Thèse de Doctorat

Université Paris 6

Soutenue le 05 décembre 2011 par

Vincent HERBERT

pour l'obtention du

DIPLÔME DE DOCTORAT ÈS SCIENCES

dans la spécialité

INFORMATIQUE

DES CODES CORRECTEURS POUR SÉCURISER L'INFORMATION NUMÉRIQUE

Directeur de thèse	
Nicolas SENDRIER	Inria Paris - Rocquencourt
Rapporteurs	
Thierry BERGER	Université de Limoges
Françoise LEVY-DIT-VEHEL	ENSTA ParisTech
Examinateurs	
Annick VALIBOUZE	Université Pierre et Marie Curie
Philippe ELBAZ-VINCENT	Université Joseph Fourier
Daniel AUGOT	Inria Saclay - Île-de-France

Des Codes Correcteurs pour Sécuriser l'Information Numérique

Error-Correcting Codes to Secure Digital Information

Vincent Herbert

Inria Rocquencourt
Équipe-projet SECRET
Domaine de Voluceau
78153 Le Chesnay

Résumé

Les codes correcteurs d'erreurs sont utilisés pour reconstituer les données numériques, qui sont sujettes à des altérations lors de leur stockage et de leur transport. Il s'agit là de l'utilisation principale des codes correcteurs mais ils peuvent encore être employés en cryptographie. Ils sont dans ce contexte un outil permettant, entre autres choses, de chiffrer des données et d'authentifier des personnes. Ces différents aspects sont traités dans ce document. Pour commencer, nous étudions la classe de codes cycliques possédant un ensemble de définition de la forme $\{1, 2^i+1, 2^j+1\}$, où i et j désignent des entiers positifs distincts. Nous concentrons notre attention sur la caractérisation des codes trois-correcteurs appartenant à cette classe ainsi que sur la distribution de poids de ces codes. Nous améliorons l'algorithme de Schaub, qui donne une minoration de la distance minimale des codes cycliques. Nous mettons en œuvre cet algorithme pour calculer l'immunité spectrale de fonctions booléennes. Cette quantité est reliée à la distance minimale de codes cycliques et est importante pour garantir la sécurité dans certains cryptosystèmes de chiffrement à flot. Dans un second temps, nous proposons une solution pour accélérer le calcul des racines de polynômes dans des corps finis de caractéristique deux. Ce calcul est la phase la plus lente du déchiffrement des cryptosystèmes de type McEliece basés sur les codes de Goppa binaires classiques. Nous fournissons une analyse de la complexité de l'algorithme sous-jacent baptisé BTZ. Nous achevons nos travaux par une étude des protocoles d'authentification à bas coût, dérivés du protocole HB, en adoptant une approche basée sur le problème du décodage par syndrome, plutôt que par l'approche standard, fondée sur le problème LPN.

Mots clés : Codes cycliques ; Énumérateur des poids ; Algorithme de Schaub ; Immunité spectrale ; Complexité dans le pire cas ; Algorithme de la trace de Berlekamp (BTA) ; Codes de Goppa binaires ; Syndrome Decoding ; Cryptologie ultralégère.

Abstract

Error-correcting codes are used to reconstitute digital data, which are proned to alterations during their storage and their transport. They can also be employed in cryptography as a tool to encrypt data and authenticate people. These different aspects are treated in this document. First of all, we study the class of cyclic codes defined by the zero set $\{1, 2^i+1, 2^j+1\}$, where i and j are distinct positive integers. We focus on the characterization of three-error correcting codes in this class as well as the weight distribution of these codes. We improve the Schaub algorithm, which gives a lower bound on the minimum distance of cyclic codes. We implement this algorithm to compute the spectral immunity of Boolean functions. This quantity is related with the minimum distance of cyclic codes and is important to guarantee the security of certain stream ciphers. Subsequently, we propose a solution to speed up the polynomial roots computation over finite fields of characterisitic two. This computation is the slowest step during the decoding of McEliece-type cryptosystems based on classical binary Goppa codes. We provide a complexity analysis of the underlying algorithm named BTZ. We complete our works by a study of low-cost authentication solutions based on the protocol HB, adopting a syndrome decoding approach, instead of the standard approach, based on the LPN problem.

Keywords : Cyclic codes ; Weight enumerator ; Schaub algorithm ; Spectral immunity ; Worst-case complexity ; Berlekamp Trace Algorithm (BTA) ; Binary Goppa codes ; Syndrome Decoding ; Lightweight Cryptology.

Remerciements

Version courte, ingrate et pudique.

— À ceux que j'ai oubliés et aux autres...
— À Alice, Bob, Charlie, Eve et Oscar.
— Au lecteur anonyme.
— Sans oublier J.P., le meilleur pour la faim.

Version traditionnelle.

Je remercie sincèrement Nicolas Sendrier pour m'avoir offert une véritable opportunité ; celle d'effectuer un stage puis une thèse au sein de l'Inria Rocquencourt. Il m'a ainsi ouvert une porte sur l'univers de la recherche ainsi que je le souhaitais. Il a su partager avec moi ses connaissances scientifiques et se montrer disponible dès que le besoin s'en faisait ressentir. Je le remercie encore pour avoir relu ma thèse et pour tous les conseils dont il m'a fait bénéficier.

Je remercie l'Inria Rocquencourt qui met à la disposition de ses chercheurs et aussi de ses « apprentis chercheurs » les meilleurs moyens informatiques avec lesquels j'ai pu travailler.

J'exprime ma gratitude à Françoise Levy-dit-Vehel et à Thierry Berger de me faire l'honneur d'accepter d'être les rapporteurs de ma thèse malgré le court laps de temps qu'il leur était imparti pour rédiger un rapport.

Je suis également redevable envers Annick Valibouze, Philippe Elbaz-Vincent et Daniel Augot pour avoir accepté de faire partie de mon jury de thèse.

Je remercie particulièrement Yvan Gérard et Xavier-François Roblot pour m'avoir initié à la cryptographie et aux codes correcteurs à l'université de Lyon. Sans eux, je n'en serais certainement pas là.

Je remercie également mes professeurs à l'université de Grenoble. Je pense particulièrement à Philippe Elbaz-Vincent et à Alexei Pantchichkine. J'aime les matières qu'ils m'ont enseignées et je conserve un souvenir excellent de leurs enseignements ainsi que de leurs qualités humaines.

Je remercie Anne Canteaut qui dirige de main de maître l'équipe SECRET et je lui souhaite de passer un excellent séjour au Danemark avec Alice.

Je remercie Pascale Charpin, sans qui - lapalissade - l'équipe SECRET ne serait pas ce qu'elle est.

Je ne remercie pas Jean-Pierre Tillich pour m'avoir piqué mon gâteau au chocolat le 14 avril 2011 - c'était un jeudi - mais je le remercie en revanche pour tout le reste. Je remercie aussi les chatons de Schrödinger, Mamdouh Abbara, Denise Maurice, Anne Marin.

Je remercie les Wallons pour nous avoir prêté Baudoin Collard en échange du Breton, Benoît Gérard et de sa sympathique petite famille.

Je remercie Matthieu Finiasz - l'Oracle - qui apporte régulièrement des solutions là où j'apporte constamment des problèmes.

Je remercie mes « collocs » du bureau 6, à savoir Maxime Côte, Rishiraj Bhattacharyya et Valentin Suder ($\forall \in < 0$).

Je remercie le Bengale pour m'avoir permis de faire la connaissance de Bhaskar Biswas, de Sunandan Chakraborty ainsi que de Sumanta Sarkar. Ils m'ont permis, entre autres choses, de découvrir la culture de leur région et je ne pense pas l'oublier de sitôt.

Je remercie mes aînés, ceux qui m'ont accueilli au projet, Maria Naya-Plasencia, Andrea Röck, Cédric Faure, Mathieu Cluzeau, Stéphane Manuel, Yann Laigle-Chapuy. Je n'oublie pas non plus Céline Blondeau et Stéphane Jacob qui sont arrivés avec moi au projet, dans un intervalle de quelques mois tout au plus.

J'ai également une pensée pour Christelle Guiziou, Ayoub Otmani, Christina Boura, Marion Bellard, Rafael Misoczki, Grégory Landais. Je suis content d'avoir rencontré chacun d'entre vous. J'ai en tête des images de plaisirs gastronomiques partagés en votre compagnie (pâtes à l'italienne, gélatine, cheese cake, tzatzikis toastés, picanha, okonomiyakis).

Je remercie aussi Sugata Gangopadhyay, Virginie Colette, Nicolas Delfosse, Ludovic Perret, Alexander Zeh, Gohar Kyureghyan, Assia Saadi, Ayça Cesmelioglu, Chrysanthi Mavromati, Yasser Toor, Skander Azzaz, André Fioravanti, Yacine Mezali, Sabine Leclerq, Eleni Boursinou, Roger Liu ainsi que Fernanda à la cafétaria. Je n'oublie pas non plus tous ceux avec qui j'ai joué au foot au gymnase mais je serais incapable de vous citer tous, aussi je m'abstiendrai de faire des jaloux et je m'arrêterais ici.

J'espère n'avoir écorché aucun de vos patronymes et je vous souhaite à tous et à chacun plein de bonnes choses.

Je remercie enfin, mes amis et ma famille — mamelles du bonheur.

Table des matières

1 Par où commencer ? 13

2 Codes algébriques 17

2.1 Structures algébriques 17
2.2 Notions générales en théorie des codes 20
2.3 Codes cycliques 22
2.4 Codes dérivés 25
2.5 Codes alternants 27

I La correction d'erreurs et les codes cycliques 29

3 Minorer la distance minimale des codes cycliques 33

3.1 Bornes théoriques 33
3.1.1 La borne de Carlitz-Uchiyama (1957) 33
3.1.2 La borne BCH (1960) 34
3.1.3 La borne de Hartmann-Tzeng (1972) 35
3.1.4 Les bornes de van Lint-Wilson (1986) 35
3.2 Borne algorithmique 36
3.2.1 L'algorithme de Schaub (1988) 36
3.2.2 Optimisations 39
3.3 La distance minimale en chiffrement à flot 40
3.3.1 Les codes cycliques et les fonctions booléennes 40
3.3.2 Minorer l'immunité spectrale de fonctions booléennes 41

4 La classe de codes cycliques $\{1, 2^i + 1, 2^j + 1\}$ 45

4.1 Caractériser les codes cycliques 3-correcteurs 45
4.1.1 Résultats théoriques 45

4.1.2 Résultats calculatoires . . . 52
4.2 Calculer leurs énumérateurs des poids . . . 53
4.3 La question de l'équivalence avec le BCH 3-correcteur . . . 55

II Le chiffrement et les codes de Goppa 57

5 Déchiffrer les cryptosystèmes de type McEliece 61
5.1 Les codes de Goppa classiques (1970) . . . 61
5.2 Les cryptosystèmes de McEliece et Niederreiter . . . 62
5.2.1 Description du cryptosystème classique de McEliece (1978) . . . 62
5.2.2 Description du cryptosystème de Niederreiter (1986) . . . 63
5.3 Algorithme de décodage algébrique des codes de Goppa . . . 63
5.4 Complexité théorique du déchiffrement . . . 64

6 Les algorithmes de recherche de racines 67
6.1 La procédure de Chien (1964) . . . 67
6.2 L'algorithme de la trace de Berlekamp (1971) . . . 68
6.3 L'algorithme de Cantor et Zassenhaus' (1981) . . . 68
6.4 Les procédures de Zinoviev (1996) . . . 69
6.4.1 Trouver les racines d'un polynôme affine . . . 69
6.5 L'algorithme Berlekamp Trace - Zinoviev . . . 70
6.5.1 Accélérer le déchiffrement de McEliece . . . 70
6.5.2 Pseudocode de BTZ . . . 70
6.5.3 Mise en œuvre . . . 70
6.5.4 Comment calculer la complexité théorique ? . . . 72
6.6 Résultats de simulation . . . 74

III L'authentification et les codes linéaires 77

7 L'authentification RFID et le protocole HB 79
7.1 La technologie RFID . . . 79
7.1.1 Le principe de fonctionnement d'une étiquette RFID . . . 79
7.1.2 Les applications de la RFID . . . 81
7.1.3 Les problématiques de sécurité liées à la RFID . . . 82
7.1.4 Les solutions de sécurité . . . 83

7.2 Les variantes de HB 84
7.2.1 Le protocole HB 84
7.2.2 Le protocole HB+ 87
7.2.3 Le protocole Random HB 89
7.2.4 Le protocole HB# 89
7.3 Étude de l'attaque sur HB# 91
7.3.1 Le protocole Trusted-HB 93
7.3.2 Le protocole HB-MP+ 94
7.3.3 Le protocole PUF-HB 97

8 Codes et LPN **99**
8.1 Quelques problèmes difficiles liés aux codes 99
8.2 Décodage par ensemble d'information 100
8.3 Les attaques sur LPN 101
8.3.1 Attaque passive sur HB 101
8.3.2 L'algorithme BKW de Blum, Kalai et Wasserman (2000) 102
8.3.3 L'algorithme de Fossorier, Mihaljevic et al. (2006) 102
8.3.4 Les algorithmes LF1 et LF2 de Levieil et Fouque (2006) 102
8.4 Attaque sur la primitive 103
8.5 Preuve de de sécurité 104
8.6 Preuve formelle de l'équivalence entre les problèmes LPN et SD 105
8.6.1 LPN est réductible à SD 105
8.6.2 SD est réductible à LPN 106
8.7 Proposition et analyse d'un nouveau schéma d'authentification low-cost basé sur LPN 107
8.7.1 Une idée de variante pour HB # 107
8.7.2 Évaluation de la sécurité du protocole en se basant sur les LPN-Solvers 108

Bibliographie **115**

Notations **129**

Acronymes **131**

Index **132**

1 Par où commencer ?

Avant de rentrer dans le vif du sujet, je souhaite prendre un peu de recul afin notamment de pouvoir motiver mes travaux. On discutera par la suite de codes correcteurs, essentiellement, mais encore de cryptographie. Des mots qui parlent bien peu au grand public et qui me sont aujourd'hui familiers. Ces domaines appartiennent à ce que certains appellent les sciences du numérique. On considère en effet depuis peu que l'informatique est une science. J'encourage le lecteur intéressé à consulter l'article suivant [DB09] pour une discussion plus détaillée du sujet en compagnie de Gilles Dowek et Gérard Berry. L'informatique est aussi un domaine jeune et en plein essor comme chacun sait. C'est un domaine déjà vaste et les codes correcteurs et la cryptographie en sont des branches. Ces branches sont reliées à celle d'un autre arbre, beaucoup plus vaste celui-là, l'arbre des mathématiques. Codes correcteurs et cryptographie reposent bien souvent sur de l'algèbre, de l'arithmétique, de la géométrie et la liste ne se veut pas exhaustive. Nous sommes donc à la frontière entre informatique et mathématiques. Je n'ai pas connaissance de statistiques sur le sujet mais on sait que la majorité des utilisateurs de l'informatique ignore le fonctionnement basique d'un ordinateur. L'utilisateur lambda ignore bien souvent ce que l'on peut considérer comme le minimum requis. Les préoccupations de l'utilisateur sont différentes. De ce fait, l'utilisateur est « déconnecté » et ne comprend ni comment ça marche ni ce que les chercheurs cherchent.

La cryptographie et la théorie des codes sont des domaines distincts mais il existe des liens puissants entre ces deux sciences [MS86]. L'objectif de la cryptographie est de « sécuriser » les données numériques contre des « ennemis » . L'objectif des codes est également de « sécuriser » les données numériques contre des « ennemis » . L'absence de contexte rend ces définitions vagues et impropres mais elle met en évidence des similitudes fortes entre les deux domaines. La différence entre eux provient du contexte et du sens que l'on donne aux mots « sécuriser » et « ennemis » . Les codes correcteurs sont utilisés notamment dans les télécommunications ou pour le stockage d'informations. Ils permettent alors de détecter et de corriger des erreurs. On entend par le terme « erreur » , une altération ou une perte partielle ou totale de l'information. Ces erreurs sont l'ennemi du codeur. Des causes multiples peuvent être à l'origine de ces erreurs. Des poussières ou des rayures peuvent dégrader les données contenues sur un disque Blu-ray ou un DVD. Des perturbations électromagnétiques peuvent détériorier les données dans les communications satellitaires ou téléphoniques. Les codes correcteurs sont encore utilisés dans diverses technologies de communication (ADSL, fibre optique et USB entre autres). Leur histoire remonte à la fin des années 50 et fait intervenir des chercheurs tels que Richard Hamming, Marcel Golay et Claude Shannon. Le lecteur peut consulter [Tho83] pour de plus amples informations sur ce sujet ainsi que [Moo05, page 44] pour avoir une vision synthétique d'événements charnières qui ont jalonné cette histoire.

Une notion fondamentale des codes correcteurs est la redondance. Plaçons nous dans le

contexte des télécommunications. Les messages transmis contiennent une information qui peut être altérée. Afin de conserver l'information, ils sont préalablements transformés. On parle de codage de l'information. Le codage peut-être schématisé comme étant une suite d'opérations mathématiques inversibles. Le codage modifie le message et le rallonge. Le message codé ne contient plus seulement l'information brute mais aussi de la redondance. Cette redondance peut permettre (ou non) au destinataire de reconstituer l'information en cas de dégradation. En contrepartie, la transmission des messages codés prend davantage de temps puisqu'il y a plus de données à transmettre. Cette capacité de pouvoir corriger ou non des erreurs dépend des propriétes du code.

Mais au fait, qu'est-ce qu'un code ? Un code peut être imagé comme un dictionnaire avec des mots inintelligibles, les mots du code. Ces mots ont pourtant un sens bien qu'il soit dissimulé. Ils révèlent leur sens par le biais de ce que l'on appelle le décodage. Les mots de code peuvent bien être décodés puisque le codage est une suite d'opérations inversibles. Le décodage d'un mot de code nous fournit dès lors un mot de notre bon vieux dictionnaire. Chaque code possède plusieurs caractéristiques fondamentales. Certains codes permettent de corriger beaucoup d'erreurs mais cela ne suffit pas. Si l'on envoie trois fois un message, le destinataire a plus de chances de le recevoir. Mais, on aura envoyé trois fois plus de données, ce qui n'est généralement pas acceptable. Cet exemple est celui du code de répétition. Les codeurs cherchent des codes « optimaux » pour chaque contexte. Il faut faire un compromis entre plusieurs contraintes. L'objectif initial est de corriger des erreurs. On ne veut pas que le codage et le décodage prennent trop de ressources (mémoire, temps). On ne souhaite pas non plus que la taille du message augmente excessivement. Les codes optimaux sont des objets mathématiques structurés comme nous le verrons par la suite.

La cryptographie possède une histoire beaucoup plus riche du fait de son ancienneté. Elle existait en effet avant notre ère, bien qu'elle ait été alors fort différente de la cryptographie contemporaine. Elle est l'une des deux composantes de la cryptologie, la seconde étant la cryptanalyse. La cryptologie fut autrefois l'art du secret et elle est aujourd'hui devenue la science du secret. Il existe de nombreux ouvrages qui relatent son histoire, parmi lesquels on peut citer [Kah96, Ste98, Sin99]. La cryptographie poursuit quatre objectifs : chiffrement, authentification, intégrité et non-répudiation. La non-répudiation fournit la preuve qu'un message a été envoyé ou reçu. L'intégrité certifie que le message reçu est identique au message envoyé. L'authentification prouve l'identité des auteurs d'un message. La confidentialité assure que seuls les destinaires d'un message pourront le lire. Sa soeur, la cryptanalyse est une discipline dont le but est d'analyser la sécurité des systèmes cryptographiques. Elle permet donc de connaître leurs faiblesses et donnent du travail au cryptographe. Cela donne lieu à des échanges perpétuels entre les deux disciplines. La cryptographie s'est longtemps cantonnée au domaine militaire. Elle possède aujourd'hui de nombreuses applications dans le civil. Son utilisation nous apporte certaines garanties de sécurité essentielles. Elle ne nous apporte néanmoins pas toutes les garanties. Il faut garder à l'esprit que la sécurité informatique est un domaine vaste qui englobe la cryptographie.

Depuis la seconde moitié du vingtième siècle, la cryptologie s'est métamorphosée en science. Une des causes essentielles de cette mutation est le développement de l'informatique et de son porte-étendard, l'ordinateur. L'information dans les ordinateurs qui nous entourent aujourd'hui est codée dans ce que l'on appelle le langage machine, il s'agit d'un langage binaire. Son alphabet possède deux lettres appelés bits (contraction de binary digits) : 0 et 1. Ils

correspondent à deux états physiques de l'ordinateur. Le 0 représente une bascule (il s'agit d'un composant électrique) non chargée et le 1, une bascule chargée d'électricité. Dans un ordinateur, chaque information est stockée dans des emplacements mémoires qui contiennent huit bascules. Quelle que soit, la nature de l'information, elle est représentée par des suites finies de 0 et de 1 regroupées par paquets de huit appelés octets. Il peut s'agir indifféremment de texte, d'images, de photos, de musiques, de vidéos. Ces différents types d'informations peuvent être codés sous forme binaire. Cela n'était pas possible avant l'avènement de l'informatique. Aujourd'hui, tout est bit [1] et cela a des conséquences. On traite de manière identique des données qui sont à l'origine de nature distincte. De plus, pour assurer la sécurité de l'information, les données sont transformées par diverses méthodes ou algorithmes. Il peut s'agir par exemple d'algorithmes de chiffrement ou d'algorithmes de correction d'erreurs. L'ordinateur est une machine à calculer. Un ordinateur qui exécute un algorithme et transforme les données, effectue en réalité un nombre fini mais très grand de calculs élémentaires. Les communications s'effectuent suivant des règles, des protocoles. Algorithmes et protocoles doivent non seulement remplir les objectifs de sécurité que l'on s'est fixé mais ils doivent transformer les données de manière « efficace » . Cela signifie plusieurs choses. Deux critères importants d'efficacité sont le temps et la mémoire. Il faut que le temps de calcul ne soit pas trop long mais aussi que la taille des données ne devienne pas trop importante.

Au cours de ma thèse, j'ai étudié plusieurs sujets distincts ayant trait aux codes correcteurs. Ce document est structuré en trois parties selon le schéma suivant :

La première partie se concentre sur l'étude de la classe des codes cycliques avec l'ensemble de définition $\{1, 2^i+1, 2^j+1\}$ et la caractérisation des codes trois-correcteurs parmi cette classe. De manière plus large, on étudie la distribution de poids des codes trois-correcteurs dans cette classe et la question de l'équivalence de ces codes avec le code BCH trois-correcteur. Cette partie présente également une amélioration sur l'algorithme de Schaub. Ce dernier permet de déterminer une borne inférieure relativement fine sur la distance minimale des codes cycliques. L'algorithme a été mis en œuvre afin de déterminer l'immunité spectrale de fonctions booléennes. Cette quantité est reliée à la distance minimale de codes cycliques et est importante pour garantir la sécurité dans certains cryptosystèmes de chiffrement à flot.

La seconde partie concerne l'algorithmique dans les corps finis et ce, dans le contexte de la cryptographie basée sur les codes. Plus spécifiquement, nous améliorons les techniques pour trouver les racines d'équations algébriques dans un corps de caractéristique 2. Nous évaluons la complexité théorique dans le pire des cas des algorithmes existants et déterminons le meilleur compromis entre le Berlekamp Trace Algorithm (BTA) et des méthodes proposées par Zinoviev en fonction des paramètres du système.

La dernière partie s'intéresse à l'étude de variantes du protocole HB+. Il s'agit de protocoles d'authentification (basés sur le problème NP-complet LPN) adaptés au contexte particulier où l'on dispose de faibles moyens de calcul et de stockage des données. Nous examinons ici la sécurité de ces protocoles et des primitives sous-jacentes avec une approche basée sur le problème du décodage d'un code linéaire plutôt que sur le problème LPN. Nous appliquons à ces protocoles, les attaques connues contre les cryptosystèmes de McEliece.

1. Ce n'est pas tout à fait exact pour être honnête. On sait traiter des informations qui ont trait à deux de nos cinq sens : la vue et à l'ouïe. Mais il paraît beaucoup plus complexe, à l'heure actuelle, de coder une odeur ou un goût par exemple.

2 Codes algébriques

Nous allons tout d'abord présenter quelques structures algébriques. Nous nous servirons de ces objets mathématiques par la suite et ce, aussi bien en théorie des codes qu'en cryptologie. Nous donnerons également un certain nombre de propriétés que possèdent ces objets. Nous allons définir certaines familles de codes linéaires en bloc avec lesquelles nous aurons à travailler. Nous nous intéresserons essentiellement aux familles des codes cycliques ainsi qu'à celles des codes alternants. Il existe bon nombre de codes linéaires, nous en recensons certains dans la Figure 2.1. Nous n'expliciterons pas tous les liens qui les relient car cela sortirait du cadre de ce document. Mais, pour commencer, nous allons devoir définir ce qu'est un code linéaire en blocs ainsi que les paramètres essentiels qui permettent de les caractériser. Nous nous intéresserons encore à certaines des propriétés de ces codes.

Définition 1 (Structure Algébrique)**.** Un ensemble est muni d'une structure algébrique si une ou plusieurs lois de composition sont définies sur cet ensemble.

Exemple 1. De très nombreuses structures algébriques ont été étudiées. Il n'est pas question de les recenser ici. On peut néanmoins citer : les groupes, les anneaux, les corps, les espaces vectoriels.

2.1 Structures algébriques

Définition 2 (Monoïde)**.** Un monoïde est un ensemble muni d'une loi de composition interne associative admettant un élément neutre.

Exemple 2. L'ensemble des entiers positifs muni de l'addition ($\mathbb{N}$,+) forme un monoïde.

Définition 3 (Groupe)**.** Un groupe est monoïde admettant pour chaque élément de l'ensemble, un élément symétrique.

Remarque 1. Un groupe $(\mathcal{G}, +)$ est dit abélien ou commutatif si $a + b = b + a, \forall a, b \in \mathcal{G}$.

Exemple 3. L'ensemble des entiers relatifs munis de l'addition ($\mathbb{Z}$,+) forme un groupe. L'ensemble S_n des permutations de $[\![1, n]\!]$ muni de la loi de composition forme le groupe symétrique d'indice n.

Notation 1. *En notation additive, l'élément symétrique d'un élément x d'un groupe $(\mathcal{G}, +)$ est appelé l'opposé de x, on le note $-x$. En notation multiplicative, l'élément symétrique d'un élément x d'un groupe $(\mathcal{G}, \times)$ est appelé l'inverse de x, on le note x^{-1}.*

Définition 4 (Groupe Cyclique)**.** En notation multiplicative, un groupe cyclique est un groupe dont tous les éléments sont des puissances d'un certain élément α. L'élément α est un générateur du groupe. Il est encore appelé élément primitif du groupe. Il n'est pas nécessairement unique. Certains auteurs considèrent qu'un groupe cyclique doit de surcroît posséder un nombre fini d'éléments.

Exemple 4. L'ensemble des entiers modulo n munis de l'addition ($\mathbb{Z}_n$,+) forme un groupe cyclique. Il s'agit de l'unique groupe cyclique d'ordre n, à isomorphisme près.

Définition 5 (Groupe Multiplement Transitif)**.** Un sous-groupe $\mathcal{H}$ de S_n est k-fois transitif si pour tout choix de k éléments distincts $i_1, \ldots, i_k \in [\![1, n]\!]$, il existe une permutation $\sigma \in \mathcal{H}$ avec :

$$\sigma(1) = i_1, \ldots, \sigma(k) = i_k.$$

Définition 6 (Semi-anneau)**.** Un semi-anneau $(\mathcal{A}, +, \times)$ est un ensemble où :
- $(\mathcal{A}, +)$ est un monoïde commutatif,
- $(\mathcal{A}, \times)$ est un monoïde,
- la multiplication est distributive par rapport à l'addition,
- l'élément neutre pour l'addition est absorbant pour le produit.

Exemple 5. La structure $(\{0, 1, X\}, +, *)$ définie par les tables suivantes est un semi-anneau commutatif. Je la présente à dessein puisqu'on la rencontrera dans la suite de ce document.

$+$	0	1	X
0	0	1	X
1	1	X	X
X	X	X	X

$*$	0	1	X
0	0	0	0
1	0	1	X
X	0	X	X

On peut se familiariser facilement avec cette structure. Pour anecdote, il est reconnu qu'en des temps anciens, l'Homme ne savait pas compter au delà de trois. Tout entier supérieur à trois était considéré comme « beaucoup » . Considérons que le symbole X désigne la quantité « beaucoup » . Lorsqu'on travaille dans la structure présentée ci-dessus, on est dans une situation très proche, tout entier supérieur à deux est alors considéré comme « beaucoup » .

Définition 7 (Anneau)**.** Un anneau $(\mathcal{A}, +, \times)$ est un ensemble tel que $(\mathcal{A}, +)$ est un groupe abélien, $(\mathcal{A}, \times)$ est un monoïde et tel que la loi $\times$ soit distributive par rapport à la loi $+$. L'anneau est dit commutatif si la loi $\times$ est commutative.

Notation 2. 0 *désigne l'élément neutre de la loi* $+$ *et* 1 *représente l'élément neutre de la loi* $\times$.

Définition 8 (Caractéristique d'un anneau)**.** Soit un anneau $(\mathcal{A}, +, \times)$. La caractéristique de $\mathcal{A}$ est le plus petit entier non nul n tel que $n.1 = 0$. S'il n'existe pas de tel entier, la caractéristique de $\mathcal{A}$ est dite nulle.

Définition 9 (Anneau intègre)**.** Un anneau intègre est un anneau commutatif qui ne possède pas de diviseur de zéro et tel que $1 \neq 0$.

Exemple 6. La caractéristique de $\mathbb{Z}_n$ est n. L'anneau $\mathbb{Z}_n$ est intègre $\Leftrightarrow$ n est premier.

Définition 10 (Corps commutatif)**.** Un corps $(\mathcal{K}, +, \times)$ est un anneau tel que $(\mathcal{K}, \times)$ soit un groupe. Le corps est dit commutatif si $(\mathcal{K}, \times)$ est un groupe abélien.

Notation 3. *Soit p un nombre premier. Soit m un entier positif et $q = p^m$.*

Exemple 7. L'ensemble $\mathbb{Z}_p[X]$ des polynômes univariés, à coefficients dans le corps $\mathbb{Z}_p$ des entiers modulo p, forme un anneau. L'ensemble des fonctions booléennes à n variables forme un anneau.

Définition 11 (Corps premier)**.** Un corps est premier s'il ne contient aucun sous-corps strict.

Proposition 1. *Il n'existe que deux types de corps premiers.*

1. *Un corps premier de caractéristique 0 est isomorphe à $\mathbb{Q}$.*
2. *Un corps premier de caractéristique p est isomorphe à $\mathbb{Z}_p$.*

Dans la suite de ce document, nous manipulerons la structure des corps finis (également appelés corps de Galois). Il existe plusieurs manières de construire ces objets. Il existe une littérature conséquente sur le sujet, il est d'usage de citer l'encyclopédique [LN97].

Théorème 1 (Wedderburn)**.** *Tout corps fini est commutatif.*

Proposition 2. *Il existe un unique corps de Galois d'ordre donné, à isomorphisme près.*

Proposition 3. *La caractéristique du corps à q éléments est p.*

Définition 12 (Polynôme minimal)**.** Soit $\mathcal{K}$, un corps commutatif et $\mathcal{L}$ une extension algébrique de $\mathcal{K}$. Le polynôme minimal d'un élément a de $\mathcal{L}$ sur $\mathcal{K}$ est le polynôme unitaire à coefficients dans $\mathcal{K}$ de plus bas degré qui admet a pour racine.

Caractérisation 1. Le corps de Galois $\mathbb{F}_q$ d'ordre q est un corps fini de la forme $\mathbb{Z}_p[X]/f$, où $f \in \mathbb{Z}_p[X]$ est le polynôme minimal sur $\mathbb{F}_p$ d'un élément primitif de $\mathbb{F}_q$.

Définition 13 (Corps des racines)**.** Le corps des racines d'un polynôme à coefficients dans $\mathcal{K}$ est la plus petite extension de $\mathcal{K}$ contenant toutes les racines du polynôme.

Caractérisation 2. Le corps de Galois $\mathbb{F}_q$ d'ordre q est l'ensemble des racines du polynôme $X^q - X$ dans son corps des racines.

Exemple 8. Le corps $\mathbb{Z}_p$ des entiers modulo p est isomorphe au corps de Galois $\mathbb{F}_p$.

Proposition 4. *Soit ϕ l'indicatrice d'Euler. Le groupe multiplicatif $\mathbb{F}_q^\times$ est cyclique, il possède $\phi(q-1)$ éléments primitifs.*

Définition 14 (Groupe Affine)**.** Soient $m > 0$ et α un élément primitif du groupe multiplicatif $\mathbb{F}_{2^m}^\times$. Le groupe affine des permutations de $\mathbb{F}_{2^m}$ est l'ensemble :

$$\left\{ \alpha^i \mapsto u\alpha^i + v, 0 \mapsto v : u, v \in \mathbb{F}_{2^m}, u \neq 0 \right\}.$$

Remarque 2. Le groupe affine est un groupe 2-transitif. On peut consulter [BM75, pp. 58-63] pour de plus amples informations sur des groupes de transformations tels que le groupe affine.

Définition 15 (Algèbre sur un corps)**.** Une $\mathcal{K}$-algèbre $(\mathcal{A}, +, \times, \cdot)$ est un ensemble tel que $(\mathcal{A}, +, \times)$ soit un anneau, $(\mathcal{A}, +, \cdot)$ soit un $\mathcal{K}$-espace vectoriel et tel que la propriété

$$(\lambda \cdot a) \times (\mu \cdot b) = (\lambda \cdot \mu) \cdot (a \times b)$$

soit vérifiée pour tout λ, μ dans $\mathcal{K}$ et pour tout a, b dans $\mathcal{A}$.

Exemple 9. L'ensemble $\mathbb{F}_q[X]/(X^n - 1)$ est une $\mathbb{F}_q$-algèbre. Cet objet sera notamment utile pour la construction des codes cycliques.

2.2 Notions générales en théorie des codes

Notation 4. *Soit $\mathcal{A}$, un alphabet. Soient k, m, n des entiers positifs. Pour tout $i > 0$, on note $\mathcal{A}^i$, l'ensemble des messages de longueur i sur $\mathcal{A}$.*

Définition 16 (Code correcteur)**.** Un code correcteur est l'image d'une application injective définie sur le produit cartésien m-ième de $\mathcal{A}^k$ et à valeurs dans $\mathcal{A}^n$. Le terme mot de code désigne un élément de cet ensemble.

Remarque 3. Dans un souci de légèreté, nous parlerons dorénavant de codes et non pas de codes correcteurs. Attention à ne pas confondre ces codes là avec les « codes secrets » utilisés en cryptographie.

Remarque 4. Il existe également des codes détecteurs. Leur rôle consiste à détecter la présence d'erreurs et le cas échéant, à retransmettre le message erroné.

Remarque 5. Nous traiterons uniquement des codes correcteurs d'erreurs. Une erreur désigne une modification d'un symbole en un autre. Il existe encore les codes correcteurs d'erreurs en rafale ainsi que les codes correcteurs d'effacements. Le rôle de ces derniers consiste à reconstituer un mot de code émis à partir de l'ensemble des symboles reçus. Un effacement désigne la disparition d'un symbole.

Remarque 6. Nous traiterons uniquement des codes en blocs, pour lesquels, les opérations de codage et décodage d'un bloc dépendent uniquement des symboles d'information de ce bloc. Cela se traduit par $m = 1$ dans la Définition 16. L'autre famille de codes est celle des codes convolutifs. Pour ces derniers, le codage et le décodage d'un bloc dépendent des symboles d'information d'autres blocs (généralement de blocs précédemment transmis). On peut naturellement faire l'analogie avec le chiffrement par blocs et le chiffrement à flot en cryptographie.

Définition 17 (Code linéaire)**.** Un code linéaire est l'image d'une application linéaire injective définie sur $\mathbb{F}_q^k$ et à valeurs dans $\mathbb{F}_q^n$. On appelle k, la dimension du code et n, la longueur de blocs du code. Le code est dit q-aire. En particulier, le code est dit binaire si $q = 2$ et ternaire si $q = 3$.

Proposition 5. *Un code linéaire est un $\mathbb{F}_q$-sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k.*

Remarque 7. On pourrait très bien définir les codes sur des anneaux finis mais nous n'en aurons pas l'utilité ici.

Les mots d'un code linéaire peuvent s'écrire de plusieurs manières selon le choix de la base du code. Cette base n'est pas unique. On représente une base d'un code linéaire sous forme matricielle.

Notation 5. *Soit C, un code linéaire q-aire de dimension k et de longueur n.*

Définition 18 (Matrice génératrice)**.** Une matrice génératrice de C est une matrice notée généralement G de taille $k \times n$ et à coefficients dans $\mathbb{F}_q$ dont les lignes forment une base de C.

Propriété 1. *Pour toute matrice inversible M d'ordre k et à coefficients dans $\mathbb{F}_q$, MG est une matrice génératrice de C.*

Définition 19 (Distance de Hamming)**.** La distance de Hamming de deux mots x et y de longueur fixe est le nombre $d(x, y)$ de positions où ces deux mots diffèrent.

Définition 20 (Espace de Hamming)**.** L'espace $\mathbb{F}_q^n$ munie de la distance de Hamming est appelé espace de Hamming.

Définition 21 (Support d'un mot)**.** Le support d'un mot $x = (x_1, x_2, \ldots, x_n)$ de $\mathbb{F}_q^n$ est l'ensemble des positions qui indexent les composantes non nulles de ce mot.

Définition 22 (Support d'un code)**.** Le suppport d'un code est l'union des supports de ses mots.

Définition 23 (Poids de Hamming)**.** Le poids de Hamming d'un mot est le cardinal de son support. On notera $w(x)$ le poids de Hamming d'un mot x.

Propriété 2. *Pour tout $x, y \in \mathbb{F}_q^n$, $d(x, y) = w(x - y)$.*

Remarque 8. La distance de Hamming est une métrique ou distance (au sens mathématique du terme) sur $\mathbb{F}_q^n$. Nous utiliserons seulement cette métrique, néanmoins il en existe d'autres. On pourra citer la métrique rang, la métrique de Lee, la métrique arithmétique ainsi que la métrique de Hamming généralisée.

Définition 24 (Distance minimale)**.** La distance minimale d d'un code est le plus petit poids non nul d'un mot du code. Il s'agit d'un paramètre essentiel d'un code.

Définition 25 (Capacité de détection)**.** La capacité de détection d'un code est la quantité $d-1$.

Définition 26 (Capacité de correction)**.** La capacité de correction t d'un code est la quantité $\lfloor \frac{d-1}{2} \rfloor$. Elle désigne le nombre maximal d'erreurs que peut corriger un code sans ambiguïté. Corriger signifie dans notre contexte, choisir le mot de code le plus proche, en termes de distance de Hamming, du mot reçu. On parle de décodage suivant la règle du plus proche voisin. Il y a ambiguïté s'il existe plusieurs possibilités de correction. Auquel cas, on peut soit choisir un mot au hasard parmi les mots de code les plus proches du mot reçu, soit demander une retransmission.

Remarque 9. Il existe d'autres types de décodage ; on citera le décodage à vraisemblance maximale et le décodage en liste [Eli57].

Notation 6. *Un code $[n, M, d]_q$ est un code q-aire de longueur n, de cardinal M et de distance minimale d. Un code linéaire $[n, k, d]_q$ est un code linéaire q-aire de longueur n, de dimension k et de distance minimale d. On a l'égalité : $M = q^k$.*

Il existe plusieurs critères d'optimalité sur les paramètres du code. Nous allons en définir certains dont nous reparlerons dans la Figure 2.1.

Définition 27 (Code parfait)**.** Un code $[n, M, d]_q$ est dit parfait si l'ensemble des boules fermées de rayon t pour la distance de Hamming et centrées sur les mots du code forment une partition de $\mathbb{F}_q^n$. Intuitivement, cela signifie que toute erreur détectée soit corrigée. Cela ne signifie néanmoins pas que l'erreur soit correctement corrigée.

Définition 28 (Code MDS)**.** Un code $[n, k, d]_q$ est dit MDS (maximum distance separable) s'il n'existe pas de code linéaire $[n, k]_q$ possédant une distance minimale strictement plus grande. Formellement, cela signifie que l'on a l'égalité $d = n - k + 1$.

2.3 Codes cycliques

La notion de codes cycliques apparaît pour la première fois par l'intermédiaire de Prange, dans [Pra57]. Ils suscitent beaucoup d'intérêt de par leur structure algébrique. Ils sont utilisés à la fois pour corriger les erreurs isolées et les erreurs en rafale.

On peut définir un code cyclique de différentes manières. Nous donnerons ici des descriptions algébriques qui nous serviront par la suite.

Notation 7. *Soit p un nombre premier. Soit $t > 0$ et q une puissance première de p. Soit $\alpha \in \mathbb{F}_{q^t}$.*

Définition 29 (Éléments conjugués)**.** Des éléments de $\mathbb{F}_{q^t}$ sont dits conjugués s'ils possèdent le même polynôme minimal sur $\mathbb{F}_q$.

Définition 30 (Classe de conjugaison)**.** La classe de conjugaison de α est l'ensemble des éléments conjugués à α.

Remarque 10. Les éléments conjugués à α sont les itérés de α par l'automorphisme de Frobenius sur $\mathbb{F}_q$. On rappelle qu'il s'agit de l'application $x \mapsto x^p$.

Notation 8. *Soit n un entier premier avec q. Soit $i \in \mathbb{Z}_n$.*

Définition 31 (Classe cyclotomique)**.** La q-classe cyclotomique de i modulo n est définie par :

$$\left\{ \left(iq^j \mod n\right) \in \mathbb{Z}_n : j \in \mathbb{N} \right\}.$$

Propriété 3. *L'ensemble des q-classes cyclotomiques modulo n forment $\mathbb{Z}_n$.*

Propriété 4. *L'ensemble des exposants de tous les éléments d'une classe de conjugaison forment une classe cyclotomique.*

FIGURE 2.1 – Certains codes linéaires en blocs

Propriété 5. *Les classes q-cyclotomiques modulo n et les facteurs irréductibles du polynôme $X^n - 1$ dans $\mathbb{F}_q[X]$ sont en correspondance bijective. Les cardinaux des classes cyclotomiques sont égaux aux degrés des polynômes irréductibles associés.*

Exemple 10. La classe de conjugaison de α est l'ensemble $\{\alpha^{q^i} : i \in \mathbb{N}\}$. On vérifie bien que l'ensemble $\{q^i \mod n : i \in \mathbb{N}\}$ forme la classe q-cyclotomique de 1 modulo n.

Notation 9. *Soient n tel que $p \nmid n$ et m tel que $n \mid q^m - 1$. Soit α une racine primitive n-ième de l'unité dans $\mathbb{F}_{q^m}$.*

Définition 32 (Ensemble de définition)**.** L'ensemble de définition d'un code q-aire de longueur n est une union de classes q-cyclotomiques modulo n.

Remarque 11. On représentera souvent l'ensemble de définition sous une forme compacte en ne notant que les représentants des classes cyclotomiques contenus dans cet ensemble.

Remarque 12. Les $\mathbb{F}_q$-algèbres $\mathbb{F}_q^n$ et $\mathbb{F}_q[X]/(X^n - 1)$ sont isomorphes. Nous ferons souvent un abus de langage en confondant les mots d'un code cyclique et leur polynôme associé.

Définition 33 (Code cyclique)**.** Le code cyclique de longueur n sur $\mathbb{F}_q$ avec l'ensemble de définition Z et la racine primitive n-ième de l'unité α est défini comme étant :

$$\left\{ c \in \mathbb{F}_q^n : c(\alpha^z) = 0, \forall z \in Z \right\}.$$

Notation 10. *Soit g un diviseur unitaire dans $\mathbb{F}_q[X]$ du polynôme $X^n - 1$.*

Définition 34 (Code cyclique)**.** Le code cyclique de longueur n sur $\mathbb{F}_q$ avec le polynôme générateur g est défini comme étant l'idéal engendré par g dans l'anneau quotient $\mathbb{F}_q[X]/(X^n - 1)$.

Remarque 13. Les deux définitions de codes cycliques données ci-dessus sont équivalentes. Le polynôme générateur g est le ppcm des polynômes minimaux sur $\mathbb{F}_q$ de α^z où $z \in Z$.

Remarque 14. En toute rigueur, les codes cycliques que l'on a définis ci-dessus s'appellent les codes cycliques à racine simple car leur polynôme générateur ne possède que des racines simples dans $\mathbb{F}_q$. On aurait pu définir des codes cycliques à racines multiples en omettant l'hypothèse que la longueur du code est première avec la caractéristique de son alphabet mais nous n'en aurons pas l'utilité.

Remarque 15. Un code cyclique est dit primitif si sa longueur $n = q^m - 1$. Dans cette configuration, une racine primitive n-ième de l'unité α est un élément primitif du groupe $\mathbb{F}_{q^m}^\times$.

Notation 11. *Soit $i \in \mathbb{Z}_n$. Soit C_i, la classe q-cyclotomique de i modulo n. Soit un ensemble S tel que $\bigcup_{s \in S} C_s = Z$ et $\bigcap_{s \in S} C_s = \varnothing$.*

Propriété 6. *La dimension k d'un code cyclique (en tant que $\mathbb{F}_q$-espace vectoriel) vaut :*

$$k = n - \deg(g) \geq n - m|\bigcup_{s \in S} C_s| = n - m \mid Z \mid .$$

La dimension k est indépendante du choix de α.

Exemple 11. Le code de parité $[n, n-1]$ est l'exemple le plus basique de code cyclique.

Notation 12. *On fixe deux entiers a et $\delta \in [\![0, n]\!]$. Soit α, une racine primitive n-ième de l'unité. Pour tout $z \in Z$, on note $M^{(z)}(X)$, le polynôme minimal de α^z sur $\mathbb{F}_q$.*

Exemple 12. Le code BCH de longueur n et de distance construite δ sur $\mathbb{F}_q$ avec la racine primitive n-ième de l'unité α est le code cyclique pour lequel on a l'égalité $S = [\![a, a+\delta-2]\!]$. En d'autres termes, le polynôme générateur g de ce code peut s'écrire sous la forme :

$$g(X) = \text{ppcm}(\{M^{(z)}(X)\}_{z\in Z}) = \prod_{s\in S} M^{(s)}(X).$$

Remarque 16. La distance construite d'un code BCH est une borne inférieure sur sa distance minimale. Autrement dit, nous construisons des codes cycliques possédant une distance minimale plus grande que (ou égale à) la distance de construction. Nous étudierons plus en détail le sujet de la distance minimale d'un code cyclique dans le Chapitre 3.

Notation 13. *Soit la fonction w_2 associant à un entier, le nombre de 1 dans son écriture binaire. Soient des entiers $m > 0$ et $0 \leq r \leq m$.*

Définition 35 (Code de Reed-Muller cyclique)**.** Le code de Reed-Muller cyclique $\mathcal{R}^*(r, m)$ d'ordre r avec m variables est un code cyclique binaire de longueur $2^m - 1$ dont l'ensemble de définition est :

$$\left\{ i : 0 < i < 2^m, \quad 0 < w_2(i) < m - r \right\}.$$

Théorème 2. *Le code $\mathcal{R}^*(r, m)$ est un sous-code du code* BCH *primitif binaire avec l'ensemble de définition $[\![1, 2^{m-r} - 2]\!]$.*

Démonstration. Les éléments de $[\![1, 2^{m-r} - 2]\!]$ sont des entiers dont la représentation binaire contient strictement moins de $m - r$ uns. L'ensemble de définition de $\mathcal{R}^*(r, m)$ contient donc $[\![1, 2^{m-r} - 2]\!]$. □

2.4 Codes dérivés

Nous allons dorénavant donner des manières de construire de nouveaux codes à partir d'un code linéaire donné. Il existe bien d'autres constructions, on peut se rapporter par exemple à [Bla03, pp. 62-63], [HP03b, pp. 13-18] et [MS83, pp. 27-29,76-77]. On pourrait également combiner des codes entre eux (*e.g.* les codes concaténés, les codes produits, les turbo-codes).

Notation 14. *Soit C un code linéaire $[n, k, d]_q$*

Définition 36 (Dual)**.** Le dual de C est l'orthogonal $C^\perp$ de C dans $\mathbb{F}_q^n$.

Remarque 17. Il s'agit d'un code linéaire $[n, n-k]_q$. On a l'égalité $\dim(C) + \dim(C^\perp) = n$. On ne sait pas en revanche exprimer la distance minimale du dual en fonction des paramètres de C. D'autre part, la dimension du dual est parfois appelée la *codimension* du code.

Propriété 7. *Le dual d'un code cyclique est un code cyclique.*

Définition 37 (Polynôme réciproque)**.** Soit $h(X)$, un polynôme de degré k. Le polynôme $h^*(X) \coloneqq X^k h(\frac{1}{X})$ est le polynôme réciproque de $h(X)$.

Propriété 8. *Les racines de $h^*(X)$ sont les inverses des racines de $h(X)$.*

Propriété 9. *Soit C un code cyclique de longueur n ayant un polynôme générateur g. Le polynôme générateur du dual de C est le polynôme réciproque unitaire de $\frac{X^n-1}{g(X)}$.*

Définition 38 (auto-dual)**.** Le code C est auto-dual si $C = C^\perp$.

Définition 39 (auto-orthogonal)**.** Le code C est auto-orthogonal si $C \subset C^\perp$.

Définition 40 (Hull)**.** Le hull de C est l'intersection de C et de $C^\perp$.

Définition 41 (Matrice de parité)**.** Une matrice de parité de C est une matrice notée généralement H de taille $(n-k) \times n$ dont les lignes forment une base de $C^\perp$. Il s'agit d'une matrice génératrice de $C^\perp$.

Notation 15. *Pour tout $1 \leq i \leq n$, on notera c_i la i-ème composante d'un mot de longueur n. L'ensemble E désignera un sous-ensemble de $\{c_1, \ldots, c_n\}$.*

Définition 42 (Code étendu)**.** Le code étendu $\overline{C}$ d'un code q-aire C défini est :

$$\left\{ (c_1, \ldots, c_n, -\sum_{i=1}^{n} c_i) : (c_1, \ldots, c_n) \in C \right\}.$$

Remarque 18. Pour $q = 2$, la dernière coordonnée de $\overline{C}$ est celle du bit de parité.

Exemple 13. Soient $m > 0$ et $0 \leq r \leq m$. Le code de Reed-Muller binaire d'ordre r avec m variables est le code étendu de $\mathcal{R}^*(r, m)$.

Définition 43 (Code de poids pair)**.** Un code binaire est un code de poids pair si tous ses mots ont un poids pair.

Exemple 14. Le code étendu d'un code binaire est un code de poids pair.

Définition 44 (Code poinçonné)**.** Le code poinçonné de C en E est obtenu en effaçant les composantes dans E de chaque mot de C.

Définition 45 (Code raccourci)**.** Le code raccourci de C en E est obtenu en poinçonnant en E, le sous-code de C contenant les mots dont toutes les composantes dans E valent 0.

Propriété 10. *Le dual d'un code poinçonné est le code raccourci du dual.*

Propriété 11. *Le dual d'un code raccourci est le code poinçonné du dual.*

On peut également construire des codes sur $\mathbb{F}_q$ à partir de codes définis sur des extensions de $\mathbb{F}_q$.

Notation 16. *Soit $m > 0$. Soit C un code linéaire $[n, k, d]_{q^m}$.*

Définition 46 (Sous-code sur un sous-corps)**.** Le sous-code $C|_{\mathbb{F}_q}$ sur $\mathbb{F}_q$ de C est défini par :

$$C|_{\mathbb{F}_q} := C \cap \mathbb{F}_q^n.$$

Remarque 19. Le code $C|_{\mathbb{F}_q}$ possède une distance minimale supérieure ou égale à celle de C.

Exemple 15. Tout code alternant est un sous-code sur un sous-corps d'un code GRS (par définition).

Nous définissons la *fonction Trace* comme étant $\mathrm{Tr}_{\mathbb{F}_q^m/\mathbb{F}_q}(\cdot) : \mathbb{F}_{q^m} \to \mathbb{F}_q$

$$\mathrm{Tr}_{\mathbb{F}_q^m/\mathbb{F}_q}(z) := z + z^q + z^{q^2} + \ldots + z^{q^{m-1}}.$$

Définition 47 (Code trace)**.** Le code trace $\mathrm{Tr}_{\mathbb{F}_q^m/\mathbb{F}_q}(C)$ sur $\mathbb{F}_q$ est défini par :

$$\mathrm{Tr}_{\mathbb{F}_q^m/\mathbb{F}_q}(C) := \{(\mathrm{Tr}_{\mathbb{F}_q^m/\mathbb{F}_q}(c_1), \ldots, \mathrm{Tr}_{\mathbb{F}_q^m/\mathbb{F}_q}(c_n)) : (c_1, \ldots, c_n) \in C\}.$$

Exemple 16. Le dual d'un code alternant est le code trace d'un code GRS [MS83, page 337].

Théorème 3 (Delsarte)**.**

$$C|_{\mathbb{F}_q} = \mathrm{Tr}_{\mathbb{F}_q^m/\mathbb{F}_q}(C^\perp)$$

Notation 17. *Soient $m > 0$ et C un code cyclique q-aire de longueur $n = q^m - 1$. Soit $\{s_1, \ldots, s_t\}$ un ensemble complet de représentants des classes cyclotomiques de l'ensemble de définition de C.*

Théorème 4. *Le dual $C^\perp$ s'écrit sous la forme :*

$$\left\{ (\mathrm{Tr}_{\mathbb{F}_q^m/\mathbb{F}_q}(\sum_{i=1}^{t} \lambda_i x^{s_i}))_{x \in \mathbb{F}_{q^m}^\times} : (\lambda_1, \ldots, \lambda_t) \in \mathbb{F}_{q^m} \right\}.$$

Une preuve de ce résultat est fournie dans [Sti08, pp. 315-318 - Proposition 9.2.4].

2.5 Codes alternants

Les codes alternants ont été introduits par Helgert [Hel74]. Des résultats importants sur ces codes sont contenus dans [Del75]. De nombreux codes appartiennent à cette grande famille de codes linéaires (voir la Figure 2.1). Nous étudierons les codes de Goppa classiques qui forment une sous-famille parmi ces codes. [Gop70, Ber73]. On notera que les codes de Goppa ont été découverts avant les codes alternants. Nous travaillerons aussi avec les codes BCH qui sont des codes cycliques alternants. Pour autant, tous les codes alternants ne sont pas cycliques et réciproquement, tous les codes cycliques ne sont pas alternants. Les codes alternants sont dérivés des codes de Reed-Solomon généralisés (GRS). Les codes de Reed-Solomon ont été découverts en 1960 [RS60]. Il s'agit de codes d'évaluation. Cela signifie que les mots d'un tel code peuvent être obtenus en évaluant des polynômes en un nombre fixé de points.

Notation 18. *Soit $m \geq 1$. Soient n un entier et q une puissance première tels que $n \leq q^m$. Soit le vecteur $\alpha := (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{F}_{q^m}^n$ tel que $\alpha_i \neq \alpha_j$ pour tout $1 \leq i < j \leq n$. Soit le vecteur $\nu := (\nu_1, \nu_2, \ldots, \nu_n)$ tel que $\nu_i \in \mathbb{F}_{q^m}^\times$ pour tout $i \in [\![1, n]\!]$. Soit $k \in [\![0, n]\!]$. La notation $\mathbb{F}_q[X]_k$ désigne l'ensemble des polynômes à coefficients dans $\mathbb{F}_q$ et de degré strictement inférieur à k.*

Définition 48 (Code de Reed-Solomon généralisé)**.** Le code $\mathrm{GRS}_k(\alpha, \nu)$ de longueur n sur $\mathbb{F}_{q^m}$ est :

$$\left\{ (\nu_1 f(\alpha_1), \nu_2 f(\alpha_2), \ldots, \nu_n f(\alpha_n)) \in \mathbb{F}_{q^m}^n : \forall f \in \mathbb{F}_{q^m}[x]_k \right\}.$$

Propriété 12. *Le dual du code $\mathrm{GRS}_k(\alpha, \nu)$ est le code $\mathrm{GRS}_{n-k}(\alpha, \gamma)^\perp$ pour un $\gamma \in (\mathbb{F}_{q^m}^\times)^n$.*

Notation 19. *Soit le vecteur $\gamma \in (\mathbb{F}_{q^m}^\times)^n$ tel que $\mathrm{GRS}_k(\alpha, \nu) = \mathrm{GRS}_{n-k}(\alpha, \gamma)^\perp$.*

Définition 49 (Code alternant)**.** Le code alternant $\mathcal{A}(\alpha, \gamma)$ de longueur n sur $\mathbb{F}_q$ est le code :

$$\mathrm{GRS}_k(\alpha, \nu)|_{\mathbb{F}_q}.$$

Remarque 20. Le terme « alternant » provient du fait que la matrice de parité d'un code alternant peut s'écrire sous la forme d'une matrice alternante. Pour r et n positifs, ces matrices s'écrivent :

$$\begin{pmatrix} f_1(\alpha_1) & f_2(\alpha_1) & \ldots & f_r(\alpha_1) \\ f_1(\alpha_2) & f_2(\alpha_2) & \ldots & f_r(\alpha_2) \\ \vdots & & \ddots & \vdots \\ f_1(\alpha_n) & f_2(\alpha_n) & \ldots & f_r(\alpha_n) \end{pmatrix}$$

Les codes BCH binaires ont été découverts indépendamment par Hocquenghem [Hoc59] en 1959 et Bose et Chaudhuri en 1960 [BRC60]. En 1961, les codes BCH non-binaires furent introduits par Gorenstein et Zierler [GZ61].

Notation 20. *Soient a et $\delta \in [\![0, n]\!]$. Soit α, une racine primitive n-ième de l'unité.*

Exemple 17. Le code BCH (voir Exemple 12) de longueur n et de distance construite δ sur $\mathbb{F}_q$ avec la racine primitive n-ième de l'unité α est le code alternant pour lequel on a les égalités

$$k = n - \delta + 1, \ \alpha = (1, \alpha, \alpha^2, \ldots, \alpha^{n-1}) \text{ et } \nu = (1, \alpha^a, \alpha^{2a}, \ldots, \alpha^{(n-1)a}).$$

Première partie

La correction d'erreurs et les codes cycliques

Introduction

La famille des codes cycliques est une classe bien connue de codes correcteurs d'erreurs. Nous traitons des codes cycliques binaires 3-correcteurs et de leurs duaux. Soit $\mathbb{F}_{2^m}$ l'extension du corps à deux éléments $\mathbb{F}_2$ de degré m et soit n un entier impair. Considérons un code cyclique binaire C de longueur n et soit α une racine primitive n-ième de l'unité dans $\mathbb{F}_{2^m}$. On peut décrire C comme un idéal principal, dans l'anneau $\mathbb{F}_2[X]/(X^n-1)$, avec le *polynôme générateur* g sur $\mathbb{F}_2$, où $n \mid (2^m-1)$. Par conséquent, les zéros de g peuvent être utilisés pour définir C. L'*ensemble de définition* Z de C est l'ensemble des exposants i de l'élément primitif α tel que α^i est une racine de g. Notez que si z est une racine de g, z^{2^i} est également une racine de g quel que soit i. En d'autres termes, Z est une union de 2-classes cyclotomiques modulo n. La plupart du temps, Z est décrit de façon concise par la liste des différents représentants de ces classes cyclotomiques. Si la longueur du code est 2^m-1, *i.e.* α est un générateur de $\mathbb{F}_{2^m}$, alors le code est dit *primitif*.

Les codes BCH forment une classe importante des codes cycliques. Un code cyclique généré par $g(X) = \mathrm{ppcm}(\{M^{(i)}(X)\}_{i\in I})$ où $M^{(i)}(X)$ est le polynôme minimal de α^i par rapport à $\mathbb{F}_2$ et où I est un ensemble de $\delta-1$ entiers consécutifs, est un code BCH avec une *distance construite* δ. En particulier, si $I = \{1, 2, \ldots, \delta-1\}$, alors le code BCH est dit *au sens strict*.

Kasami [Kas71] a introduit certaines classes de codes cycliques primitifs binaires 3-correcteurs (*i.e.* leur distance minimale est 7) similaires aux codes BCH 3-correcteurs, dans le sens où leur ensemble de définition est l'union de 3 classes cyclotomiques. Ces codes ont une dimension $k \geq n-3m$. Cela signifie aussi que, ces codes ont asymptotiquement un haut taux d'information et de plus, leur distance minimale est optimale. En effet, la borne de Hamming implique que des codes cycliques longs définis par τ classes cyclotomiques distinctes ont une capacité de correction $t \leq \tau$. Une de ces classes est $\{1, 2^\ell+1, 2^{3\ell}+1\}$ avec $\mathrm{pgcd}(\ell, m) = 1$ et m impair. Plus tard Bracken et Helleseth dans [BH09b] ont découvert la classe $\{1, 2^\ell+1, 2^{2\ell}+1\}$ avec $\mathrm{pgcd}(\ell, m) = 1$.

Nous examinons la classe $\{1, 2^i+1, 2^j+1\}$ pour $i, j > 1$, $i \neq j$. Nous tentons de généraliser la sous-classe $\{1, 2^\ell+1, 2^{3\ell}+1\}$ et $\{1, 2^\ell+1, 2^{2\ell}+1\}$ avec $\mathrm{pgcd}(\ell, m) = 1$ à la classe $\{1, 2^\ell+1, 2^{p\ell}+1\}$ avec $\mathrm{pgcd}(\ell, m) = 1$ et $p > 1$.

Une autre classe de codes cycliques 3-correcteurs avec l'ensemble de définition $\{1, 2^{\ell-1}+1, 2^\ell+1\}$ pour $m = 2\ell+1$ a été introduite dans [MS83]. La question a été soulevée de savoir si ces codes ont la même distribution de poids que le code BCH. Plus tard, il a été prouvé que cela était vrai dans [vDV96, vDV97]. Ils ont montré que les duaux de ces codes ont la même distribution de poids que le dual des codes BCH. Cela nous a motivés à étudier la distribution de poids de la classe générale $\{1, 2^i+1, 2^j+1\}$. Par le calcul, jusqu'à $m < 14$, nous vérifions que le dual de tous les codes 3-correcteurs qui ont un ensemble de définition $\{1, 2^i+1, 2^j+1\}$ et qui ne sont pas BCH, ont la même distribution de poids que le dual des codes BCH 3-correcteurs. Nous montrons que ce résultat reste vrai pour tout m impair en nous appuyant sur un théorème dû à Kasami [Kas69, Théorème 15 (ii)(1)].

Par ailleurs, nous étudions la distance minimale des duaux des codes 3-correcteurs avec un ensemble de définition $\{1, 2^i+1, 2^j+1\}$ sur $\mathbb{F}_2$ et sur $\mathbb{F}_{2^m}$. Dans la littérature, de nombreuses bornes inférieures théoriques sur la distance minimale des codes cycliques sont connues (*e.g.* [BRC60, HT72, Roo83, vLW86, Wol89]). Elles reposent soit sur les propriétés de distribu-

tions régulières de certain motifs contenus dans l'ensemble de définition, ou sur le nombre de points rationnels des courbes algébriques sur les corps finis. Schaub [Sch88] a pris une approche algorithmique pour calculer une borne inférieure sur la distance minimale d'un code cyclique donné. Cette idée est particulièrement efficace pour les codes qui ont peu de sous-codes cycliques. Nous améliorons la complexité en temps de l'algorithme de Schaub en utilisant un *critère d'élagage* basé sur la borne BCH afin d'être capable de gérer des codes qui possèdent davantage de sous-codes cycliques. Nous comparons la borne de Schaub avec la borne de Hartmann-Tzeng et la distance minimale des duaux des codes avec l'ensemble de définition $\{1, 2^i + 1, 2^j + 1\}$. Augot et Levy-dit-Vehel [AL96] ont également appliqué l'algorithme de Schaub pour trouver une borne inférieure de la distance minimale des duaux de codes BCH et ont trouvé que cet algorithme donne de meilleurs résultats que la borne de Ross et que la borne de Weil sur ces codes. Nos résultats numériques révèlent un comportement similaire de la distance minimale des duaux des codes cycliques 3-correcteurs dans la classe $\{1, 2^i + 1, 2^j + 1\}$ pour $i, j > 1$, $i \neq j$.

Finalement, nous étudions l'immunité spectrale d'une fonction booléenne, il s'agit d'une propriété cryptographique. Une haute valeur de l'immunité spectrale est une condition nécessaire afin de résister à la cryptanalyse algébrique de générateurs filtrés. Dans [HR11] le lien entre l'immunité spectrale et la distance minimale d'un code cyclique a été montré. L'immunité spectrale d'une fonction booléenne f sur $\mathbb{F}_{2^m}$ (en forme univariée) est égale au poids minimal du code cyclique 2^m-aire de longueur $n = 2^m - 1$ généré par $\mathrm{pgcd}(f(z), z^n + 1)$ ou $\mathrm{pgcd}(f(z) + 1, z^n + 1)$. Nous trouvons une borne inférieure sur l'immunité spectrale de la fonction booléenne $\mathrm{Tr}(g)$ en utilisant l'algorithme de Schaub, où g est le polynôme générateur du code avec l'ensemble de définition $\{1, 2^i + 1, 2^j + 1\}$.

3 Minorer la distance minimale des codes cycliques

3.1 Bornes théoriques

3.1.1 La borne de Carlitz-Uchiyama (1957)

La borne de Carlitz-Uchiyama [CU57] s'applique sur des codes cycliques particuliers. Il s'agit des duaux des codes BCH binaires. Cette borne repose sur un théorème de théorie des nombres démontré par André Weil [Wei48]. Ce théorème fournit une borne sur le nombre de points rationnels d'une courbe algébrique sur un corps fini. Cette borne a été améliorée par Jean-Pierre Serre [Ser83] et adaptée par Jacques Wolfmann au contexte des codes cycliques [Wol89]. Elle est également appelée borne de Weil-Carlitz-Uchiyama [MZK95]. Nous donnons dans un premier temps un résultat général pour les duaux des codes BCH.

Théorème 5. *Soit p un nombre premier et soient m et t des entiers strictement positifs. Soit C un code* BCH *au sens strict sur $\mathbb{F}_p$ de longueur $n = p^m - 1$ et de distance construite $\delta = 2t+1 > 1$. Soit w, le poids d'un mot de code du dual de C. On a $w = 0$, $w = n$ ou*

$$| w - p^m \left(1 - \frac{1}{p}\right) | \leq \frac{(p-1)(2t-1)\lfloor 2p^{\frac{m}{2}} \rfloor}{2p}.$$

Nous donnons à présent la borne de Carlitz-Uchiyama qui est un raffinement de la borne du Théorème 5 dans le cas binaire. Une preuve de ces résultats est donnée dans [Sti08, page 323].

Théorème 6. *Soient m et t des entiers strictement positifs. Soit C un code* BCH *au sens strict sur $\mathbb{F}_2$ de longueur $n = 2^m - 1$ et de distance construite $\delta = 2t + 1 > 1$. Soit w, le poids d'un mot de code du dual de C. On a $w = 0$, $w = n$ ou*

$$| w - 2^{m-1} | \leq \frac{(t-1)\lfloor 2^{\frac{m}{2}+1} \rfloor}{2}.$$

Remarque 21. Le dual d'un code BCH au sens strict est un code de poids pair.

Rodier [Rod96] a montré que la borne de Carlitz-Uchiyama est asymptotiquement fine. D'autres travaux pour estimer la distance minimale des duaux de codes BCH ont été menés, on citera notamment [AL96].

3.1.2 La borne BCH (1960)

Nous allons présenter l'une des preuves de la borne BCH. Cette preuve fait appel à la notion de polynôme de Mattson-Solomon qui intervient dans la version discrète de la transformée de Fourier.

Notation 21. *Soit α une racine primitive n-ième de l'unité dans $\mathbb{F}_{q^m}$. On note $< \alpha >$, le groupe multiplicatif engendré par α. Soit $a \in \mathbb{F}_q[X]_n$, un polynôme à coefficients dans $\mathbb{F}_q$ et de degré strictement inférieur à n.*

Définition 50 (Polynôme de Mattson-Solomon)**.** Le polynôme de Mattson-Solomon de a est :

$$A(X) \coloneqq \sum_{i=0}^{n-1} a(\alpha^{n-i}) X^i.$$

Remarque 22. On note que le nombre de non zéros de a dans $< \alpha >$ est égal au poids de A.

Définition 51 (Transformée de Fourier discrète)**.** La transformation de Fourier discrète est :

$$\begin{aligned} T : \mathbb{F}_q[X]_n &\to \mathbb{F}_q[X]_n \\ a(X) &\mapsto A(X) \end{aligned}$$

Lemme 1. *La transformée de Fourier inverse est donnée par :*

$$\begin{aligned} T^{-1} : \mathbb{F}_q[X]_n &\to \mathbb{F}_q[X]_n \\ A(X) &\mapsto n^{-1}(T \circ A)(X^{-1}) \mod X^n - 1 \end{aligned}$$

Remarque 23. Cela implique que :

$$a = n^{-1}(A(1), A(\alpha), \ldots, A(\alpha^{n-1}))$$

Il s'ensuit que le nombre de non zéros de A dans $< \alpha >$ est égal au poids de a.

Théorème 7. *Un code cyclique dont l'ensemble de définition contient δ entiers consécutifs possède une distance minimale strictement plus grande que δ.*

Démonstration. Nous traitons uniquement le cas d'un code BCH au sens strict. Soit c un mot de code non nul. Le degré du polynôme de Mattson-Solomon de c est au plus $n - \delta$. Les coefficients $c(\alpha^{n-i})$ du polynôme de Mattson-Solomon de c sont tous nuls pour $i > n - \delta$ puisque c est un multiple du polynôme générateur ayant pour ensemble de définition $[\![1, \delta]\!]$. Ce polynôme possède moins de $n - \delta$ racines. La Remarque 23 nous permet de conclure que le poids de c est strictement supérieur à δ. □

Remarque 24. La borne BCH s'applique à tous les codes cycliques. De par leur construction, les codes BCH sont particulièrement adaptés à son usage.

Remarque 25. La borne BCH dépend du nombre de zéros consécutifs dans l'ensemble de définition. L'ensemble de définition dépend à son tour de l'élément primitif choisi. On peut obtenir des bornes BCH différentes en fonction de ce dernier. Cette remarque tient également pour les bornes définies ci-après et qui dépendent de motifs réguliers contenus dans l'ensemble de définition. On consultera [HP03b, page 152] et [Bla03, page 149] afin d'être complet à ce propos ; cette dernière référence emploie le vocabulaire de la transformée de Fourier.

Remarque 26. La borne BCH donne un critère simple sur l'ensemble de définition pour obtenir un code avec une grande distance minimale. La capacité de correction n'est pour autant pas l'unique critère de performance d'un code. On cherche également à maximiser le rendement du code (le rapport entre la dimension et la longueur du code). La dimension d'un code cyclique de longueur $n = q^m - 1$ avec un ensemble de définition Z est supérieure à $n - m \mid Z \mid$. Pour cette raison, les classes cyclotomiques qui composent l'ensemble de définition ne doivent pas comporter trop d'éléments. On doit donc faire des compromis entre la capacité de correction du code et son rendement.

3.1.3 La borne de Hartmann-Tzeng (1972)

Une première amélioration de la borne BCH a été donnée par Hartmann et Tzeng dans [HT72]. Informellement, elle signifie qu'un code cyclique dont l'ensemble de définition contient $s \geq 1$ intervalles, espacés de manière « convenable » , de δ entiers consécutifs possède une distance minimale plus grande que $\delta + s$. On retombe bien sur la borne BCH pour $s = 1$.

Notation 22. *Soit C, un code cyclique de longueur n. Soient m et δ tels que $\gcd(m, n) \leq \delta$.*

Théorème 8. *Un code cyclique dont l'ensemble de définition contient les intervalles $[\![a, a + \delta]\!]$ pour tout $a \in \{mi \mod n : 1 \leq i \leq s\}$ possède une distance minimale strictement plus grande que $\delta + s$.*

3.1.4 Les bornes de van Lint-Wilson (1986)

Les bornes de van Lint-Wilson ont été introduites dans [vLW86]. Leurs auteurs appellent les techniques pour calculer ces bornes, la méthode du produit et la méthode du décalage. La méthode du décalage [HP03a, pp. 154-155] permet d'obtenir parfois une borne plus fine que la méthode du produit.

Notation 23. *Soient M et N des matrices à coefficients dans $\mathbb{F}_q$ avec n colonnes. On désigne par $M * N$ une matrice dont les lignes sont tous les produits d'Hadamard des lignes de M par celles de N. Soit C le code cyclique avec une matrice de parité $M * N$. Soit M_I (resp. N_I) la restriction de la matrice M (resp. N) aux colonnes dont la position est indiquée dans $I \subset [\![1, n]\!]$.*

Théorème 9 (Méthode du produit)**.** *Si pour tout $I \subset [\![1, n]\!]$ tel que $0 < |I| < \delta$, la somme du rang de M_I et N_I est strictement supérieure à δ alors C possède une distance minimale supérieure à δ.*

Remarque 27. Un résultat de Roos [Roo82] généralise le Théorème 8 de Hartmann-Tzeng. Il correspond à un cas particulier de la méthode du produit [Van99, page 94].

Notation 24. *Soit C un code cyclique. On considère S l'ensemble des zéros d'un mot de C dans un corps $\mathbb{F}$.*

Définition 52 (Ensemble S-indépendant)**.** On construit inductivement la famille des ensembles S-indépendants en suivant les trois règles suivantes :

1. L'ensemble vide $\varnothing$ est S-indépendant.
2. L'ensemble $A \cup \{b\}$ est S-indépendant si A est S-indépendant, $A \subset S$ et $b \in \mathbb{F} \setminus S$.
3. L'ensemble $\{ca : a \in A\}$ est S-indépendant si A est S-indépendant et $c \in \mathbb{F}^*$.

Théorème 10 (Méthode du décalage)**.** *La distance minimale de C est supérieure au cardinal de tout ensemble S-indépendant où S désigne l'ensemble des zéros de n'importe quel mot non nul de C.*

Remarque 28. Il existe naturellement d'autres bornes inférieures sur la distance minimale des codes cycliques parmi lesquelles [Jen85, Wol89, Bos01, BS06]. En particulier, on citera la borne de Jensen qui peut se révéler meilleure que les bornes obtenues par les méthodes de van Lint-Wilson dans certaines situations [RvL91] bien que ce ne soit pas le cas en général.

3.2 Borne algorithmique

Bien que les distributions de poids du code C et de son dual $C^\perp$ soient directement reliées par les identités de MacWilliams et Pless [HP03b, Chapitre 7], il n'y a pas de résultat théorique connu qui combine la distance minimale de C et $C^\perp$.

Dans cette section nous discutons sur la distance minimale du dual des codes cycliques 3-correcteurs avec l'ensemble de définition $\{1, 2^i+1, 2^j+1\}$ pour $i, j > 1$ et $i \neq j$. Notez que ces duaux sont également cycliques. Si 0 est dans l'ensemble de définition de C, alors C est un code de poids pair. Cela implique que le dual du code cyclique avec l'ensemble de définition $\{1, 2^i + 1, 2^j + 1\}$, a une distance minimale paire.

Des bornes inférieures sur la distance minimale des codes cycliques sont connues (*e.g.* [BRC60, HT72, Roo83, vLW86, Wol89]). Elles reposent soit sur des propriétés de distribution régulière de certains motifs contenus dans l'ensemble de définition, ou sur le nombre de points rationnels de courbes algébriques sur les corps finis. Schaub [MS86] a pris une approche algorithmique pour calculer une borne inférieure sur la distance minimale des codes cycliques. Cette idée est particulièrement efficace pour les codes qui ont peu de sous-codes cycliques.

Pour trouver une borne inférieure de la distance minimale d'un code dual donné, nous appliquons l'algorithme de Schaub [Sch88]. Nous proposons une amélioration avec un critère d'élagage basé sur la borne BCH. Nous comparons alors la finesse de la borne de Schaub et celle de la borne de Hartmann-Tzeng.

3.2.1 L'algorithme de Schaub (1988)

Définition 53 (Complexité linéaire)**.** La complexité linéaire d'une suite est la longueur minimale d'un registre à décalage à rétroaction linéaire (LFSR en abrégé) qui engendre cette suite.

Dans [Sch88], Schaub a introduit un algorithme qui calcule une borne inférieure de la distance minimale d'un code cyclique. Cet algorithme applique itérativement une méthode appelée Rank Bound sur des matrices symboliques. Fondamentalement, cette méthode calcule la complexité linéaire de la suite périodique infinie obtenue à partir de la transformation de Fourier discrète d'un mot c de longueur n sur $\mathbb{F}_{q^m}$, où q est une puissance première. Dans notre cas, $q = 2$ et $n = 2^m - 1$. La complexité en temps est $\mathcal{O}(n^3) = \mathcal{O}(2^{3m})$. Le théorème de Blahut assure que cette quantité est égale au poids de Hamming de c (*e.g.* [Mas98]). En termes de matrices, cela signifie que le poids de c est égal au rang de la matrice circulante $\mathcal{B}_c$ d'ordre n,

$$\mathcal{B}_c = \begin{pmatrix} A_0 & A_1 & \dots & A_{n-2} & A_{n-1} \\ A_1 & A_2 & \dots & A_{n-1} & A_0 \\ \vdots & \vdots & & \vdots & \vdots \\ A_{n-1} & A_0 & \dots & A_{n-3} & A_{n-2} \end{pmatrix},$$

où $(A_i)_{0 \leq i \leq n-1}$ est la famille des coefficients du polynôme de Mattson-Solomon de c. Considérons un code cyclique C de longueur n sur $\mathbb{F}_2$ avec l'ensemble de définition Z. D'une part, la distance minimale d de C est égale au rang minimal de $\mathcal{B}_c$, pour tout $c \in C$. Cependant, il n'est pas pratique d'utiliser l'algorithme de Berlekamp-Massey, dont la complexité en temps est $\mathcal{O}(n^2)$, pour calculer la distance minimale d'un code cyclique. D'autre part, puisque C est cyclique, les coefficients de $\mathcal{B}_c$ satisfont, pour tout $c \in C$, la propriété : $A_z = 0$ pour tout $z \in Z$. De plus, pour tout $c \in C$, l'ensemble des entiers i tel que $A_i = 0$ forme une union de classes q-cyclotomiques modulo n. L'algorithme de Schaub calcule une borne inférieure sur le rang de matrices symboliques que nous décrivons ci-dessous.

Schaub [Sch88] a défini une arithmétique avec trois symboles. On les note 0, 1 et X. Avec cette notation, 0 désigne l'élément nul de $\mathbb{F}_{2^m}$, 1 désigne n'importe quel élément non nul de $\mathbb{F}_{2^m}$ et X désigne n'importe quel élément de $\mathbb{F}_{2^m}$ dont la nullité ou la non-nullité n'est pas connue.

Le semi-anneau commutatif $(\{0, 1, X\}, +, *)$ est défini par les tables :

$+$	0	1	X
0	0	1	X
1	1	X	X
X	X	X	X

$*$	0	1	X
0	0	0	0
1	0	1	X
X	0	X	X

Si κ classes q-cyclotomiques n'appartiennent pas à Z alors l'algorithme de Schaub calcule en réalité une borne inférieure sur le rang de 2^κ matrices circulantes dans $M(\{0, 1\})$. Ces matrices ont des coefficients nuls seulement dans les positions determinées par les 2^κ unions de classes q-cyclotomiques correspondantes. Ainsi, l'algorithme de Schaub a une complexité en temps $\mathcal{O}(2^{3m+\kappa})$.

Chaque matrice peut être identifiée avec un sous-code non-linéaire de C. Ce code est défini par le mot de code de C ayant des zéros seulement sous la forme α^i, où i appartient à l'ensemble des positions des coefficients zéros dans la première ligne de la matrice. Chacun de ces codes est de la forme $\mathcal{D} \setminus \mathcal{E}$, où $\mathcal{D}$ est un sous-code cyclique de C et $\mathcal{E}$ est l'union de

Algorithme 1 Rank Bound

Entrée : une matrice non nulle M de taille $n \times n$, avec des coefficients sur $\{0, 1\}$.
Sortie : une borne inférieure sur le rang de M.

▶ *Étape d'initialisation*
{On veut construire un ensemble de lignes indépendantes de cardinalité maximale.
La première ligne de M est considérée indépendante.}
ligne_ind$[1] \leftarrow M[1]$;
{borne_rang est le nombre de lignes indépendantes.}
borne_rang $\leftarrow 1$;
▶ *Recherche de lignes nécessairement indépendantes dans* M
pour $1 \leq j \leq n$ **faire**
  pour $1 \leq i \leq$ borne_rang **faire**
    {Nous supposons que la ligne courante $M[j]$ est une combinaison linéaire de lignes prouvées indépendantes de M.}
    coeff$[i] \leftarrow X$
  fin pour
  {Nous essayons d'obtenir une contradiction sur la dépendance de $M[j]$ avec les lignes prouvés indépendantes.}
  répéter
    $k \leftarrow 1$; changement $\leftarrow$ faux ; indépendant $\leftarrow$ faux ;
    tant que $k \leq n$ **et** indépendant=faux **faire**
      ▶ *Construction du tableau des termes*
      pour $1 \leq i \leq$ borne_rang **faire**
        terme$[i] \leftarrow$ coeff$[i] *$ ligne_ind$[i][j]$;
      fin pour
      {Le coefficient $M[j][k]$ est la somme des termes.}
      somme $\leftarrow M[j][k]$
      si somme$= 0$ **alors**
        {Les cinq cas sont décrits dans la Figure 3.1.}
        cas 1 : coeff$[a] \leftarrow 0$; changement $\leftarrow$ vrai ;
        cas 2 : indépendant $\leftarrow$ vrai ;
        cas 3 : coeff$[b] \leftarrow 1$; changement $\leftarrow$ vrai ;
      sinon
        cas 4 : indépendant $\leftarrow$ vrai ;
        cas 5 : coeff$[a] \leftarrow 1$; changement $\leftarrow$ vrai ;
      fin si
      $k \leftarrow k + 1$;
    fin tant que
  jusqu'à changement=faux **ou** indépendant=vrai
  si indépendant=vrai **alors**
    borne_rang $\leftarrow$ borne_rang+1 ;
    ligne_ind[borne_rang] $\leftarrow M[j]$;
  fin si
fin pour
retourner borne_rang ;

Cas	Somme	Termes [1]	Conclusion
1	0	$0 \ldots 0 \underset{a}{X} 0 \ldots 0$	coeff$[a] = 0$
2	0	$0 \ldots 0 \underset{a}{1} 0 \ldots 0$	indépendant
3	0	$0 \ldots 0 \underset{a}{X} 0 \ldots 0 \underset{b}{1} 0 \ldots 0$	coeff$[a] = 1$
4	1	$0 \ldots 0$	indépendant
5	1	$0 \ldots 0 \underset{a}{X} 0 \ldots 0$	coeff$[a] = 1$

FIGURE 3.1 – Description des cinq cas qui permettent de déterminer soit des coefficients inconnus, soit l'indépendance de la ligne considérée, dans la méthode Rank Bound.

tous les sous-codes cycliques stricts de $\mathcal{D}$. Appelons ces sous-codes non-linéaires les *codes à zéros constants* de C, puisque leurs mots de code (en tant que polynômes) ont tous les même zéros. Les codes à zéros constants forment une partition de C. De plus, nous pouvons associer à chaque sous-code à zéros constants de C, le sous-code cyclique de C avec l'ensemble de définition correspondant. La méthode du Rank Bound consiste à construire un ensemble de lignes nécessairement indépendantes de la matrice et de retourner la cardinalité de cet ensemble. Cette cardinalité est une borne inférieure sur la distance minimale d'un code à zéros constants de C. Ainsi, la borne de Schaub est la cardinalité minimum calculée parmi tous les sous-codes considérés.

3.2.2 Optimisations

Notez que chaque matrice circulante d'ordre n sur $\{0, 1\}$ peut-être identifiée par l'entier entre 0 et $2^n - 1$ dont la représentation binaire est la première ligne de la matrice. Nous supposons que les entiers sont distincts de $2^n - 1$. Si deux entiers sont dans la même classe 2-cyclotomique modulo $(2^n - 1)$, alors les matrices correspondantes sont équivalentes et ont donc le même rang. Ainsi, il est seulement nécessaire de considérer un représentant de chaque classe 2-cyclotomique modulo $(2^n - 1)$. De plus, si la classe 2-cyclotomique modulo $(2^n - 1)$ de l'entier contient p éléments, alors nous pouvons considérer seulement la sous-matrice contenant les p premières lignes de la matrice au lieu de la matrice circulante complète. En effet, la matrice circulante est une matrice par blocs et peut être séparée par des lignes horizontales en n/p blocs où chaque bloc est la sous-matrice décrite ci-dessus. Notez que p divise n.

Une structure de données s'avère naturelle pour représenter l'ensemble des sous-codes considérés, il s'agit de la structure d'arbre. Un nœud A de l'arbre correspond à un code à zéros constants de C ou de manière équivalente au sous-code cyclique de C avec l'ensemble de définition noté Z_A. Pour notre but, le nœud racine correspond au sous-code de C qui contient les mots de code avec des zéros exactement dans les positions donnés par Z. Un nœud C est l'enfant d'un nœud parent P si et seulement si $Z_C \supset Z_P$ et $|Z_C| = |Z_P| + 1$. Le nombre de sous-codes étudiés croît exponentiellement en le nombre de classes cyclotomiques qui ne sont pas dans Z. Afin de réduire la complexité en temps, notre stratégie est d'élaguer l'arbre en utilisant la borne BCH qui est facilement calculable. On peut penser à introduire la borne de Hartmann-Tzeng pour élaguer davantage. Cependant, notre analyse empirique montre

1. Les indices a et b sont des entiers qui indiquent la position de l'élément indexé dans le tableau des termes.

que la borne de Hartmann-Tzeng ralentit le processus et ne donne pas de meilleurs résultats que la borne BCH. Si, dans un nœud, la borne de Schaub est plus petite que la borne BCH du sous-code cyclique associé de C, il devient inutile d'appliquer la méthode Rank Bound au sous-arbre dont la racine est le nœud considéré. En effet, dans chaque nœud de l'arbre élagué, la borne BCH est toujours plus grande que la borne de Schaub et donc la borne de Schaub n'est pas mise à jour dans ce sous-arbre.

3.3 La distance minimale en chiffrement à flot

Notation 25. *Soit $m > 0$. Soit f une fonction booléenne (sous forme univariée) définie sur $\mathbb{F}_{2^m}$.*

Définition 54 (Immunité algébrique)**.** L'immunité algébrique de f est :

$$\min\left\{ \deg(g) : fg = 0 \text{ ou } (1+f)g = 0 \right\}$$

où g appartient à l'ensemble des fonctions booléennes non nulles définies sur $\mathbb{F}_{2^m}$.

L'expression « immunité algébrique » est apparue dans [MPC04]. La notion d'immunité spectrale a quant à elle été introduite dans [GRHH11]. On reprend la définition d'immunité algébrique donnée dans [HR11] qui généralise modérément la définition originale.

Définition 55 (Immunité spectrale)**.** Soient α un élément primitif du groupe multiplicatif $\mathbb{F}_{2^m}^{\times}$ et $\beta \in \mathbb{F}_{2^m}^{\times}$. Pour tout $t \in \mathbb{N}$, on note $z_t \coloneqq f(\alpha^t \beta)$.

L'immunité spectrale de la suite binaire $(z_t)_{t\in\mathbb{N}}$ est la complexité linéaire minimale des suites 2^m-aires $(u_t)_{t\in\mathbb{N}}$ qui vérifient l'une des équations suivantes :

$$z_t u_t = 0, \text{ pour tout } t \in \mathbb{N},$$

$$(1+z_t)u_t = 0, \text{ pour tout } t \in \mathbb{N}.$$

3.3.1 Les codes cycliques et les fonctions booléennes

Les fonctions booléennes sont d'importants blocs de construction dans la conception de chiffrement à flot. Une haute immunité algébrique [CM03] est une condition nécessaire pour protéger le chiffrement à flot des attaques algébriques. *L'immunité spectrale* est un concept lié à l'immunité algébrique [HR11]. Si l'immunité spectrale d'une fonction booléenne est faible, alors on peut trouver l'état initial d'un générateur filtré dans lequel la fonction booléenne est utilisée. Dans [HR11] le lien entre l'immunité spectrale et la distance minimale d'un code cyclique est démontré. L'immunité spectrale d'une fonction booléenne f sur $\mathbb{F}_{2^m}$ (en forme univariée) est égale au poids minimal dans les codes cycliques 2^m-aires de longueur $n = 2^m - 1$ générés par les polynômes $\operatorname{pgcd}(f(z), z^n + 1)$ et $\operatorname{pgcd}(f(z) + 1, z^n + 1)$. Nous avons donc besoin d'un algorithme pour estimer efficacement la distance minimale d'un code cyclique sur $\mathbb{F}_{2^m}$.

3.3.2 Minorer l'immunité spectrale de fonctions booléennes

Nous appliquons notre version améliorée de l'algorithme de Schaub sur les codes cycliques de longueur $n = 2^m - 1$ sur $\mathbb{F}_{2^m}$ avec un ensemble de définition obtenu à partir des codes cycliques 3-correcteurs avec l'ensemble de définition $\{1, 2^i + 1, 2^j + 1\}$.

Nous donnons un résultat de van Lint [vLW86, Théorème 9].

Théorème 11. *Un code cyclique sur $\mathbb{F}_{2^m}$ possède la même distance minimale que son sous-code sur le sous-corps $\mathbb{F}_2$ si son ensemble de définition est une union de classe 2-cyclotomiques.*

Nous définissons aussi la *fonction Trace* comme étant $\mathrm{Tr}(\cdot) : \mathbb{F}_{2^m} \to \mathbb{F}_2$,

$$\mathrm{Tr}(z) \coloneqq z + z^2 + z^{2^2} + \ldots + z^{2^{m-1}}.$$

Dans la Table 3.1, nous donnons une borne inférieure sur l'immunité spectrale des fonctions booléennes $\mathrm{Tr}(g(.))$ en forme univariée sur $\mathbb{F}_{2^m}$, où g parcourt l'ensemble des polynômes générateurs des codes cycliques 3-correcteurs avec l'ensemble de définition $\{1, 2^i + 1, 2^j + 1\}$ pour $m < 9$. Nous notons G et H, les polynômes générateurs des codes cycliques sur $\mathbb{F}_{2^m}$ sur lesquels nous appliquons l'algorithme de Schaub,

$$G(z) \coloneqq \mathrm{pgcd}(\mathrm{Tr}(g(z)), z^n + 1),$$

$$H(z) \coloneqq \frac{z^n + 1}{G(z)}.$$

Notez que l'ensemble de définition de G contient l'ensemble de définition de g puisque $g(z)$ divise à la fois $\mathrm{Tr}(g(z))$ et $z^n + 1$. De plus, $H(z) = \mathrm{pgcd}(\mathrm{Tr}(g(z)) + 1, z^n + 1)$, puisque $\mathrm{Tr}(g(.))$ est une fonction booléenne.

Le polynôme g a des coefficients sur $\mathbb{F}_2$ puisqu'il est le produit de polynômes minimaux par rapport à $\mathbb{F}_2$. Par conséquent, la fonction booléenne $\mathrm{Tr}(g(.))$ et les polynômes générateurs G et H ont des coefficients binaires. À partir du Théorème 11, on déduit que la distance minimale de ces codes sur $\mathbb{F}_{2^m}$ est la même que celle de leurs sous-codes sur le sous-corps $\mathbb{F}_2$. En conséquence, nous appliquons l'algorithme de Schaub sur leurs sous-codes sur le sous-corps binaire, puisque ces derniers ont beaucoup moins de sous-codes cycliques.

Dans la Table 3.2, nous donnons les bornes d'Hartmann-Tzeng et de Schaub de tous les codes duaux des codes cycliques 3-correcteurs de longueur $2^m - 1$ avec $5 \leq m \leq 13$ et $Z = \{1, 2^i + 1, 2^j + 1\}$. Certaines sous-classes des codes de cette forme sont bien connues. Nous observons que la borne de Schaub avec élagage est plus fine que la borne de Hartmann-Tzeng sur cette classe de codes. La dernière colonne de la Table 3.2 indique la vraie distance minimale de ces codes. Cette quantité est calculable puisque ces codes possèdent une dimension suffisamment petite. On constate que la distance minimale de ces codes est égale pour une longueur de code donnée. On verra au chapitre suivant que ces codes possèdent en fait le même énumérateur de poids que le code BCH 3-correcteur.

Remarque 29. Nous considérons aussi le dual des codes cycliques 3-correcteurs définis avec le même ensemble de définition $\{1, 2^i + 1, 2^j + 1\}$ sur l'alphabet $\mathbb{F}_{2^m}$ pour $m < 9$. Il est intéressant de noter que nous obtenons la même borne pour les codes sur $\mathbb{F}_{2^m}$ et leur sous-codes

Longueur du Code	Ensemble de Définition	Borne Inférieure sur l'Immunité Spectrale
31	$\{1,3,5\}$	2
63	$\{1,3,5\}$	8
127	$\{1,3,5\}$	11
	$\{1,3,9\}$	13
	$\{1,5,9\}$	12
255	$\{1,3,5\}$	14
	$\{1,5,9\}$	14

TABLE 3.1 – Borne inférieure de l'immunité spectrale de toutes les fonctions booléennes $\mathrm{Tr}(g(.))$ où g est le polynôme générateur d'un code cyclique 3-correcteur avec l'ensemble de définition $\{1, 2^i + 1, 2^j + 1\}$

sur le sous-corps $\mathbb{F}_2$ avec l'algorithme de Schaub. De plus, les classes 2^m-cyclotomiques modulo n sont des singletons (on rappelle que $n = 2^m - 1$). De ce fait, le nombre de sous-codes cycliques sur $\mathbb{F}_{2^m}$ est beaucoup plus grand que dans le cas binaire. Par conséquent, l'optimisation proposée dans la Sous-section 3.2.2, qui permet de ne pas borner inférieurement le rang de matrices équivalentes, se révèle particulièrement pertinente dans cette situation.

Longueur du Code	Ensemble de Définition du Dual	Borne de Hartmann-Tzeng	Borne de Schaub avec élagage	Vraie Distance Minimale
31	$\{1,3,5\}$	8	8	8
63	$\{1,3,5\}$	16	16	16
127	$\{1,3,5\}$	32	48	48
	$\{1,3,9\}$	32	48	48
	$\{1,5,9\}$	48	48	48
255	$\{1,3,5\}$	64	96	96
	$\{1,5,9\}$	96	96	96
511	$\{1,3,5\}$	128	216	224
	$\{1,3,9\}$	128	212	224
	$\{1,3,17\}$	128	210	224
	$\{1,5,9\}$	192	218	224
	$\{1,5,17\}$	192	212	224
	$\{1,9,17\}$	224	224	224
1023	$\{1,3,5\}$	256	446	448
	$\{1,9,17\}$	448	448	448
2047	$\{1,3,5\}$	512	930	960
	$\{1,3,9\}$	512	906	960
	$\{1,3,17\}$	512	906	960
	$\{1,3,33\}$	512	876	960
	$\{1,5,9\}$	768	936	960
	$\{1,5,17\}$	768	872	960
	$\{1,5,33\}$	768	916	960
	$\{1,9,17\}$	896	902	960
	$\{1,9,33\}$	896	902	960
	$\{1,17,33\}$	960	960	960
4095	$\{1,3,5\}$	1024	1886	1920
	$\{1,5,33\}$	1536	1814	1920
8191	$\{1,3,5\}$	2048	3110	3968
	$\{1,3,9\}$	2048	3588	3968
	$\{1,3,17\}$	2048	3643	3968
	$\{1,3,65\}$	2048	3668	3968
	$\{1,5,17\}$	3072	3594	3968
	$\{1,5,33\}$	3072	3678	3968
	$\{1,5,65\}$	3072	3802	3968
	$\{1,9,17\}$	3584	3912	3968
	$\{1,9,33\}$	3584	3718	3968
	$\{1,9,65\}$	3584	3722	3968
	$\{1,17,33\}$	3840	3844	3968
	$\{1,33,65\}$	3968	3968	3968

TABLE 3.2 – Finesse de bornes sur la distance minimale du dual du code cyclique 3-correcteur sur $\mathbb{F}_2$ de longueur 2^m-1 avec l'ensemble de définition $\{1, 2^i+1, 2^j+1\}$

4 La classe de codes cycliques $\{1, 2^i + 1, 2^j + 1\}$

4.1 Caractériser les codes cycliques 3-correcteurs

Soit $Z = \{a, b, c\}$ l'ensemble de définition d'un code cyclique C, où a, b et c sont les représentants de 2-classes cyclotomiques deux à deux distinctes. Alors, la matrice de parité de C est une matrice de taille $(3m \times n)$, sur $\mathbb{F}_2$, de la forme :

$$\mathcal{H} = \begin{pmatrix} 1 & \alpha^a & \alpha^{2a} & \dots & \alpha^{(n-1)a} \\ 1 & \alpha^b & \alpha^{2b} & \dots & \alpha^{(n-1)b} \\ 1 & \alpha^c & \alpha^{2c} & \dots & \alpha^{(n-1)c} \end{pmatrix},$$

où chaque élément dans la matrice est representé comme un vecteur colonne de m-bits par rapport à une $\mathbb{F}_2$-base fixée de $\mathbb{F}_{2^m}$. Alors, le code associé à Z est le noyau binaire de $\mathcal{H}$.

Dans la Table 4.1, nous présentons la liste connue des ensembles de définition qui correspondent à des codes cycliques 3-correcteurs de longueur $2^m - 1$.

4.1.1 Résultats théoriques

Les ensembles de définition dans [Kas71] et [BH09b] sont de la forme $\{1, 2^\ell + 1, 2^{p\ell} + 1\}$, où $p = 3$ et $p = 2$ respectivement. Par conséquent, il serait intéressant de caractériser pour quelle valeur de p l'ensemble de définition $\{1, 2^\ell + 1, 2^{p\ell} + 1\}$ donne toujours un code cyclique 3-correcteur.

Dans cette section, nous présentons une condition suffisante pour que l'ensemble de définition $Z = \{1, 2^\ell + 1, 2^{p\ell} + 1\}$ avec $\mathrm{pgcd}(\ell, m) = 1$ corresponde à un code cyclique 3-correcteur.

Notation 26. *Soit* $f : \mathbb{F}_{2^m} \to \mathbb{F}_{2^m}$.

Définition 56 (fonction APN)**.** La fonction f est *presque parfaitement non-linéaire (*APN en abrégé) si pour tout $a \in \mathbb{F}_{2^m}^\times$ et tout $b \in \mathbb{F}_{2^m}$, l'équation

$$f(x + a) + f(x) = b \tag{4.1}$$

possède zéro ou deux solutions $x \in \mathbb{F}_{2^m}$.

Autrement dit, la fonction f est APN si l'on a, pour tout $a \in \mathbb{F}_{2^m}^\times$, l'égalité :

$$|\{f(x) + f(x + a), x \in \mathbb{F}_{2^m}\}| = 2^{m-1}.$$

Ensemble de définition	Conditions	Références
$\{1, 2^\ell + 1, 2^{3\ell} + 1\}$	$\gcd(\ell, m) = 1$ m impair	[Kas71]
$\{2^\ell + 1, 2^{3\ell} + 1, 2^{5\ell} + 1\}$	$\gcd(\ell, m) = 1$ m impair	[Kas71]
$\{1, 2^{\ell-1} + 1, 2^\ell + 1\}$	$m = 2\ell + 1$ m impair	[MS83]
$\{1, 2^{\ell+1} + 1, 2^{\ell+2} + 3\}$	$m = 2\ell + 1$ m impair	[CGGea00]
$\{1, 2^\ell + 1, 2^{2\ell} + 1\}$	$\gcd(\ell, m) = 1$ m quelconque	[BH09b]
$\{1, 3, 13\}$	m impair	[ZSH10]

TABLE 4.1 – Classes connues de codes cycliques 3-correcteurs de longueur $2^m - 1$

Nous donnons dans la Table 4.2, la liste des exposants monomiaux APN connus à ce jour. Nous donnons également le résultat [Cha98, Théorème 4.2] qui donne une caractérisation de tous les codes binaires primitifs de distance minimale 5.

Théorème 12. *Soient α une racine primitive n-ième de l'unité et C le code linéaire binaire défini par une matrice de parité $\mathcal{H}$ de la forme :*

$$\mathcal{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{(n-1)} \\ f(1) & f(\alpha) & f(\alpha^2) & \dots & f(\alpha^{(n-1)}) \end{pmatrix},$$

où f est une fonction sur $\mathbb{F}_{2^m}$ qui s'annule en 0.

Le code C possède une distance minimale 5 si et seulement si la fonction f est APN.

Démonstration. Soit $c = (c_1, \dots, c_n)$ un mot de $\mathbb{F}_2^n$.

On a $c \in C$ si et si seulement si :

$$\begin{cases} \sum_{i=1}^{n} c_i \alpha^i = 0, \\ \sum_{i=1}^{n} c_i f(\alpha^i) = 0. \end{cases} \tag{4.2}$$

Le poids minimal de C est supérieur à 3 puisque $\alpha^i \neq \alpha^j$ pour $i \neq j$.

L'équation 4.1 peut se réécrire sous la forme :

$$\begin{cases} x + y & = a, \\ f(x) + f(y) & = b. \end{cases} \tag{4.3}$$

pour tout $a \in \mathbb{F}_{2^m}^{\times}$ et tout $b \in \mathbb{F}_{2^m}$.

On raisonne par l'absurde. On suppose que la distance minimale est 5 et que f n'est pas APN. Soient quatre éléments de $\mathbb{F}_{2^m}$, distincts deux à deux. On les note x, y, x', y'. On les choisit de manière à ce que les couples (x, y) et (x', y') vérifient l'équation 4.3 pour un couple (α, β). On en déduit alors que :

$$\begin{cases} x + y + x' + y' & = 0, \\ f(x) + f(y) + f(x') + f(y') & = 0, \end{cases} \tag{4.4}$$

D'après l'équation 4.2, cela signifie que C contient un mot de poids 3 ou 4. On a une contradiction. De plus, on sait par [Cha98, Théorème 3.25] que la distance minimale de C est ≤ 5. Ce dernier résultat repose sur la non-existence de code linéaire $[2^m - 1, k, 6]$ tel que $k \geq 2^m - 1 - 2m$. On obtient ainsi le résultat annoncé. □

Remarque 30. Ce théorème implique qu'un code cyclique avec un ensemble de définition $Z = \{1, a\}$, où a est l'exposant d'un monôme APN, possède une distance minimale 5.

	Exposants	Conditions	Références
Gold	$2^a + 1$	$\gcd(a, m) = 1$	[Gol68]
Kasami	$2^{2a} - 2^a + 1$		[Kas71]
Niho	$2^a + 2^{\frac{a}{2}} - 1$, a pair $2^a + 2^{\frac{3a+1}{2}} - 1$, a impair	$m = 2a + 1$	[Nih72]
Inverse	$2^{2a} - 1$		[Nyb94]
Welch	$2^a + 3$		[Dob99]
Dobbertin	$2^{4a} + 2^{3a} + 2^{2a} + 2^a - 1$	$m = 5a$	[Dob01]

TABLE 4.2 – Liste des exposants monomiaux APN connus sur $\mathbb{F}_{2^m}$.

Définition 57 (Entrelacement)**.** Soit C un code de longueur n.

Un entrelacement τ est une application telle que :

$$\tau : C \mapsto \mathbb{F}_q^n \quad (c_1, \ldots, c_n) \to (c_{\sigma(1)}, \ldots, c_{\sigma(n)})$$

où σ désigne une permutation du groupe symétrique S_n.

Par commodité, on étend cette définition aux codes et on notera $\tau(C) \coloneqq \{\tau(c) : c \in C\}$.

Définition 58 (Codes équivalents)**.** Deux codes linéaires binaires sont équivalents s'ils possèdent la même orbite sous l'action de leur groupe d'entrelacements. Autrement dit, cela signifie qu'un code se déduit de l'autre par permutation des coordonnées.

Définition 59 (Groupe d'automorphismes)**.** Le groupe d'automorphismes d'un code linéaire binaire est le stabilisateur de ce code sous l'action de son groupe d'entrelacements. Il s'agit de l'ensemble des permutations des coordonnées qui laissent le code invariant.

Remarque 31. Ces définitions se limitent au contexte des codes linéaires binaires. Le lecteur interessé pourra consulter [HP03b, pp. 23-28] et [Sen02, Chapitre 1] pour étudier différentes notions d'équivalence de codes et de groupe d'automorphismes de codes dans un cadre plus large.

Proposition 6. *Soit un code linéaire binaire C de longueur n avec un groupe d'automorphismes transitif. Les n codes obtenus en poinçonnant C une fois en chaque coordonnée sont équivalents.*

Nous allons à présent donner un théorème connu dont nous nous servons par la suite.

Théorème 13. *Soit un code linéaire binaire C de distance minimale d. Si le groupe d'automorphismes de son code étendu $\overline{C}$ est transitif, alors d est impaire et C contient des mots de poids $d + 1$.*

Démonstration. On obtient des codes équivalents à C en poinçonnant $\overline{C}$ en n'importe quelle composante du fait que le groupe d'automorphismes de C est transitif. On note $\mathcal{P}$ cette assertion. Supposons que C ait une distance minimale paire d et prenons c un mot de C de poids d. Le mot étendu $\bar{c}$ est de poids d puisque $\overline{C}$ est un code de poids pair. Si l'on poinçonne $\overline{C}$ en une composante non nulle de $\bar{c}$, on obtient un code de distance minimale $d - 1$. On a une contradiction avec $\mathcal{P}$ puisque ce code n'est pas équivalent à C. Le code C possède donc une distance minimale impaire. Soient (a_i) et (A_i) les distributions de poids de C et de $\overline{C}$. Supposons que $\overline{C}$ soit de longueur N. Considérons la matrice dont les lignes sont les A_{2i} mots de $\overline{C}$. Cette matrice contient $2iA_{2i}$ éléments valant 1. On déduit de $\mathcal{P}$ que chaque colonne de la matrice contient un nombre identique de 1. On a en fait l'égalité :

$$a_{2i-1} = \frac{2iA_{2i}}{N}$$

On sait par ailleurs que $A_{2i} = a_{2i} + a_{2i-1}$, on obtient donc :

$$a_{2i} = \frac{(N - 2i)a_{2i-1}}{2i}$$

En prenant $i = d$, on montre que C possède des mots de poids $d + 1$. □

Lemme 2. *Soit d la distance minimale du code cyclique C donné par l'ensemble de définition $Z = \{1, 2^i + 1, 2^j + 1\}$ avec $\mathrm{pgcd}(i, m) = 1$ de longueur $n = 2^m - 1$. Alors $d = 5$ ou $d = 7$ et il existe des mots de code de poids $d + 1$.*

Démonstration. On sait à partir du théorème 12 que les codes cycliques avec $Z = \{1, a\}$ ont une distance minimale 5 si et seulement si l'application $x \mapsto x^a$ est APN. En conséquence, si $Z = \{1, a, b\}$ et si $x \mapsto x^a$ est APN, alors le code possède une distance minimale au moins 5 puisque c'est un sous-code du code avec l'ensemble de définition $\{1, 2^i + 1\}$. L'application $x \mapsto x^{2^i+1}$ est APN (*e.g.* [Dob01]) si et seulement si $\mathrm{pgcd}(i, m) = 1$. Par conséquent, la distance minimale de C est au moins 5. De plus, le code C contient le code Reed-Muller cyclique $\mathcal{R}^*(m-$

$3, m)$ (cf. Théorème 2). En effet, $\mathcal{R}^*(m-3, m)$ a un ensemble de définition $\cup_{i\in\mathbb{N}}\{2^i+1\}$. $\mathcal{R}^*(m-3, m)$ possède une distance minimale $2^{m-(m-3)} = 7$. En conséquence, nous sommes assurés que $5 \leq d \leq 7$. On peut prouver que le groupe d'automorphismes de C contient le groupe affine à partir d'un résultat de Kasami, Lin et Peterson [MS83, pp 236-237]. Il est donc transitif puisqu'on sait que le groupe affine est doublement transitif. Le groupe affine a été défini dans le Chapitre 2. Le Théorème 13 nous permet donc de conclure que $d = 5$ ou $d = 7$ et que C contient des mots de code de poids $d+1$. □

Théorème 14. *Considérons C, le code cyclique de longueur 2^m-1 avec l'ensemble de définition $Z = \{1, 2^\ell+1, 2^{p\ell}+1\}$ où $\mathrm{pgcd}(\ell, m) = 1$. Alors la distance minimale de C est 7 si pour tout $\beta \in \mathbb{F}_{2^m}^\times, \gamma \in \mathbb{F}_{2^m}$, l'équation*

$$x^{2^{p\ell}+1}\sum_{i=0}^{p-1}\left(\beta x^{-(2^\ell+1)}\right)^{2^{i\ell}} = \gamma \tag{4.5}$$

a au plus 5 solutions en x dans $\mathbb{F}_{2^m}^\times$.

Démonstration. La matrice de parité de C est de la forme :

$$\mathcal{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{(n-1)} \\ 1 & \alpha^{2^\ell+1} & \alpha^{2(2^\ell+1)} & \cdots & \alpha^{(n-1)(2^\ell+1)} \\ 1 & \alpha^{2^{p\ell}+1} & \alpha^{2(2^{p\ell}+1)} & \cdots & \alpha^{(n-1)(2^{p\ell}+1)} \end{pmatrix}.$$

Supposons, par l'absurde, que la distance minimale soit 5. À partir du Lemme 2, nous pouvons considérer un mot de code de poids six. Ainsi, il existe un ensemble de six colonnes dépendantes sur $\mathbb{F}_2$ dans $\mathcal{H}$. En d'autres termes, il existe six éléments distincts x, y, z, u, v, w dans $\mathbb{F}_{2^m}^\times$ tels que :

$$\begin{cases} x+y+z+u+v+w & = 0, \\ x^{2^\ell+1}+y^{2^\ell+1}+\ldots+v^{2^\ell+1}+w^{2^\ell+1} & = 0, \\ x^{2^{p\ell}+1}+y^{2^{p\ell}+1}+\ldots+v^{2^{p\ell}+1}+w^{2^{p\ell}+1} & = 0. \end{cases} \tag{4.6}$$

Ces équations sont symétriques en les variables x, y, z, u, v, w. Ainsi, le système (4.6) peut s'écrire comme :

$$\begin{cases} x+y+z & = u+v+w & = a, \\ x^{2^\ell+1}+y^{2^\ell+1}+z^{2^\ell+1} & = u^{2^\ell+1}+v^{2^\ell+1}+w^{2^\ell+1} & = b, \\ x^{2^{p\ell}+1}+y^{2^{p\ell}+1}+z^{2^{p\ell}+1} & = u^{2^{p\ell}+1}+v^{2^{p\ell}+1}+w^{2^{p\ell}+1} & = c, \end{cases}$$

où $a, b, c \in \mathbb{F}_{2^m}$. Notons que $b \neq a^{2^\ell+1}$, autrement nous aurions :

$$\begin{aligned} x+y+z+a &= 0, \\ x^{2^\ell+1}+y^{2^\ell+1}+z^{2^\ell+1}+a^{2^\ell+1} &= 0, \end{aligned}$$

ce qui signifierait qu'il y a un mot de code de poids 4 dans le code avec l'ensemble de définition $\{1, 2^\ell+1\}$ où $\mathrm{pgcd}(\ell, m) = 1$. Cela est impossible puisque ce code possède une distance

minimale 5 d'après la preuve du Lemme 2. Maintenant, nous considérons les équations :

$$\begin{aligned} x + y + z &= a, \\ x^{2^\ell+1} + y^{2^\ell+1} + z^{2^\ell+1} &= b, \\ x^{2^{p\ell}+1} + y^{2^{p\ell}+1} + z^{2^{p\ell}+1} &= c. \end{aligned}$$

Nous remplaçons x par $x + a$, y par $y + a$ et z par $z + a$.

$$\begin{aligned} x + y + z &= 0, \\ (x + a)^{2^\ell+1} + (y + a)^{2^\ell+1} + (z + a)^{2^\ell+1} &= b, \\ (x + a)^{2^{p\ell}+1} + (y + a)^{2^{p\ell}+1} + (z + a)^{2^{p\ell}+1} &= c. \end{aligned}$$

Ensuite, en développant la seconde et la troisième équation et en utilisant la relation $x + y + z = 0$ nous obtenons les équations suivantes :

$$\begin{aligned} x + y + z &= 0, \\ x^{2^\ell+1} + y^{2^\ell+1} + z^{2^\ell+1} &= b + a^{2^\ell+1}, \\ x^{2^{p\ell}+1} + y^{2^{p\ell}+1} + z^{2^{p\ell}+1} &= c + a^{2^{p\ell}+1}. \end{aligned}$$

Puis, nous remplaçons $z = x + y$ et nous obtenons :

$$\begin{cases} x^{2^\ell} y + y^{2^\ell} x &= \beta, \\ x^{2^{p\ell}} y + y^{2^{p\ell}} x = \gamma, \end{cases} \tag{4.7}$$

où $\beta = b + a^{2^\ell+1}$ et $\gamma = c + a^{2^{p\ell}+1}$. Notons que $x \neq 0$, puisque $\beta \neq 0$. Nous remplaçons y par xy. (4.7) devient :

$$\begin{cases} x^{2^\ell+1}(y + y^{2^\ell}) = \beta, & (4.8) \\ x^{2^{p\ell}+1}(y + y^{2^{p\ell}}) = \gamma. & (4.9) \end{cases}$$

À partir de (4.8), nous obtenons :

$$y + y^{2^\ell} = \beta x^{-(2^\ell+1)},$$

et en élevant à la puissance 2^ℓ de manière répétée, nous obtenons :

$$\begin{aligned} y^{2^\ell} + y^{2^{2\ell}} &= (\beta x^{-(2^\ell+1)})^{2^\ell}, \\ y^{2^{2\ell}} + y^{2^{3\ell}} &= (\beta x^{-(2^\ell+1)})^{2^{2\ell}}, \\ \cdots &= \cdots, \\ y^{2^{(p-1)\ell}} + y^{2^{p\ell}} &= (\beta x^{-(2^\ell+1)})^{2^{(p-1)\ell}}. \end{aligned}$$

En les ajoutant tous, nous obtenons :

$$y + y^{2^{p\ell}} = \sum_{i=0}^{p-1} \left(\beta x^{-(2^\ell+1)}\right)^{2^{i\ell}}.$$

Enfin, nous obtenons à partir de (4.9) :

$$x^{2^{p\ell}+1} \sum_{i=0}^{p-1} \left(\beta x^{-(2^\ell+1)}\right)^{2^{i\ell}} = \gamma.$$

Si cette équation n'a pas plus de 5 solutions sur $\mathbb{F}_{2^m}^{\times}$, alors pour tous éléments distincts x, y, z, u, v, w dans $\mathbb{F}_{2^m}^{\times}$ (4.6) n'est pas satisfaite. En conséquence il n'y a pas de mot de poids 6. Le Lemme 2 implique donc que la distance minimale soit 7. □

Nous appliquons maintenant le Théorème 14 pour $p = 2$ et $p = 3$ pour montrer que $\{1, 2^\ell + 1, 2^{2\ell}+1\}$ [BH09b] et $\{1, 2^\ell+1, 2^{3\ell}+1\}$ [Kas71] sont deux ensembles de définition qui donnent un code cyclique 3-correcteur. le Théorème 14 donne la possibilité d'obtenir de nouvelles valeurs de p pour lesquelles $\{1, 2^\ell + 1, 2^{p\ell} + 1\}$ donne un code cyclique 3-correcteur en examinant l'équation 4.5 pour $p > 3$ (voir la Remarque 32).

Ci-dessous nous présentons une conséquence d'un résultat donné dans [Blu04].

Lemme 3. *L'équation de la forme* $x^{2^\ell+1} + rx^{2^\ell} + tx + s = 0$ *n'a pas plus de trois solutions pour tout* $r, s, t \in \mathbb{F}_{2^m}$ *lorsque* $\operatorname{pgcd}(\ell, m) = 1$.

Corollaire 1. *Pour* $p = 2$*, le code* $\{1, 2^\ell + 1, 2^{p\ell} + 1\}$ *avec* $\operatorname{pgcd}(\ell, m) = 1$ *est un code cyclique 3-correcteur.*

Démonstration. Pour $p = 2$, l'équation (4.5) devient :

$$\begin{aligned}\gamma &= x^{2^{2\ell}+1}\left(\beta x^{-(2^\ell+1)} + \beta^{2^\ell} x^{-2^\ell(2^\ell+1)}\right),\\ &= \beta x^{2^{2\ell}-2^\ell} + \beta^{2^\ell} x^{1-2^\ell},\\ &= \beta x^{2^\ell(2^\ell-1)} + \beta^{2^\ell} x^{-(2^\ell-1)}.\end{aligned}$$

Rappelons que $\beta \neq 0$. Puisque $\operatorname{pgcd}(\ell, m) = 1$, nous avons $\operatorname{pgcd}(2^\ell - 1, 2^m - 1) = 1$ et ainsi $x \mapsto x^{2^\ell-1}$ est une bijection. Puis, en faisant la transformation $x = x^{2^\ell-1}$ nous obtenons :

$$\begin{aligned}\beta x^{2^\ell} + \beta^{2^\ell} x^{-1} &= \gamma,\\ x^{2^\ell+1} + \frac{\gamma}{\beta} x + \beta^{2^\ell-1} &= 0.\end{aligned} \tag{4.10}$$

Le Lemme 3 nous dit que (4.10) n'a pas plus de trois solutions. Par conséquent, l'ensemble de définition $\{1, 2^\ell + 1, 2^{2\ell} + 1\}$ donne un code cyclique 3-correcteur. □

Corollaire 2. $\{1, 2^\ell + 1, 2^{p\ell} + 1\}$ *est un code cyclique 3-correcteur si* $p = 3$*,* m *est impair et* $\operatorname{pgcd}(\ell, m) = 1$.

Démonstration. Pour $p = 3$, l'équation (4.5) devient :

$$\begin{aligned}\gamma = \quad & x^{2^{3\ell}+1}\left(\beta x^{-\left(2^\ell+1\right)} + \beta^{2^\ell} x^{-2^\ell\left(2^\ell+1\right)} + \beta^{2^{2\ell}} x^{-2^{2\ell}\left(2^\ell+1\right)}\right),\\ = \quad & \beta x^{2^\ell\left(2^{2\ell}-1\right)} + \beta^{2^\ell} x^{\left(2^{2\ell}-1\right)\left(2^\ell-1\right)} + \beta^{2^{2\ell}} x^{1-2^{2\ell}}.\end{aligned}$$

Puisque m est impair et $\operatorname{pgcd}(\ell, m) = 1$, $\operatorname{pgcd}(2^{2\ell} - 1, 2^m - 1) = 1$ et donc $x \mapsto x^{2^{2\ell}-1}$ est une bijection.

Rappelons que $\beta \neq 0$. Maintenant remplaçons x par $x^{2^{2\ell}-1}$, nous obtenons :

$$\begin{aligned} \beta x^{2^\ell} + \beta^{2^\ell} x^{(2^\ell-1)} + \beta^{2^{2\ell}} x^{-1} &= \gamma, \\ x^{2^\ell+1} + \beta^{2^\ell-1} x^{2^\ell} + \frac{\gamma}{\beta} x + \beta^{2^{2\ell}-1} &= 0. \end{aligned} \tag{4.11}$$

Par le Lemme 3, nous obtenons que (4.11) ne peut pas avoir plus de trois solutions. Par conséquent, $\{1, 2^\ell + 1, 2^{3\ell} + 1\}$ est un code cyclique 3-correcteur. □

Remarque 32. Considérons m impair. Dans ce cas, $\text{pgcd}(2^\ell + 1, 2^m - 1) = 1$ et donc $x \mapsto x^{2^\ell+1}$ est une bijection. Si nous supposons que p est impair, en appliquant la transformation $x \mapsto x^{-(2^\ell+1)}$, (4.5) devient :

$$\sum_{i=0}^{p-1} (\beta x)^{2^{i\ell}} = \gamma x^{2^{(p-1)\ell}+2^{(p-2)\ell}+\ldots+1} \tag{4.12}$$

Il serait intéressant de trouver pour quelles valeurs de p, cette équation n'a pas plus de 5 solutions non-nulles dans $\mathbb{F}_{2^m}$.

D'autre part, on note que la classe de Sloane [MS83] est entièrement couverte par la classe de Kasami [Kas69].

Proposition 7. *Soit $m = 2\ell+1$. L'ensemble de définition d'un code cyclique de longueur $2^m - 1$ de la forme $\{1, 2^{\ell-1} + 1, 2^\ell + 1\}$ peut s'écrire sous la forme $\{1, 2^\ell + 1, 2^{3\ell} + 1\}$.*

Démonstration. Il suffit de montrer que $2^{\ell-1} + 1$ est congru à $2^{3\ell} + 1$ modulo $2^m - 1$.

$$\begin{aligned} 2^{3\ell} - 2^{\ell-1} = &\quad 2^{\ell-1}(2^{2\ell+1} - 1) \mod 2^m - 1 \\ = &\quad 2^{\ell-1}(2^m - 1) \mod 2^m - 1 \\ = &\quad 0 \\ 2^{3\ell} + 1 = &\quad 2^{\ell-1} + 1 \mod 2^m - 1 \end{aligned}$$

□

4.1.2 Résultats calculatoires

Nous utilisons une implémentation de l'algorithme de Chose-Joux-Mitton [CJM02] pour chercher des mots de poids $w = 6$ dans les codes avec un ensemble de définition $\{1, 2^i+1, 2^j+1\}$ pour $m < 20$ et pour tout $i \neq j$. Nous savons en effet par le Lemme 2 que si des codes de cette forme possède une distance minimale 5, il possède au moins un mot de poids 6. Il n'est donc pas nécessaire de chercher de mot de poids 5. On choisit de chercher des mots de poids 6 plutôt que de poids 5 en raison de l'implémentation qui est optimisée pour la recherche de mots de poids 6 ou 8. La complexité en temps de cet algorithme est de toute manière identique que le poids des mots recherchés vaille 5 ou 6. Dans la Table 3.2, nous donnons la liste exhaustive des codes cycliques 3-correcteurs de cette forme jusqu'à $m = 13$. Cet algorithme a une complexité en temps $\mathcal{O}(n^{\frac{w}{2}}) = \mathcal{O}(n^3)$ et une complexité en espace $\mathcal{O}(n^{\lceil \frac{w}{4} \rceil}) = \mathcal{O}(n^2)$. À

la base, cet algorithme est employé pour trouver les multiples de polynômes de poids faible dans la cryptanalyse de chiffrement à flot. Dans notre contexte, il s'agit d'un algorithme efficace pour chercher les mots de code de poids plus petits que 8. On observe, à partir de nos résultats, que chacun de ces ensembles de définition $\{1, 2^i + 1, 2^j + 1\}$ peut être écrit sous la forme $\{1, 2^\ell + 1, 2^{p\ell} + 1\}$, où $p = 2$ ou $p = 3$ jusqu'à $m < 20$. Par conséquent, la classe $\{1, 2^i + 1, 2^j + 1\}$ des codes cycliques 3-correcteurs est complètement décrite par deux classes connues [Kas71, BH09b] pour $m < 20$.

4.2 Calculer leurs énumérateurs des poids

Poids	# Mots de code
0	1
$\frac{N \pm \sqrt{8N}}{2}$	$\frac{(N^2 - 3N + 2)(N \mp \sqrt{8N})}{96}$
$\frac{N \pm \sqrt{2N}}{2}$	$\frac{(5N^2 + 3N - 8)(N \mp \sqrt{2N})}{24}$
$\frac{N}{2}$	$\frac{9N^3 - 3N^2 + 10N - 16}{16}$

TABLE 4.3 – La distribution de poids du dual du code BCH 3-correcteur de longueur $2^m - 1$, m impair, $N = 2^m$.

Le calcul de l'énumérateur des poids d'un code n'est pas une sinécure de façon générale (voir Section 8.1). On rappelle que la distribution de poids d'un code linéaire C et celle de son code dual $C^\perp$ sont reliées par les identités de MacWilliams et Pless [HP03b, Chapitre 7]. Par conséquent, connaissant la distribution de poids de $C^\perp$ on peut obtenir la distribution de poids de C. Le code dual $C^\perp$ du code C est l'orthogonal de C. Si C est cyclique, alors $C^\perp$ est aussi un code cyclique. Nous notons $Z^\perp$ l'ensemble de définition de $C^\perp$. Le polynôme générateur de $C^\perp$ est le polynôme réciproque de $h(X) = (X^n - 1)/g(X)$. Ses racines sont les inverses des racines de h. En d'autres termes, il est établi que :

$$z \in Z^\perp \Longleftrightarrow n - z \notin Z.$$

La distribution de poids du dual du BCH 3-correcteur pour m impair a été déterminée dans [Kas69] et nous la présentons dans la Table 4.3. Cependant, pour m pair, une formule explicite pour la distribution de poids du dual du code BCH n'est pas connue. En revanche, le lecteur peut consulter [MS83, pp. 451-452,469] pour obtenir la distribution de poids du dual du code BCH 2-correcteur pour tout m ainsi que celle du dual du code BCH 3-correcteur étendu pour m pair. Dans [vDV96, vDV97], il a été montré que le dual du code $\{1, 2^{\ell-1} + 1, 2^\ell + 1\}$ a la même distribution de poids que le dual du code BCH. En outre, à partir des identités de MacWilliams et Pless, le code $\{1, 2^{\ell-1} + 1, 2^\ell + 1\}$ a la même distribution de poids que le code BCH. Cela nous a motivés pour trouver si le code $\{1, 2^i + 1, 2^j + 1\}$ a la même distribution de poids que le code BCH. Comme [vDV96, vDV97], nous étudions aussi la distribution de poids de ces codes via leurs duaux. Puisque ces duaux ont moins de mots de code, il est possible

de calculer la distribution de poids pour des degrés d'extension supérieurs. Pour cela, nous implémentons un algorithme parallèle. Nous calculons d'abord les mots du code en utilisant le codage de Gray. Après cela, nous déterminons leurs poids de Hamming avec une instruction accelérée matériellement contenue dans le jeu d'instruction SSE4.

Pour $m \leq 13$, nous vérifions numériquement que les codes cycliques 3-correcteurs avec l'ensemble de définition $\{1, 2^i+1, 2^j+1\}$ possèdent la même distribution de poids que les BCH 3-correcteurs. Nous donnons la distribution de leurs duaux dans la Table 4.3.

Nous allons à présent prouver que ce résultat reste vrai pour tout m impair en nous appuyant sur le théorème suivant dû à Kasami [Kas69, Théorème 15 (ii)(1)].

Théorème 15. *Soit $m \geq 5$ un entier impair. Les codes cycliques binaires 3-correcteurs dont le dual est un sous-code cyclique du code de Reed-Muller cyclique $\mathcal{R}^*(2, m)$ avec une dimension $3m$ possèdent la même distribution de poids que les codes* BCH *3-correcteurs.*

Corollaire 3. *Soit $m \geq 5$ un entier impair. Les codes cycliques binaires 3-correcteurs de longueur $n = 2^m - 1$ avec l'ensemble de définition $Z = \{1, 2^i + 1, 2^j + 1\}$ possèdent la même distribution de poids que les codes* BCH *3-correcteurs.*

Démonstration. Soit C un code cyclique 3-correcteur avec $Z = \{1, 2^i+1, 2^j+1\}$. On a nécessairement $i \neq j$. Les poids des représentations binaires des entiers $1, 2^i+1, 2^j+1$ sont strictement inférieurs à 3. On a donc les relations d'inclusion suivantes :

$$\mathcal{R}^*(m-3, m) \subset C$$

$$C^\perp \subset (\mathcal{R}^*(m-3, m))^\perp$$

On note w_2, la fonction associant à un entier, le nombre de 1 dans son écriture binaire et on définit T comme étant l'ensemble de définition de $\mathcal{R}^*(m-3, m)$.

L'ensemble de définition de $(\mathcal{R}^*(m-3, m))^\perp)$ est l'ensemble :

$$\begin{aligned}\left\{ i : 0 < i < 2^m, \quad n - i \notin T \right\} &= \left\{ i : 0 < i < 2^m, \quad w_2(n-i) \geq 3 \right\} \\ &= \left\{ i : 0 < i < 2^m, \quad 0 < w_2(i) \leq m-3 \right\}\end{aligned}$$

On a ainsi l'égalité $(\mathcal{R}^*(m-3, m))^\perp = (\mathcal{R}(m-3, m)^\perp)^*$ et on peut écrire :

$$C^\perp \subset (\mathcal{R}(m-3, m)^\perp)^*$$

On rappelle que l'on a la propriété suivante : $\mathcal{R}(m-3, m)^\perp = \mathcal{R}(2, m)$.

On en déduit finalement que :

$$C^\perp \subset \mathcal{R}^*(2, m)$$

Montrons à présent que la dimension k de C vaut $n - 3m$.

On rappelle que $k = n + \deg(g)$ où g est le degré du polynôme générateur de C. Dans notre situation, le polynôme g est le produit des polynômes minimaux de α, α^{2^i+1} et α^{2^j+1}. Le degré de ces polynômes est égal à la taille des classes 2-cyclotomiques des entiers $1, 2^i+1, 2^j+1$. On doit montrer que $\deg(g) = 3m$. La taille de la classe 2-cyclotomique d'un entier i vaut m si l'on a $\gcd(i, 2^m - 1) = 1$. De plus, pour des entiers positifs a et b, on a :

$$\gcd(2^a + 1, 2^b - 1) = 1 \Leftrightarrow \frac{b}{\gcd(a, b)} = 1 \mod 2$$

Pour m impair, on obtient les égalités :

$$\begin{cases} \gcd(1, 2^m - 1) & = 1, \\ \gcd(2^i + 1, 2^m - 1) & = 1, \\ \gcd(2^j + 1, 2^m - 1) & = 1. \end{cases} \tag{4.13}$$

La dimension de C vaut donc $n - 3m$ et celle de $C^\perp$ vaut $3m$.

On peut dès lors appliquer le Théorème 15 et conclure la preuve.

□

Ce résultat ne couvre pas le cas où m est pair. Nous soulevons donc la question suivante :

Problème . *Prouvez ou infirmez que, pour m pair, tous les codes cycliques 3-correcteurs avec l'ensemble de définition $\{1, 2^i+1, 2^j+1\}$ de longueur 2^m-1 aient la même distribution de poids que le code BCH 3-correcteur de longueur $2^m - 1$.*

4.3 La question de l'équivalence avec le BCH 3-correcteur

Il est au moins aussi difficile de déterminer l'équivalence entre deux codes linéaires que l'existence d'un isomorphisme entre deux graphes [PR97]. Dans [MS83], il était demandé si les codes cycliques avec l'ensemble de définition $\{1, 2^{\ell-1} + 1, 2^\ell + 1\}$ sont équivalents au code BCH 3-correcteur. Ils ont conjecturé qu'ils ne l'étaient pas. Puisque nos résultats calculatoires montrent que tous les codes cycliques 3-correcteurs ayant l'ensemble de définition $\{1, 2^i+1, 2^j+1\}$ avec $m \leq 13$ ont la même distribution de poids que les codes BCH 3-correcteurs, nous nous sommes demandés si ces codes sont équivalents au code BCH. Nous utilisons l'implémentation Magma de l'algorithme de Léon [Leo82] pour prouver la non équivalence pour $m = 7$ et $m = 8$. En particulier, pour $m = 7$, le code cyclique 3-correcteur avec l'ensemble de définition $\{1, 2^{\ell-1} + 1, 2^\ell + 1\}$, pour $\ell = 3$ n'est pas équivalent avec le code BCH 3-correcteur $\{1, 3, 5\}$. Cela supporte la conjecture faite dans [MS83, page 291]. Nous employons l'algorithme de séparation du support [Sen00] pour prouver la non équivalence pour $m = 10$. Cet algorithme possède une complexité polynomiale en la longueur du code et exponentielle en la dimension de son hull. L'énumérateur des poids du hull d'un code est un invariant par permutation. Nous notons que les codes cycliques 3-correcteurs avec l'ensemble de définition $\{1, 2^i + 1, 2^j + 1\}$ sont auto-orthogonaux (ou faiblement auto-duaux) pour $m < 20$, *i.e.* le hull du code est le code lui-même. Si nous poinçonnons deux codes équivalents en chaque position, le multi-ensemble des énumérateurs de poids de chaque code poinçonné est le même

pour les deux codes. Cet objet est la *signature* du code que nous calculons pour déterminer l'équivalence de deux codes. Les codes cycliques ont un groupe d'automorphismes transitif. Cela implique que si nous poinçonnons un code cyclique dans n'importe quelle position, nous obtenons le même énumérateur des poids pour chaque code poinçonné. Ainsi, nous poinçonnons les codes duaux dans une position fixée d'abord. Puis, nous les poinçonnons une seconde fois en chaque position. Nous calculons la signature de ces codes duaux. Nous obtenons des signatures qui sont différentes à partir de la signature du dual du code BCH $\{1, 3, 5\}$ pour $m = 10$. De sorte que nous pouvons conclure sur la non équivalence des codes cycliques 3-correcteurs non BCH avec l'ensemble de définition $\{1, 2^i + 1, 2^j + 1\}$ avec le BCH. Pour $m = 9$, nous obtenons la même signature que celle du code BCH $\{1, 3, 5\}$. Cette signature n'est pas assez discriminante pour collecter des informations sur une éventuelle permutation.

Exemple 18. On choisit d'étudier le code cyclique C à trois zéros $\{1, 9, 17\}$. On veut déterminer si ce code est équivalent au code BCH avec l'ensemble de définition $\{1, 3, 5\}$ pour des longueur $n = 512$ et $n = 1024$ (autrement dit pour $m = 9$ et $m = 10$). On met en œuvre un algorithme parallèle qui calcule l'énumérateur de poids d'un code. On se sert de cette implémentation sur plusieurs ordinateurs équipés de quatre à seize processeurs quadri-cœurs. On constate alors que ces deux codes possèdent le même énumérateur de poids pour $m = 9$ et $m = 10$. Pour cela, on calcule l'énumérateur de leurs duaux puisqu'il s'agit de codes de dimension plus petite (elle vaut $3m$ pour le dual contre $n - 3m$ pour le code). Le code $C^\perp$ et le dual du code BCH sont des codes cycliques. Ils possèdent donc chacun un groupe d'automorphismes transitif. Les hypothèses de la Proposition 6 sont donc vérifiées. On poinçonne donc $C^\perp$ et le code BCH en une position quelconque. On calcule et compare alors l'énumérateur des poids de ces codes poinçonnés une fois. Il est identique pour $m = 9$ et $m = 10$. On ne peut donc pas conclure à cette étape sur l'équivalence de C avec le code BCH. On va poinçonner les codes une seconde fois. Dans le cas présent, on les poinçonnera en chaque position puisque l'on n'est pas assuré d'obtenir des codes équivalents en poinçonnant en des positions distinctes. On compare ensuite les multi-ensembles d'énumérateur des poids. On se retrouve dans deux situations opposés selon la valeur de m. Pour $m = 9$, les deux multi-ensembles sont identiques. On ne peut dès lors pas conclure sur l'équivalence des codes. Le multi-ensemble possède un seul élément de multiplicité 510. La signature obtenue n'est *absolument* pas discriminante puisque tous les codes poinçonnés possèdent le même énumérateur des poids. On obtient donc aucune information sur une éventuelle permutation. Il faut poinçonner 1020 codes de longueur 509 en chaque coordonnée pour calculer le multi-ensemble des codes poinçonnés trois fois. Cela demande un effort calculatoire trop conséquent, il faudrait pouvoir traiter un demi million de codes. En revanche, la situation est fort différente pour $m = 10$ et nous permet alors de conclure directement sur la non-équivalence de C en longueur 1024 avec le code BCH $\{1, 3, 5\}$ en poinçonnant les codes deux fois et en calculant le multi-ensemble des énumérateurs de poids des codes obtenus. La signature est cette fois-çi *totalement* discriminante, les multi-ensembles du code BCH et de C possèdent respectivement 8 et 10 éléments.

Nous concluons cette section en établissant qu'aucun des codes cycliques 3-correcteurs non BCH *avec l'ensemble de définition* $\{1, 2^i + 1, 2^j + 1\}$ *n'est équivalent au code* BCH *3-correcteur pour* $m = 7$*,* $m = 8$ *et* $m = 10$*. La question demeure ouverte pour* $m = 9$*.*

Deuxième partie

Le chiffrement et les codes de Goppa

Introduction

La recherche de racines de polynômes sur les corps finis est un problème classique d'algèbre algorithmique. Il est considéré comme l'une des sous-étapes les plus consommatrices en temps de la procédure de décodage des codes alternants. Il existe plusieurs approches bien connues pour trouver les racines du polynôme localisateur d'erreurs. L'algorithme de recherche de racines le plus employé pour le décodage est la méthode de la recherche de Chien [Chi64], qui est une évaluation du polynôme en tous les points du corps, de telle sorte qu'il a une complexité en temps très grande pour les cas où l'on travaille sur des grands corps et que les polynômes soient de haut degré. L'algorithme de la Trace de Berlekamp (BTA) [Ber71] est une autre méthode bien connue. C'est une méthode récursive basée sur les propriétés de la fonction trace.

Le cryptosystème de McEliece est considéré comme l'un des schémas à clé publique les plus rapides. Il en existe plusieurs variantes. Le système originel de McEliece [McE78] (décrit dans la Section 5.2) ainsi que le système de McEliece hybride [BS08] utilisent les codes de Goppa binaires [Gop70]. Leur procédure de déchiffrement emploie un algorithme de décodage algébrique qui est souvent séparé en trois parties, à savoir : le calcul du syndrome, trouver la solution de l'équation clé, et la recherche des racines du polynôme localisateur d'erreurs. Cette dernière étape prend en pratique les trois quarts du temps total de déchiffrement. Nous présentons une méthode hybride impliquant BTA et une méthode proposée par Zinoviev [Zin96]. Zinoviev a proposé des procédures directes de recherche de racines pour des polynômes avec un dégré au plus 10. Notre idée est de calculer directement les racines avec les procédures de Zinoviev jusqu'à un certain degré et d'utiliser BTA pour des degrés supérieurs. De plus, nous améliorons les procédures de Zinoviev pour des polynômes de degré 2 et 3 avec des compromis temps-mémoire. Nous analysons à la fois les complexités théoriques et les complexités expérimentales de notre proposition. Nous obtenons un gain théorique de 93% sur la méthode de Chien et 46% sur BTA. Des résultats expérimentaux confirment la théorie jusqu'au degré 4 au moins. Par exemple avec $m = 11$, $t = 32$ et $d_{max} = 4$, notre méthode prend 60% du temps total de déchiffrement par rapport au 72% pour BTA et 87% pour Chien.

Le Chapitre 6 est dédié aux travaux liés à la recherche de racines de polynômes sur des corps de caractéristique 2. Dans la Section 6.5, nous présentons notre proposition et dans la Section 6.6, nous présentons les résultats de simulation pour étayer notre proposition.

Motivation

Soit $\mathbb{F}_{2^m}$ l'extension de degré m du corps à deux éléments $\mathbb{F}_2$. Nous considérons un polynôme unitaire univarié f, de degré $t > 0$, dans l'anneau des polynômes $\mathbb{F}_{2^m}[z]$. Sans perte de généralité, nous supposons que f n'a pas de racine multiple et que f se factorise en facteurs linéaires sur le corps à 2^m élements $\mathbb{F}_{2^m}$ (*e.g.* voir [LN97]). Notre but est de trouver une manière efficace, en termes de complexité en temps et en espace, de trouver les zéros de f.

Nous nous sommes spécialement intéressés à ce problème dans le contexte McEliece.

L'efficacité des algorithmes de recherche de racines est un problème que nous étudions en cryptographie basée sur les codes. Les cryptosystèmes de type McEliece sont souvent basés sur les codes de Goppa binaires (présentés dans la Section 5.1). Leur algorithme de déchiffrement emploie une procédure de décodage algébrique pour retrouver le message original à partir du texte chiffré. L'étape la plus consommatrice en temps, dans l'implémentation du décodage algébrique des codes de Goppa binaires, avec des paramètres pratiques, est la recherche des racines du polynôme localisateur d'erreurs. Ce polynôme vérifie les propriétés mentionnées ci-dessus.

Nous considérons un code de Goppa binaire de longueur n qui corrige jusqu'à t erreurs (l'algorithme utilisé est donné dans la Section 5.3). Rappelons que, en pratique, les paramètres sont choisis tels que : $n = 2^m$ et $mt \leq n$.

5 Déchiffrer les cryptosystèmes de type McEliece

5.1 Les codes de Goppa classiques (1970)

Soient $m > 0$, $n \leq 2^m$ et $a = (a_1, ..., a_n) \in \mathbb{F}_2^n$.

Le code de Goppa binaire $\Gamma(L, g)$ de longueur n est défini par :
- Support $L = (\alpha_1, ..., \alpha_n)$ n-uplet d'éléments distincts de $\mathbb{F}_{2^m}$;
- Polynôme de Goppa $g(z) \in \mathbb{F}_{2^m}[z]$, sans racine multiple, unitaire de degré $t > 0$ sans racine dans L.

En toute rigueur, le code $\Gamma(L, g)$ est un code de Goppa *séparable*, du fait que g est supposé séparable (*i.e.* sans racine multiple) sur $\mathbb{F}_{2^m}$. En pratique, on prend $n = 2^m$, le polynôme g est alors irréductible sur $\mathbb{F}_{2^m}$.

$\Gamma(L, g)$ est un sous-code sur le sous-corps $\mathbb{F}_2$ (*i.e.* le sous-code que contient tous les mots de code dont les coordonnées sont dans $\mathbb{F}_2$) d'un code géométrique de Goppa particulier sur le corps à 2^m élements $\mathbb{F}_{2^m}$. Nous avons $a \in \Gamma(L, g)$ si et seulement si :

$$R_a(z) := \sum_{i=1}^{n} \frac{a_i}{z - \alpha_i} = 0 \text{ sur } \mathbb{F}_{2^m}[z]/(g(z)).$$

R_a est appelé le syndrome du mot a. Notez que R_a est $\mathbb{F}_2$-linéaire en a.

Les codes de Goppa *binaires* ont une capacité de correction d'au moins t erreurs. Nous présentons, dans la Section 5.3, un algorithme en temps polynomial pour le décodage de ces codes qui corrige jusqu'à t erreurs.

Remarque 33. On ignore la dimension et la distance minimale des code de Goppa. Néanmoins, on connaît des bornes inférieures sur ces quantités.

Propriété 13. *La dimension de* $\Gamma(L, g)$ *est supérieure à* $n - mt$.

Propriété 14. *La distance minimale de* $\Gamma(L, g)$ *est supérieure à* $2t + 1$.

Remarque 34. La Propriété 13 s'applique également aux codes de Goppa non-binaires. La Propriété 14 s'applique uniquement aux codes de Goppa binaires séparables. La distance minimale des codes de Goppa non-binaires est minorée par $t + 1$. Autrement dit, ils possèdent une capacité de correction d'au moins $\frac{t}{2}$ erreurs.

Remarque 35. Le code Goppa $\Gamma(L, g)$ est le code alternant $\mathcal{A}(\alpha, \gamma)$ défini par :

$$k = n - t, \, \alpha = L \text{ et } \gamma = (g(\alpha_1)^{-1}, \ldots, g(\alpha_n)^{-1}).$$

Remarque 36. Les codes de Goppa peuvent également être définis en termes de résidus de fonctions rationnelles. Cette caractérisation est présentée dans [HP03b, page 524]. Une autre caractérisation en est donné dans [Bla08, pp. 101-102]. Ces codes forment aussi une sous-classe des codes générés par le théorème des restes chinois [Man75].

Remarque 37. Supposons que $n \mid (2^m - 1)$. Soit α une racine primitive n-ième de l'unité dans $\mathbb{F}_{2^m}$. Le code de Goppa binaire $\Gamma(L, g)$ où $g(z) = z^{d-1}$ et $L = (1, \alpha, \alpha^{-2}, \ldots, \alpha^{-(n-1)})$ est un code BCH au sens strict de distance construite d.

5.2 Les cryptosystèmes de McEliece et Niederreiter

À l'instar du cryptostème de Merkle-Hellman [MH78], le déchiffrement des cryptosystèmes de McEliece et de Niederreiter requiert la résolution d'une instance d'un problème NP-complet qui se révèle facile dès lors que l'on dispose de la clef secrète. Sans cette donnée, le cryptanalyste est confronté à une instance quelconque d'un problème NP-complet ; il s'agit ici du décodage d'un code linéaire arbitraire. Tandis que pour déchiffrer le message, il suffit au destinataire du message, qui détient la clef secrète, de décoder un code particulier pour lequel il connaît un algorithme efficace de décodage. Dans notre contexte, ce code particulier appartient à la classe des codes de Goppa binaires classiques. Il est à noter que le cryptosystème de McEliece [McE78] est le premier cryptosystème à clef publique qui se sert d'aléa pendant le chiffrement de message ainsi que le premier fondé sur la théorie algébrique des codes. Les schémas de McEliece et de Niederreiter associés aux codes de Goppa binaires possèdent entre autres avantages, des algorithmes de chiffrement et de déchiffrement très rapides. De plus, ils appartiennent, avec les cryptosystèmes basés sur les réseaux, à la classe restreinte des cryptosystèmes qui résisteraient au développement d'ordinateurs quantiques. En contrepartie, leur clef publique prend beaucoup d'espace mémoire, ce qui a été rédhibitoire pour leur développement dans la pratique jusqu'à présent. Il existe deux types d'attaque contre ces codes. Il s'agit des attaques structurelles et des attaques par décodage. Les attaques structurelles visent à identifier le code de Goppa dont la structure a été masquée par le biais en lui appliquant une isométrie (dans le cas binaire, il s'agit de permuter le support du code). À ce jour, les attaques structurelles sur les codes de Goppa possèdent un coût exponentiel. Les attaques par décodage ont quant à elles pour objectif de déchiffrer le cryptogramme transmis en décodant un code linéaire aléatoire. Elles sont plus efficaces que les attaques structurelles. Vous pourrez trouver plus de détails à ce sujet dans [BBD09, pp.112-127].

5.2.1 Description du cryptosystème classique de McEliece (1978)

Paramètres : Des entiers n, k, t et une famille de codes.

Clé publique : Un code linéaire $[n, k]$ binaire C, *i.e.* un $\mathbb{F}_2$-sous-espace linéaire k-dimensionnel de $\mathbb{F}_2^n$, décrit par une matrice génératrice G.

Clé privée : Un algorithme de décodage efficace pour C jusqu'à la capacité de correction t.

Chiffrement : Faire correspondre aux k bits du texte clair x le mot de code $x.G$, ajouter e, une erreur uniformément aléatoire de longueur n et de poids t pour obtenir le texte chiffré y.

Déchiffrement : Corriger les t erreurs, faire l'inverse de l'application pour obtenir le message. Cette étape est également appelée décodage.

5.2.2 Description du cryptosystème de Niederreiter (1986)

Niederreiter a proposé un cryptosystème inspiré du problème de la somme de sous-ensembles (un cas particulier du problème du sac à dos) et basé sur les codes correcteurs dans [Nie86]. Cette proposition possède une sécurité équivalente [LDmW94] au cryptosystème classique de McEliece.

Paramètres : Des entiers n, k et t, une famille de codes ainsi qu'une application [1] dite de codage à poids constant $\phi_{n,t} : \mathbb{F}_2^l \mapsto W_{n,t}$ où $W_{n,t}$ désigne l'ensemble des mots binaires de poids t et de longueur n.

Clé publique : Un code linéaire $[n, k]$ binaire C, *i.e.* un $\mathbb{F}_2$-sous-espace linéaire de codimension $r = n - k$ de $\mathbb{F}_2^n$, décrit par une matrice génératrice H.

Clé privée : Un algorithme de décodage efficace pour C jusqu'à la capacité de correction t.

Chiffrement : Faire correspondre aux l bits du texte clair x le mot d'erreur $e = \phi_{n,t}(x)$, calculer le syndrome de e pour obtenir le texte chiffré de longueur r.

Déchiffrement : Décoder le syndrome, appliquer l'inverse de $\phi_{n,t}$ pour obtenir le message.

5.3 Algorithme de décodage algébrique des codes de Goppa

Soient e, x, y des vecteurs binaires de longueur n. Nous devons trouver x, le mot de code envoyé à partir $y = x + e$ où y est le mot reçu et e, le mot erreur.

Le décodage algébrique est réalisé en trois étapes :

1. Calcul du syndrome du mot reçu

$$R_y(z) = R_e(z) = \sum_{i=1}^{n} \frac{e_i}{z - \alpha_i} \text{ sur } \mathbb{F}_{2^m}[z]/(g(z)).$$

2. Résoudre l'équation clé pour obtenir le *polynôme localisateur d'erreurs* σ_e avec l'algorithme de Patterson [Pat75].

$$R_e(z) \cdot \sigma_e(z) = \sigma_e'(z) \text{ sur } \mathbb{F}_{2^m}[z]/(g(z)).$$

   Notation : σ_e' représente la dérivée formelle de σ_e.

1. L'application $\phi_{n,t}$ est injective. En pratique, on choisit le plus petit entier l telle que $\binom{t}{n} > l$.

3. Recherche des racines du polynôme localisateur d'erreurs

$$\sigma_e(z) := \prod_{i=1}^{n} (z - \alpha_i)^{e_i} \; ; \; e_i \neq 0 \Leftrightarrow \sigma_e(\alpha_i) = 0.$$

Nous insistons sur le fait que le degré de σ_e est également le poids de Hamming de e, c'est à dire, le nombre d'erreurs à corriger. L'algorithme peut corriger jusqu'à t erreurs donc le degré de σ_e est inférieur ou égal à t. On ne connaît pas d'algorithme qui permette de corriger jusqu'à la capacité de correction les codes de Goppa. Par ailleurs, on sait qu'il existe beaucoup de bons codes de Goppa (*i.e.* possédant un grande dimension et une grande distance minimale), alors qu'il n'existe pas de bons codes BCH par exemple. Malheureusement, on ne connaît pas de moyen pour les identifier.

Rappel . Pour tout $\alpha \in \mathbb{F}_{2^m}$, la fonction $z \mapsto z - \alpha$ est inversible modulo g si et seulement si les fonctions $g(z)$ et $z \mapsto z - \alpha$ sont premières entre elles. On en déduit que le syndrome $R_y(z)$ est bien un élément de $\mathbb{F}_{2^m}[z]/(g(z))$ puisque l'on suppose que g, le polynôme de Goppa, ne s'annule pas sur les éléments du support L.

Remarque 38. La plupart des algorithmes de décodage des codes alternants se décomposent selon les trois étapes décrites ci-dessus. De nombreuses méthodes, adaptées spécialement aux codes de Goppa, sont connues pour trouver le polynôme localisateur d'erreurs [Pat75, Ret75, SKHN75, SKHN76]. Il existe aussi des méthodes plus générales pour les codes alternants [MS83, pp. 365-369] ou [Man77] pour les codes de Goppa généralisés. On invite le lecteur à consulter [HP03b, pp. 178-195] et [Bla03, page 195, pp. 226-227], pour le cas particulier des codes BCH. Ces algorithmes reposent généralement soit sur l'algorithme de Sugiyama [SKHN75, SKHN76] qui diffère de l'algorithme d'Euclide de par la condition de terminaison, soit sur l'algorithme itératif de Berlekamp-Massey [Ber68a, Chapitre 7], [Mas69]. Ces deux algorithmes sont présentés puis comparés en détail dans [Bla08, pp. 151-163, pp. 173-176]. L'algorithme de Berlekamp-Massey fut introduit par Berlekamp pour le décodage des codes BCH, puis Massey l'a reformulé, de façon plus simple, en découvrant que l'algorithme permet de trouver le LFSR de taille minimale qui synthétise une séquence donnée.

5.4 Complexité théorique du déchiffrement

Complexité Théorique = nombre d'opérations arithmétiques dans $\mathbb{F}_{2^m}$ requises pour déchiffrer dans le pire des cas.

- Calcul du syndrome $\mathcal{O}(nt)$
- Résolution de l'équation clé $\mathcal{O}(t^2)$
- Recherche des racines du polynôme localisateur d'erreurs
  - BTA $\mathcal{O}(mt^2)$
  - Recherche de Chien $\mathcal{O}(nt)$

Complexité Expérimentale = temps moyen d'exécution pour le déchiffrement.

Nous donnons ci-dessous le pourcentage de temps total passé dans chaque étape de l'algorithme de déchiffrement. Ces chiffres ont été obtenus via notre mise en œuvre de l'algorithme.

- Calcul du syndrome 11.3%

- Résolution de l'équation clé 12.0%
- Recherche des racines du polynôme localisateur d'erreurs
  - BTA 72.1%
  - Recherche de Chien 85.6% [2]
- Autres tâches 4.6%

Asymptotiquement, le calcul du syndrome est le coût dominant. pour les paramètres recommandés (*i.e.* $m = 11, t = 32$), l'étape la plus consommatrice en temps dans le déchiffrement (décodage) consiste à trouver les racines de σ_e. Les différences qui peuvent apparaître entre les complexités théoriques et expérimentales pourraient être dûes à plusieurs raisons :

- la qualité de l'implémentation, les processeurs et les compilateurs utilisés ;
- les coûts des tests conditionnels et des accès mémoire (qui sont négligés dans notre étude théorique mais qui peuvent être non marginaux en pratique) ;
- des approximations pas si bonnes (*e.g.* nous approximons le coût d'une inversion de matrice binaire d'ordre m à m^2 opérations sur le corps, nous négligeons ainsi une petite constante multiplicative) ;
- la pondération [3] des opérations arithmétiques dans le corps.

2. Les pourcentages donnés sont associés avec BTA, si nous prenons la recherche de Chien, les nombres pour le calcul du syndrome et la résolution de l'équation clé doivent varier en conséquence.

3. Dans notre étude, nous distinguons deux cas : dans le premier, l'addition et la multiplication coûtent toutes deux une opération dans $\mathbb{F}_{2^m}$, nous notons cela $K(+) = K(\times) = 1$ où K est le coût de la fonction d'une opération arithmétique, dans le second, $K(+) = 1$ et $K(\times) = m$.

6 Les algorithmes de recherche de racines

Plusieurs approches pour chercher les racines en caractéristique 2 sont possibles, leur efficacité dépend de la taille des paramètres m et t.

- La recherche de Chien (décrite dans la Section 6.1) calcule les racines en évaluant astucieusement le polynôme en tous les points de L. Cette méthode est recommandée pour des implémentations matérielles et des applications de théorie des codes dans lesquels m est petit.
- BTA est un algorithme récursif utilisant les propriétés de la fonction trace. C'est une méthode plus rapide pour des paramètres considérés sûrs pour les cryptosystèmes de type McEliece.
- La séparation des facteurs de même degré est un algorithme de Cantor et Zassenhaus [vzGG03, Chapter 14]. Sa portée est plus générale mais sous certaines adaptations, il permet de trouver les racines de polynômes en caractéristique 2 (ce cas spécifique est traité dans l'Exercice 14.16 dans [vzGG03]). De plus, il possède des similitudes avec BTA (utilisation de la fonction trace, calcul de pgcd et une structure récursive) mais il semble avoir une complexité légèrement plus importante. Sa complexité en temps est $\mathcal{O}((m + \log t)\, t^2 \log t)$ opérations dans $\mathbb{F}_{2^m}$.
- Les procédures de Zinoviev sont dédiées à la recherche de racines pour des polynômes de degrés inférieurs à 10. Pour $m \geq 11$, elles sont (théoriquement au moins) plus efficaces que la recherche de Chien.

6.1 La procédure de Chien (1964)

La recherche de Chien est un algorithme récursif. Il s'agit d'une recherche exhaustive intelligente. Soit $f(x) = a_0 + a_1 \cdot x + \ldots + a_t \cdot x^t$ un polynôme sur $\mathbb{F}_{2^m}$ et soit α un élément primitif du groupe multiplicatif $\mathbb{F}_{2^m}^{\times}$.

$$\begin{aligned} f(\alpha^i) &= a_0 + a_1 \cdot \alpha^i + \ldots + a_t \cdot (\alpha^i)^t \\ f(\alpha^{i+1}) &= a_0 + a_1 \cdot \alpha^{i+1} + \ldots + a_t \cdot (\alpha^{i+1})^t \\ &= a_0 + a_1 \cdot \alpha^i \cdot \alpha + \ldots + a_t \cdot (\alpha^i)^t \cdot \alpha^t \end{aligned}$$

Posons $a_{i,j} = a_j(\alpha^i)^j$. Il est facile d'obtenir $f(\alpha^{i+1})$ à partir $f(\alpha^i)$ puisque nous avons $a_{i+1,j} = a_{i,j} \cdot \alpha^j$. De plus, si $\sum_{j=0}^{t} a_{i,j} = 0$, alors α^i est une racine de f.

6.2 L'algorithme de la trace de Berlekamp (1971)

L'application $\mathrm{Tr}(\cdot)$ est $\mathbb{F}_2$-linéaire et nous savons que : $z^{2^m} - z = \mathrm{Tr}(z) \cdot (\mathrm{Tr}(z) - 1)$.

La *fonction Trace* possède la propriété suivante :

$$\mathrm{Tr}(z) - i = \prod_{\gamma \ s.t.\ \mathrm{Tr}(\gamma)=i} (z - \gamma), \forall i \in \mathbb{F}_2.$$

Soit $B = (\beta_1, \ldots, \beta_m)$ une base de $\mathbb{F}_{2^m}$ sur $\mathbb{F}_2$. Tout élément $\alpha \in \mathbb{F}_{2^m}$ est uniquement représenté par le m-uplet binaire $(\mathrm{Tr}(\beta_1 \cdot \alpha), \ldots, \mathrm{Tr}(\beta_m \cdot \alpha))$.

BTA scinde n'importe quel $f \in \mathbb{F}_{2^m}[z]$, tel que $f(z) \mid (z^{2^m} - z)$, en facteurs linéaires en calculant itérativement sur $\beta \in B$ et récursivement sur f :

$$g(z) \coloneqq \mathrm{pgcd}(f(z), \mathrm{Tr}(\beta \cdot z)) \text{ et } h(z) \coloneqq \frac{f(z)}{g(z)} = \mathrm{pgcd}(f(z), \mathrm{Tr}(\beta \cdot z) - 1).$$

Premier appel : $f = \sigma_e$ et $\beta = \beta_1$. BTA retourne toujours avec succès les facteurs linéaires de f à cause des propriétés de la trace données ci-dessus. Plus de détails sur BTA peuvent être trouvés dans [Ber71] et [MvOV88]. En pratique, nous précalculons à chaque appel, les polynômes $\mathrm{Tr}(\beta_i \cdot z) \mod f(z), \forall i \in \{0, \ldots, m-1\}$. Le coût de ce précalcul est $\mathcal{O}(mt^2)$. Nous insistons sur le fait que il ne s'agit pas d'un coût négligeable. En effet, rappelons que BTA sans ce précalcul a également un coût de $\mathcal{O}(mt^2)$ opérations dans $\mathbb{F}_{2^m}$.

6.3 L'algorithme de Cantor et Zassenhaus' (1981)

Soient m et r des entiers positifs, p un nombre premier et $q = p^m$. L'algorithme de Cantor et Zassenhaus' permet de factoriser sur $\mathbb{F}_q$ un polynôme qui divise $X^{p^r} - X$. Autrement dit, il permet de factoriser un polynôme possédant des facteurs de même degré r et sans facteur carré [1] [CZ81], [PZ89, pp. 81-85], [Coh93, pp. 128-130]. Il s'agit d'un algorithme probabiliste et dont la sortie est toujours correcte. Soit t le degré du polynôme. Pour $p = 2$, le nombre moyen d'opérations dans $\mathbb{F}_{2^m}$ est de l'ordre $\mathcal{O}(mt^2r)$. Il est à noter qu'on ne connaît pas d'algorithme déterministe et polynomial pour factoriser des polynômes sur les corps finis. On peut adapter cet algorithme à notre contexte, c'est-à-dire à la recherche de racines dans des corps de caractéristique 2. Cet algorithme est présenté dans [vzGG03, Chapter 14]. Dans cette situation, r vaut 1. En pratique, cet algorithme se révèle néanmoins moins rapide que l'algorithme

1. De manière plus générale, la factorisation d'un polynôme univarié sur les corps finis se déroule en trois étapes : la factorisation sans carré qui permet de supprimer les facteurs multiples, la séparation des facteurs de degrés distincts et la séparation des facteurs de même degré. Cette dernière étape est la plus complexe. Elle est réalisée par l'algorithme de Cantor et Zassenhaus'.

de Berlekamp pour la factorisation sur des corps finis de petite caractéristique. Cet écart de performance croît lorsque que le degré d'extension m croît. Le lecteur peut consulter [Sho06, pp.532-543] pour obtenir des informations supplémentaires sur ce sujet.

6.4 Les procédures de Zinoviev (1996)

Les méthodes de Zinoviev [Zin96] trouvent le multiple affine unitaire de plus petit degré de n'importe quel polynôme f de degré $d \leq 10$ sur $\mathbb{F}_{2^m}$. La notion de multiple affine est définie ci-après. À l'étape $i \geq 0$, nous calculons un multiple de f de degré $2^{\lceil \log_2 d \rceil + i}$ et nous essayons d'éliminer les termes non-linéaires en résolvant un système homogène d'équations linéaires. Si le système n'a pas de solution, nous allons à l'étape $i+1$. Par ailleurs, un algorithme proposé par Berlekamp, Rumsey et Solomon dans [BRS67] assure de trouver un multiple affine de degré 2^{d-1} et garantit ainsi que les méthodes de Zinoviev terminent, dans le pire des cas, à l'étape $d - 1 - \lceil \log_2 d \rceil$. Après cela, trouver les racines d'un polynôme affine est plus facile que dans le cas général. Pour cela, nous avons seulement à résoudre un système linéaire d'ordre m sur $\mathbb{F}_2$.

6.4.1 Trouver les racines d'un polynôme affine

Définition 60. Un polynôme linéarisé sur $\mathbb{F}_{q^m}$ est un polynôme de la forme :

$$L(z) = \sum_{i=0}^{n} l_i \cdot z^{q^i}.$$

avec $l_i \in \mathbb{F}_{q^m}$ et $l_n = 1$.

Dans notre cas, $q = 2$. Le *polynôme Trace* est un exemple de polynôme linéarisé.

Définition 61. Un polynôme affine est de la forme :

$$A(z) = L(z) + c$$

où L est un polynôme linéarisé sur $\mathbb{F}_{q^m}$ et $c \in \mathbb{F}_{q^m}$.

Soit un polynôme affine $A(z) = L(z) + c = \sum_{i=0}^{m-1} l_i \cdot z^{2^i} + c$.
Considérons que $(\alpha_1, \ldots, \alpha_m)$ est une $\mathbb{F}_2$-base de $\mathbb{F}_{2^m}$, $(l_i)_{1 \leq i \leq m}$, c et x sont des éléments de $\mathbb{F}_{2^m}$. Supposons que $x = (x_1, \ldots, x_m)$ soit une racine de A. Trouver les zéros d'un polynôme affine revient à résoudre un système linéaire. En effet, nous avons :

$$\begin{aligned}
A(x) = 0 &\Leftrightarrow L(x) = c \\
&\Leftrightarrow \sum_{i=1}^{m} x_i \cdot L(\alpha_i) = \sum_{i=1}^{m} c_i \cdot \alpha_i \quad \text{(en utilisant la linéarité de } L\text{)} \\
&\Leftrightarrow \sum_{i=1}^{m} \sum_{j=1}^{m} x_i \, l_{i,j} \cdot \alpha_i = \sum_{i=1}^{m} c_i \cdot \alpha_i \quad \text{(système linéaire en } x_i\text{)}.
\end{aligned}$$

Ensuite, nous devons déterminer les racines de f, parmi les racines du polynôme affine que nous avons trouvées. Nous évaluons juste f en ces points pour réaliser cela.

Considérons que q est une puissance première et que m est un entier positif. Donnons des définitions utiles :

6.5 L'algorithme Berlekamp Trace - Zinoviev

6.5.1 Accélérer le déchiffrement de McEliece

L'inconvénient de BTA est le grand nombre d'appels récursifs quand les paramètres du système grandissent. Nous le réduisons en mélangeant BTA avec les procédures de Zinoviev qui sont des méthodes ad-hoc pour trouver les racines de polynômes de degré ≤ 10. Cela est une technique classique employée pour diminuer le nombre de niveaux récursifs, *e.g.* l'algorithme bien connu Quicksort est optimisé avec des méthodes analogues. Nous appelons cette procédure BTZ (Berlekamp Trace - Zinoviev) dans l'ensemble de ce document. BTZ dépend d'un paramètre d_{max} qui est le degré maximal jusqu'auquel nous utilisons les méthodes de Zinoviev. Nous donnons à présent deux pseudocodes de BTZ.

6.5.2 Pseudocode de BTZ

Quelques notations :

- σ_e est le polynôme localisateur d'erreurs du mot erreur e.
- d_{max} est le degré maximal jusqu'auquel nous utilisons les procédures de Zinoviev.
- D_{Zin} est l'ensemble des degrés pour lesquels nous appliquons les procédures de Zinoviev.
- $(\beta_1, \ldots, \beta_m)$=$(\alpha, \alpha^2, \ldots, \alpha^m)$ est une base polynomiale fixée de $\mathbb{F}_{2^m}$ sur $\mathbb{F}_2$ où α est un élément primitif de $\mathbb{F}_{2^m}$.

Algorithme 2 BTZ simplifié sans précalcul - $\mathrm{BTZ}(f, d, i)$

Premier appel : $f \leftarrow \sigma_e$; $d \leftarrow d_{max} \in \{2, \ldots, 10\}$; $i \leftarrow 1$.
si degré$(f) \leq d$ **alors**
  retourner $\mathrm{ZINOVIEV}(f, d)$;
sinon
  $g \leftarrow \mathrm{pgcd}(f, \mathrm{Tr}(\beta_i \cdot z))$;
  $h \leftarrow f/g$;
  retourner $\mathrm{BTZ}(g, d, i+1) \cup \mathrm{BTZ}(h, d, i+1)$;
fin si

6.5.3 Mise en œuvre

Dans notre implémentation, nous utilisons une base polynomiale pour représenter les éléments du corps. Nous avons implémenté les procédures de Zinoviev avec des compromis temps-mémoire pour des polynômes de degré 2 et 3, qui permettent d'obtenir de meilleures

Algorithme 3 BTZ avec précalcul - $\mathrm{BTZ}(f, D, i)$

Premier appel : $f \leftarrow \sigma_e$; $D \leftarrow D_{Zin} \subset \{2, \ldots, 10\}$; $i \leftarrow 1$.
{phase de précalcul}
pour $1 \leq i \leq m$ **faire**
  $T_i \leftarrow \mathrm{Tr}(\beta_i \cdot z)) \mod f$;
fin pour
$i \leftarrow 1$;
{phase de calcul}
si degré$(f) \in D$ **alors**
  retourner $\mathrm{ZINOVIEV}(f, d)$;
sinon
  $T \leftarrow T_i \mod f$;
  $g \leftarrow \mathrm{pgcd}(f, T)$;
  $h \leftarrow f/g$;
  $i \leftarrow i + 1$;
  retourner $\mathrm{BTZ}(g, d, T) \cup \mathrm{BTZ}(h, d, T)$;
fin si

performances qu'avec les procédures originales. Nous expliquons ces nouvelles méthodes et donnons leurs complexités dans la suite.

Compromis Temps-Mémoire pour le Degré 2.

Nous voulons résoudre l'équation : $z^2 + az + b = 0$ pour $a, b \in \mathbb{F}_{2^m}$. Si les solutions existent, nous les notons z_1 et z_2. D'abord, nous effectuons un changement de variable. Nous posons $z = ax$. Nous obtenons l'équation $x^2 + x + b/a^2 = 0$. Cela coûte une division et une élévation au carré dans $\mathbb{F}_{2^m}$.

Soit i un élément de $\mathbb{F}_{2^m}$ et f_i l'application :

$$\begin{aligned} f_i : \ \mathbb{F}_{2^m} &\to \mathbb{F}_{2^m} \\ x &\mapsto x^2 + x + i \end{aligned}$$

L'équation $f_i(x) = 0$ a deux solutions dans $\mathbb{F}_{2^m}$ si et seulement si $\mathrm{Tr}(i) = 0$ (une preuve est donnée dans [BRS67]), sinon cette équation n'a pas de solution dans $\mathbb{F}_{2^m}$.

Soit T une table contenant les éléments de $\mathbb{F}_{2^m}$.

$$T[i] = \begin{cases} j & \text{if } j^2 + j = i \\ \varnothing & \text{if } \mathrm{Tr}(i) = 1 \end{cases}$$

En d'autres termes, $T[i]$ contient un des deux éléments du noyau de f_i, si i est dans l'image de $x \mapsto x^2 + x$. Notez que $j + 1$ est le second élément de ce noyau.

Maintenant, nous lisons l'élément $T[b/a^2]$ à partir la table T. Nous inversons le changement de variable en calculant : $z_1 = ab$. Puis, nous calculons la seconde solution : $z_2 = ab + a$. Cette procédure coûte une multiplication et une addition dans $\mathbb{F}_{2^m}$. Ainsi, nous avons résolu

notre problème en moins de quatre opérations dans $\mathbb{F}_{2^m}$ avec une mémoire de 2^m éléments du corps.

Compromis Temps-Mémoire pour le Degré 3.

Dans ce cas, l'équation à résoudre est : $z^3 + az^2 + bz + c = 0$ où a, b, $c \in \mathbb{F}_{2^m}$. Nous obtenons un multiple affine de degré 4 en multipliant le polynôme par $z + a$. Cela coûte deux multiplications, une élévation au carré et deux additions dans $\mathbb{F}_{2^m}$. La substitution $z = \sqrt{(a^2 + b)}x$ ($m-1$ élévations au carré répétées, avec la méthode qui n'utilise pas de mise en table [2]) et une normalisation (deux divisions) permet d'obtenir une équation de la forme : $x^4 + x^2 + dx = e$, avec d, $e \in \mathbb{F}_{2^m}$. Notons f l'application $x \mapsto x^4 + x^2 + dx$. Par construction a est une racine de $f(x) = e$, nous voulons trouver les trois autres. L'application f a un noyau de dimension deux (sauf si $d = 0$, dans ce cas, la dimension est un). Nous avons seulement stocké deux éléments qui forment une base du noyau de $x \mapsto x^4 + x^2 + dx$ dans une table pour tout $d \in \mathbb{F}_{2^m}$. Cela requiert de stocker 2×2^m éléments du corps. Remarquez que, cette phase ne dépend pas du coefficient e. Appelons la table de correspondance T et les deux éléments stockés dans la table pour un d donné : λ_1 et λ_2. Par suite, nous avons $T[d] = (\lambda_1, \lambda_2)$. Comme f est linéaire, les trois autres racines de f sont : $a + \lambda_1$, $a + \lambda_2$ et $a + \lambda_1 + \lambda_2$. Nous les obtenons avec trois additions. Enfin, nous inversons la substitution (trois multiplications) et ainsi le problème est résolu. Ici, nous avons utilisé le fait de savoir que a est une racine de f pour trouver les autres racines. Nous ne pouvons pas utiliser davantage cette information supplémentaire pour des polynômes de degré supérieur ou égal à 4.

6.5.4 Comment calculer la complexité théorique ?

Nous n'allons pas donner ici la formule de récurrence de la complexité, que nous utilisons pour calculer le nombre d'opérations requises pour exécuter BTZ dans un souci de clarté. Au lieu de cela, nous préférons expliquer comment nous l'obtenons.

À propos de BTA.

Les polynômes avec lesquels nous travaillons dans BTA sont unitaires, sans racine multiple et peuvent être factorisés en facteurs linéaires sur $\mathbb{F}_{2^m}$. Ces polynômes forment un ensemble $\mathcal{P}$. De tels polynômes de degré d sont entièrement déterminés par leurs d racines. Donc, il existe $\binom{2^m}{d}$ polynômes de la sorte.

De plus, nous savons que pour tout $\beta \in B$, une $\mathbb{F}_2$-base de $\mathbb{F}_{2^m}$, la moitié des éléments de $\mathbb{F}_{2^m}$ ont $\mathrm{Tr}(\beta \cdot z) = 0$ et que pour l'autre moitié, $\mathrm{Tr}(\beta \cdot z) = 1$.

Pour chaque phase de l'Algorithme 3 (voir la Sous-section 6.5.2), nous calculons le pgcd d'un polynôme avec $\mathrm{Tr}(\beta \cdot z)$. Le polynôme que nous obtenons contient les racines, telles que

2. Mentionnons qu'une addition et une multiplication avec une constante, sont suffisantes en utilisant la méthode proposée dans [Hub02]. Étant donné que nous considérons des extensions de taille petite ou moyenne, les données sont à mettre en table pour la mise en œuvre. Nous ne prenons pas cela en compte dans la Table 6.1.

	Addition	Mult.	Division	Carré	Inv. Matrice	Coût Total
Z_2 précalcul	m^2	m^2	0	$m(m-1)$	1	$4m^2-m$
Z_2 sans compromis	m	1	1	1	0	$m+3$
Z_2 avec compromis	1	1	1	1	0	4
Z_3 sans compromis	$2(m+1)$	$3m$	0	m	1	m^2+6m+2
Z_3 avec compromis	5	5	2	m	0	$m+12$
Z_4	$2m+9$	$3(m+1)$	8	m	1	$m^2+6m+20$
Z_5	$4m+101$	$7m+104$	1	0	1	$m^2+11m+206$

TABLE 6.1 – Nombre d'opérations dans $\mathbb{F}_{2^m}$ dans les procédures de Zinoviev

$\mathrm{Tr}(\beta \cdot z) = 0$. Nous faisons ensuite une division euclidienne qui nous donne un autre polynôme de degré $d-i$, qui contient les racines, telles que $\mathrm{Tr}(\beta \cdot z) = 1$. Quand nous calculons la complexité théorique de BTZ, nous calculons l'espérance du nombre d'opérations dans le pire des cas. Donc, nous considérons, pour tout $i \in \{0, \ldots, d\}$, la probabilité $P(d, m, i)$ que le polynôme de degré d se factorise en un couple de polynômes de degré i et $d-i$. C'est-à-dire que le calcul du pgcd nous donne un polynôme de degré i dont les racines sont parmi les éléments de $\mathbb{F}_{2^m}$, tels que $\mathrm{Tr}(\beta \cdot z) = 0$. De la même manière, avec la division euclidienne, nous obtenons un polynôme de degré $d-i$, dont les racines sont parmi les éléments de $\mathbb{F}_{2^m}$, qui vérifient $\mathrm{Tr}(\beta \cdot z) = 1$. Nous supposons que le polynôme donné est choisi aléatoirement à partir d'une distribution uniforme sur $\mathcal{P}$. Ainsi, nous obtenons :

$$P(d, m, i) = \frac{\binom{2^{m-1}}{i} \times \binom{2^{m-1}}{d-i}}{\binom{2^m}{d}}.$$

A propos des procédures de Zinoviev.

Dans la Table 6.1, nous supposons que toutes les opérations arithmétiques sur le corps ont un coût unitaire et qu'une inversion d'une matrice binaire d'ordre m coûte m^2 additions dans $\mathbb{F}_{2^m}$. Notons Z_d la procédure de Zinoviev pour le degré d où d varie entre 2 et 5. Pour $d = 2$ et $d = 3$, nous considérons deux possibilités : avec ou sans les compromis temps-mémoire. Quand nous utilisons les compromis, la complexité en espace est exponentielle en m, c'est-à-dire, en la taille du corps *i.e.* 2^m, à une constante multiplicative près. Pour des degrés supérieurs ($6 \leq d \leq 10$), la complexité en temps est $\mathcal{O}(m^2+dm+d2^d)$. Elle est exponentielle en d puisque dans le pire des cas, le multiple affine a un degré égal à 2^{d-1}. En conséquence, dans la dernière étape des procédures de Zinoviev, nous aurions évalué le polynôme dans 2^{d-1} points pour trouver ses racines. C'est une raison pour laquelle nous n'utilisons pas les méthodes de Zinoviev pour des degrés supérieurs à 10. On observe, dans la Table 6.1, que la partie la plus coûteuse dans les procédures de Zinoviev est l'inversion de matrice binaire qui permet de trouver les racines du polynôme affine (voir la Section 6.4 pour plus de détails).

6.6 Résultats de simulation

Dans la Table 6.2, nous présentons des données expérimentales pour trouver les racines d'un polynôme de degré $t = 32$ sur $\mathbb{F}_{2^m} = \mathbb{F}_{2048}$. La notation BTZ_d signifie que nous utilisons BTZ avec $d_{max} = d$, pour toutes les valeurs convenables de d.

$n = 2048, \quad t = 32, m = 11$		Chien	BTA	BTZ_2	BTZ_3	BTZ_4
# cycles CPU	trouver les racines	3200	1300	900	800	800
par octet pour	déchiffrer	3700	1800	1400	1300	1300
pourcentage [3]	le calcul du syndrome	5	10	13	14	14
de temps	résoudre l'équation clé	7	14	18	19	19
passé pour	trouver les racines	87	72	65	61	60

TABLE 6.2 – Temps de calcul pour trouver les racines d'un polynôme de degré $t = 32$ sur $\mathbb{F}_{2^m} = \mathbb{F}_{2048}$.

Dans la Table 6.3, nous présentons le nombre théorique d'opérations dans le corps pour corriger $t = 32$ erreurs d'un mot de longueur $n = 2048$ sur $\mathbb{F}_{2^{11}}$ en utilisant BTZ avec $d_{max} = 5$ et les compromis temps-mémoire (voir la Sous-section 6.5.3). Pour information, BTZ sans les compromis donne des résultats *théoriques* très proches.

Illustrons le gain de BTZ sur BTA et la procédure de Chien en termes de pourcentage de nombre d'opérations en fonction du paramètre d_{max} et du degré du polynôme t. Les résultats (donnés dans la Figure 6.1 et la Figure 6.2) prennent en compte les compromis temps-mémoire pour les degrés 2 et 3. Dans les applications cryptographiques, nous nous restreignons au cas $m = 11$, $K(\times) = m$, $t \leq 100$ et $d_{max} \leq 6$. Mais, nous présentons aussi ce type de résultats similaires pour des degrés plus élevés dans la Figure 6.3 ainsi que pour $m = 15$ dans la Figure 6.4 et $m = 40$ dans la Figure 6.5. Néanmoins, nous avons également calculé ces résultats pour $m = 8, 11, 12, 13, 14, 15, 16, 20, 30, 40$, $K(\times) = 1$, $t \leq 300$, $d_{max} \leq 10$ et à la fois *avec* et *sans* les compromis. Ces données révèlent que plus t est élevé, plus le d_{max} optimal est élevé. On peut déduire à partir des deux graphes suivants que pour $m = 11$ et $t = 32$, la valeur théorique recommandée pour d_{max} est 5. En effet, nous obtenons un gain substantiel de 46% sur BTA et 93% sur la méthode de Chien pour cette valeur de d_{max}.

3. Pourcentage restant correspondant aux autres tâches mineures.

$n = 2048, t = 32, m = 11$	Chien	BTA	BTZ_2	BTZ_3	BTZ_4	BTZ_5	BTZ_6
$K(+) = K(\times) = 1$	129k	16k	13k	11k	10k	10k	10k
$K(+) = 1, K(\times) = m$	764k	91k	65k	54k	50k	47k	48k

TABLE 6.3 – Nombre théorique d'opérations dans le corps pour corriger $t = 32$ erreurs d'un mot de longueur $n = 2048$ sur $\mathbb{F}_{2^{11}}$ en utilisant BTZ avec $d_{max} = 5$ et les compromis temps-mémoire.

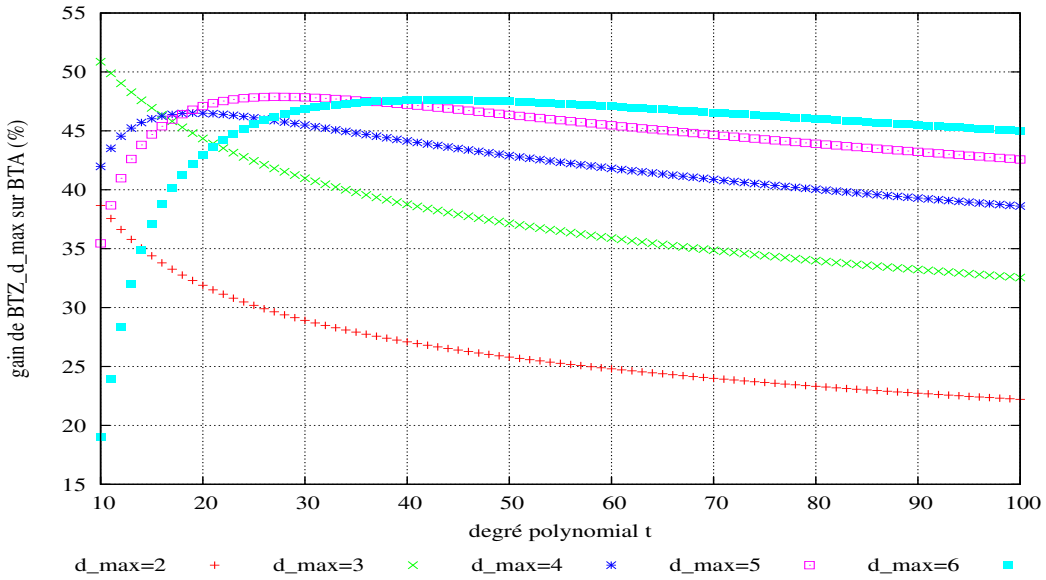

FIGURE 6.1 – $\mathrm{BTZ}_{d_{max}}$ vs. BTA ; $m = 11$; $K(+) = 1$; $K(\times) = m$
version avec les compromis temps-mémoire

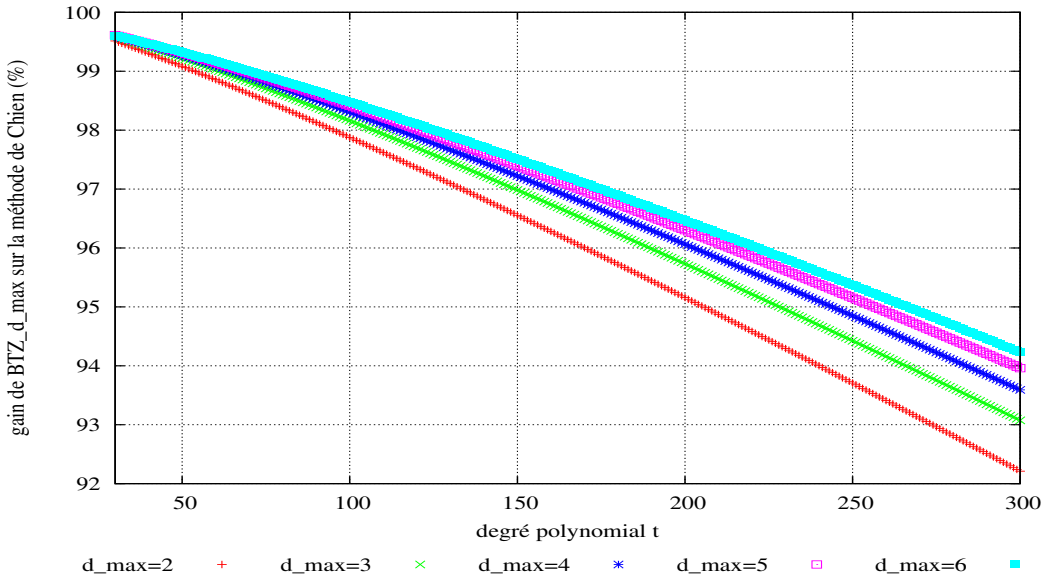

FIGURE 6.2 – $\mathrm{BTZ}_{d_{max}}$ vs. Chien ; $m = 11$; $K(+) = 1$; $K(\times) = m$
version avec les compromis temps-mémoire

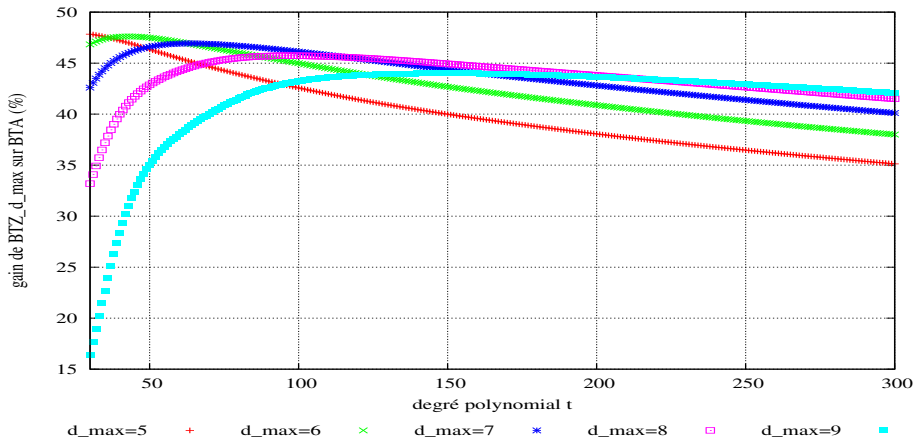

FIGURE 6.3 – $\mathrm{BTZ}_{d_{max}}$ vs. BTA ; $m = 11$; $K(+) = 1$; $K(\times) = m$
version avec les compromis temps-mémoire

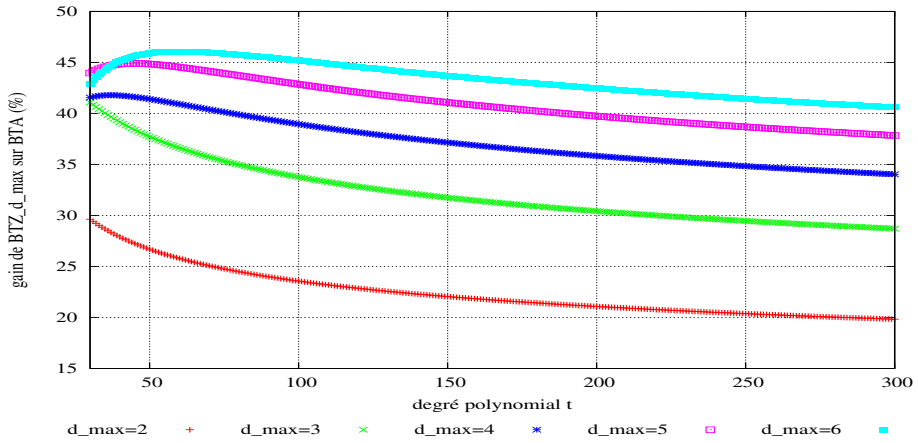

FIGURE 6.4 – $\mathrm{BTZ}_{d_{max}}$ vs. BTA ; $m = 15$; $K(+) = 1$; $K(\times) = m$
version avec les compromis temps-mémoire

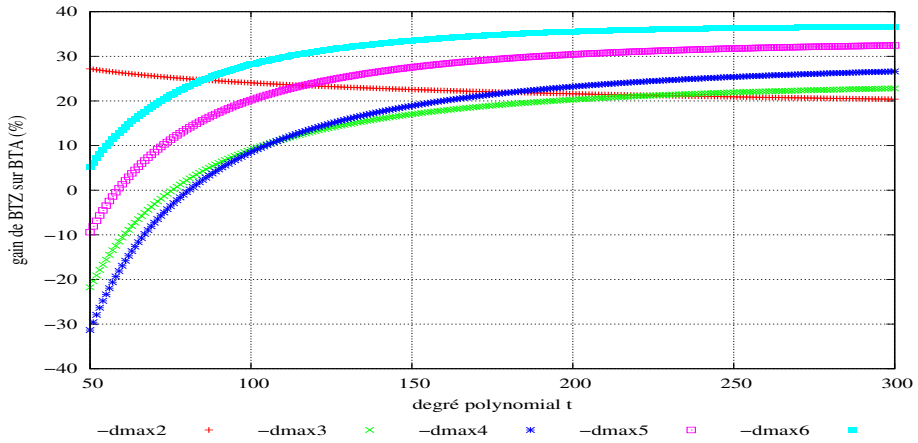

FIGURE 6.5 – $\mathrm{BTZ}_{d_{max}}$ vs. BTA ; $m = 40$; $K(+) = 1$; $K(\times) = m$
version sans les compromis temps-mémoire

Troisième partie

L'authentification et les codes linéaires

7 L'authentification RFID et le protocole HB

7.1 La technologie RFID

L'identification par radiofréquence (RFID = Radio Frequence IDentification) est une technique qui permet d'identifier à distance des objets ou des personnes munis d'un transpondeur (micro-circuit + antenne), appelé tag.

Par extension, la RFID permet aussi de récupérer des données stockées sur le tag, éventuellement récupérer le résultat d'un calcul réalisé par le tag.

De nos jours, l'identification et le suivi d'objets se développent de plus en plus. Au départ, les codes barres permettaient cette identification mais sans permettre le stockage de certaines données. La RFID est devenue par ce biais une technologie incontournable.

Le tag RFID (également connu sous les noms de : transpondeur, étiquette RFID ou encore marqueur), est composé d'une puce électronique (en anglais « chip ») reliée à une antenne bobinée ou imprimée, encapsulées dans un support. Elle est lue par un lecteur qui capte et transmet l'information. Les étiquettes RFID sont des dispositifs matériels de très petite taille. Il existe deux familles : les étiquettes actives, c'est-à-dire des étiquettes qui ont une batterie, et les étiquettes passives auxquelles l'énergie est fournie par le lecteur. Cette différence va jouer sur la puissance de calcul et sur la puissance d'émission mais dans les deux cas la fonction des étiquettes est de fournir une information d'identification à distance.

7.1.1 Le principe de fonctionnement d'une étiquette RFID

Un lecteur et/ou une antenne envoie un signal radio à une fréquence déterminée. Ce signal, capté par l'antenne de l'étiquette, permet d'accéder aux informations contenues dans la puce électronique de l'étiquette radiofréquence. Pour les étiquettes passives, l'énergie contenue dans le signal de départ permet de « renvoyer » les informations vers le lecteur d'étiquettes. Le lecteur peut être fixe ou mobile, et son antenne peut prendre plusieurs formes, et par exemple s'intégrer dans le cadre d'une porte, pour une application de contrôle d'accès. Le lecteur ou interrogateur transmet un signal selon une fréquence donnée vers une ou plusieurs étiquettes radio situées dans son champ de lecture.

Celles-ci transmettent un signal en retour. Lorsque les étiquettes sont « éveillées » par le lecteur, un dialogue s'établit selon un protocole de communication prédéfini, et les don-

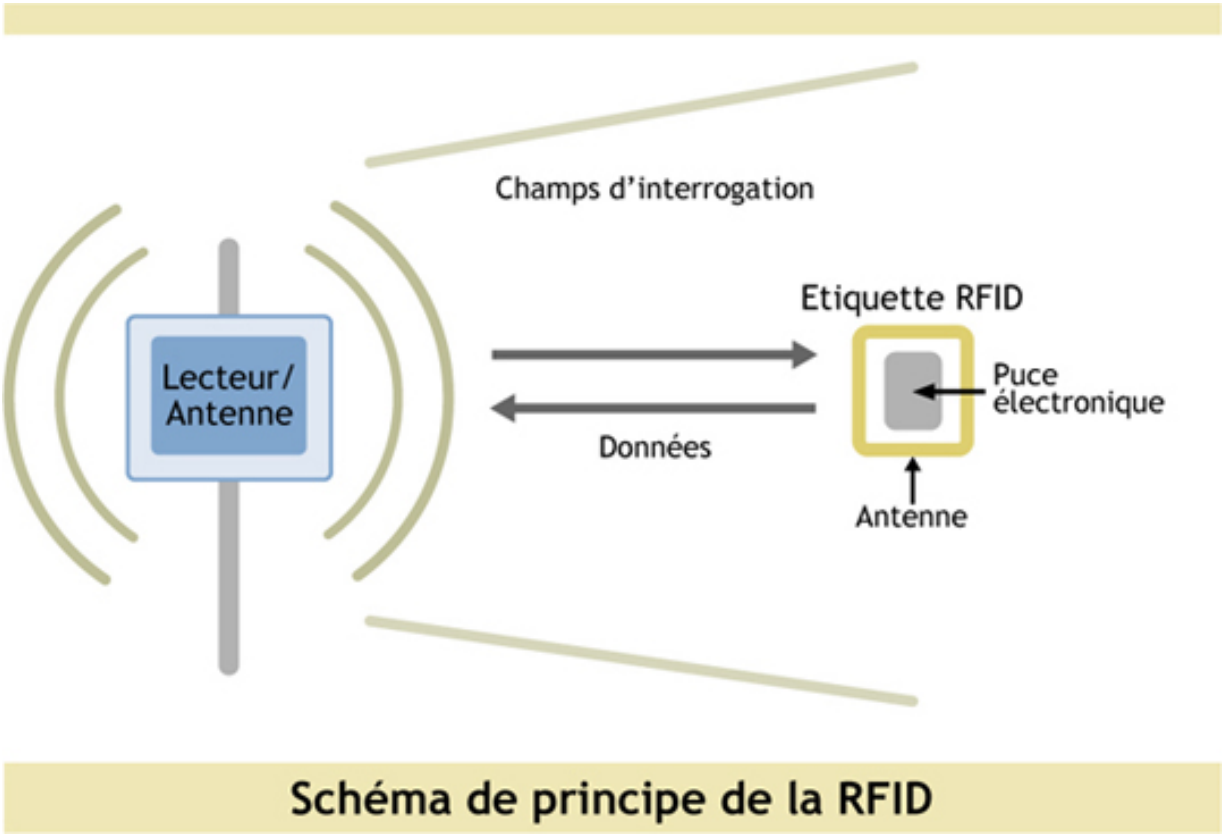

FIGURE 7.1 – Fonctionnement d'une RFID

nées sont échangées. L'étiquette peut être apposée, portée, insérée dans un objet (colis, carte, véhicule...).

Caractéristiques des RFID à bas-coût

- Énergie : reçue du lecteur (tags passifs) ;
- Communication : quelques centimètres à quelques mètres ;
- Calculs : simple mémoire jusqu'à microprocesseur ;
- Mémoire : centaine de bits jusqu'à plusieurs kilo-octets ;
- Prix : dizaine de centimes jusqu'à quelques euros.

Les particularités de la RFID

- Les tags ne peuvent être éteints.
- Les tags répondent au lecteur sans l'accord de l'utilisateur.
- La tendance est à augmenter la distance de communication.
- Les tags sont parfois invisibles ou leur présence ignorée.

FIGURE 7.2 – La RFID au quotidien

7.1.2 Les applications de la RFID

Les RFID trouvent leur emploi pour l'accès aux transports publics, le suivi industriel en chaîne de montage, les inventaires, la gestion des stocks, la logistique, etc. Le marché mondial des systèmes d'identification par radiofréquence (2,7 milliards de dollars en 2006) devrait atteindre 12 milliards de dollars en 2010 et 26 milliards de dollars en 2016, se répartissant entre 43% pour l'Amérique du Nord, 33% pour la zone Europe - Moyen Orient - Afrique, 21% pour l'Asie Pacifique, 3% pour l'Amérique Centrale et latine. La réduction du prix des étiquettes et des lecteurs RFID devrait stimuler ce marché.

Outre les enjeux stratégiques (normalisation) et techniques liés aux fréquences utilisées, le principal frein au développement de la technologie reste le retour sur investissement, relativement long, et le prix des étiquettes, qui malgré une baisse de 50% ces dernières années, se situe encore entre 0,10 et 0,20 euros. Les principales applications se trouvent dans la sécurité, le contrôle d'accès, le transport et la logistique.

Exemples d'applications de la RFID :

Identification : Obtenir l'identité du tag.
- suivi d'objets (le RFID va remplacer les codes barres) ;
- tatouage animal ;
- marquage du bétail ;
- bibliothèque ;
- inventaire ;

- traçabilité dans les chaînes de production ;
- marquage du linge dans les blanchisseries.

Authentification : Obtenir une preuve de l'identité du tag.

- passeport ;
- abonnement aux transports publics ;
- badge d'accès ;
- carte de fidélité ;
- forfait pour les remontées mécaniques ;
- télépéage ;
- clef de démarrage des voitures.

Pour anecdote, elle est également utilisée dans certains hopitaux et prisons pour retrouver les malades d'Alzheimer et tracer les libérés sur parole. On imagine aisément que certaines applications posent des problèmes d'éthique et que certaines dérives sont envisageables.

7.1.3 Les problématiques de sécurité liées à la RFID

Quels sont les objectifs du cryptologue ?

1. L'Authenticité (l'origine ou la personne doivent être reconnues) ;
2. La Confidentialité (les informations ne doivent pas être divulguées) ;
3. L'Intégrité (les informations doivent restées intactes).

Quels sont les objectifs de l'adversaire ?

1. Usurpation d'identité ;
2. Fuite d'information ;
3. Traçabilité malveillante ;
4. Déni de service (disponibilité).

Les usages faits de la RFID représentent une très grande menace potentielle pour l'intimité numérique des consommateurs. Étant donné qu'il n'y a pas besoin de contact ni de ligne de vue directe, il est très aisé d'écouter avec une antenne les étiquettes que peuvent avoir sur elle une personne donnée en cas d'absence de mécanismes de sécurité. Les autres risques pour la vie privée et la liberté sont les suivants :

- possibilité d'atteinte à la vie privée dans le cas de marqueurs furtifs ;
- discrimination par lecture à distance du passeport ;
- « marquage » abusif de personnes ayant consulté certains types d'informations dans les fichiers d'employeurs potentiels ou d'un état répressif ;
- clonage et contrefaçon.

Ils représentent aussi une menace pour la sécurité des entreprises car ils seront au coeur du contrôle d'accès aux services mais aussi au coeur de la production et de la chaîne logistique. Face à ces périls, des mesures sécuritaires doivent être mises en place. Nous allons étudier dans la Section 7.2, une solution cryptographique à certains des problèmes évoqués précédemment.

7.1.4 Les solutions de sécurité

Le coût d'un tag RFID joue un rôle important, de lui dépend la force des mécanismes de sécurité que l'on emploiera. Un tag RFID bas-coût contient approximativement 5K-10K portes logiques et au mieux, seulement 2K-3K portes peuvent être consacrées aux fonctions de sécurité alors qu'un algorithme de chiffrement par blocs nécessite 5K-10K blocs. On ne doit pas compter sur la Loi de Moore (« la puissance de calcul par unité de coût augmente dans le temps ») pour répondre à cette problématique car il existe une pression pour que les prix soient les plus bas possibles. D'après [Chi07], nous pouvons classer les protocoles d'authentification RFID en quatre classes selon des critères de puissances de calcul et des opérations réalisables par le tag.

1. La classe « full-fledged » (on pourrait traduire cela par « complètement développée ») qui se réfère aux protocoles qui permettent l'utilisation de fonctions cryptographiques traditionnelles (chiffrement symétrique, fonction à sens unique ou encore algorithmes à clé publique) Une des principales applications de cette classe de protocole est le e-Passeport.
2. La classe « simple » est pour les protocoles qui supportent l'utilisation de générateur de nombres pseudo-aléatoire (PRNG) et de fonctions de hachage à sens unique sur les tags.
3. La classe « lightweight » qui correspond aux protocoles pouvant utiliser des PRNG et des fonctions simples comme des sommes de contrôle Cyclic Redundancy Code (CRC) mais pas de fonctions de hachage.
4. La classe « ultra-lightweight » qui implique seulement des opérations logiques bit à bit sur le tag.

Il existe aujourd'hui de nombreux articles de recherche traitant de la sécurité RFID. Pour de plus amples détails, nous vous invitons à consulter [Jue06]. Pour notre part, nous nous intéresserons dans la suite à une famille de protocoles d'authentification qui utilisent essentiellement des opérations logiques et des PRGB (Pseudo Random Bit Generator) mais dont certaines variantes requièrent aussi l'emploi de fonctions de hachage Il s'agit de la famille des protocoles HB. Elle présente encore l'avantage de disposer de preuves de sécurité dans différents modèles.

Nous ne traiterons pas ici de tous les protocoles légers d'authentification existants. Néanmoins, nous pouvons en parler brièvement. Dans [VB03], Vajda et al. proposent un protocole qui utilise un chiffrement XOR avec clefs secrètes. Il existe des attaques pour un adversaire puissant. Dans [Jue04], Juels a proposé une solution basée sur les pseudonymes qui n'utilisent pas les fonctions de hachage. Les tags RFID stockent une liste courte d'identifiants aléatoires ou de pseudonymes (connus par les vérifieurs autorisés). Quand le tag est interrogé, il émet le prochain password dans la liste. Cependant, la liste des pseudonymes doit

être réutilisée ou mise à jour via un canal externe après un certains nombre d'authentifications.

Plus récemment, Lopez et al. ont proposé une famille de protocoles d'authentification mutuelle pour la RFID bas-coût : LMAP [PLHCETR06a], M^2AP [PLHCETR06b] et EMAP [PLHCETR06c] dans lesquels de simples opérations bit à bit (environ 300 portes logiques) et des additions modulo 2^m sont utilisées. Cependant, il existe des attaques passives ou actives qui permettent à l'attaquant de retrouver le secret sur le tag. Un des derniers protocoles prometteurs (selon ses auteurs) est sorti en 2007, il s'agit de SASI (Strong Authentication, Strong Integrity) [Chi07]. Au final, l'expérience montre que quasiment tous ces protocoles légers sont vulnérables d'une manière ou d'une autre. Un déploiement pratique de ces derniers est donc pour l'heure risqué tant qu'une analyse de sécurité poussée ne nous aura pas offert les garanties recherchées.

7.2 Les variantes de HB

7.2.1 Le protocole HB

Le protocole initial conçu par Hopper et Blum en 2001 est à l'origine un protocole qui permet d'authentifier un humain sur un ordinateur. Il a été adapté à l'authentification unilatérale du tag dans les RFID par Juels et Weis en 2005 du fait des similarités entre humains et tags au niveau puissance de calcul et mémoire.

Il y a deux entités : le lecteur et le tag. Le but est pour le tag de prouver son identité au lecteur. Le tag et le lecteur partagent un secret x qui est une chaîne de k bits. Le lecteur génère une chaîne de k bits de manière pseudo aléatoire, on l'appellera a ici. Il s'agit d'un challenge qui est envoyé au tag. Le tag calcule alors le produit scalaire du challenge a et du secret x auquel est rajouté un bit ν que l'on peut voir comme une variable aléatoire suivant une loi de Bernoulli de paramètre η.

Les paramètres essentiels de HB

- x : le vecteur secret partagé par le lecteur et le tag ;
- k : la taille de la clé x ;
- η : le niveau de bruit. $\eta \in]0, \frac{1}{2}[$;
- r : le nombre de rondes (ce paramètre dépend de η) ;
- t : le seuil d'acceptation.

Remarque 39. Usuellement, on choisit $\eta = \dfrac{1}{4}$ ou $\eta = \dfrac{1}{8}$.

Le protocole HB : Hopper & Blum (2001)

- Données publiques : k, r, η, $u \in]\eta, \frac{1}{2}[$
- Secret partagé : $x \in \{0,1\}^k$

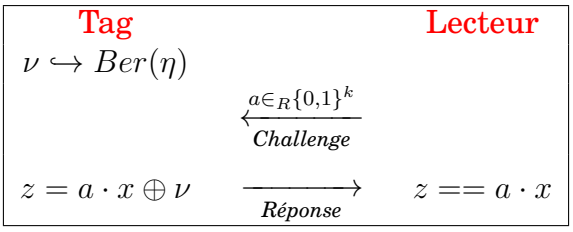

FIGURE 7.3 – Le protocole HB : Hopper & Blum (2001)

Ce protocole de type Challenge-Réponse se déroule en r rondes. Le tag est authentifié si le nombre d'erreurs est $\leq t = u \times r$. On choisit un seuil plus grand que $(\eta \times q)$ car en pratique un tel seuil provoquerait un taux de faux rejet (fausse alarme) beaucoup trop important. Il ne faut bien sur pas pour autant tomber dans l'excès inverse sous peine d'obtenir un taux de fausse acceptation (non détection) trop fort.

La sécurité de HB repose sur la difficulté du problème LPN. Lors de l'exécution du protocole $O(k^2)$ bits sont transmis. Plus précisément, le lecteur émet $k \times r$ bits et le tag en émet r.

D'après [LF06], le nombre nécessaire de bits envoyés pour réaliser l'authentification de manière fiable est celui indiqué dans la Table 7.1. Ces chiffres expriment le fait que pour certains paramètres de HB, l'authentification peut prendre plusieurs secondes même pour un tag performant. Le taux de transmission varie usuellement entre 20Ko/s et 40Ko/s. Certains d'entre eux sont 10 fois plus lents, cela dépend comme toujours des applications. Cela signifie encore que la taille des paramètres (taille du secret, nombre de rondes, niveau de bruit) doit être ajustée. On ne peut pas indéfiniment augmenter la taille de ces derniers sous peine d'avoir des protocoles inintéressants pour un usage pratique.

taille du secret \ niveau de bruit	$\frac{1}{20}$	$\frac{1}{8}$	$\frac{1}{4}$
128	4	7	18
512	16	28	73

TABLE 7.1 – Nombre nécessaire de bits pour une authentification fiable avec le protocole HB. L'unité utilisée pour quantifier les données est le kilo-octet.

Le problème LPN = Learning from Parity with Noise

Rappel . La parité d'une chaîne de bits vaut 1 si elle contient un nombre de bits impair, 0 sinon. Autrement dit, il s'agit de la somme modulo 2 des bits qui la composent.

Il existe plusieurs formulations du problème LPN [1]. Nous en donnerons deux qui sont équivalentes. Une troisième est disponible dans [KS06], elle utilise une approche différente avec les notions d'oracle, d'algorithmes PPT (probabilistiques en temps polynomial) et de

1. LPN se retrouve parfois sous l'appellation de : *Minimum Disagreement Problem*.

probabilité de succès que l'on retrouve en sécurité prouvée mais est identique dans le fond à la première définition. Pour anecdote, LPN est originellement un problème étudié en Machine Learning (en français, on parle d'apprentissage automatique), il s'agit d'un domaine d'étude important en Intelligence Artificielle.

Définition 62. (Le problème LPN - première formulation de [DK07]) Soit A une matrice binaire $(k \times q)$, x un vecteur aléatoire de k bits, η le niveau de bruit et $\nu = (\nu_1, \nu_2, .., \nu_q)$ un vecteur de bruit de q bits tel que $\nu_i \hookrightarrow Ber(\eta)$ i.i.d. Étant donnés A, η, et $z = xA \oplus \nu$, trouvez un vecteur x tel que $d(z, xA) \leq \eta \times q$.

Définition 63. (Le problème LPN - seconde formulation de [JW05]) Soit A une matrice binaire $(k \times q)$, x un vecteur aléatoire de k bits, η le niveau de bruit et ν un vecteur aléatoire de q bits tel que $w(\nu) \leq \eta \times q$. Étant donnés A, η, et $z = xA \oplus \nu$, trouvez un vecteur y de k bits tel que $w(yA \oplus z) \leq \eta \times q$.

- LPN est un problème NP-complet (pire des cas) ;
- LPN est conjecturé difficile (cas moyen).
- η est une constante fixée dans $]0, \frac{1}{2}[$ et indépendante de k.

Intuitivement, on peut aussi voir le problème LPN comme étant le suivant : Étant donné un ensemble d'équations sur $\mathbb{F}_2$, trouver un vecteur qui vérifie un maximum d'équations. Il a d'ailleurs été démontré que ce problème est difficile à approximer avec un facteur de 2 [Has97], *i.e.* il est difficile de trouver un vecteur qui vérifie la moitié du nombre optimal d'équations.

La meilleure attaque connue sur LPN demande un nombre « exponentiel » d'échantillons et de calculs pour être réalisée. Cela rend cette attaque impraticable dans les faits puisqu'il est impossible de collecter autant d'informations et d'observer un nombre exponentiel d'exécutions du protocole.

Attaque sur HB

1. HB résiste aux attaques passives ;
2. HB sensible aux attaques actives.

- On envoie un challenge a plusieurs fois au tag.
- On en déduit la valeur de $a.x$ sachant que $\eta < \frac{1}{2}$.
- On retrouve x par élimination de Gauss en retrouvant k équations avec des a linéairement indépendants.
- La complexité algorithmique asymptotique de l'élimination de Gauss (méthode naïve) est $O(k^3)$.
- On peut descendre à $O(k^{2.38})$ avec l'algorithme de Strassen. La borne inférieure triviale sur la complexité pour calucler x est $O(k^2)$. L'intérêt d'utiliser l'algorithme de Strassen est limité dans notre contexte puisque la valeur k est petite (seulement quelques centaines de bits).

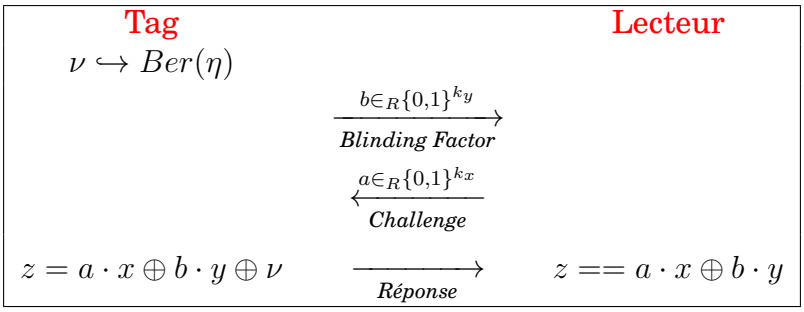

FIGURE 7.4 – Le protocole HB+ : Juels & Weis (2005)

7.2.2 Le protocole HB+

Le protocole HB+ : Juels & Weis (2005)

HB+ se déroule en r rondes de 3 passes.

Il suit un schéma classique de communication :

1. Engagement [2] ;
2. Challenge ;
3. Réponse.

Ce schéma de communication fut conservé par les variantes qui suivirent.

Les paramètres essentiels de HB+

- x, y : les vecteurs secrets partagés par le lecteur et le tag ;
- k_x, k_y : la taille de la clé x (resp. y) ;
- r : le nombre de rondes ;
- η : le niveau de bruit. $\eta \in]0, \frac{1}{2}[$;
- t : le seuil d'acceptation.

Le protocole HB+ : Juels & Weis (2005)

- Données publiques : k_x, k_y, r, η, $u \in]\eta, \frac{1}{2}[$
- Secrets partagés : $x, y \in \{0,1\}^k$

Ce protocole se déroule en r rondes. Le tag réussit à s'authentifier si le nombre d'erreurs commises est inférieure ou égale au paramètre $t = u \times r$.

Autres paramètres de sécurité

Probabilité de fausse acceptation :

$$P[FA](r,t) = \sum_{i=0}^{t-1} \binom{r}{i} \times \left(\frac{1}{2}\right)^r.$$

2. On utilise aussi les termes de *commitment* et de *promesse*.

Probabilité de faux rejet :

$$P[FR](r,t,\eta) = \sum_{i=t}^{r} \binom{r}{i} \times \eta^i \times (1-\eta)^{r-i}.$$

Ces probabilités sont indépendantes de la taille des clefs x et y.

Les modèles de sécurité

Modèle de détection : L'attaquant interagit r fois avec le tag honnête. L'attaquant interagit ensuite avec le lecteur et essaie d'imiter le vrai tag.

On va voir que ce modèle est trop minimaliste. On crée donc un modèle d'adversaire plus puissant tout en restant dans un scénario réaliste.

Modèle GRS MiM : L'attaquant peut écouter toutes les communications entre le lecteur et le tag (y compris la décision finale du lecteur d'accepter ou non le tag) et l'attaquant peut modifier n'importe quel message du lecteur pour le tag durant les r exécutions du protocole. L'attaquant interagit ensuite avec le lecteur et essaie d'imiter le vrai tag.

Sécurité de HB+

1. HB+ résiste aux attaques actives (modèle de détection).
2. HB+ sensible aux attaques Man in the Middle (modèle GRS-MiM).

La sécurité de HB+ a été prouvée dans le modèle de détection par le biais d'une réduction de LPN. Une attaque contre HB+ a été découverte dès 2005 par Gilbert, Robshaw et Seurin (GRS) dans le modèle GRS-MiM.

Attaque sur HB+ : Gilbert, Robshaw & Seurin

- Données publiques : k_x, k_y, r, η, $u \in]\eta, \frac{1}{2}[$
- Secrets partagés : $x, y \in \{0,1\}^k$

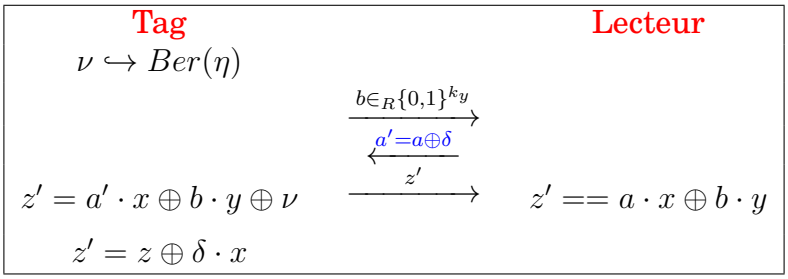

FIGURE 7.5 – Attaque sur HB+

Si $\delta \cdot x = 0$, la décision est inchangée (acceptation probable du tag). Si $\delta \cdot x = 1$, la décision est inversée (échec probable de l'authentification du tag). Cela implique qu'il y ait $1 - \eta$ bit d'information à chaque exécution du protocole.

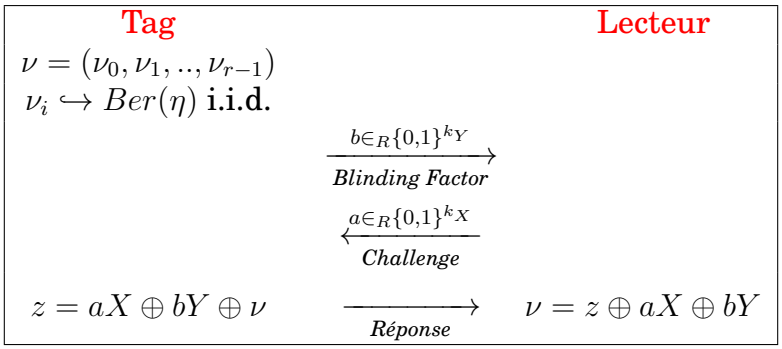

FIGURE 7.6 – Le protocole Random HB# : Gilbert, Robshaw & Seurin (2008)

7.2.3 Le protocole Random HB

Les paramètres essentiels de Random HB#

- X, Y : les matrices binaires aléatoires secrètes partagées par le lecteur et le tag ;
- k_X, k_Y : le nombre de lignes de X (resp. Y) ;
- m : le nombre de colonnes de X et Y ;
- η : le niveau de bruit. $\eta \in]0, \frac{1}{2}[$;
- t : le seuil d'acceptation.

Le protocole Random HB# : Gilbert, Robshaw & Seurin (2008)

- Données publiques : k_X, k_Y, m, η, $u \in]\eta, \frac{1}{2}[$
- Secrets partagés : $X \in \{0,1\}^{k_X \times m}$, $Y \in \{0,1\}^{k_Y \times m}$

Protocole en 1 ronde. Le tag est authentifié si $w(\nu) \leq t = u \times m$.

7.2.4 Le protocole HB#

Dans Random HB#, le tag doit stocker deux matrices binaires aléatoires $(k_X \times m)$ et $(k_Y \times m)$ où k_X, k_Y et m sont des nombres à trois chiffres. Le coût de stockage est trop important dans un contexte RFID. Gilbert, Robshaw et Seurin ont proposé une alternative pratique pour pallier ce problème, il s'agit du protocole HB#. Le protocole est identique à Random HB# si ce n'est que les matrices binaires X et Y ne sont plus aléatoires et possèdent une structure de matrice de Toeplitz.

Une matrice de Toeplitz ou matrice à diagonales constantes est une matrice dont les coefficients sur une diagonale descendant de gauche à droite sont les mêmes. Une matrice $(k \times m)$ de Toeplitz est entièrement déterminée par sa première ligne et sa première colonne. Il suffit donc de stocker $k + m - 1$ bits au lieu de $k \times m$ pour une matrice complètement aléatoire comme on peut le voir sur l'exemple suivant avec une matrice (5×5) :

$$\begin{pmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \mathbf{6} & 1 & 2 & 3 & 4 \\ \mathbf{7} & 6 & 1 & 2 & 3 \\ \mathbf{8} & 7 & 6 & 1 & 2 \end{pmatrix}$$

Remarque 40. Une matrice circulante de coefficients est une matrice de Toeplitz qui est carrée et qui est complètement spécifiée par sa première ligne de telle sorte chaque ligne de la matrice est obtenue à partir d'un décalage circulaire, d'un nombre fixe de positions, de la ligne précédente.

Problème du coût de stockage des données

Random HB# : bien en théorie mais impraticable.

coût de stockage : $k \times m$ bits.

HB# : bien pour la pratique mais moins sûre.

coût de stockage : $k + m - 1$ bits.

HB# en pratique

k_x	k_y	m	η	t	$P[FR]$	$P[FA]$	Nb de bits transmis	Nb de bits stockés
80	512	1164	0.25	405	2^{-45}	2^{-83}	1756	2918
80	512	441	0.125	113	2^{-45}	2^{-83}	1033	1472

TABLE 7.2 – Jeux de paramètres pour HB#

Remarque 41. Les données secrètes X et Y jouent deux rôles distincts. La sécurité de Y repose sur la difficulté de LPN. 512 bits sont requis pour obtenir 80 bits de sécurité en utilisant les meilleurs algorithmes connus à ce jour pour résoudre LPN [LF06].

Il est sans doute plus facile de comprendre pourquoi en se plaçant dans le contexte HB+ où les matrices n'interviennent pas directement. Le scénario de l'attaque est le suivant : un attaquant actif interagit avec le tag dans un premier temps tente dans un second d'imiter le tag. L'attaquant choisit $a = 0$ et récupère la valeur de $b \cdot y \oplus \nu$. S'il peut résoudre LPN, il récupère y. Il connaît ainsi la moitié du secret et se retrouve dans le scénario de HB. On ne peut pas récupèrer x en procédant de la sorte, le protocole n'étant pas symétrique. On impose que le secret x soit de taille 80 simplement pour empêcher une recherche exhaustive dessus.

Coût de transmission : $k_X + k_Y + m$ bits.

Coût de stockage : $k_X + k_Y + 2 \times m - 2$ bits.

Attaque sur HB# : Ouafi, Overbeck & Vaudenay (2008)

HB# a été conçu pour succéder à HB+ qui est sensible à un certain type d'attaques MiM (cf. 7.2.2). HB# a cependant été démontré sensible à une classe d'attaques plus large MiM [OOV08]. L'attaque se déroule comme suit :

1. On obtient par écoute passive un triplet HB valide $(\overline{a},\overline{b},\overline{z})$. Notre but est de trouver $\overline{w}$, le poids de l'erreur $\overline{\epsilon}$;
2. On construit un oracle qui donne $w(\overline{\epsilon})$;
3. Pour chaque i, on calcule $w(\overline{\epsilon} \oplus 2^i)$;
4. On en déduit $\overline{\epsilon}$ puis $\overline{a}X \oplus \overline{b}Y$;
5. On itère ce procédé pour d'autres triplets et on trouve X et Y.

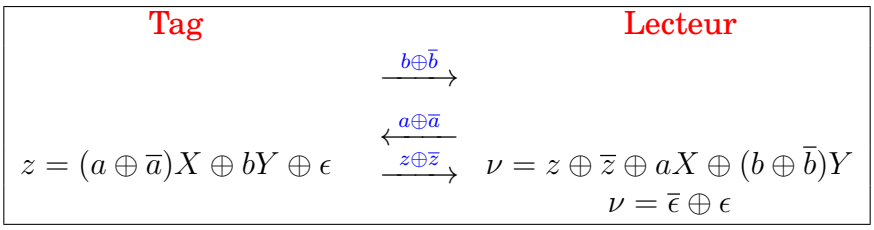

FIGURE 7.7 – Attaque sur HB# : Ouafi, Overbeck & Vaudenay (2008)

Comment trouver $\overline{w}$, le poids de l'erreur $\overline{\epsilon}$? On itère plusieurs fois l'attaque ci-contre. Le taux d'acceptation dépend de $\overline{w}$ puisqu'il ne faut pas que le vecteur $\overline{\epsilon} \oplus \epsilon$ ait un trop grand poids. En effet, pour le lecteur, tout se passe comme si l'erreur était $\epsilon \oplus \overline{\epsilon}$. On déduit $\overline{w}$ en utilisant des statistiques.

k_x	k_y	m	η	t	Random-HB#	HB#
80	512	1164	0.25	405	2^{34}	2^{25}
80	512	441	0.125	113	2^{29}	2^{21}

TABLE 7.3 – Complexité de l'attaque sur HB#

7.3 Étude de l'attaque sur HB#

Nous étudions ici l'attaque Man In The Middle de Overbeck et Vaudenay sur HB#. À la fin de l'exécution du protocole où l'attaquant intervient, le lecteur calcule le poids de $w(\overline{\varepsilon} \oplus \varepsilon)$ au lieu de calculer $w(\varepsilon)$. Le tag sera authentifié si $w(\overline{\varepsilon} \oplus \varepsilon) \leq t$. Le poids de ce vecteur dépend

de $\overline{w}$ le poids de $\overline{\varepsilon}$. On a calculé la probabilité que cet événement se réalise et on a obtenu :

$$\begin{aligned} Pr(w(\overline{\varepsilon} \oplus \varepsilon) \leq t) &= \sum_{i=0}^{\overline{w}} \sum_{j=0}^{i+t-\overline{w}} \binom{\overline{w}}{i} \times \eta^i \times (1-\eta)^{\overline{w}-i} \times \binom{m-\overline{w}}{j} \times \eta^j \times (1-\eta)^{m-\overline{w}-j} \\ &= \sum_{i=0}^{\overline{w}} \sum_{j=0}^{i+t-\overline{w}} \binom{\overline{w}}{i} \times \binom{m-\overline{w}}{j} \times \eta^{i+j} \times (1-\eta)^{m-i-j} \end{aligned}$$

On obtient alors le comportements de $f_{m,t,\eta}(\overline{w}) = Pr_{\varepsilon \hookrightarrow Bin(m,\eta)}(w(\overline{\varepsilon} \oplus \varepsilon) \leq t)$ grâce au logiciel Maple sur les deux jeux de paramètres recommandés pour HB# (à savoir $m = 1164$, $t = 405$, $\eta = 0.25$ et $m = 441$, $t = 113$, $\eta = 0.125$). La Figure 7.3 présente le graphe de cette fonction dans ces deux contextes.

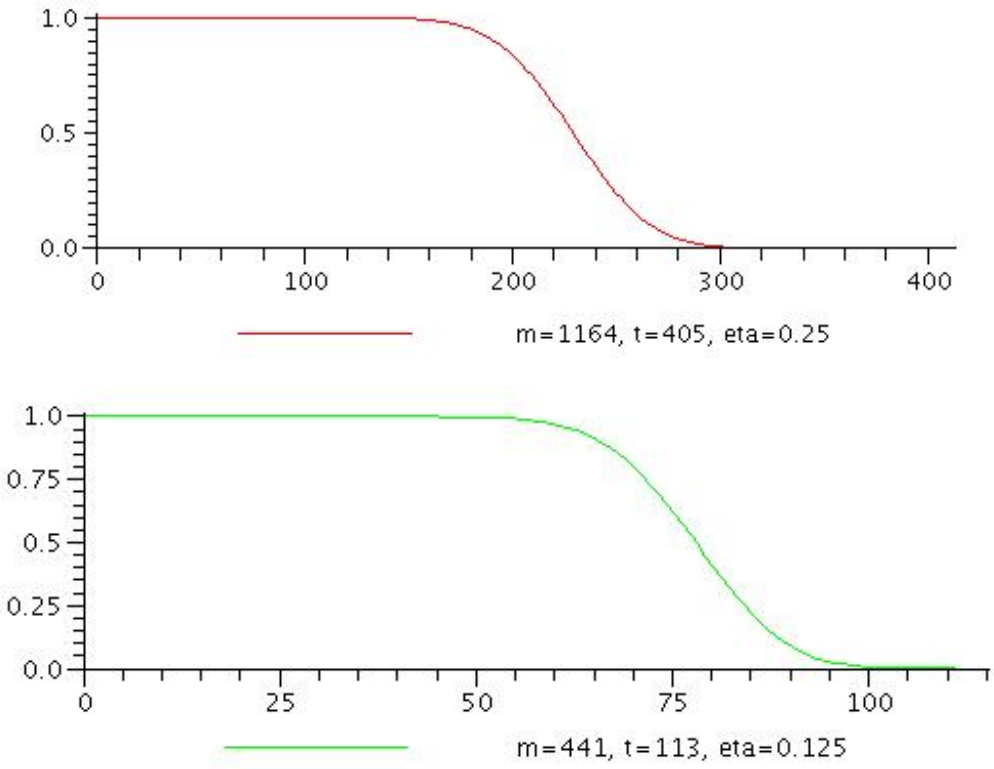

FIGURE 7.8 – Comportement de $f_{m,t,\eta}(\overline{w}) = Pr_{\varepsilon \hookrightarrow Bin(m,\eta)}(w(\overline{\varepsilon} \oplus \varepsilon) \leq t)$ pour les deux jeux de paramètres recommandés pour HB#

L'attaquant aura intérêt à conserver les triplets $(\overline{a}, \overline{b}, \overline{z})$ qui ont un poids dans la partie du graphe où la pente est forte. On rappelle que son but est d'estimer $\overline{\epsilon}$ et que pour cela il calcule le poids de $\overline{\epsilon}$ et des vecteurs qui diffèrent de 1 bit de $\overline{\epsilon}$. Il aura ainsi plus de facilité pour trouver $\overline{\epsilon}$ et ainsi mener l'attaque à bien.

7.3.1 Le protocole Trusted-HB

Le protocole Trusted-HB : Bringer & Chabanne (2008)

Trusted-HB se déroule en $r+1$ rondes.

1. On exécute le protocole HB+ (r rondes).
2. On prouve l'intégrité des communications (1 ronde).

Phase 1 du protocole Trusted-HB

- Données publiques : k_x, k_y, r, η, $u \in]\eta, \frac{1}{2}[$
- Secrets partagés : $x, y \in \{0,1\}^k$

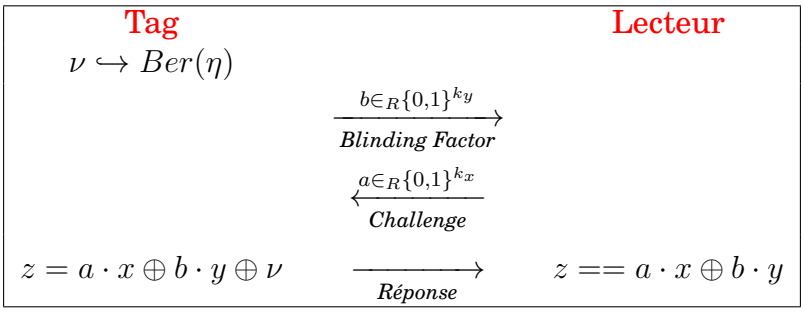

FIGURE 7.9 – Phase 1 du protocole Trusted-HB

Protocole en $r+1$ rondes. On passe à la phase 2 si le nombre d'erreurs est $\leq t = u \times r$.

La propriété suivante établit que chaque message est associé à une empreinte de manière « équilibrée ». Il s'agit d'une propriété essentielle pour assurer la sécurité du schéma [Kra94].

Définition 64. Une famille H de fonctions de hachage est ϵ-équilibrée (ou ϵ-presque universelle) si :

$$\forall x \in \{0,1\}^m, x \neq \overrightarrow{0}, c \in \{0,1\}^n, P(h \in H, h(x) = c) \leq \epsilon$$

Phase 2 du protocole Trusted-HB

- Paramètres publics : k_x, k_y, r, η, $u \in]\eta, \frac{1}{2}[$
- Clefs secrètes : $x, y \in \{0,1\}^k$
- $\nu = (\nu_0, \nu_1, .., \nu_{r-1})$, $\nu_i \hookrightarrow Ber(\eta)$ i.i.d

Soient E une « bonne » fonction pseudo-aléatoire, H une famille ϵ-équilibrée de fonctions de hachage linéaires et $h \in H$.

1. Le tag calcule $e = E(\nu) \in \{0,1\}^n$.
2. Le tag envoie $t = h(a_0, b_0, z_0, ..., a_{r-1}, b_{r-1}, z_{r-1}) \oplus e$.
3. Le lecteur obtient $\nu_i = z_i \oplus a_i \cdot x \oplus b_i \cdot y \; \forall i \in \{0, ..., r-1\}$ et calcule $e = E(\nu) \in \{0,1\}^n$.
4. Le lecteur authentifie le tag si $t = h(a_0, b_0, z_0, ..., a_{r-1}, b_{r-1}, z_{r-1}) \oplus e$.

On pourra utiliser un extracteur d'aléas pour implémenter E (*e.g.* la procédure de Von Neumann qui renvoie une séquence de bits équiprobables et indépendants statistiquement).

Un extracteur d'aléas est une procédure qui convertit de manière efficace une distribution qui contient un peu d'entropie mais qui est aussi biaisée et loin d'être uniforme en une distribution presque uniforme.

Cette notion est à ne pas confondre avec celle de générateur d'aléas. Un générateur d'aléas produit à partir d'une petite source d'aléas une sortie de taille beaucoup plus grande qui paraît toujours aléatoire à un observateur dont la puissance de calcul est bornée.

Les fonctions de hachage et la cryptographie à bas-coût sont-elles bien compatibles ? Revenons à présent sur l'utilisation qui peut paraître surprenante à première vue en cryptographie bas-coût d'une fonction de hachage. L'auteur pense à une construction basée sur les matrices binaires. La construction naïve utilise une matrice binaire aléatoire A $(n \times m)$, le message M de taille m est associé à son empreinte de taille n de la facon suivante : $h_A(M) = AM$. Une meilleure idée est de recourir une nouvelle fois aux matrices de Toeplitz qui, on le rappelle, permettent de stocker $n+m-1$ bits au lieu de $n \times m$. Mais il est encore possible de faire mieux que cela.

En effet, Krawczyk [Kra94] a proposé dès 1994, une construction basée sur les matrices de Toeplitz qui permet de réduire le coût de stockage à $2 \times n$ bits en utilisant un LSFR particulièr de longueur n. Pour être clair, on stocke deux objets de taille n : le polynôme de rétroaction p (qui doit être primitif pour assurer une périodicité optimale) et l'état initial du registre s. On considère donc une matrice de Toeplitz dont les colonnes sont les états successifs du LFSR. Cela devient intuitif si l'on pense que les matrices de Toeplitz sont caractérisées par le fait que chaque colonne est déterminé par un décalage vers le bas de la précédente colonne et par l'ajout d'un nouvel élément en haut de la colonne. Le LFSR change d'état avec chaque bit du message. Si le bit vaut 1, l'état est stocké dans le registre accumuluteur, s'il vaut 0, il est ignoré. Formellement, pour un message $M = M_1 M_2 ... M_{m-1}$, on obtient :

$$h_{p,s} = \bigoplus_{j=0}^{m-1} M_j(s_j, s_{j+1} ..., s_{j+n-1})$$

Son travail est pensé dans le cadre de l'authentification de message avec des fonctions de hachage non cryptographiques *i.e.* pas nécessairement à sens unique. Dans ce cadre la taille du message est bien supérieure à celle de l'empreinte, *i.e.* $m >> n$. On imagine donc les répercussions de cela en termes de stockage de bits. Il a de plus étudié la sécurité de cette construction, elle se révèle sensiblement la même que pour une matrice complètement alétoire mais pour un coût nettement plus bas en aléas, taille de clé et complexité d'implémentation. Les LFSR présentent en plus l'avantage d'être facilement implantables en hardware.

7.3.2 Le protocole HB-MP+

Nous allons parler dans cette section du protocole HB-MP+ qui est une version améliorée du protocole HB-MP [MP07] inventé par Munilla et Peinado (d'où le nom de ce protocole). HB-MP+ date d'avril 2008 et a été crée par [LMM08]. Le protocole HB-MP s'est révélé sensible à une attaque MiM auquelle HB-MP+ résiste. Il utilise des rotations de la clé secrète qui

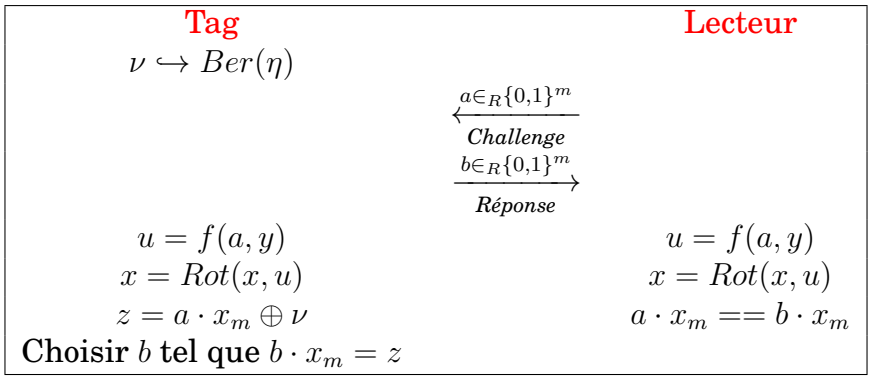

FIGURE 7.10 – Le protocole Improved HB-MP : Leng, Mayes & Markantonakis (2008)

sont prévisibles pour l'attaquant. L'idée dans HB-MP+ est de randomiser ces rotations. Une autre idée est d'utiliser des clés de ronde que l'on obtiendrait par rotation ou de manière plus générale par une fonction à sens unique.

Les paramètres essentiels de Improved HB-MP et de HB-MP+

- x,y : les vecteurs secrets partagés par le lecteur et le tag ;
- k : la taille des clefs x et y ;
- m : la longueur des messages échangés entre les parties ;
- x_m : vecteur binaire de longueur m constitué des bits de poids faibles de x ;
- $Rot(x, u)$: opérateur de décalage circulaire à droite du vecteur x de u positions (spécifique à Improved HB-MP) ;
- x_s : clef de ronde (spécifique à HB-MP+) ;
- f : fonction à sens unique ;
- r : le nombre de rondes ;
- η : le niveau de bruit. $\eta \in]0, \frac{1}{2}[$;
- t : le seuil d'acceptation.

Remarque 42. Dans le protocole Improved-HB, le secret x est modifié à chaque ronde par décalage. Dans le protocole HB-MP+, on utilise une clef de ronde obtenue grâce à une fonction à sens unique. D'autre part, le paramètre m est strictement inférieur au paramètre k.

Le protocole Improved HB-MP : Leng, Mayes & Markantonakis (2008)

- Données publiques : k, r, η, $u \in]\eta, \frac{1}{2}[$, m, f
- Secrets partagés : x, $y \in \{0,1\}^k$

Protocole de type Challenge-Réponse en r rondes.

Le tag est authentifié si le nombre d'erreurs est $\leq t = u \times r$.

Le protocole HB-MP+ : Leng, Mayes & Markantonakis (2008)

- Données publiques : k, r, η, $u \in]\eta, \frac{1}{2}[$, m, f

– Secret partagé : $x \in \{0,1\}^k$

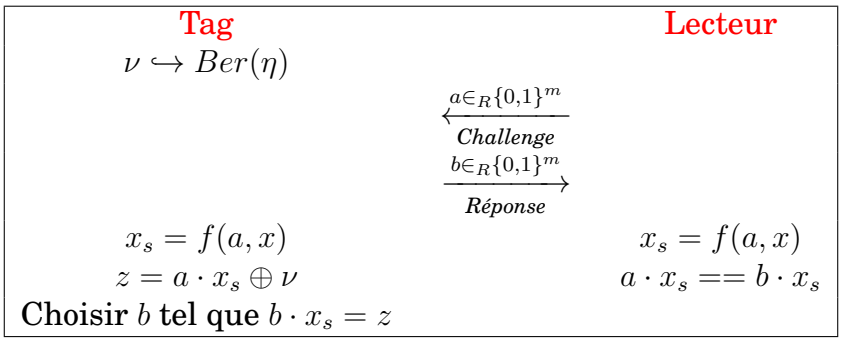

FIGURE 7.11 – Le protocole HB-MP+ : Leng, Mayes & Markantonakis (2008)

Ce protocole de type Challenge-Réponse se déroule en r rondes. Le tag est authentifié si le nombre d'erreurs est $\leq t = u \times r$.

Algorithme pour trouver un b valide

– Données : a, x, n.
– Sortie : b tel que $b \cdot x = a \cdot x \oplus \nu$ avec $Pr(\nu = 1) = \epsilon$.

```
Calcule z=a.x
Tant que n>0
  Genere un vecteur aléatoire b de k bits
  Si b.x=z
    Sortie Boucle
  n=n-1
Fin Tant que
Envoie b
Fin
```

Avantages et inconvénients de HB-MP+

– L'utilisation d'une clef de ronde permet de choisir des paramètres plus petits. La taille m de la clef de ronde peut-être réduite à 224 bits (contre 512 dans HB+). Le niveau de sécurité est moindre (52 bits) mais il s'agit là, on le rappelle, d'une clé de ronde.
– Le protocole se déroule en 2 passes (contre 3 dans HB+). Le coût de transmission est ainsi réduit de manière non négligeable.
– Ce protocole nécessite toujours un trop grand nombre de rondes.
– Le coût de transmission, bien que diminué, est trop important pour les systèmes RFID actuels.
– L'utilisation d'une fonction à sens unique est une bonne chose d'un point de vue théorique mais en pratique...

7.3.3 Le protocole PUF-HB

Pour finir cette partie, nous évoquerons succintement le protocole PUF-HB introduit en 2008 dans [HS08]. Cette construction n'est pas à proprement parler une variante de HB. En vérité, elle réunit deux familles de fonctions d'authentification : HB d'un côté et PUF (Physically Unclonable Functions) de l'autre. Ce protocole se veut être le premier basé sur HB qui résistent aux modifications d'un point de vue hardware. Il s'agit là d'un aspect que nous ne traiterons pas ici largement mais qui vaut la peine d'être explicité néanmoins.

Il existe plusieurs types d'attaques et nous pouvons distinguer deux grandes familles : les attaques logiques et les attaques physiques. Les attaques logiques sont celles qui nous préoccupent, elles portent sur le canal de données. Les attaques physiques se concentrent sur les canaux auxiliaires, on parle alors d'attaque par canal auxiliaire. L'importance de ces attaques dans le contexte RFID est loin d'être négligeable puisque les adversaires ont un accès physique aux tags et aux lecteurs étant donnée la nature des applications. On peut là encore subdiviser les attaques physiques en deux sous groupes : attaque passive où l'attaquant observe les canaux auxiliaires (temps de calcul, consommation d'énergie, émanation électromagnétique, température...) et attaque active où l'attaquant a la capacité d'injecter des fautes pendant les calculs. Un protocole qui résiste à ces attaques est dit « tamper-resilient ». Une des approches possibles pour remplir cette propriété sont les PUF. Une PUF est une fonction pseudo-aléatoire physique (un circuit challenge-réponse). L'idée que l'on doit retenir est que deux PUF qui possèdent le même circuit logique produisent des sorties différentes selon les paramètres d'implémentation du circuit. Ces variations d'implémentation ne sont pas contrôlables et sont directement liées aux aspects physiques de l'environnement (température, niveau de pression, ondes électromagnétiques, fluctuations quantiques).

Pour clore ce chapitre, voici la liste de toutes les variantes de HB parues à ce jour :

- HB : Hopper, Blum (Asiacrypt 2001) ;
- HB+ : Juels, Weis (Crypto 2005) ;
- HB++ : Bringer, Chabanne (2006) ;
- variante de Piramathu : (2006) ;
- HB* : Duc, Kim (2007) ;
- HP MP, HB MP' : Munilla, Peinado (2007) ;
- HB#, Random HB# : Gilbert, Robshaw, Seurin (Eurocrypt 2008) ;
- Trusted HB : Bringer, Chabanne (2008) ;
- Improved HP MP, HB MP+ : Leng, Mayes, Markantonakis (2008).
- PUF-HB : Ghait Hammouri, Berk Sunar (2008).
- HB-MP++ : Yoon (2009).
- Tree-based HB : Halevi, Saxena, Halevi (2010).
- NLHB : Madhavan, Hangaraj, Sankarasubramaniam et Viswanathan (2010).
- HB^N : Bosley, Haralambiev et Nicolosi (2011).

FIGURE 7.12 – Liste des variantes de HB

8 Codes et LPN

8.1 Quelques problèmes difficiles liés aux codes

Définition 65. (Problème du décodage borné d'un code linéaire)

Données :
– G : matrice binaire $k \times n$ de rang k
– x : vecteur binaire de longueur n
– w : entier positif
Problème : Trouver, s'il existe un vecteur binaire m de longueur k tel que $d(x, mG) \leq w$.

Naturellement, la difficulté de ce problème varie en fonction la valeur de w. On renvoie la lecteur à [Fin04, page 16] pour une discussion à ce propos.

Définition 66. (Syndrome Decoding)

Données :
– H : matrice binaire $n \times r$.
– s : vecteur binaire de longueur r
– w : entier positif
Problème : Trouver, s'il existe, un vecteur binaire non nul e de longueur n et de poids inférieur ou égal à w tel que $eH = s$.

Ce problème est *NP-complet* [1] [BMvT78]. On le retrouve parfois sous la dénomination de « problème du poids des cosets » . Plusieurs variantes de ce problème ont été étudiés (*e.g.* [Fin04, pp. 79-81]). La variante où la matrice H possède un rang r est équivalente au problème du décodage borné d'un code linéaire [Can96, page 23] et elle est également NP-complète [BMvT78, Section IV].

Définition 67. (Problème de la distance minimale)

Données :
– H : matrice binaire $n \times r$.
– w : entier positif
Problème : Trouver, s'il existe, un vecteur binaire non nul e de longueur n et de poids inférieur ou égal à w tel que $eH = 0$.

1. On a énoncé la version calculatoire du problème du Syndrome Decoding. Lorsqu'il est question de NP-complétude, on parle en toute rigueur de la version décisionnelle du problème. Cette remarque s'applique encore par la suite.

Le problème de la distance minimale est une variante importante du problème du Syndrome Decoding. Vardy a déterminé dans [Var97] la NP-complétude de ce problème. Cette question était restée en suspens de longues années [Wel71, BMvT78].

Définition 68. (Problème du décodage complet d'un code linéaire)

Données :
– G : matrice binaire $k \times n$ de rang k
– x : vecteur binaire de longueur n
Problème : Trouver un vecteur binaire m de longueur k tel que $d(x, mG)$ soit minimale.

Ce problème calculatoire est très difficile. Il s'agit de pouvoir décoder n'importe quel mot de l'espace. Il s'agit en fait du problème du décodage borné jusqu'au rayon de recouvrement. Cette quantité est la valeur minimale r telle que les boules centrées sur les mots du code de rayon r recouvrent l'espace entier. La version décisionnelle du problème est laissée en exercice.

8.2 Décodage par ensemble d'information

Le décodage par ensemble d'information (Information Set Decoding) est une technique datant des années 60. Elle s'oppose aux techniques de recherches exhaustives (qui énumèrent les mots de code ou les vecteurs d'erreur) du fait qu'elle exploite la redondance du code. L'idée principale étant ici de constater qu'il suffit pour décoder un vecteur x de trouver un ensemble de k positions d'information ne contenant aucune erreur.

Définition 69. (Set Information Decoding)
Soit C un code linéaire de longueur n et de dimension k, et $\mathbf{G}$ une matrice génératrice de C. Un sous-ensemble $\mathbf{I}$ de {1,. . .,n} de taille k est un **ensemble d'information** pour le code C si et seulement si la restriction du code à ces k positions est un espace vectoriel de dimension k. Ceci équivaut à dire que la matrice carrée correspondant à la restriction de $\mathbf{G}$ aux positions de $\mathbf{I}$ est inversible.
De manière similaire, **un ensemble de redondance** pour le code est le complémentaire d'un ensemble d'information.

Objectif : Trouver un vecteur d'erreurs de poids inférieur ou égal à w.

1. On choisit aléatoirement un ensemble d'information I ;
2. On met la matrice $\mathbf{G}$ sous forme systématique $\mathbf{U}^{-1}\mathbf{G} = (Id, Z)_I$ et on décompose le vecteur x sous la forme $(x_i, x_j)_I$ ce qui signifie que x_i est la restriction de x à $\mathbf{I}$ et x_j la restriction de x à son complémentaire. $\mathbf{U}$ est la restriction de $\mathbf{G}$ à $\mathbf{I}$;
3. On calcule $w(x_j + x_i Z)$. Si le poids de ce vecteur est inférieur ou égal à w, alors $e = (0, x_j + x_i Z)_I$. Sinon, on itère l'algorithme.

TABLE 8.1 – Algorithme de décodage par ensemble d'information

Cet algorithme permet de modifier un certain motif d'erreurs. On corrige tous les vecteurs ne possédant aucune position erronnée dans l'ensemble d'information $\mathbf{I}$

Remarque 43. La valeur de w est de l'ordre de la capacité de correction ou de la distance minimale. On effectue ici un décodage borné.

Remarque 44. On n'obtient pas nécessairement un ensemble d'information en choisissant aléatoirement k positions. Moins du tiers des choix de k colonnes fournissent un ensemble d'information. Cependant, le coût de l'élimination de Gauss n'augmente que négligemment, il suffit tout au plus d'effectuer quelques permutations sur les colonnes de G.

Remarque 45. Pour un code MDS, on a la propriété que n'importe quel choix de k colonnes de G donne une famille de vecteurs linéairement indépendants.

8.3 Les attaques sur LPN

Motivation

L'intérêt d'étudier les meilleurs algorithmes résolvant LPN ou SD provient du fait que la sécurité des protocoles HB repose sur ces problèmes difficiles et donc qu'il est nécessaire de les connaître pour pouvoir évaluer la sécurité du protocole. Nous nous fions à ces algorithmes pour estimer la taille des paramètres pour un niveau de sécurité donné. Si de nouveaux algorithmes plus performants paraissent, notre marge de sécurité diminue et il faudra donc redimensionner les paramètres en conséquence.

8.3.1 Attaque passive sur HB

Nous récoltons par écoute passive de plusieurs exécutions du protocole HB n couples de challenges-réponses (a, z). Pour LF2, le LPN-solver qui nécessite le moins de requêtes, n est de l'ordre de 10000 ce qui correspond à une dizaine d'authentifications. Les colonnes de la matrice génératrice G $(k \times n)$ sont formées par les challenges a émis par le lecteur. Le code C engendré par G est bien un code linéaire aléatoire. La clé secrète x est un message dont l'encodage dans C fournit les réponses z. La réponse z du lecteur est un mot de code bruité. On peut imaginer, pour modéliser ce phénomène, que le mot de code est transmis sur un canal binaire symétrique de probabilité de transition η. Le but de l'attaquant est ici de retrouver le message émis (*i.e.* la clef secrète x) à partir de z. Nous avons alors un problème de décodage que nous résolvons en employant l'un des algorithmes présentés ci-dessous. Nous remarquons que la dimension du code vaut k (taille conseillée : 512 bits) et est donc fixée par le protocole alors que la longueur du code n peut-être très grande puisqu'elle correspond au nombre d'observations du protocole par l'adversaire. Le rendement peut donc être rendu aussi faible que le veut l'adversaire. De manière similaire, nous notons que la réponse z du tag est bruitée avec un taux d'erreur fixe η qui vaut généralement 0.125 ou 0.25 et que la capacité de correction théorique de notre code est bien au dessus du taux d'erreur du fait de la quantité d'information que l'adversaire peut engranger.

8.3.2 L'algorithme BKW de Blum, Kalai et Wasserman (2000)

Nous n'allons pas détailler de manière exhaustive l'algorithme. Nous nous contenterons de donner les grandes lignes pour que vous ayez un aperçu de ce qui est fait. Le lecteur intéressé pourra se reporter au papier original [BKW00].

On tire au hasard s colonnes de la matrice génératrice G. On partitionne ces s chaînes selon leurs b derniers bits. On a donc au plus 2^b classes d'équivalences. On choisit une colonne dans chacune de ces classes et on l'ajoute à toutes celles qui sont dans sa classe puis on l'élimine. On possède alors des colonnes dont les b derniers bits sont nuls. On répète cette procédure jusqu'à ce que l'on n'ait plus que des vecteurs dont uniquement les b premiers bits sont non nuls. On cherche ensuite à trouver plusieurs combinaisons XOR de ces colonnes qui donnent les vecteurs e_i de la base canonique. On estime alors la valeur de x_i par un vote majoritaire. L'idée de l'algorithme repose sur le paradoxe des anniversaires généralisé proposé par Wagner [Wag02].

8.3.3 L'algorithme de Fossorier, Mihaljevic et al. (2006)

Cet algorithme s'appuie sur deux éléments :

1. L'algorithme BKW qui était considéré jusqu'alors comme étant le meilleur algorithme pour résoudre LPN
2. Les attaques par corrélation rapide utilisées en chiffrement à flot.

L'algorithme de Fossorier et al. [FMI+06] est en fait une généralisation de BKW avec des paramètres optimisés. La complexité de BKW est sous-exponentielle et est estimée à $2^{O(\frac{k}{log(k)})}$.

Le lien entre LPN et les attaques par corrélation rapide

Bien que ces problèmes apparaissent dans des contextes différents (à savoir le Machine Learning et le chiffrement à flot), il convient de noter qu'ils peuvent tous deux être vus comme un problème de résolution d'un système surdéfini (plus d'équations que d'inconnues) d'équations linéaires bruitées. Pourtant, l'angle d'attaque pris par les cryptanalystes a été différent de celui pris dans BKW jusqu'en 2006. En plus de cela, de nouvelles techniques d'attaques par corrélation rapide sont apparues ces dernières années. Fossorier et al. ont mis à profit cela dans leur travail sur LPN. Nous ne parlerons pas ici en détail des attaques par corrélation rapide pour ne pas nous disperser mais le lecteur intéressé pourra consulter [CJM02] ainsi que plusieurs articles de Fossorier et Mihaljevic. Dans [FMI+06], Fossorier et al. insiste sur le fait qu'il y a une différence notable entre les LPN-solvers et les attaques par corrélation rapide puisque pour LPN, aucune phase de précalcul n'est permise.

8.3.4 Les algorithmes LF1 et LF2 de Levieil et Fouque (2006)

L'article de Levieil et Fouque [LF06] qui présente les algorithmes LF1 et LF2 est paru en 2006. Il semblerait que leurs travaux aient été menés de manières indépendantes de ceux de Fossorier et Mihaljjevic, qui datent eux aussi de 2006. LF1 et LF2 s'appuient encore une fois

sur l'algorithme BKW [BKW00]. L'algorithme LF2 est considéré à l'heure comme le meilleur algorithme pour résoudre LPN, il est de fait utilisé aujourd'hui encore pour dimensionner les paramètres de HB+. Il utilise plusieurs outils comme la transformée de Walsh-Hadamard (pour LF1) qui permet d'accélérer une phase de calcul et le paradoxe de Wagner. [Wag02] (pour LF2) qui permet d'améliorer heuristiquement l'algorithme. Dans LF1, pour diminuer la complexité en temps et en requêtes, lors de la phase ultime de l'algorithme, il ne jette pas la plupart des équations contrairement à ce qui est fait dans BKW. L'algorithme fait intervenir des paramètres a et b qui vérifient la relation : $k = a \times b$. La complexité en requêtes reste importante mais diminue par rapport à BKW, l'algorithme nécessite $q = (8 \times b + 200) \times \delta^{-2^a} + (a-1) \times 2^b$ requêtes et comparativement à cela les complexités en temps et en mémoire sont petites. LF2 est une amélioration de LF1 reposant sur un argument heuristique qui n'a donc pas été prouvé. Au lieu de choisir un vecteur dans chaque classe d'équivalence et de l'ajouter à tous les autres, on calcule toutes les sommes deux à deux de vecteurs dans chaque classe, cela permet en pratique de diminuer le nombre de requêtes nécessaires. Personne n'a encore donné de borne sur le nombre de requêtes nécessaires mais l'algorithme LF2 a été implémenté et fonctionne relativement bien en pratique.

8.4 Attaque sur la primitive

On peut distinguer deux catégories d'attaque : les attaques sur le protocole (*e.g.* Man in the Middle) et les attaques sur la primitive. Nous nous sommes intéressés à ces dernières. La primitive de HB est le problème NP-complet LPN, cela signifie que l'on a réalisé des preuves de sécurité des protocoles de type HB en effectuant une réduction polynomiale de LPN. On se pose dès lors les questions suivantes :

- Est-il possible de faire une réduction du Syndrome Decoding à la sécurité de HB quitte à utiliser des hypothèses fortes ?
- Peut-on paramétriser de manière plus sûre HB en adoptant cette approche basée sur les codes ?

On utilise des fonctions écrites en C permettant de calculer par approximation le facteur de travail (workfactor) pour décoder un code arbitraire linéaire en utilisant les techniques d'Information Set Decoding vues dans la Section 8.2 (Stern, Canteaut-Chabaud). Il a fallu comprendre ces programmes puis les adapter à HB . On considère trois paramètres : la longueur du code n , sa dimension k et le niveau de bruit η. On a utilisé des paramètres de taille identique à ceux utilisés dans les attaques sur LPN. On a donc : $n = 10000$, $k = 512$ et $\eta = 0.125$ ou bien $\eta = 0.25$. Nous rappelons que les précédents travaux se basaient sur une approche LPN. Nous avons cherché à évaluer si ces paramètres demeurent sûrs en adoptant une approche basée sur le problème SD. Le cas échéant, il aurait fallu modifier en conséquence la taille de certains paramètres pour assurer la sécurité de ces protocoles contre les meilleurs attaques connues contre SD. Nous avons donc calculé le workfactor. Son calcul dépend de deux facteurs : le coût d'une itération et le nombre d'itérations effectuées. Les résultats obtenus ne sont pas meilleurs qu'avec LPN. Deux situations se présentent à nous. Dans le premier cas, le facteur de travail est beaucoup trop grand ($>> 2^{80}$). Dans le second cas, on trouve que la densité est positive. Cela signifie qu'il existe plusieurs mots de poids w dans le code étudié.

Le workfactor indique la complexité pour trouver un mot quelconque du code qui possède un poids. Ce n'est pas ce qui nous intéresse ici puisque nous recherchons une solution précise, il s'agit du secret partagé pour assurer l'authentification. En conséquence, plus la densité est élevée, plus la probabilité que la solution trouvée soit la bonne est faible. On obtient donc un résultat négatif. En conclusion, l'approche basée sur le problème SD se révèle être inadaptée à ce contexte. On peut expliquer ce résultat en remarquant que le code étudié possède des paramètres inhabituels. Il possède en effet un rendement extrêmement faible (de l'ordre de $\frac{1}{100}$) qui le rend impropre à être utilisé pour fiabiliser des communications. Les attaques basées sur SD n'ont pas été étudiées pour des codes possédant ce type de paramètres. Elles se révèlent être inefficaces en pratique contre des codes possédant ce type de paramètres.

8.5 Preuve de de sécurité

Une tendance significative de la recherche la plus récente en cryptographie, en particulier pour la cryptographie asymétrique, vise à substituer aux approches heuristiques une «sécurité prouvée». Le simple fait qu'un algorithme cryptographique ait résisté durant plusieurs années aux attaques des cryptanalystes a constitué pendant longtemps la seule forme possible de validation. Un paradigme totalement distinct provient du concept de « sécurité prouvée ». Cette approche propose des preuves qui ramènent la sécurité du schéma proposé à celle d'un problème que l'on peut considérer comme difficile, tel que la factorisation des entiers ou le problème du logarithme discret.

On distingue habituellement les preuves « idéalisées » où l'on identifie les fonctions de hachage à des fonctions aléatoires (modèle de l'oracle aléatoire) et les chiffrements par bloc à des permutations aléatoires (modèle du chiffrement idéal) des preuves « réelles » où les hypothèses sont plus réalistes (résistance des fonctions de hachage à la recherche de collisions, ou simplement à la recherche de pré-images). Les méthodes mises en oeuvre dans la méthodologie des preuves de sécurité sont à l'origine celles de la théorie de la complexité, où un algorithme cryptographique est réalisé par une machine de Turing polynomiale probabiliste. Les preuves de sécurité sont des arguments relatifs ramenant la sécurité d'un algorithme cryptographique à la difficulté supposée d'un problème particulier issu en général de la théorie des nombres, en faisant éventuellement appel à un oracle pour calculer certaines fonctions (cas des preuves «idéalisées»). Une preuve dans un modèle « idéal » est une indication que le schéma est correctement construit, mais cela ne donne pas une garantie parfaite de la sécurité du schéma lors de son utilisation pratique. Au contraire, une preuve de sécurité dans un modèle « standard » est particulièrement utile car elle permet d'avoir une confiance parfaite dans la sécurité du schéma lors de son utilisation pratique. Les preuves peuvent être non strictement réductibles (« loose » en anglais) ou strictement réductibles (« tight » en anglais) . Une preuve « loose » utilise un attaquant et résout le problème difficile avec une probabilité faible comparée à celle de l'attaquant. Au contraire, une preuve « tight » résout le problème avec une probabilité très proche de celle de l'attaquant. Ainsi, une preuve « tight » est un bien meilleur gage de sécurité mais elle est généralement difficile à trouver..

Les preuves de sécurité font donc appel à des algorithmes probabilistes. Des majorations sur les probabilités apparaissent dans ce contexte ce qui a tendance à obscurcir l'argumen-

tation. Pour remédier à cela, Victor Shoup a proposé un cadre élégant pour organiser les techniques probabilistes à l'aide de jeux successifs [Sho04].

On veut à présent effectuer une réduction de sécurité basée sur SD. Notre travail consiste dans un premier temps à étudier les différentes techniques pour effectuer une preuve de sécurité.

Explicitons tout d'abord le contexte ainsi que les hypothèses. L'attaquant a le rôle du lecteur, son objectif est de retrouver le secret $X||Y$. Le lecteur décide de la valeur des challenges a. Il peut par exemple les mettre à 0. Auquel cas, le tag lui renverra $bY \oplus \nu$. On peut aussi émettre l'hypothèse forte que l'attaquant connaît le secret X, il peut décider alors d'annuler aX en choisissant a avec soin. Toujours est-il que dorénavant l'on se retrouve dans le contexte du décodage d'un mot bruité $bY \oplus \nu$ dans un code linéaire. La dimension du code est la taille du secret y, la longueur du code est un multiple du nombre de challenges puisqu'un tag peut s'identifier plusieurs fois dans sa vie. Le poids de ce mot est $u \times m$ ($\eta \times r$ pour HB+) où $u \in [\eta, \frac{1}{2}[$, m est le nombre de colonnes de X et Y, r est le nombre de rondes dans HB+ (rôle équivalent à m). Usuellement, u vaut 1/4.

8.6 Preuve formelle de l'équivalence entre les problèmes LPN et SD

Nous allons montrer ici que LPN se réduit à SD et réciproquement. Pour commencer rappelons les données des problèmes ainsi que leur énoncé.

- Données pour LPN : $z \in \{0,1\}^n$, $A \in \{0,1\}^{k \times n}$, $\eta \in [0, \frac{1}{2}[$.
- Question LPN : Trouver, s'il existe, $x \in \{0,1\}^k$ tel que $d_H(z, xA) \leq \eta \times n$.
- Données pour SD : $s \in \{0,1\}^r$, $H \in \{0,1\}^{r \times n}$, $w \geq 0$.
- Question SD : Trouver, s'il existe, $e \in \{0,1\}^n$ tel que $eH^T = s$ et $w(e) \leq w$.

8.6.1 LPN est réductible à SD

Nous allons montrer ici que si nous avons un oracle pour résoudre SD, nous pouvons résoudre LPN. C'est-à-dire que nous allons transformer une instance quelconque de LPN en une instance de SD, nous pourrons ainsi faire appel à notre oracle qui résout SD pour solutionner LPN.

1. On dispose de $z \in \{0,1\}^n$, $A \in \{0,1\}^{k \times n}$, $\eta \in [0, \frac{1}{2}[$.
2. On cherche s'il existe, $x \in \{0,1\}^k$ tel que $d_H(z, xA) \leq \eta \times n$.
3. On pose $k = r - n$, H matrice de parité de $< A >$ obtenue par un pivot de Gauss, $w = \lfloor \eta \times n \rfloor$.

4. On prend :

$$\begin{aligned} s &= zH^T \\ &= (xA+e)H^T \text{ pour un certain } e \in \{0,1\}^n. \\ &= xAH^T + eH^T \\ &= eH^T \end{aligned}$$

5. On fait appel à l'oracle SD$(s, H, w) = e$.
6. S'il existe une solution, on obtient $e \in \{0,1\}^n$ tel que $eH^T = s$ et $w(e) \leq w \leq \eta \times n$.
7. On a donc : $(z+e) \in < A >$, *i.e.*. il existe $x \in \{0,1\}^k$ tel que $z + e = xA$.
8. Sous réserve que $rg(A) = k$, on trouve alors $x = (xA)A^{-1}$ tel que $d_H(z, xA) = w(e) \leq \eta \times n$.

8.6.2 SD est réductible à LPN

Nous allons montrer que si nous disposons d'un oracle pour résoudre LPN, nous pouvons alors résoudre SD.

1. On dispose de $s \in \{0,1\}^r$, $H \in \{0,1\}^{r \times n}$, $w \geq 0$.
2. On cherche s'il existe, $e \in \{0,1\}^n$ tel que $eH^T = s$ et $w(e) \leq w$.
3. On prend H une matrice de parité de $< A >$ obtenue par un pivot de Gauss.
4. On pose $\eta = \lfloor \frac{w}{n} \rfloor$.
5. On fait un pivot de Gauss sur H, on obtient $M = (Id_r||R) = UH$
6. On vient de supposer implicitement que H est de rang k.
7. On prend $z \in \{0,1\}^n$ tel que $z = (sU^T||0)$.
8. On vérifie si le syndrome de z est bien s.

$$\begin{aligned} zH^T &= (sU^T||0_k)H^T \\ &= (sU^T||0_k)(U^{-1}M)^T \\ &= (sU^T||0_k)M^T(U^{-1})^T \\ &= sU^T(U^{-1})^T \\ &= s \end{aligned}$$

9. On utilise l'oracle LPN$(z, A, \eta) = x$.
10. S'il existe une solution, on obtient $x \in \{0,1\}^k$ tel que $d_H(z, xA) \leq \eta \times n$.
11. On construit $e = xA + z$.
12. On a bien : $w(e) \leq \eta \times \leq w$ et $eH^T = zH^T = s$

Remarque 46. On n'a eu besoin que d'un seul appel à l'oracle pour effectuer la réduction dans les deux cas. On obtient donc une réduction fine entre les problèmes LPN et SD.

Nous venons de voir que le problème LPN peut se reformuler comme étant le problème du décodage d'un code linéaire en blocs. Ce problème, on l'a vu dans la Section 8.3, a été largement étudié par le passé notamment dans la cryptanalyse des cryptosystèmes de McEliece. Il nous paraît essentiel de mettre en évidence deux faits qui différencient les contextes HB et McEliece. Lors de la cryptanalyse de HB, le rendement (ou taux d'information) k/n est très faible et de plus le niveau de bruit du canal binaire symétrique sous-jacent est plutôt élevé. Nous exprimons ici le fait que les algorithmes de décodage d'un code linéaire n'ont pas été étudiés, optimisés pour ces types de paramètres. Il est donc envisageable d'améliorer certaines choses.

8.7 Proposition et analyse d'un nouveau schéma d'authentification low-cost basé sur LPN

8.7.1 Une idée de variante pour HB

Motivations et approche du problème

Tout d'abord, notre objectif en présentant une nouvelle variante de HB n'est pas de supplanter les précédentes versions. Nous nous contenterons modestement d'analyser notre proposition et de déduire à partir du dimensionnement des paramètres dans quelles gammes d'applications, cette dernière pourrait être utilisable. Nous avons vu dans la première partie du rapport que l'éventail des applications possibles pour la RFID est très large. Naturellement, elles ne requièrent pas toutes les mêmes critères et niveaux de sécurité. Nous nous restreindrons de plus à une analyse dans le modèle de détection vu en 7.2.2. Plus explicitement, nous nous intéresserons à un adversaire passif et aux attaques dites par « key-recovery ». On s'intéresse ici essentiellement aux attaques sur la primitive LPN plutôt qu'aux attaques protocolaires. On rappelle que les meilleurs algorithmes pour résoudre LPN sont LF1 et LF2 de Levieil et Fouque [LF06], on citera également l'algorithme de Fossorier, Mihaljevic et al. [FMI+06] qui s'appuient tous trois sur l'algorithme BKW de Blum, Kalai et Wasserman [BKW00].

Notre proposition s'appuie essentiellement sur la variante HB # de Gilbert, Robshaw et Seurin (2008) vue en 7.2.4. Après l'avoir énoncée, nous donnerons les avantages et inconvénients de notre version relativement à l'originale. Nous donnerons pour finir plusieurs idées pour optimiser le tout.

Énoncé :

- Données publiques : k_x, k_y, m, η, $u \in]\eta, \frac{1}{2}[$
- Secrets partagés : $x \in_R \{0,1\}^{k_x}$, $y \in_R \{0,1\}^{k_y}$

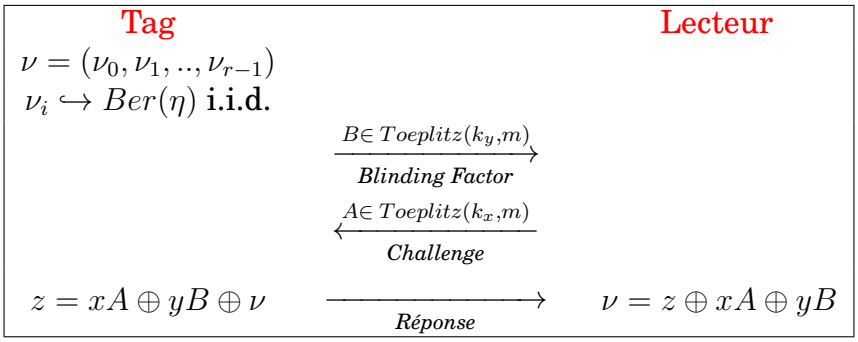

FIGURE 8.1 – Un nouveau schéma d'authentification low-cost basé sur LPN

Notre proposition conserve l'idée des matrices de Toeplitz et se déroule en 1 ronde. L'idée consiste à employer les matrices de Toeplitz pour communiquer le challenge et le blinding factor et non plus pour stocker les secrets. De manière immédiate, on observe que le coût de communication est réduit alors que le coût de stockage augmente. Selon les applications, cela pourra avoir ou non un effet bénéfique.

8.7.2 Évaluation de la sécurité du protocole en se basant sur les LPN-Solvers

La première question qui se pose à nous est comment peut-on évaluer la sécurité d'un protocole ce qui revient encore à comment dimensionner les paramètres pour rendre le protocole sûr en l'état actuel des connaissances. Notre entreprise est la suivante : nous allons comme nous l'avions déjà évoqué précedemment nous ramener au problème LPN. Pour cela, nous collectons pour la cryptanalyse des triplets blinding factor-challenges-réponses échangés durant les exécutions du protocole par des protagonistes honnêtes. Nous formons à partir de là, une matrice G et un vecteur z qui nous permettent de définir une instance du problème LPN. Chaque couple blinding factor-challenge correspond à une sous-matrice de G, deux matrices de Toeplitz superposées l'une sur l'autre et chaque réponse correspond à un sous ensemble des coordonnées de z. On dispose également du niveau de bruit η qui est un paramètre public. On résoud alors le problème LPN en utilisant les LPN-solvers, *i.e.*, les algorithmes dont nous avons discuté dans le chapitre précédent. On pourrait de plus simplifier la donne avec un attaquant actif qui se ferait passer pour le lecteur et annulerait la partie challenge de la matrice formée.

Nous ferons plusieurs remarques à partir de là. Tout d'abord les sessions d'authentification utilisent la même clé secrète, c'est la raison pour laquelle, nous pouvons utiliser les données collectées sur plusieurs sessions distinctes afin de retrouver le secret. On rappelle aussi que les challenges et les blinding factors utilisés sont des matrices de Toeplitz, les m colonnes sont donc liées par la relation suivante. La colonne $i+1$ étant obtenue par un décalage vers le bas de la colonne i et l'ajout d'un nouvel élément. Cela note une différence avec une

évaluation de HB où les vecteurs colonnes seraient indépendants sur l'ensemble des bits. Une dernière remarque pour dire que l'on obtient par écoute passive de s exécutions du protocole, une matrice G de $k_X + k_Y$ lignes et de $s \times m$ colonnes.

Il ne faut pas oublier de préciser la nature de ce que l'on calcule. La marge de sécurité calculée dans les différents articles sur le sujet (et ce depuis l'article de Juels et Weis [JW05]) mesure la complexité pour retrouver un bit de sécurité du secret et non la totalité.

k_x	k_y	m	η	t	$P[FR]$	$P[FA]$	Nb de bits transmis	Nb de bits stockés
80	512	1164	0.25	405	2^{-45}	2^{-83}	4082	592
80	512	441	0.125	113	2^{-45}	2^{-83}	1913	592

TABLE 8.2 – Jeux de paramètres pour HB#

Coût de stockage : $k_x + k_y$ bits.

Coût de transmission : $k_x + k_y + 3 \times m - 2$ bits.

Optimisations et idées envisageables

- Utiliser des matrices quasi-circulantes, Toeplitz par blocs.
- Généraliser notre version et HB# en utilisant des matrices de Toeplitz pour le secret, le challenge et le blinding factor. Cela permettra peut-être de trouver un meilleur compromis entre coût de communication, coût de stockage et sécurité.
- Adapter la phase 2 du protocole Trusted-HB à notre protocole ou à HB# (vu en 7.3.1).
- Utiliser un nombre d'erreurs dépendant d'une fonction simple à calculer voire un nombre d'erreurs fixé à l'avance. Exemple : Le lecteur et le tag partagent une fonction f, le lecteur envoie x au tag, le tag doit commettre $f(x)$ erreurs sans quoi il est rejeté. On se ramène ainsi véritablement à un problème de codes.
- Modifier l'ordre de l'échange des données, *i.e.* envoyer le challenge puis le blinding factor.
- Augmenter le niveau de bruit au dessus de $\frac{1}{4}$ permettrait de diminuer k_y et ainsi de diminuer les coûts de calcul et de stockage par la même occasion.
- Utiliser des challenges, blinding factors qui ont un nombre variable de lignes. Formellement, k_x et k_y varient mais la somme $k_x + k_y$ reste constante. Évidemment, il faut restreindre ces conditions. Autrement, un faux lecteur peut facilement retrouver le secret en choisissant entièrement le challenge. Ce n'est donc pas une solution idéale à première vue puisque là encore, il faudrait fixer un seuil. Néanmoins, cela reste une alternative à signaler.
- Rendre l'authentification bilatérale en demandant au lecteur de renvoyer le nombre exact d'erreurs injectées dans la réponse.
- Limiter le nombre d'échecs à l'authentification pour limiter les possibilités de l'attaquant à l'instar de ce qui est fait avec les cartes bancaires.
- Et enfin rappelons nous que l'idée fondamentale de HB est d'exploiter la similitude homme machine d'un point de vue capacité de stockage et de calcul. Aussi, nous paraîtrait-il judicieux d'étudier les différents protocoles HumanAut connus à ce jour afin d'essayer de les adapter au contexte RFID.

Table des figures

2.1 Certains codes linéaires en blocs 23

3.1 La méthode Rank Bound 39

6.1 $\mathrm{BTZ}_{d_{max}}$ vs. BTA ; $m = 11$; $K(+) = 1$; $K(\times) = m$ 75
6.2 $\mathrm{BTZ}_{d_{max}}$ vs. Chien ; $m = 11$; $K(+) = 1$; $K(\times) = m$ 75
6.3 $\mathrm{BTZ}_{d_{max}}$ vs. BTA ; $m = 11$; $K(+) = 1$; $K(\times) = m$ 76
6.4 $\mathrm{BTZ}_{d_{max}}$ vs. BTA ; $m = 15$; $K(+) = 1$; $K(\times) = m$ 76
6.5 $\mathrm{BTZ}_{d_{max}}$ vs. BTA ; $m = 40$; $K(+) = 1$; $K(\times) = m$ 76

7.1 Fonctionnement d'une RFID 80
7.2 La RFID au quotidien 81
7.3 Le protocole HB : Hopper & Blum (2001) 85
7.4 Le protocole HB+ : Juels & Weis (2005) 87
7.5 Attaque sur HB+ 88
7.6 Le protocole Random HB# : Gilbert, Robshaw & Seurin (2008) 89
7.7 Attaque sur HB# : Ouafi, Overbeck & Vaudenay (2008) 91
7.8 Comportement de $f_{m,t,\eta}(\overline{w}) = Pr_{\varepsilon \hookrightarrow Bin(m,\eta)}(w(\overline{\varepsilon} \oplus \varepsilon) \leq t)$ 92
7.9 Phase 1 du protocole Trusted-HB 93
7.10 Le protocole Improved HB-MP : Leng, Mayes & Markantonakis (2008) 95
7.11 Le protocole HB-MP+ : Leng, Mayes & Markantonakis (2008) 96
7.12 Liste des variantes de HB 97

8.1 Un nouveau schéma d'authentification low-cost basé sur LPN 108

Liste des tableaux

3.1 Borne inférieure de l'immunité spectrale de fonctions booléennes 42
3.2 Bornes sur la distance minimale du dual du code cyclique 3-correcteur 43

4.1 Classes connues de codes cycliques 3-correcteurs de longueur $2^m - 1$ 46
4.2 Liste des exposants monomiaux APN connus sur $\mathbb{F}_{2^m}$. 47
4.3 La distribution de poids du dual du code BCH 3-correcteur de longueur $2^m - 1$. 53

6.1 Nombre d'opérations dans $\mathbb{F}_{2^m}$ dans les procédures de Zinoviev 73
6.2 Temps de calcul pour trouver les racines d'un polynôme sur $\mathbb{F}_{2^m}$ 74
6.3 Nombre théorique d'opérations pour corriger des erreurs dans McEliece 74

7.1 Nombre nécessaire de bits pour une authentification fiable avec le protocole HB. 85
7.2 Jeux de paramètres pour HB# 90
7.3 Complexité de l'attaque sur HB# 91

8.1 Algorithme de décodage par ensemble d'information 100
8.2 Jeux de paramètres pour HB# 109

Bibliographie

[ACS92] Daniel Augot, Pascale Charpin, and Nicolas Sendrier. Studying the locator polynomials of minimum weight codewords of BCH codes. *IEEE Transactions on Information Theory*, 38(3) :960–973, 1992.

[AGa07] Gildas Avoine, Marc Girault, and al. RFID, Instrument de sécurité ou de surveillance ? MISC n°33, 2007. http://www.miscmag.com/fr/.

[AL96] Daniel Augot and Françoise Levy-dit-Vehel. Bounds on the minimum distance of the duals of BCH codes. *IEEE Transactions on Information Theory*, 42(4) :1257–1260, 1996.

[Aud97] Michèle Audin. Conseils aux auteurs de textes mathématiques, 1997. Texte disponible sur : www-irma.u-strasbg.fr/~maudin/newhowto.ps.

[Aug93] Daniel Augot. *Etude algébrique des mots de poids minimum des codes cycliques, méthodes d'algebre linéaire sur les corps finis*. PhD thesis, Université Paris 6, 1993.

[BBD09] Daniel Julius Bernstein, Johannes Buchmann, and Erik Dahmén. *Post-quantum cryptography*. Springer, 2009.

[BC08] Julien Bringer and Herve Chabanne. Trusted-HB : a low-cost version of HB+ secure against man-in-the-middle attacks. Cryptology ePrint Archive, Report 2008/042, 2008.

[BCE06] Julien Bringer, Hervé Chabanne, and Dottax Emmanuelle. HB^{++} : a light-weight authentication protocol secure against some attacks. In *IEEE International Conference on Pervasive Services, Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing – SecPerU 2006*, Lyon, France, June 2006. IEEE, IEEE Computer Society Press.

[Ber68a] Elwyn Ralph Berlekamp. *Algebraic coding theory*, volume 111. McGraw-Hill New York, 1968.

[Ber68b] Elwyn Ralph Berlekamp. Weight Enumeration Theorems. In *Sixth Allerton Conference Circuit and Systems Theory*, pages 161–170. Univ. of illinois Press, Urbana, IL, 1968.

[Ber69] S. D. Berman. On the theory of group codes. *Cybernetics and Systems Analysis*, 3 :25–31, 1969.

[Ber71] Elwyn R. Berlekamp. Factoring polynomials over large finite fields. In *SYMSAC '71 - Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, page 223, New York, USA, 1971. ACM.

[Ber73] Elwyn Ralph Berlekamp. Goppa codes. *IEEE Transactions on Information Theory*, 19 :590–592, 1973.

[BH09a] Bhaskar Biswas and Vincent Herbert. Efficient Root Finding of Polynomials over Fields of Characteristic 2. In *WEWoRC 2009*, LNCS. Springer-Verlag, 2009. http://www-rocq.inria.fr/secret/Vincent.Herbert/documents/article-weworc09.pdf.

[BH09b] Carl Bracken and Tor Helleseth. Triple-Error-Correcting BCH-like codes. In *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory - Volume 3*, ISIT'09, pages 1723–1725, Piscataway, NJ, USA, 2009. IEEE Press.

[BHN11] Carl Bosley, Kristiyan Haralambiev, and Antonio Nicolosi. HB^N : An HB-like protocol secure against man-in-the-middle attacks. Cryptology ePrint Archive, Report 2011/350, 2011.

[Bih03] Eli Biham, editor. *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*. Springer, 2003.

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *STOC '00 : Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 435–440, New York, NY, USA, 2000. ACM.

[Bla03] Richard E. Blahut. *Algebraic codes for data transmission*. Cambridge Univ Pr, 2003.

[Bla08] Richard E. Blahut. *Algebraic codes on lines, planes, and curves*. Cambridge Univ Pr, 2008.

[BLP08] Daniel Julius Bernstein, Tanja Lange, and Christiane Peters. Attacking and defending the McEliece cryptosystem. In *PQCrypto*, pages 31–46, 2008.

[Blu04] Antonia W. Bluher. On x^{q+1}+ax+b. *Finite Fields and Their Applications*, 10(3) :285–305, 2004.

[BM75] Ian F. Blake and Ronald C. Mullin. The Mathematical Theory of Coding. 1975.

[BMvT78] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C.A. van Tilborg. On the inherent intractability of certain coding problems ieee trans. *Inform. Theory, IT-24 (3)*, pages 384–386, 1978.

[Bos01] Nigel Boston. Bounding minimum distances of cyclic codes using Algebraic Geometry. *Electronic Notes in Discrete Mathematics*, 6 :385–394, 2001.

[Bra08] Carl Bracken. New Families of Triple-Error-Correcting Codes with BCH Parameters. *CoRR*, abs/0803.3553, 2008.

[BRC60] Raj Chandra Bose and Dwijendra Kumar Ray-Chaudhuri. On a class of error correcting binary group codes. *Inform. and Control*, 3 :68–79, 1960.

[BRS67] Elwyn R. Berlekamp, Howard Rumsey, and Gustave Solomon. On the solution of algebraic equations over finite fields. In *Information*

and Control, volume 10, pages 553–564, June 1967. Version pdf téléchargeable sur : http://dblp.uni-trier.de/db/journals/iandc/iandc10.html#BerlekampRS67.

[BS06] Emanuele Betti and Massimiliano Sala. A new bound for the minimum distance of a cyclic code from its defining set. *IEEE Transactions on Information Theory*, 52(8) :3700–3706, 2006.

[BS08] Bhaskar Biswas and Nicolas Sendrier. McEliece cryptosystem implementation : Theory and Practice. In *PQCrypto*, pages 47–62, 2008.

[Can96] Anne Canteaut. *Attaque de cryptosystèmes à mots de poids faible et construction de fonctions t-résilientes*. PhD thesis, Université Paris VI, 1996.

[CC97] Jay Cheng and Chi-chao Chao. On generalized Hamming weights of binary primitive BCH codes with minimum distance one less than a power of two. *IEEE Transactions on Information Theory*, 43 :294–298, 1997.

[CC98] Anne Canteaut and Florent Chabaud. A New Algorithm for Finding Minimum-Weight Words in a LinearCode : Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *IEEE Transactions on Information Theory*, 44(1) :367–378, 1998.

[CCD+05] Dario Catalano, Ronald Cramer, Ivan Damgard, Giovanni Di Crescenzo, David Pointcheval, and Tsuyoshi Takagi. *Contemporary Cryptology (Advanced Courses in Mathematics - CRM Barcelona)*. Birkhauser, 2005.

[CCO69] Robert T. Chien, B. Cunningham, and I. Oldham. Hybrid methods for finding roots of a polynomial–with application to BCH decoding. *IEEE Transactions on Information Theory*, 15 :329–335, 1969.

[CF09] Mathieu Cluzeau and Matthieu Finiasz. Recovering a code's length and synchronization from a noisy intercepted bitstream. In *Proceedings of the 2009 IEEE International Symposium on Information Theory*, pages 2737–2741. IEEE, 2009. http://www-roc.inria.fr/secret/Matthieu.Finiasz/research/2009/cluzeau-finiasz-isit09.pdf.

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In *ASIACRYPT*, pages 157–174, 2001.

[CGGea00] Anchung Chang, Peter Gaal, Solomon W. Golomb, and Guang Gong et al. On a conjectured ideal autocorrelation sequence, a related triple-error correcting cyclic code. *IEEE Transactions on Information Theory*, 46(2) :680–687, 2000.

[Cha94a] Florent Chabaud. On the security of some cryptosystems based on error-correcting codes. In *Theory and Application of Cryptographic Techniques*, pages 131–139, 1994.

[Cha94b] Pascale Charpin. Weight distributions of cosets of two-error-correcting binary BCH codes, extended or not. *IEEE Transactions on Information Theory*, 40 :1425–1442, 1994.

[Cha98] Pascale Charpin. Open problems on cyclic codes. In V. S. Pless and W. C. Huffman, editors, *Handbook of Coding Theory*, pages 963–1063. Amsterdam : Elsevier, 1998. Volume 1, Part 1, Chapter 11.

[Chi64] Robert T. Chien. Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes. In *IEEE Transactions on Information Theory*, volume 10, pages 357–363, 1964.

[Chi07] Hung-Yu Chien. Sasi : A new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. *IEEE Trans. Dependable Secur. Comput.*, 4(4) :337–340, 2007.

[CHS05] Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In *In Theory of Cryptography, Proceedings of TCC 2005, Lecture Notes in Computer Science*, pages 17–33. Springer-Verlag, 2005.

[CHZ06] Pascale Charpin, Tor Helleseth, and Victor A. Zinoviev. The coset distribution of triple-error-correcting binary primitive BCH codes. *IEEE Transactions on Information Theory*, 52 :1727–1732, 2006.

[CJM02] Philippe Chose, Antoine Joux, and Michel Mitton. Fast correlation attacks : An algorithmic point of view. In *EUROCRYPT*, pages 209–221, 2002.

[CM03] Nicolas Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *EUROCRYPT*, pages 345–359, 2003.

[CM06] Benoît Chevallier-Mames. *Cryptographie à clé publique : Constructions et preuves de sécurité*. PhD thesis, Université Paris VII, 2006.

[Coh93] Henri Cohen. *A course in computational algebraic number theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[CS84] Don Coppersmith and Gadiel Seroussi. On the minimum distance of some quadratic residue codes. *IEEE Transactions on Information Theory*, 30, 1984.

[CS11] Pascale Charpin and Sumanta Sarkar. Polynomials with linear structure and Maiorana-McFarland construction. *IEEE Transactions on Information Theory*, 57(6) :3796–3804, 2011.

[CTN10] Jose Carrijo, Rafael Tonicelli, and Anderson C. A. Nascimento. A Fault Analytic Method against HB+. Cryptology ePrint Archive, Report 2010/508, 2010.

[CU57] Leonard Carlitz and Saburo Uchiyama. Bounds for exponential sums. *Duke Mathematical Journal*, 24 :37–41, 1957.

[CZ81] David G. Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36 :587–587, 1981.

[DB09] Gilles Dowek and Gérard Berry. L'informatique oblige à repenser la classification des sciences : questions à Gilles Dowek. On a longtemps confondu l'informatique avec ses usages, questions à Gérard Berry ; propos recueillis par Dominique Chouchan. *La Recherche. Les Cahiers de l'Inria*, 06 2009. Version pdf téléchargeable sur : http://hal.inria.fr/inria-00527531/PDF/inria-n431-juin09.pdf.

[Del75] Philippe Delsarte. On subfield subcodes of modified Reed-Solomon codes (corresp.). *IEEE Transactions on Information Theory*, 21 :575–576, 1975.

[DFK97] Y. Desaki, Toru Fujiwara, and Tadao Kasami. The weight distributions of extended binary primitive BCH codes of length 128. *IEEE Transactions on Information Theory*, 43 :1364–1371, 1997.

[DK07] Dang Nguyen Duc and Kwangjo Kim. Securing HB+ against GRS man-in-the-middle attack. *In Institutes of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security*, 2007.

[DLC07] Frédéric Didier and Yann Laigle-Chapuy. Finding low-weight polynomial multiples using discrete logarithm. *Computing Research Repository*, abs/cs/070, 2007.

[Dob99] Hans Dobbertin. Almost perfect nonlinear power functions on GF(2^n) : The Welch case. *IEEE Transactions on Information Theory*, 45(4) :1271–1275, 1999.

[Dob01] Hans Dobbertin. Almost perfect nonlinear power functions on GF(2^n) : a new case for n divisible by 5. In *Finite fields and applications : proceedings of the Fifth International Conference on Finite Fields and Applications Fq5, held at the University of Augsburg, Germany, August 2-6, 1999*, page 113. Springer Verlag, 2001.

[ea09] William A. Stein et al. *Sage Mathematics Software*, 2009. www.sagemath.org.

[EKP06] Yves Edel, Gohar M. M. Kyureghyan, and Alexander Pott. A new APN function which is not equivalent to a power mapping. *IEEE Transactions on Information Theory*, 52(2) :744–747, 2006.

[Eli57] Peter Elias. *List decoding for noisy channels*. Massachusetts Institute of Technology, Research Laboratory of Electronics, 1957.

[Fin04] Matthieu Finiasz. *Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique*. PhD thesis, INRIA – Ecole Polytechnique, oct 2004. http://www-roc.inria.fr/secret/Matthieu.Finiasz/research/2004/finiasz-these.pdf.

[FMI+06] Marc P.C. Fossorier, Miodrag J. Mihaljevic, Hideki Imai, Yang Cuiz, and Kanta Matsuuraz. A novel algorithm for solving the LPN problem and its application to security evaluation of the HB protocol for RFID authentication, 2006. Disponible sur : http://eprint.iacr.org/2006/197.pdf.

[Fou01] Pierre-Alain Fouque. *Le partage de clés cryptograhiques : Théorie et Pratique*. PhD thesis, Université Paris VII, 2001.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself : Practical solutions to identification and signature problems. In *International Crytology Conference*, pages 186–194, 1986.

[FS09a] Matthieu Finiasz and Nicolas Sendrier. Security bounds for the design of code-based cryptosystems. In *ASIACRYPT*, pages 88–105, 2009.

[FS09b] Dmitry Frumkin and Adi Shamir. Un-Trusted-HB : Security Vulnerabilities of Trusted-HB. In *Workshop on RFID Security – RFIDSec'09*, Leuven, Belgium, July 2009.

[FTC03] S. Fedorenko, P. Trifonov, and E. Costa. Improved hybrid algorithm for finding roots of error-locator polynomials. In *European Transactions on Telecommunications*, volume 14, pages 411–416, 2003.

[GG07] Philippe Gaborit and Marc Girault. Lightweight code-based identification and signature. *IEEE Int. Symp. Inf. Theory*, pages 191–195, 2007.

[GMZZ11] Zbigniew Golebiewski, Krzysztof Majcher, Filip Zagorski, and Marcin Zawada. Practical attacks on HB and HB+ protocols. In Claudio Ardagna and Jianying Zhou, editors, *Workshop on Information Security Theory and Practice – WISTP'11*, volume 6633 of *Lecture Notes in Computer Science*, pages 244–253, Heraklion, Crete, Greece, June 2011. Springer.

[Gol68] Robert Gold. Maximal recursive sequences with 3-valued recursive cross-correlation functions (corresp.). *Information Theory, IEEE Transactions on*, 14(1) :154–156, 1968.

[Gop70] Valery D. Goppa. A new class of linear error-correcting codes. In *Probl. Inform. Transm.*, volume 6, pages 207–212, 1970.

[GRHH11] Guang Gong, Sondre Rønjom, Tor Helleseth, and Honggang Hu. Fast Discrete Fourier Spectra Attacks on Stream Ciphers. *IEEE Transactions on Information Theory*, 57(8) :5555–5565, 2011.

[GRS05] Henri Gilbert, Matt Robshaw, and Herve Sibert. An active attack against HB+ - a provably secure lightweight authentication protocol. Cryptology ePrint Archive, Report 2005/237, 2005.

[GRS08a] Henri Gilbert, Matthew J.B. Robshaw, and Yannick Seurin. Good variants of HB+ are hard to find. *In Proceedings of Financial Crypto 2008 (to appear)*, 2008.

[GRS08b] Henri Gilbert, Matthew J.B. Robshaw, and Yannick Seurin. HB# : Increasing the security and efficiency of HB+. Cryptology ePrint Archive, Report 2008/028, 2008.

[GZ61] Daniel Gorenstein and Neal Zierler. A class of error-correcting codes in p^m symbols. *Siam Journal on Control and Optimization*, 1961.

[Has97] Johan Hastad. Some optimal inapproximability results. In *Journal of the ACM*, pages 1–10, 1997.

[HB06] Nicholas J Hopper and Manuel Blum. Secure human identification protocols. *In C. Boyd, editor, Advances in Cryptology - Asiacrypt '01, volume 2248 in LNCS, pages 52-66*, 2006.

[Hel74] Hermann J. Helgert. Alternant codes. *Information and Computation/information and Control*, 26 :369–380, 1974.

[Hoc59] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2 :147–158, 1959.

[HP03a] William C. Huffman and Vera Pless. *Fundamentals of error-correcting codes*. Cambridge Univ. Press, 2003.

[HP03b] William Cary Huffman and Vera Pless. *Fundamentals of error-correcting codes*. Cambridge University Press, 2003.

[HR11] Tor Helleseth and Sondre Rønjom. Simplifying algebraic attacks with univariate analysis. In *Information Theory and Applications Workshop (ITA), 2011*, pages 1 –7, feb. 2011.

[HS08] Ghaith Hammouri and Berk Sunar. PUF-HB : A tamper-resilient HB based authentication protocol. In *ACNS*, pages 346–365, 2008.

[HS11] Vincent Herbert and Sumanta Sarkar. On the Triple-Error-Correcting Cyclic Codes with Zero Set $\{1, 2^i + 1, 2^j + 1\}$. In *IMACC 2011*, LNCS. Springer-Verlag, 2011.

[HSH09] Tzipora Halevi, Nitesh Saxena, and Shai Halevi. Using HB Family of Protocols for Privacy-Preserving Authentication of RFID Tags in a Population. In *Workshop on RFID Security – RFIDSec'09*, Leuven, Belgium, July 2009.

[HSH10] Tzipora Halevi, Nitesh Saxena, and Shai Halevi. Tree-based HB Protocols for Privacy-Preserving Authentication of RFID Tags. *Journal of Computer Security - Special Issue on RFID System Security*, 2010.

[HT72] Carlos R. P. Hartmann and Kenneth K. Tzeng. Generalizations of the BCH bound. *Information and Control*, 20(5) :489–498, 1972.

[Hub02] K. Huber. Note on decoding binary Goppa codes. In *Electronics Letters*, volume 32, pages 102–103, August 2002.

[Jen85] Jørn M. Jensen. The concatenated structure of cyclic and abelian codes. *IEEE Transactions on Information Theory*, 31 :788–793, 1985.

[Jue04] Ari Juels. Minimalist Cryptography for Low-Cost RFID Tags. In Carlo Blundo and Stelvio Cimato, editors, *International Conference on Security in Communication Networks – SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 149–164, Amalfi, Italia, September 2004. Springer-Verlag.

[Jue06] Ari Juels. RFID security and privacy : A research survey. *IEEE Journal on Selected Areas in Communications*, 24(2) :381–394, February 2006.

[JW05] Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. *V. Shoup (Ed.) : Crypto 2005, LNCS 3621, pp. 293-308*, 2005.

[Kah96] David Kahn. *The Codebreakers : The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, rev sub edition, December 1996.

[Kai10] J. F. Kaiser. Richard Hamming - You and Your Research. *Simula Research Laboratory*, pages 37–60, 2010.

[Kas69] Tadao Kasami. Weight Distributions of BCH Codes. In *Combinatorial Mathematics and Its Applications*, pages 335–357. Chapell Hill, NC : University of North Carolina Press, 1969.

[Kas71] Tadao Kasami. The Weight Enumerators for Several Classes of Subcodes of the 2nd Order Binary Reed-Muller Codes. *Information and Control*, 18(4) :369–394, 1971.

[KLP67] Tadao Kasami, Shu Lin, and William W. Peterson. Some results on cyclic codes which are invariant under the affine group and their application. *Information and Control*, 11(5/6) :475–496, 1967.

[Knu02] Lars R. Knudsen, editor. *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*. Springer, 2002.

[Kob07] Neal Koblitz. The uneasy relationship between mathematics and cryptography. *Notices of the AMS*, pages 973 – 979, September 2007.

[Kra94] Hugo Krawczyk. LFSR-based hashing and authentication. In *CRYPTO '94 : Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–139, London, UK, 1994. Springer-Verlag.

[KS06] Jonathan Katz and Adam Smith. Analyzing the HB and HB+ protocols in the "large error" case. Cryptology ePrint Archive, Report 2006/326, 2006.

[KSS10] Jonathan Katz, Ji Sun Shin, and Adam Smith. Parallel and Concurrent Security of the HB and HB+ Protocols. *Journal of Cryptology*, 23(3) :402–421, July 2010.

[LDmW94] Yuan Xing Li, Robert H. Deng, and Xin mei Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Transactions on Information Theory*, 40(1) :271–, 1994.

[Leo82] Jeffrey S. Leon. Computing automorphism groups of error-correcting codes. *IEEE Transactions on Information Theory*, 28(3) :496–510, 1982.

[LF06] Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2006. Disponible sur : http://dblp.uni-trier.de/db/conf/scn/scn2006.html#LevieilF06.

[LMM08] Xuefei Leng, Keith Mayes, and Konstantinos Markantonakis. HB-MP+ protocol : An improvement on the HB-MP protocol. *IEEE International Conference on RFID*, pages 118–124, April 2008.

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite fields*. Encyclopedia of mathematics and its applications. Cambridge University Press, 1997.

[Loi01] Pierre Loidreau. *Etude et optimisation de cryptosystèmes à clé publique fondés sur la théorie des codes correcteurs*. PhD thesis, Ecole Polytechnique, 5 2001.

[Man75] David M. Mandelbaum. On the derivation of Goppa codes (Corresp.). *IEEE Transactions on Information Theory*, 21 :110–111, 1975.

[Man77] David M. Mandelbaum. A method for decoding of generalized Goppa codes (Corresp.). *IEEE Transactions on Information Theory*, 23 :137–140, 1977.

[Man80] David M. Mandelbaum. Two applications of cyclotomic cosets to certain BCH codes (corresp.). *IEEE Transactions on Information Theory*, 26 :737–738, 1980.

[Mas69] James L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15 :122–127, 1969.

[Mas95] James L. Massey. Some applications of coding theory in cryptography. *Codes and Ciphers : Cryptography and Coding IV*, pages 33–47, 1995.

[Mas98] James L. Massey. The discrete Fourier transform in coding and cryptography. In *IEEE Inform. Theory Workshop, ITW 98*, pages 9–11, 1998.

[McE78] Robert J. McEliece. A public key cryptosystem based on algebraic coding theory. *DSN progress report, Jet Propulsion Laboratory*, 1978.

[MH78] Ralph Merkle and Martin Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24 :525–530, 1978.

[MK93] Oscar Moreno and P. Vijay Kumar. Minimum distance bounds for cyclic codes and Deligne's theorem. *IEEE Transactions on Information Theory*, 39 :1524–1534, 1993.

[Moo05] Todd K. Moon. *Error correction coding : mathematical methods and algorithms*. Wiley-Blackwell, 2005.

[MP07] J. Munilla and A. Peinado. HB-MP : A further step in the HB-family of lightweight authentication protocols. *Computer Networks*, 51(9) :2262–2267, June 2007. Disponible sur : http://www.sciencedirect.com/science/article/pii/S1389128607000242.

[MPC04] Willi Meier, Enes Pasalic, and Claude Carlet. Algebraic attacks and decomposition of Boolean functions. In *Advances in Cryptology-EUROCRYPT 2004*, pages 474–491. Springer, 2004.

[MS83] Florence J. MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes (North-Holland Mathematical Library)*. North Holland, January 1983.

[MS86] James L. Massey and Thomas Schaub. Linear complexity in coding theory. In *Coding Theory and Applications*, pages 19–32, 1986.

[MTSV10] Mukundan Madhavan, Andrew Thangaraj, Yogesh Sankarasubramaniam, and Kapali Viswanathan. NLHB : A Non-Linear Hopper Blum Protocol. arXiv.org, 2010.

[MvOV88] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Some computational aspects of root finding in GF(q^m). In *ISSAC*, pages 259–270, 1988.

[MZK95] O. Moreno, V. A. Zinoviev, and P. V. Kumar. An extension of the Weil-Carlitz-Uchiyama bound. *Finite Fields and Their Applications*, 1(3) :360 – 371, 1995. http://www.sciencedirect.com/science/article/pii/S107157978571026X.

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2) :159–166, 1986.

[Nih72] Yoji Niho. *Multi-Valued Cross-Correlation Functions between Two Maximal Linear Recursive Sequences*. Defense Technical Information Center, 1972.

[Nyb94] Kaisa Nyberg. Differentially uniform mappings for cryptography. In *Advances in cryptology - Eurocrypt 93*, pages 55–64. Springer, 1994.

[OOV08] Khaled Ouafi, Raphael Overbeck, and Serge Vaudenay. MiM attack against HB#. *Rump Session of EuroCrypt 2008*, 2008.

[Pat75] Nicholas Patterson. The algebraic decoding of Goppa codes. In *Information Theory, IEEE Transactions on*, volume 21, pages 203–207, 1975.

[Per98] Daniel Perrin. *Cours d'algèbre*. CAPES-agrég mathématiques. Ellipses, 1998.

[Per11] Edoardo Persichetti. Compact McEliece keys based on quasi-dyadic Srivastava codes. Cryptology ePrint Archive, Report 2011/179, 2011.

[Pet61] William W. Peterson. *Error-correcting codes*. M.I.T. Press, 1961.

[Pha05] Duong Hieu Phan. *Sécurité et efficacité des schémas cryptographiques*. PhD thesis, École Polytechnique, 2005.

[Pir06] Selwyn Piramuthu. HB and related lightweight authentication protocols for secure RFID tag/reader authentication. In *Collaborative Electronic Commerce Technology and Research – CollECTeR 2006*, Basel, Switzerland, June 2006.

[PLHCETR06a] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan Estevez-Tapiador, and Arturo Ribagorda. LMAP : A Real Lightweight Mutual Authentication Protocol for Low-cost RFID tags. Printed handout of Workshop on RFID Security – RFIDSec 06, July 2006.

[PLHCETR06b] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan Estevez-Tapiador, and Arturo Ribagorda. M2AP : A Minimalist Mutual-Authentication Protocol for Low-cost RFID Tags. In *International Conference on Ubiquitous Intelligence and Computing – UIC'06*, volume 4159 of *Lecture Notes in Computer Science*, pages 912–923. Springer-Verlag, September 2006.

[PLHCETR06c] Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda. EMAP : An Efficient Mutual Authentication Protocol for Low-Cost RFID Tags. In *OTM Federated Conferences and Workshop : IS Workshop – IS'06*, volume 4277 of *Lecture Notes in Computer Science*, pages 352–361. Springer-Verlag, November 2006.

[Poi02] David Pointcheval. *Le chiffrement asymétrique et la sécurité prouvée*. Habilitation à diriger des recherches, Université Paris VII, 2002.

[PPVD96] Ruud Pellikaan, M. Perret, S. G. Vladut, and Walter De. The shift bound for cyclic, Reed-Muller and geometric Goppa codes. In *Walter de Gruyter*, pages 155–174, 1996.

[PR97] Erez Petrank and Ron M. Roth. Is code equivalence easy to decide ? *Information Theory, IEEE Transactions on*, 43(5) :1602–1604, 1997.

[Pra57] Eugene Prange. *Cyclic Error-Correcting codes in two symbols*. Air Force Cambridge Research Center, 1957.

[PS03] Federico Ponchio and Massimiliano Sala. A lower bound on the distance of cyclic codes, 2003.

[PZ89] M.E. Pohst and H. Zassenhaus. Algorithmic algebraic number theory. encyclopaedia of mathematics and its applications, 1989.

[Ret75] Charles T. Retter. Decoding Goppa codes with a BCH decoder (Corresp.). *IEEE Transactions on Information Theory*, 21 :112–112, 1975.

[RH07] Sondre Rønjom and Tor Helleseth. A new attack on the filter generator. *IEEE Transactions on Information Theory*, 53(5) :1752–1758, 2007.

[Rod96] François Rodier. Estimation asymptotique de la distance minimale du dual des codes bch et polynômes de dickson. *Discrete Mathematics*, 149(1-3) :205–221, 1996.

[Roo82] Cees Roos. A generalization of the BCH bound for cyclic codes, including the Hartmann-Tzeng bound. *Journal of Combinatorial Theory*, 33 :229–232, 1982.

[Roo83] Cees Roos. A new lower bound for the minimum distance of a cyclic code. *Information Theory, IEEE Transactions on*, 29(3) :330 – 332, May 1983.

[RS60] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite field. *Siam Journal on Applied Mathematics*, 1960.

[RSCD97] F. Reinhardt, H. Soeder, J. Cuénat, and J. Dablanc. *Atlas des mathématiques*. La Pochothèque. Encyclopédies d'aujourd'hui. Librairie générale française, 1997.

[RvL91] P. J. N. De Rooij and Jacobus H. van Lint. More on the minimum distance of cyclic codes. *IEEE Transactions on Information Theory*, 37 :187–189, 1991.

[SA10] Mohammad Reza Sohizadeh Abyaneh. On the Security of Non-Linear HB (NLHB) Protocol Against Passive Attack. Cryptology ePrint Archive, Report 2010/402, 2010.

[Sam67] Pierre Samuel. *Théorie algébrique des nombres*. Collection Méthodes. Hermann, 1967.

[Sch88] Thomas Schaub. *A linear complexity approach to cyclic codes*. PhD thesis, Swiss Federal Institute of Technology, Zürich, 1988. Diss. ETH No. 8730.

[Sen00] Nicolas Sendrier. Finding the permutation between equivalent codes : the support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4) :1193–1203, jul 2000.

[Sen02] Nicolas Sendrier. Cryptosystèmes à clé publique basés sur les codes correcteurs d'erreurs, Habilitation à diriger les recherches, Université Pierre et Marie Curie, Paris 6, Paris, France. *Mars*, 2002.

[Ser83] Jean-Pierre Serre. Sur le nombre de points rationnels d'une courbe algébrique sur un corps fini. *C.R. Acad. Sci. Paris*, 296(I) :397–402, 1983.

[Sho04] Victor Shoup. Sequences of games : a tool for taming complexity in security proofs, 2004.

[Sho06] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge University Press, 2006.

[Sin99] Simon Singh. *Histoire des codes secrets*. Éditions Jean-Claude Lattès, 1999.

[SKHN75] Yasuo Sugiyama, Masao Kasahara, Shigeichi Hirasawa, and Toshihiko Namekawa. A method for solving key equation for decoding Goppa codes. *Information and Computation/information and Control*, 27 :87–99, 1975.

[SKHN76] Yasuo Sugiyama, Masao Kasahara, Shigeichi Hirasawa, and Toshihiko Namekawa. An erasures-and-errors decoding algorithm for Goppa codes (Corresp.). *IEEE Transactions on Information Theory*, 22 :238–241, 1976.

[Ste94] Jacques Stern. Can one design a signature scheme based on error-correcting codes ? In *ASIACRYPT*, pages 424–426, 1994.

[Ste98] Jacques Stern. *La Science du Secret*. Éditions Odile Jacob, 1998.

[Sti08] Henning Stichtenoth. *Algebraic Function Fields and Codes*. Springer Publishing Company, Incorporated, 2008.

[Str10] Falko Strenzke. A Smart Card Implementation of the McEliece PKC. In *WISTP'10*, pages 47–59, 2010.

[Tho83] Thomas M. Thompson. *From error-correcting codes through sphere packings to simple groups*, volume 21 of *Carus mathematical monographs*. pub-MATH-ASSOC-AMER, pub-MATH-ASSOC-AMER :adr, 1983.

[Van99] J.H. Van Lint. *Introduction to coding theory*, volume 86. Springer Verlag, 1999.

[Var97] Alexander Vardy. The intractability of computing the minimum distance of a code. *Information Theory, IEEE Transactions on*, 43(6) :1757–1766, 1997.

[VB03] István Vajda and Levente Buttyán. Lightweight Authentication Protocols for Low-Cost RFID Tags. In *Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003*, Seattle, WA, USA, October 2003.

[vDFDF03] E. R. van Dam and D. Fon-Der-Flaass. Codes, graphs, and schemes from nonlinear functions. *Eur. J. Comb.*, 24(1) :85–98, 2003.

[vDGvDV95] Gerard van Der Geer and Marcel van Der Vlugt. Generalized Hamming weights of BCH(3) revisited. *IEEE Transactions on Information Theory*, 41 :300–301, 1995.

[vDV96] Marcel van Der Vlugt. Non-BCH Triple-Error-Correcting codes. *IEEE Transactions on Information Theory*, 42(5) :1612–1614, 1996.

[vDV97] Marcel van Der Vlugt. Addendum to "Non-BCH triple-error-correcting codes". *Information Theory, IEEE Transactions on*, 43(5) :1747, 1997.

[vLW86] Jacobus H. van Lint and Richard M. Wilson. On the minimum distance of cyclic codes. *IEEE Transactions on Information Theory*, 32(1) :23–40, 1986.

[vOV89] Paul C. van Oorschot and Scott A. Vanstone. A geometric approach to root finding in GF(q^m). *IEEE Transactions on Information Theory*, 35 :444–453, 1989.

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge Univ. Press, 2003.

[vzGP01] Joachim von zur Gathen and Daniel Panario. Factoring polynomials over finite fields : A survey. *J. Symb. Comput.*, 31(1/2) :3–17, 2001.

[Wag02] David Wagner. A generalized birthday problem (extended abstract). In *In Advances in Cryptology - CRYPTO 2002*, page 3. Springer, 2002.

[War02] Henry S. Warren. *Hacker's Delight*. Addison-Wesley, 2002.

[WB86] Lloyd R. Welch and Elwyn Ralph Berlekamp. Error correction for algebraic block codes, 1986. US Patent 4,633,470.

[Wei48] André Weil. *Variétés abéliennes et courbes algébriques*. Actualités scientifiques et industrielles. Hermann, 1948.

[Wel71] Dominique J.A. Welsh. Combinatorial problems in matroid theory. *Combinatorial Mathematics and its Applications (ed. DJA Welsh), Academic Press, London*, pages 291–306, 1971.

[Win99] Arne Winterhof. Cyclic codes and the Frobenius automorphism. *Archiv Der Mathematik*, 72 :43–46, 1999.

[Wol89] Jacques Wolfmann. New bounds on cyclic codes from algebraic curves. In *Proceedings of the 3rd International Colloquium on Coding Theory and Applications*, pages 47–62, London, UK, 1989. Springer-Verlag.

[WS79] Lloyd R. Welch and Robert A. Scholtz. Continued fractions and Berlekamp's algorithm. *IEEE Transactions on Information Theory*, 25 :19–27, 1979.

[XL03] Chaoping Xing and San Ling. *Coding Theory : A First Course*. Cambridge University Press, New York, NY, USA, 2003.

[Yoo09] Bongno Yoon. HB-MP++ Protocol : An Ultra Light-weight Authentication Protocol for RFID System. In *IEEE International Conference on RFID – IEEE RFID 2009*, Orlando, Florida, USA, April 2009. IEEE, IEEE Computer Society.

[Zin96] Victor A. Zinoviev. On the solution of equations of degree ≤ 10 over finite fields GF(2^q). In *Rapport de recherche INRIA* n^o *2829*, 1996.

[ZSH10] Xiangyong Zeng, Jinyong Shan, and Lei Hu. A Triple-Error-Correcting Cyclic Code from the Gold and Kasami-Welch APN Power Functions. *Computing Research Repository*, abs/1003.5, 2010.

[Zém00] Gilles Zémor. *Cours de Cryptographie*. Cassini, 2000.

Notations

$\lfloor x \rfloor$: la partie entière inférieure du nombre réel x
$[\![a, b]\!]$: les entiers compris entre a et b
$\deg(f)$: le degré du polynôme f
$\min(E)$: l'élément minimal (s'il existe) de l'ensemble E
M_I : la restriction de la matrice M aux colonnes dont la position est indiquée dans I
$\mathbb{Z}_n$: l'anneau des entiers modulo n
$\mathcal{A}[X]_n$: l'ensemble des polynômes de degré inférieur à n et à coefficient dans l'anneau $\mathcal{A}$
$x \times y$: la multiplication des nombres réels x et y
$x \cdot y$: le produit scalaire modulo 2 des vecteurs x et y
$||$: opérateur de concaténation
$==$: opérateur de comparaison
$w(x)$: poids de Hamming du vecteur binaire x
$d(x, y)$: distance de Hamming entre les vecteurs binaires x et y
$\oplus$: addition bit à bit modulo 2 (XOR)
$X \hookrightarrow Ber(p)$: la variable aléatoire X suit la loi de Bernoulli de paramètre p
$\{0, 1\}^{n \times k}$: ensemble des matrices binaires de taille $(n \times k)$
$a \in_R A$: a est choisi au hasard dans l'ensemble A
$x = (a, b)_I$: a est la restriction de x à I, b est la restriction de x au complémentaire de I
2^i : vecteur biniare nul partout sauf en la i-ème position
$< M >$: code engendré par la matrice M
$rg(M)$: le rang de la matrice M
Id_n : la matrice identité d'ordre n
0_k : le vecteur nul de longueur k
$Toeplitz(m, n)$: une matrice de Toeplitz binaire de dimension (m, n)

Acronymes

APN Almost Perfect Nonlinear (presque parfaitement non-linéaire)
BKW Blum, Kalai, Wasserman
BTA Berlekamp Trace Algorithm
BTZ Berlekamp Trace - Zinoviev
Code BCH Code de Bose-Chaudhuri-Hocquenghem
Code GRS Code de Reed-Solomon généralisé
Code RM Code de Reed-Muller
CRC Cyclic Redundancy Code
e.g. exempli gratia
et al. et les autres
GRS Gilbert, Robshaw, Seurin
HB Hopper, Blum
HumanAut Human Authentication
i.e. id est
i.i.d. indépendantes et identiquement distribués
LF Levieil et Fouque
LFSR Linear Feedback Shift Register (registre à décalage à rétroaction linéaire)
LPN Learning from Parity with Noise
MHB Puzzle Matrix based HB Puzzle
MiM Man In The Middle
NP Nondeterministic Polynomial Time
pp. pages
PPT Probabilistic Polynomial Time
PRBG Pseudo Random Bit Generator
PRNG Pseudo Random Number Generator
RFID Radio Frequence IDentification
SD Syndrome Decoding

Index

ϵ-équilibrée, 93

algèbre sur un corps, 20, 24
algorithme
  Berlekamp-Massey, 37
  BKW, 102, 103, 107
  BTA, voir Trace de Berlekamp
  BTZ, voir Trace de Berlekamp - Zinoviev
  Canteaut-Chabaud, 103
  Cantor et Zassenhaus', 67, 68
  Chose-Joux-Mitton, 52
  Léon, 55
  LF2, 101–103, 107
  parallèle, 54
  Rank Bound, 37–40
  recherche de Chien, 67
  Séparation du support, 55
  Schaub, 32, 36, 37
  Trace de Berlekamp, 15, 59, 64, 65, 67, 68, 70, 72, 74–76
  Trace de Berlekamp - Zinoviev, 70–74
  Zinoviev, 15, 69, 70, 73
attaque
  algébrique, 40
  corrélation rapide, 102
  Man In the Middle, 88, 91, 94
  par décodage, 62
  structurelle, 62

classe cyclotomique, 22, 24, 27, 31, 35, 37, 39
code
  à zéros constants, 39
  alternant, 27, 28, 62
  auto-orthogonal, 26, 55
  BCH, 25, 28, 31, 33, 62, 64
  cyclique, 24, 27, 31, 36, 41, 42, 45, 48, 51, 55
  dual, 25, 27, 31, 36, 41, 42, 53, 56
  équivalent, 47, 56
  étendu, 26, 48, 53
  Goppa classique, 59, 62, 63
  GRS, voir Reed-Solomon généralisé
  poinçonné, 26, 48, 56
  Reed-Muller cyclique, 25, 48
  Reed-Solomon généralisé, 28
  trace, 27
complexité linéaire, 36, 37, 40
critère d'élagage, 32, 36, 41, 42
cryptographie ultralégère, 83
cryptosystème
  de type McEliece, 59, 62, 67, 70, 74, 107
  Niederreiter, 62

décodage
  code Goppa classique, 59, 63
  code linéaire, 99, 100
  par ensemble d'information, 100
distribution de poids, 31, 53–55

ensemble de définition, 24, 27, 31, 34, 36, 41, 42, 45, 49, 52, 56
entrelacement, 47
énumérateur des poids, 53, 56
équivalence de codes, voir code équivalent
extracteur d'aléas, 94

groupe
  affine, 19
  d'automorphismes, 48, 56
  d'entrelacements, 47
  multiplement transitif, 18, 19, 48, 56

hull d'un code linéaire, 26, 55

immunité
algébrique, 40
spectrale, 32, 40, 41

low-cost, 80, 83, 84, 94, 107
LPN-solver, 101, 102, 108

Machine Learning, 86
matrice de Toeplitz, 89

paradoxe des anniversaires généralisé, 102, 103
polynôme
affine, 69, 72
de Goppa, 61, 64
de rétroaction, 94
linéarisé, 69
localisateur d'erreurs, 59, 63, 64, 70
Mattson-Solomon, 34, 37
minimal, 15, 19, 22, 24, 25, 31, 41
réciproque, 26, 53
preuves de sécurité, 104, 105
problème
décodage borné, 99, 101
décodage complet, 100
distance minimale, 99
LPN, 85, 86, 88, 90, 103, 105–108
Syndrome Decoding, 99, 100, 103, 105, 106
protocole
HB, 79, 84–86, 103
HB+, 87, 88, 91, 93
HB-MP, 94
HB-MP+, 94–96
HB#, 89–91
Improved HB-MP, 95
PUF-HB, 97
Random HB#, 89, 90
Trusted-HB, 93

RFID, 79–81, 83, 84, 89, 96, 97, 107, 109

sécurité prouvée, 86, 104
semi-anneau, 18, 37
signature d'un code, 56
sous-code sur un sous-corps, 27, 41, 61
support
code, 21, 47, 55
code de Goppa, 61, 64

tag, 79, 83–89, 93, 95, 96, 105

workfactor, 103, 104