de $\mathbf{F}_{q^m}$, alors le sous-code trace sur $(C \mid \mathbf{F}_{q^s})$ de C est équivalent au produit direct de $\bar{s} = m/s$ codes de GABIDULIN sur $\mathbf{F}_{q^s}$ de longueur s et de distance rang minimale d .

Avant de prouver la proposition, j'ai tout d'abord besoin du lemme suivant :

Lemme 3 Soit $\gamma_1, \gamma_2, \dots, \gamma_{\bar{s}}$ une base de $\mathbf{F}_{q^m}/\mathbf{F}_{q^s}$, et $\mathbf{h} = (h_1, h_2, \dots, h_m)$ un vecteur de longueur m et de rang m maximum sur $\mathbf{F}_q$.

Considérons la matrice

$$\mathcal{P}(\mathbf{h}) = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\bar{s}1} & h_{\bar{s}2} & \cdots & h_{\bar{s}m} \end{pmatrix}$$

qui est la matrice des coordonnées h_j de $\mathbf{h}$ sur la base γ_i . Alors il existe une matrice $\mathbf{S}$ inversible, à coefficients dans le corps de base $\mathbf{F}_q$ telle que

$$\mathbf{A} = \mathcal{P}(\mathbf{h})\mathbf{S},$$

avec

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1s} & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & a_{21} & \cdots & a_{2s} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & a_{\bar{s}1} & \cdots & a_{\bar{s}m} \end{pmatrix}$$

où $a_{ij} \in \mathbf{F}_{q^s}$.

Preuve du théorème

Supposons qu'il existe une matrice $\mathbf{S}$ ayant les propriétés du lemme. Comme $\mathbf{S}$ est à coefficients dans $\mathbf{F}_q$, on a

$$\mathbf{A}_i = \begin{pmatrix} a_{11}^{[i]} & \cdots & a_{1s}^{[i]} & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & a_{21}^{[i]} & \cdots & a_{2s}^{[i]} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & a_{\bar{s}1}^{[i]} & \cdots & a_{\bar{s}m}^{[i]} \end{pmatrix} = \mathbf{H}_i \mathbf{S},$$

où $\mathbf{H}_i = \left(h_{kj}^{[i]} \right)_{k=1, j=1}^{\bar{s}, m}$. Partant, on a également

$$\begin{pmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{d-2} \end{pmatrix} = \begin{pmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_{d-2} \end{pmatrix} \mathbf{S}.$$

En multipliant à gauche par une matrice de permutation $\mathbf{P}'$ ad hoc, on permute les lignes de la matrice. On obtient alors

$$\mathcal{A} = \begin{pmatrix} \mathbf{A}'_0 & 0 & \cdots & 0 \\ 0 & \mathbf{A}'_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{A}'_{d-2} \end{pmatrix} = \mathbf{P}' \begin{pmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_{d-2} \end{pmatrix} \mathbf{S},$$

où les matrices

$$\mathbf{A}'_i = \begin{pmatrix} a_{i1} & a_{i2} & \cdots & a_{is} \\ a_{i1}^{[1]} & a_{i2}^{[1]} & \cdots & a_{is}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1}^{[d-2]} & a_{i2}^{[d-2]} & \cdots & a_{is}^{[d-2]} \end{pmatrix}$$

sont des matrices de parité de *codes de GABIDULIN* de longueur s sur $\mathbf{F}_{q^s}$ et de distance rang minimale $d - 2$. Le code de matrice de parité $\mathcal{A}$ est produit direct de *codes de GABIDULIN*. Donc le *sous-code trace* $(C \mid \mathbf{F}_{q^s})$ est équivalent à un produit direct de *codes de GABIDULIN* de longueur inférieure à m . D'après la proposition 22 c'est donc un *code MRD*.

■

Preuve du lemme

Comme les éléments $h_1, \dots, h_m$ sont libres sur $\mathbf{F}_q$, il existe une matrices $\mathbf{S}'$ à coefficients dans $\mathbf{F}_q$ telle que

$$\bar{s} \left\{ \begin{pmatrix} a_{11} & \cdots & a_{1s} & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ * & \cdots & * & a_{21} & \cdots & a_{2s} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \ddots & \vdots \\ * & \cdots & * & * & \cdots & * & \cdots & a_{\bar{s}1} & \cdots & a_{\bar{s}s} \end{pmatrix} = \mathcal{P}(\mathbf{h})\mathbf{S}' \right.$$

où les familles $a_{i1}, a_{i2}, \dots, a_{is}$ sont libres sur $\mathbf{F}_q$ pour tout $i \leq \bar{s}$. Ces familles sont donc des bases de $\mathbf{F}_{q^s}$. Partant, par des combinaisons linéaires de colonnes, on supprime les termes $*$ de la matrice et on obtient la forme désirée.

■

2 Cryptosystèmes utilisant la métrique rang

Les études sur les propriétés de la métrique rang de la section précédente ont permis de construire un algorithme polynomial de décodage décodant les *codes de GABIDULIN* jusqu'à leur capacité de correction. Un tel algorithme s'applique également aux codes qui en sont dérivés. Cette propriété fut utilisée afin de concevoir un système de chiffrement à clé publique inspiré du système de chiffrement de MCELIECE, en utilisant des *codes de GABIDULIN* en lieu et place des codes de GOPPA. Sa sécurité repose entre autres sur la difficulté de résoudre le problème 11 du *décodage rang borné*. Grâce aux propriétés de la métrique rang ce cryptosystème résiste aux *attaques par décodages* décrites dans la section 1.2.1 du chapitre 1. En théorie, on peut donc se permettre d'utiliser des taille de clés de quelques milliers de bits seulement, comparés aux centaines de milliers de bits de clé nécessaires pour assurer la sécurité du cryptosystème de MCELIECE. Cependant, si la

taille de la *clé publique* n'est pas suffisante, des *attaques structurelles* très efficaces permettent de retrouver un décodeur du *code public*, diminuant ainsi l'avantage de ce cryptosystème sur le cryptosystème de McELIECE.

Dans cette section, dans un premier temps, je décris le système de chiffrement GPT qui utilise comme *espace des clés* les *codes de GABIDULIN*. Dans un deuxième temps, je présente les *attaques de GIBSON* dont la redoutable efficacité impose que l'on choisisse des tailles de clés relativement importantes. Dans une troisième partie, je montre que l'on peut pallier ce défaut en utilisant comme espace des clés une famille «bien choisie» de sous-codes de *codes de GABIDULIN*, dont on contrôle la structure.

2.1 Le cryptosystème GPT

Le système de chiffrement GPT du nom de ses concepteurs – GABIDULIN, PARAMONOV ET TRETJAKOV fut présenté en 1991 dans [GPT91]. Il utilise comme *clé publique* une matrice génératrice d'un *code de GABIDULIN* aléatoire masquée par l'ajout d'une *matrice de distorsion*. La sécurité du cryptosystème contre les *attaques structurelles* repose entièrement sur la structure de cette matrice particulière.

2.1.1 Description du cryptosystème

Le fonctionnement du système GPT est calqué sur celui du système de McELIECE décrit dans la section 1.1 du chapitre 1. Au lieu d'utiliser des codes disposant d'un algorithme de décodage pour la métrique de HAMMING on utilise les codes de GABIDULIN qui ont un algorithme de décodage pour la métrique rang. Malheureusement, comme ces codes sont très structurés, on doit briser leur structure en se servant d'une matrice spécifique appelée *matrice de distorsion*. Le principe général du système est décrit dans le tableau 4.2. Le code sur $\mathbf{F}_{q^m}$ de matrice génératrice $\mathbf{G}_{\text{pub}}$ est appelé *code public*.

Complexité du cryptosystème

- *Complexité en temps* : Si on néglige le coût de la génération du mot $\mathbf{e}$ de rang $t - t_1$, le coût du chiffrement est en moyenne de $nk/2$ multiplications dans $\mathbf{F}_{q^m}$. Le coût du déchiffrement d'un message est égal au coût d'un décodage d'un mot, auquel on ajoute le coût d'une inversion de matrice $k \times k$. Le coût du système de chiffrement est donc égal à $O(t^3 + nt + k^3)$ multiplications dans $\mathbf{F}_{q^m}$.
- *Taille de la clé publique* : tout élément de $\mathbf{F}_{q^m}$ est stocké sur $m \log_2(q)$ bits. La taille de la clé publique $\mathbf{G}_{\text{pub}}$ est donc $mnk \log_2(q)$ bits.

Exemple 15

Les auteurs ont proposé les paramètres suivants

$$q = 2, \quad m = n = 20, \quad k = 12, \quad t_1 = 1.$$

Comme les éléments de $\mathbf{F}_{2^{20}}$ sont codés sur 20 bits, la taille de la *clé publique* $\mathbf{G}_{\text{pub}}$ est égale à $20 \times 20 \times 12 = 4800$ bits.

Clé privée : le triplet $(\mathbf{G}, \mathbf{S}, \mathbf{X})$, où

– $\mathbf{G} = \left(g_j^{[i]} \right)_{j=1, i=0}^{n, k-1}$ est une matrice $k \times n$ génératrice d'un code de GABIDULIN sur $\mathbf{F}_{q^m}$, de longueur $n \leq m$, de capacité de correction construite $t = \lfloor (d-1)/2 \rfloor$.

– $\mathbf{S}$, une matrice inversible aléatoire de taille $k \times k$, à coefficients dans $\mathbf{F}_{q^m}$.

– $\mathbf{X}$ est une matrice $k \times n$ à coefficients dans $\mathbf{F}_{q^m}$ vérifiant :

Pour tout $\mathbf{c} = (c_1, c_2, \dots, c_k) \in \mathbf{F}_{q^m}^k$, $\text{Rg}(\mathbf{cX}) \leq t_1$ où $t_1 < t$.

$\mathbf{X}$ est appelée *matrice de distorsion* de paramètre t_1 .

Clé publique :

1. $\mathbf{G}_{\text{pub}} = \mathbf{S}\mathbf{G} + \mathbf{X}$.

2. Le paramètre de distorsion t_1 .

Les procédures de *chiffrement* et de *déchiffrement* sont les suivantes

– *Chiffrement* : soit $\mathbf{c}$ un mot de longueur k . L'émetteur calcule et envoie

$$\mathbf{y} = \mathbf{c}\mathbf{G}_{\text{pub}} + \mathbf{e},$$

où $\mathbf{e}$ est un vecteur d'erreur aléatoire de longueur n , et de rang $\leq t - t_1$.

– *Déchiffrement* :

comme $\mathbf{y} = \mathbf{c}\mathbf{S}\mathbf{G} + (\mathbf{cX} + \mathbf{e})$, le mot $(\mathbf{cX} + \mathbf{e})$ est de rang inférieur à t . En effet,

$$\begin{cases} \text{Rg}(\mathbf{cX}) & \leq t_1, \\ \text{Rg}(\mathbf{e}) & \leq t - t_1. \end{cases}$$

Comme le rang d'un mot est une norme, on a

$$\text{Rg}(\mathbf{cX} + \mathbf{e}) \leq \text{Rg}(\mathbf{cX}) + \text{Rg}(\mathbf{e}) \leq t.$$

En appliquant l'algorithme de décodage des codes de GABIDULIN, le receveur recouvre le mot $\mathbf{c}' = \mathbf{c}\mathbf{S}$. Puis, il calcule $\mathbf{c} = \mathbf{c}'\mathbf{S}^{-1}$.

TABLEAU 4.2 : Système de chiffrement GPT

Commentaires : Dans le cryptosystème de McELIECE, la *clé publique* est une matrice génératrice aléatoire du *code public* permuté. En revanche dans le système GPT, il serait superflu d'utiliser un code de GABIDULIN permuté, puisque ceux-ci sont stables par permutation. Plus généralement, toute multiplication à droite de la matrice $\mathbf{G} = \left(g_j^{[i]} \right)_{i=0, j=1}^{k-1, n}$ par une matrice $\mathbf{S} = (s_{ij})_{i=1, j=1}^{n, n}$ inversible, à coefficients dans $\mathbf{F}_q$ donne la matrice

$$\mathbf{GS} = \left(s_j^{[i]} \right)_{i=0, j=1}^{k-1, n}$$

où $s_j = \sum_{k=1}^n g_k s_{kj}$. $\mathbf{GS}$ est donc également une matrice génératrice d'un *code de GABIDULIN*.

2.1.2 Rôle de la matrice de distorsion

La nouveauté du système comparé au système de McELIECE est l'utilisation d'une *matrice de distorsion* $\mathbf{X}$ dont le rôle s'avère crucial. Celle-ci doit prévenir toute *attaque structurelle* consistant à recouvrer un décodeur du *code public*. La suite de cette section a pour but de montrer l'importance de la *matrice de distorsion* et d'en proposer des constructions possibles.

Supposons que la matrice de distorsion soit la matrice nulle $\mathbf{0}$, *i.e.* la *clé publique* s'écrit

$$\mathbf{G}_{\text{pub}} = \mathbf{SG}$$

où $\mathbf{S}$ est une matrice $k \times k$ inversible, à coefficients dans $\mathbf{F}_q^m$, et où

$$\mathbf{G} = \begin{pmatrix} g_1 & g_2 & \cdots & g_n \\ g_1^{[1]} & g_2^{[1]} & \cdots & g_n^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{[k-1]} & g_2^{[k-1]} & \cdots & g_n^{[k-1]} \end{pmatrix}.$$

Il est alors très facile de retrouver une matrice génératrice $\mathbf{G}'$ d'un code de GABIDULIN – pas forcément égale à $\mathbf{G}$ – ainsi qu'une matrice inversible $\mathbf{S}'$ vérifiant

$$\mathbf{G}_{\text{pub}} = \mathbf{S}'\mathbf{G}'.$$

À partir de là, on recouvre un décodeur de la *clé publique* en temps polynomial. Le principe de l'attaque suit les étapes suivantes :

1. On met la matrice $\mathbf{G}_{\text{pub}}$ sous forme systématique, en opérant des transformations sur les lignes. On obtient alors la matrice $k \times n$

$$(\mathbf{I} \mid \mathbf{R}) = \left(\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & R_{0,k+1} & R_{0,k+2} & \cdots & R_{0,n} \\ 0 & 1 & \cdots & 0 & R_{1,k+1} & R_{1,k+2} & \cdots & R_{1,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & R_{k-1,k+1} & R_{k-1,k+2} & \cdots & R_{k-1,n} \end{array} \right).$$

L'étape de réduction se fait en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.

2. Écrivons la matrice génératrice du code de GABIDULIN utilisée dans le système sous la forme

$$\mathbf{G} = (\mathbf{G}_1 \mid \mathbf{G}_2),$$

où $\mathbf{G}_1$ est la matrice carrée $k \times k$ constituée des k premières colonnes de $\mathbf{G}$, et $\mathbf{G}_2$ est la matrice $k \times (n - k)$ résiduelle.

$\mathbf{G}_1$ est une matrice inversible. En effet c'est une matrice génératrice d'un code de GABIDULIN de longueur k et de dimension k , et donc elle est de rang k . D'où, en factorisant par $\mathbf{G}_1$, on a

$$\mathbf{G} = \mathbf{G}_1 (\mathbf{I} \mid \mathbf{G}_1^{-1} \mathbf{G}_2)$$

Comme, à support fixé, la forme systématique d'une matrice est unique, $\mathbf{G}_1$, $\mathbf{G}_2$, et $\mathbf{R}$ vérifient l'équation suivante

$$\mathbf{G}_1 \mathbf{R} = \mathbf{G}_2. \quad (4.22)$$

Si de plus $\mathbf{G} = (g_j^{[i]})$, alors l'équation devient

$$g_1^{[i]} R_{0,j} + g_2^{[i]} R_{1,j} + \cdots + g_k^{[i]} R_{k-1,j} = g_j^{[i]}, \quad \begin{cases} i = 0, 1, \dots, k-1, \\ j = k+1, \dots, n. \end{cases} \quad (4.23)$$

C'est un système de $k(n - k)$ équations à n inconnues qui sont les coordonnées du vecteur $\mathbf{g} = (g_1, g_2, \dots, g_n)$. Bien que ce système soit hautement non-linéaire, on peut le relinéariser en élevant chacune de ses équations à la puissance $[-i] = q^i$. Le système est alors équivalent au système linéaire suivant

$$g_1 R_{0,j}^{[-i]} + g_2 R_{1,j}^{[-i]} + \cdots + g_k R_{k-1,j}^{[-i]} = g_j, \quad \begin{cases} i = 0, 1, \dots, k-1, \\ j = k+1, \dots, n, \end{cases}$$

de $k(n - k)$ équations à n inconnues très redondant. Si on pose

$$\mathbf{R}_j = \begin{pmatrix} R_{0,j}^{[0]} & R_{0,j}^{[-1]} & \cdots & R_{0,j}^{[1-k]} \\ R_{1,j}^{[0]} & R_{1,j}^{[-1]} & \cdots & R_{1,j}^{[1-k]} \\ \vdots & \vdots & \ddots & \vdots \\ R_{k-1,j}^{[0]} & R_{k-1,j}^{[-1]} & \cdots & R_{k-1,j}^{[1-k]} \end{pmatrix}$$

résoudre (4.23) est équivalent à résoudre le système

$$(g_1, g_2, \dots, g_k) \mathbf{R}_j = \underbrace{(g_j, g_j, \dots, g_j)}_{k \text{ fois}},$$

pour $j = k+1, \dots, n$, et avec les matrices $\mathbf{R}_j$ sont inversibles. De plus, comme GIBSON a montré qu'on pouvait sans perte de généralité fixer un des g_j à la valeur 1 – voir [Gib95b] – ce système est inversible en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.

En résumé

Proposition 28 (Importance de la matrice de distorsion) *Si la matrice de distorsion $\mathbf{X}$ de taille $k \times n$ d'un cryptosystème GPT est nulle, un attaquant recouvre un décodeur du code public en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.*

On vient d'établir l'importance de la matrice de distorsion. À présent il est naturel de se demander comment faire pour la construire.

Construction de la matrice de distorsion $\mathbf{X}$ La matrice de distorsion de taille $k \times n$, d'un cryptosystème GPT dont le *paramètre de distorsion* est t_1 , doit vérifier la propriété suivante :

$$\forall \mathbf{c} = (c_1, c_2, \dots, c_k) \in \mathbf{F}_{q^m}^k, \quad \text{Rg}(\mathbf{cX}) \leq t_1.$$

En pratique t_1 doit être inférieur à la capacité de correction du code, partant inférieur à la longueur n du code. GABIDULIN a présenté une telle construction dans [Gab95]. Ici, je propose une construction simplifiée qui lui est équivalente.

Proposition 29 (Construction de la matrice de distorsion)

Considérons

1. $\mathbf{A}' = (\underbrace{\mathbf{Q} \mid \mathbf{0}}_n)$ où $\mathbf{Q}$ est une matrice de taille $k \times t_1$ à coefficients dans $\mathbf{F}_{q^m}$ et où $\mathbf{0}$ est la matrice $k \times (n - t_1)$ remplie de 0. La matrice $\mathbf{Q}$ peut être par exemple tirée aléatoirement.
2. $\mathbf{B}'$ une matrice $n \times n$ inversible à coefficients dans $\mathbf{F}_q$.

Soit $\mathbf{X} = \mathbf{A}'\mathbf{B}'$, alors

1. la matrice $\mathbf{X}$ est une matrice $k \times n$ à coefficients dans $\mathbf{F}_{q^m}$ vérifiant

$$\forall \mathbf{c} = (c_1, c_2, \dots, c_k) \in \mathbf{F}_{q^m}^k, \quad \text{Rg}(\mathbf{cX}) \leq t_1$$

2. Le nombre de matrices ainsi construites est égal à

$$\prod_{i=0}^{t_1-1} \frac{(q^{mk} - q^{mi})}{(q^{mt_1} - q^{mi})} \cdot \prod_{i=0}^{n-1} (q^n - q^i).$$

Preuve

Soit $\mathbf{c}$ un vecteur quelconque de longueur k dans $\mathbf{F}_{q^m}$, on a

$$\mathbf{cA}' = (\mathbf{cQ} \mid \mathbf{0}).$$

Comme $\mathbf{cQ}$ est de longueur t_1 , le vecteur $\mathbf{cA}'$ a au plus t_1 coordonnées non nulles. Donc $\text{Rg}(\mathbf{cA}') \leq t_1$. Comme $\mathbf{B}'$ est une matrice à coefficients dans $\mathbf{F}_q$, la multiplication à droite par $\mathbf{B}'$ ne modifie pas le rang. Donc

$$\text{Rg}(\mathbf{cA}'\mathbf{B}') \leq t_1.$$

Le nombre d'espace vectoriels de dimension t_1 et de longueur k sur $\mathbf{F}_{q^m}$ est égal au *polynôme de GAUSS*

$$\begin{bmatrix} k \\ t_1 \end{bmatrix} = \prod_{i=0}^{t_1-1} \frac{(q^{mk} - q^{mi})}{(q^{mt_1} - q^{mi})}.$$

Multiplier à droite la matrice $\mathbf{A}'$ par une matrice $\mathbf{B}'$ inversible à coefficients dans $\mathbf{F}_q$ revient à faire des combinaisons linéaires sur $\mathbf{F}_q$ de colonnes de $\mathbf{A}'$. Il y a exactement

$$\prod_{i=0}^{n-1} (q^n - q^i)$$

matrices $\mathbf{B}'$ de taille $n \times n$ inversibles à coefficients dans $\mathbf{F}_q$.

■

Cette proposition donne un moyen pratique de construire un très grand nombre de *matrices de distorsion*. Avec les paramètres originaux

$$q = 2, \quad m = n = 20, \quad k = 12, \quad t_1 = 1,$$

on obtient $\approx 2^{640}$ *matrices de distorsion différentes*. En prenant par exemple

$$q = 2, \quad m = n = 32, \quad k = 17, \quad t_1 = 5$$

il y a $\approx 2^{2080}$ matrices de distorsion possibles.

Le nombre gigantesque de *matrices de distorsion* empêche tout type d'attaque consistant à énumérer l'ensemble des *matrices de distorsions* possibles, même pour de petits paramètres.

2.1.3 Attaques par décodage

Le système de chiffrement de McELIECE est très sensible aux *attaques par décodage*. C'est pourquoi il est nécessaire d'avoir des tailles de clé de l'ordre de quelques centaines de kbits, indépendamment de la famille de codes choisie comme *espace des clés*.

En revanche, le cryptosystème GPT ne souffre pas de cet inconvénient. On utilise des tailles de clés de l'ordre de quelques milliers de bits tout en gardant une très bonne sécurité contre les attaques par décodage. Une *attaque par décodage* contre le système GPT consiste à résoudre le problème suivant :

Problème 14 (Décodage borné en métrique rang)

Étant donné un code C de longueur n sur $\mathbf{F}_{q^m}$, et étant donné un mot $\mathbf{y}$ de longueur n sur $\mathbf{F}_{q^m}$ dont la distance rang au code C est $\leq r$, retrouver l'unique mot $\mathbf{c}$ de longueur n tel que

$$\begin{cases} \mathbf{c} \in C \\ \text{Rg}(\mathbf{y} - \mathbf{c}) \leq r \end{cases}$$

Ce problème n'a rien à voir avec le problème du *décodage borné* pour la métrique de HAMMING ni avec le décodage par ensembles d'information tel qu'il est présenté dans [CC98] ou dans [Can96, Ch. 3, Sec. 4].

Le succès de l'attaque par décodage par ensembles d'information repose sur le fait que le vecteur d'erreur est de petit poids. Donc peu de positions d'un mot sont corrompues. En revanche, si on parvient à corrompre beaucoup de positions, ces attaques deviennent irréalisables. L'avantage de la *métrique rang* sur la *métrique de HAMMING* est donc que l'on peut corrompre toutes les positions d'un mot, du moment que l'erreur ajoutée est de petit rang. Par exemple le vecteur

$$\mathbf{e} = \underbrace{(\alpha, \alpha, \dots, \alpha)}_{n \text{ fois}}$$

où $\alpha \in \mathbf{F}_{q^m}^*$ est de rang 1, et corrompt toutes les positions.

Les algorithmes de décodage de LEE–BRICKELL et CANTEAUT–CHABAUD ne sont donc pas applicables. À l'heure actuelle, pour résoudre le problème 14, on ne connaît pas d'autre moyen que d'énumérer les sous-espaces vectoriels de dimension t de $\mathbf{F}_{q^m}^n$.

Attaque par énumération des bases N'ayant pas de meilleure méthode que la recherche exhaustive, recouvrer un message $\mathbf{c}$ à partir du chiffré intercepté $\mathbf{y} = \mathbf{c}\mathbf{G}_{\text{pub}} + \mathbf{e}$ revient à compter les vecteurs d'erreur $\mathbf{e}$ de rang $r \leq t - t_1$ tels que

$$\mathbf{y}\mathbf{H}_{\text{pub}}^T = \mathbf{e}\mathbf{H}_{\text{pub}}^T = \mathbf{s}, \quad (4.24)$$

où $\mathbf{s}$ est le syndrome du mot $\mathbf{y}$, $\mathbf{H}_{\text{pub}}$ est une matrice de parité du code engendré par $\mathbf{G}_{\text{pub}}$, *i.e.* $\mathbf{H}_{\text{pub}}$ est une matrice $(n - k) \times n$ telle que

$$\mathbf{G}_{\text{pub}}\mathbf{H}_{\text{pub}}^T = \mathbf{0}.$$

Tout mot $\mathbf{e}$ de longueur n et de rang r s'écrit

$$\mathbf{e} = (E_1, E_2, \dots, E_r)\mathbf{S},$$

où $E_1, E_2, \dots, E_r \in \mathbf{F}_{q^m}$ forment une famille libre sur $\mathbf{F}_q$ et $\mathbf{S}$ est une matrice $r \times n$ sur $\mathbf{F}_q$ de rang t .

Résoudre le système (4.24) revient donc à trouver $E_1, E_2, \dots, E_r$ et $\mathbf{S}$ tels que

$$(E_1, E_2, \dots, E_r)\mathbf{S}\mathbf{H}_{\text{pub}}^T = \mathbf{s}.$$

Une fois que les $E_1, E_2, \dots, E_r$ sont déterminés, cette équation est ramenée à un système linéaire de $(n - k)m$ équations à nr inconnues – par projection sur $\mathbf{F}_q$ – qui sont les coefficients de la matrice $\mathbf{S}$. La résolution du système se fait en $O(n^3r^3)$ multiplications dans $\mathbf{F}_{q^m}$.

Pour trouver $E_1, E_2, \dots, E_r$, il faut énumérer les sous-espaces vectoriels de dimension r de $\mathbf{F}_q^m$. Il y en a

$$\begin{bmatrix} m \\ r \end{bmatrix} = \frac{(q^m - 1)(q^m - q) \cdots (q^m - q^{r-1})}{(q^r - 1)(q^r - q) \cdots (q^r - q^{r-1})}.$$

Pour de petits paramètres, ceux proposés par exemple dans [GPT91]

$$q = 2, \quad m = n = 20, \quad k = 12, \quad r = 3,$$

la taille de la *clé publique* est de $(12 \times 20) \times 20 = 4800$ bits. Le nombre de sous-espaces vectoriels de dimension r est $\approx 2^{52}$. Ceci est très insuffisant du point de vue de la sécurité, mais la taille de l'espace à énumérer peut être augmentée de manière considérable en choisissant

$$q = 2, \quad m = n = 32, \quad k = 17, \quad r = 5,$$

la taille de la *clé publique* est de $(17 \times 32) \times 32 = 17408$ bits. Le nombre de sous-espace vectoriels à énumérer est alors $\approx 2^{136}$.

Autres attaques Un autre inconvénient du système est qu'il n'est pas déterministe. Comme dans le cas du cryptosystème de McELIECE, quand un même clair a été chiffré de deux manières différentes, on peut le détecter. Contrairement au cas de la *métrique de HAMMING*, dans lequel on retrouve facilement le message initial, cette information n'est désormais plus exploitable.

Soit $\mathbf{c}$ un mot que l'on chiffre de deux manières différentes par un cryptosystème GPT, de clé publique $\mathbf{G}_{\text{pub}}$:

$$\begin{cases} \mathbf{y}_1 = \mathbf{c}\mathbf{G}_{\text{pub}} + \mathbf{e}_1 \\ \mathbf{y}_2 = \mathbf{c}\mathbf{G}_{\text{pub}} + \mathbf{e}_2 \end{cases}$$

où $\text{Rg}(\mathbf{e}_1) \leq t - t_1$ et $\text{Rg}(\mathbf{e}_2) \leq t - t_1$. Le mot $\mathbf{y}_1 - \mathbf{y}_2$ est donc de rang inférieur à $2(t - t_1)$. Cette quantité est inférieure à la *distance rang minimale* du *code public*. Un attaquant détecte donc cette configuration relativement aisément.

Cependant, contrairement au cas de la métrique de HAMMING, cette propriété ne facilite pas du tout la recherche du *clair*. L'attaque décrite dans la section 1.2.2 du chapitre 1 s'appuie sur le fait que, comme les erreurs sont de poids de HAMMING faible, on retrouve les positions corrompues facilement. Dans le cas de la métrique rang en revanche, les erreurs pouvant être de poids aussi élevé que l'on veut, on ne peut pas appliquer cette attaque.

2.2 Les attaques de GIBSON

J. K. GIBSON a présenté successivement deux attaques structurelles contre le cryptosystème GPT – voir [Gib95b, Gib96]. Celles-ci offrent le grand intérêt d'attaquer le cryptosystème par la *matrice de distorsion* en montrant le rôle crucial joué par la structure des codes de GABIDULIN. Bien qu'elles soient de complexité exponentielle, ces attaques montrent également que la matrice de distorsion doit être choisie de «manière judicieuse».

Les deux attaques – dites de GIBSON – ont pour but de résoudre le problème suivant permettant de recouvrer un décodeur du *code public*.

Problème 15

- Étant donné $(\mathbf{G}_{\text{pub}}, t_1)$ la clé publique d'un cryptosystème GPT, où
- $\mathbf{G}_{\text{pub}}$ est une matrice $k \times n$ à coefficients dans $\mathbf{F}_{q^m}$.
 - t_1 est le paramètre de distorsion,

trouver un triplet $(\mathbf{G}, \mathbf{S}, \mathbf{X})$ tel que

$$\mathbf{G}_{\text{pub}} = \mathbf{S}\mathbf{G} + \mathbf{X},$$

avec

- $\mathbf{S}$, une matrice $k \times k$ inversible à coefficients dans $\mathbf{F}_{q^m}$,
- $\mathbf{G} = \left(g_j^{[i]} \right)_{i=0, j=1}^{k-1, n}$, une matrice génératrice d'un code de GABIDULIN,
- $\mathbf{X}$, une matrice $k \times n$ à coefficients dans $\mathbf{F}_{q^m}$ vérifiant

$$\forall \mathbf{c} \in \mathbf{F}_{q^m}^k, \quad \text{Rg}(\mathbf{c}\mathbf{X}) \leq t_1.$$

Bien que le triplet $(\mathbf{G}, \mathbf{S}, \mathbf{X})$ constituant la *clé privée* soit trivialement solution du problème 15, il peut y avoir d'autres triplets $(\mathbf{G}', \mathbf{S}', \mathbf{X}')$ qui sont également solution du problème. Malheureusement, on ne sait pas les dénombrer – voir [Her00].

Dans cette section, je présente les deux attaques de GIBSON dans l'ordre chronologique. La première attaque se comprend bien, et montre que, modulo «quelques» conditions, on peut résoudre le problème 15 en temps polynomial. Ensuite, je présente une version édulcorée de la seconde attaque. La complexité que je trouve est plus importante que celle mesurée par GIBSON, mais j'ai délibérément choisi d'éliminer les nombreuses heuristiques, qui en rendent la compréhension plus obscure qu'elle ne devrait être.

2.2.1 Première attaque

La première attaque – chronologiquement parlant – montre que le concepteur du système doit prendre un *paramètre de distorsion* t_1 suffisamment grand pour masquer la structure des codes de GABIDULIN.

Dans la section 2.1.2 nous avons vu que la *matrice de distorsion* tenait un rôle essentiel dans la sécurité du système. En effet, si celle-ci est nulle –i.e. $\mathbf{X} = 0$ – alors un attaquant résout le problème 15 en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$. GIBSON – voir [Gib95b, Gab95] – suit la même démarche et montre que le problème 15 peut être résolu en temps exponentiel en fonction du *paramètre de distorsion* t_1 .

Dans un premier temps, je réduis le problème 15 à un problème équivalent. Ce dernier revient à résoudre un système *presque* linéaire dont les inconnues sont des éléments de $\mathbf{F}_{q^m}$. Cette réduction est possible grâce aux propriétés de la famille des *codes de GABIDULIN*. Dans un second temps, je montre que ce système «presque» linéaire peut être résolu en temps polynomial pourvu qu'on ait deviné les valeurs de t_1 éléments dans $\mathbf{F}_{q^m}$.

Proposition 30 (Première attaque de GIBSON) *Une solution du problème 15 peut-être déterminée en $O((n-k)k^3q^{mt_1})$ multiplications dans $\mathbf{F}_{q^m}$.*

À partir de là, je décris dans un premier temps l'étape de réduction du problème 15, puis l'étape de résolution du problème réduit. Par souci de clarté, je commence par décrire les étapes de l'attaque avec $t_1 = 1$. Dans le cryptosystème original, la matrice de distorsion d'un système où $t_1 = 1$ s'écrit

$$\mathbf{X} = \mathbf{a}^T \mathbf{b},$$

où $\mathbf{a} = (a_1, a_2, \dots, a_k)$ et $\mathbf{b} = (b_1, b_2, \dots, b_n)$ sont deux vecteurs non nuls à coefficients dans $\mathbf{F}_{q^m}$.

Étape 1 – Réduction du problème Sans perte de généralité, on peut supposer que les k premières colonnes de la matrice $\mathbf{G}_{\text{pub}}$ forment un ensemble d'information, *i.e.* que la matrice constituée des k premières colonnes de $\mathbf{G}_{\text{pub}}$ est inversible. Si tel n'est pas le cas on permute alors les colonnes de $\mathbf{G}_{\text{pub}}$. Ensuite, on procède comme dans la suite en appliquant cette permutation aux matrices qui entrent en jeu.

Par des opérations linéaires sur les lignes et les colonnes de $\mathbf{G}_{\text{pub}}$, l'attaquant détermine la forme échelonnée $(\mathbf{I} \mid \mathbf{R})$ de la clé publique $\mathbf{G}_{\text{pub}}$ où $\mathbf{I}$ est la matrice identité d'ordre k et $\mathbf{R} = (R_{i,j})_{i=0, j=k+1}^{i=k-1, j=n}$ est une matrice à coefficients dans $\mathbf{F}_{q^m}$. Cette opération coûte en moyenne $k^3/2$ multiplications sur $\mathbf{F}_{q^m}$ en utilisant la méthode du pivot de GAUSS.

Posons $\mathbf{G}_{\text{pub}} = \mathbf{S}(\mathbf{G} + \mathbf{Y})$ où $\mathbf{Y} = \mathbf{S}^{-1}\mathbf{X}$ et

$$\mathbf{G} = (\mathbf{G}_1 \mid \mathbf{G}_2), \quad \mathbf{Y} = (\mathbf{Y}_1 \mid \mathbf{Y}_2),$$

les matrices $\mathbf{G}_1, \mathbf{Y}_1$ constituant respectivement les matrices carrées $k \times k$ formées des k premières colonnes de $\mathbf{G}$ et de $\mathbf{Y}$.

Par conséquent la matrice $\mathbf{S}(\mathbf{G}_1 + \mathbf{Y}_1)$ est la matrice composée des k premières colonnes de $\mathbf{G}_{\text{pub}}$. Comme, par hypothèse, elle est inversible et comme la forme échelonnée de $\mathbf{G}_{\text{pub}}$ est unique, on en déduit l'équation matricielle

$$(\mathbf{S}(\mathbf{G}_1 + \mathbf{Y}_1))^{-1} \mathbf{G}_{\text{pub}} = (\mathbf{I} \mid \mathbf{R})$$

aux inconnues $\mathbf{S}, \mathbf{G}_1$ et $\mathbf{Y}_1$. Donc, en multipliant à gauche l'équation

$$\mathbf{G}_{\text{pub}} = \mathbf{S}(\mathbf{G} + \mathbf{Y})$$

par $(\mathbf{S}(\mathbf{G}_1 + \mathbf{Y}_1))^{-1}$ on obtient l'équation suivante vérifiée par $\mathbf{G}$ et $\mathbf{Y}$

$$(\mathbf{I} \mid \mathbf{R}) = ((\mathbf{G}_1 + \mathbf{Y}_1))^{-1} (\mathbf{G} + \mathbf{Y}).$$

En développant, on obtient

$$\mathbf{G}_1 \mathbf{R} = \mathbf{G}_2 + \mathbf{Y}_2 - \mathbf{Y}_1 \mathbf{R}. \quad (4.25)$$

Les inconnues de cette équation sont les matrices $\mathbf{G}_1$, $\mathbf{G}_2$, $\mathbf{Y}_1$, $\mathbf{Y}_2$. Soit $\mathbf{b} = (\mathbf{b}_1 \mid \mathbf{b}_2)$ où $\mathbf{b}_1 = (b_1, b_2, \dots, b_k)$ et $\mathbf{b}_2 = (b_{k+1}, b_{k+2}, \dots, b_n)$. On introduit les notations suivantes identiques à celles données dans [Gab95] :

$$\begin{cases} \mathbf{c} = (c_1, c_2, \dots, c_k) & = (\mathbf{S}^{-1}\mathbf{a}^T)^T, \\ \mathbf{d} = (d_{k+1}, d_{k+2}, \dots, d_n) & = \mathbf{b}_1 - \mathbf{b}_2\mathbf{R}, \\ \mathbf{f} = (c_1, c_2^{[-1]}, \dots, c_k^{[1-k]}) & . \end{cases}$$

Avec ces nouvelles notations, et comme $\mathbf{Y} = \mathbf{S}^{-1}\mathbf{a}^T\mathbf{b}$, le système (4.25) se réécrit

$$\mathbf{G}_1\mathbf{R} = \mathbf{G}_2 + \mathbf{c}^T\mathbf{d}. \quad (4.26)$$

En développant les équations coordonnée par coordonnée, on obtient le système équivalent

$$g_1^{[i-1]}R_{0,j} + g_2^{[i-1]}R_{1,j} + \dots + g_k^{[i-1]}R_{k-1,j} = g_j^{[i]} + c_i d_j, \quad \begin{cases} i = 1, \dots, k, \\ j = k+1, \dots, n. \end{cases}$$

Ce système aux inconnues g_i , d_i , c_i n'est pas linéaire. Cependant il est possible de le relinéariser en élevant la i ème équation à la puissance $[-i] = q^{-i}$, ce qui donne

$$g_1 R_{0,j}^{[-i+1]} + g_2 R_{1,j}^{[-i+1]} + \dots + g_k R_{k-1,j}^{[-i+1]} = g_j + f_i d_j^{[-i]}, \quad \begin{cases} i = 1, \dots, k, \\ j = k+1, \dots, n. \end{cases}$$

Et en posant pour $j = k+1, \dots, n$

$$\mathbf{R}_j = \begin{pmatrix} R_{0,j}^{[0]} & R_{0,j}^{[-1]} & \dots & R_{0,j}^{[1-k]} \\ R_{1,j}^{[0]} & R_{1,j}^{[-1]} & \dots & R_{1,j}^{[1-k]} \\ \vdots & \vdots & \ddots & \vdots \\ R_{k-1,j}^{[0]} & R_{k-1,j}^{[-1]} & \dots & R_{k-1,j}^{[1-k]} \end{pmatrix}, \quad \mathbf{D}_j = \begin{pmatrix} d_j & 0 & \dots & 0 \\ 0 & d_j^{[-1]} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_j^{[1-k]} \end{pmatrix},$$

on obtient le système équivalent

$$(g_1, g_2, \dots, g_k)\mathbf{R}_j = \underbrace{(g_j, g_j, \dots, g_j)}_{k \text{ fois}} + \mathbf{f} \mathbf{D}_j,$$

pour $j = k+1, \dots, n$. Le problème 15 a donc été réduit en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$ – c'est le coût de la mise sous forme systématique de la clé publique – au problème «presque» linéaire suivant qui lui est équivalent.

Problème 16 (Problème réduit)

Trouver deux vecteurs $\mathbf{g}_1 = (g_1, g_2, \dots, g_k)$, et $\mathbf{f} = (f_1, f_2, \dots, f_k)$ à coefficients dans $\mathbf{F}_{q^m}$, et des éléments $d_{k+1}, \dots, d_n, g_{k+1}, \dots, g_n$ de $\mathbf{F}_{q^m}$ vérifiant le système d'équations

$$\begin{cases} \mathbf{g}_1\mathbf{R}_{k+1} & = \underbrace{(g_{k+1}, g_{k+1}, \dots, g_{k+1})}_{k \text{ fois}} + \mathbf{f} \mathbf{D}_{k+1}, \\ \vdots & \vdots \\ \mathbf{g}_1\mathbf{R}_n & = \underbrace{(g_n, g_n, \dots, g_n)}_{k \text{ fois}} + \mathbf{f} \mathbf{D}_n. \end{cases} \quad (4.27)$$

Jusqu'ici le coût des étapes effectives de réduction du problème représente le coût du calcul de la matrice $\mathbf{R}$ à partir de la *clé publique*. Cette étape se fait en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$. À partir de la matrice $\mathbf{R}$, le calcul des valeurs $R_{i,j}^{[s]}$ se fait en temps linéaire pourvu qu'on ait choisi une base normale de $\mathbf{F}_{q^m}$ sur $\mathbf{F}_q$. La réduction du problème 15 au problème 16 se fait donc *grosso modo* en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.

Si la partie non linéaire du système (4.27) était nulle, résoudre le problème réduit ne coûterait qu'une inversion de matrice de plus, comme je l'ai montré dans la section 2.1.2. Cependant, la partie non linéaire est difficile à traiter. GIBSON s'est astucieusement attaqué à ce problème en montrant qu'il suffisait de deviner une valeur – par exemple d_{k+2} – pour pouvoir ensuite résoudre le problème 16 en temps polynomial.

Étape 2 – Résolution du problème réduit : Sans perte de généralité, on peut supposer que l'un des d_i est égal à 1. On peut de même supposer que l'un des g_i est égal à 1. Prenons par exemple $g_{k+1} = 1$ et $d_{k+1} = 1$, d'où $\mathbf{D}_{k+1} = \mathbf{I}$. Posons également $\mathbf{g}_1 = (g_1, g_2, \dots, g_k)$ et $\mathbf{1} = \underbrace{(1, 1, \dots, 1)}_{k \text{ fois}}$.

Si on a deviné correctement la valeur de d_{k+2} , on peut trouver une solution au problème 16 en temps polynomial en $O(m^3)$ multiplications dans $\mathbf{F}_{q^m}$.

La première étape consiste à écrire le sous-système d'équations correspondant à $j = k + 1, k + 2, k + 3$:

$$\mathbf{g}_1 \mathbf{R}_{k+1} = \mathbf{1} + \mathbf{f}, \quad (4.28)$$

$$\mathbf{g}_1 \mathbf{R}_{k+2} = g_{k+2} \mathbf{1} + \mathbf{f} \mathbf{D}_{k+2}, \quad (4.29)$$

$$\mathbf{g}_1 \mathbf{R}_{k+3} = g_{k+3} \mathbf{1} + \mathbf{f} \mathbf{D}_{k+3}, \quad (4.30)$$

et à montrer que, connaissant d_{k+2} , on peut calculer d_{k+3} en temps polynomial. Dans un premier temps, on réécrit le système d'équations sous une forme différente.

En utilisant l'expression $\mathbf{f} = \mathbf{g}_1 \mathbf{R}_{k+1} - \mathbf{1}$ donnée par l'équation (4.28) et en remplaçant $\mathbf{f}$ par cette valeur dans l'équation (4.30) on obtient l'équation

$$g_{k+3} \mathbf{1} = \mathbf{g}_1 (\mathbf{R}_{k+3} - \mathbf{R}_{k+1} \mathbf{D}_{k+3}) + \mathbf{1} \mathbf{D}_{k+3}. \quad (4.31)$$

Si de plus on suppose que la matrice $\mathbf{R}_{k+2} - \mathbf{R}_{k+1} \mathbf{D}_{k+2}$ est inversible, alors en posant $\mathbf{U} = (\mathbf{R}_{k+2} - \mathbf{R}_{k+1} \mathbf{D}_{k+2})^{-1}$, et en combinant les équations (4.28) et (4.29), on obtient l'équation suivante qui donne le vecteur $\mathbf{g}_1$ en fonction de g_{k+2} et d_{k+2}

$$\mathbf{g}_1 = g_{k+2} \mathbf{1} \mathbf{U} - \mathbf{1} \mathbf{D}_{k+2} \mathbf{U}. \quad (4.32)$$

En substituant cette équation au vecteur $\mathbf{g}_1$ dans (4.31), on obtient

$$g_{k+3} \mathbf{1} + g_{k+2} \mathbf{v} + \mathbf{w} = 0, \quad (4.33)$$

où

$$\begin{cases} \mathbf{v} &= \mathbf{1}\mathbf{U}(\mathbf{R}_{k+1}\mathbf{D}_{k+3} - \mathbf{R}_{k+3}), \\ \mathbf{w} &= (\mathbf{1}\mathbf{D}_{k+2}\mathbf{U}\mathbf{R}_{k+1} + \mathbf{1})\mathbf{D}_{k+3} - \mathbf{1}\mathbf{D}_{k+2}\mathbf{U}\mathbf{R}_{k+3}. \end{cases}$$

1. *Calcul de d_{k+3} en fonction de d_{k+2} :* L'équation

$$g_{k+3}\mathbf{1} + g_{k+2}\mathbf{v} + \mathbf{w} = 0$$

n'a de solution que si les trois vecteurs $\mathbf{1}$, $\mathbf{v}$ et $\mathbf{w}$ ne sont pas linéairement indépendants sur $\mathbf{F}_{q^m}$. Je souhaite épargner la résolution technique et inintéressante de ce système au lecteur. J'espère qu'il me fera confiance si j'écris que, comme la matrice $\mathbf{D}_{k+3}$ est une matrice diagonale, les vecteurs $\mathbf{v}$, $\mathbf{w}$, et $\mathbf{1}$ sont linéairement dépendants si et seulement si pour $i = 1, \dots, k-2$, les déterminants

$$\begin{vmatrix} 1 & 1 & 1 \\ a_i - b_i d_{k+3}^{[-i+1]} & a_{i+1} - b_{i+1} d_{k+3}^{[-i]} & a_{i+2} - b_{i+2} d_{k+3}^{[-i-1]} \\ a'_i - b'_i d_{k+3}^{[-i+1]} & a'_{i+1} - b'_{i+1} d_{k+3}^{[-i]} & a'_{i+2} - b'_{i+2} d_{k+3}^{[-i-1]} \end{vmatrix}$$

où a_i, b_i, a'_i, b'_i dépendent de d_{k+2} sont tous nuls. Par exemple, en développant le déterminant correspondant à $i = 1$ on voit que pour que la condition d'annulation soit satisfaite, d_{k+3} doit être racine d'un polynôme de degré $q + q^2$. De la même manière, on peut ramener chaque équation obtenue à la détermination des racines d'un polynôme de degré $q + q^2$. Il suffit alors de calculer le *pgcd* de ces polynômes – en utilisant par exemple un algorithme d'EUCLIDE étendu – puis d'en déterminer les racines. Puisque les degrés des polynômes sont petits en général – degré 6 si $q = 2$ –, déterminer d_{k+3} est peu coûteux comparé aux autres étapes de l'attaque.

2. *Calcul de g_{k+2} et de g_{k+3} en fonction de d_{k+2} :* connaissant d_{k+3} en fonction de d_{k+2} , on en déduit les coordonnées des vecteurs $\mathbf{v} = (v_0, v_1, \dots, v_{k-1})$, et $\mathbf{w} = (w_0, w_1, \dots, w_{k-1})$ en fonction de d_{k+2} . Résoudre le système

$$g_{k+3}\mathbf{1} + g_{k+2}\mathbf{v} + \mathbf{w} = 0$$

est donc équivalent à résoudre un système linéaire à deux inconnues g_{k+3} et g_{k+2}

$$g_{k+3} + g_{k+2}v_i + w_i = 0,$$

pour $i = 1, \dots, k$.

En conséquence, si on connaît d_{k+3} – ce qui fut fait dans l'étape précédente – et d_{k+2} , on détermine g_{k+2} et g_{k+3} en temps constant.

3. *Calcul de $\mathbf{g}_1$ en fonction de d_{k+2} :* L'équation (4.32) donne le vecteur $\mathbf{g}_1$ en fonction de g_{k+2} – déterminé à l'étape précédente en fonction de d_{k+2} – et de $\mathbf{U}$ – connue en fonction de d_{k+2} . Cette étape coûte donc $O(k^2)$ multiplications dans $\mathbf{F}_{q^m}$.

4. *Calcul de $\mathbf{f}$ en fonction de $\mathbf{g}_1$* : l'équation (4.28) nous donne

$$\mathbf{f} = \mathbf{g}_1 \mathbf{R}_{k+1} - \mathbf{1}.$$

Le calcul de $\mathbf{f}$ coûte la multiplication d'un vecteur de longueur k par une matrice $k \times n$ à coefficients dans $\mathbf{F}_{q^m}$, soit $O(kn)$ multiplications dans $\mathbf{F}_{q^m}$.

Ces étapes successives prouvent donc que, connaissant la valeur de d_{k+2} dans $\mathbf{F}_{q^m}$, on détermine

$$\mathbf{g}_1 = (g_1, g_2, \dots, g_k), \mathbf{f}, g_{k+2}, g_{k+3}, d_{k+3}$$

en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$. Une fois que ces variables sont déterminées, trouver les solutions g_j et $\mathbf{D}_j$ du système

$$\mathbf{g}_1 \mathbf{R}_j = \underbrace{(g_j, g_j, \dots, g_j)}_{k \text{ fois}} + \mathbf{f} \mathbf{D}_j,$$

où $j = k + 4, \dots, n$ se fait en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$ par étape. Il suffit en effet d'inverser quelques matrices à coefficients dans $\mathbf{F}_{q^m}$. La complexité totale de l'étape de résolution du problème réduit 16 est donc de $O((n-k)k^3)$ multiplications dans $\mathbf{F}_{q^m}$ pourvu qu'on connaisse $d_{k+2} \in \mathbf{F}_{q^m}$. La complexité totale est donc $O((n-k)k^3 q^m)$.

L'algorithme permettant de résoudre le problème 16 est détaillé dans la table 4.3.

Attaque sur un paramètre de distorsion $t_1 > 1$ Je ne détaillerai pas le principe de l'attaque de GIBSON lorsque la *matrice de distorsion* a un rang t_1 supérieur à 1. Le principe est très similaire. On commence par réduire le problème 15 à un problème réduit. On peut alors écrire la *matrice de distorsion*,

$$\mathbf{X} = \mathbf{A}\mathbf{B},$$

où $\mathbf{A}$ est une matrice $k \times t_1$ à coefficients dans $\mathbf{F}_{q^m}$ de rang t_1 , et $\mathbf{B}$ est une matrice $t_1 \times n$ à coefficients dans $\mathbf{F}_q$. Résoudre le problème 15 revient à résoudre le système d'équations

$$g_1 R_{0,j}^{[-i]} + \dots + g_k R_{k-1,j}^{[-i]} = g_j + \sum_{s=0}^{t_1} f_{i,s} d_{s-1,j}^{[-i]}, \quad \begin{cases} i = 0, \dots, k-1, \\ j = k+1, \dots, n, \end{cases}$$

Aux inconnues $g_1, \dots, g_n$, $(d_{i,j})_{i=0, j=k+1}^{t_1-1, n}$ et $(f_{i,j})_{i=0, j=1}^{k-1, s}$. Dans ce cas précis, on montre [Gab95] qu'il suffit de deviner $d_{0, k+t_1}, \dots, d_{t_1-1, k+t_1}$ dans $\mathbf{F}_{q^m}$ pour résoudre ce système en $O((n-k)k^3)$ multiplications dans $\mathbf{F}_q^m$, en énumérant $\mathbf{F}_{q^m}$ par exemple. Dans ce cas, la complexité totale de l'attaque est donc égale à

$$O((n-k)k^3 q^{t_1 m}) \text{ multiplications dans } \mathbf{F}_{q^m}.$$

Données : $g_{k+1} = 1$, $d_{k+1} = 1$, des matrices $(\mathbf{R}_j)_{j=k+1}^n$.

Résultat : $\mathbf{g} = (g_1, \dots, g_k)$, $\mathbf{f}$, $d_{k+2}, \dots, d_n$, solutions de

$$\mathbf{g}_1 \mathbf{R}_j = \underbrace{(g_j, g_j, \dots, g_j)}_{k \text{ fois}} + \mathbf{f} \mathbf{D}_j,$$

pour $j = k+1, \dots, n$ et où

$$\mathbf{D}_j = \begin{pmatrix} d_j & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_j^{[1-k]} \end{pmatrix}.$$

Déroulement de l'attaque :

1. On tire aléatoirement une valeur pour $d_{k+2} \in \mathbf{F}_{q^m}$,
2. Pour $i = 0, \dots, k-1$, on calcule la solution d_{k+3} du système

$$\begin{vmatrix} 1 & 1 & 1 \\ a_i - b_i d_{k+3}^{[-i+1]} & a_{i+1} - b_{i+1} d_{k+3}^{[-i]} & a_{i+2} - b_{i+2} d_{k+3}^{[-i-1]} \\ a'_i - b'_i d_{k+3}^{[-i+1]} & a'_{i+1} - b'_{i+1} d_{k+3}^{[-i]} & a'_{i+2} - b'_{i+2} d_{k+3}^{[-i-1]} \end{vmatrix} = 0,$$

où les a_i, b_i, a'_i, b'_i s'expriment en fonction de d_{k+2} .

3. On calcule g_{k+3} et g_{k+2} en inversant le système d'équations

$$g_{k+3} + g_{k+2} v_i + w_i = 0, \quad i = 0, \dots, k-1$$

4. On calcule $\mathbf{g}_1$ sachant que

$$\mathbf{g}_1 = g_{k+2} \mathbf{1U} + \mathbf{1D}_{k+2} \mathbf{U}$$

5. On détermine $\mathbf{f} = \mathbf{g}_1 \mathbf{R}_{k+1} - \mathbf{1}$.
6. On résout les systèmes restant

$$\mathbf{g}_1 \mathbf{R}_j = \underbrace{(g_j, g_j, \dots, g_j)}_{k \text{ fois}} + \mathbf{f} \mathbf{D}_j, \quad j = k+3, \dots, n$$

aux inconnues g_j et d_j .

TABLEAU 4.3: Première attaque de GIBSON – cas où le paramètre de distortion $t_1 = 1$

2.2.2 Seconde attaque

En 1996, GIBSON publia une deuxième attaque contre le cryptosystème GPT, voir [Gib96], qui se révéla bien plus efficace. Dans un premier temps, je montre qu'on peut réduire le problème 15 au *problème de recherche* d'une matrice à coefficients dans $\mathbf{F}_q$ ayant certaines propriétés. En effet, à partir de la solution de ce *problème de recherche*, on résout le problème 15 en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$. Dans un deuxième temps, je montre, sans entrer dans les détails, quelle fut l'approche envisagée par GIBSON permettant de résoudre le problème de recherche.

Réduction au *problème de recherche* : Une matrice $\mathbf{X}$ de taille $k \times n$ et de *paramètre de distorsion* t_1 peut s'écrire sous la forme

$$\mathbf{X} = \mathbf{A}\mathbf{B},$$

où $\mathbf{A}$ est une matrice de taille $k \times t_1$ à coefficients dans $\mathbf{F}_{q^m}$ et $\mathbf{B}$ est une matrice de taille $t_1 \times n$ à coefficients dans $\mathbf{F}_q$ et de rang t_1 . Sans perte de généralité, on peut supposer que $\mathbf{B}$ s'écrit

$$\mathbf{B} = (\mathbf{B}_k \mid \mathbf{I}_{t_1} \mid \mathbf{B}_f), \quad (4.34)$$

où $\mathbf{B}_k$ est la matrice des k premières colonnes de $\mathbf{B}$, et où $\mathbf{B}_f$ est la matrice des $f = n - k - t_1$ dernières colonnes de $\mathbf{B}$. Si tel n'était pas le cas, il suffirait de permuter les colonnes de $\mathbf{B}$ pour obtenir la forme (4.34).

Soit $\mathbf{C}$ la matrice $(n - t_1) \times n$ telle que

$$\mathbf{C} = \left(\begin{array}{c|c} \mathbf{I}_k & 0 \\ -\mathbf{B}_k & \mathbf{B}_f \\ 0 & -\mathbf{I}_f \end{array} \right). \quad (4.35)$$

$\mathbf{C}$ est une matrice de rang $n - t_1$, à coefficients dans $\mathbf{F}_q$, qui ne dépend que de la matrice $\mathbf{B}$. Par construction, $\mathbf{C}$ vérifie $\mathbf{B}\mathbf{C} = \mathbf{0}$.

Le problème de recherche qui m'intéresse est le suivant

Problème 17

Étant donnée une matrice $\mathbf{G}_{\text{pub}}$ à coefficients dans $\mathbf{F}_{q^m}$ telle que

$$\mathbf{G}_{\text{pub}} = \mathbf{S}\mathbf{G} + \mathbf{A}\mathbf{B},$$

où $\mathbf{S}$ est une matrice $k \times k$ inversible à coefficients dans $\mathbf{F}_{q^m}$, où $\mathbf{G}$ est une matrice $k \times n$ génératrice d'un code de GABIDULIN, et où

$$\mathbf{B} = (\mathbf{B}_k \mid \mathbf{I}_{t_1} \mid \mathbf{B}_f)$$

est une matrice $t_1 \times n$ de rang t_1 sur $\mathbf{F}_q$, trouver une matrice $\mathbf{C}$ de taille $n \times (n - t_1)$ à coefficients dans $\mathbf{F}_q$ telle que

$$\begin{cases} \mathbf{C} & = \left(\begin{array}{c|c} \mathbf{I}_k & 0 \\ -\mathbf{B}_k & \mathbf{B}_f \\ 0 & -\mathbf{I}_f \end{array} \right), \\ \mathbf{G}_{\text{pub}}\mathbf{C} & = \mathbf{S}\mathbf{G}\mathbf{C}. \end{cases}$$

Il est intéressant de remarquer que les hypothèses du problème 17 sont très faibles. En effet, les seules informations publiques sont :

1. La matrice $\mathbf{G}_{\text{pub}}$, la clé publique d'un cryptosystème GPT.
2. Le fait que la matrice $\mathbf{S}$ est inversible.
3. Le fait que la matrice $\mathbf{G}$ étant une matrice génératrice d'un code de GABIDULIN a une structure particulière.

Trouver une solution au problème 17 sans faire de recherche exhaustive sur les matrices $\mathbf{C}$ possibles relève donc du tour de force.

À présent, je montre que, résoudre le problème de recherche 17 permet de résoudre le problème 15 en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$. Considérons un cryptosystème GPT de *clé publique* $(\mathbf{G}_{\text{pub}}, t_1)$, où

$$\mathbf{G}_{\text{pub}} = \mathbf{S}\mathbf{G} + \mathbf{A}\mathbf{B}.$$

Supposons que l'on connaisse la matrice $\mathbf{C}$, solution du problème 17 avec ces paramètres. On a

$$\mathbf{G}_{\text{pub}}\mathbf{C} = \mathbf{S}\mathbf{G}\mathbf{C}, \quad (4.36)$$

où $\mathbf{S}$ est une matrice inversible de taille $k \times k$ et où $\mathbf{G} = \left(g_j^{[i]} \right)_{i=0, j=1}^{k-1, n}$ est une matrice génératrice d'un code de GABIDULIN. Posons $\mathbf{G}' = \mathbf{G}\mathbf{C}$, et $\mathbf{Z} = \mathbf{G}_{\text{pub}}\mathbf{C}$. Avec ces notations :

1. On détermine $\mathbf{S}$ et $\mathbf{G}'$ en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$: l'équation (4.36) se réécrit

$$\mathbf{Z} = \mathbf{S}\mathbf{G}', \quad (4.37)$$

où $\mathbf{Z} = \mathbf{G}_{\text{pub}}\mathbf{C}$ est connue et où $\mathbf{G}' = \mathbf{G}\mathbf{C}$. Comme $\mathbf{C}$ est à coefficients dans $\mathbf{F}_q$, on a $\mathbf{G}' = \left(h_j^{[i]} \right)_{i=0, j=1}^{k-1, n-t_1}$, pour certains éléments h_j . De plus, comme $\mathbf{C}$ est de rang $n-t_1$, et que les g_j sont linéairement indépendants sur $\mathbf{F}_q$, les h_j sont linéairement indépendants sur $\mathbf{F}_q$. Par conséquent, le code engendré par la matrice $\mathbf{G}'$ est un code de GABIDULIN de longueur $n-t_1$ et de dimension k . Résoudre l'équation (4.37) revient donc à casser une instance du cryptosystème GPT dont la *matrice de distorsion* est nulle. D'après les résultats de la section 2.1.2, un attaquant y parvient en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.

2. On détermine la matrice $\mathbf{G}$ en $O(n^3)$ multiplications dans $\mathbf{F}_q$:

À partir de la connaissance de la matrice $\mathbf{G}'$, on cherche une matrice

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

telle que les $g_i \in \mathbf{F}_{q^m}$ soient linéairement indépendants sur $\mathbf{F}_q$, et qui soit solution de l'équation

$$\mathbf{G}' = \mathbf{G}\mathbf{C}. \quad (4.38)$$

Soit

$$\begin{cases} \mathbf{g} &= (g_1, g_2, \dots, g_n) &= (\mathbf{g}_k, \mathbf{g}_{t_1}, \mathbf{g}_f), \\ \mathbf{h} &= (h_1, h_2, \dots, h_{n-t_1}) &= (\mathbf{h}_k, \mathbf{h}_f). \end{cases}$$

Comme $\mathbf{C}$ est à coefficients dans $\mathbf{F}_q$, résoudre l'équation (4.38) est équivalent à trouver un vecteur $\mathbf{g}$ à coefficients dans $\mathbf{F}_{q^m}$ de rang m sur $\mathbf{F}_q$ tel que

$$(\mathbf{g}_k, \mathbf{g}_f) = (\mathbf{h}_k, \mathbf{h}_f) + \mathbf{g}_{t_1} (\mathbf{B}_k \mid -\mathbf{B}_f). \quad (4.39)$$

Un moyen consiste à tirer aléatoirement un vecteur $\mathbf{g}_{t_1}$ dans $\mathbf{F}_{q^m}$ dont les coordonnées sont linéairement indépendantes sur $\mathbf{F}_q$. Puis on détermine les vecteurs $\mathbf{g}_k$ et $\mathbf{g}_f$ en résolvant l'équation (4.39). Si les coordonnées du vecteur $\mathbf{g} = (\mathbf{g}_k, \mathbf{g}_{t_1}, \mathbf{g}_f)$ ne sont pas linéairement indépendantes sur $\mathbf{F}_q$, on tire aléatoirement un autre vecteur $\mathbf{g}_{t_1}$. Expérimentalement – voir [GO00] – il suffit de faire une poignée de tirages pour trouver un vecteur $\mathbf{g}$ convenable. Déterminer $\mathbf{g}$ coûte donc le calcul d'un petit nombre de déterminants dans $\mathbf{F}_q$, soit $O(n^3)$ opérations dans $\mathbf{F}_q$.

3. On détermine la matrice $\mathbf{A}$ en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$:

À l'issue des deux premières étapes on a déterminé une matrice $\mathbf{S}$ inversible à coefficients dans $\mathbf{F}_{q^m}$ et une matrice $\mathbf{G}$ génératrice d'un code de GABIDULIN, telles que

$$\mathbf{G}_{\text{pub}} \mathbf{C} = \mathbf{S} \mathbf{G} \mathbf{C}.$$

Par construction, on a $\mathbf{B} \mathbf{C} = \mathbf{0}$. Donc, il existe une matrice $\mathbf{A}$ à coefficients dans $\mathbf{F}_{q^m}$ de taille $k \times t_1$ telle que

$$\mathbf{G}_{\text{pub}} = \mathbf{S} \mathbf{G} + \mathbf{A} \mathbf{B},$$

où $\mathbf{B} = (\mathbf{B}_k \mid \mathbf{I}_{t_1} \mid \mathbf{B}_f)$. Comme les matrices $\mathbf{S}$ et $\mathbf{G}$ ont été déterminées à l'issue des deux premières étapes, la seule inconnue de cette équation est la matrice $\mathbf{A}$. Trouver $\mathbf{A}$ revient donc à résoudre un système linéaire de rang maximal de kn équations à kt_1 inconnue. La résolution de ce système nécessite $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.

On peut facilement vérifier que les matrices $\mathbf{G}$, $\mathbf{S}$, $\mathbf{A} \mathbf{B}$ satisfont les hypothèses du problème 15.

En conclusion, si a déterminé la matrice $\mathbf{C}$ solution du *problème de recherche*, on trouve une solution du problème 17 en $O(k^3)$ multiplications dans $\mathbf{F}_{q^m}$.

Désormais, on s'attache à résoudre le *problème de recherche* 17 plutôt que le problème 15. On peut d'une part le résoudre en énumérant toutes les matrices $\mathbf{C}$ possibles. D'après leur construction, l'espace à énumérer est de

taille $q^{t_1(n-t_1)}$. La complexité de l'algorithme pour résoudre le problème 15 est dans ce cas égale à

$$O\left(k^3 q^{t_1(n-t_1)}\right)$$

multiplications dans $\mathbf{F}_{q^m}$. La complexité de la seconde attaque est donc inférieure à la complexité de la *première attaque* d'un facteur exponentiel, puisque $n \leq m$.

A priori on dispose de peu d'information pour résoudre le problème 17. En effet, la seule information disponible est :

1. La matrice $\mathbf{G}_{\text{pub}}$.
2. Le paramètre de distorsion t_1 .
3. Le fait que la matrice $\mathbf{S}$ est inversible.
4. Le fait que la matrice $\mathbf{G}$ est une matrice génératrice d'un code de GABIDULIN.

Cependant, encore une fois grâce à la structure particulière des codes de GABIDULIN, on peut éviter la recherche exhaustive. Voici le lemme qui est la clef de voûte sous-tendant la résolution la plus efficace connue du problème 17.

Lemme 4 (Propriété fondamentale) Soit $\mathbf{G} = \left(g_j^{[i]}\right)_{i=0, j=1}^{k-1, k}$ et $\mathbf{G}' = \left(h_j^{[i]}\right)_{i=0, j=1}^{k-1, s}$ deux matrices génératrices de codes de GABIDULIN à coefficients dans $\mathbf{F}_{q^m}$. Soit $\mathbf{Z}$ une matrice $k \times s$ à coefficients dans $\mathbf{F}_{q^m}$ telle que

$$\mathbf{GZ} = \mathbf{G}'.$$

Soit $\mathbf{Z}^{[1]}$, la matrice $\mathbf{Z}$ dont tous les coefficients sont élevés à la puissance $[1] = q$, i.e. si $\mathbf{Z} = (z_{i,j})$, alors $\mathbf{Z}^{[1]} = (z_{i,j}^{[1]})$. Alors la matrice $\mathbf{Z} - \mathbf{Z}^{[1]}$ est de rang 1 et n'a pas de coordonnées nulle.

Preuve

L'égalité $\mathbf{GZ} = \mathbf{G}'$ se réécrit

$$\begin{pmatrix} g_1 & \cdots & g_k \\ \vdots & \ddots & \vdots \\ g_1^{[k-1]} & \cdots & g_k^{[k-1]} \end{pmatrix} \mathbf{Z} = \begin{pmatrix} h_1 & \cdots & h_s \\ \vdots & \ddots & \vdots \\ h_1^{[k-1]} & \cdots & h_s^{[k-1]} \end{pmatrix}.$$

Donc, en élevant le système à la puissance $[1]$, on a par linéarité

$$\begin{pmatrix} g_1^{[1]} & \cdots & g_k^{[1]} \\ \vdots & \ddots & \vdots \\ g_1^{[k]} & \cdots & g_k^{[k]} \end{pmatrix} \mathbf{Z}^{[1]} = \begin{pmatrix} h_1^{[1]} & \cdots & h_s^{[1]} \\ \vdots & \ddots & \vdots \\ h_1^{[k]} & \cdots & h_s^{[k]} \end{pmatrix}.$$

Si on pose $\mathbf{L}$ comme étant la matrice de taille $(k-1) \times k$ constituée des $k-1$ dernières lignes de $\mathbf{G}$, alors

$$\mathbf{L}(\mathbf{Z} - \mathbf{Z}^{[1]}) = 0.$$

Comme $g_1, g_2, \dots, g_k$ sont linéairement indépendants sur $\mathbf{F}_q$, $\mathbf{L}$ est de rang $k-1$ donc son noyau est de rang 1. Donc $\mathbf{Z} - \mathbf{Z}^{[1]}$ est de rang 1.

■

En utilisant ce lemme, GIBSON réduit la résolution du problème 17 à deux sous-étapes.

Réduction 1 en $\mathcal{O}(m^2q^{t_1})$ multiplications dans $\mathbf{F}_{q^m}$: Cette partie de la réduction montre comment déterminer la matrice $\mathbf{C}$ solution du problème 17 quand $\mathbf{C}$ s'écrit

$$\mathbf{C} = \left(\begin{array}{c|c} \mathbf{I}_k & 0 \\ -\mathbf{B}_k & \mathbf{B}'' \\ 0 & -\mathbf{I}_f \end{array} \right)$$

et lorsqu'on connaît la matrice $\mathbf{B}_k$ et la première colonne $\mathbf{b}_1$ de la matrice $\mathbf{B}''$. Si on écrit

$$\begin{cases} \mathbf{G}_{\text{pub}} = (\mathbf{G}_k \mid \mathbf{G}_{t_1} \mid \mathbf{G}_f), \\ \mathbf{G}' = (\mathbf{G}'_k \mid \mathbf{G}'_f), \end{cases}$$

alors l'équation $\mathbf{G}_{\text{pub}}\mathbf{C} = \mathbf{S}\mathbf{G}'$, où $\mathbf{G}'$ est une matrice génératrice d'un *code de GABIDULIN*, se réécrit

$$(\mathbf{G}_k \mid \mathbf{G}_{t_1} \mid \mathbf{G}_f) \left(\begin{array}{c|c} \mathbf{I}_k & 0 \\ -\mathbf{B}_k & \mathbf{B}'' \\ 0 & -\mathbf{I}_f \end{array} \right) = (\mathbf{S}\mathbf{G}'_k \mid \mathbf{S}\mathbf{G}'_f). \quad (4.40)$$

Comme $\mathbf{G}_{\text{pub}}$ est publique, les matrices $\mathbf{G}_k$ et $\mathbf{G}_{t_1}$ sont connues. De plus on a supposé que la matrice $\mathbf{B}_k$ était connue. Donc la matrice $\mathbf{Q} = \mathbf{G}_k - \mathbf{G}_{t_1}\mathbf{B}_k = \mathbf{S}\mathbf{G}'_k$ est connue. De plus, comme $\mathbf{S}$ et $\mathbf{G}'_k$ sont inversibles, la matrice $\mathbf{Q}$ est également inversible. Posons

$$\begin{cases} \mathbf{Q} = \mathbf{G}_k - \mathbf{G}_{t_1}\mathbf{B}_k, \\ \mathbf{Z} = \mathbf{Q}^{-1}(\mathbf{G}_{t_1}\mathbf{B}'' - \mathbf{G}_f). \end{cases}$$

Résoudre le système (4.40) revient à résoudre les équations

$$\mathbf{Q} = \mathbf{S}\mathbf{G}'_k, \quad (4.41)$$

$$\mathbf{G}'_f = \mathbf{G}'_k\mathbf{Z}, \quad (4.42)$$

où $\mathbf{G}'_k$ est une matrice de taille $k \times k$ génératrice d'un *code de GABIDULIN* engendré par $(h_1, h_2, \dots, h_k)$ et où $\mathbf{G}'_f$ est une matrice de taille $k \times f$ génératrice d'un code de *GABIDULIN* engendré par $(h_{k+1}, h_{k+2}, \dots, h_n)$. Les

<p><i>Données :</i></p> <ul style="list-style-type: none"> - La matrice $\mathbf{U}$. - Le vecteur $\mathbf{v}$. - Le vecteur $\mathbf{b}_1$. <p><i>Résultat :</i> la matrice $\mathbf{B}'' = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_f)$.</p> <p><i>Pour tout vecteur $\mathbf{u} \in \mathbf{F}_q^{t_1}$, faire</i></p> <ol style="list-style-type: none"> 1. Calculer $\mathbf{U} \begin{pmatrix} \mathbf{B}_u \\ \mathbf{I}_f \end{pmatrix} = (\mathbf{v}, \mathbf{v}_1, \dots, \mathbf{v}_{f-1}),$ <p>où</p> $\mathbf{B}_u = (\mathbf{b}_1, \underbrace{\mathbf{u}, \dots, \mathbf{u}}_{f-1 \text{ fois}}).$ 2. Pour tous les indices j tels que $\mathbf{v}_j \neq 0$ est proportionnel à $\mathbf{v}$, on pose $\mathbf{b}_j = \mathbf{v}_j$.
--

TABLEAU 4.4 : Seconde attaque de GIBSON : réduction 2

inconnues du système sont les matrices $\mathbf{G}' = (\mathbf{G}'_k \mid \mathbf{G}'_f)$ et $\mathbf{Z}$. Bien qu'on n'ait aucune autre information sur $\mathbf{G}'$, on sait que la matrice $\mathbf{Z}$, solution de l'équation (4.42), satisfait les propriétés requises pour qu'on puisse appliquer le lemme 4. La matrice $\mathbf{Z} - \mathbf{Z}^{[1]}$ est donc une matrice de rang 1 sans coordonnée nulles.

Comme $\mathbf{B}''$ est à coefficients $\mathbf{F}_q$ - *i.e.* $\mathbf{B}''^{[1]} = \mathbf{B}''$ -, et comme $\mathbf{Z} = \mathbf{Q}^{-1}(\mathbf{G}_{t_1} \mathbf{B}'' - \mathbf{G}_f)$, la matrice $\mathbf{Z} - \mathbf{Z}^{[1]}$ vérifie l'équation

$$\mathbf{Z} - \mathbf{Z}^{[1]} = \mathbf{U} \begin{pmatrix} \mathbf{B}'' \\ \mathbf{I}_f \end{pmatrix} \quad (4.43)$$

où $\mathbf{U}$ est une matrice connue, puisqu'elle est fonction de la matrice $\mathbf{B}_k$ qui est connue. Comme la première colonne $\mathbf{b}_1$ de la matrice $\mathbf{B}''$ est connue, la première colonne $\mathbf{v}$ de la matrice $\mathbf{Z} - \mathbf{Z}^{[1]}$ est également connue. Comme $\mathbf{Z} - \mathbf{Z}^{[1]}$ est de rang 1 et sans colonne nulle, on peut écrire

$$\mathbf{Z} - \mathbf{Z}^{[1]} = (\mathbf{v}, \lambda_1 \mathbf{v}, \dots, \lambda_{f-1} \mathbf{v}),$$

où $\lambda_i \in \mathbf{F}_{q^m}^*$.

Cette information est suffisante pour retrouver les colonnes de $\mathbf{B}''$ en $O(m^2 q^{t_1})$ multiplications dans $\mathbf{F}_{q^m}$. La procédure pour déterminer la matrice $\mathbf{B}''$ est décrite dans le tableau 4.4.

À la fin de cette procédure, la matrice $\mathbf{B}'' = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_f)$ ainsi construite vérifie le système (4.43). Le coût de cette opération est donc grosso modo $O(m^2 q^{t_1})$ opérations sur le corps $\mathbf{F}_{q^m}$.

Remarque 4 Cette version de l'étape de réduction n'est pas celle qui fut proposée par GIBSON. J'ai délibérément choisi d'en omettre des parties afin de souligner les idées mathématiques sous-tendant cette réduction. GIBSON prétend qu'il est possible de trouver la matrice $\mathbf{B}''$ en $O(m^2q^{t_1+1-r})$ opérations, où r désigne le rang de la matrice $\mathbf{U}$ de l'équation (4.43).

Réduction 2 : Écrivons la matrice $\mathbf{C}$ que l'on recherche sous la forme

$$\mathbf{C} = \left(\begin{array}{c|c} \mathbf{I}_k & 0 \\ -\mathbf{B}' & \mathbf{B}'' \\ 0 & -\mathbf{I}_f \end{array} \right).$$

Le but de cette réduction est de déterminer la matrice $\mathbf{B}'$, en utilisant encore une fois l'équation vérifiée par la matrice $\mathbf{C}$:

$$\mathbf{G}_{\text{pub}}\mathbf{C} = \mathbf{S}\mathbf{G}' \quad (4.44)$$

où $\mathbf{G}_{\text{pub}}$ est connue, $\mathbf{S}$ est inversible, et $\mathbf{G}'$ est une matrice génératrice d'un code de GABIDULIN.

Dans la version donnée par GIBSON, l'étude de la complexité de cette réduction fait appel à un certain nombre d'heuristiques. Afin de simplifier la procédure, je suppose que $k \leq n - k$. Par des permutations *ad hoc*, puis en mettant ensuite la matrice $\mathbf{G}_{\text{pub}}$ sous forme systématique, on peut sans perte de généralités supposer que $\mathbf{G}_{\text{pub}}$ est de rang maximal k et s'écrit

$$\mathbf{G}_{\text{pub}} = \left(\mathbf{G}_k \mid \begin{array}{c} \mathbf{I}_{t_1} \\ 0 \end{array} \mid \begin{array}{c} 0 \\ \mathbf{I}_{k-t_1} \end{array} \mid \mathbf{H} \right).$$

Si on pose $\mathbf{G}' = (\mathbf{G}'_k \mid \mathbf{G}'_f)$, l'équation (4.44) est équivalente à

$$\begin{cases} \mathbf{Q} & = \mathbf{S}\mathbf{G}'_k, \\ \mathbf{G}'_f & = \mathbf{G}'_k\mathbf{Z}. \end{cases}$$

où, si on note par $\mathbf{B}''_{k-t_1}$ les $k - t_1$ premières colonnes de $\mathbf{B}''$,

$$\mathbf{Q} = \mathbf{G}_k - \begin{pmatrix} \mathbf{B}' \\ 0 \end{pmatrix}, \quad \text{et} \quad \mathbf{Z} = \mathbf{Q}^{-1} \left(\begin{array}{c} \mathbf{B}''_{k-t_1} \\ \mathbf{I}_{k-t_1} \end{array} \mid \mathbf{H}' \right).$$

L'expression de la matrice $\mathbf{H}'$ n'a pas d'importance. En effet, je me limiterai à l'étude des $k - t_1$ premières colonnes des matrices considérées. La matrice $\mathbf{Z}$ satisfaisant les conditions du lemme 4, la matrice $\mathbf{Z} - \mathbf{Z}^{[1]}$ est de rang 1. Partant, la matrice

$$\left(\mathbf{Q}^{-1} - (\mathbf{Q}^{-1})^{[1]} \right) \left(\begin{array}{c} \mathbf{B}''_{k-t_1} \\ \mathbf{I}_{k-t_1} \end{array} \right) \quad (4.45)$$

qui est la matrice des $k - t_1$ premières colonnes de $\mathbf{Z} - \mathbf{Z}^{[1]}$ est également de rang 1.

Cette propriété est la clé de voûte de l'étape de réduction. Le processus est détaillé par GIBSON dans [Gib96]. Cependant, il est frustrant de constater que la complexité de l'algorithme de recherche d'une matrice $\mathbf{B}'$ candidate repose sur des considérations purement heuristiques. La complexité déterminée par GIBSON est $O((k + t_1 + 2)2^{d(k+2)})$ multiplications dans $\mathbf{F}_{q^m}$ où d est le rang moyen des matrices

$$\mathbf{Q}^{-1} - (\mathbf{Q}^{-1})^{[1]}.$$

Le fait qu'on doive utiliser des heuristiques provient de ce que ces matrices dépendent de la matrice $\mathbf{B}'$ que l'on recherche. GIBSON prétend qu'en moyenne le paramètre d vaut

$$d = \max(0, t_1 - 2s)$$

où s est le rang sur $\mathbf{F}_{q^m}$ de la matrice $\mathbf{A}$ partie de la matrice de distorsion qui s'écrit $\mathbf{X} = \mathbf{AB}$.

GIBSON a implémenté la seconde attaque pour de petits paramètres, mais du fait du grand nombre d'heuristiques intervenant, il est extrêmement difficile d'évaluer réellement le coût total de cette attaque.

2.3 Un cryptosystème de type NIEDERREITER fondé sur la métrique rang

Les attaques de GIBSON utilisent des propriétés algébriques propres aux codes de GABIDULIN. Partant, on peut tenter de les contrer en utilisant des familles de codes moins *optimales* du point de vue des paramètres, mais plus résistantes. GABIDULIN et OURIVSKI ont proposé plusieurs modifications du cryptosystème original. En prônant par exemple l'utilisation de sous-codes – voir [GO00] – ils sont parvenus à élaborer des familles de codes dérivés de codes de GABIDULIN qui résistent aux *attaques de GIBSON*.

De plus avec Thierry BERGER nous avons montré dans [BL00] qu'on pouvait concevoir une version du système de type NIEDERREITER. Cette version présente des avantages par rapport au système original. On peut d'une part réduire la taille de la *clé publique* en ne publiant que la partie redondante de la matrice, et d'autre part, il existe un moyen naturel très rapide et apparemment sûr de coder les messages afin de les utiliser dans le système. De cette façon, on résout un des problèmes fondamentaux des systèmes de chiffrement de type NIEDERREITER, qui est celui de l'encodage des messages. Dans cette section, je présente dans un premier temps le système de chiffrement de NIEDERREITER utilisant la *métrique rang*. Dans un second temps, je décris la procédure permettant d'encoder puis de decoder des messages afin qu'ils soient utilisables dans le cryptosystème. Enfin je montre qu'en utilisant des sous-codes de codes de GABIDULIN bien choisis, on peut rendre les attaques de GIBSON inopérantes.

- *Clé privée* : un code C tiré aléatoirement dans la famille $\mathcal{G}$, appelé *code caché*, donné par une matrice de parité $\mathbf{H}$, de taille $n \times (n-k)$ à coefficients dans $\mathbf{F}_{q^m}$, et une matrice $\mathbf{S}$ sur $\mathbf{F}_{q^m}$ inversible, de taille $(n-k) \times (n-k)$.
- *Clé publique* : la matrice

$$\mathbf{H}_{\text{pub}} = \mathbf{S}\mathbf{H}.$$

Les procédures de chiffrement et de déchiffrement sont les suivantes :

1. *Chiffrement* : soit $\mathbf{e} = (e_1, e_2, \dots, e_n)$ un mot de $\mathbf{F}_{q^m}$, de longueur n , et de rang inférieur à t . L'émetteur envoie le mot

$$\mathbf{e}' = \mathbf{e}\mathbf{H}_{\text{pub}}^T.$$

2. *Déchiffrement* : le destinataire calcule $\mathbf{S}^{-1}\mathbf{e}' = \mathbf{e}\mathbf{H}^T$, puis il décode $\mathbf{e}$ en utilisant l'algorithme de décodage du code C .

TABLEAU 4.5: Cryptosystème de NIEDERREITER fondé sur le décodage de la métrique rang

2.3.1 Description du système

Le système que nous avons proposé avec Thierry BERGER – voir [BL00] – repose sur la constatation que les systèmes de chiffrement de type NIEDERREITER, qui utilisent comme *clé publique* des *matrices de parité* de codes, ont en général de meilleurs paramètres que les systèmes de type MCELIECE qui se servent de *matrices génératrices* de codes. De plus, dans les systèmes de chiffrement de type NIEDERREITER, l'émetteur n'a pas besoin de disposer d'un générateur pseudo-aléatoire puisque le chiffrement est déterministe. Pour plus de détails, le lecteur pourra se reporter à la section 1 du chapitre 1.

Soit $\mathcal{G}$ une famille de codes de longueur n sur l'alphabet $\mathbf{F}_{q^m}$, de dimension k pour lesquels on dispose d'algorithmes polynomiaux de décodage jusqu'à la *distance rang* t . La famille $\mathcal{G}$ constitue l'*espace des clés* du système. Le principe du cryptosystème est décrit dans le tableau 4.5.

Il comporte trois avantages immédiats sur le cryptosystème GPT.

1. Il n'y a pas de *matrice de distorsion*. La sécurité du système contre les *attaques structurelles* repose exclusivement sur les codes constituant l'*espace des clés* $\mathcal{G}$.
2. Il n'y a pas de générateur pseudo-aléatoire. Le chiffrement étant par nature déterministe, deux mots identiques sont chiffrés de manière identique. Ce cryptosystème est donc résistant aux *attaques actives* qui exploitent la nature non déterministe du cryptosystème GPT.
3. Si l'on met la matrice $\mathbf{H}_{\text{pub}}$ sous la forme systématique

$$\mathbf{H}_{\text{pub}} = (\mathbf{I} \mid \mathbf{R}),$$

on peut, sans affecter la sécurité du système, ne publier que la *matrice de redondance* $\mathbf{R}$. La taille de la *clé publique* est donc égale à $(n - k)km \log_2 q$ bits.

Exemple 16

Prenons comme *espace des clés*, la famille des *codes de GABIDULIN* de longueur $n = 32$ sur le corps $\mathbf{F}_{2^{32}}$, et de dimension $k = 16$. Si on publie la matrice $\mathbf{H}_{\text{pub}}$, alors la taille de la *clé publique* est égale à 17408 bits. Si, en revanche, le concepteur ne publie que la matrice de redondance, la taille de la *clé publique* tombe à 8192 bits.

Le *taux de transmission théorique* du cryptosystème GPT est simplement égal à k/n , si on considère des codes de longueur n et de dimension k . Dans le cas du cryptosystème GPT, la quantité d'information transmise effectivement est égale à la quantité d'information transmissible. Par conséquent, le *taux de transmission théorique* est égal au *taux de transmission pratique*.

Dans le cas de la version NIEDERREITER du cryptosystème, la différence dépend de la façon dont les mots ont été encodés. Le taux de transmission théorique est assez facile à calculer.

Proposition 31 (Taux de transmission théorique) *Soit un cryptosystème de type NIEDERREITER utilisant des codes de longueur n sur $\mathbf{F}_{q^m}$, de dimension k , et corrigeant t erreurs rangs. Alors le taux de transmission théorique du système est égal à*

$$\tau = \frac{\log_q N_{m,t} + \log_q N_{n,t} - \log_q N_{t,t}}{m(n - k)} \quad (4.46)$$

où $N_{m,t} = (q^m - 1)(q^m - q) \cdots (q^m - q^{t-1})$.

Preuve

Tout mot $\mathbf{e}$ de longueur n et de rang t à coefficients dans $\mathbf{F}_{q^m}$ s'écrit sous la forme

$$\mathbf{e} = (E_1, E_2, \dots, E_t)\mathbf{S},$$

où $E_1, E_2, \dots, E_t \in \mathbf{F}_{q^m}^t$ forment une famille libre sur $\mathbf{F}_q$, et où $\mathbf{S}$ est une matrice $t \times n$ de rang t à coefficients dans $\mathbf{F}_q$. Pour compter le nombre de mots de rang t , on compte d'abord le nombre de sous-espace vectoriels de dimension t de $\mathbf{F}_{q^m}^n$. Il y en a exactement

$$\frac{\prod_{i=0}^{t-1} (q^m - q^i)}{\prod_{i=0}^{t-1} (q^t - q^i)} = \frac{N_{m,t}}{N_{t,t}}.$$

et il y a exactement

$$(q^n - 1)(q^n - q) \cdots (q^n - q^{t-1}) = N_{m,t}$$

choix possibles pour la matrice $\mathbf{S}$ à coefficients dans $\mathbf{F}_q$, de taille $t \times n$ et de rang t .

Donc le logarithme en base q du nombre de mots de longueur n et de rang t est égal à

$$\text{Log}_q N_{m,t} + \text{Log}_q N_{n,t} - \text{Log}_q N_{t,t}.$$

Comme, par définition, le *taux de transmission théorique* est égal au rapport entre le logarithme en base q du nombre de mots possibles et le logarithme en base q du nombre de mots sur $\mathbf{F}_{q^m}$ de longueur $n - k$, on en déduit la formule (4.46).

■

Exemple 17

En prenant comme *code caché* du cryptosystème un code de GABIDULIN sur $\mathbf{F}_{232}$, de longueur 32, et de dimension 16, alors le *taux de transmission théorique* du système est $\tau \approx 0,757$.

2.3.2 Encodage et décodage des messages

Quoique le cryptosystème décrit dans le tableau 4.5 présente certains avantages comparé au cryptosystème GPT, comme pour le cryptosystème de NIEDERREITER classique, il reste un inconvénient non négligeable. En effet, étant donné un message que l'on doit chiffrer, il faut d'abord encoder celui-ci en blocs de longueur n sur $\mathbf{F}_{q^m}$ et de rang égal à t . Une fois que le message est encodé, on peut le chiffrer. Réciproquement, après déchiffrement, le récepteur a des blocs de longueur n sur $\mathbf{F}_{q^m}$, et de rang t . Il faut donc décoder les blocs pour enfin recouvrer le message initial.

De plus, les procédures d'encodage et de décodage doivent au moins vérifier les trois propriétés suivantes :

- Le *taux de transmission pratique* du cryptosystème doit être le plus proche possible du *taux de transmission théorique* du cryptosystème.
- Les complexités des procédures d'encodage et de décodage doivent être négligeables comparées aux complexités des procédures de chiffrement et de déchiffrement.
- La procédure d'encodage doit *diffuser* l'information. Cette condition est un prérequis.

Dorénavant nous considérons le cas où $q = 2$, qui est le cas pratique. Sans perte de généralité, on peut supposer que la base de taille m de l'extension $\mathbf{F}_{2^m}/\mathbf{F}_2$ dans laquelle les calculs sont effectués est publique.

Procédure d'encodage :

Le message binaire que l'on doit encoder est tout d'abord découpé en blocs de longueur $l = t(m + n - 2t)$ bits. Soit $\mathbf{f}$ un de ces blocs. Le bloc $\mathbf{f}$ est scindé en deux blocs $\mathbf{f}_1$ et $\mathbf{f}_2$ de longueur respective $t(n - t)$ bits et $(m - t)t$ bits.

1. Le bloc $\mathbf{f}_1 = (f_{1,1}, f_{1,2}, \dots, f_{1,t(n-t)})$ de longueur $t(n-t)$ est transformé en une matrice de taille $t \times (n-t)$:

$$\mathbf{F}_1 = \begin{pmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,n-t} \\ f_{1,n-t+1} & f_{1,n-t+2} & \cdots & f_{1,2(n-t)} \\ \vdots & \vdots & \ddots & \vdots \\ f_{1,(t-1)(n-t)+1} & f_{1,(t-1)(n-t)+2} & \cdots & f_{1,t(n-t)} \end{pmatrix}.$$

2. Le bloc $\mathbf{f}_2 = (f_{2,1}, f_{2,2}, \dots, f_{2,(m-t)t})$ de longueur $(m-t)t$ est transformé en une matrice de taille $(m-t) \times t$:

$$\mathbf{F}_2 = \begin{pmatrix} f_{2,1} & f_{2,2} & \cdots & f_{2,t} \\ f_{2,t+1} & f_{2,n-t+2} & \cdots & f_{2,2t} \\ \vdots & \vdots & \ddots & \vdots \\ f_{2,(m-t-1)t+1} & f_{2,(m-t-1)t+2} & \cdots & f_{2,(m-t)t} \end{pmatrix}.$$

3. La *matrice d'encodage* la matrice binaire de taille $m \times n$:

$$\mathbf{F} = \left(\begin{array}{c|c} \mathbf{I}_{t \times t} & \mathbf{F}_1 \\ \hline & \\ \mathbf{F}_2 & \mathbf{F}_2 \mathbf{F}_1 \end{array} \right). \quad (4.47)$$

On voit facilement que $\mathbf{F}$ est de rang t .

4. En utilisant la base publique de taille m de l'extension $\mathbf{F}_{2^m}/\mathbf{F}_2$, on transforme la matrice $\mathbf{F}$ en un mot $\mathbf{e}$ de longueur n et de rang t .

Procédure de décodage :

Après déchiffrement, le récepteur dispose d'un mot $\mathbf{e}$ sur $\mathbf{F}_{2^m}$, de longueur n et de rang t . La base de $\mathbf{F}_{2^m}/\mathbf{F}_2$ étant connue, on détermine la *matrice d'encodage* $\mathbf{F}$, en étendant chaque coordonnée de $\mathbf{e}$ sur cette base. Partant, de la forme (4.47) de la matrice $\mathbf{F}$, on détermine les matrices $\mathbf{F}_1$ et $\mathbf{F}_2$. À partir de $\mathbf{F}_1$ et $\mathbf{F}_2$, on déduit les blocs binaires $\mathbf{f}_1$ et $\mathbf{f}_2$, puis le bloc initial $\mathbf{f} = (\mathbf{f}_1 \mid \mathbf{f}_2)$.

La seule étape de la procédure d'encodage qui ne se fait pas en temps linéaire est la multiplication $\mathbf{F}_2 \mathbf{F}_1$ des matrices binaires $\mathbf{F}_1$ de taille $t \times (n-t)$ et $\mathbf{F}_2$ de taille $(m-t) \times t$. Celle-ci prend en moyenne

$$\frac{t(m-t)(n-t)}{2} \text{ opérations binaires.}$$

Taux de transmission pratique :

Le nombre de bits d'information transmissibles en utilisant la procédure d'encodage est égal à la longueur $l = t(m+n-2t)$ d'un bloc. Le *taux de transmission pratique* τ_p est donc égal à

$$\tau_p = \frac{t(m+n-2t)}{m(n-k)}.$$

τ_p est très proche du *taux de transmission théorique* du système τ . En effet, supposons que t est petit comparé à m et à n , ce qui est le cas en pratique. Alors mt et nt sont de bonnes approximations respectivement de

$$\text{Log}_2 N_{m,t}, \quad \text{et de} \quad \text{Log}_2 N_{n,t}.$$

Avec ces approximations, le *taux de transmission théorique* τ du cryptosystème, donné dans la proposition 31 donnant est égal à

$$\tau = \frac{mt + nt - \text{Log}_2 N_{t,t}}{m(n-k)}.$$

Donc

$$\tau - \tau_p \approx \frac{2t - \text{Log}_2 N_{t,t}}{m(n-k)}.$$

Exemple 18

Si on prend comme paramètres

$$m = n = 32, \quad k = 16, \quad t = 8,$$

le taux de transmission pratique du cryptosystème est égal à $\tau_p = 0,75$. Ce résultat est très proche du *taux de transmission théorique* atteignable $\tau \approx 0,757$ de l'exemple 17.

Les procédures d'encodage et de décodage vérifient donc les deux premières propriétés requises dans l'introduction de cette section. Cependant, d'un point de vue cryptographique je ne sais pas dire grand chose, sinon que la procédure d'encodage semble bien diffuser l'information dans tout le bloc encodé. En effet, quand on encode un message, l'information relative au message est diffusée sur les lignes des matrices $\mathbf{F}_1$ et $\mathbf{F}_2$. Partant cette information est diffusée dans toute la matrice $\mathbf{F}$.

2.3.3 Quels codes utiliser ?

Pour jouer le rôle de l'espace des clés du cryptosystème décrit dans le tableau 4.5, on peut naturellement utiliser les *codes de GABIDULIN*, puisque, d'une part ils disposent d'un algorithme polynomial de décodage, et d'autre part ce sont les codes ayant les meilleurs paramètres pour résister aux *attaques par décodage*.

Comme les codes de GABIDULIN sont stables par dualité – voir le théorème 10 –, la *matrice de parité* d'un code de GABIDULIN est également *matrice génératrice* d'un code de GABIDULIN. Donc, résoudre l'équation

$$\mathbf{H}_{\text{pub}} = \mathbf{S}\mathbf{H}$$

où $\mathbf{H}$ est une matrice de parité d'un code de GABIDULIN, et $\mathbf{S}$ est une matrice inversible équivaut à retrouver le code dont une matrice de parité est $\mathbf{H}_{\text{pub}}$, quand la *matrice de distorsion* est nulle. On a vu dans la section 2.1.2 que

ce système se résolvait très facilement en $O(k^3)$ opérations dans le corps que l'on considère. La version NIEDERREITER peut donc souffrir d'*attaques structurelles* de type *attaques de GIBSON*.

Dans cette partie, je montre que l'on peut choisir un *espace des clés* constitué de sous-codes de *codes de GABIDULIN* de manière à rendre inefficace les *attaques de GIBSON*.

Soit $\mathbf{H} = \left(h_j^{[i]} \right)_{i=0, j=1}^{d-2, n}$ une matrice de parité d'un code de GABIDULIN de *distance rang* minimale d , de longueur n sur $\mathbf{F}_{q^m}$. Le code C de matrice de parité

$$\mathbf{H}' = \begin{pmatrix} h_1 & h_2 & \cdots & h_n \\ \vdots & \vdots & \ddots & \vdots \\ h_1^{[d-2]} & h_2^{[d-2]} & \cdots & h_n^{[d-2]} \\ a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rn} \end{pmatrix}$$

est un sous-code du code de matrice de parité $\mathbf{H}$. Il est donc de *distance rang* minimale supérieure à d . De plus, si les vecteurs $(a_{i1}, a_{i2}, \dots, a_{in})$ sont choisis linéairement indépendants et en dehors de l'espace vectoriel engendré par les lignes $(h_1^{[i]}, \dots, h_n^{[i]})$, alors la dimension de C est égale à $n - (d + r)$, et on a

Proposition 32 *L'algorithme polynomial de décodage des codes de GABIDULIN décrit dans la section 1.2.2 permet de décoder le code de matrice de parité $\mathbf{H}'$ jusqu'au rang $\lfloor (d - 1)/2 \rfloor$.*

Afin d'éviter les attaques de GIBSON, l'idée ici est de déstructurer la famille des codes de GABIDULIN constituant l'*espace des clés* en utilisant des sous-codes. Dans ce cas l'utilisation d'une matrice de *distorsion* est inutile.

Tout sous-code d'un code de GABIDULIN peut donc être utilisé dans la version NIEDERREITER du système de chiffrement, mais tous ne permettent pas de résister de manière efficace aux attaques possibles. Considérons le cryptosystème de *clé privée*

$$\mathbf{H}' = \begin{pmatrix} \mathbf{H} \\ \mathbf{A} \end{pmatrix}$$

où $\mathbf{H}$ est une *matrice de parité* de taille $(d - 1) \times n$ d'un *code de GABIDULIN* sur $\mathbf{F}_{q^m}$ de *distance rang* minimale d , $\mathbf{A} = (a_{ij})$ est une matrice $r \times n$ quelconque à coefficients dans $\mathbf{F}_{q^m}$.

Étant donnée la *clé publique* $\mathbf{H}_{\text{pub}} = (\mathbf{I}_{d+r-1} \mid \mathbf{R})$ mise sous forme systématique, on cherche à résoudre l'équation

$$(\mathbf{I}_{d+r-1} \mid \mathbf{R}) = \mathbf{S}\mathbf{H}', \quad (4.48)$$

où $\mathbf{S}$ est une matrice inversible de taille $(d-r+1) \times (d-r+1)$ à coefficients dans $\mathbf{F}_{q^m}$.

Si on note $\mathbf{H} = (\mathbf{H}_1 \mid \mathbf{H}_2)$, $\mathbf{A} = (\mathbf{A}_1 \mid \mathbf{A}_2)$ et $\mathbf{S} = (\mathbf{S}_1 \mid \mathbf{S}_2)$, où $\mathbf{H}_1$, $\mathbf{A}_1$, $\mathbf{S}_1$ désignent les matrices formées des $d+r-1$ premières colonnes des matrices respectives $\mathbf{H}$, $\mathbf{A}$ et $\mathbf{S}$, l'équation (4.48) devient

$$\mathbf{S}_1\mathbf{H}_2 + \mathbf{S}_2\mathbf{A}_2 = (\mathbf{S}_1\mathbf{H}_1 + \mathbf{S}_2\mathbf{A}_1)\mathbf{R}. \quad (4.49)$$

Quand on utilisait un *code de GABIDULIN* plutôt qu'un sous-code, un artifice permettait de supprimer la matrice $\mathbf{S}$ de l'équation – voir la section 2.1.2. Cependant, dans ce cas je ne connais pas de réduction permettant de relinéariser le système, même lorsque r est petit.

La résistance du système contre les *attaques structurelles* peut être évaluée en modifiant les *attaques de GIBSON*. En effet, la matrice de parité $\mathbf{H}'$ choisie comme *clé privée* est la somme d'une *matrice de parité* d'un code de GABIDULIN de *distance rang minimale* $d+r$ et d'une *matrice de distorsion* de la manière suivante :

$$\mathbf{H}' = \begin{pmatrix} h_1 & \cdots & h_n \\ \vdots & \ddots & \vdots \\ h_1^{[d-2]} & \cdots & h_n^{[d-2]} \\ h_1^{[d-1]} & \cdots & h_n^{[d-1]} \\ \vdots & \ddots & \vdots \\ h_1^{[d+r-2]} & \cdots & h_n^{[d+r-2]} \end{pmatrix} + \mathbf{Y}.$$

où $\mathbf{Y}$ est une matrice de rang r qui s'écrit

$$\mathbf{Y} = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \\ a_{11} - h_1^{[d-1]} & \cdots & a_{1n} - h_n^{[d-1]} \\ \vdots & \ddots & \vdots \\ a_{r1} - h_1^{[d+r-2]} & \cdots & a_{rn} - h_n^{[d+r-2]} \end{pmatrix}.$$

Avec ces notations, l'équation vérifiée par la *clé publique* $\mathbf{H}_{\text{pub}}$ devient

$$\mathbf{H}_{\text{pub}} = \mathbf{S}\mathbf{H}' + \underbrace{\mathbf{S}\mathbf{Y}}_{\mathbf{X}}.$$

La matrice $\mathbf{X} = \mathbf{S}\mathbf{Y}$ joue ici un rôle équivalent à celui de la *matrice de distorsion* dans la version GPT du système. Comme la matrice $\mathbf{S}$ de taille $(d+r-1) \times (d+r-1)$ est inversible, le rang de $\mathbf{X}$ est égal au rang de $\mathbf{Y}$, c'est-à-dire à r . Afin d'évaluer la complexité des *attaques de GIBSON*, je dois évaluer le *paramètre de distorsion* t_1 de cette matrice $\mathbf{X}$. Par définition, t_1 est égal à :

$$t_1 = \text{Max}_{\mathbf{c} \in \mathbf{F}_{q^m}^{d+r-1}} (\text{Rg}(\mathbf{cX})).$$

On peut donc rendre t_1 aussi grand qu'on veut. En effet, le paramètre t_1 est, par définition, supérieur au *rang* de la première ligne de la matrice $\mathbf{X}$. Puisque la matrice $\mathbf{S}$ est inversible, t_1 est également supérieur au *rang* de la première ligne de la matrice $\mathbf{Y}$ qui est le vecteur

$$\mathbf{a} = \left(a_{11} - h_1^{[d-1]}, a_{12} - h_2^{[d-1]}, \dots, a_{1n} - h_n^{[d-1]} \right).$$

Si on choisit les éléments a_{1j} de telle façon que la première ligne $\mathbf{a}$ de la matrice $\mathbf{Y}$ soit de *rang* maximum égal à n , on a $t_1 = n$. En conséquence, en évaluant la complexité des *attaques de GIBSON* contre ce système, on trouve :

- *Première attaque* : en supprimant les termes polynomiaux on obtient une complexité de l'ordre de q^{mn} multiplications dans $\mathbf{F}_{q^m}$.
- *Seconde attaque* : en supprimant les termes polynomiaux on a une complexité de l'ordre de $q^{(d+r+1)(n-2r)}$ multiplications dans $\mathbf{F}_{q^m}$.

Choix de paramètres

Les choix de paramètres que je propose sont expérimentaux et constituent selon moi un bon compromis entre la taille de la clé et la résistance aux diverses attaques – attaques par décodage et attaques de GIBSON.

1. Soit $\mathbf{H}$ une matrice de parité de taille $(d-1) \times n$ d'un *code de GABIDULIN* C de longueur $n = 32$ sur $\mathbf{F}_{2^{32}}$, de distance minimale $d = 11$.
2. Soit $\mathbf{Y}$ une matrice de taille $r \times n$ à coefficients dans $\mathbf{F}_{2^{32}}$, de rang $r = 5$, telle que la matrice

$$\begin{pmatrix} \mathbf{H} \\ \mathbf{Y} \end{pmatrix}$$

soit de rang $d + r - 1 = 15$.

Par conséquent, le code de longueur 32 sur $\mathbf{F}_{2^{32}}$, de *matrice de parité*

$$\mathbf{H}' = \begin{pmatrix} \mathbf{H} \\ \mathbf{Y} \end{pmatrix}$$

est un sous-code de dimension $n - (d + r - 1) = 17$ du code C . Donc il dispose d'un algorithme polynomial de décodage jusqu'à la *distance rang* $(d-1)/2 = 5$. Chaque élément du corps $\mathbf{F}_{2^{32}}$ peut être codé sur 32 bits, donc on a

1. *Taille de la clé publique* : $\mathbf{H}_{\text{pub}} = \mathbf{SH}'$ est une matrice de taille $(15 \times 32) \times 32 = 15360$ bits. Si on publie uniquement la *matrice de redondance* de taille $(d + r - 1) \times (n - d + r - 1)$, on peut ramener la taille de la *clé publique* à $(15 \times 17) \times 32 = 8160$ bits.

2. *Taux de transmission pratique* : on envoie dans le canal de transmission des mots de *norme rang* égale à $(d - 1)/2 = 5$. Si on utilise la procédure d'encodage décrite dans la section précédente, alors le *taux de transmission pratique* est égal à

$$\tau_p = 0,56.$$

Les diverses attaques contre le cryptosystème ont les complexités suivantes :

- *Attaque par énumération des bases* : pour cette attaque consistant à décoder un mot intercepté relativement au *code public*, on doit énumérer un espace de taille égale à $2^{(m-t-1)t} = 2^{130}$.
- *Première attaque de GIBSON* : recherche exhaustive dans un espace de $2^{nm} = 2^{1024}$ éléments.
- *Seconde attaque de GIBSON* : recherche exhaustive dans un espace de $2^{(d+1+r)(n-2r)} = 2^{289}$ éléments.

3 Conclusion

Depuis qu'elle fut introduite en 1985, la *métrique rang* n'a pas suscité de grand engouement dans la communauté des théoriciens de l'information. Outre les travaux de GABIDULIN sur les codes du même nom, leurs algorithmes de décodage, et quelques articles de ROTH traitant du sujet – voir [Gab85b, Gab85a, Gab91, Rot91, Rot96, Rot97] –, très peu de choses furent publiées sur la *métrique rang*. C'est donc une voie qui demeure ouverte et à explorer. D'autant plus que les problèmes de décodage relativement à cette métrique interviennent dans des problèmes de complexité aux multiples applications en cryptanalyse – comme le problème *MinRank*.

D'un point de vue purement cryptographique, les attaques de GIBSON montrent qu'il est assez facile de modifier le cryptosystème GPT afin de le rendre résistant aux *attaques structurelles* connues, en utilisant des sous-codes de codes de GABIDULIN convenables par exemple. D'autres propositions furent faites dans [GO00, GO01]. Ce cryptosystème semble donc constituer une alternative intéressante au cryptosystème de McELIECE. En effet, sa vitesse en chiffrement et en déchiffrement est équivalente, et les clés sont suffisamment petites pour pouvoir être implantées sur des supports à ressources limitées.

Conclusion et perspectives

Cette thèse m'a donné l'occasion de présenter et d'étudier deux systèmes de chiffrement à clé publique utilisant des familles de codes correcteurs, dont la sécurité repose sur des problèmes de décodage, l'un pour la métrique de HAMMING, l'autre pour la *métrique rang*.

Perspectives en métrique de HAMMING

Lors des trois premiers chapitres, j'ai introduit les cryptosystèmes de MCELIECE et de NIEDERREITER qui reposent sur le problème du *décodage borné* d'un code dans la *métrique de HAMMING*, la métrique «habituelle» en théorie des codes correcteurs. Les deux systèmes de chiffrement peuvent être cryptanalysés de deux manières différentes :

- Par les *attaques par décodage* consistant à décoder les messages chiffrés interceptés dans le *code public* avec des algorithmes de décodage généraux. Leur complexité implique que la taille de la clé publique doit être de l'ordre de plusieurs centaines de milliers de bits.
- Par les *attaques structurelles* qui consistent à exploiter la structure du *code public* afin d'en recouvrer un décodeur en temps polynomial. Leur efficacité dépend de la famille de codes utilisée dans la conception du cryptosystème.

Les cryptosystèmes construits avec des codes de GOPPA sont très résistants aux *attaques structurelles*. On ne connaît en effet aucun *invariant* de code qui permette de distinguer si un code aléatoire est équivalent à un code de GOPPA. Cependant, si les codes sont «mal choisis», par exemple si leur *groupe d'automorphismes* est non trivial, ils forment des ensembles de *clés faibles* des cryptosystèmes permettant, d'une part de réduire la taille de l'espace des clés à considérer, et d'autre part de réduire la complexité des tests en utilisant des codes dérivés de longueur et de dimension plus petite appelé *codes s -projetés*. D'un autre côté, ces codes de GOPPA particuliers, peuvent être utilisés afin d'accroître la résistance des cryptosystèmes contre les *attaques par décodage*. Dans ce cas en effet, on peut construire des ensembles de motifs d'erreurs corrigibles de grand poids, appelés *ensembles t -décodables par tour*.

En 1994, V. M. SIDEL'NIKOV proposa un système de chiffrement alternatif, de conception analogue au cryptosystème de MCELIECE [Sid94]. Celui-ci

utilise comme clé publique un code de REED–MULLER permuté, et comme algorithme de décodage, l'algorithme [SP92] qui permet de décoder de manière probabiliste au delà de la capacité de correction du code. Cependant, la *capacité de correction* du code de REED–MULLER étant petite, la taille du code considéré doit être suffisamment grande afin de résister aux *attaques par décodage*.

Le futur de ces cryptosystèmes dépend d'une part de leur résistance aux différentes attaques, et d'autre part dépend de leur adaptabilité aux contraintes de la cryptographie moderne. Des *schémas d'identification* reposant sur le problème du *décodage borné* en métrique de HAMMING existent. La première version fut présentée par HARARI dans [Har88], et ceux qui résistent encore à la cryptanalyse peuvent être trouvés dans [Ste93], et dans [Vér95]. La taille des clés à stocker et le nombre de bits échangés dans ces schémas est très important. Jusqu'à une période très récente, on ne savait pas élaborer de signatures numériques utilisant des codes correcteurs. COURTOIS, FINIASZ et SENDRIER viennent d'en proposer une utilisant les propriétés des codes de GOPPA [CFS01].

En définitive, le problème majeur de ces cryptosystèmes reste la taille des paramètres qui empêche toute implantation sur des supports à ressource limitée. Afin de pallier ce problème, on peut envisager d'utiliser des propriétés de structures de codes pour l'atténuer.

Perspectives en métrique rang

Dans le chapitre 4, j'ai introduit les éléments essentiels de la métrique rang, ainsi que le cryptosystème GPT dont la sécurité repose sur le problème du *décodage borné* d'un code dans la *métrique rang*. La propriété de décodage étant intrinsèque à la métrique choisie, le cryptosystème est résistant aux *attaques par décodage* décrites dans le chapitre 1. Le système de chiffrement peut néanmoins être cryptanalysé de deux manières différentes :

- Par les attaques par *énumération des bases*, qui sont les équivalentes, pour la *métrique rang* des *attaques par décodage*. Elles consistent à décoder les messages chiffrés interceptés en énumérant les *vecteurs d'erreurs* possibles. Partant, en prenant une *clé publique* de quelques milliers de bits seulement, le facteur de travail de ces attaques est impraticable.
- Par les *attaques structurelles*. Dans ce cas, elles se résument aux attaques de GIBSON, qui exploitent des particularités des codes de GABIDULIN. Bien qu'elles soient de complexité exponentielle, il est nécessaire d'utiliser des clés publiques de grande taille afin de s'en prémunir. Cependant, si on utilise des sous-codes de codes de GABIDULIN pour perturber la structure, ces attaques deviennent impraticables.

Les problématiques attachées aux systèmes de chiffrement fondés sur le *décodage rang* demeurent les mêmes qu'en métrique de HAMMING. Un

schéma d'identification fut proposé par CHEN en 1996 – voir [Che96]–, et cryptanalysé par CHABAUD et STERN dans [CS96]. La cryptanalyse est néanmoins de complexité exponentielle, et échoue lorsqu'on augmente les paramètres. Il reste à considérer les possibilités autour du développement de signatures numériques, en utilisant les propriétés de la *métrique rang*.

Que choisir ?

La réponse à cette question n'est pas triviale. C'est plutôt une profession de foi. En effet, on est naturellement incité à privilégier le cryptosystème GPT par rapport aux cryptosystèmes de McELIECE et de NIEDERREITER. Les vitesses en chiffrement et en déchiffrement sont équivalentes, le taux de transmission est identique, mais la taille de la *clé publique* du cryptosystème GPT est de 30 à 40 fois plus petite, pour une sécurité équivalente. Donc pour des implantations sur des supports à ressources limitées – comme des puces électroniques – le cryptosystème GPT s'impose.

À l'inverse, le cryptosystème de McELIECE résiste depuis plus de vingt ans aux tentatives de cryptanalyse. La littérature concernant les codes de GOPPA, les problèmes de décodage en métrique de HAMMING, est très abondante. La conviction en la sécurité du cryptosystème de McELIECE s'appuie sur beaucoup plus de faits avérés. De plus, COURTOIS, FINIASZ, et SENDERIER ont récemment montré comment élaborer des signatures courtes avec des codes de GOPPA.

En conclusion, je me refuse à trancher et je dirai simplement que les deux types de cryptosystèmes constituent une alternative à ne pas négliger aux systèmes de chiffrement à *clé publique* déjà existants.

Annexe A

Factorisation des trinômes sur $\mathbf{F}_3$

1 Préliminaires

1.1 Discriminant de polynômes et factorisation

Définition 29 (Discriminant d'un polynôme) Soit $\mathbf{F}_{p^m}$ le corps fini de caractéristique p pair ou impair contenant p^m éléments, soit $f(x) \in \mathbf{F}_{p^m}[x]$ et $\mathbf{F}_{p^{mt}}$ le corps de décomposition de f . Alors le discriminant du polynôme f est la quantité $D(f) \in \mathbf{F}_{p^{mt}}$ telle que

$$D(f) = (-1)^{n(n-1)/2} \prod_{\alpha_i \neq \alpha_j} (\alpha_i - \alpha_j),$$

où $(\alpha_i)_{i=1}^n \in \mathbf{F}_{p^{mt}}$ sont les racines de f comptées avec multiplicité.

De là on déduit les propriétés suivantes du discriminant :

- Comme $D(f)$ est une fonction symétrique des racines de f , $D(f)$ appartient au corps contenant les coefficients de f , *i.e.*

$$D(f) \in \mathbf{F}_{p^m} \subset \mathbf{F}_{p^{mt}}.$$

- $D(f) = 0 \iff f$ a des racines multiples.

Il existe un lien étroit entre la valeur du discriminant d'un polynôme et la parité du nombre de facteurs dans sa factorisation. Le résultat le plus général fut prouvé par SWAN dans [Swa62]. En appliquant ce théorème aux polynômes sur $\mathbf{F}_{p^m}$ où p est impair, on a

Théorème 13 Soit $\mathbf{F}_{p^m}$ un corps fini de caractéristique impaire, soit $f(x) \in \mathbf{F}_{p^m}[x]$ de degré n , soit r le nombre de facteurs irréductibles de $f(x)$ sur $\mathbf{F}_{p^m}$, alors

$$r \equiv n \pmod{2} \iff D(f) \text{ est un carré dans } \mathbf{F}_{p^m}.$$

Étant donné un polynôme sur un corps fini quelconque, il n'est en général pas possible de déterminer une formule explicite pour son discriminant. Cependant, dans le cas où le polynôme $f(x)$ est de la forme

$$f(x) = x^n + ax^k + b,$$

on a le résultat suivant [Swa62]

Théorème 14 *Soit $n > k > 0$, $d = \text{pgcd}(n, k)$ et $n = n_1d$, $k = k_1d$, alors*

$$D(x^n + ax^k + b) = (-1)^{n(n-1)/2} b^{k-1} \left[n^{n_1} b^{n_1 - k_1} - (-1)^{n_1} (n - k)^{n_1 - k_1} k^{k_1} a^{n_1} \right]^d.$$

2 Discriminant des trinômes sur $\mathbf{F}_3$

Dans la suite, je traite des trinômes à coefficients dans le corps de caractéristique impaire $\mathbf{F}_3$. D'après le théorème 13, afin d'évaluer la parité du nombre de facteurs des trinômes à coefficients dans $\mathbf{F}_3$, on doit évaluer le discriminant des trinômes dans $\mathbf{F}_3$. Dans le cas des trinômes, on utilise la formule générale donnée dans le théorème 14.

Il y a exactement quatre polynômes unitaires sur $\mathbf{F}_3 = \{0, 1, -1\}$ qui sont :

$$x^n + x^k + 1, \quad x^n + x^k - 1, \quad x^n - x^k + 1, \quad x^n - x^k - 1.$$

D'après le théorème 14,

$$D(x^n + ax^k + b) = (-1)^{n(n-1)/2} b^{k-1} \left[n^{n_1} b^{n_1 - k_1} - (-1)^{n_1} (n - k)^{n_1 - k_1} k^{k_1} a^{n_1} \right]^d.$$

Afin de calculer le discriminant à valeurs dans $\mathbf{F}_3$, je considère

1. les valeurs de a , b dans $\mathbf{F}_3$, c'est-à-dire, le type du trinôme,
2. les valeurs prises par n et k , considérés comme des éléments de $\mathbf{F}_3$, *i.e.* les entiers n et k réduits modulo 3,
3. la valeur de $(-1)^{n(n-1)/2}$ qui dépend de la valeur de n modulo 4,
4. la parité de d , le pgcd de n et de k . Celui-ci dépend de la parité de n et de k , en effet

$$d \equiv 0 \pmod{2} \iff \begin{cases} n \equiv 0 \pmod{2}, \\ k \equiv 0 \pmod{2}, \end{cases}$$

5. la parité de n_1 et de k_1 où $n = dn_1$ et $k = dk_1$.

Afin de déterminer la parité de ces paramètres, j'introduis ici la notion de *valuation binaire* sur des entiers.

Définition 30 (Valuation binaire) *Soit n un entier et soit la quantité v_2 définie par*

$$\begin{cases} n = 2^{v_2(n)} n_2, \\ n_2 \text{ est impair}, \end{cases}$$

$v_2(n)$ est appelée *valuation binaire* de n .

À partir de cette définition, on a la proposition suivante donnant des informations sur la parité de n_1 , k_1 , $n_1 - k_1$.

Proposition 33 *Soit $n = n_1d$, $k = k_1d$, avec $d = \text{pgcd}(n, k)$. Alors*

- n_1 est pair $\iff v_2(n) > v_2(k)$,
- k_1 est pair $\iff v_2(n) < v_2(k)$,
- $n_1 - k_1$ est pair $\iff v_2(n) = v_2(k)$.

Les remarques précédentes indiquent que l'on doit séparer l'étude des trinômes sur $\mathbf{F}_3$ en plusieurs cas, chacun traitant non seulement de la valeur des paramètres a , b , n , k pris dans $\mathbf{F}_3$ mais aussi de la parité de n et de k ainsi que de leur valuation binaire.

3 Degré congru à 0 mod 3

Puisque le degré n des polynômes est tel que $n \equiv 0 \pmod{3}$, le terme $n^{n_1}b^{n_1-k_1}$ dans l'expression du discriminant disparaît, d'où

$$D(x^n + ax^k + b) = (-1)^{n(n-1)/2} b^{k-1} (-1)^{d-k} k^n \cdot a^n$$

Comme a et b sont des éléments non nuls de $\mathbf{F}_3$, le discriminant $D(x^n + ax^k + b)$ est égal à 0 si et seulement si k est un multiple de 3, d'où

Proposition 34 *Si $n \equiv 0 \pmod{3}$, alors le trinôme $x^n + ax^k + b$ a des racines multiples si et seulement si $k \equiv 0 \pmod{3}$.*

Dans tous les autres cas, on doit scinder les trinômes en fonction des valeurs de n modulo 12 et de k modulo 6. Dans la section suivante, je suppose que k n'est pas nul dans $\mathbf{F}_3$.

Théorème 15 *Si $n \equiv 0 \pmod{3}$ et $k \not\equiv 0 \pmod{3}$, alors*

- $D(x^n + x^k + 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 0 \pmod{12}, & k \equiv 1, 2 \pmod{3}, \\ n \equiv 3 \pmod{12}, & k \equiv 4, 5 \pmod{6}, \\ n \equiv 9 \pmod{12}, & k \equiv 1, 2 \pmod{6}. \end{cases}$$

- $D(x^n + x^k - 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 0 \pmod{12}, & k \equiv 1, 5 \pmod{6}, \\ n \equiv 3 \pmod{12}, & k \equiv 2 \pmod{3}, \\ n \equiv 6 \pmod{12}, & k \equiv 2, 4 \pmod{6}, \\ n \equiv 9 \pmod{12}, & k \equiv 1 \pmod{3}. \end{cases}$$

- $D(x^n - x^k + 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 0 \pmod{12}, & k \equiv 1, 2 \pmod{3}, \\ n \equiv 3 \pmod{12}, & k \equiv 1, 2 \pmod{3}, \\ n \equiv 9 \pmod{12}, & k \equiv 4, 5 \pmod{6}. \end{cases}$$

– $D(x^n - x^k - 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 0 \pmod{12}, & k \equiv 1, 5 \pmod{6}, \\ n \equiv 3 \pmod{12}, & k \equiv 1 \pmod{3}, \\ n \equiv 6 \pmod{12}, & k \equiv 2, 4 \pmod{6}, \\ n \equiv 9 \pmod{12}, & k \equiv 2 \pmod{3}. \end{cases}$$

Preuve

J'étudie tout d'abord le trinôme $T_1 = x^n + x^k + 1$ en distinguant deux cas.

1. Si n est pair c'est-à-dire si $n \equiv 0 \pmod{12}$ ou $n \equiv 6 \pmod{12}$. Comme, k n'est pas nul dans $\mathbf{F}_3$, alors $k^n \equiv 1 \pmod{3}$ quel que soit k . Comme d est pair si et seulement si k est pair, $d - k$ est pair pour tout k . Partant,

$$D(T_1) = (-1)^{n(n-1)/2}.$$

Il s'ensuit que $D(T_1)$ est égal à 1 si et seulement si $n \equiv 0 \pmod{12}$.

2. Si n est impair, alors $k^n \equiv k \pmod{3}$, et le *pgcd* d , de n et de k , est impair également, d'où

$$D(T_1) = (-1)^{n(n-1)/2} \cdot (-1)^{k+1} \cdot k,$$

et comme k n'est pas nul dans $\mathbf{F}_3$,

$$D(T_1) = 1 \iff \begin{cases} n \equiv 3 \pmod{12}, & k \equiv 4, 5 \pmod{6}, \\ n \equiv 9 \pmod{12}, & k \equiv 1, 2 \pmod{6}. \end{cases}$$

Comme $D(x^n + ax^k + b) = b^{k-1} a^n D(T_1)$, je déduis le résultat demandé pour les autres trinômes de $D(T_1)$.

■

4 Degré congru à 1 mod 3

Dans ce cas il n'y a aucune simplification de l'expression du discriminant. On doit étudier sa valeur pour chaque trinôme possible. En effet le discriminant s'écrit alors

$$D(x^n + ax^k + b) = (-1)^{n(n-1)/2} b^{k-1} \left[b^{n_1-k_1} - (-1)^{n_1} (1-k)^{n_1-k_1} k^{k_1} a^{n_1} \right]^d,$$

où d est le *pgcd* de n et de k , et où $n = n_1 d$, $k = k_1 d$. La valeur du discriminant dépend ainsi non seulement de la valeur de n et de k modulo 3 mais aussi de la parité de n_1 , k_1 , d , c'est-à-dire de la valuation binaire v_2 de n et de k .

4.1 Trinôme $x^n + x^k + 1$

Pour le trinôme $x^n + x^k + 1$, le discriminant est

$$D(x^n + x^k + 1) = (-1)^{n(n-1)/2} \left[1 - (-1)^{n_1} (1-k)^{n_1-k_1} k^{k_1} \right]^d.$$

Théorème 16 *Si $n \equiv 1 \pmod{3}$, alors*

- $x^n + x^k + 1$ a des racines multiples si et seulement si $D(x^n + x^k + 1) = 0$,
i.e. si et seulement si

$$k \equiv 2 \pmod{3}.$$

- Dans tous les autres cas, $D(x^n + x^k + 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 1 \pmod{12}, & k \equiv 0, 1 \pmod{3}, \\ n \equiv 4 \pmod{12}, & k \equiv 0, 1 \pmod{3}. \end{cases}$$

Preuve

La preuve est scindée en deux sous-cas.

1. Cas $k \equiv 2 \pmod{3}$

$$D(x^n + x^k + 1) = (-1)^{n(n-1)/2} \left[1 - (-1)^{n_1} (-1)^{n_1-k_1} (-1)^{k_1} \right]^d = 0.$$

Partant, $x^n + x^k + 1$ a des racines multiples si et seulement si $k \equiv 2 \pmod{3}$.

2. Cas $k \equiv 0, 1 \pmod{3}$

Ou bien k est égal à 0 dans $\mathbf{F}_3$, ou bien $1-k$ est égal à 0 dans $\mathbf{F}_3$.
Donc le terme $(-1)^{n_1} (1-k)^{n_1-k_1} k^{k_1}$ disparaît dans l'expression
du discriminant. D'où

$$D(x^n + x^k + 1) = (-1)^{n(n-1)/2}.$$

Le discriminant n'est donc jamais nul et

$$D(x^n + x^k + 1) = 1 \iff n \equiv 1, 4 \pmod{12}.$$

■

4.2 Trinôme $x^n + x^k - 1$

Pour ce trinôme, le discriminant devient

$$D(x^n + x^k - 1) = (-1)^{n(n-1)/2} (-1)^{k-1} \left[(-1)^{n_1-k_1} - (-1)^{n_1} (1-k)^{n_1-k_1} k^{k_1} \right]^d.$$

Le théorème suivant est calqué sur celui de la section 4.1.

Théorème 17 *Si $n \equiv 1 \pmod{3}$, alors*

- $x^n + x^k - 1$ a des racines multiples si et seulement si $D(x^n + x^k - 1) = 0$, ce qui est équivalent à

$$k \equiv 2 \pmod{3} \text{ et } v_2(k) = v_2(n).$$

où v_2 désigne la valuation binaire,

- dans tous les autres cas, $D(x^n + x^k - 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 1 \pmod{12}, & k \equiv 0, 1 \pmod{3}, \\ n \equiv 4 \pmod{12}, & k \equiv 5 \pmod{6}, \\ n \equiv 7 \pmod{12}, & k \equiv 2 \pmod{6}, \\ n \equiv 10 \pmod{12}, & k \equiv 0, 1 \pmod{3}, \quad k \equiv 2 \pmod{6}. \end{cases}$$

Preuve

La preuve est scindée en deux sous-cas.

1. *Cas $k \equiv 0, 1 \pmod{3}$.*

Ou bien k est nul dans $\mathbf{F}_3$, ou bien $1 - k$ est nul dans $\mathbf{F}_3$. Le terme $(-1)^{n_1}(1 - k)^{n_1 - k_1}k^{k_1}$ disparaît de l'expression du discriminant, et

$$D(x^n + x^k - 1) = (-1)^{n(n-1)/2}(-1)^{n-1}.$$

Le discriminant n'est donc jamais nul, et

$$D(x^n + x^k - 1) = 1 \iff n \equiv 1, 10 \pmod{12}.$$

2. *Cas $k \equiv 2 \pmod{3}$.*

Dans ce cas, on a

$$D(x^n + x^k - 1) = (-1)^{n(n-1)/2}(-1)^{k-1} \left[(-1)^{n_1 - k_1} - 1 \right]^d.$$

Donc $D(x^n + x^k - 1) = 0$ si et seulement si $n_1 - k_1$ pair c'est-à-dire, d'après la proposition 33, si et seulement si $v_2(k) = v_2(n)$.

Si $v_2(k) \neq v_2(n)$ alors $(-1)^{n_1 - k_1} - 1 = 1$ dans $\mathbf{F}_3$, et

$$D(x^n + x^k - 1) = (-1)^{n(n-1)/2}(-1)^{k-1}.$$

Dans ce cas,

$$D(x^n + x^k - 1) = 1 \iff \begin{cases} n \equiv 4 \pmod{12}, & k \equiv 5 \pmod{6}, \\ n \equiv 7 \pmod{12}, & k \equiv 2 \pmod{6}, \\ n \equiv 10 \pmod{12}, & k \equiv 2 \pmod{6}. \end{cases}$$

■

4.3 Trinôme $x^n - x^k + 1$

Pour ce trinôme, le discriminant devient

$$D(x^n - x^k + 1) = (-1)^{n(n-1)/2} \left[1 - (1-k)^{n_1-k_1} k^{k_1} \right]^d.$$

Théorème 18 *Si $n \equiv 1 \pmod{3}$, alors*

- $x^n - x^k + 1$ a des racines multiples si et seulement si $D(x^n - x^k + 1) = 0$, c'est-à-dire, si et seulement si

$$k \equiv 2 \pmod{3} \text{ et } v_2(k) < v_2(n),$$

où v_2 désigne la valuation binaire,

- dans tous les autres cas, $D(x^n - x^k + 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 1 \pmod{12}, & k \equiv 0, 1 \pmod{3}, \\ n \equiv 4 \pmod{12}, & k \equiv 0, 1 \pmod{3}, & k \equiv 2 \pmod{6}, \\ n \equiv 7 \pmod{12}, & & k \equiv 5 \pmod{6}. \end{cases}$$

Preuve

La preuve est scindée en deux sous-cas

1. *Cas $k \equiv 2 \pmod{3}$.*

$$D(x^n - x^k + 1) = (-1)^{n(n-1)/2} [1 - (-1)^{n_1}]^d.$$

Le discriminant est donc nul si et seulement si n_1 est pair. D'après la proposition 33, c'est équivalent à $v_2(k) < v_2(n)$.

Chaque fois que $v_2(k) \geq v_2(n)$, on a

$$D(x^n - x^k + 1) = (-1)^{n(n-1)/2} (-1)^d.$$

Comme d est pair, si et seulement si n et k sont pairs, lorsque $k \equiv 2 \pmod{3}$, on a

$$D(x^n - x^k + 1) = 1 \iff \begin{cases} n \equiv 4 \pmod{12}, & k \equiv 2 \pmod{6}, \\ n \equiv 7 \pmod{12}, & k \equiv 5 \pmod{6}. \end{cases}$$

2. *Cas $k \equiv 0, 1 \pmod{3}$.*

Ou bien k est nul dans $\mathbf{F}_3$, ou bien $1-k$ est nul dans $\mathbf{F}_3$. Donc le terme $(1-k)^{n_1-k_1} k^{k_1}$ disparaît de l'expression du discriminant

$$D(x^n - x^k + 1) = (-1)^{n(n-1)/2}.$$

Partant, le discriminant n'est jamais nul et

$$D(x^n - x^k + 1) = 1 \iff n \equiv 1, 4 \pmod{12}.$$

■

4.4 Trinôme $x^n - x^k - 1$

Pour ce trinôme, le discriminant est égal à

$$D(x^n - x^k - 1) = (-1)^{n(n-1)/2} (-1)^{k-1} \left[(-1)^{n_1-k_1} - (1-k)^{n_1-k_1} k^{k_1} \right]^d.$$

Théorème 19 *Si $n \equiv 1 \pmod{3}$, alors*

- $x^n - x^k - 1$ a des racines multiples si et seulement si $D(x^n - x^k - 1) = 0$,
c'est-à-dire, si et seulement si

$$k \equiv 2 \pmod{3} \text{ et } v_2(k) > v_2(n),$$

où v_2 désigne la valuation binaire,

- dans tous les autres cas, $D(x^n - x^k - 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 1 \pmod{12}, & k \equiv 0, 1 \pmod{3}, \\ n \equiv 4 \pmod{12}, & k \equiv 5 \pmod{6}, \\ n \equiv 7 \pmod{12}, & k \equiv 5 \pmod{6}, \\ n \equiv 10 \pmod{12}, & k \equiv 0, 1 \pmod{3}, \quad k \equiv 2 \pmod{6}. \end{cases}$$

Preuve

La preuve est scindée en deux sous-cas :

1. Cas $k \equiv 2 \pmod{3}$.

$$D(x^n - x^k - 1) = (-1)^{n(n-1)/2} (-1)^k \left[(-1)^{n_1-k_1} - (-1)^{n_1} \right]^d.$$

Le discriminant est nul si et seulement si k_1 est pair. D'après la proposition 33, ceci est équivalent à

$$v_2(k) > v_2(n).$$

Chaque fois que $v_2(k) \leq v_2(n)$, on a

$$D(x^n - x^k - 1) = (-1)^{n(n-1)/2} (-1)^{k-1} (-1)^n \left[(-1)^{k_1} - 1 \right]^d.$$

Comme k_1 est impair, on a $\left[(-1)^{k_1} - 1 \right]^d = (-2)^d = 1^d$. Par conséquent

$$D(x^n - x^k - 1) = (-1)^{n(n-1)/2} (-1)^{k-1} (-1)^n,$$

et

$$D(x^n - x^k - 1) = 1 \iff \begin{cases} n \equiv 4 \pmod{12}, & k \equiv 5 \pmod{6}, \\ n \equiv 7 \pmod{12}, & k \equiv 5 \pmod{6}, \\ n \equiv 10 \pmod{12}, & k \equiv 2 \pmod{6}. \end{cases}$$

2. Cas $k \equiv 0, 1 \pmod{3}$

Ou bien k est nul dans $\mathbf{F}_3$, ou bien $1 - k$ est nul dans $\mathbf{F}_3$. Donc le terme $(1 - k)^{n_1 - k_1} k^{k_1}$ disparaît de l'expression du discriminant, et

$$D(x^n - x^k - 1) = (-1)^{n(n-1)/2} (-1)^{k-1} (-1)^{n-k} = (-1)^{n(n-1)/2} (-1)^{n-1}$$

Le discriminant ne peut donc jamais être nul et ne dépend que de la valeur de n modulo 3. D'où

$$D(x^n - x^k - 1) = 1 \iff \begin{cases} n \equiv 1 \pmod{12}, & k \equiv 0, 1, \pmod{3} \\ n \equiv 10 \pmod{12}, & k \equiv 0, 1, \pmod{3} \end{cases}$$

■

5 Degré congru à 2 mod 3

Dans ce cas, on remplace la valeur de n dans l'expression du discriminant par la valeur -1 . Le discriminant devient alors

$$D(x^n + ax^k + b) = (-1)^{n(n-1)/2} b^{k-1} (-1)^n \left[b^{n_1 - k_1} - (-1)^{n_1 - k_1} (1 + k)^{n_1 - k_1} k^{k_1} a^{n_1} \right]^d,$$

où d est le *pgcd* de n et de k et où $n = n_1 d$, $k = k_1 d$. La valeur du discriminant ne dépend plus seulement des valeurs de n ni de k modulo 3, mais également de la parité de n_1 , k_1 , d , c'est-à-dire de la valuation binaire v_2 de n et de k .

Dans cette section, j'épargnerai au courageux lecteur qui est parvenu jusque là, les preuves techniques qui sont exactement du même tonneau que celles décrites dans la section précédente.

5.1 Trinôme $x^n + x^k + 1$

Pour le trinôme $x^n + x^k + 1$, le discriminant s'écrit

$$D(x^n + x^k + 1) = (-1)^{n(n-1)/2} (-1)^n \left[1 - (-1)^{n_1 - k_1} (1 + k)^{n_1 - k_1} k^{k_1} \right]^d.$$

Théorème 20 *Si $n \equiv 2 \pmod{3}$, alors*

- $x^n + x^k + 1$ a des racines multiples si et seulement si $D(x^n + x^k + 1) = 0$
c'est-à-dire si et seulement si

$$k \equiv 1 \pmod{3},$$

- dans tous les autres cas, $D(x^n + x^k + 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 8 \pmod{12}, & k \equiv 0, 2 \pmod{3}, \\ n \equiv 11 \pmod{12}, & k \equiv 0, 2 \pmod{3}. \end{cases}$$

5.2 Trinôme $x^n + x^k - 1$

Pour le trinôme $x^n + x^k - 1$, le discriminant s'écrit

$$D(x^n + x^k - 1) = (-1)^{n(n-1)/2+1} \left[1 - (1+k)^{n_1-k_1} k^{k_1} \right]^d.$$

Théorème 21 *Si $n \equiv 2 \pmod{3}$, alors*

- $x^n + x^k - 1$ a des racines multiples si et seulement si $D(x^n + x^k - 1) = 0$
c'est-à-dire si et seulement si

$$k \equiv 1 \pmod{3} \text{ et } v_2(k) = v_2(n),$$

où v_2 désigne la valuation binaire,

- dans tous les autres cas, $D(x^n + x^k - 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 2 \pmod{12}, & k \equiv 0, 2 \pmod{3}, & k \equiv 4 \pmod{6}, \\ n \equiv 5 \pmod{12}, & & k \equiv 4 \pmod{6}, \\ n \equiv 8 \pmod{12}, & & k \equiv 1 \pmod{6}, \\ n \equiv 11 \pmod{12}, & k \equiv 0, 2 \pmod{3}. \end{cases}$$

5.3 Trinôme $x^n - x^k + 1$

Pour le trinôme $x^n - x^k + 1$, le discriminant s'écrit

$$D(x^n - x^k + 1) = (-1)^{n(n-1)/2} (-1)^n \left[1 - (-1)^{k_1} (1+k)^{n_1-k_1} k^{k_1} \right]^d.$$

Théorème 22 *Si $n \equiv 2 \pmod{3}$, alors*

- $x^n - x^k + 1$ a des racines multiples si et seulement si $D(x^n - x^k + 1) = 0$
c'est-à-dire si et seulement si

$$k \equiv 1 \pmod{3} \text{ et } v_2(k) < v_2(n),$$

où v_2 désigne la valuation binaire,

- dans tous les autres cas, $D(x^n - x^k + 1) = 1$ si et seulement si

$$\begin{aligned} n \equiv 5 \pmod{12}, & \quad k \equiv 1 \pmod{3}, \\ n \equiv 8 \pmod{12}, & \quad k \equiv 0, 2 \pmod{3}, \quad k \equiv 4 \pmod{6}, \\ n \equiv 11 \pmod{12}, & \quad k \equiv 0, 2 \pmod{3}. \end{aligned}$$

5.4 Trinôme $x^n - x^k - 1$

Pour le trinôme $x^n - x^k - 1$, le discriminant s'écrit

$$D(x^n - x^k - 1) = (-1)^{n(n-1)/2} (-1)^{n+k-1} \left[(-1)^{n_1-k_1} - (-1)^{k_1} (1+k)^{n_1-k_1} k^{k_1} \right]^d.$$

Théorème 23 *Si $n \equiv 2 \pmod{3}$, alors*

- $x^n - x^k - 1$ a des racines multiples si et seulement si $D(x^n - x^k - 1) = 0$, c'est-à-dire si et seulement si

$$k \equiv 1 \pmod{3} \text{ et } v_2(k) > v_2(n),$$

où v_2 désigne la valuation binaire,

- dans tous les autres cas, $D(x^n - x^k - 1) = 1$ si et seulement si

$$\begin{cases} n \equiv 2 \pmod{12}, & k \equiv 0, 2 \pmod{3}, & k \equiv 4 \pmod{6}, \\ n \equiv 5 \pmod{12}, & & k \equiv 1 \pmod{6}, \\ n \equiv 8 \pmod{12}, & & k \equiv 1 \pmod{6}, \\ n \equiv 11 \pmod{12}, & k \equiv 0, 2 \pmod{3}. \end{cases}$$

6 Tests d'irréductibilité

Les résultats trouvés dans les sections précédentes peuvent être exploités afin d'obtenir un tableau donnant les valeurs de n et de k pour lesquelles existent des trinômes irréductibles à coefficients dans $\mathbf{F}_3$.

Parité du nombre de facteurs : Dans le corps $\mathbf{F}_3 = \{0, 1, -1\}$, la seule valeur non nulle qui est un carré est 1. Ainsi, en appliquant le théorème 13 aux polynômes de $\mathbf{F}_3[z]$, j'obtiens le corollaire suivant :

Corollaire 10 *Soit $f(z) \in \mathbf{F}_3[z]$ de degré n sans racines multiples, i.e. $D(f) \neq 0$, soit r le nombre de facteurs irréductibles de $f(z)$ sur $\mathbf{F}_3$, alors*

$$r \equiv n \pmod{2} \iff D(f) = 1.$$

Pour chaque trinôme $x^n + ax^k + b$ et pour chaque valeur de n , on construit le tableau A.1. Ce tableau donne les valeurs de k pour lesquelles le nombre de facteurs dans la factorisation de $x^n + ax^k + b$ sur $\mathbf{F}_3$ est impair.

Comme un polynôme irréductible n'a que lui-même comme diviseur non-trivial, le nombre de facteurs de sa factorisation est impair. Partant, les paramètres des trinômes candidats à l'irréductibilité peuvent être trouvés dans le tableau A.1.

Le test d'irréductibilité peut donc être quelque peu simplifié. Supposez que quelqu'un désire tester l'irréductibilité d'un trinôme $x^n + ax^k + b$ sur $\mathbf{F}_3$ tel que $D(x^n + ax^k + b)$ ne soit pas nul. Il procède de la façon suivante

1. si l'entrée k correspondant au trinôme $x^n + ax^k + b$ n'est pas dans la table alors $x^n + ax^k + b$ n'est pas irréductible.
2. si l'entrée k est dans la table, alors il applique l'un des algorithmes rapides qui testent l'irréductibilité des polynômes – l'algorithme de SHoup par exemple [Sho90].

Remarque 5

- La valeur 1 est racine du trinôme $x^n + x^k + 1$ sur $\mathbf{F}_3$ quels que soient k et n .
- Le tableau A.1 montre qu'il existe des valeurs du degré n pour lesquelles il n'existe pas de trinôme irréductible.

7 Trinômes primitifs

À partir des résultats précédents, je peux dériver des conditions sur la parité du degré n pour lesquelles un trinôme sur $\mathbf{F}_3$ est primitif. Dans cette section, je ne considère pas le trinôme $x^n + x^k + 1$ dont 1 est une racine triviale et qui n'est donc jamais primitif.

Le résultat général à propos de la *primitivité* d'un polynôme est le suivant [BGL94].

Théorème 24 Soit $f(x) = \sum_{i=0}^n f_i x^i$ un polynôme à coefficients dans $\mathbf{F}_q$ et soit

$$F(x) = \sum_{i=0}^n f_i x^{2^i - 1},$$

alors $f(x)$ est primitif sur $\mathbf{F}_q$ si et seulement si $F(x)$ est irréductible sur $\mathbf{F}_q$.

Dans le cas des trinômes à coefficients dans $\mathbf{F}_3$, le théorème 24 devient

Corollaire 11 Le trinôme $x^n + ax^k + b \in \mathbf{F}_3[x]$ est primitif si et seulement si le trinôme $x^{3^n - 1} + ax^{3^k - 1} + b$ est irréductible.

De ceci, je déduis la proposition suivante

Proposition 35

- Si $x^n + x^k - 1$ est primitif sur $\mathbf{F}_3$ alors n est pair,
- Si $x^n - x^k - 1$ est primitif sur $\mathbf{F}_3$ alors n est pair,
- Si $x^n - x^k + 1$ est primitif sur $\mathbf{F}_3$ alors n est impair.

Preuve

La preuve considère les trinômes $x^{3^n - 1} + ax^{3^k - 1} + b$, puis vérifie dans le tableau A.1 s'il existe des trinômes irréductibles possibles pour les paramètres $3^n - 1$ et $3^k - 1$. On utilise ensuite le corollaire 11.

Quel que soit k , on a $3^k \equiv 3 \pmod{6}$ et quel que soit n on a,

$$3^n \equiv 3 \pmod{12} \iff n \text{ impair}$$

on scinde alors les cas selon la parité de n .

n	$x^n + x^k + 1$	$x^n + x^k - 1$
0 mod 12	pas de k possible	$k \equiv 2, 4 \pmod{6}$
3 mod 12	$k \equiv 4, 5 \pmod{6}$	$k \equiv 2 \pmod{3}$
6 mod 12	$k \equiv 1, 2 \pmod{3}$	$k \equiv 1, 5 \pmod{6}$
9 mod 12	$k \equiv 1, 2 \pmod{6}$	$k \equiv 1 \pmod{3}$
1 mod 12	$k \equiv 0, 1 \pmod{3}$	$k \equiv 0, 1 \pmod{3}$
4 mod 12	pas de k possible	$k \equiv 0, 1 \pmod{3},$ $k \equiv 2 \pmod{6}, v_2(k) \neq v_2(n)$
7 mod 12	pas de k possible	$k \equiv 2 \pmod{6}$
10 mod 12	$k \equiv 0, 1 \pmod{3}$	$k \equiv 5 \pmod{6}$
2 mod 12	$k \equiv 0, 2 \pmod{3}$	$k \equiv 1 \pmod{6}$
5 mod 12	pas de k possible	$k \equiv 4 \pmod{6}$
8 mod 12	pas de k possible	$k \equiv 0, 2 \pmod{3},$ $k \equiv 4 \pmod{6}, v_2(k) \neq v_2(n)$
11 mod 12	$k \equiv 0, 2 \pmod{3}$	$k \equiv 0, 2 \pmod{3},$
n	$x^n - x^k + 1$	$x^n - x^k - 1$
0 mod 12	pas de k possible	$k \equiv 2, 4 \pmod{6}$
3 mod 12	$k \equiv 1, 2 \pmod{3}$	$k \equiv 1 \pmod{3}$
6 mod 12	$k \equiv 1, 2 \pmod{3}$	$k \equiv 1, 5 \pmod{6}$
9 mod 12	$k \equiv 4, 5 \pmod{6}$	$k \equiv 2 \pmod{3}$
1 mod 12	$k \equiv 0, 1 \pmod{3}$	$k \equiv 0, 1 \pmod{3}$
4 mod 12	pas de k possible	$k \equiv 0, 1 \pmod{3},$ $k \equiv 2 \pmod{6}, v_2(k) \leq v_2(n)$
7 mod 12	$k \equiv 5 \pmod{6}$	$k \equiv 5 \pmod{6}$
10 mod 12	$k \equiv 0, 1 \pmod{3},$ $k \equiv 2 \pmod{6}, v_2(k) \geq v_2(n)$	$k \equiv 5 \pmod{6}$
2 mod 12	$k \equiv 0, 2 \pmod{3},$ $k \equiv 4 \pmod{6}$	$k \equiv 1 \pmod{6}$
5 mod 12	$k \equiv 1 \pmod{3}$	$k \equiv 1 \pmod{6}$
8 mod 12	pas de k possible	$k \equiv 0, 2 \pmod{3},$ $k \equiv 4 \pmod{6}, v_2(k) \leq v_2(n)$
11 mod 12	$k \equiv 0, 2 \pmod{3}$	$k \equiv 0, 2 \pmod{3}$

TABLEAU A.1: Valeurs de k pour lesquelles les trinômes sur $\mathbf{F}_3$ ont un nombre impair de facteurs

1. Si n est impair, alors $3^n - 1 \equiv 2 \pmod{12}$. La quantité $3^k - 1 \equiv 2 \pmod{6}$ apparaît dans le tableau A.1 mais uniquement pour le trinôme

$$x^{3^n-1} - x^{3^k-1} + 1.$$

En conséquence, si n est impair, alors les trinômes $x^{3^n-1} + x^{3^k-1} - 1$ et $x^{3^n-1} - x^{3^k-1} - 1$ ont, ou bien un nombre pair de facteurs, ou bien une racine dans $\mathbf{F}_3$.

2. Si n est pair, alors $3^n - 1 \equiv 8 \pmod{12}$.

La quantité $3^k - 1 \equiv 2 \pmod{6}$ apparaît bien dans le tableau A.1 pour les trinômes

$$x^{3^n-1} + x^{3^k-1} - 1, x^{3^n-1} - x^{3^k-1} - 1.$$

Par conséquent, si n est pair, le trinôme $x^{3^n-1} - x^{3^k-1} + 1$ a un nombre pair de facteurs $\mathbf{F}_3$.

■

Annexe B

Rappels de théorie des codes correcteurs

Dans cette annexe, je présente les définitions essentielles de théorie des codes permettant une bonne compréhension du document. Le lecteur désireux d'en savoir plus se reportera au «Handbook» de théorie des codes [PH98], et aux livres de McELIECE, puis de LIDL et de NIEDERREITER [McE87, LN97] sur les corps finis.

1 Éléments de corps finis

Toute la thèse s'articule autour de la notion de corps finis et de leurs propriétés. Étant donné p un nombre premier, il existe un corps commutatif à p^u éléments unique à isomorphisme près, noté $\mathbf{F}_{p^u}$. Réciproquement, tout corps fini commutatif est un corps du type $\mathbf{F}_{p^m}$ pour un p , premier, et un m entier positif.

Définition 31 *Étant donné un corps fini $\mathbf{F}_{p^u}$ à p^u éléments, p premier,*

- *Le nombre premier p est appelé caractéristique du corps.*
- *Le nombre entier positif u est appelé degré de l'extension $\mathbf{F}_{p^u}/\mathbf{F}_p$.*

Sur un corps fini $\mathbf{F}_{q^m}$ où q est une puissance d'un nombre premier, $q = p^u$, l'application

$$\begin{aligned}\sigma : \mathbf{F}_{q^m} &\rightarrow \mathbf{F}_{q^m}, \\ z &\mapsto \sigma(z) = z^q,\end{aligned}$$

est un automorphisme de $\mathbf{F}_{q^m}$ dont la restriction à $\mathbf{F}_q$ est égale à l'identité. Cet automorphisme est appelé *automorphisme de FROBENIUS* de $\mathbf{F}_{q^m}/\mathbf{F}_q$.

Une grande importance est attachée aux sous-corps $\mathbf{F}_q$ d'un corps $\mathbf{F}_{q^m}$ donné où q est une puissance d'un nombre premier, $q = p^u$.

Avec l'automorphisme de FROBENIUS, on peut construire un opérateur de projection de $\mathbf{F}_{q^m}$ dans $\mathbf{F}_q$, appelé *opérateur Trace*

Définition 32 (Opérateur Trace) Soit $\mathbf{F}_{q^m}$ un corps fini. Soit σ , l'automorphisme de FROBENIUS de $\mathbf{F}_{q^m}/\mathbf{F}_q$. Alors l'application

$$\begin{aligned} \text{Tr}_{q^m/q} : \mathbf{F}_{q^m} &\mapsto \mathbf{F}_q, \\ z &\rightarrow \text{Tr}_{q^m/q}(z) = \sum_{i=1}^m \sigma^i(z), \end{aligned}$$

est appelée opérateur Trace de $\mathbf{F}_{q^m}/\mathbf{F}_q$.

Les propriétés importantes de l'opérateur Trace sont les suivantes

Propriétés 7 Soit $\text{Tr}_{q^m/q}$ l'opérateur Trace de $\mathbf{F}_{q^m}/\mathbf{F}_q$. Alors

1. $\text{Tr}_{q^m/q}(\alpha + \beta) = \text{Tr}_{q^m/q}(\alpha) + \text{Tr}_{q^m/q}(\beta)$, pour tous $\alpha, \beta \in \mathbf{F}_{q^m}$.
2. $\text{Tr}_{q^m/q}(c\alpha) = c \text{Tr}_{q^m/q}(\alpha)$, pour tout $c \in \mathbf{F}_q$, et pour tout $\alpha \in \mathbf{F}_{q^m}$.
3. Si on a $\mathbf{F}_q \subset \mathbf{F}_{q^s} \subset \mathbf{F}_{q^m}$, alors, pour tout $\alpha \in \mathbf{F}_{q^m}$,

$$\text{Tr}_{q^m/q}(\alpha) = \text{Tr}_{q^s/q} [\text{Tr}_{q^m/q^s}(\alpha)],$$

Pour des informations complémentaires, le lecteur pourra se reporter à [LN97, pp. 54–63].

2 Fondamentaux de théorie des codes

Soit $\mathbf{F}_{q^m}$ un corps fini à q^m éléments. Soit

$$\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n)$$

un vecteur de longueur n d'éléments distincts quelconques.

Définition 33 Un code C de longueur n et de taille M sur $\mathbf{F}_{q^m}$, et de vecteur générateur $\mathbf{L}$ est un ensemble de M mots

$$\mathbf{a} = (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n}),$$

où $a_{\alpha_i} \in \mathbf{F}_{q^m}$. Si de plus C est un espace vectoriel sur $\mathbf{F}_{q^m}$, on dit que C est un code linéaire de dimension $\text{Log}_{q^m}(M)$.

À cette définition est naturellement liée la notion de support d'un mot, qui, étant donné un mot de longueur n sur $\mathbf{F}_{q^m}$ compte le nombre de coordonnées non nulles de ce mot.

Définition 34 (Support d'un mot) Soit $\mathbf{a} = (a_{\alpha_1}, a_{\alpha_2}, \dots, a_{\alpha_n})$, un mot de vecteur générateur $\mathbf{L} = (\alpha_1, \alpha_2, \dots, \alpha_n)$. Le support de $\mathbf{a}$, noté $\text{Supp}(\mathbf{a})$ est l'ensemble des éléments α_i de $\mathbf{L}$ tels que

$$a_{\alpha_i} \neq 0.$$

On peut munir l'espace des mots de même vecteur générateur $\mathbf{L}$ d'une norme appelée le poids d'un mot et notée wt par :

$$\text{wt}(\mathbf{a}) = |\text{Supp}(\mathbf{a})|.$$

Dans la suite, je ne considère plus que les codes linéaires :

Définition 35 (Distance minimale) *Étant donné C , un code linéaire, la distance minimale d de C est par définition égale au plus petit poids d'un mot non nul de C , i.e.*

$$d = \text{Min}_{\mathbf{a} \in C} (\text{wt}(\mathbf{a})).$$

Il existe une inégalité qui lie la distance minimale, la longueur et la dimension d'un code.

Définition 36 (Borne de SINGLETON) *Soit C un code linéaire, de longueur n , de dimension k et de distance minimale d , alors*

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

Les codes dont les paramètres satisfont l'égalité

$$k = n - d + 1$$

sont appelés codes MDS – pour *Maximum Distance Separable*.

Soit C un code linéaire sur $\mathbf{F}_{q^m}$, de longueur n , de dimension k . Alors, comme C est un espace vectoriel de longueur n sur $\mathbf{F}_{q^m}$, il peut être caractérisé par des matrices $\mathbf{G}$ de taille $k \times n$ à coefficients dans $\mathbf{F}_{q^m}$ telles que

$$C = \{\mathbf{xG} \mid \mathbf{x} \in \mathbf{F}_{q^m}^n\}.$$

Une telle matrice $\mathbf{G}$ est appelée *matrice génératrice* de C . Si une matrice génératrice $\mathbf{G}$ du code C s'écrit

$$\mathbf{G} = (\mathbf{I}_k \mid \mathbf{R}),$$

où $\mathbf{I}_k$ désigne la matrice identité d'ordre k , alors $\mathbf{G}$ est unique. On dit que $\mathbf{G}$ est la matrice génératrice du code C mise sous *forme systématique*. La matrice $\mathbf{R}$ s'appelle *matrice de redondance* du code C .

Soit $\mathbf{G}$ une matrice de taille $k \times n$ sur $\mathbf{F}_q$, génératrice du code C . Alors une matrice $\mathbf{H}$ de taille $(n - k) \times n$ telle que

$$\mathbf{GH}^T = \mathbf{0}$$

est appelée *matrice de parité* de C . Le code C de matrice de parité $\mathbf{H}$ peut donc également être caractérisé de la façon suivante :

$$C = \{\mathbf{a} \in \mathbf{F}_{q^m}^n \mid \mathbf{aH}^T = \mathbf{0}\}.$$

Définition 37 (Syndrome et translaté) Soit C un code linéaire sur $\mathbf{F}_{q^m}$, de longueur n , de dimension k , et de matrice de parité $\mathbf{H}$. Soit $\mathbf{y} \in \mathbf{F}_{q^m}^n$ quelconque.

1. Le mot $\mathbf{s} = \mathbf{y}\mathbf{H}^T$, de longueur $n - k$ sur $\mathbf{F}_{q^m}$ est appelé syndrome de $\mathbf{x}$.
2. L'ensemble des mots de $\mathbf{F}_{q^m}^n$ ayant le même syndrome $\mathbf{s}$ que $\mathbf{y}$ égal à

$$\mathbf{x} + C.$$

L'ensemble $\mathbf{x} + C$ est appelé translaté de $\mathbf{x}$ dans C ,

Finalement, j'introduis la notion de sous-code trace d'un code.

Définition 38 (Sous-code trace) Soit C un code linéaire sur $\mathbf{F}_{q^m}$, alors on appelle sous-code trace de C le sous-code de C sur $\mathbf{F}_q$ formé des mots de C à coefficients dans $\mathbf{F}_q$. Il est noté $(C | \mathbf{F}_q)$, i.e.

$$(C | \mathbf{F}_q) = \{\mathbf{a} \in C \mid \mathbf{a} \in \mathbf{F}_q^n\}.$$

Bibliographie

- [ACH83] ADLER (R. L.), COPPERSMITH (D.) et HASSNER (M.). – Algorithms for sliding block codes : An application of symbolic dynamics to information theory. *IEEE Transactions on Information Theory*, vol. 29, n1, 1983.
- [AjMj74] ASSMUS JR. (E. F.) et MATTSON JR. (H. F.). – Coding and combinatorics. *SIAM Review*, vol. 16, 1974.
- [AM87] ADAMS (C. M.) et MEIJER (H.). – Security related comments regarding McEliece’s public-key cryptosystem. In : *Advances in Cryptology - CRYPTO’87*, éd. par POMERANCE (C.), LNCS. pp. 224–228. – Springer-Verlag, 1987.
- [AS94] AUGOT (D.) et SENDRIER (N.). – Idempotents and the BCH bound. *IEEE Transactions on Information Theory*, vol. 40, n1, 1994, pp. 224–228.
- [Aug01] AUGOT (D.) (édité par). – *Workshop on Coding and Cryptography 2001*. – Écoles de Coëtquidan, janvier 2001. ISBN : 2-7261-1179-3.
- [BB00] BLANCHET (F.) et BOMMIER (G.). – Binary quasi-cyclic Goppa codes. *Designs, Codes and Cryptography*, vol. 20, 2000, pp. 107–124.
- [BE97] BIERBRAUER (J.) et EDEL (Y.). – New code parameters from Reed-Solomon subfield subcodes. *IEEE Transactions on Information Theory*, vol. 43, n3, mai 1997, pp. 953–968.
- [Ber73] BERLEKAMP (E. R.). – Goppa codes. *IEEE Transactions on Information Theory*, vol. 19, n5, septembre 1973, pp. 590–592. – invited paper.
- [Ber82] BERLEKAMP (E. R.). – Bit-serial Reed-Solomon encoders. *IEEE Transactions on Information Theory*, vol. 28, n6, novembre 1982, pp. 869–874.
- [Ber84] BERLEKAMP (E. R.). – *Algebraic Coding Theory*. – Aegean Press Park, 1984, revised 1984 édition.
- [Ber99] BERGER (T. P.). – New classes of cyclic extended Goppa codes. *IEEE Transactions on Information Theory*, vol. 45, n4, 1999, pp. 1264–1266.

- [Ber00a] BERGER (T. P.). – Goppa and related codes invariant under a prescribed permutation. *IEEE Transactions on Information Theory*, vol. 46, n7, novembre 2000, pp. 2628–2633.
- [Ber00b] BERGER (T. P.). – On the cyclicity of Goppa codes, parity-check subcodes of Goppa codes and extended Goppa codes. *Finite Fields and applications*, vol. 6, 2000, pp. 255–281.
- [BGL94] BLAKE (I. F.), GAO (S.) et LAMBERT (R.). – Constructive problems for irreducible polynomials over finite fields. In : *Third Canadian Workshop on Information theory and Applications*, pp. 1–23. – 1994.
- [BL00] BERGER (T. P.) et LOIDREAU (P.). – A Niederreiter version of the GPT public-key cryptosystem. In : *Seventh International Workshop on Algebraic and Combinatorial Coding Theory*, pp. 72–78. – Bansko, Bulgarie, juin 2000.
- [BM73] BERLEKAMP (E. R.) et MORENO (O.). – Extended double-error correcting Goppa codes are cyclic. *IEEE Transactions on Information Theory*, vol. 18, 1973, pp. 404–414.
- [BMvT78] BERLEKAMP (E. R.), McELIECE (R. J.) et VAN TILBORG (H. C.). – On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, vol. 24, n3, mai 1978.
- [Bom66] BOMBIERI (E.). – Exponential sums in finite fields. *American Journal of Mathematics*, vol. 88, 1966, pp. 71–105.
- [BP91] BRUALDI (R. A.) et PLESS (V. S.). – Weight enumerators of self-dual codes. *IEEE Transactions on Information Theory*, vol. 37, n4, 1991, pp. 1222–1225.
- [BRS67] BERLEKAMP (E. R.), RUMSEY (H.) et SOLOMON (G.). – On the solutions of algebraic equations over finite fields. *Information and Control*, vol. 10, 1967, pp. 553–564.
- [Can96] CANTEAUT (A.). – *Attaques de cryptosystèmes à mots de poids faible et construction de fonctions t-résilientes*. – Thèse de doctorat, Université Paris-VI, 1996.
- [CC98] CANTEAUT (A.) et CHABAUD (F.). – A new algorithm for finding minimum-weight words in a linear code : Application to McEliece’s cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, vol. 44, n1, janvier 1998, pp. 367–378.
- [CFS01] COURTOIS (N.), FINIASZ (M.) et SENDRIER (N.). – *How to achieve a McEliece-based Digital Signature Scheme*. – Rapport de recherche, <http://eprint.iacr.org/2001/010/>, INRIA, 2001.
- [Che78] CHEN (C.-L.). – Equivalent irreducible Goppa codes. *IEEE Transactions on Information Theory*, vol. 24, n6, 1978.

- [Che96] CHEN (K.). – A new identification algorithm. *In : Cryptographic policy and algorithms*. pp. 244–249. – Springer, 1996.
- [CL87] CONAN (J.) et LOELOELIAN (M.). – A transform approach to Goppa codes. *IEEE Transactions on Information Theory*, vol. 33, 1987, pp. 105–115.
- [Cov73] COVER (T. M.). – Enumerative source encoding. *IEEE Transactions on Information Theory*, vol. 19, n1, 1973.
- [CS96] CHABAUD (F.) et STERN (J.). – The cryptographic security of the syndrome decoding problem for rank distance codes. *In : Advances in Cryptology - ASIACRYPT '96*, éd. par KIM (K.) et MATSUMOTO (T.), LNCS. – Springer-Verlag, novembre 1996.
- [CS98] CANTEAUT (A.) et SENDRIER (N.). – Cryptanalysis of the original McEliece cryptosystem. *In : Advances in Cryptology - ASIACRYPT'98*, éd. par OHTA (K.) et PEI (D.), LNCS, n1514, pp. 187–199. – 1998.
- [CTZ99] CHARPIN (P.), TIETÄVÄINEN (A.) et ZINOVIEV (V.). – On the minimum distance of non-binary cyclic codes. *Designs, Codes and Cryptography*, vol. 17, n1, 1999, pp. 81–85.
- [CU57] CARLITZ (L.) et UCHIYAMA (S.). – Bounds for exponential sums. *Duke Mathematical Journal*, vol. 26, 1957, pp. 37–41.
- [Del75a] DELSARTE (P.). – Four fundamental parameters of a code and their combinatorial significance. *Information and Control*, vol. 23, n5, 1975.
- [Del75b] DELSARTE (P.). – On subfield subcodes of modified Reed–Solomon codes. *IEEE Transactions on Information Theory*, vol. 20, 1975, pp. 575–576.
- [DH76] DIFFIE (W.) et HELLMAN (M.). – New directions in cryptography. *IEEE Transactions on Information Theory*, vol. 22, n6, 1976, pp. 644–654.
- [Epk93] EPKENHANS (M.). – Trace forms of trinomials. *IEEE Transactions on Information Theory*, vol. 155, 1993, pp. 211–220.
- [ETV00] ELIA (M.), TARICCO (G.) et VITERBO (V.). – On the classification of binary Goppa codes. *In : proceedings of ISITA, Honolulu*. – novembre 2000.
- [Gab85a] GABIDULIN (E. M.). – Optimal array error-correcting codes. *Problemy Peredachi Informatsii*, vol. 21, n2, 1985, pp. 102–106.
- [Gab85b] GABIDULIN (E. M.). – Theory of codes with maximal rank distance. *Problemy Peredachi Informatsii*, vol. 21, juillet 1985, pp. 1–12. – in russian.
- [Gab91] GABIDULIN (E. M.). – A fast matrix decoding algorithm for rank-error correcting codes. *In : Algebraic coding*, éd. par COHEN

- (G.), LITSYN (S.), LOBSTEIN (A.) et ZÉMOR (G.), LNCS. pp. 126–133. – Springer-Verlag, 1991.
- [Gab92] GABIDULIN (E. M.). – Comments on “Maximum-rank array codes and their application to crisscross error-correction”. *IEEE Transactions on Information Theory*, vol. 38, n3, mai 1992, p. 1183.
- [Gab95] GABIDULIN (E. M.). – Public-key cryptosystems based on linear codes over large alphabets : efficiency and weakness. In : *Codes and Cyphers*, éd. par FARRELL (P. G.). pp. 17–31. – Formara Limited, Southend-on-sea, Essex, 1995.
- [GG99] GOLOMB (S. W.) et GONG (G.). – Periodic binary sequences with the “Trinomial Property”. *IEEE Transactions on Information Theory*, vol. 45, n4, mai 1999, pp. 1276–1279.
- [Gib91] GIBSON (J. K.). – Equivalent Goppa codes and trapdoors to McEliece’s public-key cryptosystem. In : *Advances in Cryptology - EUROCRYPT’91*, éd. par DAVIES (D. W.), LNCS, n547. pp. 517–521. – Springer-Verlag, 1991.
- [Gib95a] GIBSON (J. K.). – *Algebraic Coded Cryptosystems*. – Thèse de doctorat, Royal Holloway and Bedford New College, University of London, 1995.
- [Gib95b] GIBSON (J. K.). – Severely denting the Gabidulin version of the McEliece public-key cryptosystem. *Designs, Codes and Cryptography*, vol. 6, 1995, pp. 37–45.
- [Gib96] GIBSON (J. K.). – The security of the Gabidulin public-key cryptosystem. In : *EUROCRYPT’96*, éd. par MAURER (U.), pp. 212–223. – 1996.
- [GK72] GABIDULIN (E. M.) et KORZHIK (V. I.). – Codes correcting lattice-pattern errors. *Izvestiya VUZ. Radioelektronika*, vol. 15, n4, 1972, pp. 492–498.
- [GK94] GABIDULIN (E. M.) et KJELSEN (O.). – How to avoid the Sidel’nikov-Shestakov attack. In : *Error control, cryptography, and speech compression*, éd. par O. CHMORA (S. B. WICKER), LNCS. pp. 33–40. – Springer-Verlag, 1994.
- [GL00] GABIDULIN (E. M.) et LOIDREAU (P.). – Subfield subcodes of maximum-rank distance codes. In : *Seventh International Workshop on Algebraic and Combinatorial Coding Theory, ACCT’2000*. pp. 151–156. – Bansko, Bulgarie, juin 2000.
- [GO00] GABIDULIN (E. M.) et OURIVSKI (A. V.). – Improved GPT public-key cryptosystems. In : *Coding Communications and Broadcasting*, éd. par FARRELL (P.), DARNELL (M.) et HONARY (B.), pp. 73–102. – 2000.

- [GO01] GABIDULIN (E. M.) et OURIVSKI (A. V.). – Modified GPT PKC with right scrambler. *In : Workshop on Coding and Cryptography 2001*, éd. par AUGOT (D.), pp. 233–242. – janvier 2001.
- [Gol67] GOLOMB (S. W.). – *Shift Register Sequences*, chap. 5. – Holden-Day, 1967.
- [Gop70] GOPPA (V. D.). – A new class of linear error-correcting codes. *Problemy Peredachi Informatsii*, vol. 6, n3, 1970, pp. 207–212.
- [GPT91] GABIDULIN (E. M.), PARAMONOV (A. V.) et TRETJAKOV (O. V.). – Ideals over a non-commutative ring and their application in cryptology. *LNCS*, vol. 573, 1991, pp. 482 – 489.
- [GS98] GABIDULIN (E. M.) et SIMONIS (J.). – Metrics generated by families of subspaces. *IEEE Transactions on Information Theory*, vol. 44, n3, mai 1998, pp. 1336–1342.
- [GZ68] GOLDSTEIN (R. M.) et ZIERLER (N.). – On trinomial recurrences. *IEEE Transactions on Information Theory*, vol. 14, n1, janvier 1968, pp. 150–151.
- [Har88] HARARI (S.). – A new authentication algorithm. *In : Coding Theory and Applications*, éd. par COHEN (G.) et WOLFMANN (J.), LNCS. pp. 91–105. – Springer-Verlag, 1988.
- [Hei87] HEIMAN (R.). – *On the security of cryptosystems based on linear error-correcting codes*. – Rapport de DEA, Weizmann Institute of Science, 1987.
- [Her00] HERSANS (A.). – *Étude et Implémentation des attaques de Gibson contre le cryptosystème*. – Rapport de DEA, Université Paris VI, 2000.
- [HK97] HELLESETH (T.) et KLØVE (T.). – The Newton radius of codes. *IEEE Transactions on Information Theory*, vol. 43, n6, 1997, pp. 1820–1831.
- [HKM78] HELLESETH (T.), KLØVE (T.) et MYKKELTVEIT (J.). – On the covering radius of binary codes. *IEEE Transactions on Information Theory*, vol. 24, n5, septembre 1978, pp. 627–628.
- [HNS99] HERNQUIST (E.), NEWHART (D. W.) et STONG (R. A.). – Finding short and light codewords in BCH codes. *IEEE Transactions on Information Theory*, vol. 45, n1, janvier 1999, pp. 255–260.
- [HS73] HELGERT (H. J.) et STINAFF (R. D.). – Shortened BCH codes. *IEEE Transactions on Information Theory*, vol. 19, n6, novembre 1973, pp. 818–819.
- [Jen95] JENSEN (J. M.). – Subgroup subcodes. *IEEE Transactions on Information Theory*, vol. 41, n3, mai 1995, pp. 781–785.
- [JMV81] JUNGnickel (D.), MENEZES (A. J.) et VANSTONE (S. A.). – On the number of self-dual bases of $\text{GF}(q^m)$ over $\text{GF}(q)$. *Proceedings of the American Mathematical Society*, vol. 109, 1981, pp. 23–29.

- [Kah96] KAHN (D.). – *The Code Breakers : the story of secret writing.* – MacMillan Publishing, 1996, 2 édition.
- [KI01] KOBARA (K.) et IMAI (H.). – Semantically secure McEliece public-key cryptosystems – conversions for mceliece pkc. *In : Proceedings of the International Workshop on Practice and Theory in Public Key Cryptography.* – 2001.
- [Kob94] KOBLITZ (N.). – *A Course in Number Theory and Cryptography.* – Springer, 1994, 2 édition, *Graduate Texts in Mathematics*, volume 114.
- [KT70] KASAMI (T.) et TOKURA (N.). – On the weight structure of Reed-Muller codes. *IEEE Transactions on Information Theory*, vol. 16, n6, novembre 1970, pp. 752–758.
- [Lan98] LANGEVIN (P.). – Weights of Abelian codes. *Designs, Codes and Cryptography*, vol. 14, 1998, pp. 239–245.
- [LB88] LEE (P. J.) et BRICKELL (E. F.). – An observation on the security of McEliece’s public-key cryptosystem. *In : Advances in Cryptology - EUROCRYPT’88*, éd. par GÜNTER (C. G.), LNCS, pp. 275–280. – Springer-Verlag, 1988.
- [LDW94] LI (Y. X.), DENG (R. H.) et WANG (X. M.). – On the equivalence of McEliece’s and Niederreiter’s public-key cryptosystems. *IEEE Transactions on Information Theory*, vol. 40, n1, 1994, pp. 271–273.
- [Leo88] LEON (J. S.). – A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, vol. 34, n5, 1988, pp. 1354–1359.
- [LLL82] LENSTRA (A. K.), LENSTRA (H. W.) et LOVÁČZ (L.). – Factoring polynomials with rational coefficients. *Mathematische Annalen*, vol. 261, n4, 1982, pp. 515–534.
- [LN97] LIDL (R.) et NIEDERREITER (H.). – *Finite Fields.* – Cambridge University Press, 1997, 2 édition.
- [Loi] LOIDREAU (P.). – Codes derived from binary Goppa codes. *Problemy Peredachi Informatsii.* – à paraître.
- [Loi97] LOIDREAU (P.). – *Éléments sur les codes de GOPPA en relation avec le protocole de McEliece.* – Rapport de DEA, École Polytechnique, 1997.
- [Loi98] LOIDREAU (P.). – Codes derived from Goppa codes. *In : Sixth International Workshop on Algebraic and Combinatorial Coding Theory, ACCT’98*, pp. 179–182. – Pskov, Russie, septembre 1998.
- [Loi99] LOIDREAU (P.). – On the factorization of trinomials over $\text{GF}(3)$, août 1999. *Finite Fields and applications*, Fq5.
- [Loi00a] LOIDREAU (P.). – Large weight patterns decoding in Goppa codes and application to cryptography. *In : Proceedings of 2000*

- IEEE International Symposium on Information Theory*, p. 186.
– juin 2000.
- [Loi00b] LOIDREAU (P.). – *On the factorization of trinomials over $GF(3)$* .
– Rapport de recherche 3918, <http://www.inria.fr/rrrt/rr-3918.html>, INRIA, avril 2000.
- [Loi00c] LOIDREAU (P.). – Strengthening McEliece public-key cryptosystem. In : *Advances in Cryptology - ASIACRYPT 2000*, éd. par OKAMOTO (T.), LNCS. IACR. – Springer-Verlag, décembre 2000.
- [LS91] LEMPEL (A.) et SEROUSSI (G.). – Explicit formulas for self-complementary normal bases in certain finite fields. *IEEE Transactions on Information Theory*, vol. 37, n 4, juillet 1991, pp. 1220–1222.
- [LS98] LOIDREAU (P.) et SENDRIER (N.). – Some weak keys in McEliece public-key cryptosystem. In : *IEEE International Symposium on Information Theory, ISIT'98*, p. 382. – Cambridge, MA, USA, 1998.
- [LS01] LOIDREAU (P.) et SENDRIER (N.). – Weak keys in McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, vol. 47, n3, mars 2001.
- [Man75] MANDELBAUM (D. M.). – On the derivation of Goppa codes. *IEEE Transactions on Information Theory*, vol. 21, n1, 1975, pp. 110–111.
- [Man82] MANDELBAUM (D. M.). – Decoding of erasures and errors for certain RS codes by decreased redundancy. *IEEE Transactions on Information Theory*, vol. 28, n2, mars 1982, pp. 330–336.
- [McE78] MCELIECE (R. J.). – *A public-key cryptosystem based on algebraic coding theory*. – Rapport de recherche, Jet Propulsion Lab. DSN Progress Report, 1978.
- [McE87] MCELIECE (R. J.). – *Finite Fields for Computer Scientists and Engineers*. – Kluwer Academic Publishers, 1987.
- [Men93] MENEZES (A. J.). – *Applications of Finite Fields*. – Kluwer Academic Publishers, 1993.
- [MH78] MERKLE (R. C.) et HELLMAN (M.). – Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, vol. 24, n0, septembre 1978, pp. 525–530.
- [MS77] MACWILLIAMS (F. J.) et SLOANE (N. J. A.). – *The Theory of Error-Correcting Codes*. – North Holland, 1977.
- [Mun98] MUNEMASA (A.). – Orthogonal arrays, primitive trinomials and shift-register sequences. *Finite Fields and applications*, vol. 4, 1998, pp. 252–260.

- [MvOV97] MENEZES (A. J.), VAN OORSCHOT (P. C.) et VANSTONE (S. A.). – *Handbook of Applied Cryptography*. – CRC Press, 1997.
- [Nie86] NIEDERREITER (H.). – Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, vol. 15, n2, 1986, pp. 159 – 166.
- [Odl90] ODLYZKO (A. M.). – The rise and fall of knapsack cryptosystems. In : *Cryptology and Computational Number Theory*, éd. par POMERANCE (C.). pp. 75–88. – American Mathematical Society, 1990.
- [Ore33] ORE (Ö.). – On a special class of polynomials. *Transactions of the American Mathematical Society*, vol. 35, 1933, pp. 559–584.
- [Ore34] ORE (Ö.). – Contribution to the theory of finite fields. *Transactions of the American Mathematical Society*, vol. 36, 1934, pp. 243–274.
- [Pat75] PATTERSON (N. J.). – The algebraic decoding of Goppa codes. *IEEE Transactions on Information Theory*, vol. 21, 1975, pp. 203–207.
- [Pat96] PATARIN (J.). – Hidden fields equations (HFE) and isomorphisms of polynomials (IP) : two new families of asymmetric algorithms. In : *Advances in Cryptology - EUROCRYPT'96*, éd. par MAURER (U.), LNCS. – Springer-Verlag, 1996.
- [PH98] PLESS (V. S.) et HUFFMAN (W. C.) (édité par). – *Handbook of Coding Theory*. – Elsevier Science B.V., 1998.
- [PR97] PETRANK (E.) et ROTH (R. M.). – Is code equivalence easy to decide? *IEEE Transactions on Information Theory*, vol. 43, n5, septembre 1997, pp. 1602–1604.
- [RF01] RYAN (J. A.) et FITZPATRICK (P.). – The number of inequivalent irreducible Goppa codes. In : *Workshop on Coding and Cryptography 2001*, éd. par AUGOT (D.), pp. 209–216. – janvier 2001.
- [RHAS92] ROSEIRO (A. M.), HALL (J. I.), ADNEY (J. E.) et SIEGEL (M.). – The trace operator and redundancy of Goppa codes. *IEEE Transactions on Information Theory*, vol. 38, n 3, 1992, pp. 1130–1133.
- [Rod92] RODIER (F.). – On the dimension of Goppa codes. *AAECC*, vol. 3, 1992, pp. 263–267.
- [Rot91] ROTH (R. M.). – Maximum-Rank array codes and their application to crisscross error correction. *IEEE Transactions on Information Theory*, vol. 37, n2, mars 1991, pp. 328–336.

- [Rot92] ROTH (R. M.). – Authors reply to “Comments on maximum rank-array codes and their application to crisscross error-detection”. *IEEE Transactions on Information Theory*, vol. 38, n3, mai 1992, p. 1183.
- [Rot96] ROTH (R. M.). – Tensor codes for the rank metric. *IEEE Transactions on Information Theory*, vol. 42, n6, novembre 1996, pp. 2146–2157.
- [Rot97] ROTH (R. M.). – Probabilistic crisscross error correction. *IEEE Transactions on Information Theory*, vol. 43, n5, septembre 1997, pp. 1425–1437.
- [RSA78] RIVEST (R. L.), SHAMIR (A.) et ADLEMAN (L. M.). – A method for obtaining digital signatures and public-key cryptosystems. *Communication of the ACM*, vol. 21, n3, 1978, pp. 120–126.
- [SB95] SCHNABL (G.) et BOSSERT (M.). – Soft decision decoding of Reed–Muller codes as generalized multiple concatenated codes. *IEEE Transactions on Information Theory*, vol. 41, n1, 1995, pp. 304–308.
- [Sch96] SCHNEIER (B.). – *Applied Cryptography*. – John Wiley and Sons, Inc., 1996, 2 édition.
- [Sen91] SENDRIER (N.). – *Codes correcteurs d’erreurs à haut pouvoir de correction*. – Thèse de doctorat, Université Paris VI, 1991.
- [Sen95a] SENDRIER (N.). – Efficient generation of binary words of given weight. In : *Cryptography and Coding, 5th IMA Conference, Circencester, UK*, éd. par BOYD (C.), LNCS, pp. 184–187. – décembre 1995.
- [Sen95b] SENDRIER (N.). – *On the dimension of the hull*. – Rapport de Recherche 2682, INRIA, octobre 1995.
- [Sen95c] SENDRIER (N.). – *On the structure of a randomly permuted concatenated code*. – Rapport de Recherche 2460, INRIA, janvier 1995.
- [Sen96] SENDRIER (N.). – *Un algorithme pour trouver la permutation entre deux codes binaires équivalents*. – Rapport de Recherche 2853, INRIA, avril 1996.
- [Sen97] SENDRIER (N.). – On the dimension of the hull. *SIAM Journal on Discrete Mathematics*, vol. 10, n2, mai 1997, pp. 282–293.
- [Sen98] SENDRIER (N.). – On the concatenated structure of a linear code. *AAECC*, vol. 9, n3, 1998, pp. 221–242.
- [Sen99] SENDRIER (N.). – *The Support Splitting Algorithm*. – Rapport de recherche 3637, INRIA, mars 1999.

- [Sen00] SENDRIER (N.). – Finding the permutation between equivalent codes : the support splitting algorithm. *IEEE Transactions on Information Theory*, vol. 46, n4, juillet 2000, pp. 1193–1203.
- [Sha49] SHANNON (C. E.). – Communication theory of secrecy systems. *Bell system technical journal*, vol. 28, 1949, pp. 656–715.
- [Sho90] SHOUP (V.). – New algorithms for finding irreducible polynomials over finite fields. *Mathematics for Computers*, vol. 54, 1990, pp. 435–447.
- [Shp92] SHPARLINSKI (I. E.). – *Computational and Algorithmic Problems in Finite Fields*. – Kluwer Academic Publishers, 1992, *Mathematics and Its Applications*, volume 88.
- [Sid94] SIDEL'NIKOV (V. M.). – A public-key cryptosystem based on binary Reed-Muller codes. *Discrete Mathematics and Applications*, vol. 4, n3, 1994, pp. 191–207.
- [Sin99] SINGH (S.). – *Histoire des codes secrets*. – Jean-Claude Lattès, 1999.
- [Ske99] SKERSYS (G.). – *Calcul du groupe d'automorphismes des codes. Détermination de l'équivalence des codes*. – Thèse de doctorat, Université de Limoges, octobre 1999.
- [Ske01] SKERSYS (G.). – The average dimension of the hull of cyclic codes. In : *Workshop on Coding and Cryptography 2001*, éd. par AUGOT (D.), pp. 477–486. – janvier 2001.
- [SKF96] SUGITA (T.), KASAMI (T.) et FUJIWARA (T.). – The weight distribution of the third-order Reed-Muller code of length 512. *IEEE Transactions on Information Theory*, vol. 42, n5, septembre 1996, pp. 1622–1625.
- [SP92] SIDEL'NIKOV (V. M.) et PERSHAKOV (A. S.). – Decoding of Reed-Muller codes with a large number of errors. *Problemy Peredachi Informatsii*, vol. 28, n3, 1992, pp. 80–94.
- [SS92] SIDEL'NIKOV (V. M.) et SHESTAKOV (S. O.). – On cryptosystems based on generalized Reed-Solomon codes. *Discrete Mathematics*, vol. 4, n3, 1992, pp. 57–63. – en russe.
- [Ste89] STERN (J.). – A method for finding codewords of small weight. In : *Coding Theory and Applications*, éd. par COHEN (G.) et WOLFMANN (J.), LNCS. Springer-Verlag, pp. 106–113. – 1989.
- [Ste93] STERN (J.). – A new identification scheme based on syndrome decoding. In : *Advances in Cryptology – CRYPTO'93*, éd. par STINSON (D. R.), LNCS. pp. 13–21. – Springer-Verlag, 1993.
- [Sti90] STICHTENOTH (H.). – On the dimension of subfield subcodes. *IEEE Transactions on Information Theory*, vol. 36, n1, janvier 1990, pp. 90–93.

- [Sti95] STINSON (D. R.). – *Cryptography : Theory and Practice*. – CRC Press, Inc., 1995.
- [Swa62] SWAN (R. G.). – Factorization of polynomials over finite fields. *Pacific Journal of Mathematics*, vol. 12, n2, 1962, pp. 1099–1106.
- [TZ75] TZENG (K. K.) et ZIMMERMANN (K. P.). – The true dimension of certain binary Goppa codes. *IEEE Transactions on Information Theory*, vol. 21, 1975, pp. 712–716.
- [Var97] VARDY (A.). – The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, vol. 43, n6, novembre 1997, pp. 1757–1766.
- [vdV90] VAN DER VLUGT (M.). – The true dimension of certain binary Goppa codes. *IEEE Transactions on Information Theory*, vol. 36, n2, 1990, pp. 397–398.
- [vdV91] VAN DER VLUGT (M.). – On the dimension of trace codes. *IEEE Transactions on Information Theory*, vol. 36, n1, janvier 1991, pp. 196–199.
- [Vér95] VÉRON (P.). – *Problèmes SD, Opérateur Trace, Schémas d'Identification et Codes de Goppa*. – Thèse de doctorat, Université de Toulon et du Var, 1995.
- [Vér98] VÉRON (P.). – Goppa codes and trace operator. *IEEE Transactions on Information Theory*, vol. 44, n1, janvier 1998, pp. 290–294.
- [Vis97] VISHNE (U.). – Factorization of trinomials over galois fields of characteristic 2. *Finite Fields and applications*, vol. 3, 1997, pp. 370–377.
- [vT88] VAN TILBURG (J.). – On the McEliece public-key cryptosystem. In : *Advances in Cryptology - CRYPTO'88*, éd. par GOLDWASSER (S.), LNCS. pp. 119–131. – Springer-Verlag, 1988.
- [Wol85] WOLFMANN (J.). – A class of doubly even self-dual binary codes. *Discrete Mathematics*, vol. 56, 1985, pp. 299–303.
- [Zém00] ZÉMOR (G.). – *Cours de Cryptographie*. – Cassini, novembre 2000.
- [Zie70] ZIERLER (N.). – On $x^n + x + 1$ over $\text{GF}(2)$. *Information and Control*, vol. 16, 1970, pp. 502–505.
- [ZR95] ZAIN (A. A.) et RAJAN (B. S.). – Algebraic characterization of MDS group codes over cyclic groups. *IEEE Transactions on Information Theory*, vol. 41, n6, novembre 1995.

Index

- t -décodabilité par tour, 108–113
 - extensions de degré 5, 111
- BOMBIERI, 103
- CONAN, 136
- LOELOELIAN, 136
- McELIECE
 - paramètres de, 113
- SENDRIER, 59
- SHANNON, 11
- VAN DER VLUGT, 103
- algorithme
 - SSA, 71–78
 - BERLEKAMP–MASSEY, 141
 - CANTEAUT–CHABAUD, 30, 156
 - LEE–BRICKELL, 35, 156
- attaque
 - SIDEL’NIKOV–SHESTAKOV, 42–45, 65
 - active, 30, 35
 - par décodage, 29
 - passive, 30
 - structurelle, 29, 65–71
- attaques de GIBSON
 - problème de recherche, 166
- attaques de GIBSON, 157–172
 - première attaque, 158–165
 - seconde attaque, 165–172
- automorphisme de FROBENIUS, 78
- borne
 - GILBERT–VARSHAMOV, 126
 - SINGLETON, 39, 126
- borne de SINGLETON, 203
- caractéristique d’un corps, 201
- clé faible, 14, 78, 84
- clé privée, 12
- clé publique, 12
- code, 202
 - s -projeté, 94–106
 - construction, 95–97
 - dimension, 98–105
 - distance minimale, 105–106
 - propriétés, 98–106
 - sous-code trace, 98
 - équations de parité, 99
- REED–SOLOMON généralisé
 - matrice systématique, 41
- GABIDULIN, 131–149
 - algorithme de décodage, 135–142
 - dual, 132
 - produit direct, 148
 - sous-code trace, 142–149
- GOPPA, 45–55
 - aléatoire, 76
 - binaire, 52–55
 - corps générateur, 49
 - dual, 51
 - groupe d’automorphismes, 94–95
 - polynôme générateur, 49
 - polynôme générateur binaire, 78
 - sous-code idempotent, 78–84
 - sous-code invariant, 96
 - vecteur générateur, 49
- GOPPA irréductible, 49, 70, 87
- GOPPA étendu, 64
 - classes d’équivalence, 64
- HUFFMAN, 23
- REED–SOLOMON généralisé, 39, 40
 - aléatoire, 76

- BCH, 49–50
- classe d'équivalence, 61
- concaténé, 57, 59
- discriminance d'un invariant, 72
- distance rang minimale, 124
- dual, 41
- groupe, 126
- groupe d'automorphismes, 61
- hull, 63
- invariant, 72
- MDS, 128
- MRD, 128
 - distribution des rangs, 129
- poinçonné, 62
- public, 24
- vecteur générateur, 60
 - cellules d'une partition, 73
 - équivalence de partitions, 73
- équivalence, 61
- étendu, 61
 - groupe d'automorphismes, 62
 - équivalence, 62
- code caché, 21, 24, 65, 115
- code linéaire, 202
- code public, 21, 65, 115, 150
- corps de base, 123
- cryptographie
 - à clé publique, 11
 - à clé secrète, 11
- cryptosystème
 - MCÉLIECE, 21–23
 - attaque structurelle, 66
 - clés faibles, 84–91
 - complexité, 22
 - espace des clés, 21
 - paramètres originaux, 23
 - MCÉLIECE modifié, 114
 - sécurité, 116
 - NIEDERREITER, 24–26
 - complexité, 24
 - décodage, 177
 - encodage, 175
 - espace des clés, 24
 - taux de transmission, 25, 177
 - asymétrique, 12
 - GPT
 - attaque par décodage, 155
 - attaque par énumération des bases, 156–157
 - complexité en mémoire, 150
 - complexité en temps, 150
 - matrice de distorsion, 152–155
 - sac-à-dos, 29
 - symétrique, 12
- distance minimale d'un code, 203
- distance rang, 123
- décodage
 - algorithme trivial, 30
 - par ensembles d'information, 30–35
 - par vraisemblance maximale, 130
- ensemble d'information, 32
- extension d'un corps, 201
- fonction de MOEBIUS, 70
- forme systématique, 203
- idempotent, 80
- matrice de parité, 203
- matrice de redondance, 203
- matrice génératrice, 203
- mot
 - invariant, 94
 - ordre, 94
 - vecteur générateur, 60
- motif, 111
- métrique rang, 123–130
 - capacité de correction, 130
 - distance, 123, 125
 - décodage, 129
 - norme, 123
- opérateur Trace, 202
- opérateur trace, 81, 99
- ordre d'un mot, 94

- partition étiquetée, 73
- petit théorème de FERMAT, 105
- polynôme
 - de GAUSS, 154
 - discriminant, 187
 - localisateur, 39
 - masquant, 114
- polynôme linéarisé, 134–135
 - algorithme d'EUCLIDE, 135
- problème
 - distance minimale, 28
 - décodage borné, 20
 - décodage complet, 28
 - décodage rang borné, 129
 - factorisation d'entiers, 13
 - isomorphisme de graphes, 71
 - logarithme discret, 13
 - permutation entre deux codes
 - équivalent, 71
 - équivalence de codes, 71
- rang d'un mot, 123
- RSA, 11
- schéma d'identification, 12
- signature numérique, 12
- sous-code trace, 204
- syndrome, 204
- translaté, 204
- trinôme
 - discriminant, 188
 - primitif, 198
- valuation binaire, 189
- vecteur d'erreur, 21, 151
- vecteur générateur, 39, 202