Etude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques

Julia Chaulet

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Julia CHAULET

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Étude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques

soutenue le 30 mars 2017

devant le jury composé de :

Nicolas SENDRIER	Inria Paris	Directeur de thèse
Philippe GABORIT	Université de Limoges	Rapporteur
Pierre LOIDREAU	DGA-MI, IRMAR, Rennes	Rapporteur
Jean-Claude BAJARD	UPMC Paris	Examinateur
Caroline FONTAINE	CNRS et IMT Atlantique	Examinatrice
Jean-Pierre TILLICH	Inria Paris	Examinateur
Éric GARRIDO	Thales TCS	Invité

Étude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques

Julia Chaulet

Sous la direction de Nicolas Sendrier

Équipe-Projet SECRET
Inria de Paris,
2 rue Simone IFF,
75 012 Paris

Laboratoire Chiffre
Thales Communication & Services,
4 Avenue des Louvresses,
92 230 Gennevilliers

Remerciements

Après ces 3 ans de thèse, la liste des personnes à remercier me semble interminable... Cette thèse fût pour moi une aventure extraordinaire, que j'ai eu la chance de partager avec des personnes formidables et qui ne se résume pas seulement aux résultats scientifiques qui sont présentés dans ce manuscrit.

D'avance, je tiens à m'excuser auprès de ceux qui ne trouverons pas leur nom sur ces pages : n'hésitez pas à me solliciter, je vous remercierai de vive voix.

Tout d'abord, je tiens à remercier Nicolas, mon directeur de thèse, qui a largement contribué au bon déroulement de cette thèse. Que ce soit sur le plan scientifique ou humain, j'ai toujours senti beaucoup de bienveillance et de soutiens de ta part. Merci sincèrement pour la confiance dont tu as fais preuve à mon égard et les innombrables heures que tu as consacré à m'expliquer du plus simple au plus élaboré des concepts.

Un immense merci à Éric, qui a porté cette thèse au sein de Thales et sans qui je n'aurai pas eu la chance de vivre cette aventure dans de si bonnes conditions. Merci pour ta gentillesse, ton soutien, ta confiance et pour l'enthousiasme et la curiosité dont tu as toujours fait preuve concernant les travaux que j'ai réalisé durant cette thèse.

Je tiens à remercier chaleureusement Philippe Gaborit et Pierre Loidreau qui ont accepté de prendre le temps, bien qu'ayant des emplois du temps plus que chargé, de rapporter ce manuscrit. J'aimerai aussi remercier les autres membres de mon jury : Jean-Claude Bajard, Caroline Fontaine et Jean-Pierre Tillich, qui ont accepté d'être là aujourd'hui.

En remontant le fil du temps, tout cela n'aurai pas eu lieu sans le soutien de Sébastien Canard. Merci de m'avoir offert l'opportunité d'effectuer mon stage de fin d'étude à Caen. Malgré le climat normand, j'ai tellement apprécié ces 6 mois, que j'ai décidé de m'engager dans cette voie pour 3 années supplémentaires.

Je tiens à remercier les permanents de l'équipe SECRET : Anne, Marìa, Pascale, Gaëtan, André, Anthony. Il ne fait aucun doute que c'est en grande partie grâce à vous si il fait bon vivre et "bon travailler" dans cette équipe !

J'aimerai aussi remercier les personnes avec qui j'ai pu collaborer au cours de ma thèse. Tout d'abord Jean-Pierre, merci pour la patience dont tu as fais preuve à mon égard et pour toute l'aide que tu m'as apportée lors des (très) nombreuses répétitions de présentation. Je tiens à remercier Vlad

et son humour qui dépasse les frontières des langues ! Merci aussi à Magali et Ayoub, avec qui j'ai partagé, avec plaisir, quelques journées studieuses à Rouen.

Un immense merci à Christelle : ton sourire toujours aux lèvres, ta bonne humeur constante, nos nombreuses discussions... Un rayon de soleil !

Grâce à chacun des membres du laboratoire Chiffre, j'ai eu l'occasion de passer de très bons moments à Gennevilliers. Sonia, tu m'as accompagnée dans le début de thèse à Thales et tu as réussi à me faire comprendre l'importance de se déconnecter, merci ! Olivier O., merci pour ton expertise en problèmes complexes des mathématiques, qui a été pour moi une aide salutaire ! Thomas P., je tiens à te remercier pour les discussions codes/réseaux qui m'ont permis de mieux comprendre la cryptographie post-quantique dans son ensemble. Merci aussi à Philippe, Emelyne, Thomas R., Alexandre, Renaud et David pour leur gentillesse, leur bonne humeur et j'ose dire (pour certains) leur humour toujours raffiné. Merci infiniment à Ange d'avoir relu ce manuscrit, malgré une compilation particulièrement pénible. Tes précieux conseils m'ont permis, sans aucun doute, d'améliorer la qualité du manuscrit.

Merci à toutes les personnes que j'ai croisées au cours de mon séjour à l'Inria, en tant que doctorants, ex-doctorants, post-doctorants ou stagiaires. Je pense à Christina (merci pour ces nombreux conseils avisés), Valentin (je suis encore en stage maintenant tu penses ?), Irene (pour l'ambiance chaleureuse, au sens propre comme au figuré, qui régnait dans notre bureau et pour les nombreux "*ne te stresse pas*" qui m'ont beaucoup aidé), Gregory (grâce à toi, j'ai pu voir Niagara et un feu d'artifice, en une soirée, rien que ça !), Aurélie ("*Me, I want ni of all ! !*"), Joëlle (et ses merveilleux gâteaux), Virginie (et nos *so funny* leçons d'anglais partagées), Audrey (merci pour toute l'aide et le soutien que tu m'as apportée depuis le tout début de la thèse et qui dépasse le cadre du doctorat...).

À leur côté, j'ai franchi la dernière ligne droite. Merci à Rodolfo, de m'avoir supportée en tant que co-bureau trop frileuse. Kaushik, merci pour ces nombreuses discussions qui m'ont aidé à voir un peu plus loin que la thèse. Et plus que tout merci d'avoir perdu ce pari, tu me dois une grosse somme je crois, c'est écrit dans le marbre, ou presque ! Yann, merci d'apporter tant de fraîcheur à la team, notamment grâce à ces chansons de Diams qui nous restent en tête de longues heures. Sébastien, malgré les éclairs au chocolat, merci d'avoir partager de nombreux déjeuner avec moi. Thomas, tu es le plus gentil et intelligent Normalien Agrégé que je connaisse... Oui, tu es le seul ! Plus sérieusement, merci de m'avoir autant aidée en si peu de temps. Kevin, merci de nous supporter avec bonne humeur, ces quelques jours par semaine où tu partages avec nous ce bureau. Vivien, merci pour ton enthousiasme sans faille et ta non exigence humoristique. Antoine, merci pour ton sens de l'organisation inné qui nous as tous fait passer de très bons moments avec l'équipe. Xavier, merci d'avoir supporter mon anglais douteux pendant ces quelques *lessons*.

Que serait une thèse si il n'y avait pas eu de master ? Margaux, merci pour ton soutien durant ces deux années, que nous avons quasiment vécu, littéralement, ensemble. Tes réserves illimités de gâteaux, chocolats et boissons diverses m'ont aidée à ne pas devenir folle ! Merci aussi d'avoir joué le rôle décisif d'entremetteuse entre cette thèse et moi. Merci à Romain (même si tes cheveux sont plus longs que les miens), Yvan (du rêve) et Valentin, qui ont ensoleillé mes études malgré le climat breton.

Nicolas m'a souvent répété que, ce manuscrit, c'était *mon* manuscrit. Ceux qui me connaissent savent que je suis plutôt sentimentale. L'occasion était trop belle !

Par le plus grand des hasards, nous avons suivi la même voie, mais alors que j'ai les yeux rivés sur la théorie, toi, tu mets les mains dans le cambouis et tu fais en sorte que tout ça fonctionne bien et soit utilisé. Quentin, merci de m'avoir hébergé, d'avoir toujours été disponible et de m'avoir fait comprendre l'importance de la cryptographie dans le "vrai monde". En tant que cryptopathe, je n'aurai jamais imaginé qu'il y ait tant de moyens de contourner la sécurité de nombreux systèmes.

Je tiens à remercier tous les antibois, valbonais, roquefortois (et autres communes de moins de 5 000 habitants) qui m'ont fait me sentir à la maison même dans ce milieu hostile qu'est Paris. Merci à Pedro, Flav, Xav, Stouf, Emi, Djé, Vio, Charlotte, Romain, Camille, Luca, Thomas, Ouai Stéphane, Allan… et notre bon vieil ami le *pastaga*. Certains d'entres eux sont dans la salle aujourd'hui et ont probablement ouvert ce manuscrit : "Ho quels magnifiques dessins ! !". C'est vrai, ils sont magnifiques ! Merci infiniment d'être venu : du fin fond de la salle ou de l'autre bout du monde, vous êtes des potos géniaux…

Merci aussi à Andreea, *sunshine*. Avec toi, j'ai pu découvrir de magnifiques endroits, œuvres, musiques etc. Merci d'avoir rêvé, de continuer à rêver et j'espère, d'ici peu, de rêver avec moi à l'autre bout du monde.

Lucas, tu m'as accompagnée d'Antibes à Rennes, de Rennes à Caen, puis de Caen à Paris. Et plus que ça, tu m'as soutenue, épaulée, quelques fois supportée aussi, durant toutes ces a..ventures et j'espère durant beaucoup d'autres encore. Je ne vais pas m'étendre ici sur le rôle que tu as joué tout au long de cette thèse, tu auras compris que quelques mots ne suffiraient pas… Quant au Minot, Fanny, MJ, Gib, Lélé, Juju et toute la *smalla* : j'ai partagé tellement de bons moments avec vous, merci infiniment de vous impliquer autant et avec autant de bienveillance, de générosité et de gentillesse.

Benjamin, tu as suivi cette aventure de loin, mais bien souvent il m'a suffit de me rappeler de nos discussions pour surmonter les moments difficiles (d'ailleurs, Lola veut des cousins !) et apprécier les meilleurs. Charlotte, ma *sista'*, le *masking tape* de la famille : tu nous protèges, nous unis et tu nous rends plus beaux ! Merci pour tout… Prune et Lola, vous m'avez fait découvrir Nekfeu, la lessive/boule-relaxante, l'encre à lèvre et tant d'autres nouveautés qui me permettent de garder les pieds sur terre et de me sentir de plus en plus jeune. Merci mes nièces préférées ! Et merci à toutes les 3 pour ces week-ends Cagoles m'ont permis d'arborer un bronzage parfait tout au long de l'année… J'ai une pensée très émue pour Philippe, que j'ai trop peu connu. Grâce aux nombreuses photos que tu nous as laissées, j'ai pu voir à travers tes yeux : merci de m'avoir montré un monde aussi magnifique.

Merci à mon père, Françis : *"[...] de diviser chacune des difficultés que j'examinerais en autant de parcelles qu'il se pourrait, et qu'il serait requis pour les mieux résoudre"*. Ce bon vieux Descartes ! Je sais qu'a tes yeux, cette gratitude est un peu comme les anniversaires, Noël ou jouer aux dés. Merci pour ça aussi…

Grazie mille a te, mamina. É gracie a te se sono qui oggi. Tu m'as soutenue, épaulée, poussée, tu as toujours cru en moi et m'a donnée confiance en moi. Tu as *toujours* été là pour moi… Tu me diras sûrement que c'est ton rôle, en tant que maman, mais c'est un rôle que tu maîtrises parfaitement.

Table des matières

Notations **7**

Introduction **11**

I Préliminaires **15**

1 Code correcteur d'erreurs **17**

1.1 Code linéaire binaire 18

1.1.1 Description d'un code 18

1.1.2 Décodeurs 21

1.1.3 Opérations sur les codes 21

1.2 Code quasi-cyclique 23

2 Cryptographie fondée sur les codes linéaires **27**

2.1 Cryptosystème de McEliece 29

2.2 Cryptosystème de Niederreiter 30

2.3 Sécurité 31

2.3.1 Problèmes difficiles de la théorie des codes 31

2.3.2 Réductions de sécurité 32

2.3.3 Attaques génériques 36

2.4 Choix d'une famille de codes 39

II Les codes QC-MDPC **43**

Introduction **45**

3 Généralités **47**

3.1 Définition 47

3.2 Les codes QC-MDPC pour la cryptographie 48

4 Primitives cryptographiques utilisant les codes QC-MDPC **49**

4.1 Cryptosystème de McEliece 50

4.1.1 Description du cryptosystème 50

4.1.2 Sécurité 53

4.1.3 Paramètres proposés 55

4.2 Cryptosystème de Niederreiter 56

4.3 Schéma d'échange de clé 57

5 Algorithme de *bit flipping* **59**

5.1 Convention 60

5.2 Fonctionnement et description de l'algorithme 61

5.3 Résultats observés pour les codes QC-MDPC 69

5.4 Conclusion 74

5.5 Mise en œuvre : état de l'art 74

III Analyse du décodage **77**

Introduction **79**

6 Amélioration du calcul du seuil **81**

6.1 Calcul du seuil 81

6.1.1 Probabilité sur les équations de parité 82

6.1.2 Probabilité qu'une position de l'erreur soit modifiée 83

6.1.3 Seuil optimal 86

6.1.4 Distribution du poids de l'erreur 88

6.1.5 Distribution du poids du syndrome 90

6.1.6 Calcul de la variance 92

6.1.7 Conclusion 94
6.2 Pires cas pour le décodage 95
6.3 Calcul adaptatif du seuil 101
6.3.1 Détails du processus 101
6.3.2 Résultats et conclusion 105

7 Analyse théorique **107**
7.1 Objectifs 107
7.2 Grandeurs fondamentales de l'algorithme 108
7.3 Distributions à la première itération 111
7.4 Résultats observés pour différents paramètres 117
7.4.1 Pour une sécurité de 80 bits 117
7.4.2 Pour une sécurité de 128 bits 119

Conclusions et perspectives **123**

IV Cryptanalyse d'un schéma de chiffrement de McEliece utilisant les codes polaires **127**

Introduction **129**

8 Codes polaires **131**
8.1 Construction 131
8.2 McEliece avec les codes polaires 132

9 Équivalence de codes **135**
9.1 Code permuté 135
9.2 Résolution du problème d'équivalence de codes 137

10 Codes monomiaux décroissants **141**
10.1 Codes de Reed-Muller 141
10.2 Définition 143
10.3 Propriétés 145
10.3.1 Description du dual 145
10.3.2 Groupe de permutation 148

10.3.3 Distance minimale 149

11 Cryptanalyse du schéma de McEliece proposé dans [SK14] 153

11.1 Description de l'attaque 155

11.1.1 Recherche des mots de poids minimal 155

11.1.2 Signature des orbites dans l'ensemble des mots de poids minimal du code . . 155

11.1.3 Calcul des orbites dans l'ensemble des mots de poids minimal du code permuté 158

11.1.4 Identification des espaces affines 158

11.1.5 Problème d'équivalence de codes pour un petit code monomial décroissant . 159

11.1.6 Étape d'induction 160

11.2 Implémentation de l'attaque 161

Conclusions et perspectives 163

Bibliographie 164

Table des figures 173

Liste des tableaux 177

Notations

Notations génériques

— $\mathbb{F}_2$ le corps fini à deux éléments
— $\mathcal{C}$ un code linéaire binaire
— $\dim(\mathcal{C})$ la dimension du code $\mathcal{C}$
— $\mathcal{C}^\perp$ le dual du code $\mathcal{C}$
— $\mathcal{H}(\mathcal{C})$ le hull de $\mathcal{C}$
— $\mathcal{P}_\mathcal{I}(\mathcal{C})$ le poinçonné de $\mathcal{C}$ sur l'ensemble $\mathcal{I}$
— $\mathcal{S}_\mathcal{I}(\mathcal{C})$ le raccourci de $\mathcal{C}$ sur l'ensemble $\mathcal{I}$
— M^t la transposée de la matrice M
— I_k la matrice identité de taille $k \times k$
— $\langle \boldsymbol{v}, \boldsymbol{v}' \rangle$ le produit scalaire des vecteurs $\boldsymbol{v}$ et $\boldsymbol{v}'$ dans $\mathbb{F}_2$
— $|E|$ le cardinal de l'ensemble E
— $E \sqcup F$ l'union disjointe des ensembles E et F
— $\mathrm{Ber}(p)$ la loi de Bernouilli de paramètre p
— $\mathcal{B}(N, p)$ la loi binomiale de paramètres N et p
— $\mathrm{wt}(\boldsymbol{v})$ le poids de Hamming du vecteur $\boldsymbol{v}$
— $\mathrm{Supp}(\boldsymbol{v})$ le support du vecteur $\boldsymbol{v}$

Parties II et III

— $\mathcal{H}(n, r, w)$ l'ensemble des matrices de parités creuses qui décrivent un code $[n, r, w]$-MDPC
— $\mathcal{H}_{\mathrm{QC}}(n, r, w)$ l'ensemble des matrices de parités creuses qui décrivent un code $[n, r, w]$-QC-MDPC
— $\mathcal{U}_t$ l'ensemble des lois uniformes sur $\{\boldsymbol{v} \in \mathbb{F}_2^n \ : \ \mathrm{wt}(\boldsymbol{v}) = t\}$

— $\mathrm{Circ}(\mathbb{F}_2^r)$ l'ensemble des matrices circulantes de $\mathbb{F}_2^r$
— $\mathcal{R}$ l'anneau de polynôme $\mathbb{F}_2[X]/(X^r - 1)$
— φ l'isomorphisme entre $\mathrm{Circ}(\mathbb{F}_2^r)$ et $\mathcal{R}$

Notations spécifiques : à partir du chapitre 5

— $\boldsymbol{v}$ est confondu avec le support de $\boldsymbol{v}$
— $|\boldsymbol{v}|$ le poids de Hamming du vecteur $\boldsymbol{v}$
— $\boldsymbol{v} \cap \boldsymbol{v}'$ le vecteur dont le support est l'intersection des supports de $\boldsymbol{v}$ et $\boldsymbol{v}'$
— $\boldsymbol{h}_i$ la i-ème équation de parité définie par H
— $\boldsymbol{h}_{*,j}$ la transposée de la j-ème colonne de la matrice H

Partie IV

— $A \otimes B$ le produit de Kronecker des matrices A et B
— $\mathcal{R}(r, m)$ le code de Reed-Muller d'ordre r en m variables
— S_n le groupe symétrique d'indice n
— $\mathbb{R}_2[x_0, \ldots, x_{m-1}]$ l'anneau de polynôme $\mathbb{F}_2[x_0, \ldots, x_{m-1}]/(x_0^2 - x_0, \ldots, x_{m-1}^2 - x_{m-1})$
— $\mathsf{ev}(g)$ l'évaluation du polynôme g sur $\mathbb{F}_2^m$
— $\deg(g)$ le degré du polynôme g
— $\mathcal{M}$ l'ensemble des monômes de $\mathbb{R}_2[x_0, \ldots, x_{m-1}]$
— $\check{f}$ le complémentaire du monôme f
— $\mathbb{LTA}_m$ le groupe affine triangulaire inférieur
— $\mathcal{O}_{\boldsymbol{x}}^{\mathcal{G}}$ l'orbite de $\boldsymbol{x}$ selon l'action de $\mathcal{G}$

Introduction

Les travaux présentés dans ce manuscrit sont le résultat de mes trois années de thèse effectuées à l'Inria Paris sous la direction de Nicolas Sendrier de février 2014 à 2017. Cette thèse a été financée par une bourse CIFRE au sein de Thales TCS et sous la tutelle d'Éric Garrido.

Dans ce manuscrit, nous allons présenter différents résultats autour de la cryptographie fondée sur les codes correcteurs d'erreurs.

Communément, la cryptographie est définie comme l'art de protéger les informations. Au contraire, la cryptanalyse, elle, regroupe les techniques introduites afin de passer outre ces protections et de retrouver les informations secrètes. Ces deux disciplines sont regroupés sous le terme générique de cryptologie. Historiquement, cryptographie et cryptanalyse ont vastement été employées et étudiées pour et par les armées afin, d'un côté, d'assurer la confidentialité des transmissions et de l'autre, de collecter des informations sensibles sur ses adversaires. Aujourd'hui, le développement de nombreux outils numériques, que ce soit pour la communication, le stockage d'informations etc. et la généralisation de leur emploi, font de la cryptologie l'une des disciplines phares de l'informatique et des mathématiques. En effet, afin d'assurer la confidentialité, l'authentification ou l'intégrité d'une conversation (téléphone, email, etc.) ou de données (données personnelles, cloud, etc.), il est nécessaire de recourir à la cryptographie. De nos jours, les primitives cryptographiques se spécialisent d'autant plus que les contextes nécessitant leur emploi se diversifient.

Il est d'usage de distinguer deux grandes familles de primitives : la cryptographie à clé secrète (ou symétrique), qui nécessite au préalable que les deux parties échangent un secret (une clé) et la cryptographie à clé publique (ou asymétrique), qui utilise une paire de clés. Dans ce dernier cas, chaque partie possède un couple de clés dont l'une est connue de son seul utilisateur (la clé privée), tandis que l'autre est rendue publique (la clé publique). Son fonctionnement s'apparente à celui d'un annuaire : étant donné la clé publique ou le numéro d'un utilisateur, nous pouvons lui communiquer des informations, qu'il ne pourra retrouver que s'il possède la clé privée associée, ou la bonne ligne téléphonique. Ces deux types de cryptosystèmes répondent donc à des problématiques très différentes. En premier lieu, la cryptographie à clé publique est, la plupart du temps, utilisée pour échanger une clé qui sera par la suite exploitée en tant que secret (clé) dans un cryptosystème symétrique. En effet, de manière générale, la cryptographie à clé secrète fournit des algorithmes plus efficaces en terme de temps et de stockage que la cryptographie à clé publique.

Largement étudiée et développée, la cryptologie fait aujourd'hui face à un nouveau bouleverse-

ment : l'ordinateur quantique, dont l'emploi remet en question la sécurité d'une partie des cryptosystèmes utilisés de nos jours. Il faut donc trouver de nouvelles solutions et plus particulièrement pour la cryptographie à clé publique. En effet, il s'avère que la sécurité de la plupart des cryptosystèmes à clé publique mis en œuvre aujourd'hui repose sur l'hypothèse que certains problèmes issus de la théorie des nombres sont difficiles à résoudre (par exemple, la factorisation d'un grand nombre en ses facteurs premiers). Certains algorithmes spécifiquement introduits pour l'ordinateur quantique résolvent une partie de ces problèmes de manière efficace. Il est donc nécessaire de trouver d'autres solutions, qui, si un ordinateur quantique est construit, permettraient toujours, entres autres, d'assurer la confidentialité des données. Les réseaux euclidiens, les fonctions de hachages, les codes correcteurs d'erreurs ou les polynômes multivariés offrent de bonnes garanties en ce sens. Dans ce manuscrit, nous allons plus particulièrement nous intéresser aux solutions qui utilisent les codes correcteurs d'erreurs.

Les codes correcteurs d'erreurs, contrairement à la cryptographie, n'ont pas pour but d'assurer la confidentialité des données, mais la transmission fiable de celles-ci. En d'autres termes, ils permettent de s'assurer que, malgré le fait que des perturbations quelconques puissent altérer le message lors de sa transmission, le message reçu contiendra assez d'informations pour retrouver le message initial. Ils sont, par exemple, utilisés en télécommunications, ou pour le stockage de données, depuis de nombreuses années.

Le premier cryptosystème fondé sur les codes correcteurs d'erreurs a été introduit en 1978 par McEliece et porte son nom. La sécurité de ce cryptosystème repose, en partie, sur un problème dur de la théorie des codes qui ne semble pas pouvoir être résolu rapidement grâce à un ordinateur quantique. Nous avons donc de bonnes raisons de penser que l'utilisation de ces primitives reste sûre même si un adversaire possède un ordinateur quantique. Cette propriété fait de ces primitives des cryptosystèmes dits post-quantiques.

Dans ce manuscrit, nous étudierons en détails la proposition de [MTSB12], qui suggère l'utilisation des codes MDPC (*Moderate Density Parity Check codes*) quasi-cycliques dans le cryptosystème de McEliece. Grâce à l'emploi de ces codes, la taille des clés est très raisonnable (la taille de la clé publique est environ 5 kbits pour 80 bits de sécurité). De plus, nous verrons que, dans ce cas, l'utilisation d'une structure quasi-cyclique offre toujours de bonnes garanties concernant la sécurité des primitives cryptographiques (contrairement à la plupart des propositions qui considèrent des codes quasi-cycliques). Enfin, les algorithmes de chiffrement et de déchiffrement sont très efficaces et ne nécessitent que des opérations binaires.

Les travaux effectués durant cette thèse concernent plus particulièrement les propriétés de l'algorithme de décodage appliqué lors de l'étape de déchiffrement. Les motivations de cette étude sont multiples : à l'origine, cet algorithme de décodage a été introduit pour les codes LDPC (*Low Density Parity Check codes*) dont la structure est quasiment la même que celle des codes MDPC. Il a donc fallu reprendre l'analyse théorique de cet algorithme, explicitée par Gallager ([Gal63]), dans ce cas particulier, c'est-à-dire lorsque l'étape de décodage est appliquée pour déchiffrer et non pour corriger une erreur provenant d'un canal de transmission. La première différence réside donc dans le modèle de l'erreur considérée. La deuxième différence est que, pour assurer la sécurité des primitives proposées, nous devons analyser les risques liés aux attaques par canaux cachés. Ces risques peuvent, par exemple, découler des opérations qui doivent être effectuées sur des données sensibles. Il faut donc, dans un premier temps, définir l'ensemble des données auxquelles l'adversaire a accès, et dans quelle mesure ces données peuvent être exploitées pour retrouver des informations

sensibles. Un autre facteur de risque est le fait que le décodage puisse échouer. En effet, ces échecs au décodage peuvent fournir des informations sur le motif d'erreur et la clé privée, qui sont des données sensibles. L'objectif de l'analyse du comportement du décodage est donc aussi d'estimer précisément la probabilité d'échec au décodage de sorte à pouvoir évaluer la sécurité du cryptosystème contre ce type d'attaques. Dans le même temps, cette analyse pourrait être exploitée pour optimiser l'algorithme de décodage grâce à une meilleur compréhension de son fonctionnement. Ces résultats ont fait l'objet d'une publication : [CS16].

Dans un second temps, nous étudierons la proposition [SK14] qui suggère l'emploi d'une autre famille de codes, appelée codes polaires, pour le cryptosystème de McEliece. Nous verrons que, au vu de nos résultats, ce cryptosystème n'est pas sûre. Plus particulièrement, nous expliciterons un algorithme pour retrouver la clé privée à partir de la clé publique. Dans cette proposition, la clé privée est une permutation tandis que la clé publique est un code polaire permuté. En effet, les codes polaires, comme les codes de Reed-Muller, sont définis de manières uniques à paramètres fixés (et à permutation des coordonnées près). Pour retrouver la clé privée, nous avons donc du reconstruire la permutation secrète, à partir de la donnée du code (défini par les paramètres du cryptosystème) et de son permuté. Pour ce faire, nous avons recouru à de nouvelles techniques afin de résoudre une équivalence de codes. En effet, l'algorithme le plus efficace pour le résoudre n'étant pas assez performant dans ce contexte, nous avons du examiner les propriétés de ces codes afin de trouver un moyen de l'adapter. Ceci nous a permis de découvrir de nombreux liens entre les codes polaires et les codes de Reed-Muller. Ainsi nous introduirons une nouvelle famille de codes : les codes monomiaux décroissants, dont les codes de Reed-Muller et (certains) codes polaires font partie. Ces résultats sont donc aussi d'un intérêt indépendant pour la théorie des codes. Ces travaux ont aussi été publiés dans une conférence internationale (voir [BCD+16]).

Organisation du manuscrit

Dans la partie I, nous commencerons par rappeler les notions de base de la théorie des codes correcteurs d'erreurs (chapitre 1) et de la cryptographie (chapitre 2).

Nous spécialiserons notre étude sur les codes MDPC dans la partie II. Nous commencerons par introduire leur définition dans le chapitre 3. Ensuite, nous étudierons différentes primitives cryptographiques faisant appel à ces codes dans le chapitre 4. Enfin, nous détaillerons l'algorithme de décodage *bit flipping*, utilisé comme décodeurs pour les codes MDPC en cryptographie, dans le chapitre 5.

La partie III traitera de l'analyse de cet algorithme dans un contexte cryptographique. Nous verrons, dans le chapitre 6, comment améliorer l'efficacité du décodage et dans le chapitre 7, nous expliquerons les raisons pour lesquelles les améliorations proposées fonctionnent.

La dernière partie (partie IV) fournira les détails de la cryptanalyse d'un schéma de chiffrement de McEliece utiliant les codes polaires. Nous commencerons par définir cette famille de codes dans le chapitre 8. Ensuite, nous nous intéresserons au problème d'équivalence de codes et à sa résolution dans le chapitre 9. Le chapitre 10 nous permettra d'introduire une nouvelle famille de codes : les codes monomiaux décroissants et leur propriété. Enfin, nous terminerons cette partie par la description de la cryptanalyse du schéma de McEliece utilisant les codes polaires dans le chapitre 11.

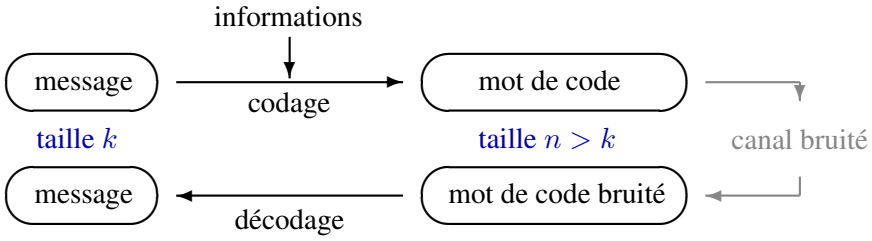
Partie I

Préliminaires

1 Code correcteur d'erreurs

Lorsqu'une information est transmise à travers un canal vers un récepteur, par exemple en télécommunication, la fiabilité de la transmission est souvent difficile à garantir. En effet, dans de nombreux contextes, il faut prendre en compte le fait que le canal utilisé génère des erreurs ou des effacements. En d'autres termes, ces canaux peuvent être bruités. Les codes correcteurs d'erreurs sont un outil qui permet d'assurer la transmission fiable d'informations à travers un tel canal. Le principe fondamental est simple, puisqu'il s'agit d'ajouter de la redondance au message transmis, de sorte que cet excédent d'information aide à retrouver le message initial. Il existe de nombreuses façons de construire une telle redondance. Dans ce manuscrit, nous nous intéresserons aux codes correcteurs d'erreurs linéaires qui, comme nous allons le voir, sont des sous-espaces vectoriels. Leur principal intérêt est qu'ils permettent de former un message de taille n, à partir d'un message de taille k (avec $k \leqslant n$) en exploitant les relations linéaires définies par la structure du code. Ces relations linéaires sont propres au code considéré et offrent la possibilité de retrouver l'information initiale lorsque le code est connu.

L'étape d'expansion du message, qui consiste à ajouter de l'information, est appelée codage. Elle permet de transformer un message de taille k en un mot de code, c'est-à-dire un vecteur de taille n appartenant au sous-espace vectoriel défini par le code considéré. Le vecteur reçu en sortie de canal est appelé mot de code bruité, puisqu'il est composé d'un mot de code et du bruit produit par le canal. L'étape finale, qui consiste à retrouver le message initial à partir de ce mot de code bruité, est appelé décodage.



La modélisation des canaux est un enjeu majeur pour la théorie de l'information afin de comprendre les propriétés du bruit qu'ils vont produire et de l'estimer. En effet, le choix de l'algorithme de

décodage et le bon fonctionnement de celui-ci dépendent du modèle du canal utilisé. Dans ce manuscrit, nous aurons besoin de définir le modèle du canal binaire symétrique. Ce canal n'admet en entrée et en sortie que des valeurs binaires et est caractérisé par la probabilité p qu'un bit soit modifié lors de sa transmission dans ce canal, comme le montre la figure 1.1. Dans la suite du manuscrit, un tel canal sera noté $\mathrm{BSC}\,(p)$ où p est défini comme ci-dessous.

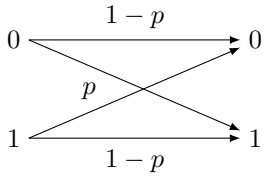


FIGURE 1.1 – Canal binaire symétrique de probabilité d'erreur p

Propriété 1 (Canal binaire symétrique)**.** *Soit $b_e \in \{0,1\}$ le bit émis à travers un canal $\mathrm{BSC}\,(p)$ et $b_r \in \{0,1\}$ le bit reçu en sortie de ce canal. On a*

*(*i) *pour $b \in \{0,1\}$, $\mathbb{P}[b_r = b | b_e = b] = 1 - p$,*

*(*ii) *pour $b \in \{0,1\}$, $\mathbb{P}[b_r = b \oplus 1 | b_e = b] = p$*

1.1 Code linéaire binaire

1.1.1 Description d'un code

Dans cette section, nous définirons les codes linéaires binaires, c'est-à-dire à valeur dans $\mathbb{F}_2$, le corps fini à deux éléments. De plus, nous donnerons quelques outils qui nous aiderons à caractériser ces codes et leurs éléments.

Définition 1 (Code linéaire binaire)**.** *Un code linéaire binaire $\mathcal{C}$ de longueur n et dimension k est un sous-espace vectoriel engendré par k vecteurs libres de $\mathbb{F}_2^n$.*

Remarque 1. *Dans la suite, un code linéaire binaire $\mathcal{C}$ de longueur n et dimension k sera appelé $[n,k]$-code linéaire. De plus, nous noterons $\dim(\mathcal{C})$ la dimension de $\mathcal{C}$.*

Définition 2 (Taux de transmission)**.** *Le taux de transmission d'un $[n,k]$-code linéaire est*

$$R \stackrel{def}{=} \frac{k}{n}$$

En tant que sous-espace vectoriel, un code linéaire est donc entièrement décrit grâce à une base de vecteurs. Pour des raisons pratiques, une telle base sera donnée sous la forme d'une matrice, appelée matrice génératrice du code.

Définition 3 (Matrice génératrice)**.** *Soient $\mathcal{C}$ un $[n,k]$-code linéaire et $G \in \mathbb{F}_2^{k\times n}$ une matrice génératrice de $\mathcal{C}$, alors*

$$\mathcal{C} = \{\boldsymbol{m}G \ : \boldsymbol{m} \in \mathbb{F}_2^k\}$$

Remarque 2. *Pour $\mathcal{C}$ un $[n,k]$-code linéaire, toute matrice formée de k vecteurs libres et générateurs de $\mathcal{C}$ est une matrice génératrice de $\mathcal{C}$.*

Définition 4 (Forme systématique)**.** *Soient $\mathcal{C}$ un $[n,k]$-code linéaire et $G \in \mathbb{F}_2^{k\times n}$ une matrice génératrice de $\mathcal{C}$. G est dite sous forme systématique si elle s'écrit $G = [I_k|G']$ où I_k désigne la matrice identité de taille k.*

L'opération qui consiste à multiplier un vecteur de $\mathbb{F}_2^k$ par la matrice génératrice d'un code est appelée codage et le vecteur du code ainsi construit est appelé mot de code.

Une autre façon de décrire un code est de le considérer comme le noyau d'une application linéaire. La description de cette application linéaire, grâce à une matrice dite de parité, fournit une définition équivalente du code.

Définition 5 (Matrice de parité)**.** *Une matrice de parité $H \in \mathbb{F}_2^{(n-k)\times n}$ de $\mathcal{C}$ un $[n,k]$-code linéaire est telle que*

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{F}_2^n \ : \ \boldsymbol{x}H^t = 0\}$$

Remarque 3. *Soit $\mathcal{C}$ un $[n,k]$-code linéaire. Pour toute matrice génératrice $G \in \mathbb{F}_2^{k\times n}$ et toute matrice de parité $H \in \mathbb{F}_2^{(n-k)\times n}$ de $\mathcal{C}$ on a*

$$GH^t = 0$$

La donnée d'une matrice génératrice sous forme systématique permet de déduire, de façon immédiate, une matrice de parité sous forme systématique du même code.

Propriété 2. *Soient $\mathcal{C}$ un $[n,k]$-code linéaire et $G = [I_k|G']$, avec $G' \in \mathbb{F}_2^{k\times(n-k)}$, une matrice génératrice de $\mathcal{C}$ sous forme systématique. La matrice $H = [G'^t|I_{n-k}] \in \mathbb{F}_2^{(n-k)\times n}$ est une matrice de parité de $\mathcal{C}$ sous forme systématique.*

Une matrice de parité d'un code définit elle aussi une base de vecteurs libres. En ce sens, elle permet aussi d'engendrer un code, appelé code dual.

Définition 6 (Code dual)**.** *Soient $\mathcal{C}$ un $[n,n-r]$-code linéaire et $H \in \mathbb{F}_2^{r\times n}$ une matrice de parité de $\mathcal{C}$. Le code dual de $\mathcal{C}$, noté $\mathcal{C}^\perp$, est le $[n,r]$-code linéaire engendré par la matrice H.*

Propriété 3. *Soient $\mathcal{C}$ un code linéaire de longueur n et $\mathcal{C}^\perp$ son dual. Alors,*

$$\forall \mathbf{c} \in \mathcal{C} \text{ et } \forall \mathbf{c}' \in \mathcal{C}^\perp, \text{ on a } \langle \mathbf{c}, \mathbf{c}' \rangle = 0$$

où $\langle \mathbf{c}, \mathbf{c}' \rangle \overset{def}{=} \sum_{j=1}^{n} c_j c'_j \mod 2$ *est le produit scalaire de* $\mathbf{c}$ *et* $\mathbf{c}'$.

Définition 7 (Codimension)**.** *Soit $\mathcal{C}$ un $[n,n-r]$-code linéaire. La codimension de $\mathcal{C}$ est $n-(n-r) = r$ et correspond à la dimension de $\mathcal{C}^\perp$.*

Définition 8 (Code faiblement auto-dual)**.** *Un code linéaire $\mathcal{C}$ est dit faiblement auto-dual s'il vérifie*

$$\mathcal{C} \subseteq \mathcal{C}^\perp$$

Remarque 4. *Si $\mathcal{C}$ est un code faiblement auto-dual alors $\mathcal{C} \cap \mathcal{C}^{\perp} = \mathcal{C}$.*

La donnée d'une matrice de parité d'un code permet de vérifier, grâce à une opération polynomiale, l'appartenance d'un vecteur au code. Plus généralement, ces matrices sont souvent employées dans les étapes de décodage et notamment parce qu'elles sont impliquées dans le calcul du syndrome d'un vecteur.

Définition 9 (Équation de parité)**.** *Soit $H \in \mathbb{F}_2^{r \times n}$ une matrice de parité de $\mathcal{C}$, un $[n, n-r]$-code linéaire. Les lignes de H sont appelées équations de parité de $\mathcal{C}$.*

Remarque 5. *Soient $\boldsymbol{v} \in \mathbb{F}_2^n$ et $H = (h_{i,j})_{1 \leqslant i \leqslant r,\, 1 \leqslant j \leqslant n} \in \mathbb{F}_2^{r \times n}$ une matrice de parité de $\mathcal{C}$, un $[n, n-r]$-code linéaire. Alors*

$$\boldsymbol{v} \in \mathcal{C} \Leftrightarrow \forall i \in \{1, \ldots, r\},\ \sum_{j=1}^{n} h_{i,j} v_j = 0 \mod 2$$

Autrement dit, $\boldsymbol{v} \in \mathcal{C}$ si, et seulement si, $\boldsymbol{v}$ vérifie les r équations de parité définies par H.

Définition 10 (Syndrome)**.** *Soient $\boldsymbol{v} \in \mathbb{F}_2^n$ et $H \in \mathbb{F}_2^{r \times n}$ une matrice de parité de $\mathcal{C}$ un $[n, n-r]$-code linéaire. Le vecteur $\boldsymbol{s} = \boldsymbol{v}H^t \in \mathbb{F}_2^r$ est appelé syndrome de $\boldsymbol{v}$ (par rapport à H).*

Remarque 6. *La définition 5 permet de déduire que tout mot de code donne un syndrome nul.*

La modélisation du bruit produit par un canal est essentielle pour déterminer la méthode de décodage appropriée. Pour quantifier ce bruit, lorsque celui-ci est binaire (comme dans le modèle du canal binaire symétrique) nous avons besoin de définir une métrique puis une distance. Celles-ci nous aiderons d'une part à estimer la quantité de bruit générée par le canal et d'autre part à quantifier la distance entre le mot de code bruité et le code.

Définition 11 (Poids de Hamming)**.** *Soit $\boldsymbol{v} \in \mathbb{F}_2^n$. Le poids de Hamming de $\boldsymbol{v}$, noté $\mathrm{wt}(\boldsymbol{v})$, est le nombre de ses coefficients non nuls.*

Définition 12 (Distance de Hamming)**.** *Soient $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{F}_2^n$. La distance (de Hamming) entre $\boldsymbol{v}$ et $\boldsymbol{v}'$ est définie comme $d(\boldsymbol{v}, \boldsymbol{v}') \stackrel{def}{=} \mathrm{wt}(\boldsymbol{v}' - \boldsymbol{v})$.*

Définition 13 (Distance minimale)**.** *La distance minimale d'un code est la plus petite distance entre deux mots distincts du code.*

Parallèlement, nous définissons le support d'un vecteur comme les indices des coordonnées non nulles de celui-ci.

Définition 14 (Support d'un vecteur)**.** *Le support de $\boldsymbol{v} \in \mathbb{F}_2^n$ est défini comme*

$$\mathrm{Supp}(\boldsymbol{v}) \stackrel{def}{=} \{j \ :\ v_j = 1\} \subseteq \{1, \ldots, n\}$$

Remarque 7. *Le poids de Hamming d'un vecteur $\boldsymbol{v} \in \mathbb{F}_2^n$ se définit de manière équivalence comme $\mathrm{wt}(\boldsymbol{v}) = |\,\mathrm{Supp}(\boldsymbol{v})|$.*

1.1.2 Décodeurs

Le principal intérêt des codes correcteurs d'erreurs est de fournir un outil permettant de retrouver un mot du code à partir d'un mot de code bruité : le décodeur. De manière plus générale, un décodeur n'est autre qu'une application linéaire, dont l'espace de départ est $\mathbb{F}_2^n$ et l'espace d'arrivée est le code. Cependant, il existe des décodeurs dont les propriétés s'avèrent plus intéressantes que d'autres. Par exemple, certains décodeurs retournent le mot de code le plus proche du mot de code bruité donné en entrée.

Définition 15 (Décodeur)**.** *Soit $\mathcal{C}$ un code linéaire de longueur n. Toute application $\Psi_{\mathcal{C}} : \mathbb{F}_2^n \longrightarrow \mathcal{C}$ est appelée décodeur pour $\mathcal{C}$.*

Il existe un second type de décodeur, dit décodeur par syndrome, qui, à partir du syndrome d'un mot de code bruité et d'une matrice de parité, renvoie le vecteur d'erreur associé au mot de code bruité. En d'autres termes, pour $\mathcal{C}$ un $[n, k]$-code linéaire, $H \in \mathbb{F}_2^{(n-k)\times n}$ une matrice de parité de $\mathcal{C}$, $\boldsymbol{y} \in \mathbb{F}_2^n$ et $\boldsymbol{s} = \boldsymbol{y}H^t \in \mathbb{F}_2^{n-k}$, un décodeur par syndrome retourne $\boldsymbol{e} \in \mathbb{F}_2^n$ tel que $\boldsymbol{e}H^t = \boldsymbol{y}H^t = \boldsymbol{s} \in \mathbb{F}_2^{n-k}$ et $\boldsymbol{e} \neq \boldsymbol{y}$. La donnée d'un décodeur par syndrome permet de construire un décodeur classique puisque si un tel $\boldsymbol{e}$ est trouvé, alors $\boldsymbol{y} - \boldsymbol{e} \in \mathcal{C}$ est un mot du code.

Définition 16 (Décodeur par syndrome)**.** *Soient $\mathcal{C}$ un $[n, n-r]$-code linéaire et $H \in \mathbb{F}_2^{r\times n}$ une matrice de parité $\mathcal{C}$. L'application $\psi_H : \mathbb{F}_2^r \to \mathbb{F}_2^n$ est un décodeur par syndrome pour H si*

$$\forall \boldsymbol{s} \in \mathbb{F}_2^r, \psi_H(\boldsymbol{s})H^t = \boldsymbol{s}$$

Définition 17 (Décodeur à distance bornée)**.** *Soit $\mathcal{C}$ un code linéaire de longueur n. Un décodeur $\Psi_{\mathcal{C}}$ est un décodeur à distance bornée $t \in \mathbb{N}$ si*

$$\forall \boldsymbol{c} \in \mathcal{C} \text{ et } \forall \boldsymbol{y} \in \mathbb{F}_2^n, d(\boldsymbol{c}, \boldsymbol{y}) \leqslant t \Rightarrow \Psi_{\mathcal{C}}(\boldsymbol{y}) = \boldsymbol{c}$$

Définition 18 (Décodeur par syndrome à distance bornée)**.** *Soient $\mathcal{C}$ un $[n, n-r]$-code linéaire, $H \in \mathbb{F}_2^{r\times n}$ une matrice de parité $\mathcal{C}$, $t \in \mathbb{N}$ une borne fixée et $\psi_H : \mathbb{F}_2^r \to \mathbb{F}_2^n$ un décodeur par syndrome pour H. ψ_H est un décodeur par syndrome à distance bornée t si*

$$\forall \boldsymbol{e} \in \mathbb{F}_2^n, \mathrm{wt}(\boldsymbol{e}) \leqslant t \Rightarrow \psi_H(\boldsymbol{e}H^t) = \boldsymbol{e}$$

1.1.3 Opérations sur les codes

Dans certains contextes, il est utile de construire, à partir d'un code donné, d'autres codes dont la définition est similaire mais qui peuvent avoir des propriétés très différentes. Pour ce faire, il existe plusieurs outils, comme le poinçonnage ou le raccourcissement.

Définition 19 (Code poinçonné)**.** *Soient $\mathcal{C}$ un code linéaire de longueur n et $\mathcal{J} \subseteq \{1, \ldots, n\}$. Le code poinçonné de $\mathcal{C}$ par $\mathcal{J}$ est*

$$\mathcal{P}_{\mathcal{J}}(\mathcal{C}) \stackrel{def}{=} \left\{ (c_i)_{i \notin \mathcal{J}} \ : \ \boldsymbol{c} \in \mathcal{C} \right\}$$

Pour poinçonner un code sur un ensemble fixé, il suffit donc de ne considérer que les positions du code qui n'appartiennent pas à cet ensemble.

Définition 20 (Code raccourci)**.** *Soient $\mathcal{C}$ un code linéaire de longueur n et $\mathcal{J} \subseteq \{1, \ldots, n\}$. Le code raccourci de $\mathcal{C}$ par $\mathcal{J}$ est*

$$\mathcal{S}_{\mathcal{J}}(\mathcal{C}) \stackrel{def}{=} \left\{ (c_i)_{i \notin \mathcal{J}} \ : \ \exists\, \boldsymbol{c} = (c_i)_i \in \mathcal{C} \text{ tel que } \forall i \in \mathcal{J},\ c_i = 0 \right\}$$

Afin de raccourcir un code sur un ensemble donné, nous ne conservons que les mots de codes dont le support forme une intersection vide avec cet ensemble, puis nous poinçonnons ces mots de code sur ce même ensemble.

Le raccourci d'un code sur un ensemble donné consiste d'une part, à retirer les mots de codes dont le support appartient à cet ensemble, et d'autre part à supprimer du code les positions qui sont dans cet ensemble.

Sans perte de généralité, si $\mathcal{J} = \{j\}$, nous noterons plus simplement $\mathcal{P}_j(\mathcal{C})$ ou $\mathcal{S}_j(\mathcal{C})$ au lieu de $\mathcal{P}_{\{j\}}(\mathcal{C})$ ou $\mathcal{S}_{\{j\}}(\mathcal{C})$.

Propriété 4. *Soient $\mathcal{C}$ un code linéaire de longueur n et $\mathcal{J} \subseteq \{1, \ldots, n\}$, alors*

$$(\mathcal{S}_{\mathcal{J}}(\mathcal{C}))^{\perp} = \mathcal{P}_{\mathcal{J}}\left(\mathcal{C}^{\perp}\right)$$

Démonstration. Soient $\mathcal{C}$ un code linéaire de longueur n et $\mathcal{J} \subseteq \{1, \ldots, n\}$, nous montrons que :

(i) $\mathcal{P}_{\mathcal{J}}\left(\mathcal{C}^{\perp}\right) \subseteq (\mathcal{S}_{\mathcal{J}}(\mathcal{C}))^{\perp}$.
Soit $\boldsymbol{c} \in \mathcal{S}_{\mathcal{J}}(\mathcal{C})$. Par définition, $\exists\, \boldsymbol{c}' \in \mathcal{C}$ tel que $\forall j \notin \mathcal{J}$, $c'_j = c_j$ et $c'_j = 0$, $\forall j \in \mathcal{J}$. Considérons maintenant $\boldsymbol{y} \in \mathcal{P}_{\mathcal{J}}\left(\mathcal{C}^{\perp}\right)$. On a donc $\exists\, \boldsymbol{y}' \in \mathcal{C}^{\perp}$ tel que $y'_j = y_j$ pour tout $j \notin \mathcal{J}$. Comme $\boldsymbol{c}' \in \mathcal{C}$ et $\boldsymbol{y}' \in \mathcal{C}^{\perp}$, nous savons que $\langle \boldsymbol{c}', \boldsymbol{y}' \rangle = 0$. De plus, $\langle \boldsymbol{c}, \boldsymbol{y} \rangle = \langle \boldsymbol{c}', \boldsymbol{y}' \rangle$. D'où $\langle \boldsymbol{c}, \boldsymbol{y} \rangle = 0$, donc $\boldsymbol{y} \in (\mathcal{S}_{\mathcal{J}}(\mathcal{C}))^{\perp}$.

(ii) $(\mathcal{S}_{\mathcal{J}}(\mathcal{C}))^{\perp} \subseteq \mathcal{P}_{\mathcal{J}}\left(\mathcal{C}^{\perp}\right)$.
Soit $\boldsymbol{c} \in (\mathcal{S}_{\mathcal{J}}(\mathcal{C}))^{\perp}$ et montrons que $\boldsymbol{c} \in \mathcal{P}_{\mathcal{J}}\left(\mathcal{C}^{\perp}\right)$. Par définition, $\forall \boldsymbol{y} \in \mathcal{S}_{\mathcal{J}}(\mathcal{C})$, on a $\langle \boldsymbol{c}, \boldsymbol{y} \rangle = 0$. D'un côté, $\exists\, \boldsymbol{y}' \in \mathcal{C}$ tel que $\forall j \in \mathcal{J}$, $y'_j = 0$ et $\forall j \notin \mathcal{J}$, $y'_j = y'_j$ par définition de $\mathcal{S}_{\mathcal{J}}(\mathcal{C})$.
D'un autre côté, soit $\boldsymbol{c}' \in \mathbb{F}_2^n$ tel que $c'_j = c_j$ pour tout $j \notin \mathcal{J}$. On a alors $\langle \boldsymbol{c}', \boldsymbol{y}' \rangle = \langle \boldsymbol{c}, \boldsymbol{y} \rangle = 0$. Donc $\boldsymbol{c}' \in \mathcal{C}^{\perp}$ (puisque $\boldsymbol{y}' \in \mathcal{C}$) et $c'_j = c_j$ pour tout $j \notin \mathcal{J}$.
En d'autres termes, $\boldsymbol{c} \in \mathcal{P}_{\mathcal{J}}\left(\mathcal{C}^{\perp}\right)$.

□

1.2 Code quasi-cyclique

Dans cette section, nous allons nous intéresser aux codes quasi-cycliques. L'intérêt de ces codes est qu'ils sont entièrement décrits grâce à une seule ligne d'une matrice de parité ou génératrice. En effet, le reste des mots de codes est déduit à l'aide de décalages successifs (cycliques) et des combinaisons linéaires de ceux-ci.

Premièrement, nous définissons les codes cycliques.

Définition 21 (Code cyclique)**.** *Un code $\mathcal{C}$ de longueur n est dit cyclique si, et seulement si,*

$$\forall\, (c_1, c_2, \ldots, c_n) \in \mathcal{C} \Rightarrow (c_n, c_1, \ldots, c_{n-1}) \in \mathcal{C}$$

Autrement dit, dans un code cyclique, tout décalage cyclique d'un mot de code forme encore un mot de code.

Ces codes sont donc engendrés par une matrice construite à partir du décalage successif d'un de ses mots de code.

Définition 22 (Matrice circulante)**.** *Une matrice carrée est dite circulante si elle est construite de sorte que ses lignes soient les décalages cycliques successifs de la première ligne.*

Remarque 8. *Une matrice circulante est entièrement définie par la donnée d'une seule de ses lignes. Dans la suite, une matrice circulante $A \in \mathbb{F}_2^{r\times r}$ sera notée $A = C(a_1, a_2, \ldots, a_r)$ où $(a_1, a_2, \ldots, a_r) \in \mathbb{F}_2^r$ définit la première ligne de A (par convention).*

L'ensemble des matrices circulantes de $\mathbb{F}_2^{r\times r}$ sera noté $\mathrm{Circ}(\mathbb{F}_2^r)$.

Ainsi, $A = C(a_1, \ldots, a_r) \in \mathrm{Circ}\,(\mathbb{F}_2^r)$ si, et seulement si, A est de la forme suivante :

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & \ldots & a_r \\ a_r & a_1 & a_2 & \ldots & a_{r-1} \\ a_{r-1} & a_r & a_1 & \ldots & a_{r-2} \\ & & & \ddots & \\ a_2 & a_3 & a_4 & \ldots & a_1 \end{bmatrix}$$

Propriété 5. *Un code cyclique admet au moins une matrice génératrice (ou de parité) circulante.*

Propriété 6. *Soit $A = C(a_1, a_2, \ldots, a_r) = (a_{i,j})_{1\leqslant i,j\leqslant r} \in \mathrm{Circ}(\mathbb{F}_2^r)$. On note $\boldsymbol{a} = (a_1, a_2, \ldots, a_r) = (a_{1,1}, a_{1,2}, \ldots, a_{1,r})$ la première ligne de A. Pour $1 \leqslant i, j \leqslant r$, les autres coefficients de la matrice se déduisent comme*

$$a_{i,j} = a_{(j-i \mod r)+1}$$

Remarque 9. *Une matrice quasi-circulante est une matrice formée de blocs circulants.*

Exemple 1. *La matrice suivante est quasi-circulante :*

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & b_1 & b_2 & b_3 \\ a_3 & a_1 & a_2 & b_3 & b_1 & b_2 \\ a_2 & a_3 & a_1 & b_2 & b_3 & b_1 \end{bmatrix}$$

En effet, $A = [A'|B']$ *où* A' *et* B' *sont des blocs circulants puisque*

$$A' = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_3 & a_1 & a_2 \\ a_2 & a_3 & a_1 \end{bmatrix} \text{ et } B' = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_3 & b_1 & b_2 \\ b_2 & b_3 & b_1 \end{bmatrix}$$

Définition 23 (Code quasi-cyclique)**.** *Un code quasi-cyclique est un code qui admet une matrice génératrice (ou de parité) quasi-circulante.*

Propriété 7. *Un code quasi-cyclique admet au moins une matrice de parité quasi-circulante.*

Définition 24 (Ordre et indice)**.** *L'ordre* p *d'une matrice quasi-circulante est la taille des blocs circulants qui la composent. L'indice d'une matrice quasi-circulante de dimension* $n \times k$ *et d'ordre* p *désigne le rapport* $\frac{n}{p}$*.*

Propriété 8. *Soit* $A \in \mathrm{Circ}(\mathbb{F}_2^k)$ *une matrice circulante, alors*

(i) $A^t \in \mathrm{Circ}(\mathbb{F}_2^k)$*,*

(ii) *soit* $B \in \mathrm{Circ}(\mathbb{F}_2^k)$*,* $AB = BA \in \mathrm{Circ}(\mathbb{F}_2^k)$*,*

(iii) *si de plus* A *est inversible alors* $A^{-1} \in \mathrm{Circ}(\mathbb{F}_2^k)$*.*

Propriété 9. *Un code quasi-cyclique admet au moins une matrice génératrice sous forme systématique et quasi-circulante.*

Démonstration. Soit $\mathcal{C}$ un $[n, k]$-code linéaire quasi-cyclique.

Soit $G \in \mathbb{F}_2^{k \times n}$ avec $n = k_0 k$ une matrice génératrice de $\mathcal{C}$, telle que $G = [G_1|G_2|\dots|G_{k_0}]$ avec $\forall i \in \{1, \dots, k_0\}$, $G_i \in \mathrm{Circ}(\mathbb{F}_2^k)$ une matrice circulante.

Sans perte de généralité, nous supposons que G_1 est inversible (puisque la matrice considérée est de rang plein, l'un des blocs au moins est inversible), c'est-à-dire qu'il existe G_1^{-1} telle que $G_1^{-1}G_1 = I_k$. Soit $G' = G_1^{-1}G$, on a $G' = [I_k|G_1^{-1}G_2|\dots|G_1^{-1}G_{k_0}]$.

D'après la propriété 8, les matrices $G_1^{-1}G_i$ sont circulantes $\forall i \in \{1, \dots, k_0\}$. G' est donc une matrice génératrice de $\mathcal{C}$ quasi-circulante et sous forme systématique. □

Propriété 10. *Soit* $\mathrm{Circ}(\mathbb{F}_2^r)$ *l'ensemble des matrices circulantes de* $\mathbb{F}_2^{r \times r}$*. On définit* φ *comme l'isomorphisme suivant.*

$$\begin{array}{ccccc} \varphi : & \mathrm{Circ}(\mathbb{F}_2^r) & \longrightarrow & \mathbb{F}_2[X]/(X^r - 1) \\ & \begin{pmatrix} a_1 & a_2 & \dots & a_r \\ & \circlearrowright & & \end{pmatrix} & \longmapsto & a_1 + a_2 X + \dots + a_r X^{r-1} \end{array}$$

Comme une matrice circulante $A = C(\boldsymbol{a}) \in \mathrm{Circ}(\mathbb{F}_2^r)$ est totalement décrite par la donnée de $\boldsymbol{a} \in \mathbb{F}_2^r$, nous noterons plus généralement $\varphi(A) = \varphi(\boldsymbol{a}) = a(X) \in \mathbb{F}_2[X]/(X^r - 1)$.

Propriété 11. *Soient $A = C(\boldsymbol{a}) \in \mathrm{Circ}(\mathbb{F}_2^r)$ une matrice circulante et*

$$\varphi(\boldsymbol{a}) = P(X) = \sum_{i=0}^{r-1} p_{i+1} X^i \in \mathbb{F}_2[X]/(X^r - 1)$$

son image par l'isomorphisme $\varphi : \mathrm{Circ}(\mathbb{F}_2^r) \longrightarrow \mathbb{F}_2[X]/(X^r - 1)$. L'image de $A^t \in \mathrm{Circ}(\mathbb{F}_2^r)$ par φ est le polynôme transposé de P défini par

$$\begin{aligned} P^t(X) &\stackrel{def}{=} p_1 + p_r X + p_{r-1} X^2 + \ldots + p_3 X^{r-2} + p_2 X^{r-1} \\ &= \sum_{i=1}^{r} p_{(r-i+1 \mod r)+1} X^{i-1} \end{aligned}$$

Définition 25 (Correspondance vecteur-polynôme)**.** *Soit $\boldsymbol{v} = (v_1, v_2, \ldots, v_r) \in \mathbb{F}_2^r$. Le polynôme correspondant à $\boldsymbol{v}$ dans $\mathbb{F}_2[X]/(X^r - 1)$ est $v(X) = v_1 + v_2 X + v_3 X^2 + \ldots + v_r X^{r-1}$.*

Cette correspondance s'étend au vecteur de $\mathbb{F}_2^n$, en appliquant la bijection de $\mathbb{F}_2^n$ dans $\mathbb{F}_2^r \times \mathbb{F}_2^r$. Autrement dit, $\boldsymbol{v} = (\boldsymbol{v_L}|\boldsymbol{v_R}) \longrightarrow (\boldsymbol{v_L}, \boldsymbol{v_R})$, où $\boldsymbol{v_L}, \boldsymbol{v_R} \in \mathbb{F}_2^r$.

Dans la suite, l'anneau de polynôme $\mathbb{F}_2[X]/(X^r - 1)$ sera noté $\mathcal{R}$. Aussi, sans perte de généralité, nous noterons, pour $P(X) = \sum_{i=0}^{r-1} p_{i+1} X^i \in \mathcal{R}$,

$$\mathrm{wt}(P(X)) \stackrel{\text{def}}{=} |\{i \ : 1 \leqslant i \leqslant r \text{ et } p_i = 1\}|$$

Remarque 10. *L'addition dans $\mathbb{F}_2^r$ est équivalente à l'addition dans $\mathcal{R}$ lorsque l'on identifie les vecteurs de $\mathbb{F}_2^r$ aux coefficients des polynômes de $\mathcal{R}$.*

Propriété 12. *Soient $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$,*

$$\begin{array}{ccccccccc} f & : & \mathrm{Circ}(\mathbb{F}_2^r) \times \mathbb{F}_2^r & \longrightarrow & \mathbb{F}_2^r & \textit{et} & f_{\mathcal{R}} & : & \mathcal{R} \times \mathcal{R} & \longrightarrow & \mathcal{R} \\ & & (A, \boldsymbol{v}) & \longmapsto & \boldsymbol{v}A & & & & (a, v) & \longmapsto & v \cdot a \end{array}$$

Si $\varphi(A) = a \in \mathcal{R}$ et $\varphi(\boldsymbol{v}) = v \in \mathcal{R}$ alors $f(A, \boldsymbol{v})$ et $f_{\mathcal{R}}(a, v)$ sont équivalents.

Démonstration. Soient $A \in \mathrm{Circ}(\mathbb{F}_2^r)$, $\boldsymbol{a} \in \mathbb{F}_2^r$ et $\boldsymbol{y} = \boldsymbol{v}A \in \mathbb{F}_2^r$.

Notons $A = C(\boldsymbol{a}) = (a_{i,j})_{1 \leqslant i,j \leqslant r} \in \mathrm{Circ}(\mathbb{F}_2^r)$ où

$$\boldsymbol{a} = (a_1, a_2, \ldots, a_r) \stackrel{\text{def}}{=} (a_{1,1}, a_{1,2}, \ldots, a_{1,r})$$

En effectuant une multiplication classique on obtient :

$$\begin{aligned} \forall j, 1 \leqslant j \leqslant r \ : \ y_j &= \sum_{i=1}^{r} v_i a_{i,j} \mod 2 \\ &= \sum_{i=1}^{r} v_i a_{(j-i \mod r)+1} \mod 2 \text{ en utilisant la propriété 6.} \end{aligned}$$

D'un autre côté, soient $(v, a) \in \mathcal{R}^2$ et notons $y = av \in \mathcal{R}$.

Par convention, nous avons défini $v(X) = v_1 + v_2 X + \ldots + v_r X^{r-1} = \sum_{i=1}^{r} v_i X^{i-1}$ et de la même façon, $a(X) = \sum_{k=1}^{r} a_k X^{k-1}$.

Pour simplifier la preuve, nous nous ramenons à la définition équivalente suivante $v(X) = v'_0 + v'_1 X + \ldots + v'_{r-1} X^{r-1}$ où $\forall i, 0 \leqslant i \leqslant r-1$, $v'_i \stackrel{\text{def}}{=} v_{i+1}$. De même, nous noterons $a(X) = \sum_{k=0}^{r-1} a'_k X^k$ avec $\forall k, 0 \leqslant k \leqslant r-1$, $a'_k \stackrel{\text{def}}{=} a_{k+1}$ et $y(X) = \sum_{j=0}^{r-1} y'_j X^j$ avec $\forall j, 0 \leqslant j \leqslant r-1$, $y'_j \stackrel{\text{def}}{=} y_{j+1}$.

Par définition, on a donc $\forall j \in \{0, \ldots, r-1\}$,

$$\begin{aligned} y'_j &\stackrel{\text{def}}{=} \sum_{i+k=j \bmod r} v'_i a'_k \\ &= \sum_{k=j-i \bmod r} v'_i a'_k \\ &= \sum_{i=0}^{r-1} v'_i a'_{(j-i) \bmod r} \\ \text{d'où } y_{j+1} &= \sum_{i=0}^{r} v_{i+1} a_{((j-i) \bmod r)+1} \end{aligned}$$

Posons maintenant $j' \stackrel{\text{def}}{=} j+1$ et $i' \stackrel{\text{def}}{=} i+1$, on obtient :

$$\begin{aligned} \forall j' \in \{1, \ldots, r\},\ y_{j'} &= \sum_{i'=1}^{r} v_{i'} a_{(((j'-1)-(i'-1)) \bmod r)+1} \\ &= \sum_{i'=1}^{r} v_{i'} a_{((j'-i') \bmod r)+1} \end{aligned}$$

L'utilisation de la définition 25 nous permet de conclure l'égalité.

□

Remarque 11. *Pour $\boldsymbol{v} \in \mathbb{F}_2^r$, l'opération de décalage cyclique d'un bit sur la droite correspond pour $\varphi(\boldsymbol{v}) \stackrel{\text{def}}{=} v(X) \in \mathbb{F}_2[X]/(X^r - 1)$ à l'opération $v(X)X$. Autrement dit, si $\boldsymbol{v} = (v_1, \ldots, v_r)$ alors $\varphi((v_r, v_1, \ldots, v_{r-1})) = v(X)X \in \mathbb{F}_2[X]/(X^r - 1)$.*

Par extension, nous identifierons le décalage cyclique de j bits sur la droite dans $\mathbb{F}_2^r$, à la multiplication par X^j dans $\mathbb{F}_2[X]/(X^r - 1)$.

2 Cryptographie fondée sur les codes linéaires

La cryptographie a pour but de sécuriser la transmission de données sensibles dans un environnement non sécurisé, c'est-à-dire lorsque les messages émis peuvent être écoutés ou interceptés par une tierce personne. Le principe est d'utiliser un algorithme dit de chiffrement, prenant en entrée un message et une clé, qui va transformer ce message de sorte qu'il devienne inintelligible. Ce dernier, généralement appelé chiffré (ou message chiffré), ne révèle donc *a priori* aucune information sur le message clair. Cependant, pour que le destinataire retrouve le message clair à partir de ce chiffré, il va employer un algorithme de déchiffrement. Celui-ci ne pourra être appliqué sans la connaissance d'un secret, qui sera appelé clé.

La sécurité des cryptosystèmes est déterminée par le nombre d'opérations nécessaires pour retrouver le message clair, à partir du chiffré, lorsque la clé n'est pas connue. Nous considérons qu'un cryptosystème dont le niveau de sécurité est N bits repose sur des problèmes dont la résolution nécessite au moins 2^N opérations.

Les primitives cryptographiques sont divisées en deux familles, qui dépendent de la manière d'exploiter ces clés : la cryptographie à clé secrète (ou cryptographie symétrique) et la cryptographie à clé publique (ou cryptographie asymétrique). De manière générale, la cryptographie à clé secrète fournit des outils sensiblement plus efficaces (en terme de stockage et de temps) mais elle nécessite au préalable une phase d'échange de clé (qui sera utilisée aussi bien pour chiffrer que pour déchiffrer) entre les participants.

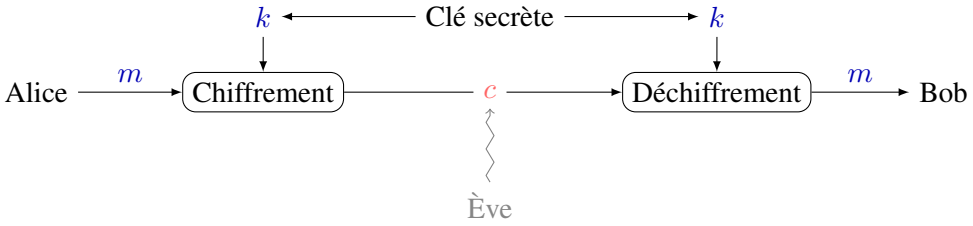


FIGURE 2.1 – Illustration du fonctionnement d'un schéma de chiffrement symétrique

C'est en 1976, que Withfield Diffie et Martin Hellman ont introduit la cryptographie à clé publique [DH76] pour répondre à ce problème en adoptant des couples de clés dont l'une est publiée, donc connue de tous (généralement désignée par pk pour *public key*) et l'autre est privée (notée sk pour *secret key*), c'est-à dire connue du seul destinataire du chiffré. Ceci facilite la transmission sécurisée de message à une personne, ou plus précisément l'envoi de clé.

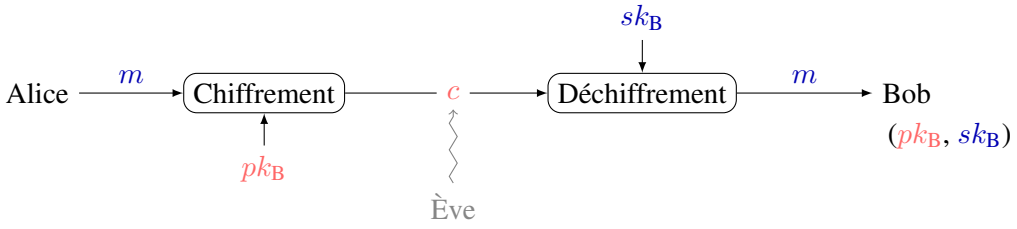


FIGURE 2.2 – Illustration du fonctionnement d'un schéma de chiffrement asymétrique

Le principe général de ces cryptosystèmes est d'employer une fonction simple à appliquer mais difficile à inverser pour transformer le message. Le destinataire, lui, connaît un secret, une trappe, qui permet d'inverser rapidement cette fonction. La sécurité de ces cryptosystèmes dépend donc de la difficulté d'inverser cette fonction sans connaître la trappe. De nos jours, la plupart des cryptosystèmes utilisés en pratique sont basés sur des problèmes issus de la théorie des nombres : la factorisation d'un entier en ses diviseurs premiers pour RSA [RSA78], la résolution du logarithme discret dans un corps fini [DH76] ou sur des courbes elliptiques. Il est maintenant prouvé que de tels problèmes seront plus faciles à résoudre en utilisant un ordinateur quantique, comme le montre [Sho94] où Peter W. Shor explicite des algorithmes polynomiaux pour résoudre ces problèmes à l'aide d'un ordinateur quantique. Ceci remet donc en question le niveau de sécurité de ces cryptosystèmes lorsqu'un tel ordinateur sera disponible.

À l'heure actuelle, la problématique est donc de trouver des primitives cryptographiques dont le niveau de sécurité ne serait pas aussi drastiquement diminué par l'utilisation d'un ordinateur quantique. Pour cela, il faut, dans un premier temps, trouver des problèmes qui resteraient difficiles à résoudre, même si un tel ordinateur est disponible. Plusieurs pistes sont envisagées et entres autres : les réseaux euclidiens, les polynômes multivariés, les fonctions de hachages et les codes correcteurs d'erreurs.

Dans ce manuscrit, nous allons plus particulièrement nous intéresser aux solutions qui utilisent des codes correcteurs d'erreurs. Nous allons, dans un premier temps, expliciter le premier cryptosystème de ce type, qui fut publié en 1978 par Robert J. McEliece. Dans un second temps, nous allons donner une description du cryptosystème de Niederreiter qui est une variante duale de ce cryptosystème. Ensuite, nous donnerons les outils qui permettent d'analyser la sécurité de ces cryptosystèmes. Nous terminerons ce chapitre par une discussion sur le choix des familles de codes dont nous disposons pour construire des primitives cryptographiques utilisant les codes correcteurs d'erreurs efficaces et sûres.

2.1 Cryptosystème de McEliece

Le cryptosystème de McEliece, introduit dans [McE78], est la première proposition de cryptosystème à clé publique qui utilise les codes correcteurs d'erreurs. Dans un premier temps, le message à transmettre va d'abord être codé pour former un mot du code. Ensuite, l'émetteur du message va simuler un canal bruité pour ajouter des erreurs à ce mot de code. Contrairement à l'usage classique des codes correcteurs d'erreurs, ce bruit est, la plupart du temps, représenté comme un vecteur binaire de poids fixe qui est construit par l'émetteur et non par un canal de transmission. Ce bruit ne pourra être corrigé que si le récepteur du message connaît un algorithme de décodage efficace associé au code utilisé. Sa sécurité repose donc en partie sur la difficulté de décoder un mot de code bruité lorsque la famille de codes n'est pas connue. Nous verrons qu'il existe de bonnes raisons de penser que ce problème est difficile et qu'il ne sera pas beaucoup plus facile à résoudre à l'aide d'un ordinateur quantique. De plus, comme la connaissance de la structure du code permet, en général, de trouver un décodeur efficace, il faut s'assurer que le code public ne soit pas distinguable d'un code aléatoire.

Dans le cryptosystème de McEliece, il faut d'abord déterminer les paramètres du code (sa longueur et sa dimension) et la valeur de t (le nombre d'erreurs ajoutées au mot de code construit à partir du message clair), en même temps qu'une famille de codes capable de corriger jusqu'à t erreurs. Les valeurs des paramètres déterminent le niveau de sécurité du cryptosystème obtenu. La clé privée est un algorithme de décodage efficace pour la famille de codes considérée. Quant à la clé publique, c'est une matrice génératrice construite à partir d'un code choisi au hasard dans cette famille de codes, qui est ensuite modifiée de sorte que la structure du code privé ne puisse pas être découverte. Il existe différentes techniques pour masquer la structure du code et, parmi elles, l'utilisation combinée d'une matrice de permutation et d'une matrice inversible. En effet, multiplier une matrice génératrice d'un code par une matrice de permutation revient à construire un code dont les positions sont permutées. La multiplication par une matrice inversible, elle, permettra d'effectuer un changement de base. Après ces deux opérations et si la famille de codes est bien choisie, la matrice publique devrait être indistinguable d'une matrice aléatoire (ne révélant aucune information sur la structure du code privé).

Remarque 12. *Soit $\mathcal{C}$ un code linéaire de longueur n engendré par une matrice génératrice $G \in \mathbb{F}_2^{k\times n}$. L'action de la permutation $P \in \mathbb{F}_2^{n\times n}$ sur G est équivalente à l'action d'une permutation π des éléments de $\{1,\dots,n\}$ sur les positions du code $\mathcal{C}$ telle que $\forall (i,j) \in \{1,\dots,n\}^2$,*

$$\pi(j) = i \Leftrightarrow P_{i,j} = 1$$

Dans [McE78], McEliece suggère d'utiliser les codes de Goppa comme famille de codes pour ce cryptosystème et à l'heure actuelle cette proposition est toujours considérée comme sûre.

Génération des paramètres
Soit $\mathcal{C}$ un $[n, k]$-code linéaire muni d'un algorithme de décodage $\Psi_{\mathcal{C}}$, capable de corriger t erreurs.

Génération des clés

— Clé publique : $G' = SGP \in \mathbb{F}_2^{k\times n}$ où $G \in \mathbb{F}_2^{k\times n}$ est une matrice génératrice de $\mathcal{C}$, S est une matrice inversible de $\mathbb{F}_2^{k\times k}$ et P une matrice de permutation dans $\mathbb{F}_2^{n\times n}$.

— Clé privée : $\Psi_{\mathcal{C}}$, $S \in \mathbb{F}_2^{k\times k}$, $G \in \mathbb{F}_2^{k\times n}$ et $P \in \mathbb{F}_2^{n\times n}$ définis comme ci-dessus.

Chiffrement
Soient $\boldsymbol{m} \in \mathbb{F}_2^k$ le message clair et $G' \in \mathbb{F}_2^{k\times n}$ la clé publique.
On choisit aléatoirement $\boldsymbol{e} \in \mathbb{F}_2^n$ tel que $wt(\boldsymbol{e}) = t$.
Le chiffré est $\boldsymbol{c} = \boldsymbol{m}G' + \boldsymbol{e} \in \mathbb{F}_2^n$.

Déchiffrement
Étant donnés $\boldsymbol{c} \in \mathbb{F}_2^n$ un chiffré et $(\Psi_{\mathcal{C}}, S, G, P)$ la clé privée.
On calcule $\boldsymbol{c}P^{-1} = \boldsymbol{m}SG + \boldsymbol{e}P^{-1}$, puis on trouve $\boldsymbol{m}SG = \Psi_{\mathcal{C}}(\boldsymbol{c}P^{-1})$.
Enfin, on en déduit $\boldsymbol{m}S$ puis on retrouve $\boldsymbol{m} = \boldsymbol{m}SS^{-1}$.

TABLE 2.1 – Cryptosystème de McEliece

2.2 Cryptosystème de Niederreiter

Le cryptosystème de Niederreiter est considéré comme la version duale du cryptosystème de McEliece. Dans [Nie86], Harald Niederreiter propose d'adopter comme secret la matrice de parité d'un code, le message à chiffrer devient une erreur de poids fixe et le chiffré un syndrome. Le déchiffrement s'effectue grâce à un décodeur par syndrome à distance bornée. Comme pour le cryptosystème de McEliece, la matrice de parité publique ne doit apporter aucune information sur le code privé. La plupart du temps, la technique employée pour masquer la structure de ce code est la même que pour le cryptosystème de McEliece, c'est-à-dire une permutation des positions et un changement de base. L'un des intérêts de ce cryptosystème est que le message chiffré est de taille k alors qu'il est construit à partir d'un message clair de longueur n, où $k \leqslant n$. Dans ce cas, la sécurité du cryptosystème repose sur la difficulté de retrouver le code privé et la difficulté du décodage par syndrome à distance bornée lorsque le code utilisé n'est pas connu.

Dans [Nie86], Niederreiter propose d'utiliser plusieurs familles de codes et entre autres, les codes de Reed Solomon, les codes concaténés et les codes de Goppa. Comme pour le cryptosystème de McEliece, parmi l'ensemble de ces propositions seule l'utilisation des codes de Goppa est considérée comme sûre encore aujourd'hui.

Remarque 13. *Le cryptosystème de Niederreiter nécessite l'utilisation de messages clairs de poids fixe t. Une façon de construire de tels messages à partir de messages clairs de poids quelconque est d'appliquer des algorithmes d'encodage à poids constant, comme par exemple [Sen05, Sch72].*

Génération des paramètres
Soit $\mathcal{C}$ un $[n, k]$-code linéaire muni d'un algorithme de décodage par syndrome ψ_H, capable de corriger t erreurs.

Génération des clés

— Clé privée : ψ_H, $S \in \mathbb{F}_2^{(n-k)\times(n-k)}$ une matrice inversible, $P \in \mathbb{F}_2^{n\times n}$ une matrice de permutation et $H \in \mathbb{F}_2^{(n-k)\times n}$ une matrice de parité de $\mathcal{C}$.

— Clé publique : $H' = SHP \in \mathbb{F}_2^{(n-k)\times n}$ où S, H et P sont définis comme ci-dessus.

Chiffrement
Soient $\boldsymbol{e} \in \mathbb{F}_2^n$ tel que $\mathrm{wt}(\boldsymbol{e}) = t$ le message clair et $H' \in \mathbb{F}_2^{(n-k)\times n}$ la clé publique.
Le chiffré est le syndrome $\boldsymbol{s} = H'\boldsymbol{e}^t \in \mathbb{F}_2^{n-k}$

Déchiffrement
Étant donnés $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ un chiffré et (ψ_H, S, H, P) la clé privée.
On calcule $S^{-1}\boldsymbol{s} = HP\boldsymbol{e}^t$.
On trouve $P\boldsymbol{e} = \psi_H(S^{-1}\boldsymbol{s})$ et on en déduit $\boldsymbol{e} = P^{-1}P\boldsymbol{e}$.

TABLE 2.2 – Cryptosystème de Niederreiter

2.3 Sécurité

La sécurité du cryptosystème de McEliece (ou de Niederreiter) repose en premier lieu sur la difficulté de résoudre des problèmes liés au décodage. En effet, pour le cryptosystème de McEliece, il doit être assez dur de décoder lorsque la famille de codes n'est pas connue, tandis que pour le cryptosystème de Niederreiter, c'est un décodage par syndrome qui est appliqué. Lorsque ces cryptosystèmes sont employés en pratique, il faut estimer la difficulté de résoudre ces problèmes avec les outils dont nous disposons aujourd'hui. Cela permet de trouver des paramètres et des familles de codes qui assurent que ces cryptosystèmes seront à un niveau de sécurité donné dans la mesure où l'adversaire dispose de ces outils. Ce type d'attaques est appelé attaques génériques, puisqu'elles ne dépendent pas de la famille de codes utilisée. Dans un second temps, pour qu'un adversaire ne puisse pas retrouver la famille de codes privée et donc un algorithme de décodage efficace, il faut s'assurer que la matrice publique est indistinguable d'une matrice aléatoire. Cette propriété dépend entièrement de la famille de codes considérée et explique en partie le fait que de nombreuses propositions d'instanciation du cryptosystème de McEliece aient été attaquées.

2.3.1 Problèmes difficiles de la théorie des codes

Pour analyser la sécurité du cryptosystème de McEliece, comme du cryptosystème de Niederreiter, il faut distinguer d'une part la sécurité du message et de l'autre la sécurité de la clé.

En effet, pour retrouver le message clair, un adversaire doit, soit retrouver la clé privée à partir de la clé publique, soit décoder un mot de code bruité dans un code qu'il ne connaît pas. L'indistinguabilité de la clé assure que la matrice publique choisie ne peut pas être distinguée d'une matrice aléatoire.

Ce n'est cependant pas un problème générique de la théorie des codes, puisque sa difficulté dépend directement du code utilisé. Tandis que le problème qui consiste à décoder un mot de code bruité (avec une erreur de poids fixe) est appelé décodage à distance bornée (ou décodage par syndrome à distance bornée pour le cryptosystème de Niederreiter), est lui un problème générique, étudié depuis de nombreuses années.

Problème 1 (Décodage par syndrome - Décisionnel)**.**
Paramètres : $n, k, t \in \mathbb{N}$
Instance : $H \in \mathbb{F}_2^{(n-k)\times n}$ *et* $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$
Question : *Existe-t'il un vecteur* $\boldsymbol{e} \in \mathbb{F}_2^n$ *tel que*

$$\boldsymbol{e}H^t = \boldsymbol{s} \text{ et } \mathrm{wt}(\boldsymbol{e}) \leqslant t$$

Ce problème décisionnel a été prouvé NP-complet dans [BMvT78] et le problème calculatoire associé est lui NP-dur. C'est un bon argument pour recourir à ce problème comme base de la sécurité d'un cryptosystème. Cependant, il n'existe pas de preuve qu'il soit dur en moyenne (la preuve fournit des arguments pour montrer qu'il existe des instances pour lesquelles ce problème est dur). Mais, malgré de nombreuses années de recherche sur le sujet, il n'existe, à l'heure actuelle, aucun algorithme qui le résout en temps sous-exponentiel (par rapport au poids de l'erreur t) même en utilisant un ordinateur quantique. Cette constatation est appuyée par les résultats de [Ale11], qui montre que, sous certaines hypothèses, ce problème de décodage est difficile en moyenne. Ainsi, l'utilisation des cryptosystèmes à clé publique dont la sécurité repose sur ce problème est considérée comme sûre par la communauté scientifique.

Dans les sections suivantes, nous allons tout d'abord donner les réductions de sécurité appliquées pour s'assurer que la sécurité de ces cryptosystèmes repose sur la difficulté de résoudre des problèmes durs. De plus, nous donnerons différentes attaques génériques, qui nous permettront de comprendre comment choisir les paramètres des codes utilisés.

2.3.2 Réductions de sécurité

Une réduction de sécurité est une preuve que si un attaquant est capable de casser le cryptosystème alors il est aussi capable de résoudre un ou plusieurs problèmes considérés comme impossibles à résoudre en un temps polynomial. Nous allons présenter les réductions de sécurité pour le cryptosystème de Niederreiter introduit dans [Sen10]. Celles-ci nous permettent de donner les mêmes arguments concernant la sécurité du cryptosystème de McEliece, puisqu'il a été prouvé dans [LDW94] que la sécurité de ces deux schémas est équivalente.

De façon informelle, la sécurité du cryptosystème de Niederreiter repose d'une part sur la difficulté de résoudre un problème de décodage par syndrome à distance bornée, et d'autre part sur la difficulté de distinguer une matrice de parité du code privé, d'une matrice aléatoire. Dans la suite, nous décrirons les définitions formelles d'un distingueur, d'un décodeur et d'un adversaire (attaquant) contre le cryptosystème de Niederreiter.

Dans un premier temps, nous explicitons les notations suivantes :

— $\mathcal{F}_{n,k}$ est une famille de $[n, k]$-code linéaire capable de corriger t erreurs,

— $\mathcal{H}_{n,k}$ désigne l'ensemble des matrices de rang plein de $\mathbb{F}_2^{(n-k)\times n}$,

— $\mathcal{K}_{n,k} \subset \mathcal{H}_{n,k}$ est l'ensemble des clés publiques de $\mathcal{F}_{n,k}$,

— $\mathcal{S}_n(0,t)$ est la sphère centrée en zéro et de rayon t sur $\mathbb{F}_2^n$,

— Ω désigne l'espace probabilisé dont les événements appartiennent à l'ensemble $\mathcal{H}_{n,k} \times \mathcal{S}_n(0,t)$ et muni de la distribution uniforme.

Définition 26 (Distingueur)**.** *Un programme $\mathcal{D} : \mathcal{H}_{n,k} \to \{0,1\}$ est un (T,ε)-distingueur pour $\mathcal{K}_{n,k}$ s'il s'exécute en un temps au plus T et si son avantage $\mathrm{Adv}(\mathcal{D},\mathcal{K}_{n,k})$ est supérieur à ε où*

$$\mathrm{Adv}(\mathcal{D},\mathcal{K}_{n,k}) \stackrel{def}{=} |\mathbb{P}_\Omega[\mathcal{D}(H)=1|H \in \mathcal{K}_{n,k}] - \mathbb{P}_\Omega[\mathcal{D}(H)=1]|$$

Un distingueur est donc un programme qui doit différencier les matrices considérées comme clé publique (c'est-à-dire des matrices génératrices ou de parités d'une famille de codes fixée) de matrices aléatoires de rang plein. Lorsque son avantage est supérieur à ε, pour un ε non négligeable, il y a de bonnes chances pour que le distingueur fasse la différence entre une clé publique et une matrice aléatoire et donc qu'il puisse retrouver la structure du code privé ainsi qu'un algorithme de décodage efficace.

Définition 27 (Décodeur)**.** *Un programme $\psi : \mathcal{H}_{n,k} \times \mathbb{F}_2^{n-k} \to \mathcal{S}_n(0,t)$ est un (T,ε)-décodeur pour $(\mathcal{H}_{n,k},t)$ s'il s'exécute en un temps au plus T et que sa probabilité de succès $\mathrm{Succ}(\psi)$ est supérieure à ε où*

$$\mathrm{Succ}(\psi) \stackrel{def}{=} \mathbb{P}_\Omega[\psi(H,\boldsymbol{e}H^t)=\boldsymbol{e}]$$

Ici, un décodeur est un programme qui résout en un temps raisonnable et avec une probabilité non négligeable, le problème du décodage borné.

Définition 28 (Adversaire)**.** *Un programme $\mathcal{A} : \mathcal{H}_{n,k} \times \mathbb{F}_2^n \to \mathcal{S}_n(0,t)$ est un (T,ε)- adversaire contre le cryptosystème de Niederreiter, ou $\mathcal{K}_{n,k}$, s'il s'exécute en temps au plus T et qu'il vérifie $\mathrm{Succ}(\mathcal{A},\mathcal{K}_{n,k}) > \varepsilon$, avec*

$$\mathrm{Succ}(\mathcal{A},\mathcal{K}_{n,k}) \stackrel{def}{=} \mathbb{P}_\Omega[\mathcal{A}(H,\boldsymbol{e}H^t)=\boldsymbol{e}|H \in \mathcal{K}_{n,k}]$$

Un adversaire est donc un programme qui, avec une probabilité non négligeable, arrive à résoudre le problème du décodage par syndrome, lorsque la matrice de parité utilisée est une clé publique.

Nous allons voir que si un tel adversaire existe alors, c'est qu'il connaît soit un décodeur, soit un distingueur. C'est-à-dire qu'il est capable de décoder efficacement dans un code aléatoire, ou qu'il sait reconnaître une matrice aléatoire d'une clé publique.

Propriété 13. *Soient n, k et t fixés. S'il existe un (T,ε)-adversaire contre $\mathcal{K}_{n,k}$ alors, soit il existe un $(T,\varepsilon/2)$-décodeur pour $(\mathcal{H}_{n,k},t)$, soit il existe un $(T+O(n^2),\varepsilon/2)$-distingueur pour $\mathcal{K}_{n,k}$.*

Remarque 14. *Cette propriété et sa preuve ont été introduites dans [Sen10].*

Ainsi, un tel adversaire est capable de résoudre le problème de décodage par syndrome ou de distinguabilité du code public par rapport aux codes aléatoires. En utilisant les notations précédentes, ces problèmes se décrivent de la manière suivante.

Problème 2 (Décodage par syndrome - Calculatoire)**.**
Paramètres : $n, k, t \in \mathbb{N}$ *et* $\mathcal{H}_{n,k}$ *l'ensemble des matrices aléatoires de rang plein.*
Instance : $H \in \mathcal{H}_{n,k}$ *et* $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$.
Question : *Trouver* $\boldsymbol{e} \in \mathbb{F}_2^n$ *tel que* $\boldsymbol{e}H^t = \boldsymbol{s}$ *et* $\mathrm{wt}(\boldsymbol{e}) \leqslant t$.

Problème 3 (Distinguabilité d'un code)**.**
Paramètres : $n, k \in \mathbb{N}$, $\mathcal{H}_{n,k}$, $\mathcal{F}_{n,k}$ *une famille de* $[n, k]$*-codes linéaires et* $\mathcal{K}_{n,k}$ *l'ensemble des clés publiques définies par* $\mathcal{F}_{n,k}$.
Instance : $H \in \mathcal{H}_{n,k}$.
Question : *Décider si* $H \in \mathcal{K}_{n,k}$.

Ainsi, si pour n, k et $t \in \mathbb{N}$ et $\mathcal{K}_{n,k}$ fixés, ces deux problèmes sont durs à résoudre (c'est-à-dire s'il n'existe pas d'algorithme polynomial qui les résout), le cryptosystème de McEliece est considéré comme sûr.

Cependant, pour que la réduction de sécurité définie par la propriété 13 puisse être appliquée, il faut que le message clair et le code privé soient distribués uniformément (conformément à la définition de l'espace probabilisé Ω). Cette condition fait du cryptosystème de McEliece (et de Niederreiter) un schéma de chiffrement dit *One Way Encryption scheme*, ou schéma de chiffrement à sens unique.

Propriété 14. *Soient* $n, k, t \in \mathbb{N}$, $\mathcal{F}_{n,k}$ *une famille de codes capable de corriger jusqu'à* t *erreurs qui définit un ensemble de clés publiques* $\mathcal{K}_{n,k}$ *et* $\mathcal{H}_{n,k}$ *l'ensemble des matrices de rang plein de* $\mathbb{F}_2^{(n-k)\times n}$. *Si*

(i) *il n'existe pas d'algorithmes polynomiaux pour résoudre les problèmes 2 et 3,*

(ii) *le message clair est tiré uniformément dans* $\mathbb{F}_2^n$,

(iii) *la clé publique est tirée uniformément dans* $\mathcal{K}_{n,k}$,

(iv) *l'erreur de poids* t *est tirée uniformément dans* $\mathcal{S}_n(0, t)$,

alors le cryptosystème de McEliece ou de Niederreiter est un schéma de chiffrement à sens unique.

Pour que la condition (i) soit assurée, il faut que les paramètres (dont la famille de codes utilisée) soient bien choisis. Les conditions (ii) à (iv) seront, quant à elles, assurées par l'emploi d'une conversion sémantiquement sûre. Cette fonction a aussi l'avantage de protéger les schémas de chiffrement contre d'autres types d'attaques. L'exemple le plus connu étant l'utilisation combinée de RSA avec l'OAEP (*Optimal Asymmetric Encryption Padding*) introduit dans [BR95], qui permet de protéger le cryptosystème contre les attaques par clairs et chiffrés choisis [FOPS04].

De manière similaire, l'emploi d'une conversion sémantiquement sûre garantit la résistance aux attaques par clairs et chiffrés choisis du cryptosystème de McEliece (et de Niederreiter).

La fonction considérée pour la conversion sémantiquement sûre doit être inversible, déterministe (pour permettre le déchiffrement) et peu coûteuse. L'OAEP répond à cette problématique, en adoptant une construction proche de celle des réseaux de Feistel et des fonctions de hachage, comme le montre la figure 2.3.

Remarque 15. *Dans la figure 2.3,* $\mathsf{hash}_1 : \mathbb{F}_2^{K_0} \longrightarrow \mathbb{F}_2^{N-K_0}$ *et* $\mathsf{hash}_2 : \mathbb{F}_2^{N-K_0} \longrightarrow \mathbb{F}_2^{K_0}$ *sont des fonctions de hachages. Le message initial est* $\boldsymbol{m} \in \mathbb{F}_2^{N-K_0-K_1}$, $\mathsf{rand} \in \mathbb{F}_2^{K_0}$ *est un nombre aléatoire et le message en sortie est* $\boldsymbol{m}' = (\boldsymbol{m}'_L | \boldsymbol{m}'_R) \in \mathbb{F}_2^N$.

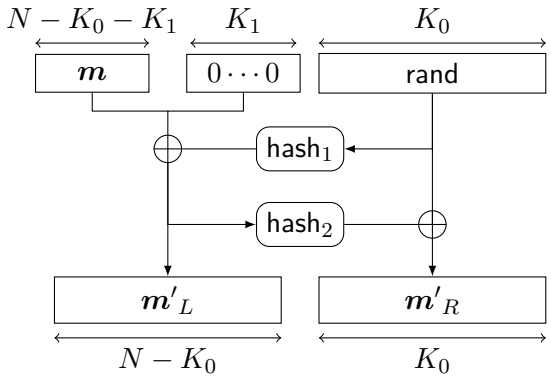


FIGURE 2.3 – Fonctionnement de l'OAEP

Un autre avantage à appliquer une fonction de conversion sémantiquement sûre pour le cryptosystème de McEliece est que la matrice publique peut alors être donnée sous forme systématique (puisque le message à chiffrer ne révèle plus d'information sur le message initial).

Génération des paramètres
Soit $\mathcal{C}$ un $[n, k]$-code linéaire muni d'un algorithme de décodage $\Psi_{\mathcal{C}}$, capable de corriger t erreurs.

Génération des clés

— Clé publique : $G = [I_k|G'] \in \mathbb{F}_2^{k \times n}$ une matrice génératrice de $\mathcal{C}$ sous forme systématique.

— Clé privée : $\Psi_{\mathcal{C}}$ un algorithme de décodage pour $\mathcal{C}$ capable de corriger t erreurs.

Chiffrement
Soit $\boldsymbol{m'} \in \mathbb{F}_2^k$ le message [a] et $G \in \mathbb{F}_2^{k \times n}$ la clé publique.
On choisit aléatoirement $\boldsymbol{e} \in \mathbb{F}_2^n$ tel que $wt(\boldsymbol{e}) = t$.
Le chiffré est $\boldsymbol{c} = \boldsymbol{m'}G + \boldsymbol{e} \in \mathbb{F}_2^n$

Déchiffrement
Étant donnés $\Psi_{\mathcal{C}}$ et $\boldsymbol{c} \in \mathbb{F}_2^n$ un chiffré, on trouve $\boldsymbol{m'}G = \Psi_{\mathcal{C}}(\boldsymbol{c})$.
Comme G est sous forme systématique, on a $\boldsymbol{m'}G = (\boldsymbol{m'}|\boldsymbol{m'}G') \in \mathbb{F}_2^n$.
Le message correspond donc aux k premiers bits de $\boldsymbol{m'}G \in \mathbb{F}_2^n$.

a. $\boldsymbol{m'}$ est construit à partir de l'image du message clair par la fonction de conversion sémantiquement sûre.

TABLE 2.3 – Cryptosystème de McEliece avec conversion sémantiquement sûre

Remarque 16. *Dans la table 2.3, le message $\boldsymbol{m'} \in \mathbb{F}_2^k$ qui va être chiffré est l'image du message clair par la fonction de conversion sémantiquement sûre utilisée (ou une partie de cette image). De même, l'erreur $\boldsymbol{e} \in \mathbb{F}_2^n$ de poids fixe t est déduite de cette image.*

2.3.3 Attaques génériques

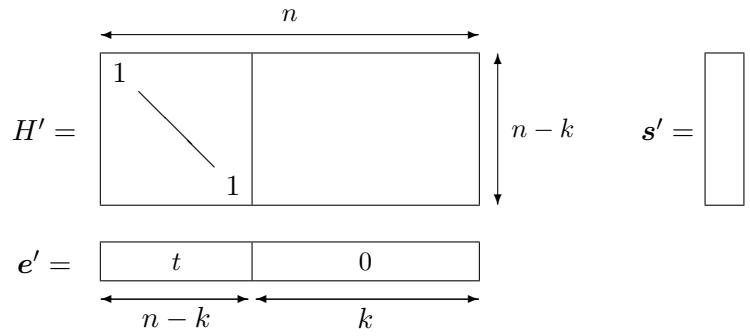
Les attaques génériques consistent à appliquer un algorithme de décodage sur le chiffré sans que la structure du code privé soit connue. Ainsi, ces attaques correspondent à résoudre le problème du décodage (ou du décodage par syndrome) borné. Autrement dit, pour $t \in \mathbb{N}$ fixé, $H \in \mathbb{F}_2^{(n-k)\times n}$ et $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$, il faut trouver $\boldsymbol{e} \in \mathbb{F}_2^n$ tel que $\boldsymbol{s} = \boldsymbol{e}H^t$ avec $\mathrm{wt}(\boldsymbol{e}) = t$.

Il existe de nombreux algorithmes pour résoudre ces problèmes, mais dans notre contexte, nous considérerons seulement les méthodes les plus efficaces. En effet, c'est la complexité de ces algorithmes qui nous permettra de choisir les paramètres du code et le nombre d'erreurs à ajouter en fonction du niveau de sécurité attendu. Le premier algorithme à considérer est la recherche exhaustive, qui consiste simplement à tester tous les mots de poids fixe et à vérifier s'ils sont solutions du système d'équations ou non. Cette méthode est exponentielle ($O\binom{n}{t}$ où t est le nombre d'erreurs ajoutées), comme le sont toutes les méthodes connues à l'heure actuelle.

Les plus efficaces emploient les algorithmes de décodage par ensemble d'informations (ou *Information Set Decoding*). De manière générale, pour un $[n, k]$-code linéaire, un ensemble d'informations est un ensemble de taille k de positions du code qui forment un espace vectoriel de dimension k. Supposons que nous ayons trouvé $I = \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$ un tel ensemble et supposons qu'en plus son support forme une intersection vide avec le support de l'erreur. Notons $P \in \mathbb{F}_2^{n\times n}$ une matrice de permutation qui, pour tout $i \in I$, a pour image un élément de l'ensemble $\{(n-k)+1, \ldots, n\}$. La matrice HP ainsi construite est ensuite mise sous forme systématique, en utilisant une matrice inversible $U \in \mathbb{F}_2^{r\times r}$. Nous obtenons donc un système équivalent

$$\boldsymbol{e}H^t = \boldsymbol{s} \text{ avec } \mathrm{wt}(\boldsymbol{e}) = t \Leftrightarrow \boldsymbol{e}P(UHP)^t = \boldsymbol{s}U^t \text{ avec } \mathrm{wt}(\boldsymbol{e}P) = t$$

Dans la suite, nous noterons $\boldsymbol{e}' = \boldsymbol{e}P$, $\boldsymbol{s}' = \boldsymbol{s}U^t$ et $H' = UHP = [I_r|H'_R]$.



Comme le montre la figure ci-dessus, trouver une solution du système correspond à trouver $\boldsymbol{e}' \in \mathbb{F}_2^n$ de la forme $\boldsymbol{e}' = (\boldsymbol{e}'_L|\boldsymbol{e}'_R) \in \mathbb{F}_2^n$ avec $\boldsymbol{e}'_L \in \mathbb{F}_2^r$ et $\mathrm{wt}(\boldsymbol{e}'_L) = t$, $\boldsymbol{e}'_R \in \mathbb{F}_2^k$ et $\mathrm{wt}(\boldsymbol{e}'_R) = 0$, tel que $\boldsymbol{e}'[I_r|H'_R]^t = \boldsymbol{e}'_L = \boldsymbol{s}'$.

Ainsi, une fois que le choix de P et U est effectué, si $\mathrm{wt}(\boldsymbol{s}U^t) = t$ alors $\boldsymbol{e} = \boldsymbol{e}'P^{-1}$ est une solution du système. En effet, nous avons alors trouvé $\boldsymbol{e}' = (\boldsymbol{s}U^t|0\ldots0)$ avec $\mathrm{wt}(\boldsymbol{e}') = t$ qui vérifie $\boldsymbol{e}'H' = \boldsymbol{s}'$ et comme la multiplication par une matrice de permutation ne change pas le poids d'un vecteur, $\mathrm{wt}(\boldsymbol{e}) = t$ et $\boldsymbol{e}$ vérifie $\boldsymbol{e}H^t = \boldsymbol{s}$.

L'algorithme 1 présente de façon sommaire l'algorithme que nous venons d'introduire, qui est

le premier algorithme de décodage par ensemble d'informations et qui a été proposé par Eugene Prange [Pra62].

Algorithme 1 Décodage par ensemble d'informations de Prange

Entrées : $H \in \mathbb{F}_2^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbb{F}_2^{n-k}$ et $t \in \mathbb{N}$
Sortie : $\boldsymbol{e} \in \mathbb{F}_2^n$ tel que $\mathrm{wt}(\boldsymbol{e}) = t$ et $\boldsymbol{e}H^t = \boldsymbol{s}$

Choisir $P \in \mathbb{F}_2^{n\times n}$ une matrice de permutation aléatoire
Trouver $U \in \mathbb{F}_2^{(n-k)\times(n-k)}$ une matrice inversible telle que UHP soit sous forme systématique
while $\mathrm{wt}(\boldsymbol{s}U^t) \neq t$ **do**
    Tirer une autre matrice de permutation P et en déduire la nouvelle valeur de U
end while
$\boldsymbol{e}' \leftarrow (\boldsymbol{s}U^t|0\dots0)$
return $\boldsymbol{e}'P^{-1}$

La complexité de cet algorithme dépend en partie de la probabilité de trouver une matrice inversible P telle que les k dernières colonnes de la matrice HP forment un ensemble d'informations. De manière équivalente, cela correspond à trouver P telle que les $n-k$ premières colonnes de HP ne définissent pas un ensemble d'informations. La probabilité de trouver une telle matrice est

$$\frac{\binom{n-k}{t}}{\binom{n}{t}}$$

puisque le calcul de cette probabilité revient à trouver le nombre de vecteurs de poids t dont le support est contenu dans les $n-k$ positions qui n'appartiennent pas à l'ensemble d'informations, lorsque les vecteurs de poids t sont tirés uniformément dans l'ensemble des vecteurs de $\mathbb{F}_2^n$.

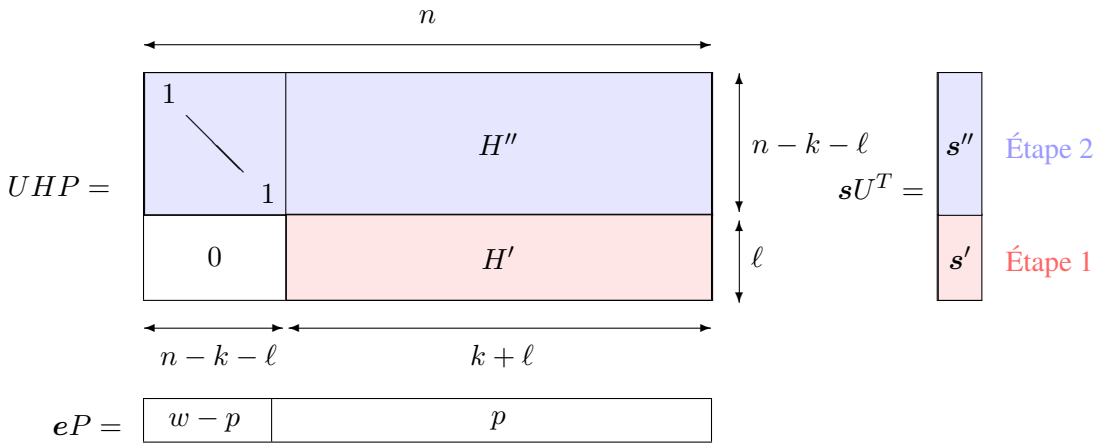
De plus, seule l'opération qui consiste à trouver U telle que UHP soit sous forme systématique a un coût non négligeable (correspondant au coût d'un pivot de Gauss) et polynomial. Cette étape doit être effectuée à chaque fois que la matrice de permutation choisie n'est pas adaptée et donc doit être répétée $\frac{\binom{n}{t}}{\binom{n-k}{t}}$ fois en moyenne. Le coût total de l'algorithme, aussi appelé *work factor* est

$$\mathrm{WF}_{\text{Prange}}(n,k,t) = 2^{c_{\text{Pra}}n(1+o(1))}$$

où $c_{\text{Pra}} = h(\tau) - (1-R)h(\frac{\tau}{1-R})$ avec $\tau = t/n$ le taux d'erreur, $R = k/n$ le taux de transmission et h la fonction d'entropie binaire ($h(x) = -x\log_2 x - (1-x)\log_2(1-x)$).

Depuis la publication de [Pra62], de nombreuses variantes de l'algorithme de Prange ont été proposées, visant à diminuer le coût de cet algorithme. L'une des premières idées fut de relaxer la condition sur la répartition du poids de l'erreur en autorisant d'en placer quelques unes, typiquement p, sur l'ensemble d'informations. Cette technique fut proposée par Pil J. Lee et Ernest F. Brickell dans [LB88]. Elle permet d'augmenter la probabilité de trouver le "bon" ensemble d'informations à $\frac{\binom{n-k}{t-p}\binom{k}{p}}{\binom{n}{t}}$ et donc de diminuer le nombre de pivots de Gauss à effectuer. Cependant, dans cet algorithme, il faut rajouter une étape qui correspond à trouver un ensemble de p colonnes de UHP dont la somme avec le syndrome $\boldsymbol{s}U^t$ a un poids égal à $w-p$. La première étape consiste donc à trouver une

solution pour le sous-problème du décodage par syndrome dans un code plus petit, avec une borne très faible (ici p). La deuxième étape est la même que celle introduite par Prange, sur un sous-problème. L'avantage de cette méthode est que le coût du pivot sur la deuxième étape est réduit (puisque le système comporte moins d'équations). Les améliorations suivantes visent à réduire le coût de l'étape 1.



Remarque 17. *Le choix des valeurs de ℓ et p permet d'optimiser le compromis entre le coût en mémoire et en temps de ces algorithmes.*

L'idée de Jacques Stern [Ste88] et Ilya Dumer [Dum91] fut d'exploiter le paradoxe des anniversaires pour augmenter la probabilité de trouver les combinaisons de colonnes adéquates lors de la première étape. Ces améliorations permettent, pour la première fois, de diminuer la constante asymptotique du coût de l'algorithme. En 1998, [CS98] fournit la première implémentation effective d'un algorithme de décodage par ensemble d'informations proche de celui introduit par [Ste88].

Les propositions [MMT11] et [BJMM12] ont recourt à des techniques moins coûteuses en temps mais plus coûteuses en mémoire pour exécuter l'étape 1.

Enfin, en 2015, Alexander May et Ilya Ozerov ont suggéré une nouvelle technique pour améliorer l'étape 1. L'algorithme présenté dans [MO15] permet de gagner encore un peu sur l'exposant, mais, pour les paramètres employés en cryptographie, il demeure trop coûteux en mémoire pour être appliqué en pratique.

Lorsque nous devrons évaluer le coût du décodage par ensemble d'informations, pour un $[n, k]$-code linéaire et un nombre d'erreur à corriger égal à t, nous utiliserons le coût de la meilleure attaque connue. Les paramètres seront donc choisis, pour un niveau de sécurité fixé, en fonction du coût des algorithmes [MMT11] et [BJMM12].

Cette quantité peut être définie comme $\mathrm{WF}_{\mathcal{A}} = 2^{c_{\mathcal{A}} n(1+o(1))}$ où $\mathcal{A}$ est l'algorithme considéré. Nous avons vu que pour l'algorithme [Pra62], $c_{\text{Pra}} = h(\tau) - (1-R)h\left(\frac{\tau}{1-R}\right)$. En comparaison, voici la valeur de cette constante pour différents algorithmes de décodage par ensemble d'informations. Par exemple, en fixant $R = 1/2$ et $\tau = 0.11$, on obtient :

— $c_{\text{Pra}} = 0.12$ pour [Pra62],

— $c_{\text{BJMM}} = 0.1$ pour [BJMM12],

— $c_{\text{MO}} = 0.095$ pour [MO15].

Remarque 18. *Les résultats de [TS16] montrent que tous ces algorithmes ont en fait le même coût asymptotique pour une quantité d'erreurs sous-linéaire en la longueur du code. En d'autres termes, pour $t = o(n)$, [TS16] montre que, asymptotiquement*

$$\mathrm{WF}_{ISD}(n, k, t) = 2^{ct(1+o(1))} \text{ avec } c = \log_2\left(\frac{1}{1-R}\right)$$

Ce résultat fournit un argument supplémentaire pour penser que la complexité des algorithmes de décodage par ensemble d'informations ne semble pas pouvoir être fortement améliorée.

La complexité de ces algorithmes lorsqu'ils sont résolus à l'aide d'un ordinateur quantique fut étudiée dans [Ber10]. L'utilisation de l'algorithme de Grover [Gro96] diminue la constante de l'algorithme de Prange par 2. Plus récemment, [Kac16] montre que la constante pour les algorithmes [Dum91], [BJMM12] et [MMT11] est diminuée d'un peu plus d'un facteur deux. Il apparaît donc que, pour les primitives de type McEliece, nous devrons choisir des paramètres qui correspondent à une sécurité de $2N$ bits en cryptographie classique, pour obtenir N bits de sécurité face à l'ordinateur quantique.

2.4 Choix d'une famille de codes

À l'origine, McEliece a suggéré l'utilisation des codes de Goppa et jusqu'à présent, il n'existe pas d'arguments prouvant que ce cryptosystème est non sûr (sauf pour des taux de transmission proches de 1 [FGO$^+$11]).

L'efficacité et la performance des algorithmes de décodage des familles de codes issues de la théorie de l'information sont, en général, très bien étudiées. Il est donc relativement aisé de trouver des familles de codes munies d'un algorithme de décodage efficace capable de corriger un nombre d'erreurs fixé. L'intérêt d'utiliser des codes dont la capacité de correction est la plus grande possible réside dans la réduction des tailles de clés. En effet, nous avons vu dans la section précédente, que la complexité de la résolution du problème de décodage générique est (asymptotiquement) $2^{ct(1+o(1))}$ avec $c = \log_2\left(\frac{1}{1-R}\right)$ où R est le taux de transmission du code et t est le nombre d'erreurs à corriger. Donc, lorsque le taux de transmission est fixé, la sécurité du cryptosystème augmente avec le poids de l'erreur. Autrement dit, si R est fixé, plus la capacité de correction du code est bonne, plus nous pouvons diminuer la longueur du code, tout en conservant un niveau de sécurité (par rapport au décodage générique) fixé.

Le point crucial dans le choix d'une famille de codes est donc d'évaluer la difficulté de résoudre le problème d'indistinguabilité du code public. En effet, il n'existe pas de méthode générique pour le résoudre puisqu'il dépend des propriétés structurelles du code utilisé.

En 1994, Vladimir M. Sildenikov a proposé d'utiliser les codes de Reed-Muller [Sid94], mais ce schéma de chiffrement fut attaqué dans [MS07]. Leur emploi dans le cryptosystème de McEliece change l'évaluation de la sécurité du cryptosystème, puisqu'ils sont uniques à paramètres fixés (à permutation près). La clé publique est donc le code lui-même et le code privé en est une permutation

(sur les positions du code). La sécurité de ce cryptosystème repose donc en partie sur la difficulté de résoudre une équivalence de code (voir le chapitre 9).

Dans [JM96], les auteurs suggèrent l'utilisation des codes géométriques alternants, mais ce cryptosystème a lui aussi été prouvé non sûr [FM08, CMCP14].

Par la suite, Carl Löndahl et Thomas Johansson ont proposé de recourir à des codes de convolution [LJ12]. L'article [LT13] montre cependant que le problème de distinguabilité de la clé est résolu efficacement pour cette famille de codes et permet de retrouver la clé privée.

Les différentes propositions énoncées ci-dessus indiquent que la structure algébrique d'un code reste difficile à masquer et donc qu'il semble difficile d'assurer l'indistinguabilité du code public lorsque celui-ci est trop structuré.

Partie II

Les codes QC-MDPC

Introduction

La plupart des familles de codes correcteurs d'erreurs ont été construites afin que les propriétés de leurs décodeurs soient les meilleures possibles, au sens de la théorie de l'information, c'est-à-dire de sorte à ce que la capacité de correction de ces décodeurs soit la plus grande possible et qu'ils soient les moins coûteux possibles. Les codes MDPC (*Moderate Density Parity Check codes*), eux, ont été définis de sorte que leurs propriétés soient adaptées à leur utilisation dans le cryptosystème de McEliece.

Comme nous allons le voir, ces codes sont très peu structurés puisqu'ils sont simplement définis par une matrice de parité creuse dont les lignes ont un poids fixe (et modéré). L'idée de tirer parti de ce type de code vient de [MRAS00]. Dans cet article, les auteurs étudient l'utilisation des codes LDPC (*Low Density Parity Check*), dont la définition est quasiment la même que celle des codes MDPC. La seule différence étant que dans ce cas, la matrice de parité considérée est très creuse (le poids des lignes étant constant en fonction de la longueur du code). L'expérience des nombreuses propositions de familles de codes pour le cryptosystème de McEliece (voir la section 2.4) semble indiquer que la résolution du problème d'indistinguabilité du code public peut être facilitée par l'emploi de codes très structurés. D'un autre côté, dans la plupart des cas, c'est la structure des codes qui est exploitée pour construire des algorithmes de décodages efficaces et performants. Les codes LDPC semblent donc être une proposition très intéressante puisqu'ils sont très peu structurés et sont munis d'algorithmes de décodage très efficaces et performants. Cependant, ces codes possèdent, de par leur définition, de nombreux mots de poids faible dans leur dual. Cette propriété sera exploitée comme distingueur par [MRAS00].

Une autre difficulté pour le choix des codes est qu'il faut prendre en compte la taille des clés publique et privée. Typiquement, ces clés étant des matrices, leur taille est au moins quadratique en la taille du message clair. Certaines propositions ont donc porté sur l'utilisation des codes quasi-cycliques (voir la section 1.2) puisque ceux-ci sont entièrement décrits grâce à une seule ligne d'une de leur matrice génératrice (ou de parité). La première proposition ayant recourt aux codes quasi-cycliques est [Gab05] où Philippe Gaborit suggère d'utiliser des sous-codes de code BCH. Cependant, les auteurs de [OTD08] ont explicité un algorithme qui permet de retrouver la clé privée à partir de la clé publique. En 2009, Rafael Misoczki et Paulo Barreto ont considéré l'emploi des codes de Goppa quasi-dyadiques [MB09]. Dans la continuité, l'utilisation de codes quasi-cycliques alternants fut soumise dans [BCGO09]. Ces deux schémas présentent cependant des faiblesses, explicitées dans [FOPT10].

Il semble donc nécessaire de prendre des précautions dès lors que des codes quasi-cycliques ou quasi-dyadiques sont employés dans le but de réduire la taille des clés.

En 2007, Marco Baldi et Franco Chiaraluce présentent une nouvelle variante du cryptosystème de McEliece [BC07] avec des codes quasi-cycliques LDPC dont la structure (les mots de poids faible dans le dual) semble être cachée. Mais la technique adoptée pour masquer la présence de mots de poids faible fut remise en question puisque [OTD08] exploite cette faille pour retrouver la clé privée. Les propositions [BCG06, BCGM07] considèrent aussi les codes LDPC et souffrent des mêmes faiblesses.

En 2008, Baldi, Barreto et Chiaraluce proposent une nouvelle variante du cryptosystème de McEliece [BBC08] avec des codes quasi-cycliques LDPC dont les mots de poids faible dans le dual sont masqués en appliquant une nouvelle technique. L'idée exposée dans cet article est la suivante. Premièrement, il faut tirer un code aléatoire $\mathcal{C}$ dans l'ensemble des codes quasi-cycliques LDPC de longueur n et de dimension k. Pour ce faire, nous pouvons construire une matrice $H \in \mathbb{F}_2^{(n-k)\times n}$ formée de k_0 blocs circulant de taille p (où $n-k = k_0 p$) et dont le poids des lignes est w. Ensuite, nous choisissons $G' \in \mathbb{F}_2^{k\times n}$ une matrice génératrice de $\mathcal{C}$ sous forme systématique (et quasi-circulante). Pour masquer les mots de poids faible dans le dual de $\mathcal{C}$, les auteurs suggèrent d'employer une matrice circulante inversible $Q \in \mathbb{F}_2^{n\times n}$ telle que le poids des lignes de Q est égal à q et une matrice circulante inversible $S \in \mathbb{F}_2^{k\times k}$ dont les lignes ont un poids d'environ $\frac{k_0 p}{2}$. La matrice publique est donc $G' = S^{-1}GQ^{-1}$ qui est la matrice génératrice d'un code dont le dual admet des mots de poids au moins wq par construction (la sécurité de la clé repose donc sur la difficulté de trouver des mots de poids wq dans le dual du code public). La clé privée quant à elle est formée des matrices H, S et Q. Dans ce cas, la clé publique est donc constituée de n bits, tandis que la clé privée comporte $n + n + n = 3n$ bits. Le chiffrement d'un message clair $\boldsymbol{m} \in \mathbb{F}_2^k$ consiste à calculer $\boldsymbol{c} = \boldsymbol{m}G' + \boldsymbol{e} \in \mathbb{F}_2^n$ où $\mathrm{wt}(\boldsymbol{e}) = t'$. Pour déchiffrer à partir d'un chiffré $\boldsymbol{c} \in \mathbb{F}_2^n$, on commence par calculer $\boldsymbol{c}' = \boldsymbol{c}Q = \boldsymbol{m}S^{-1}G + \boldsymbol{e}Q$. S'en suit l'étape de décodage dans le code LDPC $\mathcal{C}$ dont la matrice G est génératrice et durant laquelle l'erreur $\boldsymbol{e}Q$ doit être retrouvée. Il faut donc choisir un algorithme de décodage capable de décoder $t'q$ erreurs. Les codes LDPC semblent être un choix adéquat puisqu'ils possèdent des algorithmes de décodage ayant une grande capacité de correction. Après cette étape de décodage, nous pouvons retrouver le mot de code $\boldsymbol{m}S^{-1}G$. Comme G est sous forme systématique, les k premières positions du vecteur obtenu correspondent à $\boldsymbol{m}S^{-1}$. La dernière étape consiste donc à calculer $(\boldsymbol{m}S^{-1})S$ pour retrouver $\boldsymbol{m} \in \mathbb{F}_2^k$ le message clair.

L'idée de [MTSB12] est de se placer directement dans un code dont une matrice de parité a une densité modérée. De cette façon, le poids des mots dans le dual du code est borné inférieurement et ces mots de poids faible seront masqués en adoptant comme clé publique une forme systématique d'une matrice génératrice du même code, qui, de par sa construction, est dense et quasi-circulante. Le déchiffrement s'opère alors en appliquant un algorithme de décodage des codes LDPC. Nous allons voir que cet algorithme n'est pas aussi performant dans ce cadre que dans celui des LDPC. Cependant, la proportion d'erreurs à corriger étant assez faible, la faible capacité de correction de cet algorithme ne représente pas un inconvénient majeur.

3 Généralités

3.1 Définition

Les codes MDPC (*Moderate Density Parity Check*) ont été introduits dans [MTSB12]. Ils sont définis comme des codes linéaires qui admettent une matrice de parité à densité modérée, c'est-à-dire que les équations de parité définies par cette matrice sont de poids modéré. Ils diffèrent des codes LDPC (*Low Density Parity Check*) par la densité de cette matrice : pour les codes LDPC, le poids des équations de parité est constant et très faible. Ces codes ont été proposés par Robert G. Gallager en 1963 dans [Gal63] puis redécouverts par David J. C. MacKay en 1995 (voir [Mac95]). Ils sont munis d'algorithmes de décodage itératifs très efficaces et performants.

Définition 29 (Code LDPC)**.** *Un code $[n, r, w]$-LDPC est un code de longueur n, codimension r et qui admet une matrice de parité dont les lignes ont un poids fixe w, où $w = O(1)$.*

Définition 30 (Code MDPC)**.** *Un code $[n, r, w]$-MDPC est un code de longueur n, codimension r et qui admet une matrice de parité dont les lignes ont un poids fixe w, où $w = O(\sqrt{n})$.*

Dans la suite, l'ensemble des matrices de taille $n \times r$ dont les lignes ont un poids w sera noté $\mathcal{H}(n, r, w)$.

Remarque 19. *Un code $[n, r, w]$-MDPC quasi-cyclique sera noté $[n, r, w]$-QC-MDPC.*

Ainsi, un code $[n, r, w]$-QC-MDPC admet donc une matrice de parité de la forme

$$H = \begin{bmatrix} H_{1,1} & H_{1,2} & \dots & H_{1,n_0} \\ H_{2,1} & H_{2,2} & \dots & H_{2,n_0} \\ & & \vdots & \\ H_{r_0,1} & H_{r_0,2} & \dots & H_{r_0,n_0} \end{bmatrix}$$

où, pour $1 \leqslant i \leqslant r_0$ et $1 \leqslant j \leqslant n_0$, $H_{i,j} = C(\boldsymbol{h}_{i,j}) \in \operatorname{Circ} \mathbb{F}_2^{r/r_0}$ et $\sum_{j=1}^{n_0} \operatorname{wt}(\boldsymbol{h}_{i,j}) = w$.

3.2 Les codes QC-MDPC pour la cryptographie

Le choix des paramètres des codes QC-MDPC dans le cadre de leur utilisation en cryptographie est déterminé de sorte qu'ils garantissent un niveau de sécurité fixé. Cependant, il faut aussi prendre en compte les faiblesses "structurelles" de certains jeux de paramètres.

En effet, [LJKS$^+$16] fournit une attaque (par rapport au décodage générique) pour les codes quasi-cycliques dont la taille des blocs est paire. Plus généralement, les auteurs de [FL08] montrent que l'emploi d'une matrice publique quasi-circulante dont la taille des blocs n'est pas un nombre premier facilite la résolution du décodage générique.

Nous considérons donc des blocs dont la taille est un nombre premier afin de se prémunir contre ce type d'attaques. De plus, l'emploi de codes quasi-cycliques d'indice 2 et de taux de transmission $\frac{1}{2}$ permet de simplifier la mise en œuvre des primitives cryptographiques proposées et l'analyse de leur sécurité.

D'autre part, les algorithmes de générations de clés que nous allons présenter ci-après nécessitent de tirer aléatoirement une matrice circulante inversible dont les lignes ont un poids d. Or, si d est pair, une telle matrice n'est pas inversible. Nous choisirons donc d impair de sorte que la probabilité de tirer une matrice inversible soit non négligeable (en pratique, cette probabilité est très proche de 1).

Enfin, nous restreignons notre choix à celui des matrices quasi-circulantes régulières, c'est-à-dire dont les blocs ont des lignes de même poids. Nous fixerons $w = 2d$ (puisque la matrice est constituée de deux blocs), où w représente le poids des lignes de la matrice de parité.

Remarque 20. *L'ensemble des clés est donc restreint à l'ensemble des matrices quasi-circulantes qui sont de plus régulières. Il est cependant aisé de vérifier que la taille de cet ensemble n'est pas significativement plus petite que celle des matrices quasi-circulantes quelconques.*

Par la suite, nous considérerons donc des codes $[n, r, w]$-QC-MDPC où n, r et w sont définis tels que :

— r soit premier,
— $n = 2r$,
— $w = 2d$ avec d impair et $w = O(\sqrt{n})$.

De manière plus générale, $\mathcal{H}(n, r, w)$ désignera l'ensemble des matrices sur $\mathbb{F}_2^{r\times n}$ qui définissent des équations de parité dont le poids est égal à w et dont les paramètres vérifient les conditions précédentes. Nous noterons $\mathcal{H}_{\text{QC}}(n, r, w)$ les matrices de $\mathcal{H}(n, r, w)$ qui, de plus, sont quasi-circulantes.

Recourir à l'isomorphisme φ (voir la propriété 10) permet de décrire la matrice de parité quasi-circulante sous sa forme creuse, à l'aide de deux pôlynomes creux de $\mathbb{F}_2[X]/(X^r - 1)$.

Définition 31 (Représentation de la matrice de parité sous forme polynomiale)**.** *Soient* $H = [C(\boldsymbol{h}_L)|C(\boldsymbol{h}_R)] \in \mathbb{F}_2^{r\times n}$ *et* $\mathcal{C}$ *un code* $[n, r, w]$*-QC-MDPC.*
Soient $(h_L(X), h_R(X)) \in (\mathbb{F}_2[X]/(X^r - 1))^2)$ *tels que* $h_L(X) = \varphi(\boldsymbol{h}_L)$ *et* $h_R(X) = \varphi(\boldsymbol{h}_R)$.

$$H \text{ est une matrice de parité de } \mathcal{C} \quad \Leftrightarrow \quad (h_L(X), h_R(X)) \text{ est une description de } \mathcal{C}$$

Remarque 21. *Nous étendrons donc la notation* $H = [C(\boldsymbol{h}_L)|C(\boldsymbol{h}_R)] \in \mathcal{H}_{QC}(n, r, w)$ *à* $(h_L, h_R) \in \mathcal{H}_{QC}(n, r, w)$ *lorsque* $h_L(X) = \varphi(\boldsymbol{h}_L)$ *et* $h_R(X) = \varphi(\boldsymbol{h}_R)$.

4 Primitives cryptographiques utilisant les codes QC-MDPC

Dans ce chapitre nous allons présenter différentes primitives cryptographiques qui utilisent les codes QC-MDPC. Dans un premier temps, nous allons expliciter le cryptosystème de McEliece et sa sécurité lorsque celui-ci est instancié avec les codes QC-MDPC. Dans un second temps, nous verrons comment employer ces codes dans le cryptosystème de Niederreiter. Enfin, nous proposerons un schéma d'échange de clé simple et peu coûteux basé sur l'utilisation des codes QC-MDPC.

Les avantages de l'utilisation de tels codes en cryptographie sont nombreux. En effet, la taille des clés est très raisonnable, puisque, par exemple, pour le cryptosystème de McEliece et une sécurité de 80 bits la clé publique est d'environ $5\ 000$ bits. Ensuite, la sécurité de ces primitives est bien comprise et formulée de manière très simple. De plus, nous verrons que les opérations nécessaires pour chiffrer, déchiffrer ou pour échanger une clé de session sont très simples et peu coûteuses, puisqu'elles correspondent, la plupart du temps, à des opérations binaires. Aussi, nous verrons que le formalisme polynomial de la description de ces codes peut être exploité pour diminuer le coût en mémoire et en temps de ces opérations. Pour toutes ces raisons, les primitives décrites ci-après semblent adéquates lorsque l'environnement considéré est restreint.

4.1 Cryptosystème de McEliece

4.1.1 Description du cryptosystème

Dans cette section nous allons décrire le cryptosystème de McEliece utilisant les codes QC-MDPC. L'application d'une conversion sémantiquement sûre, comme introduite dans la sous-section 2.3.2, et du formalisme polynomial (voir la définition 31) permettent de simplifier sa description.

Comme nous l'avons vu dans la section 2.1, le cryptosystème de McEliece nécessite l'utilisation d'un algorithme de décodage lors de l'étape de déchiffrement.

Problème 4 (Décodage des codes QC-MDPC)**.**
Paramètres : $r, d, t \in \mathbb{N}$ *et* $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$
Instance : $h_L, h_R, s \in \mathcal{R}$ *tel que* $\mathrm{wt}(h_L) = \mathrm{wt}(h_R) = d$
Question : *Trouver* $(e_L, e_R) \in \mathcal{R}^2$ *tel que*

$$e_L h_L + e_R h_R = s \text{ et } \mathrm{wt}(e_L) + \mathrm{wt}(e_R) \leqslant t$$

Nous considérerons que pour $(h_L, h_R) \in \mathcal{H}_{\mathrm{QC}}(n, r, w)$ et lorsque t est suffisamment petit, il existe un algorithme qui résout le problème 4 efficacement.

Remarque 22. *La donnée de* $(h_L, h_R) \in \mathcal{H}_{QC}(n, r, w)$ *est équivalente à la donnée de* $(h_L^t, h_R^t) \in \mathcal{H}_{QC}(n, r, w)$*. Pour simplifier les énoncés et sans perte de généralité, nous avons choisi de décrire le problème 4, c'est-à-dire un décodage par syndrome, lorsque le code admet comme matrice de parité* $(h_L^t, h_R^t) \in \mathcal{H}_{QC}(n, r, w)$*.*

Tout algorithme qui résout le problème 4 permet aussi de décoder un mot de code bruité à distance t du code. En effet, soient $\mathcal{C}$ un code $[2r, r, 2d]$-QC-MDPC dont une matrice de parité sous sa représentation creuse est décrite par $(h_L^t, h_R^t) \in \mathcal{H}_{\mathrm{QC}}(n, r, w)$ tels que $\mathrm{wt}(h_L) = \mathrm{wt}(h_R) = d$ et $y = (y_L, y_R) \in (\mathbb{F}_2[X]/(X^r - 1))^2$ tel que $\exists\, c \in \mathcal{C}$ avec $\mathrm{wt}(c - y) = t$. Étant donnés $h_L, h_R \in \mathbb{F}_2[X]/(X^r - 1)$ et $s = y_L h_L + y_R h_R \in \mathbb{F}_2[X]/(X^r - 1)$, résoudre le problème 4 revient à trouver $(e_L, e_R) \in (\mathbb{F}_2[X]/(X^r - 1))^2$ tel que $e_L h_L + e_R h_R = y_L h_L + y_R h_R$ et $\mathrm{wt}(e_L) + \mathrm{wt}(e_R) \leqslant t$. On obtient donc le mot de code $(y_L - e_L, y_R - e_R) \in \mathcal{C}$.

Dans la suite, nous supposerons qu'il existe un algorithme capable de résoudre le problème 4 efficacement (pour des paramètres bien choisis) et dans ce cas, nous dirons donc qu'un code $[n, r, w]$-QC-MDPC est capable de corriger t erreurs grâce à la donnée de $(h_L, h_R) \in \mathcal{H}_{\mathrm{QC}}(n, r, w)$ une représentation d'une matrice de parité creuse du code.

Dans la sous-section 2.3.2, nous avons introduit le cryptosystème de McEliece avec conversion sémantiquement sûre en utilisant un formalisme vectoriel (voir la table 2.3). Nous allons voir que ces deux définitions du cryptosystème de McEliece sont équivalentes.

Propriété 15. *Les définitions du cryptosystème de McEliece explicitées par les tables 2.3 et 4.1 sont équivalentes.*

Démonstration. Dans cette preuve, nous détaillerons les différentes étapes du chiffrement en explicitant l'équivalence entre les deux formulations.

Génération des paramètres
Soient $r, d, t \in \mathbb{N}$ tels que r soit premier, d impair, $t = O(\sqrt{2r})$ et $w = O(\sqrt{2r})$.
Soient $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$ un anneau de polynôme et $\mathcal{C}$ un code $[2r, r, 2d]$-QC-MDPC capable de corriger jusqu'à t erreurs.

Génération des clés
Choisir $h_L, h_R \in \mathcal{R}$ aléatoires avec $\mathrm{wt}(h_L) = \mathrm{wt}(h_R) = d$ et tel que h_L soit inversible dans $\mathcal{R}$.
Soit $h = h_R h_L^{-1} \in \mathcal{R}$.

— Clé privée : $(h_L, h_R) \in \mathcal{R}^2$

— Clé publique : $h \in \mathcal{R}$

Chiffrement
Soit $m \in \mathcal{R}$ le message et $h \in \mathcal{R}$ la clé publique.
On choisit aléatoirement $e = (e_L, e_R) \in \mathcal{R}^2$ tel que $wt(e) = t$.
Le chiffré est $c = (c_L, c_R) = (mh + e_L, m + e_R) \in \mathcal{R}^2$

Déchiffrement
Étant donnés $c \in \mathcal{R}^2$ un chiffré et $(h_L, h_R) \in \mathcal{R}^2$ la clé privée, on retrouve $(e_L, e_R) \in \mathcal{R}^2$ tel que $c_L h_L + c_R h_R = e_L h_L + e_R h_R$ et $\mathrm{wt}(e_L) + \mathrm{wt}(e_R) \leqslant t$ utilisant un algorithme efficace pour résoudre le problème 4.
On déduit le message clair $m \in \mathcal{R}$ à partir de $(c_L - e_L, c_R - e_R) = (mh, m)$.

TABLE 4.1 – Version polynomiale du cryptosystème de McEliece utilisant les codes QC-MDPC, avec conversion sémantiquement sûre

Générations des paramètres

Dans la section 2.1, la génération des paramètres consiste à choisir $\mathcal{C}$ un $[n, k]$-code linéaire muni d'un algorithme de décodage $\psi_{\mathcal{C}}$ capable de corriger jusqu'à t erreurs.

Ici, le code choisi est $\mathcal{C}'$ un code $[2r, r, 2d]$-QC-MDPC capable de corriger jusqu'à t erreurs et nous noterons $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$.

Générations des clés

Dans le cas classique, la génération des clés s'opère en choisissant une matrice génératrice $G \in \mathbb{F}_2^{k\times n}$ de $\mathcal{C}$ sous forme systématique, *i.e.* $G = [G'|I_k] \in \mathbb{F}_2^{k\times n}$. La clé publique est donc la matrice $G \in \mathbb{F}_2^{k\times n}$ tandis que la clé privée est l'algorithme de décodage $\psi_{\mathcal{C}}$.

Dès lors que nous considérons les codes QC-MDPC, la description de $\mathcal{C}'$ un code $[2r, r, 2d]$-QC-MDPC sera donnée grâce à $(h_L, h_R) \in \mathcal{H}_{\text{QC}}(n, r, w)$ tel que $(h_L^t, h_R^t) \in \mathcal{H}_{\text{QC}}(n, r, w)$ soit une matrice de parité du code $\mathcal{C}'$. De plus, comme nous l'avons vu précédemment, la donnée de $(h_L, h_R) \in \mathcal{H}_{\text{QC}}(n, r, w)$ fournit un décodeur capable de corriger jusqu'à t erreurs. La clé publique est $h = h_R h_L^{-1} \in \mathcal{R}$ qui correspond à la donnée d'une matrice génératrice du code sous forme systématique. En effet, à partir de $(h_L^t, h_R^t) \in \mathcal{H}_{\text{QC}}(n, r, w)$ une matrice de parité de $\mathcal{C}'$, nous déduisons $(1, h_R^t(h_L^{-1})^t) \in \mathcal{R}^2$ une matrice de parité sous forme systématique du code. Ainsi, nous obtenons $((h_R^t(h_L^{-1})^t)^t, 1) = (h_R h_L^{-1}, 1) \in \mathcal{R}^2$ une représentation d'une matrice génératrice de $\mathcal{C}'$ sous forme systématique (voir la propriété 2). La clé privée est $(h_L, h_R) \in \mathcal{H}_{\text{QC}}(n, r, w)$ dont la donnée permet de résoudre efficacement le problème de décodage 4.

Chiffrement

Dans les deux cas, le chiffrement s'exécute en deux étapes. La première étape consiste à trouver l'image du message clair dans le code, en utilisant la matrice génératrice publique. La deuxième étape est de tirer aléatoirement une erreur de poids t qui va permettre de bruiter le mot de code obtenu. Soit $\boldsymbol{m} \in \mathbb{F}_2^k$ le message clair. Dans le cas classique, la clé publique $G = [G'|I_k]$ et une erreur $\boldsymbol{e} \in \mathbb{F}_2^n$ telle que $\text{wt}(\boldsymbol{e}) = t$ sont utilisées pour calculer le chiffré $\boldsymbol{c} = \boldsymbol{m}G + \boldsymbol{e} = (\boldsymbol{m}G'|\boldsymbol{m}) + \boldsymbol{e} \in \mathbb{F}_2^n$.

Dans le cas des codes QC-MDPC, le message $m \in \mathcal{R}$ est chiffré en utilisant la clé publique $h = h_R h_L^{-1} \in \mathcal{R}$ et une erreur aléatoire $e = (e_L, e_R) \in \mathcal{R}^2$ telle que $\text{wt}(e_L) + \text{wt}(e_R) = t$ de sorte que le chiffré $c = (c_L, c_R) \in \mathcal{R}^2$ corresponde à

$$(c_L, c_R) = m(h, 1) + (e_L, e_R) = (mh + e_L, m + e_R) \in \mathcal{R}^2$$

Déchiffrement

Les étapes de déchiffrement sont équivalentes : dans les deux cas, le décodage s'effectue en utilisant la clé privée. Dans un cas, c'est la donnée de l'algorithme de décodage qui est privé, tandis que dans l'autre, c'est la description d'une matrice de parité creuse du code considéré qui permet de décoder.

□

4.1.2 Sécurité

Dans la sous-section 2.3.2, nous avons introduit la réduction de sécurité qui nous permet de prouver que la sécurité du cryptosystème de McEliece repose : d'une part, sur la difficulté de résoudre le problème du décodage générique et d'autre part, sur la difficulté de la résolution du problème de distinguabilité de la clé publique.

Sécurité du message

Nous allons expliciter la complexité du décodage générique lorsque le cryptosystème de McEliece est instancié grâce aux codes $[2r, r, 2d]$-QC-MDPC et un poids d'erreur t. Pour évaluer la sécurité de ce cryptosystème nous supposerons que le code utilisé n'est pas connu.

Le problème du décodage générique est bien défini et, à l'heure actuelle, les algorithmes les plus efficaces pour le résoudre sont les algorithmes de décodage par ensemble d'informations (voir la sous-section 2.3.3). La complexité de l'algorithme le plus efficace ([BJMM12, MMT11]) sera notée $\mathrm{WF}_{\mathrm{ISD}}$ (pour *Work Factor of the Information Set Decoding*). Cependant, dans notre contexte, c'est la difficulté de résoudre le problème de décodage générique dans un code quasi-cyclique de taux de transmission $1/2$ et d'indice 2 qui nous intéresse. Pour décrire ce problème, nous emploierons le formalisme polynomial introduit précédemment à l'aide de l'isomorphisme $\varphi : \mathrm{Circ}(\mathbb{F}_2^r) \longrightarrow \mathbb{F}_2[X]/(X^r - 1)$.

Problème 5 (Décodage générique dans un code quasi-cyclique)**.**
Paramètres : $r, t \in \mathbb{N}$ *et* $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$
Instance : $h, s \in \mathcal{R}$.
Question : *Trouver* $(e_L, e_R) \in \mathcal{R}^2$ *tel que*

$$e_L + e_R h = s \text{ et } \mathrm{wt}(e_L) + \mathrm{wt}(e_R) \leqslant t$$

Ce problème est un sous-problème du problème 2 qui est NP-complet. À l'heure actuelle, il est considéré comme dur en moyenne et les algorithmes les plus efficaces pour le résoudre sont les algorithmes de décodage par ensemble d'informations.

Dans le cas quasi-cyclique, la donnée d'une instance du problème 5 permet de construire r instances ayant la même solution, puisqu'il suffit de considérer les décalages cycliques successifs à droite de cette instance pour former une nouvelle instance ayant la même solution (à décalage près). En effet, pour $r, t \in \mathbb{N}$ et $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$ fixés, la donnée de $(h, s) \in \mathcal{R}^2$ permet de construire les instances équivalentes dont l'entrée est $(h, s \cdot X^j) \in \mathcal{R}^2$ pour $1 \leqslant j \leqslant r - 1$. En effet chercher $(e_L, e_R) \in \mathcal{R}^2$ comme solution du problème 5 *i.e.* tel que $e_L + e_R h = s$ et $\mathrm{wt}(e_L) + \mathrm{wt}(e_R) = t$ est équivalent à chercher $(e'_L, e'_R) \in \mathcal{R}^2$ comme solution du problème $e'_L + e'_R h = s \cdot X^j$ pour $1 \leqslant j \leqslant r - 1$ avec $\mathrm{wt}(e'_L) + \mathrm{wt}(e'_R) = t$. Si un tel $(e_L, e_R) \in \mathcal{R}^2$ existe alors $(e'_L, e'_R) = (e_L \cdot X^j, e_R \cdot X^j)$ est solution du problème équivalent mentionné ci-dessus.

Dans ce cas, [Sen11] prouve que la complexité de l'algorithme est divisée par la racine du nombre d'instances disponibles du problème.

La complexité du décodage générique lorsque le cryptosystème de McEliece est instancié grâce

aux codes $[2r, r, w]$-QC-MDPC et un poids d'erreur t est donc

$$\frac{\mathrm{WF}_{\mathrm{ISD}}(2r, r, t)}{\sqrt{r}}$$

Sécurité de la clé

La réduction de sécurité (propriété 13) introduite dans la sous-section 2.3.2 prouve que la sécurité du cryptosystème repose d'une part sur le problème de décodage générique et d'autre part sur le problème de distinguabilité de la clé publique. Dans notre contexte, le problème de distinguabilité d'un code $[2r, r, w]$-QC-MDPC revient à décider de l'existence d'un mot de poids w dans le dual du code défini par la matrice publique. Nous appliquerons l'isomorphisme $\varphi : \mathbb{F}_2^r \longrightarrow \mathcal{R}$ qui permet de donner une version plus claire du problème de distinguabilité d'un code QC-MDPC.

Problème 6 (Distinguabilité d'un code QC-MDPC - Décisionnel)**.**
Paramètres : $r, w \in \mathbb{N}$ *et* $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$
Instance : $h \in \mathcal{R}$.
Question : *Existe-t'il* $(h_L, h_R) \in \mathcal{R}^2$ *tels que*

$$h_L h + h_R = 0 \textit{ et } \mathrm{wt}(h_L) = \mathrm{wt}(h_R) = w/2$$

Cependant, en pratique c'est le problème de recherche associé au problème 6 qui nous intéresse. En effet, pour retrouver la clé privée à partir de la clé publique, il faut résoudre le problème suivant.

Problème 7 (Distinguabilité d'un code QC-MDPC - Recherche)**.**
Paramètres : $r, w \in \mathbb{N}$ *et* $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$
Instance : $h \in \mathcal{R}$.
Question : *Trouver* $(h_L, h_R) \in \mathcal{R}^2$ *tels que*

$$h_L h + h_R = 0 \textit{ et } \mathrm{wt}(h_L) = \mathrm{wt}(h_R) = w/2$$

Bien qu'il existe un écart théorique entre la résolution des problèmes 6 et 7, nous considérerons qu'en pratique ces deux problèmes se résolvent de la même manière et ont le même coût. En effet, à l'heure actuelle, résoudre le problème décisionnel de distinguabilité d'un code QC-MDPC (*i.e.* le problème 6) revient, comme le problème de recherche associé (problème 7), à trouver un mot de poids faible dans le dual du code.

Les algorithmes les plus efficaces pour résoudre ces problèmes sont donc les algorithmes de décodage par ensemble d'informations. Ainsi, nous exploiterons la correspondance entre les polynômes de $\mathcal{R}$ et les vecteurs de $\mathbb{F}_2^r$ afin de décrire les algorithmes les plus efficaces pour résoudre ce problème et dans ce contexte, la structure quasi-cyclique ne permet pas d'améliorer le coût de ces attaques.

Dans un premier temps, rappelons que les algorithmes de décodage par ensemble d'informations prennent en entrée un syndrome $\boldsymbol{s} \in \mathbb{F}_2^r$, une matrice de parité $H \in \mathbb{F}_2^{r \times 2r}$ et permettent de trouver $\boldsymbol{e} \in \mathbb{F}_2^{2r}$ tel que $\boldsymbol{s} = \boldsymbol{e}H^t$ et $\mathrm{wt}(\boldsymbol{e}) = t$ où t est un paramètre de l'algorithme. Si ce type d'algorithme est appliqué avec $\boldsymbol{s} = 0 \in \mathbb{F}_2^r$ et $G \in \mathbb{F}_2^{r \times 2r}$ en entrée, alors ils permettent de retrouver $\boldsymbol{c} \in \mathbb{F}_2^{2r}$ tel que $\boldsymbol{c}G^t = 0$ et $\mathrm{wt}(\boldsymbol{c}) = w$ où w est le paramètre de l'algorithme qui correspond à la borne sur le

poids de la solution à chercher (*i.e.* t). Autrement dit, nous avons alors trouvé un mot du dual du code engendré par G de poids w.

En conclusion, pour retrouver un mot du dual de poids w d'un code $[2r, r, w]$-QC-MDPC, c'est-à-dire pour résoudre le problème 7, la meilleure stratégie est d'appliquer un algorithme de décodage par ensemble d'informations ayant pour entrée un syndrome nul et une matrice génératrice de ce code avec w comme borne sur le poids de la solution à chercher. La seule différence avec un code quelconque est que, dans le cas des codes MDPC, nous savons que ce problème a r solutions (qui forment une matrice de parité du code). La probabilité de trouver un mot de poids faible dans ces codes est donc r fois celle de trouver un tel mot dans un code aléatoire. Sa complexité est donc

$$\frac{\mathrm{WF}_{\mathrm{ISD}}(2r, r, w)}{r}$$

Remarque 23. *Dans ce contexte, les codes considérés sont quasi-cycliques de taux de transmission* $1/2$ *et d'indice* 2*. Une matrice génératrice d'un code* $[2r, r, w]$*-QC-MDPC est donc de taille* $2r \times r$*, tout comme une matrice de parité du code.*

Dans le cas quasi-cyclique, retrouver la clé privée, c'est-à-dire une matrice de parité du code sous sa représentation creuse, équivaut à trouver un mot du dual du code de poids modéré. En effet, d'une part, le dual d'un code $[n, r, w]$-QC-MDPC peut être décrit grâce à une matrice qui définit des équations de parité de poids w. D'autre part, la matrice considérée étant quasi-circulante, la donnée d'une de ses lignes permet de la décrire entièrement.

Remarque 24. *Dans le cas non quasi-cyclique, pour retrouver une matrice de parité creuse d'un code* $[n, r, w]$*-MDPC, il faut trouver* r *mots de poids* w *dans le dual du code. La complexité de cette attaque est donc* $\mathrm{WF}_{\mathrm{ISD}}(n, r, w)$*.*

4.1.3 Paramètres proposés

Les auteurs de [MTSB12] proposent, à titre indicatif, une série de paramètres en fonction du niveau de sécurité souhaitée.

Sécurité	r	w	t	Clé publique	Clé privée	Clair	Chiffré
80	4801	90	84	4801	9602	4801	9602
128	9857	142	134	9857	19714	9857	19714
256	32771	274	264	32771	65542	32771	65542

TABLE 4.2 – Choix des paramètres des codes $[2r, r, w]$-QC-MDPC pour le cryptosystème de McEliece avec un poids d'erreur initial t

Ces paramètres sont donnés à titre indicatif. Ils peuvent être ajustés suivant les contraintes imposées par le contexte d'application de cette primitive.

L'algorithme considéré pour résoudre le problème 4 pour un $[n, r, w]$-QC-MDPC et une erreur de poids t, est probabiliste et sa probabilité d'échec dépend, en partie, de la valeur de $\frac{w}{n}$ et $\frac{t}{n}$. Ainsi, dans un contexte où la probabilité d'échec doit être très faible, nous pouvons augmenter la valeur de n

sans modifier (de beaucoup) la valeur de w et t. Rappelons que la sécurité du cryptosystème repose d'une part, sur la sécurité du message, donnée par $\frac{\mathrm{WF}_{\mathrm{ISD}}(2r,r,t)}{\sqrt{r}}$ et d'autre part, sur la sécurité de la clé, définie par $\frac{\mathrm{WF}_{\mathrm{ISD}}(2r,r,w)}{r}$

Comme nous considérons des codes de taux de transmissions $R = \frac{1}{2}$, la codimension d'un code $[n, r, w]$-QC-MDPC est égale à sa dimension. Autrement dit, $k = r = \frac{n}{2}$.

Dans ce cas, $\mathrm{WF}_{\mathrm{ISD}}(2r, k, t) = \mathrm{WF}_{\mathrm{ISD}}(2r, r, t) = 2^{ct(1+o(1))}$ avec $c = \log_2(\frac{1}{1-R}) = 1$ et $\mathrm{WF}_{\mathrm{ISD}}(2r, r, w) = 2^{cw(1+o(1))}$ fournissent une bonne approximation de la complexité des algorithmes de décodage par ensemble d'informations. Ainsi,

— la sécurité de la clé est donnée par $\frac{2^{w(1+o(1))}}{r}$,

— la sécurité du message est égale à $\frac{2^{t(1+o(1))}}{\sqrt{r}}$.

4.2 Cryptosystème de Niederreiter

Nous allons décrire le cryptosystème de Niederreiter instancié avec les codes QC-MDPC. Pour ce faire, nous utiliserons la description polynomiale de ces codes.

Génération des paramètres
Soient $r, d, t \in \mathbb{N}$ tels que r soit premier, d impair, $t = O(\sqrt{2r})$ et $2d = O(\sqrt{2r})$.
On pose $\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$.
Soit $\mathcal{C}$ un code $[2r, r, 2d]$-QC-MDPC capable de corriger jusqu'à t erreurs.

Génération des clés
Choisir h_L, $h_R \in \mathcal{R}$ aléatoires avec $\mathrm{wt}(h_L) = \mathrm{wt}(h_R) = d$ et tels que h_L soit inversible dans $\mathcal{R}$.

— Clé privée : h_L, $h_R \in \mathcal{R}$

— Clé publique : $h = h_R h_L^{-1} \in \mathcal{R}$

Chiffrement
Soit $(e_L, e_R) \in \mathcal{R}^2$ tel que $\mathrm{wt}(e_L) + \mathrm{wt}(e_R) = t$ le message clair et $h \in \mathcal{R}$ la clé publique.
Le chiffré est le syndrome $s = e_L + e_R h \in \mathcal{R}$

Déchiffrement
Étant donnés $s \in \mathcal{R}$ un chiffré et $(h_L, h_R) \in \mathcal{R}^2$ la clé privée, on déduit les valeurs de e_L et e_R comme solution de $sh_L = e_L h_L + e_R h_R$ et $\mathrm{wt}(e_L) + \mathrm{wt}(e_R) \leqslant t$.

TABLE 4.3 – Description polynomiale du cryptosystème de Niederreiter avec conversion sémantiquement sûre utilisant les codes QC-MDPC

La sécurité de ce cryptosystème repose d'une part sur l'indistinguabilité de la clé publique (une matrice de parité sous forme systématique) et d'autre part sur le problème de décodage par syndrome. Ces deux problèmes sont équivalents aux problèmes 5 et 6 donc le calcul du niveau de sécurité de ce cryptosystème repose sur les formules explicitées dans la sous-section 4.1.2.

4.3 Schéma d'échange de clé

Un schéma d'échange de clé est une primitive cryptographique qui permet d'échanger une clé de session qui sera, par exemple, utilisée dans un second temps comme clé secrète dans un cryptosystème symétrique. Ce protocole s'effectue donc entre deux entités, que nous appellerons Alice et Bob, et nécessite plusieurs échanges entre les participants. À la fin du protocole, Alice et Bob partageront une clé de session commune K, qui ne sera utilisée qu'une seule fois. Il est important de noter que ce protocole assure la *forward secrecy*, ou confidentialité persistante, de la communication. En d'autres termes, un attaquant ne sera pas capable de retrouver *a posteriori* la clé de session K, même s'il connaît le secret (ici, une matrice de parité sous sa représentation creuse).

Plus généralement, tout cryptosystème à clé publique peut être adapté pour construire un schéma d'échange de clé. Dans un premier temps, Alice tire aléatoirement une clé privée parmi l'ensemble des clés possibles puis en déduit la clé publique. Dans un second temps, Alice envoie cette clé publique à Bob. Grâce à cette clé publique, Bob chiffre un message aléatoire et envoie ce chiffré à Alice. Puisqu'Alice connaît la clé privée, elle peut déchiffrer ce message et retrouver le message clair. Les deux parties partagent donc comme secret cet aléa. La dernière étape consiste simplement à déduire la clé de session à partir de ce message aléatoire, par exemple, en appliquant une fonction de hachage.

Paramètres : $r, d, t \in \mathbb{N}$ tels que r soit premier, d impair, $2d$ et t soient de l'ordre de $\sqrt{2r}$
$\mathcal{R} = \mathbb{F}_2[X]/(X^r - 1)$
$\mathcal{C}$ un code $[2r, r, 2d]$-QC-MDPC capable de corriger jusqu'à t erreurs.

Alice		Bob
Choisir aléatoirement $h_L, h_R \in \mathcal{R}$ avec $\mathrm{wt}(h_L) = \mathrm{wt}(h_R) = d$ et h_L inversible	$h = h_R h_L^{-1}$ →	
		Tirer aléatoirement $e_L, e_R \in \mathcal{R}$ tels que $\mathrm{wt}(e_L) + \mathrm{wt}(e_R) = t$
Déduire les valeurs de e_L et e_R comme solution de $sh_L = e_L h_L + e_R h_R$	← $s = e_L + e_R h$	

Alice et Bob partagent la même clé de session $K = \mathsf{hash}(e_L, e_R)$

FIGURE 4.1 – Protocole d'échange de clé utilisant les codes QC-MDPC

La sécurité de cette primitive cryptographique repose sur la difficulté de résoudre les problèmes de décodage par syndrome et de distinguabilité de la clé publique, comme pour le cryptosystème de Niederreiter. La principale différence entre cette primitive et les schémas de chiffrement à clé publique classiques est que les "clés" utilisées sont générées aléatoirement pour chaque session ce qui assure la *forward secrecy*.

5 Algorithme de *bit flipping*

Dans ce chapitre, nous allons décrire l'algorithme de *bit flipping* introduit par Gallager en 1963 pour le décodage des codes LDPC. Cet algorithme est itératif et probabiliste, c'est-à-dire qu'il est constitué d'une série d'opérations qui seront répétés plusieurs fois et qu'il retourne la bonne solution avec une certaine probabilité. Dans [Gal63], Gallager fournit une analyse théorique de cet algorithme, qui permet d'exprimer la probabilité que le décodeur échoue. Dans la suite, cette probabilité sera appelée probabilité d'échec au décodage. À l'origine, cet algorithme a été introduit pour le décodage des codes LDPC, qui admettent des équations de parité de poids très faible. L'idée de Gallager fut d'utiliser cette propriété pour construire un décodeur statistique (voir [Jab01, Ove06]) dont le biais provient de la constatation suivante : avec une grande probabilité, une équation de parité insatisfaite fait intervenir un petit nombre (typiquement 1) de positions du support de l'erreur et celles qui sont satisfaites, ne font, avec une grande probabilité, pas intervenir de positions du support.

Nous allons voir que cet algorithme peut aussi être employé comme décodeur pour les codes MDPC puisque ceux-ci admettent aussi des équations de parité de poids faible. Plus particulièrement, notre étude portera sur les propriétés de ce décodeur lorsque celui-ci est employé lors de l'étape de déchiffrement du cryptosystème de McEliece utilisant les codes MDPC (voir la section 4.1). Bien qu'en cryptographie, il est plus avantageux, en pratique, de considérer des codes quasi-cycliques MDPC, dans ce chapitre, nous étudierons plus généralement le décodage des codes MDPC. La seule différence notable entre le comportement des algorithmes de décodage dans ces deux contextes est que, dans le cas quasi-cyclique, employer la description polynomiale permet d'optimiser les calculs. Par contre, le comportement de l'algorithme semble être équivalent dans les deux cas.

L'organisation de ce chapitre est la suivante. Nous allons commencer par décrire l'algorithme de *bit flipping* tout en expliquant son fonctionnement. Ensuite, nous illustrerons les résultats explicités dans la section 5.2 grâce à l'analyse des résultats expérimentaux du décodage des codes MDPC. Enfin, nous terminerons ce chapitre par un aperçu de l'implémentation de quelques variantes de cet algorithme, optimisées pour différents contextes.

5.1 Convention

Dans la suite de ce chapitre, nous utiliserons la convention suivante : pour $\boldsymbol{v}$ un vecteur binaire de longueur quelconque, nous noterons $j \in \boldsymbol{v} \Leftrightarrow v_j = 1$. Autrement dit, nous confondrons les vecteurs avec leurs supports : pour $\boldsymbol{v} \in \mathbb{F}_2^n$,

$$\boldsymbol{v} \stackrel{\text{def}}{=} \{j \ : \ 1 \leqslant j \leqslant n \text{ et } v_j = 1\}$$

Ainsi, nous utiliserons la notation ensembliste $|\boldsymbol{v}|$ pour désigner le poids de Hamming de $\boldsymbol{v}$. De même pour $\boldsymbol{v}$ et $\boldsymbol{v}'$ deux vecteurs binaires de $\mathbb{F}_2^n$, $\boldsymbol{v} \cap \boldsymbol{v}'$ désignera le AND binaire usuel entre les vecteurs $\boldsymbol{v}$ et $\boldsymbol{v}'$ *i.e.*

$$\boldsymbol{v} \cap \boldsymbol{v}' \stackrel{\text{def}}{=} \{j \ : \ 1 \leqslant j \leqslant n, \ j \in \boldsymbol{v} \text{ et } j \in \boldsymbol{v}'\}$$

Enfin, pour $\mathcal{J} \subseteq \{1, \ldots, n\}$, nous noterons

$$\boldsymbol{v} - \mathcal{J} \stackrel{\text{def}}{=} \{j \ : \ 1 \leqslant j \leqslant n, \ j \in \boldsymbol{v} \text{ et } j \notin \mathcal{J}\}$$

Lorsque nous considérerons une matrice de parité $H = (h_{i,j})_{1 \leqslant i \leqslant r, \ 1 \leqslant j \leqslant n} \in \mathbb{F}_2^{r \times n}$ d'un code $[n, r, w]$-MDPC (ou LDPC), nous désignerons par $\boldsymbol{h}_i$ pour $1 \leqslant i \leqslant r$ les équations de parité définies par cette matrice. Par convention, nous supposerons que

$$\boldsymbol{h}_i \stackrel{\text{def}}{=} (h_{i,1}, \ldots, h_{i,n})$$

De même, nous noterons $\boldsymbol{h}_{*,j} \in \mathbb{F}_2^r$, pour $1 \leqslant j \leqslant n$, la transposée de la j-ème colonne de H. Autrement dit,

$$\boldsymbol{h}_{*,j} \stackrel{\text{def}}{=} (h_{1,j}, \ldots, h_{r,j})$$

Nous utiliserons, pour le calcul des probabilités, les notations suivantes : pour X une variable aléatoire à valeur dans $\mathbb{F}_2$, $X \sim \mathrm{Ber}(p)$ signifie que X suit une loi de Bernouilli de paramètre p, *i.e.* $\mathbb{P}[X = 1] = p$ et $\mathbb{P}[X = 0] = 1 - p$.

Pour $(X_i)_{1 \leqslant i \leqslant N}$ une suite de variables aléatoires, nous dirons que ces variables sont indépendantes et identiquement distribuées, que nous abrégerons par iid, si pour $1 \leqslant i \leqslant N$, les variables aléatoires suivent la même loi et sont indépendantes. Rappelons que ces N variables aléatoires sont indépendantes si, et seulement si, pour tout ensemble $I \subseteq \{1, \ldots, N\}$, $\mathbb{P}\left[\bigcap_{i \in I} X_i\right] = \prod_{i \in I} \mathbb{P}[X_i]$.

Remarque 25. *Soit $(X_i)_{1 \leqslant i \leqslant N}$ une suite de variables aléatoires iid telle que $\forall \ 1 \leqslant i \leqslant N$ et $X_i \sim \mathrm{Ber}(p)$. Alors, $X = \sum_{i=1}^{N} X_i$ suit une loi binomiale de paramètres N et p,* i.e.

$$\mathbb{P}[X = \ell] = \binom{N}{\ell} p^\ell (1-p)^{N-\ell}$$

Dans la suite, une telle variable aléatoire sera notée $X \sim \mathcal{B}(N, p)$.

5.2 Fonctionnement et description de l'algorithme

Dans cette section, nous allons détailler le fonctionnement et décrire l'algorithme de *bit flipping* comme introduit par Gallager dans [Gal63]. Le contexte dans lequel nous considérons cet algorithme est son utilisation pour le décodage des codes MDPC lors de l'étape de déchiffrement du schéma de chiffrement de McEliece. Bien que nous nous concentrerons dans un premier temps sur le fonctionnement de l'algorithme pour le décodage des codes LDPC, nous utiliserons les paramètres introduits dans 3.2 pour le dimensionnement des codes. Autrement dit, nous considérons des codes LDPC de taux de transmission $\frac{1}{2}$ et régulier (le poids des lignes des différents blocs est équilibré).

L'idée générale de l'algorithme de *bit flipping* de Gallager est la suivante. Étant donnés $\boldsymbol{y} \in \mathbb{F}_2^n$ un mot de code bruité et $\mathcal{C}$ un code $[n, r, w]$-LDPC (avec $n = 2r$) dont une matrice de parité creuse est $H \in \mathbb{F}_2^{r \times n}$, nous allons utiliser $\boldsymbol{s} = \boldsymbol{y}H^t \in \mathbb{F}_2^r$ pour identifier les positions du vecteur $\boldsymbol{y}$ qui ne vérifient pas les équations de parité de H. Rappelons que, si une coordonnée du syndrome, disons i pour $1 \leqslant i \leqslant r$, est non nulle, c'est que $\langle \boldsymbol{y}, \boldsymbol{h}_i \rangle = 1$. Autrement dit, $\boldsymbol{y}$ ne vérifie pas la i-ème équation de parité définie par H. Ceci se produit, si, et seulement si, un nombre impair de coordonnées du vecteur $\boldsymbol{y}$ qui interviennent dans cette équation sont erronées. La spécificité des codes LDPC et MDPC réside dans le fait que ces équations de parité ont un poids très faible (ou modéré). En utilisant le support de ces équations de parité, nous pouvons déterminer les positions de $\boldsymbol{y}$ susceptibles d'être fausses.

En effet, nous allons voir que la probabilité qu'une équation de parité faisant intervenir une position de $\boldsymbol{y}$ soit fausse est différente selon que cette position soit en erreur ou non. Mais pour cela, nous définirons, dans un premier temps, le modèle du canal considéré. Dans cette section, nous utiliserons le modèle étudié par Gallager dans [Gal63] : le modèle du canal binaire symétrique. Nous considérons donc

$$\boldsymbol{e} \sim \mathrm{BSC}\left(P_0\right) \ i.e. \ \boldsymbol{e} = (e_1, \ldots, e_n), \text{ avec } \forall j \in \{1, \ldots, n\}, \ e_j \sim \mathrm{Ber}\left(P_0\right)$$

Par hypothèse, nous supposons que le mot de code bruité $\boldsymbol{y}$ correspond à $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} \in \mathbb{F}_2^n$ où $\boldsymbol{c} \in \mathcal{C}$ et $\boldsymbol{e} \sim \mathrm{BSC}\left(P_0\right)$.

Dans un premier temps, pour j tel que $1 \leqslant j \leqslant n$ fixé, nous allons expliciter la distribution des variables aléatoires $\langle \boldsymbol{h}, \boldsymbol{y} \rangle = \langle \boldsymbol{h}, \boldsymbol{e} \rangle$ pour $\boldsymbol{h} \in \mathbb{F}_2^r$ tel que $|\boldsymbol{h}| = w$ et $j \in \boldsymbol{h}$.

Pour $j \in \{1, \ldots, n\}$ fixé, nous noterons $\mathcal{H}_0$ l'hypothèse $j \notin \boldsymbol{e}$ et $\mathcal{H}_1$ l'hypothèse $j \in \boldsymbol{e}$. La distribution des variables aléatoires $\langle \boldsymbol{h}, \boldsymbol{e} \rangle$ est différente selon que l'une ou l'autre des hypothèses soit vérifiée. Nous supposerons que ces variables sont iid, lorsque, respectivement $\mathcal{H}_0$ ou $\mathcal{H}_1$ est vérifiée.

Propriété 16. *Soient $\boldsymbol{e} \sim \mathrm{BSC}\left(P_0\right)$, $j \in \{1, \ldots, n\}$, $H \in \mathbb{F}_2^{r \times n}$ une matrice de parité sous sa représentation creuse d'un code $[n, r, w]$-LDPC (ou MDPC) et $\boldsymbol{h} \in \{\boldsymbol{h}_i \ : \ 1 \leqslant i \leqslant r \text{ et } j \in \boldsymbol{h}_i\}$ une équations de parité définie par cette matrice qui fait intervenir la j-ème position de l'erreur.*
Notons $\rho_0 \overset{def}{=} \mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1 | j \notin \boldsymbol{e}]$, on a alors

$$\rho_0 = \frac{1 - (1 - 2P_0)^{w-1}}{2}$$

Démonstration. Soit $j \notin \boldsymbol{e}$ fixé, c'est-à-dire sous l'hypothèse $\mathcal{H}_0$.

Tout d'abord, on a, pour $\boldsymbol{h} \in \{\boldsymbol{h}_i \ : \ 1 \leqslant i \leqslant r \text{ et } j \in \boldsymbol{h}_i\}$, $\langle \boldsymbol{h}, \boldsymbol{e} \rangle = \bigoplus_{j'=1}^{n} h_{j'} e_{j'} = e_j \bigoplus_{j' \neq j} h_{j'} e_{j'}$.

Nous supposons ici que $e_j = 0$, puisque $j \notin \boldsymbol{e}$. D'où $\langle \boldsymbol{h}, \boldsymbol{e} \rangle = \bigoplus_{j' \neq j} h_{j'} e_{j'}$. En d'autres termes, $\langle \boldsymbol{h}, \boldsymbol{e} \rangle = \bigoplus_{j' \in \boldsymbol{h} - \{j\}} e_{j'}$.

Donc $\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1$ si, et seulement si, un nombre impair de positions de l'erreur interviennent dans l'équation de parité $\boldsymbol{h}$ privée de la j-ème position de l'erreur.

Autrement dit, $\mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1 | j \notin \boldsymbol{e}] = \sum_{\substack{i=0 \\ i \text{ impair}}}^{w-1} \binom{w-1}{i} P_0^i (1-P_0)^{w-1-i}$

Lemme 1. *Soient $p \in [0,1]$, $w \in \mathbb{N}$, $f_1(p) = (p + (1-p))^{w-1}$ et $f_2(p) = (-p + (1-p))^{w-1}$ alors*

$$\frac{f_1(p) - f_2(p)}{2} = \sum_{\substack{i=0 \\ i \text{ impair}}}^{w-1} \binom{w-1}{i} p^i (1-p)^{(w-1)-i}$$

Démonstration. Il suffit de remarquer que les termes pairs de cette somme s'annulent puisque, en utilisant la formule du binôme de Newton on obtient :

$$f_1(p) = \sum_{i=0}^{w-1} \binom{w-1}{i} p^i (1-p)^{(w-1)-i}$$

et

$$f_2(p) = \sum_{i=0}^{w-1} \binom{w-1}{i} (-1)^i p^i (1-p)^{(w-1)-i}$$

□

On conclut grâce au lemme :

$$\sum_{\substack{i=0 \\ i \text{ impair}}}^{w-1} \binom{w-1}{i} P_0^i (1-P_0)^{w-1-i} = \frac{f_1(P_0) - f_2(P_0)}{2} = \frac{(P_0 + 1 - P_0)^{w-1} - (-P_0 + 1 - P_0)^{w-1}}{2}$$

Donc

$$\mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1 | j \notin \boldsymbol{e}] = \frac{1 - (1 - 2P_0)^{w-1}}{2}$$

□

Remarque 26. *Il se déduit directement de la propriété précédente que, pour $\boldsymbol{h} \in \mathbb{F}_2^r$ une équation de parité de poids w telle que $j \in \boldsymbol{h}$ et $\boldsymbol{e} \sim \mathrm{BSC}(P_0)$, alors*

$$\mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1 | j \in \boldsymbol{e}] = \mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 0 | j \notin \boldsymbol{e}] = 1 - \rho_0$$

Autrement dit, pour $\boldsymbol{h} \in \{\boldsymbol{h}_i \ : \ 1 \leqslant i \leqslant r \text{ et } j \in \boldsymbol{h}_i\}$ et pour $\boldsymbol{e} \sim \mathrm{BSC}(P_0)$ et sous l'hypothèse $\mathcal{H}_0$, $\langle \boldsymbol{h}, \boldsymbol{e} \rangle \sim \mathrm{Ber}(\rho_0)$, tandis que sous l'hypothèse $\mathcal{H}_1$, $\langle \boldsymbol{h}, \boldsymbol{e} \rangle \sim \mathrm{Ber}(1-\rho_0)$. De plus, on a $\rho_0 = \frac{1-(1-2P_0)^{w-1}}{2} = \frac{1-\varepsilon}{2}$ où $\varepsilon = (1-2P_0)^{w-1}$. C'est-à-dire que

$$\text{sous } \mathcal{H}_0 \ : \mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1] = \frac{1-\varepsilon}{2}$$

$$\text{sous } \mathcal{H}_1 \ : \mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1] = \frac{1+\varepsilon}{2}$$

Il y a donc un biais ε entre la probabilité qu'une équation de parité impliquant la j-ème position de l'erreur soit fausse suivant que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée. Pour ε proche de 1 (*i.e.* $P_0 < 1/2$ et w assez faible), $\mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1 | j \in \boldsymbol{e}] > \mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1 | j \notin \boldsymbol{e}]$. Autrement dit, une équation de parité impliquant une position du support de l'erreur aura plus tendance à ne pas être satisfaite. Cette propriété va nous aider à construire un distingueur permettant de deviner l'hypothèse la plus probable.

Pour ce faire, nous allons augmenter la taille de notre échantillon, c'est-à-dire trouver la valeur de $\langle \boldsymbol{h}, \boldsymbol{e} \rangle$ pour plusieurs $\boldsymbol{h}$ tels que $j \in \boldsymbol{h}$. Dans ce contexte, nous allons considérer que la matrice de parité H est régulière, c'est-à-dire pour $1 \leqslant j \leqslant n$, $|\{i \ : \ 1 \leqslant i \leqslant r, \ j \in \boldsymbol{h}_i\}| \stackrel{\text{def}}{=} d$. Ainsi, nous disposons d'un échantillon de taille d d'équations de parité impliquant chacune des coordonnées du vecteur d'erreur. Dans la suite, pour $1 \leqslant j \leqslant n$, $\sigma_j \stackrel{\text{def}}{=} |\{i \ : \ 1 \leqslant i \leqslant r, \ j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1\}|$ désignera le nombre d'équation de parité insatisfaites impliquant la j-ème position du vecteur bruité. Nous supposerons que ces variables aléatoires σ_j sont iid pour $j \in \boldsymbol{e}$ et $j \notin \boldsymbol{e}$.

Propriété 17. *Soient $\boldsymbol{e} \sim \mathrm{BSC}(P_0)$, $j \in \{1, \dots, n\}$, $H \in \mathbb{F}_2^{r \times n}$ une matrice de parité régulière sous sa représentation creuse d'un code $[n, r, w]$-LDPC (ou MDPC) et $\{\boldsymbol{h}_i \ : \ 1 \leqslant i \leqslant r\}$ l'ensemble des équations de parité définies par cette matrice. Posons $\sigma_j \stackrel{\text{def}}{=} |\{i \ : \ 1 \leqslant i \leqslant r, \ j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1\}|$. Alors,*

$$\begin{array}{ll} \textit{sous l'hypothèse } \mathcal{H}_0 \ : & \sigma_j \sim \mathcal{B}(d, \rho_0) \\ \textit{sous l'hypothèse } \mathcal{H}_1 \ : & \sigma_j \sim \mathcal{B}(d, (1-\rho_0)) \end{array}$$

Démonstration. Pour $1 \leqslant j \leqslant n$, $\sigma_j = |\{i \ : \ 1 \leqslant i \leqslant r, \ j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1\}| = \sum\limits_{i \ : j \in \boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$ est, par hypothèse, une somme de d variables de Bernouilli iid, d'où le résultat (voir remarque 25). □

En pratique, pour $1 \leqslant j \leqslant n$, le calcul de σ_j s'effectue plus efficacement en utilisant la définition équivalente suivante $\sigma_j = |\boldsymbol{h}_{*,j} \cap \boldsymbol{s}|$.

Propriété 18. *Pour $\boldsymbol{e} \in \mathbb{F}_2^n$ et $H \in \mathbb{F}_2^{r \times n}$ qui définie $\{\boldsymbol{h}_i \ : \ 1 \leqslant i \leqslant r \text{ et } |\boldsymbol{h}_i| = w\}$ un ensemble d'équations de parité, nous posons pour $1 \leqslant j \leqslant n$, $\sigma_j \stackrel{\text{def}}{=} |\{i \ : \ 1 \leqslant i \leqslant r, \ j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1\}|$. Soit $\boldsymbol{s} = \boldsymbol{e}H^t$ alors*

$$\sigma_j = |\boldsymbol{h}_{*,j} \cap \boldsymbol{s}|$$

Démonstration. Ce résultat se déduit directement de la définition de σ_j et de $\boldsymbol{s}$. En effet, rappelons que pour $1 \leqslant j \leqslant n$, σ_j correspond au nombre d'équations de parité insatisfaite faisant intervenir la j-ème position de l'erreur. Autrement dit, $\sigma_j = \sum_{i \ : \ j \in \boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$.

D'un autre côté, $\boldsymbol{s} = (s_i)_{1 \leqslant i \leqslant r}$ où $s_i = \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$.

Ainsi, $|\boldsymbol{h}_{*,j} \cap \boldsymbol{s}| = \sum_{i=1}^{r} h_{i,j} s_i = \sum_{i\,:\,j\in\boldsymbol{h}_i} s_i = \sum_{i\,:\,j\in\boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle$. □

Le tableau 5.1 récapitule les différents résultats énoncés précédemment. Nous considérons $1 \leqslant j \leqslant n$, $H \in \mathbb{F}_2^{r\times n}$ une matrice de parité sous sa représentation creuse d'un code $[n, r, w]$-LDPC (ou MDPC) qui définie l'ensemble $\{\boldsymbol{h}_i \ : \ 1 \leqslant i \leqslant r\}$ d'équations de parité et

$$\sigma_j = |\{i \ : \ 1 \leqslant i \leqslant r,\ j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{y}\rangle = 1\}|$$

Le tableau 5.1 met en évidence le biais entre les distributions de σ_j selon que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée. Cette distribution fournit donc un distingueur qui permet de déterminer l'hypothèse la plus probable.

Sous l'hypothèse $\mathcal{H}_0$	Sous l'hypothèse $\mathcal{H}_1$
$\langle \boldsymbol{h}_i, \boldsymbol{e}\rangle \sim \mathrm{Ber}\left(\frac{1-\varepsilon}{2}\right)$	$\langle \boldsymbol{h}_i, \boldsymbol{e}\rangle \sim \mathrm{Ber}\left(\frac{1+\varepsilon}{2}\right)$
$\sigma_j \sim \mathcal{B}\left(d, \frac{1-\varepsilon}{2}\right)$	$\sigma_j \sim \mathcal{B}\left(d, \frac{1+\varepsilon}{2}\right)$
$\mathbb{E}[\sigma_j] = d\left(\frac{1-\varepsilon}{2}\right)$	$\mathbb{E}[\sigma_j] = d\left(\frac{1+\varepsilon}{2}\right)$

TABLE 5.1 – Récapitulatif des résultats suivant que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée

Dans ce cas, l'approche classique consisterai à utiliser un grand nombre d'échantillon (c'est-à-dire un grand nombre de $\boldsymbol{h}$ telles que $j \in \boldsymbol{h}$) de sorte que la probabilité de prendre une mauvaise décision soit négligeable. Dans notre contexte, la taille de l'échantillon, c'est-à-dire d, est fixée et il faut donc trouver une autre approche.

Remarque 27. *Dans ce contexte, augmenter la valeur de d n'améliorera pas forcément le distingueur. En effet, rappelons que $\varepsilon = (1-2P_0)^{w-1}$ où w est le poids des lignes de la matrice creuse $H \in \mathbb{F}_2^{r\times n}$, qui, par définition, est proportionnel au poids de ces colonnes. La borne de Chernoff nous permet de trouver une approximation de la valeur optimale de d, qui correspond à $d > \frac{1}{\varepsilon^2} = \frac{1}{(1-2P_0)^{2(2d-1)}}$.*

L'idée introduite par Gallager est d'utiliser un seuil de décision, noté b, tel que si, pour $1 \leqslant j \leqslant n$, $\sigma_j \geqslant b$ alors la valeur de la j-ème coordonnée de $\boldsymbol{e}$ est considérée comme fausse. Intuitivement, cette approche sera efficace si les valeurs de $\mathbb{E}[\sigma_j]$ sont à une distance suffisamment grande pour $\mathcal{H}_0$ et $\mathcal{H}_1$. Autrement dit, si ε est assez proche de 1.

Propriété 19. *L'algorithme 2 renvoie bien un mot du code lorsque la matrice de parité $H \in \mathbb{F}_2^{r\times n}$ est assez creuse et lorsque le mot de code bruité $\boldsymbol{y} \in \mathbb{F}_2^n$ est assez proche du code (*i.e. *si $\exists\, \boldsymbol{c} \in \mathcal{C}$ tel que $|\boldsymbol{c} - \boldsymbol{y}| \leqslant t$ avec t assez petit).*

Le choix de la valeur du seuil b est un point crucial puisque ce choix va, à paramètres fixés, déterminer la probabilité que le poids de l'erreur diminue au cours de cette itération comme le montre les propriétés suivantes. Cette étude a donc aussi pour objectif de déterminer la probabilité que le distingueur fournisse une mauvaise information.

Algorithme 2 *Bit flipping* de [Gal63]

Entrées :
$H \in \mathbb{F}_2^{r\times n}$ une matrice de parité sous sa représentation creuse de $\mathcal{C}$, un code $[n, r, w]$-LDPC (ou MDPC), dont les équations de parité seront notées $\boldsymbol{h}_i$ pour $1 \leqslant i \leqslant r$.
$\boldsymbol{y} \in \mathbb{F}_2^n$
Sortie : Un mot du code $\mathcal{C}$

$\boldsymbol{s} \leftarrow \boldsymbol{y}H^t$ ▷ Calcul du syndrome
while $|\boldsymbol{s}| \neq 0$ **do**
  for $j \in \{1, \ldots, n\}$ **do**
    if $|\boldsymbol{h}_{*,j} \cap \boldsymbol{s}| \geqslant b$ **then**
      $y_j \leftarrow y_j \oplus 1$ ▷ On met à jour la valeur de y_j
    end if
  end for
  $\boldsymbol{s} \leftarrow \boldsymbol{y}H^t$
end while
return $\boldsymbol{y}$

Propriété 20. *Soient* $\boldsymbol{e} \sim \mathrm{BSC}(P_0)$, $j \in \{1, \ldots, n\}$ *et* $\boldsymbol{e}^{(1)}$ *l'erreur obtenue après une itération de l'algorithme de* bit flipping. *On a*

$$\textit{sous l'hypothèse } \mathcal{H}_0 \ : \quad \mathbb{P}[e_j^{(1)} = 1] = \sum_{\ell=b}^{d} \binom{d}{\ell} {\rho_0}^{\ell} (1-\rho_0)^{d-\ell}$$
$$\textit{sous l'hypothèse } \mathcal{H}_1 \ : \quad \mathbb{P}[e_j^{(1)} = 1] = \sum_{\ell=0}^{b-1} \binom{d}{\ell} (1-\rho_0)^{\ell} {\rho_0}^{d-\ell}$$

Démonstration. Rappelons que nous supposons que, pour $j \in \boldsymbol{e}$ et $j \notin \boldsymbol{e}$ respectivement, les variables aléatoires σ_j sont iid. Ainsi,

$$\begin{aligned}\mathbb{P}[e_j^{(1)} = 1 | j \in \boldsymbol{e}] &= \mathbb{P}[\sigma_j < b | j \in \boldsymbol{e}] \\ &= \sum_{\ell=0}^{b-1} \mathbb{P}[\sigma_j = \ell | j \in \boldsymbol{e}]\end{aligned}$$

De même, on a $\mathbb{P}[e_j^{(1)} = 1 | j \notin \boldsymbol{e}] = \mathbb{P}[\sigma_j \geqslant b | j \notin \boldsymbol{e}] = \sum_{\ell=b}^{d} \mathbb{P}[\sigma_j = \ell | j \notin \boldsymbol{e}]$ □

Grâce à ces propriétés, nous déduisons la probabilité qu'une position de l'erreur soit non nulle après la première itération de l'algorithme.

Propriété 21. *Soient* $\boldsymbol{e} \sim \mathrm{BSC}(P_0)$, $j \in \{1, \ldots, n\}$ *et* $P_1 = \mathbb{P}[e_j^{(1)} = 1]$ *sachant que le seuil* b *a été utilisé.*

$$P_1 = (1-P_0) \sum_{\ell=b}^{d} \binom{d}{\ell} {\rho_0}^{\ell} (1-\rho_0)^{d-\ell} + P_0 \sum_{\ell=0}^{b-1} \binom{d}{\ell} (1-\rho_0)^{\ell} {\rho_0}^{d-\ell}$$

Cette propriété peut-être étendue par récurrence pour calculer $P_u = \mathbb{P}[e_j^{(u)} = 1]$ à la u-ème itération. Dans ce contexte, l'erreur obtenue à la u-ème itération suit donc aussi une loi de Bernouilli, dont le paramètre est P_u, *i.e.* $\boldsymbol{e}^{(u)} \sim \mathrm{BSC}\,(P_u)$ pour toute itération u.

Propriété 22. *Soient* $\boldsymbol{e} \sim \mathrm{BSC}\,(P_0)$, $j \in \{1, \ldots, n\}$, $P_u = \mathbb{P}[e_j^{(u)} = 1]$ *sachant que le seuil* b *a été utilisé et* $\rho_u = \frac{1-(1-2P_u)^{w-1}}{2}$.

$$P_{u+1} = (1 - P_0) \sum_{\ell=b}^{d} \binom{d}{\ell} {\rho_u}^{\ell} (1 - \rho_u)^{d-\ell} + P_0 \sum_{\ell=0}^{b_u - 1} \binom{d}{\ell} (1 - \rho_u)^{\ell} \rho_u^{d-\ell}$$

D'après [Gal63], le meilleur choix de b_u revient à trouver le plus petit entier b tel que

$$\frac{1 - P_0}{P_0} \leqslant \left(\frac{1 + (1 - 2P_u)^{w-1}}{1 - (1 - 2P_u)^{w-1}} \right)^{2b-d}$$

Ce choix permet de s'assurer que la suite des $(P_u)_{u \geqslant 0}$ tend vers 0.

Jusqu'à présent, nous avons limité notre étude au comportement de l'erreur au cours de la première itération. Dans ce cas, nous avons montré que le distingueur proposé par Gallager sera d'autant meilleur que la probabilité P_0 telle que $\boldsymbol{e} \sim \mathrm{BSC}\,(P_0)$ et que w, *i.e.* le poids des lignes de la matrice de parité creuse considérée, sont petits. La propriété 22 montre que l'erreur suit une loi de Bernouilli quelque soit l'itération considérée. Autrement dit, pour tout u on a $\boldsymbol{e}^{(u)} \sim \mathrm{Ber}\,(P_u)$. Cette constatation nous permet d'étendre tous les résultats exprimés jusqu'à maintenant à toute itération de l'algorithme. Pour $1 \leqslant j \leqslant n$, notons $\mathcal{H}_0^{(u)}$ l'hypothèse $j \notin \boldsymbol{e}^{(u)}$ et $\mathcal{H}_1^{(u)}$ l'hypothèse $j \in \boldsymbol{e}^{(u)}$. Comme la matrice $H \in \mathbb{F}_2^{r \times n}$ est fixée, l'ensemble des équations de parité défini par cette matrice $\{\boldsymbol{h}_i \ : \ 1 \leqslant i \leqslant r\}$ est aussi fixé . Les résultats sur les variables aléatoires $\langle \boldsymbol{h}_i, \boldsymbol{e}^{(u)} \rangle$ où $1 \leqslant i \leqslant r$ ne dépendent donc que du modèle de l'erreur considérée. Autrement dit, $\langle \boldsymbol{h}_i, \boldsymbol{e}^{(u)} \rangle \sim \mathrm{Ber}\,(\rho_u)$ où $\rho_u = \frac{1-(1-2P_u)^{w-1}}{2}$. Il s'en déduit que la valeur des compteurs à la u-ème itération, que l'on notera $\sigma^{(u)}$ sont aussi connus.

Sous l'hypothèse $\mathcal{H}_0^{(u)}$	Sous l'hypothèse $\mathcal{H}_1^{(u)}$
$\langle \boldsymbol{h}_i, \boldsymbol{e}^{(u)} \rangle \sim \mathrm{Ber}\,(\rho_u)$	$\langle \boldsymbol{h}_i, \boldsymbol{e}^{(u)} \rangle \sim \mathrm{Ber}\,(1 - \rho_u)$
$\sigma_j^{(u)} \sim \mathcal{B}\,(d, \rho_u)$	$\sigma_j^{(u)} \sim \mathcal{B}\,(d, 1 - \rho_u)$
$\mathbb{E}[\sigma_j^{(u)}] = d\rho_u$	$\mathbb{E}[\sigma_j^{(u)}] = d\,(1 - \rho_u)$

TABLE 5.2 – Récapitulatif des résultats suivant que $\mathcal{H}_0^{(u)}$ ou $\mathcal{H}_1^{(u)}$ soit vérifiée

Le tableau 5.2 reprend les résultats énoncés précédemment pour toute itération u. Il est clair que le distingueur fournit par la variable aléatoire $\sigma_j^{(u)}$ reste efficace quelque soit l'itération considérée. En effet, en supposant que la suite des $(P_u)_{u \geqslant 0}$ soit décroissante et puisque w est fixé, la suite des $\rho_u \overset{\text{def}}{=} \frac{1-(1-2P_u)^{w-1}}{2}$ est décroissante. Autrement dit, au cours des itérations, la probabilité que le distingueur fournisse une mauvaise information sera plus faible.

Nous pouvons donc estimer, pour n'importe laquelle des itérations, le nombre de positions juste et fausse qui ont une valeur de compteur donnée. Ceci nous permet de déterminer le meilleur seuil (en moyenne) pour chaque itération, c'est-à-dire pour lequel nous avons une forte probabilité de faire baisser le poids de l'erreur au cours de l'itération.

Pour illustrer ces résultats, nous présenterons une analyse de l'algorithme de *bit flipping* pour des codes $[9602, 4801, 90]$-MDPC. Par exemple, la figure 5.1 représente le nombre moyen de position juste et fausse respectivement, ayant une valeur de compteur donnée, pour la première itération lorsque le poids de l'erreur initial est 84 *i.e* $P_0 = \frac{t}{n} = \frac{84}{9602}$. Autrement dit, ces courbes sont données par les équations :

$$n(1 - P_0)\binom{d}{\ell}\rho_0^{\ell}(1 - \rho_0)^{d-\ell} \tag{5.1}$$

$$nP_0\binom{d}{\ell}(1 - \rho_0)^{\ell}\rho_0^{d-\ell} \tag{5.2}$$

L'analyse de la figure 5.1 indique que, pour les paramètres considérés, le seuil devrait être fixé à 28 à la première itération. En effet, à partir de cette valeur, les positions fausses ayant une valeur de compteur supérieur à ce seuil seront plus nombreuses que les positions justes ayant une valeur de compteur aussi supérieur à celui-ci. En d'autres termes, la probabilité que le nombre d'erreur augmente est inférieure à celle que le nombre d'erreur diminue.

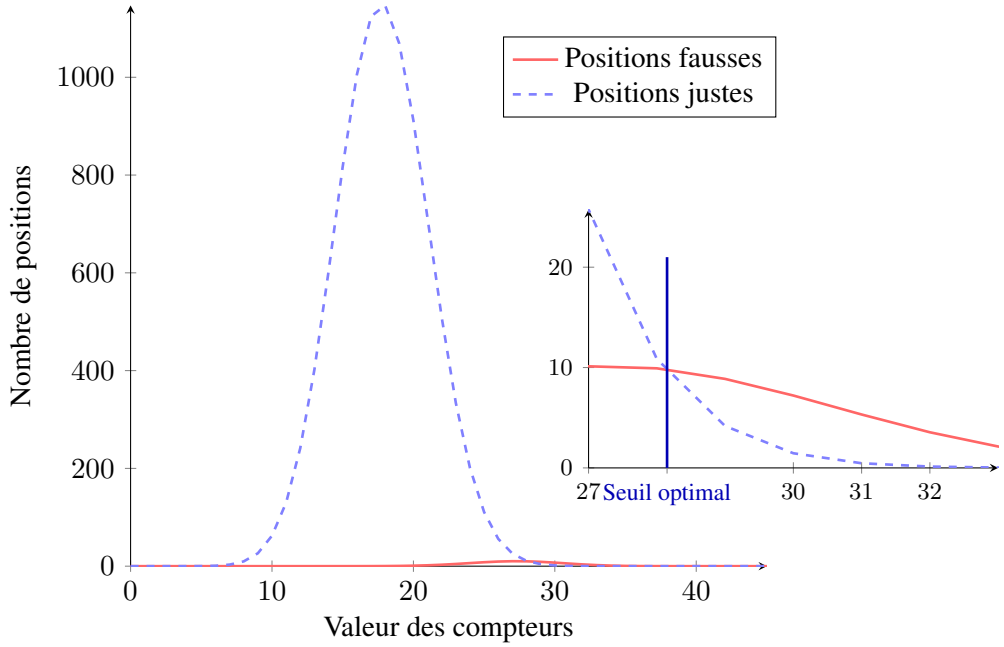


FIGURE 5.1 – Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la première itération

La figure 5.2 représente quant à elle, les courbes qui correspondent à

$$n(1-P_1)\binom{d}{\ell}\rho_1^{\ell}(1-\rho_1)^{d-\ell} \tag{5.3}$$

et

$$nP_1\binom{d}{\ell}(1-\rho_1)^{\ell}\rho_1^{d-\ell} \tag{5.4}$$

où $P_1 = (1-P_0)\sum_{\ell=b_1}^{d}\binom{d}{\ell}\rho_0{}^{\ell}(1-\rho_0)^{d-\ell} + P_0\sum_{\ell=0}^{b_1-1}\binom{d}{\ell}(1-\rho_0)^{\ell}\rho_0^{d-\ell}$ et $\rho_1 = \frac{1-(1-2P_1)^{w-1}}{2}$.

En d'autres termes, elle correspond au nombre de positions justes (donné par l'équation 5.3) et fausses (équation 5.4), qui ont une valeur de compteur donnée au cours de la deuxième itération de l'algorithme 2. La figure 5.2 indique que le seuil optimal pour la deuxième itération est 25.

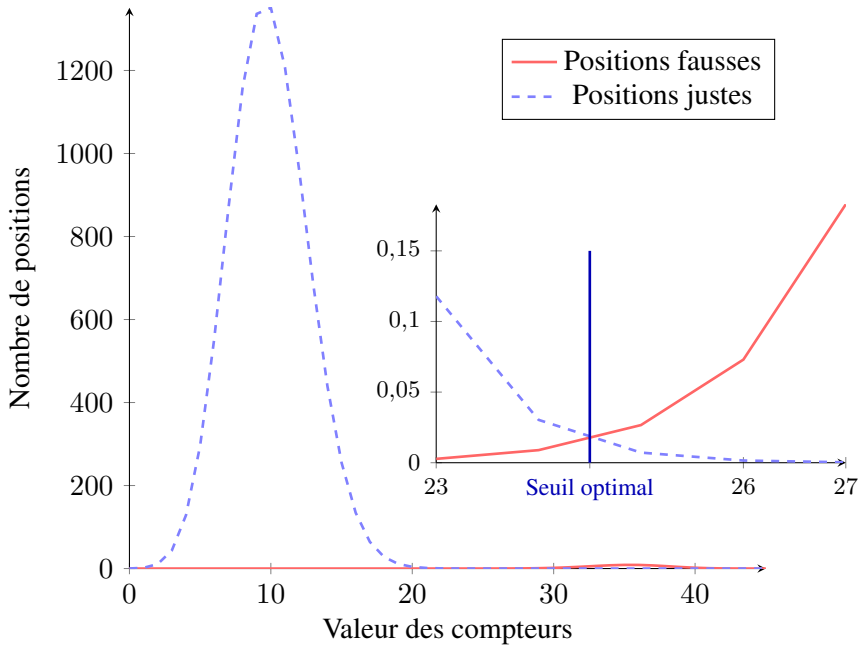


FIGURE 5.2 – Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la deuxième itération

Nous pouvons aussi remarquer qu'à la première itération, les courbes qui correspondent au nombre moyen de positions fausses et justes ayant une valeur de compteur donnée sont plus "rapprochées" qu'à la deuxième itération. Plus précisément, la différence entre l'espérance des compteurs des positions justes et celle des positions fausses est plus grande à la première itération qu'à la seconde. Ainsi, la probabilité de prendre une mauvaise décision est plus grande à la première itération qu'à la deuxième. Ce qui correspond bien aux propriétés du distingueur que nous avons explicitées ci-avant. En effet, plus la probabilité qu'une position de l'erreur soit non nulle est grande, plus la probabilité que le distingueur fournisse une mauvaise information est faible.

5.3 Résultats observés pour les codes QC-MDPC

Dans cette section, nous allons comparer les distributions théoriques et observées des différentes variables aléatoires introduites précédemment. Pour ce faire, nous utiliserons des codes $[9602, 4801, 90]$-QC-MDPC et considérerons que l'erreur suit le modèle du canal binaire symétrique avec une probabilité $P_0 = \frac{84}{9602}$ qu'une coordonnée soit égale à 1.

Cette étude a pour but de vérifier que les modèles que nous avons présentés jusqu'à présent correspondent bien à ce que nous pouvons observer dans le cas du décodage des codes QC-MDPC pour la cryptographie. Si nous pouvons constater que ces modèles sont une bonne approximation du comportement du décodeur dans ce contexte, nous pourrons estimer la probabilité d'échec au décodage après un nombre fixé d'itérations, en utilisant les formules explicitées dans la section précédente.

Plus particulièrement, nous allons expliciter la distribution des compteurs (nombre d'équations de parités insatisfaites) des positions justes et fausses respectivement. Le comportement de ces variables aléatoires détermine, en partie, l'efficacité du décodeur puisque leurs distributions permettent d'évaluer celle du poids de l'erreur après chaque itération. Dans cette section, nous considérons que les seuils ont été calculés grâce aux formules explicitées dans la section 5.2.

Remarque 28. *Les distributions observées ont été déduites de simulations sur* $15\ 10^6$ *positions fausses et* $15\ 10^7$ *positions justes pour la première itération. Quant à la deuxième itération, les résultats sont déduits de simulations sur* $10\ 10^6$ *positions fausses et* $60\ 10^7$ *positions justes. De plus, nous avons choisi d'utiliser les seuils précalculés grâce à l'analyse présentée dans la section précédente. Ainsi, nous avons utilisé un seuil égal à* 28 *à la première itération.*

Évènement	Cas considéré	Espérance	Variance
Compteurs des positions fausses	Théorique	27.17	10.76
	Observé	27.20	10.73
Compteurs des positions justes	Théorique	17.82	10.76
	Observé	17.89	10.76

TABLE 5.3 – Comparaison de l'espérance et de la variance théoriques et observées de la valeur des compteurs pour les positions justes et fausses au début de la première itération

Évènement	Cas considéré	Espérance	Variance
Compteurs des positions fausses	Théorique	35.13	7.70
	Observé	26.68	9.95
Compteurs des positions justes	Théorique	9.86	7.70
	Observé	13.82	9.82

TABLE 5.4 – Comparaison de l'espérance et de la variance théoriques et observées de la valeur des compteurs pour les positions justes et fausses au début de la deuxième itération

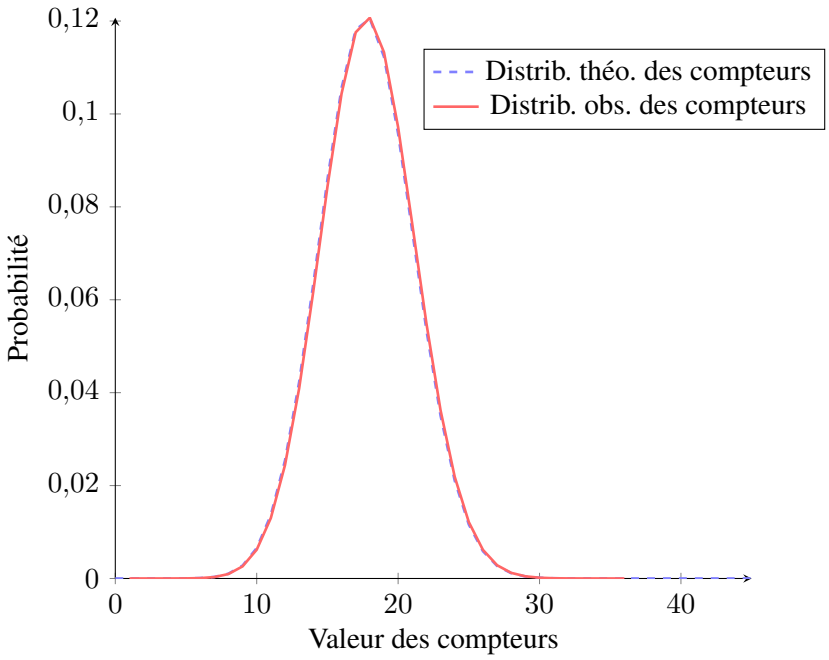


FIGURE 5.3 – Comparaison des distributions théoriques et observées des compteurs des positions justes au début de la première itération

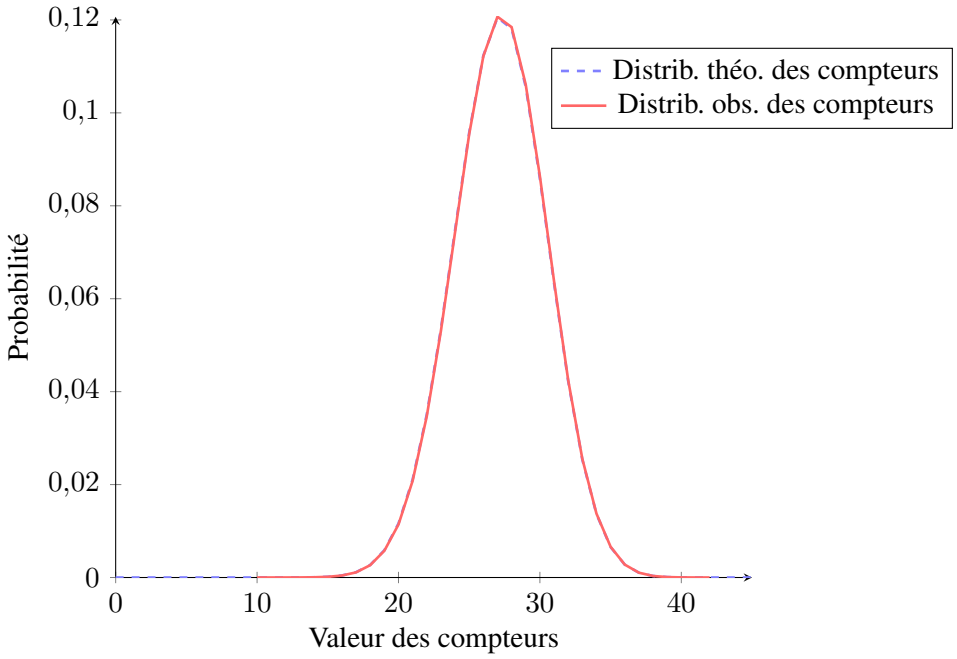


FIGURE 5.4 – Comparaison des distributions théoriques et observées des compteurs des positions fausses au début de la première itération

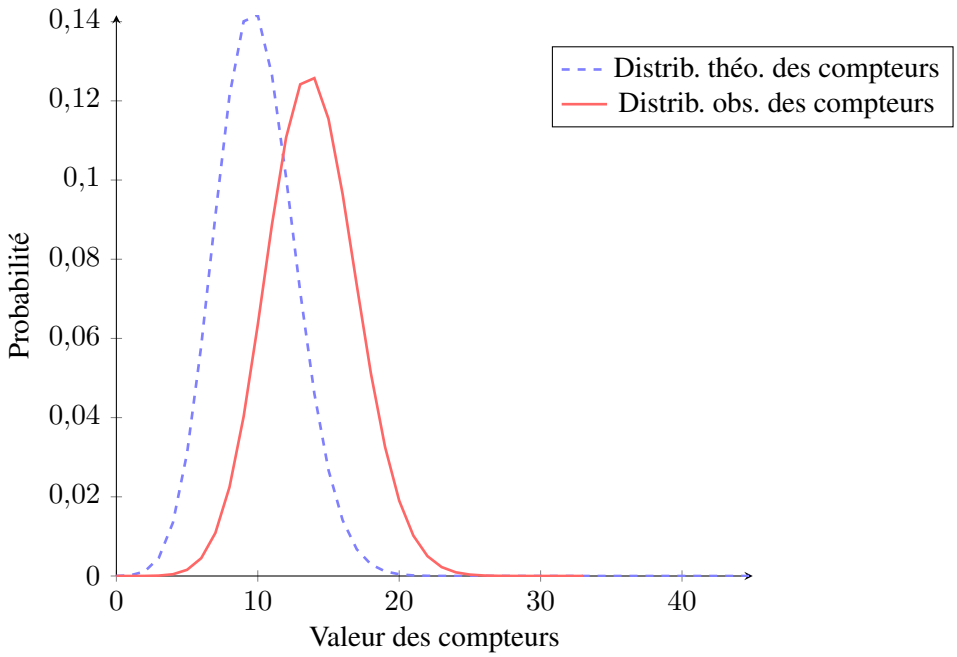


FIGURE 5.5 – Comparaison des distributions théoriques et observées des compteurs des positions justes au début de la deuxième itération

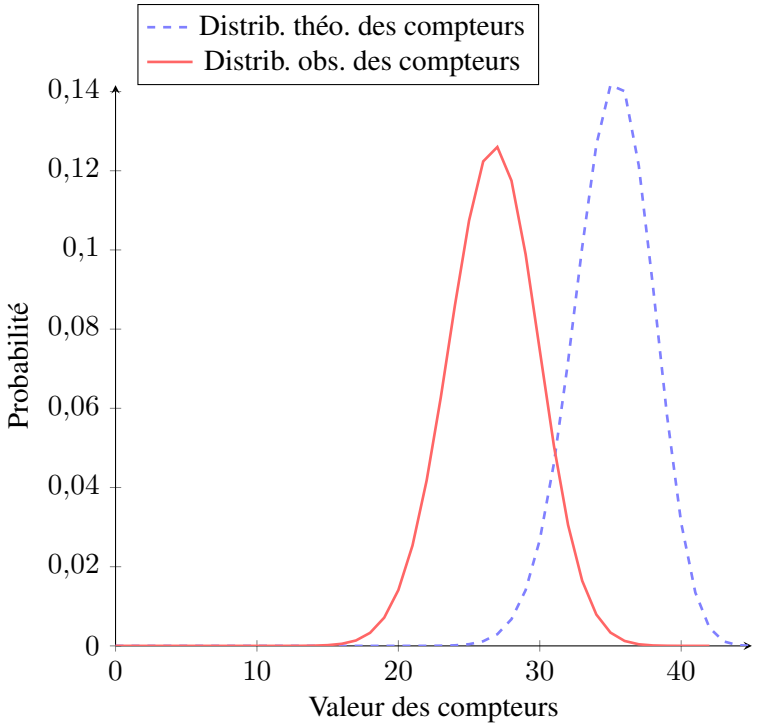


FIGURE 5.6 – Comparaison des distributions théoriques et observées des compteurs des positions fausses au début de la deuxième itération

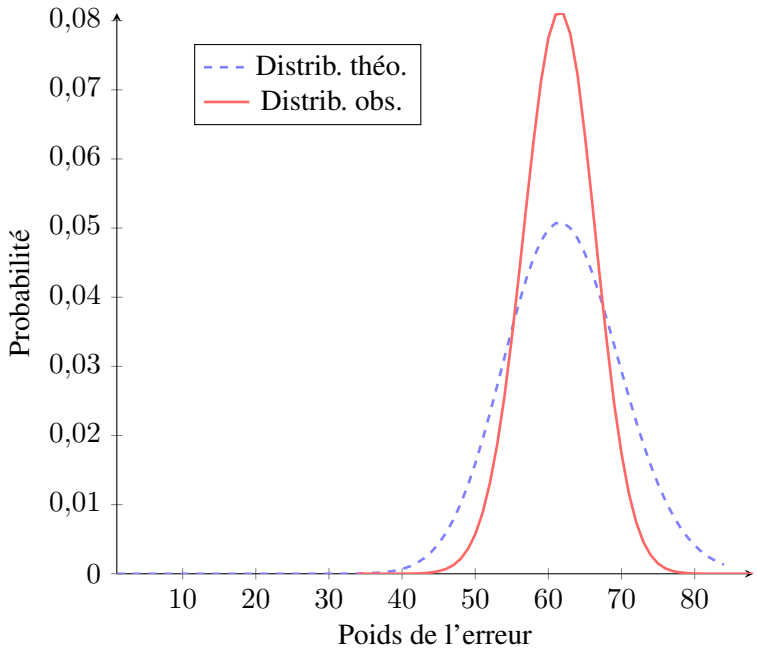


FIGURE 5.7 – Comparaison des distributions théoriques et observées du poids de l'erreur à la fin de la première itération

	Valeurs théoriques	Valeurs observées
$\mathbb{E}[t_1]$	62.069	61.403
$\mathbb{V}[t_1]$	61.668	23.937

TABLE 5.5 – Comparaison de l'espérance et de la variance du poids de l'erreur à la fin de la première itération

Les figures 5.3, 5.4 et la table 5.3 indiquent que la distribution des compteurs des positions justes et fausses respectivement, à la première itération sont bien approximées. Cependant, dès le début de la seconde itération, les figures 5.6, 5.6 et la table 5.4 montrent que les distributions théoriques et observées ne sont pas assez proches pour que les modèles théoriques utilisés fournissent des approximations précises. Entres autres, nous pouvons voir, grâce à la figure 5.7 et la table 5.5 que la distribution théorique du poids de l'erreur à la fin de la première itération ne coïncide pas avec la distribution observée.

Les résultats montrent les limites de cette approche pour le cas des codes QC-MDPC où l'erreur à décoder est de poids fixe t. Premièrement, le modèle de $\boldsymbol{e} \sim \mathrm{BSC}\left(\frac{t}{n}\right)$ reste une approximation du modèle étudié, puisqu'il ne prend pas en compte le fait que le poids de l'erreur initiale est fixé ($t = 84$), mais utilise une erreur dont le poids tend asymptotiquement vers t.

Remarque 29. *D'après la borne de Chernoff, si* $\forall j \in \{1, \ldots, n\}$, $e_j \sim B(\frac{t}{n})$, *alors pour tout* $\varepsilon > 0$, $\mathbb{P}[|\operatorname{wt}(\boldsymbol{e}) - t| \leqslant \varepsilon t] \leqslant e^{-\frac{t\varepsilon^2}{3}}$.

Deuxièmement, dans l'analyse proposée ci-dessus, nous considérons que les variables aléatoires étudiées ne sont pas corrélées, ce qui reste une approximation du modèle réel dans le cas des codes MDPC dont le poids des équations est très supérieur à celui des codes LDPC.

Enfin, nous avons supposé que l'erreur suivait une loi binomiale quelque soit l'itération considérée, c'est-à-dire que chaque position de l'erreur à une probabilité fixé d'être égale à 1. Dans le cas des codes LDPC, chaque itération peut être analysée indépendamment. Cependant, les résultats présentés dans cette section montrent que nous ne pouvons adopter ce raisonnement pour les codes MDPC.

5.4 Conclusion

L'algorithme de *bit flipping* reste une solution pertinente pour le décodage des codes MDPC dans un contexte cryptographique du fait de la simplicité de sa mise en œuvre. Cependant, dans ce contexte, il est difficile de garantir une probabilité d'échec au décodage fixée. En effet, nous avons vu que l'estimation de celle-ci dépend de la précision avec laquelle nous pouvons estimer la distribution de l'erreur à la fin de chaque itération. Rappelons que cette distribution, quant à elle, dépend de la distribution des compteurs des positions justes et fausses respectivement et de la valeur du seuil à chaque itération. Les distributions des compteurs sont plutôt bien comprises à la première itération (voir les figures 5.3, 5.4 et la table 5.3). Ce qui implique que la valeur théorique du seuil optimal à la première itération semble être fiable. Cependant, à la deuxième itération, la précision des estimations de ces distributions dépend de celle de la distribution du poids de l'erreur à la fin de la première itération, qui semble être un peu moins fidèle à nos observations dans ce cas (voir la figure 5.7 et la table 5.5).

Les estimations fournies par ce modèle sont donc insuffisantes, dans notre contexte, pour : d'une part, donner une bonne approximation théorique de la probabilité d'échec au décodage et d'autre part, trouver les valeurs des seuils, pour chaque itération, permettant d'obtenir une probabilité d'échec au décodage faible.

5.5 Mise en œuvre : état de l'art

Les premières propositions de décodeurs pour les codes MDPC utilisent une autre méthode pour le calcul du seuil. Dans [MTSB12], Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier et Paulo Barreto suggèrent de fixer, pour toute itération, $b = \max_{1 \leqslant j \leqslant n}(\sigma_j) - \delta$ où δ est un petit entier (typiquement 5) et σ_j correspond au nombre d'équations de parité insatisfaites impliquant la j-ème position de l'erreur. Si le décodage échoue au bout des I itérations, une nouvelle instance de décodage est effectuée avec un δ plus petit. Le principal avantage de cette méthode est que les décisions sont très conservatrices puisqu'en générale, un petit nombre de positions ont une valeur de seuil très élevée par rapport à la moyenne. D'un autre côté, le décodage nécessite un nombre d'itérations plutôt grand puisque peu de positions sont modifiées à chaque itération.

Les auteurs de [HvMG13] proposent une implémentation sur FPGA et AVR *microcontrollers* du cryptosystème de McEliece utilisant les codes QC-MDPC. Dans ces environnements, le nombre d'opérations à exécuter et la taille des données à manipuler doivent être les plus faibles possibles. Dans cet article, les auteurs fournissent un nouvel algorithme de décodage ainsi qu'une comparaison de l'efficacité et de la performance de différents décodeurs. La première idée exposée dans cet article consiste à mettre à jour le syndrome après chaque modification du vecteur d'erreur. La deuxième idée étant de comparer le poids de ce syndrome à zéro dès qu'une telle mise à jour est effectuée, de sorte que l'algorithme soit arrêté dès que le syndrome est nul (sans attendre la fin de l'itération). Au vu des résultats présentés dans [HvMG13], il apparaît que le décodeur nécessitant le nombre moyen d'itérations le plus faible et dont la probabilité d'échec au décodage est la plus petite soit un décodeur utilisant cette nouvelle technique pour la mise à jour du syndrome et dont les seuils sont précalculés grâce aux formules de [Gal63].

Remarque 30. *Pour $H \in \mathcal{H}(n, r, w)$ et $\boldsymbol{e} \in \mathbb{F}_2^n$, posons $\boldsymbol{s} = \boldsymbol{e}H^t$ et $\boldsymbol{h}_i$, pour $1 \leqslant i \leqslant r$ les équations de parités définies par H. Nous avons déjà remarqué (voir la propriété 18) que $\sigma_j = |\{i \,:\, 1 \leqslant i \leqslant r,\ j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1\}| = |\boldsymbol{s} \cap \boldsymbol{h}_{*,j}|$. De façon similaire, le poids du syndrome est mis à jour en utilisant : $\boldsymbol{s} \leftarrow \boldsymbol{s} \oplus \boldsymbol{h}_{*,j}$ lorsque $\sigma_j \geqslant b$ où b est le seuil de l'itération courante.*

En 2014, Ingo Von Maurich et Tim Güneysu proposent une implémentation du même cryptosystème sur FPGA [vMG14a]. Le but de cette proposition étant de fournir une implémentation à bas coût (*lightweight encryption*) de celui-ci. Les résultats explicités dans cet article indiquent que les performances de ce cryptosystème sont très raisonnables et montrent donc que son utilisation est appropriée dans un environnement contraint.

Une étude rigoureuse de la possibilité d'utiliser ce cryptosystème dans des contextes restreints doit prendre en compte d'éventuelles attaques par canaux cachés résultant de l'analyse de la consommation de courant, du temps etc. des différentes opérations effectuées. L'article [CEvMS15] présente la première attaque de ce type sur le cryptosystème de McEliece utilisant les codes QC-MDPC. Les auteurs montrent en effet qu'il est possible d'utiliser une analyse différentielle de la consommation de courant (*Differential Power Analysis*) lors du calcul du syndrome, afin de retrouver la matrice de parité creuse secrète. [vMG14b] propose de nouvelles attaques utilisant une analyse de la consommation de courant (*Simple Power Analysis*) d'une part et une attaque temporelle (*Timing Attack*) d'autre part. Cependant, cet article présente des contremesures pour ce type d'attaques, qui emploient des techniques de masquage, ce qui laisse à penser que ces contremesures peuvent être améliorées et étendues pour différents types d'attaque par canaux cachés.

Pour se protéger contre les attaques temporelles, une autre solution consiste à utiliser une implémentation à temps constant. Dans ce cas, chacune des étapes de l'algorithme requiert le même nombre d'opérations et le même temps pour s'effectuer. En 2016, Tung Chou propose une implémentation à temps constant du cryptosystème de Niederreiter utilisant les codes QC-MDPC dans [Cho16]. L'autre particularité de cette proposition est qu'elle emploie le formalisme polynomial des codes QC-MDPC afin d'augmenter l'efficacité des opérations.

Enfin, [MOG15] fournit une implémentation de ce cryptosystème sur différents supports, y compris en *software* ainsi qu'une nouvelle technique pour le choix du seuil. En effet, les auteurs proposent d'utiliser les seuils précalculés grâce aux formules de [Gal63], tout en précisant que si le décodage échoue, ces seuils seront incrémentés avant de tenter un nouveau décodage avec ces nouveaux seuils. Les résultats présentés dans cet article indiquent que cette technique, couplée avec le test sur la valeur du syndrome après chaque modification de bit, permet d'atteindre les mêmes performances que le meilleur décodeur présenté dans [HvMG13].

Le cryptosystème de McEliece (ou de Niederreiter) avec les codes QC-MDPC est donc bien adapté aux contextes où l'environnement est contraint. En effet, d'une part, les données à manipuler sont de tailles raisonnables. D'autre part, le chiffrement et le déchiffrement peuvent être optimisés selon les propriétés des processeurs disponibles. En effet, dans le cas classique, il est possible de n'utiliser que les opérations binaires usuelles. Et si le processeur est adapté pour cet usage, l'emploi du formalisme polynomial permet d'effectuer les opérations efficacement dans un anneau de polynôme.

Partie III

Analyse du décodage

Introduction

L'utilisation des codes QC-MDPC dans le cryptosystème de McEliece permet de construire un schéma de chiffrement post-quantique dont les clés ont une taille raisonnable et dont le chiffrement et le déchiffrement n'utilisent que des opérations binaires. C'est donc un bon candidat pour l'implémentation embarquée, sur des microcontrôleurs, FPGA ou des implémentations à bas coût ([HvMG13, vMG14a, MOG15, Cho16, BCS13]).

Dans ce contexte, il faut tout particulièrement prendre en compte, lors de l'analyse de sécurité, le fait que certaines informations sur la consommation de courant ou le temps de déchiffrement peuvent être exploitées pour construire des attaques par canaux cachés.

Ici, l'algorithme de déchiffrement consiste principalement à décoder un mot de code bruité. L'algorithme de décodage appliqué est un algorithme itératif et probabiliste. Autrement dit, le nombre d'itérations nécessaires pour retrouver le mot de code initial varie en fonction des instances. De plus, certains décodages peuvent échouer, dans le cas où le poids de l'erreur augmente au cours des itérations. Ces deux comportements ne sont pas souhaitables, car ils peuvent permettre d'extraire des informations sur le secret (la matrice de parité et l'erreur utilisées). En effet, récemment, [GJS16] fournit une attaque du cryptosystème de McEliece qui exploite le fait que la probabilité d'échec au décodage soit non nulle. Cependant, en pratique, pour les paramètres proposés dans [MTSB12] et en adoptant une technique de conversion sémantiquement sûre, cette attaque semble trop coûteuse.

Une contremesure possible contre ce type d'attaque est de limiter le nombre d'instances de chiffrement/déchiffrement avec les mêmes clés. Une autre façon de s'en protéger est de recourir à un décodage à temps constant, dont la probabilité d'échec au décodage est négligeable, c'est-à-dire de l'ordre du niveau de sécurité du cryptosystème. Pour ce faire, il est possible, par exemple, de fixer le nombre d'itérations de l'algorithme et de simuler de fausses itérations si nécessaire. Dans un souci de performance, le nombre maximal d'itérations ne doit pas être trop élevé. Cependant, d'un autre côté, il doit être assez grand pour que toutes les instances sauf un nombre négligeable aboutissent.

L'algorithme de *bit flipping* (voir section 5.2) a été introduit comme décodeur pour les codes LDPC et lorsque l'erreur suit le modèle du canal binaire symétrique. Dans la section 5.3, nous avons vu que l'analyse du comportement de l'algorithme fournie par Gallager, explicitée dans [Gal63], pouvait être adaptée pour le décodage des codes MDPC. Au vu des résultats présentés dans cette section, il apparaît que cette analyse n'est pas assez fine pour donner de bonnes garanties sur la probabilité d'échec au décodage dans un contexte cryptographique. Autrement dit, les estimations

fournies par cette analyse théorique ne coïncident pas fidèlement aux valeurs que nous avons pu observer en pratique.

L'enjeu de cette partie est donc de fournir d'autres outils pour analyser le comportement du décodeur. Ces résultats pourront peut être, à terme, permettre d'optimiser le décodages des codes MDPC dans un contexte cryptographique, voire de fournir une analyse théorique précise de la probabilité d'échec au décodage de ceux-ci.

6 Amélioration du calcul du seuil

Afin d'affiner la compréhension et le comportement de l'algorithme de *bit flipping* lorsque celui-ci est utilisé pour décoder les codes MDPC dans un contexte cryptographique, nous allons expliciter différentes techniques permettant d'améliorer le calcul du seuil. Cette amélioration ayant pour but, non pas de rendre l'algorithme le plus efficace possible en moyenne, mais de diminuer la probabilité d'échec au décodage après un petit nombre d'itérations. La première piste évoquée est de reprendre l'analyse de Gallager en changeant le modèle d'erreur considérée. En effet, dans un contexte cryptographique, l'erreur est tirée aléatoirement parmi un ensemble de vecteurs de longueur et de poids fixe. Le modèle du canal binaire symétrique ne semble donc pas très adapté pour ce contexte. La deuxième piste que nous expliciterons, consiste à employer un processus heuristique, pour trouver "à la main" les seuils qui permettent de minimiser la probabilité d'échec au décodage. L'idée principale étant de trouver la meilleure valeur de seuil en fonction du poids du syndrome observé et non de l'itération courante.

6.1 Calcul du seuil

Comme nous l'avons précisé dans l'introduction, nous allons étudier l'emploi de l'algorithme de décodage 2 lors de l'étape de déchiffrement du cryptosystème de McEliece (ou de tout autre primitive cryptographique utilisant les codes MDPC). Ainsi, les données que nous connaissons, *i.e.* qui sont observables, sont :

— la matrice de parité $H \in \mathcal{H}(n, r, w)$ du code MDPC considéré (la clé privée),

— le chiffré qui est un mot de code bruité,

— le syndrome (et son poids) du chiffré, qui est mis à jour à chaque itération,

— le nombre d'équations de parité non vérifiées par chacune des positions du chiffré.

L'erreur n'est donc pas connue mais nous savons que l'erreur initiale à un poids fixe t. Ainsi, nous pouvons donner une estimation des distributions de certaines grandeurs qui dépendent de l'erreur, pour la première itération en considérant le modèle où l'erreur est tirée uniformément dans l'ensemble des vecteurs de $\mathbb{F}_2^n$ de poids t. Dans la suite, cet ensemble, muni d'une loi uniforme, sera noté $\mathcal{U}_t$.

Comme précédemment, $\mathcal{H}(n, r, w)$ désignera l'ensemble des matrices de $\mathbb{F}_2^{r\times n}$ dont les lignes ont un poids fixe w et les colonnes ont un poids fixe $d = \frac{w}{2}$.

Nous nous plaçons donc dans le modèle $\boldsymbol{e} \sim \mathcal{U}_t$ et nous considérons $H \in \mathcal{H}(n, r, w)$ fixée avec $\boldsymbol{h}_i$, pour $1 \leqslant i \leqslant r$ désignant les équations de parité définies par H.

Dans cette section, nous étudierons les distributions de certaines variables aléatoires au cours de la première itération de l'algorithme 2. Dans un premier temps, nous allons expliciter la probabilité qu'une équation de parité soit insatisfaite. Ceci nous permettra de donner la distribution des compteurs des positions justes et fausses au cours de la première itération. Nous comparerons ces résultats aux résultats obtenus par simulation du décodage sur des codes $[9602, 4801, 90]$-MDPC lorsque l'erreur initial a un poids fixe t. La donnée de ces distributions nous permettra, dans un second temps, de déterminer la valeur optimale du seuil à la première itération dans ce modèle. Enfin, nous donnerons une estimation de la distribution du poids de l'erreur après la première itération du décodage, lorsque le seuil optimal est utilisé. Ces résultats seront comparés aux résultats obtenus par simulation.

6.1.1 Probabilité sur les équations de parité

Dans un premier temps, nous allons expliciter la probabilité qu'une équation de parité soit fausse en fonction de la valeur du bit e_j.

De façon analogue à l'analyse présentée dans la section 5.2, nous allons supposer que pour $i \in \{1, \dots, r\}$, les variables aléatoires $\langle \boldsymbol{h}_i, \boldsymbol{e}\rangle \ : \mathcal{U}_t \to \{0, 1\}$ sont indépendantes et identiquement distribuées. Ces hypothèses nous permettent de donner les distributions théoriques de ces variables aléatoires qui correspondent bien, comme nous allons le voir, aux résultats observés en pratique. Pour $j \in \{1, \dots, n\}$ fixé, nous noterons $\mathcal{H}_0$ l'hypothèse $j \notin \boldsymbol{e}$ et $\mathcal{H}_1$ l'hypothèse $j \in \boldsymbol{e}$.

Propriété 23. *Soient $\boldsymbol{e} \sim \mathcal{U}_t$, $H \in \mathcal{H}(n, r, w)$ et $\boldsymbol{h}_i$ où $1 \leqslant i \leqslant r$ les équations de parité définies par H. Pour $j \in \{1, \dots, n\}$ et $\forall i \in \{1, \dots, r\}$ notons $p^1_{n,w,t} \overset{def}{=} \mathbb{P}[\langle \boldsymbol{h}_i, \boldsymbol{e}\rangle = 1]$ sous l'hypothèse $\mathcal{H}_1$ et $p^0_{n,w,t} \overset{def}{=} \mathbb{P}[\langle \boldsymbol{h}_i, \boldsymbol{e}\rangle = 1]$ sous l'hypothèse $\mathcal{H}_0$. On a*

$$p^1_{n,w,t} = \sum_{\substack{\ell=0 \\ \ell \text{ pair}}}^{t-1} \frac{\binom{w-1}{\ell}\binom{n-w}{t-1-\ell}}{\binom{n-1}{t-1}} \qquad \textit{et } p^0_{n,w,t} = \sum_{\substack{\ell=0 \\ \ell \text{ impair}}}^{t} \frac{\binom{w-1}{\ell}\binom{n-w}{t-\ell}}{\binom{n-1}{t}}$$

Démonstration. Considérons $\boldsymbol{h}$ une équation de parité de H dont le j-ième coefficient est non nul, c'est-à-dire telle que $j \in \boldsymbol{h}$.

Premièrement, si $j \in \boldsymbol{e}$ alors $\langle \boldsymbol{h}, \boldsymbol{e}\rangle = \bigoplus_{j'=0}^{n} h_{j'}e_{j'} = e_j h_j \bigoplus_{j'\neq j} h_{j'}e_{j'}$. Mais, comme $j \in \boldsymbol{e}$ et $j \in \boldsymbol{h}$ par hypothèse, $\langle \boldsymbol{h}, \boldsymbol{e}\rangle = 1 \oplus \Big(\bigoplus_{j'\neq j} h_{j'}e_{j'}\Big)$.

Donc

$$\mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e}\rangle = 1 | j \in \boldsymbol{e}] = \mathbb{P}[\bigoplus_{j'\neq j} h_{j'}e_{j'} = 0 | j \in \boldsymbol{e}]$$

Or, $\bigoplus_{j' \neq j} h_{j'} e_{j'} = 0 \Leftrightarrow |(\boldsymbol{h} - \{j\}) \cap (\boldsymbol{e} - \{j\})| = 2\ell$ avec $\ell \in \mathbb{N}$.

Autrement dit, pour que $\bigoplus_{j' \neq j} h_{j'} e_{j'} = 0$, il faut que 2ℓ positions de $\boldsymbol{e} - \{j\}$ coïncident avec des positions de $\boldsymbol{h} - \{j\}$. Nous devons donc placer 2ℓ positions de $\boldsymbol{e} - \{j\}$ parmi les $|\boldsymbol{h} - \{j\}| = w - 1$ positions de $\boldsymbol{h} - \{j\}$. Les $|\boldsymbol{e} - \{j\}| - 2\ell = t - 1 - 2\ell$ positions restantes seront placées parmi les positions de $\boldsymbol{h} - \{j\}$ dont la valeur est nulle.

Ainsi,

$$\mathbb{P}[|(\boldsymbol{h} - \{j\}) \cap (\boldsymbol{e} - \{j\})| = 2\ell | j \in \boldsymbol{e}] = \frac{\binom{w-1}{2\ell}\binom{n-w}{t-1-2\ell}}{\binom{n-1}{t-1}}$$

En conclusion,

$$\mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1 | j \in \boldsymbol{e}] = \sum_{2\ell} \mathbb{P}[|(\boldsymbol{h} - \{j\}) \cap (\boldsymbol{e} - \{j\})| = 2\ell | j \in \boldsymbol{e}]$$

Deuxièmement, si $j \notin \boldsymbol{e}$ alors $\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 0 \oplus \bigoplus_{j' \neq j} h_{j'} e_{j'} = \bigoplus_{j' \neq j} h_{j'} e_{j'}$.

Ici, $\bigoplus_{j' \neq j} h_{j'} e_{j'} = 1 \Leftrightarrow |(\boldsymbol{h} - \{j\}) \cap (\boldsymbol{e} - \{j\})| = 2\ell + 1$ avec $\ell \in \mathbb{N}$. Mais, comme $j \notin \boldsymbol{e}$, $|\boldsymbol{e} - \{j\}| = t$,

$$\mathbb{P}[|(\boldsymbol{h} - \{j\}) \cap (\boldsymbol{e} - \{j\})| = 2\ell + 1 | j \notin \boldsymbol{e}] = \frac{\binom{w-1}{2\ell+1}\binom{n-w}{t-(2\ell+1)}}{\binom{n-1}{t}}$$

On conclut grâce à

$$\mathbb{P}[\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1 | j \notin \boldsymbol{e}] = \sum_{2\ell+1} \mathbb{P}[|(\boldsymbol{h} - \{j\}) \cap (\boldsymbol{e} - \{j\})| = 2\ell + 1 | j \in \boldsymbol{e}]$$

□

6.1.2 Probabilité qu'une position de l'erreur soit modifiée

Nous nous intéressons maintenant à la probabilité que la valeur d'une position soit modifiée au cours de la première itération et nous verrons que cette probabilité dépend de la valeur de l'erreur à cette position. Comme précédemment, nous ferons l'hypothèse que le nombre d'équations de parité insatisfaites impliquant une position est indépendant de la position considérée.

Dans un premier temps, nous allons expliciter la distribution, pour $j \in \{1, \ldots, r\}$, des variables aléatoires

$$\sigma_j \stackrel{\text{def}}{=} |\{i \ : \ 1 \leqslant i \leqslant r, j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1\}|$$

La modélisation de ces variables aléatoires nous permet d'expliciter la probabilité qu'une position soit modifiée, suivant qu'elle appartienne au support de l'erreur ou non. Ainsi, nous pouvons d'une part, trouver le seuil de la première itération qui minimise (en moyenne) le poids de l'erreur après la première itération. D'autre part, cela fournit une estimation de la distribution du poids de l'erreur après la première itération, en fonction du seuil utilisé.

Propriété 24. *Soient $\boldsymbol{e} \sim \mathcal{U}_t$, $H \in \mathcal{H}(n, r, w)$ et $\boldsymbol{h}_i$ pour $1 \leqslant i \leqslant r$ les équations de parité définies par H. Pour $j \in \{1, \ldots, n\}$, la variable aléatoire σ_j une loi binomiale $\mathcal{B}\left(d, p^1_{n,w,t}\right)$ lorsque $\mathcal{H}_1$ est vérifiée, et $\mathcal{B}\left(d, p^0_{n,w,t}\right)$ lorsque $\mathcal{H}_0$ est vérifiée.*

Démonstration. σ_j est une somme de d variables aléatoires de Bernouilli iid par hypothèse. D'où le résultat (voir la remarque 25). □

Le tableau 6.1 reprend les résultats explicités jusqu'à présent. Contrairement à l'analyse décrite dans la section 5.2, le biais entre les distributions des σ_j selon que $\mathcal{H}_0$ ou $\mathcal{H}_1$ est vérifiée n'apparaît pas clairement. Cependant, comme le montre les figures 6.1, 6.2 et la table 6.2 ce biais existe bien, donc nous pouvons exploiter cette propriété pour choisir un seuil optimal dans ce modèle.

Sous l'hypothèse $\mathcal{H}_0$	Sous l'hypothèse $\mathcal{H}_1$
$p^0_{n,w,t} = \sum_{\substack{\ell=0 \\ \ell \text{ impair}}}^{t} \frac{\binom{w-1}{\ell}\binom{n-w}{t-\ell}}{\binom{n-1}{t}}$	$p^1_{n,w,t} = \sum_{\substack{\ell=0 \\ \ell \text{ pair}}}^{t-1} \frac{\binom{w-1}{\ell}\binom{n-w}{t-1-\ell}}{\binom{n-1}{t-1}}$
$\langle \boldsymbol{h}_i, \boldsymbol{e} \rangle \sim \mathrm{Ber}\left(p^0_{n,w,t}\right)$	$\langle \boldsymbol{h}_i, \boldsymbol{e} \rangle \sim \mathrm{Ber}\left(p^1_{n,w,t}\right)$
$\sigma_j \sim \mathcal{B}\left(d, p^0_{n,w,t}\right)$	$\sigma_j \sim \mathcal{B}\left(d, p^1_{n,w,t}\right)$
$\mathbb{E}[\sigma_j] = dp^0_{n,w,t}$	$\mathbb{E}[\sigma_j] = dp^1_{n,w,t}$

TABLE 6.1 – Récapitulatif des résultats suivant que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée

Afin d'illustrer la propriété 24, nous allons comparer les distributions théoriques et observées, des compteurs des positions justes et fausses au début de la première itération. Pour ce faire, nous considérons les codes $[9602, 4801, 90]$-MDPC et nous appliquons l'algorithme 2 pour décoder une erreur de poids $t = 84$. Les figures 6.1 et 6.2 correspondent respectivement aux distributions théoriques (données par la propriété 24) et observées des compteurs des positions justes et fausses. Pour une lecture plus précise de la différence entre ces distributions, nous explicitons la variance et l'espérance théorique et observée de ces distributions dans la table 6.2. Ces résultats indiquent que ces distributions sont très proches (voir les figures 6.1 et 6.2).

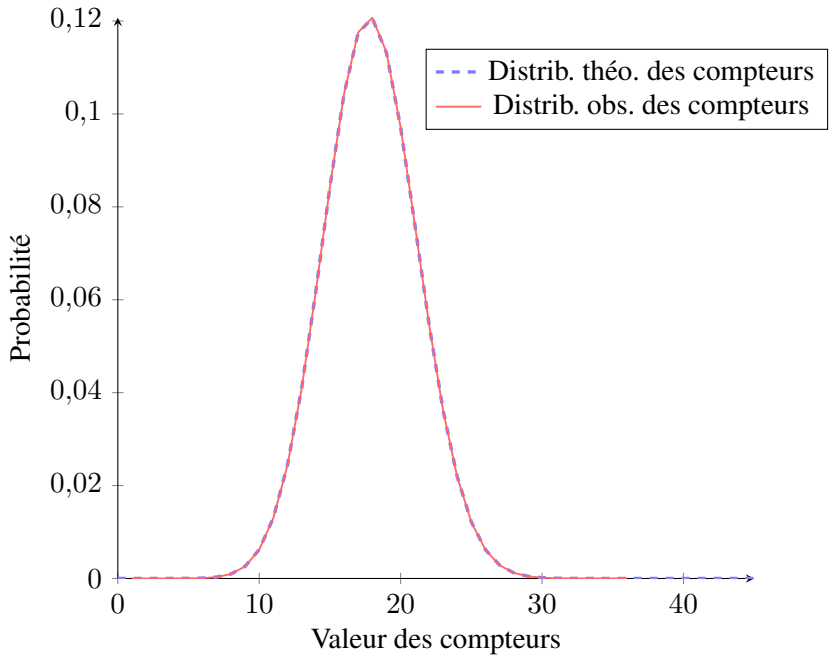


FIGURE 6.1 – Comparaison des distributions théoriques et observées des compteurs des positions justes au début de la première itération

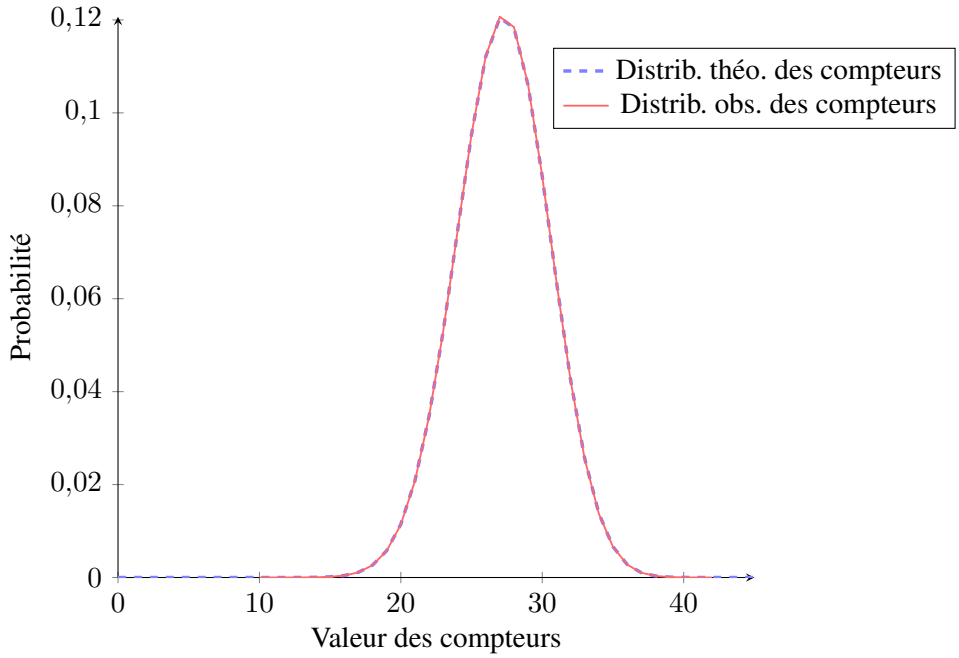


FIGURE 6.2 – Comparaison des distributions théoriques et observées des compteurs des positions fausses au début de la première itération

	Sous l'hypothèse $\mathcal{H}_0$	Sous l'hypothèse $\mathcal{H}_1$
$\mathbb{E}[\sigma_j]$ observée	17.893	27.195
$\mathbb{E}[\sigma_j]$ théorique	17.993	27.195
$\mathbb{V}[\sigma_j]$ observée	10.768	10.733
$\mathbb{V}[\sigma_j]$ théorique	10.778	10.760

TABLE 6.2 – Comparaison de l'espérance et de la variance des compteurs théoriques et observées, suivant que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée

Nous allons maintenant pouvoir expliciter la probabilité qu'une position soit modifiée au cours de la première itération en fonction du seuil utilisé et selon que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée.

Propriété 25. *Soient $\boldsymbol{e} \sim \mathcal{U}_t$ et $H \in \mathcal{H}(n,r,w)$. Pour $1 \leqslant j \leqslant n$, $P^1_{n,t,w,b_1} \overset{def}{=} \mathbb{P}[e_j^{(1)} = 0 | j \in \boldsymbol{e}]$ désigne la probabilité qu'un bit faux soit corrigé et $P^0_{n,t,w,b_1} \overset{def}{=} \mathbb{P}[e_j^{(1)} = 1 | j \notin \boldsymbol{e}]$ la probabilité qu'un bit juste soit modifié lors de la première itération du décodage, lorsque le seuil est égal à b_1. On a*

$$\begin{cases} P^1_{n,t,w,b_1} = \displaystyle\sum_{c \geqslant b_1} \binom{d}{c} {p^1_{n,w,t}}^c (1 - p^1_{n,w,t})^{d-c} \\ P^0_{n,t,w,b_1} = \displaystyle\sum_{c \geqslant b_1} \binom{d}{c} {p^0_{n,w,t}}^c (1 - p^0_{n,w,t})^{d-c} \end{cases}$$

Démonstration. Rappelons que pour $1 \leqslant j \leqslant n$, e_j est modifié si, et seulement si, $\sigma_j \geqslant b_1$. Donc

$$\begin{aligned} \mathbb{P}[e_j^{(1)} = 0 | j \in \boldsymbol{e}] &= \mathbb{P}[\sigma_j \geqslant b_1 | j \in \boldsymbol{e}] \\ &= \sum_{c=b_1}^{d} \mathbb{P}[\sigma_j = c | j \in \boldsymbol{e}] \end{aligned}$$

et de même pour

$$\begin{aligned} \mathbb{P}[e_j^{(1)} = 1 | j \notin \boldsymbol{e}] &= \mathbb{P}[\sigma_j \geqslant b_1 | j \in \boldsymbol{e}] \\ &= \sum_{c=b_1}^{d} \mathbb{P}[\sigma_j = c | j \in \boldsymbol{e}] \end{aligned}$$

□

Remarque 31. *Nous supposons ici que les paramètres n, w et t sont fixés, donc dans la suite, nous écrirons plus succinctement $p^1_{n,w,t} = p^1$, $p^0_{n,w,t} = p^0$, $P^1_{n,t,w,b_1} = P^1_{b_1}$ et $P^0_{n,t,w,b_1} = P^0_{b_1}$.*

6.1.3 Seuil optimal

Le calcul du seuil optimal en moyenne permet de trouver les seuils qui optimisent le décodage de la majorité des instances. Rappelons que nous connaissons la probabilité, pour les positions justes et fausses respectivement, de modifier la valeur de cette position au cours de la première itération. Nous allons donc décrire une méthode qui permet de minimiser le poids de l'erreur obtenue après la première itération de l'algorithme de décodage 2, en choisissant une valeur de seuil adaptée.

Rappelons, que dans la sous-section précédente, nous avons montré que, pour $1 \leqslant j \leqslant n$ fixé :

— sous $\mathcal{H}_0$, $\sigma_j \sim \mathcal{B}(d, p^0)$,

— sous $\mathcal{H}_1$, $\sigma_j \sim \mathcal{B}(d, p^1)$.

Nous cherchons maintenant à estimer, pour $0 \leqslant c \leqslant d$, les valeurs de $Y_c^0 \stackrel{\text{def}}{=} |\{j \ : \ j \notin \boldsymbol{e} \text{ et } \sigma_j = c\}|$ et $Y_c^1 \stackrel{\text{def}}{=} |\{j \ : \ j \in \boldsymbol{e} \text{ et } \sigma_j = c\}|$ qui correspondent respectivement, au nombre de positions justes et fausses de l'erreur qui ont une valeur de compteur donnée.

Nous pouvons remarquer que $Y_c^0 = \sum_{j=1}^{n} \mathbb{1}_{j \notin \boldsymbol{e} \text{ et } \sigma_j = c}(j) = \sum_{j \notin \boldsymbol{e}} \mathbb{1}_{\sigma_j = c}(j)$ où $\mathbb{1}$ est la fonction indicatrice usuelle (*i.e.* $\mathbb{1}_A(j) = 1 \Leftrightarrow j \in A$ et vaut 0 si $j \notin A$, avec A un ensemble quelconque). Autrement dit,

$$\mathbb{E}[Y_c^0] = (n - |\boldsymbol{e}|) \cdot \mathbb{P}[\sigma_j = c | j \notin \boldsymbol{e}] \tag{6.1}$$

De la même façon, nous pouvons voir que

$$\mathbb{E}[Y_c^1] = |\boldsymbol{e}| \cdot \mathbb{P}[\sigma_j = c | j \in \boldsymbol{e}] \tag{6.2}$$

La figure 6.3 représente les courbes qui correspondent à l'équation 6.1 (pour les positions justes) et l'équation 6.2 (pour les positions fausses), pour $n = 9602$, $r = 4801$, $w = 90$, $d = 45$ et $t = 84$.

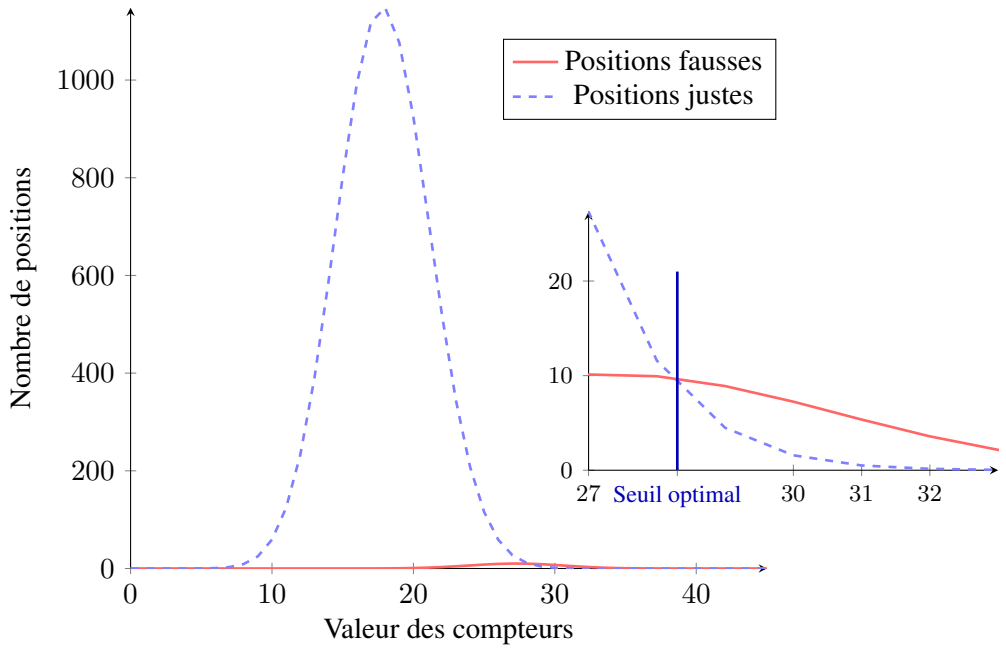


FIGURE 6.3 – Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la première itération

Remarque 32. *D'après ce graphique, pour les codes* $[9802, 4801, 90]$*-QC-MDPC et un poids d'erreur initial* $t = 84$*, nous estimons que le seuil optimal pour la première itération est* $b_1 = 29$*.*

6.1.4 Distribution du poids de l'erreur

À la première itération, le poids de l'erreur est connue (c'est un des paramètres du cryptosystème) et cela nous permet de fournir les distributions précises des valeurs des compteurs des positions justes et fausses. Pour exprimer ces distributions à l'itération suivante, nous devons, d'une part, estimer la distribution du poids de l'erreur à la fin de la première itération et d'autre part, connaître la valeur du seuil qui a été utilisé. En effet, le poids de l'erreur à la fin de la première itération, notée $\boldsymbol{e}^{(1)}$, peut être représenté comme la différence entre le poids de l'erreur $\boldsymbol{e}$ et le nombre de positions justes et fausses qui vont être changées au cours de l'itération.

Propriété 26. *Soient $\boldsymbol{e} \sim \mathcal{U}_t$ et $H \in \mathcal{H}(n, r, w)$. Nous considérons $\boldsymbol{e}^{(1)}$ l'erreur obtenue à la fin de la première itération de l'algorithme 2. Notons $\Pi(n, w, t, b_1, t_1) \stackrel{def}{=} \mathbb{P}[|\boldsymbol{e}^{(1)}| = t_1]$ la probabilité que le poids de l'erreur vaille t_1 en supposant que le seuil est égal à b_1. On a*

$$\Pi(n, w, t, b_1, t_1) = \sum_{c=0}^{t-t_1} \binom{n-t}{c} {P_{b_1}^0}^{c} (1 - P_{b_1}^0)^{n-t-c} \binom{t}{c + (t - t_1)} {P_{b_1}^1}^{c+(t-t_1)} (1 - P_{b_1}^1)^{t-c-(t-t_1)}$$

Démonstration. On a $\mathbb{P}[|\boldsymbol{e}^{(1)}| = t_1] = \mathbb{P}[|\boldsymbol{e}^{(1)}| = t - \delta]$ où δ est le nombre de positions de l'erreur effectivement corrigées à la première itération. Ici, $\delta = \delta^- - \delta^+$ où δ^+ est le nombre de positions justes de l'erreur que l'on a modifié et δ^- est le nombre de positions fausses de l'erreur que l'on a corrigé. C'est-à-dire, $\delta^+ \stackrel{\text{def}}{=} |\{j \notin \boldsymbol{e} \ : \ \boldsymbol{e}_j^{(1)} = 1\}|$ et $\delta^- \stackrel{\text{def}}{=} |\{j \in \boldsymbol{e} \ : \ \boldsymbol{e}_j^{(1)} = 0\}|$.

Soit d^+ la variable aléatoire suivante :

$$\begin{array}{rccl} d^+ : \{1, \ldots, n\} - \boldsymbol{e} & \longrightarrow & \{0, 1\} \\ j & \longmapsto & \begin{cases} 1 & \text{si } j \in \boldsymbol{e}^{(1)} \\ 0 & \text{sinon} \end{cases} \end{array}$$

Par définition $d^+ \sim \mathrm{Ber}(P_{b_1}^0)$, car $\mathbb{P}[d^+ = 1] = \mathbb{P}[e_j^{(1)} = 1 | j \notin \boldsymbol{e}]$

De même, on définit

$$\begin{array}{rccl} d^- : \boldsymbol{e} & \to & \{0, 1\} \\ j & \mapsto & \begin{cases} 1 & \text{si } j \notin \boldsymbol{e}^{(1)} \\ 0 & \text{sinon} \end{cases} \end{array}$$

et on a $d^- \sim \mathrm{Ber}(P_{b_1}^1)$.

De plus, on a $\delta^+ = \sum\limits_{j \notin \boldsymbol{e}} d^+(j)$ et $\delta^- = \sum\limits_{j \in \boldsymbol{e}} d^-(j)$.

Donc nous pouvons exprimer δ^+ et δ^- respectivement, comme des sommes de variables aléatoires de Bernouilli iid et, comme nous l'avons mentionné dans la remarque 25, nous pouvons conclure que $\delta^+ \sim \mathcal{B}\left(n - |\boldsymbol{e}|, P_{b_1}^0\right)$ et $\delta^- \sim \mathcal{B}\left(|\boldsymbol{e}|, P_{b_1}^1\right)$.

D'un autre côté, on a

$$\begin{aligned}\mathbb{P}[\boldsymbol{e}^{(1)} = t - \delta] &= \mathbb{P}[\bigcup_{c=0}^{\delta} \{\delta^+ = c\} \cap \{\delta^- = \delta + c\}] \\ &= \sum_{c=0}^{\delta} \mathbb{P}[\delta^+ = c]\mathbb{P}[\delta^- = c + \delta]\end{aligned}$$

D'où le résultat.

□

Remarque 33. *Les paramètres n et w étant fixés, nous désignerons plus succinctement $\Pi(n, w, t, b_1, t_1) = \Pi(t, b_1, t_1)$.*

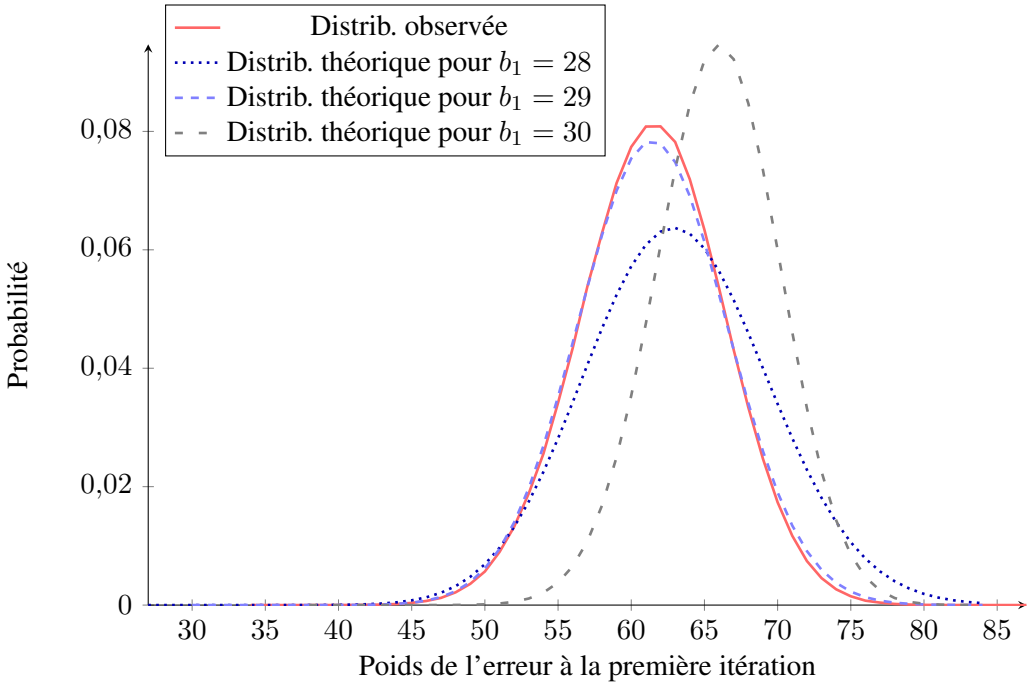


FIGURE 6.4 – Comparaison des distributions du poids de l'erreur après une itération du *bit flipping* observées pour un seuil fixe (29) et les valeurs théoriques pour différents seuils

	Valeurs théoriques			Valeurs observées
Seuil utilisé	28	29	30	29
$\mathbb{E}[t_1]$	63.061	61.446	65.858	61.395
$\mathbb{V}[t_1]$	38.975	25.842	17.733	23.958

TABLE 6.3 – Comparaison de l'espérance et de la variance du poids de l'erreur à la fin de la première itération, pour différentes valeurs de seuils

La figure 6.4 et le tableau 6.3 indiquent que les modèles théoriques ne sont pas assez précis pour estimer finement la distribution de l'erreur après la première itération de l'algorithme de *bit flipping*. Une des raisons de cette divergence semble être que, dans tous les calculs de distributions exprimés jusqu'à présent, nous avons supposé les équations de parités mutuellement indépendantes.

À titre indicatif, nous pouvons expliciter la distribution du poids de l'erreur après la première itération en fonction de b_1 et b_2 comme explicité ci-dessous.

Propriété 27. *Soient $\boldsymbol{e} \sim \mathcal{U}_t$ et $H \in \mathcal{H}(n, r, w)$. Nous considérons $\boldsymbol{e}^{(2)}$ l'erreur obtenue à la fin de la deuxième itération de l'algorithme 2. Nous supposons de plus que les seuils de la première et de la deuxième itération de l'algorithme sont fixés à b_1 et b_2 respectivement. Dans ce cas, nous avons*

$$\mathbb{P}[|\boldsymbol{e}^{(2)}| = t_2] = \sum_{T=0}^{t} \mathit{\Pi}(t, b_1, l) \cdot \mathit{\Pi}(T, b_2, t_2)$$

Démonstration. Il suffit de remarquer que $\mathit{\Pi}(t_{u+1}, b_u, t_u) = \mathbb{P}[|\boldsymbol{e}^{(u+1)}| = t_{u+1}| \ |\boldsymbol{e}^{(u)}| = t_u]$ pour un seuil égal à b_u.
Cela implique que :

$$\begin{aligned}\mathbb{P}[|\boldsymbol{e}^{(2)}| = t_2] &= \sum_{T=0}^{t} \mathbb{P}[|\boldsymbol{e}^{(1)}| = T] \cdot \mathbb{P}[|\boldsymbol{e}^{(2)}| = t_2| \ |\boldsymbol{e}^{(1)}| = T] \\ &= \sum_{T=0}^{t} \mathit{\Pi}(t, b_1, T) \cdot \mathit{\Pi}(T, b_2, t_2)\end{aligned}$$

□

La figure 6.5 indique que l'estimation théorique de la distribution du poids de l'erreur à la fin de la deuxième itération ne correspond pas précisément à ce que nous pouvons observer en pratique. De plus, nous pouvons remarquer que les valeurs théoriques obtenues sont beaucoup plus favorables que nos observations. Ce modèle ne peut donc être considéré comme une borne théorique dans le pire cas.

6.1.5 Distribution du poids du syndrome

L'analyse de la distribution du poids du syndrome n'a pas pour objectif, *a priori*, d'améliorer le calcul du seuil. Cependant, celle-ci nous intéresse pour comprendre le comportement de l'algorithme.

Comme précédemment, la distribution du poids du syndrome dépend du poids de l'erreur considéré. En ce sens, l'estimation la plus précise que nous pouvons donner (sans connaître précisément la distribution du poids de l'erreur à la fin de chaque itération), correspond au poids du syndrome avant la première itération.

Nous nous plaçons donc toujours dans le modèle $\boldsymbol{e} \sim \mathcal{U}_t$ et nous considérons $H \in \mathcal{H}(n, r, w)$ ainsi que $\boldsymbol{h}_i$ pour $1 \leqslant i \leqslant r$ les équations de parité définies par H. De plus, nous définissons $\boldsymbol{s} \stackrel{\text{def}}{=} \boldsymbol{e}H^t$ la variable aléatoire qui correspond au syndrome. Ainsi, pour $1 \leqslant i \leqslant r$, nous désignons de manière équivalente, les variables aléatoires supposées iid $s_i \stackrel{\text{def}}{=} \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$.

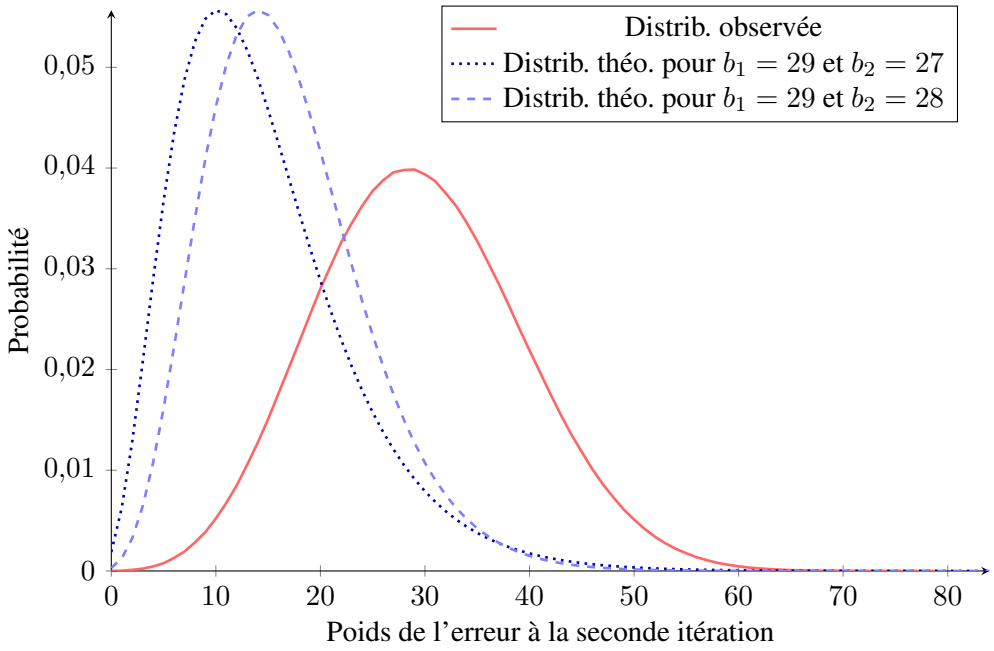


FIGURE 6.5 – Comparaison des distributions du poids de l'erreur après deux itérations du *bit flipping* observées pour les seuils $b_1 = 29$ et $b_2 = 27$ et les valeurs théoriques pour différents seuils

Propriété 28. *Soient $\boldsymbol{e} \sim \mathcal{U}_t$, $H \in \mathcal{H}(n, r, w)$ et $\boldsymbol{h}_i$, pour $1 \leqslant i \leqslant r$, les équations de parité définies par H. Posons $\boldsymbol{s} \stackrel{def}{=} \boldsymbol{e}H^t$. On a, pour $1 \leqslant i \leqslant r$,*

$$\mathbb{P}[s_i = 1] = \sum_{\substack{\ell=0 \\ \ell \text{ impair}}}^{w} \frac{\binom{w}{\ell}\binom{n-w}{t-\ell}}{\binom{n}{t}}$$

Démonstration. Pour $1 \leqslant i \leqslant r$, $s_i = \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$ par définition. Ainsi, $s_i = 1 \Leftrightarrow \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 2\ell + 1$ avec $\ell \in \mathbb{N}$.

Autrement dit, pour que $s_i = 1$ il faut que $2\ell + 1$ positions non nulles de l'erreur (de poids t) appartiennent au support de l'équation $\boldsymbol{h}_i$ (qui contient w éléments). D'où le résultat. □

Propriété 29. *Soient $\boldsymbol{e} \sim \mathcal{U}_t$, $H \in \mathcal{H}(n, r, w)$, $\boldsymbol{h}_i$, pour $1 \leqslant i \leqslant r$, les équations de parité définies par H et $\boldsymbol{s} = \boldsymbol{e}H^t$. En considérant que, pour $1 \leqslant i \leqslant r$, les variables aléatoires $s_i = \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$, sont iid, alors*

$$|\boldsymbol{s}| \sim \mathcal{B}\left(r, \mathbb{P}[s_i = 1]\right)$$

Démonstration. Il suffit de remarquer que $|\boldsymbol{s}| = \sum_{i=1}^{r} s_i$, où, pour $1 \leqslant i \leqslant r$, les s_i sont par hypothèse des variables aléatoires iid telles que $s_i \sim \text{Ber}(\mathbb{P}[s_i = 1])$. □

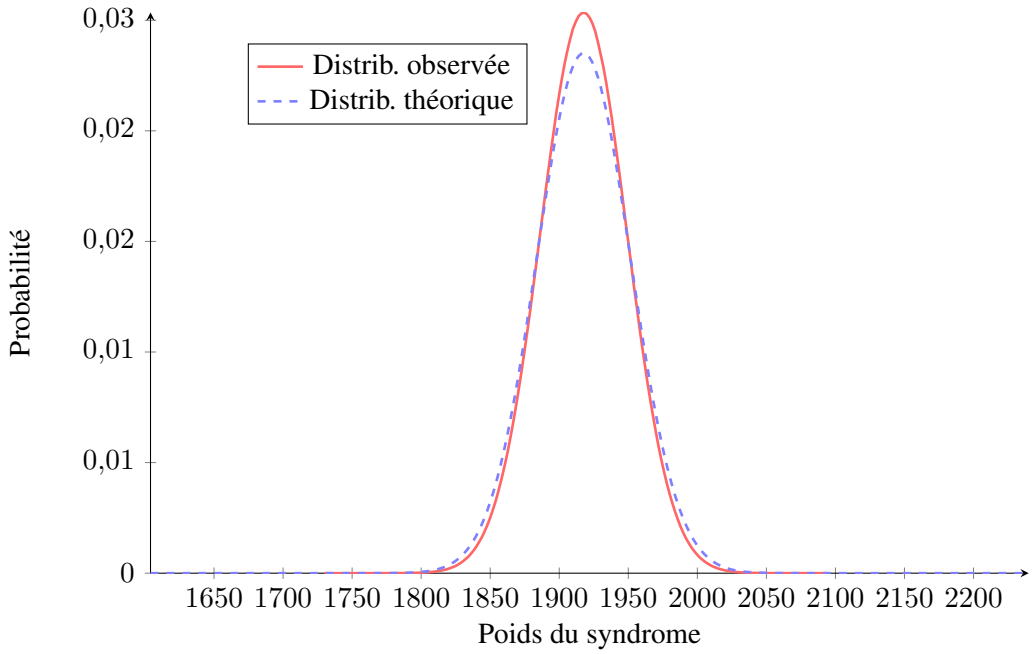


FIGURE 6.6 – Comparaison de la distribution théorique et observée du poids du syndrome à la première itération

	Valeurs théoriques	Valeurs observées
$\mathbb{E}[S_0]$	1917.681	1917.679
$\mathbb{V}[S_0]$	1151.695	990.424

TABLE 6.4 – Comparaison de l'espérance et de la variance du poids du syndrome observé et théorique au début de la première itération

La figure 6.6 et le tableau 6.4 indiquent que le comportement de la distribution du syndrome observée est assez bien estimé par le modèle théorique adopté. Malgré tout, nous pouvons remarquer que l'estimation de la variance de cette variable aléatoire n'est pas assez précise.

6.1.6 Calcul de la variance

Comme nous l'avons remarqué précédemment, la valeur de la variance du syndrome n'est pas la même pour les résultats théoriques et observés. Nous allons donc essayer de comprendre pourquoi notre modèle théorique diffère de ce que nous observons en pratique.

Rappelons que lors du calcul de la distribution du poids du syndrome à la première itération, nous avons supposé que les positions du syndrome étaient indépendantes et identiquement distribuées. Ceci reste une approximation du modèle réel. Nous allons donc essayer d'estimer la corrélation entre les différentes positions du syndrome afin de mieux estimer la variance de cette variable aléatoire.

Comme précédemment, nous considérons $H \in \mathcal{H}(n, r, w)$ une matrice de parité fixée. Nous

supposons de plus que $\boldsymbol{e} \sim \mathcal{U}_t$ et nous noterons $\boldsymbol{s} = \boldsymbol{e}H^t$.

En considérant que les positions du syndrome sont corrélées, nous allons recourir à la définition équivalente de la variance suivante :

$$\mathbb{V}[|\boldsymbol{s}|] = \mathbb{V}[\sum_{i=1}^{r} s_i] = \sum_{i=1}^{r} \mathbb{V}[s_i] + \sum_{i_1 \neq i_2} \mathrm{Cov}[s_{i_1}, s_{i_2}]$$

et

$$\mathrm{Cov}[s_{i_1}, s_{i_2}] = \mathbb{E}[s_{i_1} s_{i_2}] - \mathbb{E}[s_{i_1}]\mathbb{E}[s_{i_2}]$$

$\mathbb{E}[s_i]$ étant connu pour $1 \leqslant i \leqslant r$, il nous reste à calculer $\mathbb{E}[s_{i_1} s_{i_2}] = \mathbb{P}[\{s_{i_1} = 1\} \cap \{s_{i_2} = 1\}]$ avec $i_1 \neq i_2$.

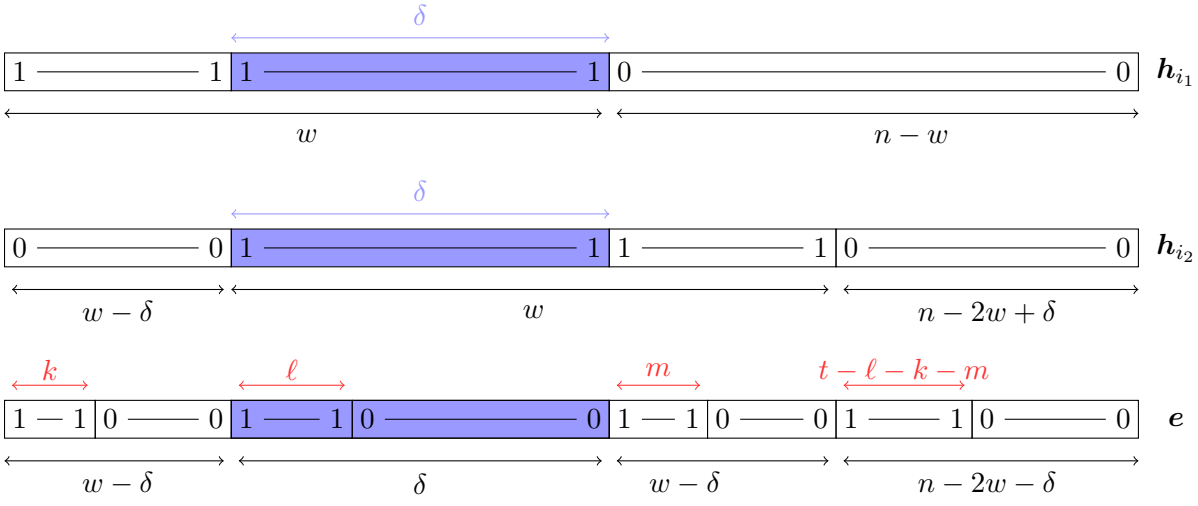
Propriété 30. *Soient* $(i_1, i_2) \in \{1, \ldots, r\}^2$ *et* $s_{i_1} = \langle \boldsymbol{h}_{i_1}, \boldsymbol{e} \rangle, s_{i_2} = \langle \boldsymbol{h}_{i_2}, \boldsymbol{e} \rangle$ *deux positions distinctes du syndrome. Posons* $\delta \stackrel{def}{=} |\boldsymbol{h}_{i_1} \cap \boldsymbol{h}_{i_2}|$.

Alors,

$$\mathbb{E}[s_{i_1} s_{i_2}] = \frac{1}{\binom{n}{t}} \sum_{\substack{\ell=0 \\ \ell \text{ impair}}}^{\delta} \sum_{\substack{m=0 \\ m \text{ pair}}}^{\min(t-\ell, w-\delta)} \sum_{\substack{k=0 \\ k \text{ pair}}}^{\min(t-\ell-k, w-\delta)} \binom{\delta}{\ell} \binom{w-\delta}{k} \binom{w-\delta}{m} \binom{n-2w+\delta}{t-\ell-k-m}$$
$$+ \frac{1}{\binom{n}{t}} \sum_{\substack{\ell=0 \\ \ell \text{ pair}}}^{\delta} \sum_{\substack{m=0 \\ m \text{ impair}}}^{\min(t-\ell, w-\delta)} \sum_{\substack{k=0 \\ k \text{ impair}}}^{\min(t-\ell-k, w-\delta)} \binom{\delta}{\ell} \binom{w-\delta}{k} \binom{w-\delta}{m} \binom{n-2w+\delta}{t-\ell-k-m}$$

Démonstration. Rappelons que δ est ici fixé, donc trouver $\mathbb{P}_{\boldsymbol{e},\delta}[\{s_{i_1} = 1\} \cap \{s_{i_2} = 1\}]$ revient à dénombrer les $\boldsymbol{e} \sim \mathcal{U}_t$ tels que $\langle \boldsymbol{h}_{i_1}, \boldsymbol{e} \rangle = \langle \boldsymbol{h}_{i_2}, \boldsymbol{e} \rangle = 1$.

Pour qu'une erreur ne satisfasse ni $\boldsymbol{h}_{i_1}$ ni $\boldsymbol{h}_{i_2}$ il faut qu'elle soit de la configuration suivante.



On a

$$\begin{cases} \langle \boldsymbol{h}_{i_1}, \boldsymbol{e} \rangle &= (k + \ell) \mod 2 \\ \langle \boldsymbol{h}_{i_2}, \boldsymbol{e} \rangle &= (\ell + m) \mod 2 \end{cases}$$

Donc pour ℓ fixé, on a :

- si ℓ est pair,

$$\begin{cases} \langle \boldsymbol{h}_{i_1}, \boldsymbol{e} \rangle &= 1 \\ \langle \boldsymbol{h}_{i_2}, \boldsymbol{e} \rangle &= 1 \end{cases} \Leftrightarrow \begin{cases} k+\ell \text{ impair} &\Rightarrow k \text{ impair} \\ \ell+m \text{ impair} &\Rightarrow m \text{ impair} \end{cases}$$

- réciproquement, si ℓ est impair, il faut prendre k et m pairs.

□

Nous pouvons donc, lorsque la matrice de parité $H \in \mathcal{H}(n, r, w)$ est fixée, estimer la corrélation entre les différentes positions du syndrome en exploitant la taille des intersections des équations de parité auxquelles elles correspondent.

6.1.7 Conclusion

Cette section fournit les bases de l'analyse théorique de l'algorithme de décodage *bit flipping*. En effet, nous avons explicité le lien entre le choix du seuil et l'efficacité du décodage. Bien que certains des modèles théoriques ne semblent pas correspondre parfaitement à ce que nous pouvons observer en pratique, ils fournissent malgré tout les bases nécessaires à la compréhension du comportement de l'algorithme. Cependant, les résultats présentés ne permettent pas d'améliorer de façon significative le calcul du seuil. En effet, d'une part, nous pouvons estimer le meilleur seuil (pour ces modèles) à la première et la deuxième itération, mais, au delà, nous sommes contraints de recourir à une approche heuristique. D'autre part, la probabilité d'échec au décodage après un petit nombre d'itérations, lorsque ces seuils sont utilisés, n'est pas négligeable et ne peut être estimée théoriquement.

Suite à ces constations, notre démarche fut d'appliquer une approche heuristique, afin de déterminer une procédure pour le calcul du seuil de sorte que le décodage ait une probabilité d'échec négligeable. En d'autres termes, nous nous sommes concentrés sur l'amélioration du comportement des instances dont le décodage est plus lent, ou qui échoue, que nous appellerons par la suite pire cas pour le décodage.

6.2 Pires cas pour le décodage

Les résultats de la section précédente semblent indiquer qu'il reste difficile de fournir une estimation théorique de la distribution du poids de l'erreur au cours de l'algorithme de *bit flipping*. Puisque nous nous plaçons dans un contexte cryptographique, nous devons envisager qu'un échec au décodage puisse être exploité et mener à une attaque (voir [GJS16]). En ce sens, nous souhaiterions nous assurer que tous les motifs d'erreurs, sauf un nombre négligeable (de l'ordre de la sécurité du cryptosystème) puissent être décodés.

La problématique que nous considérons est donc l'amélioration du comportement de l'algorithme pour des instances (*i.e.* des erreurs) dont le décodage échoue ou qui nécessitent beaucoup d'itérations pour être décodées. La première modification que nous pouvons effectuer sur l'algorithme est de changer le choix des valeurs des seuils. En effet, nous avons déjà pu constater que l'estimation des meilleurs seuils dépend du modèle théorique de l'erreur et du comportement des variables aléatoires impliquées. Bien que le modèle de l'erreur soit assez bien compris, il reste quelques approximations dans la modélisation des différentes variables aléatoires considérées. De plus, sans estimation fidèle de la distribution du poids de l'erreur à la fin de chaque itération, il nous est impossible de garantir que la probabilité d'échec au décodage est négligeable. De la même façon, nous ne pourrions pas déterminer les valeurs des seuils qui permettraient d'améliorer le comportement de l'algorithme pour les pires cas.

Définition 32 (Pire cas pour le décodage)**.** *Les pires cas pour le décodage sont, à matrice de parité fixée, les erreurs qui se décodent en un grand nombre d'itérations (très supérieur à la moyenne) ou dont le décodage diverge.*

Nous avons donc décidé de recourir à une approche heuristique afin d'améliorer le comportement du décodage dans les pires cas. Entres autres, cela nous a aussi permis de mieux comprendre le comportement de l'algorithme et plus particulièrement pour les pires cas.

La première question à laquelle nous nous sommes intéressés est de savoir si nous pouvions exploiter les informations disponibles lors du décodage, pour prédire, dès les premières itérations, qu'une instance donnerait un pire cas. Dans le contexte considéré, les informations disponibles sont celles qui sont accessibles lorsque le décodage est effectué lors du déchiffrement. L'erreur, et plus particulièrement son support, ne sont donc pas connus. En effet, rappelons que le chiffré reçu correspond à $\boldsymbol{c} = \boldsymbol{m}G + \boldsymbol{e}$ où

— $\boldsymbol{e} \in \mathbb{F}_2^n$ est choisi aléatoirement parmi l'ensemble des vecteurs de poids t et de longueur n lors du chiffrement,

— $G \in \mathbb{F}_2^{k \times n}$ est la clé publique,

— $\boldsymbol{m} \in \mathbb{F}_2^k$ est le message clair.

Pour effectuer le déchiffrement, nous utilisons la clé privée $H \in \mathbb{F}_2^{(n-k)\times n}$, qui est une matrice de parité creuse du code MDPC engendré par la matrice G.

Définition 33 (Données monitorables et observables)**.** *Les données monitorables de l'algorithme de décodage sont toutes les données calculées, ou dont la valeur peut être déduite lors du décodage lorsque celui-ci s'effectue en boîte blanche (toutes les informations sont disponibles).*

Les données observables sont les données qui sont disponibles durant le décodage, pour celui qui effectue le décodage (l'erreur n'est pas connue).

Autrement dit, nous considérons, à chaque itération, les données observables suivantes :

(i) le syndrome, $\boldsymbol{s} = \boldsymbol{c}H^t$ à la première itération, puis qui est mis à jour à chaque itération,

(ii) le nombre d'équations de parité non satisfaites par chacune des positions de l'erreur, *i.e.* $\sigma_j = |\boldsymbol{s} \cap \boldsymbol{h}_{*,j}|$ pour $1 \leqslant j \leqslant n$,

(iii) le seuil b pour cette itération et qui nous permet de déduire f le nombre de postions telles que $\sigma_j \geqslant b$ (c'est-à-dire le nombre de positions modifiées au cours de l'itération).

Les données monitorables du décodage sont quant à elles :

(i) l'erreur $\boldsymbol{e}$ pour chaque itération,

(ii) puisque le support de $\boldsymbol{e}$ est connu, nous pouvons différencier les valeurs de σ_j selon que $j \in \boldsymbol{e}$ ou $j \notin \boldsymbol{e}$,

(iii) l'ensemble des données observables.

Nous avons donc effectué de nombreuses simulations de décodage, au cours desquelles certaines informations monitorables étaient conservées : le poids du syndrome S_u au début de la u-ième itération, le nombre de bits modifiés f_u pendant la u-ième itération et le poids de l'erreur t_u à la fin de l'itération u.

L'approche que nous avons adoptée est illustrée par les résultats et les détails des simulations que nous avons effectuées sur des codes $[9602, 4801, 90]$-QC-MDPC avec une erreur initiale de poids 84.

Remarque 34. *L'emploi des codes QC-MDPC permet de réduire la taille des clés pour les primitives cryptographiques qui les font intervenir. Cependant, le caractère quasi-cyclique ne modifie pas le comportement de ces codes lors du décodage.*

Nous avons choisi d'utiliser l'algorithme 2 avec au plus 10 itérations et les seuils fixes suivant $(29, 27, 25, 23, 21, 20, 19, 18, 17, 16)$. Les résultats présentés ont été déduits de décodage pour les codes MDPC, construits à partir 100 matrices de parité aléatoires et 10^5 erreurs aléatoires pour chacune de ces matrices, soit un total de 10^7 instances de décodages.

Au cours de ces décodages, nous avons trouvé 3 instances dont le décodage diverge. Nous noterons ces pires cas $\mathcal{W}_1$, $\mathcal{W}_2$ et $\mathcal{W}_3$ et nous allons donc comparer les valeurs des données monitorables de ces pires cas avec celles des cas considérés comme moyens.

Cas considéré	Itér. 1	Itér. 2	Itér. 3	Itér. 4	Itér. 5
$\mathcal{W}_1$	82	79	69	54	112
$\mathcal{W}_2$	84	78	69	47	43
$\mathcal{W}_3$	84	77	69	54	97
Esp. cas moyens	61.403	29.517	1.116	$0.645\ 10^{-3}$	$0.14\ 10^{-5}$

TABLE 6.5 – Comparaison de la moyenne du poids de l'erreur pour les cas moyens avec le poids de l'erreur après les 5 premières itérations pour les pires cas

Remarque 35. *Cet exemple est une illustration des résultats que nous avons obtenus après de nombreuses simulations. Ici, nous avons obtenu* 3 *pires cas, mais nous considérons, au vu des nombreuses autres simulations effectuées, que le comportement des ces pires cas correspond bien à celui des pires cas de manière générale. Nous prendrons donc les résultats de ces simulations comme base pour l'analyse des pires cas.*

Comme le montre la figure 6.7, nous avons constaté que le poids du syndrome des pires cas au début du décodage n'est pas distinguable de celui d'un décodage qui se déroule bien. En effet, nous observons que $S_0(\mathcal{W}_1) = 1932$, $S_0(\mathcal{W}_2) = 1910$ et $S_0(\mathcal{W}_3) = 1872$ tandis que les instances typiques ont un poids de syndrome moyen égal à $\mathbb{E}[S_0] = 1917.676$ au début de la première itération, tandis que la variance observée du poids du syndrome à la première itération est $\mathbb{V}[S_0] = 990.424$.

Dans la suite des itérations, le poids du syndrome pour les pires cas est plus élevé que la moyenne comme le montre la figure 6.8. Cela est principalement dû au fait que, dans les pires cas, le poids de l'erreur ne diminue pas beaucoup au cours d'une itération.

De plus, le mauvais comportement d'un décodage s'explique bien par le nombre de positions modifiées au cours des itérations. En effet, nous avons pu remarquer qu'au cours des premières itérations du décodage d'un pire cas, cette valeur est plutôt faible et ceci contribue au fait que le poids de l'erreur diminue très peu (voir les figures 6.9 et 6.10).

Nous avons aussi comparé les résultats des décodages des cas moyens au décodage de nombreuses erreurs par une matrice de parité $H_{\mathcal{W}_0}$ ayant coïncidé avec le pire cas $\mathcal{W}_0$. Nous comparons les résultats obtenus précédemment pour les cas moyens, avec les résultats du décodage de 10^7 erreurs aléatoires en utilisant uniquement la matrice de parité $H_{\mathcal{W}_0}$.

De même, nous avons effectué 30 10^3 instances de décodage avec des matrices de parité aléatoires ainsi que : d'une part, une erreur fixée, considérée comme un cas moyen et d'autre part, le pire cas $\mathcal{W}_0$.

Au vu de ces résultats, il apparaît que c'est bien le couple formé de cette matrice et de l'erreur qui produit un pire cas, puisque les instances de décodage utilisant cette même matrice ont un comportement équivalent à celui d'une instance moyenne, comme le montrent les tables 6.6 et 6.8. De même, une erreur ayant produit un pire cas lors d'une instance de décodage avec une matrice de parité fixée, ne produira pas forcément de pire cas pour d'autres instances, cf. table 6.9 et 6.7. Ainsi, il semble difficile de sélectionner de "bonnes" matrices de parité pour le décodage, ou de ne pas considérer certains motifs d'erreur. Dans la suite, et pour simplifier l'analyse, nous considérerons un pire cas comme étant l'erreur utilisée pour l'instance de décodage qui nécessite de nombreuses itérations ou qui diverge.

Ces résultats semblent indiquer qu'afin de favoriser le décodage des pires cas, le calcul des seuils devrait prendre en compte les paramètres de l'instance considérée. Autrement dit, il serait préférable que le choix du seuil varie en fonction des instances et des informations dont nous disposons sur celles-ci, plutôt que du nombre d'itérations effectuées. Dans notre contexte, le poids de l'erreur et son support ne sont pas connus lorsque nous effectuons le décodage. Par contre, le poids du syndrome et le nombre de positions modifiées sont des valeurs disponibles *i.e.* des données observables. Notre démarche a donc été de chercher des règles, qui dépendent du poids du syndrome, pour calculer le seuil, non pas en fonction de l'itération, mais de l'instance considérée (et plus particulièrement de la valeur du poids du syndrome).

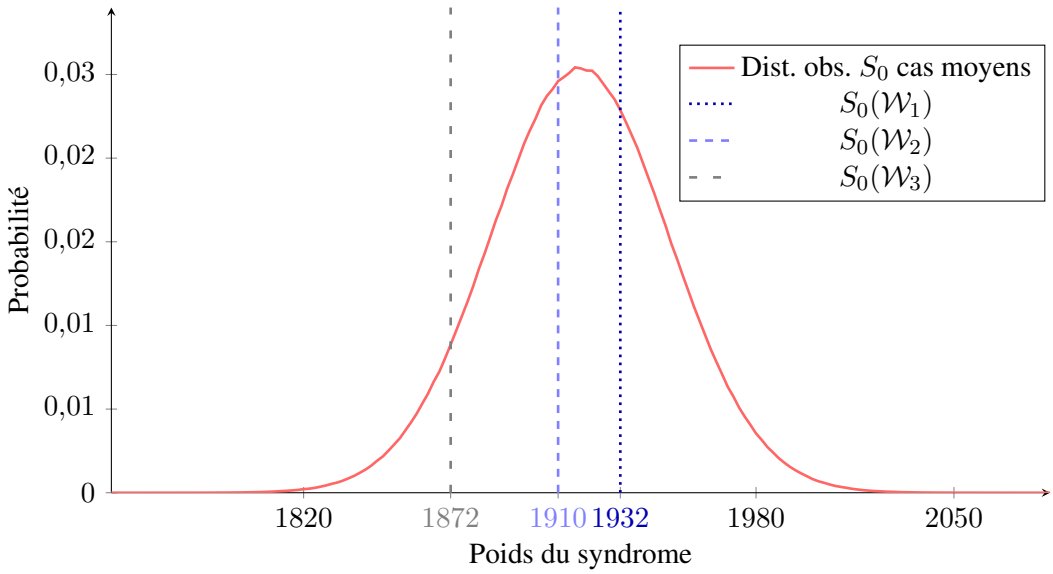


FIGURE 6.7 – Distribution du poids du syndrome et poids du syndrome des pires cas observés au début de la première itération

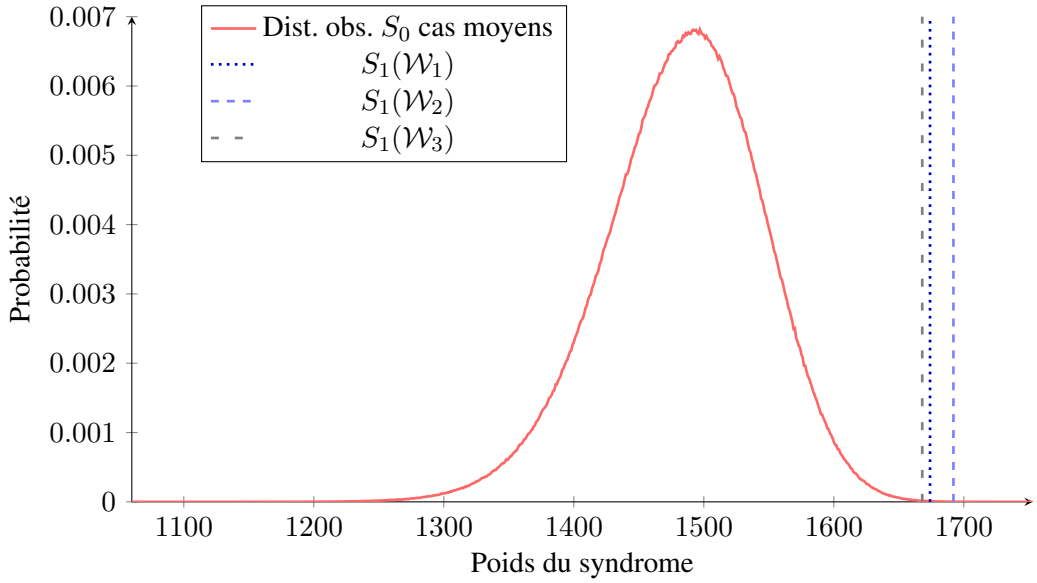


FIGURE 6.8 – Distribution du poids du syndrome et poids du syndrome des pires cas observés au début de la seconde itération

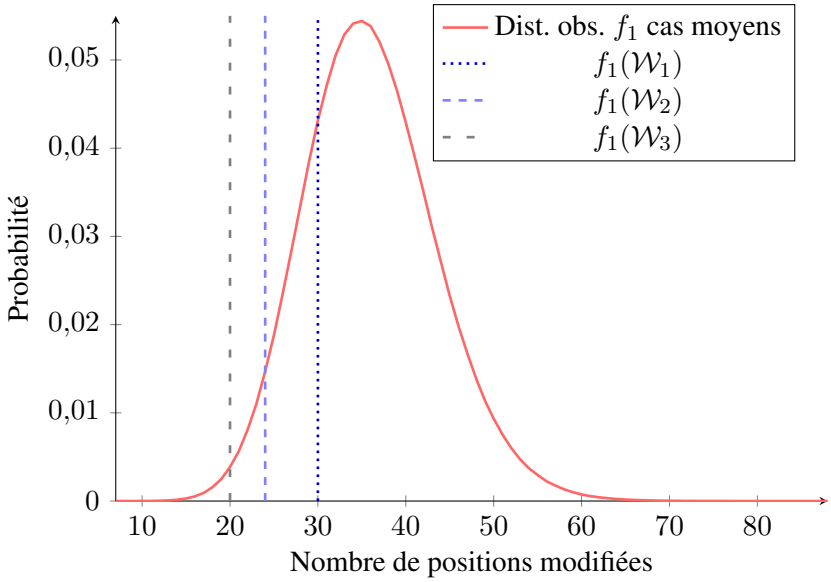


FIGURE 6.9 – Distribution du nombre moyen de positions modifiées et nombre de positions modifiées pour les pires cas au cours de la première itération

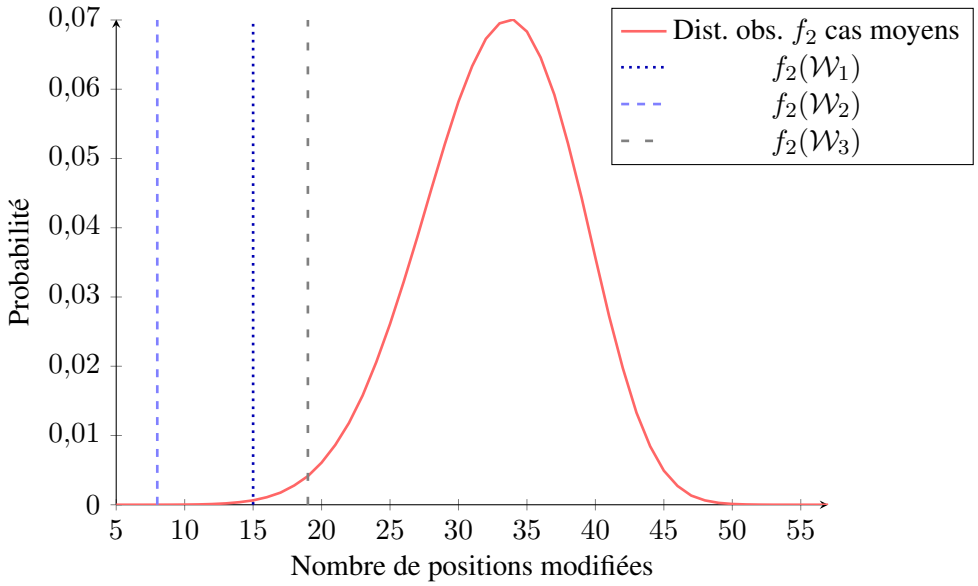


FIGURE 6.10 – Distribution du nombre de positions modifiées moyen et nombre de positions modifiées pour les pires cas au cours de la deuxième itération

Itération	Cas moyen	Matrice $H_{\mathcal{W}_0}$
2	0.999	0.999
3	0.306	0.292
4	$0.241\ 10^{-3}$	$0.304\ 10^{-3}$
5	$0.900\ 10^{-6}$	$0.900\ 10^{-5}$
6	$0.300\ 10^{-6}$	$0.100\ 10^{-6}$
$\leqslant 7$	$0.300\ 10^{-6}$	$0.100\ 10^{-6}$

TABLE 6.6 – Comparaison de la probabilité d'échec au décodage après un nombre d'itérations fixé pour les cas moyens et la matrice de parité $H_{\mathcal{W}_0}$

Itération	Cas moyen	$\mathcal{W}_0$
2	1	0.999
3	0.297	0.302
4	$0.236\ 10^{-3}$	$0.245\ 10^{-3}$
5	$0.337\ 10^{-4}$	0
$\leqslant 6$	0	0

TABLE 6.7 – Comparaison de la probabilité d'échec au décodage après un nombre d'itérations fixé pour une erreur fixe "moyenne" et le pire cas $\mathcal{W}_0$

	S_0		S_1		t_1		f_1	
Cas étudié	$\mathbb{E}[S_0]$	$\mathbb{V}[S_0]$	$\mathbb{E}[S_1]$	$\mathbb{V}[S_1]$	$\mathbb{E}[t_1]$	$\mathbb{V}[t_1]$	$\mathbb{E}[f_1]$	$\mathbb{V}[f_1]$
Cas moyen	1917.68	990.30	1483.70	3571.46	61.40	23.96	36.02	55.37
$H_{\mathcal{W}_0}$	1917.68	986.38	1483.62	3579.77	61.34	23.95	35.92	54.97

TABLE 6.8 – Comparaison de l'espérance et de la variance de différentes données monitorables des cas moyens et des simulations avec $H_{\mathcal{W}_0}$

	S_0		S_1		t_1		f_1	
Cas étudié	$\mathbb{E}[S_0]$	$\mathbb{V}[S_0]$	$\mathbb{E}[S_1]$	$\mathbb{V}[S_1]$	$\mathbb{E}[t_1]$	$\mathbb{V}[t_1]$	$\mathbb{E}[f_1]$	$\mathbb{V}[f_1]$
Erreur "moyenne"	1917.68	991.23	1483.39	3550.10	61.37	23.73	36.10	55.66
$\mathcal{W}_0$	1917.48	959.10	1484.00	3569.53	61.35	35.05	35.99	54.99

TABLE 6.9 – Comparaison de l'espérance et de la variance de différentes données monitorables pour des erreurs "moyennes" et $\mathcal{W}_0$

Ce choix s'appuie sur la constatation suivante : si le poids du syndrome ne diminue pas beaucoup au cours des itérations, la probabilité qu'il en soit de même pour le poids de l'erreur est assez grande. Le choix d'un seuil en fonction du poids du syndrome nous permettrait de corriger ce comportement, en favorisant sa diminution.

6.3 Calcul adaptatif du seuil

Le but de ce processus est de trouver le seuil, défini comme une fonction du poids du syndrome, qui permet de minimiser le nombre de pires cas et le nombre maximal d'itérations du décodage, tout en s'assurant que sa probabilité d'échec reste faible après de nombreuses simulations.

Nous avons décidé d'adopter une démarche heuristique afin de déterminer, à l'aide de données monitorables sur de nombreuses instances de décodage, l'ensemble des pires cas et comment leur décodage pouvait être amélioré.

6.3.1 Détails du processus

La première étape consiste donc à choisir une fonction de seuil initiale et à exécuter de nombreux décodages grâce à l'algorithme 2 avec des erreurs et des matrices aléatoires, au cours desquels une partie des informations monitorables est conservée. Une information primordiale est le nombre d'itérations nécessaires pour décoder, notée I, sachant que le nombre maximal d'itérations de l'algorithme est fixé à I_{max}. Ensuite, nous aurons besoin de conserver le poids du syndrome à chaque itération, noté $\boldsymbol{S} = (S_u)_{0 \leqslant u \leqslant I}$ avec $S_0 = \mathrm{wt}(\boldsymbol{e}H^t)$ le poids du syndrome de l'erreur initiale. Nous collecterons aussi $\boldsymbol{f} = (f_u)_{1 \leqslant u \leqslant I}$, où f_u correspond au nombre de bits modifiés au cours de la u-ième itération. Enfin, le poids de l'erreur obtenue à la fin de chaque itération est aussi sauvegardé de sorte que $\boldsymbol{t} = (t_u)_{1 \leqslant u \leqslant I}$ où t_u est le poids de l'erreur à la fin de l'itération u (par convention, $t_0 = t$ désignera le poids initial de l'erreur). L'ensemble de ces données est associé à l'instance correspondante, *i.e.* à la fonction de seuil b et à l'erreur $\boldsymbol{e}$ considérées. Ainsi, les données monitorables conservées, pour chacune des instances effectuées avec un seuil égal à b, seront notées $\mathcal{M}_b$ avec :

$$\mathcal{M}_b \ : \boldsymbol{e} \longrightarrow (I, \boldsymbol{S}, \boldsymbol{f}, \boldsymbol{t})$$

où $\boldsymbol{e}$,I, $\boldsymbol{S}$, $\boldsymbol{f}$ et $\boldsymbol{t}$ sont définis comme précédemment.

Ensuite, nous analyserons les données fournies par ces nombreuses instances de décodage. Comme nous souhaitons optimiser le calcul du seuil pour les pires cas, nous aurons, en premier lieu, besoin de définir ces pires cas afin de les détecter. Pour une fonction b fixée, nous définissons $I_{\mathcal{F}}$ le nombre d'itérations du décodage à partir duquel l'instance est considérée comme un pire cas. Dans un premier temps, nous fixerons par exemple $I_{\mathcal{F}} = I_{max}$. L'ensemble des pires cas sera donc défini par :

$$\mathcal{F}_{b,I_{\mathcal{F}}} \stackrel{\text{def}}{=} \{(\boldsymbol{e}, I) \ : \forall u < I, t_u \neq 0 \text{ et } t_I = 0 \text{ avec } I \geqslant I_{\mathcal{F}}\}$$

Enfin, l'analyse des données monitorables collectées à partir de nombreuses simulations nous permettra d'ajuster la valeur du seuil en fonction des résultats obtenus. Une solution possible, par exemple, est de comparer le nombre de positions modifiées pendant une itération, à la différence

entre le poids de l'erreur au début et à la fin de l'itération. Nous pourrons ainsi déterminer le nombre de positions justes et fausses modifiées afin d'adapter la valeur du seuil qui correspond au poids de syndrome observé.

Remarque 36. *Le but de cette étape n'est pas d'optimiser le décodage itération après itération, mais bien de diminuer le nombre total d'itérations nécessaires pour décoder l'instance. Dans certains cas, l'analyse du nombre de positions modifiées et du nombre d'erreurs corrigées indique que le seuil devrait être modifié mais la meilleure décision reste encore de conserver le seuil actuel. De manière générale, il est préférable de modifier le seuil seulement dans le cas où les résultats sont très explicites.*

Ensuite, ces étapes sont réitérées, jusqu'à ce que pour un $I_{\mathcal{F}}$ raisonnable, nous ayons trouvé b tel que $\mathcal{F}_{b,I_{\mathcal{F}}} = \emptyset$.

Remarque 37. *La fonction de seuil que nous qualifierons d'optimale est celle qui remplit les conditions mentionnées ci-dessus et qui est telle que, pour tout changement de la fonction b, les résultats obtenus soient moins bons,* i.e. *$I_{\mathcal{F}}$ augmente ou $\mathcal{F}_{b,I_{\mathcal{F}}} \neq \emptyset$.*

Les différentes étapes de ce processus sont donc :

(i) la mise en œuvre du décodage avec sauvegarde des données monitorables,

(ii) l'analyse de ces données,

(iii) l'ajustement du seuil.

Nous décrivons ci-après ces 3 étapes en détail.

Décodage avec sauvegarde des données monitorables

La première étape consiste à effectuer de nombreux décodages en utilisant des erreurs aléatoires $\boldsymbol{e} \in \mathbb{F}_2^n$ de poids t, au cours desquels les données monitorables $\mathcal{M}_b(\boldsymbol{e}) \stackrel{\text{def}}{=} \{(I, \boldsymbol{f}, \boldsymbol{t}, \boldsymbol{S})\}$ sont conservées (et associées à l'erreur $\boldsymbol{e}$ considérée). Ces décodages s'effectuent dans un premier temps avec un nombre d'itérations I_{max} assez grand (typiquement 10), de sorte que seules les instances divergentes n'aboutissent pas. Au début du processus, la fonction de seuil b correspondra aux seuils précalculés, pour chaque itération, grâce aux formules de Gallager (voir le chapitre 5).

Au cours de ce processus, cette étape est réitérée à chaque fois que la fonction b est modifiée.

Analyse des données monitorables

L'analyse des données monitorables $\mathcal{M}_b$ acquises grâce à la première étape permet de déterminer d'une part, le nombre d'itérations $I_{\mathcal{F}}$ à partir duquel une instance est considérée comme un pire cas et d'autre part, l'ensemble des pires cas $\mathcal{F}_{b,I_{\mathcal{F}}}$ pour la fonction de seuil b et cette valeur de $I_{\mathcal{F}}$.

Pour fixer la valeur de $I_{\mathcal{F}}$, nous pouvons, par exemple, utiliser le maximum du nombre d'itérations rencontré à quelques unités près *i.e.* $I_{\mathcal{F}} \leftarrow \max_{I \in \mathcal{M}_b} I - \delta$. L'ajustement de la valeur de δ permet, lorsque plusieurs itérations de ce processus ont été effectuées, d'affiner notre définition des pires cas.

Plus le nombre de pires cas diminue, plus il est intéressant d'augmenter la valeur de δ de sorte que le nombre maximal d'itérations de l'algorithme diminue.

Durant cette étape, nous allons distinguer les valeurs monitorables des instances considérées comme des pires cas. Étant donnée l'ensemble des $\mathcal{M}_b$, nous allons donc chercher les erreurs $\boldsymbol{e}$ qui vérifient $\mathcal{M}_b(\boldsymbol{e}) = (I, \boldsymbol{f}, \boldsymbol{t}, \boldsymbol{S})$ avec $I \geqslant I_{\mathcal{F}}$. Lorsqu'une telle erreur est rencontrée, elle est ajoutée à l'ensemble des pires cas $\mathcal{F}_{b,I_{\mathcal{F}}}$ déjà rencontrés pour cette fonction de seuil et cette valeur de $I_{\mathcal{F}}$.

Si l'ensemble $\mathcal{F}_{b,I_{\mathcal{F}}}$ est vide et que la valeur de $I_{\mathcal{F}}$ paraît raisonnable, le processus est terminé : la fonction b dont nous disposons permet de limiter le nombre de pire cas et le nombre maximal d'itérations est assez faible. Dans ce cas, les premières étapes sont réitérées, tout en conservant un plus grand nombre de données monitorables, afin de s'assurer que la probabilité d'échec au décodage demeure assez faible.

Dans le cas contraire, $\mathcal{F}_{b,I_{\mathcal{F}}} \neq \emptyset$, nous trouverons une fonction de seuil plus adaptée par l'analyse des données monitorables.

Ajustement du seuil

L'ajustement du seuil est l'étape clé de ce processus, puisqu'il permet de déterminer une fonction de seuil plus appropriée pour les paramètres donnés. En effet, cette étape a pour but de diminuer le nombre de pires cas et le nombre maximal d'itérations nécessaires au décodage.

Dans un premier temps, la fonction de seuil est précalculée, en fonction des itérations, grâce aux formules de Gallager, le but étant d'adopter, par la suite, un seuil qui dépend uniquement du poids du syndrome de l'instance et de l'itération considérée. Comme il faut, dans un premier temps, initialiser la fonction de seuil en fonction du poids du syndrome, nous avons associé la valeur des seuils précalculés aux moyennes des poids de syndrome pour chaque itération, grâce à une fonction dite "en escalier". Cette solution à l'avantage de ne pas détériorer le décodage des instances moyennes.

Une fois qu'un ensemble de pires cas pour cette fonction de seuil est déterminé, nous adaptons la valeur de cette fonction pour les poids de syndrome de cet ensemble d'éléments. Nous devons signaler que cet ajustement est fait de façon heuristique. Il ne garantit donc pas que la fonction ainsi produite soit optimale. Une solution possible est, par exemple, d'utiliser l'algorithme 3.

Une fois cette étape terminée, nous suggérons de réitérer l'étape de sauvegarde de données monitorables pour de nombreuses instances. Et ce jusqu'à ce que l'analyse de ces données indique que la fonction de seuil ainsi modifiée permet un décodage optimal.

Algorithme 3 Exemple de test pour l'ajustement du seuil

Entrées : Pour $\boldsymbol{e} \in \mathcal{F}_{b,I_{\mathcal{F}}}$, on utilise $\mathcal{M}_b(\boldsymbol{e}) = (I, \boldsymbol{f}, \boldsymbol{t}, \boldsymbol{S})$ et b la valeur du seuil en fonction du poids du syndrome

Sortie : La fonction b mise à jour

function AJUSTEMENT_SEUIL($I, \boldsymbol{f}, \boldsymbol{t}, \boldsymbol{S}, b$)
  for $1 \leqslant u \leqslant I$ **do**
    if $f_u > t_{u-1} - t_u$ **then**
      $b(S_{u-1}) \leftarrow b(S_{u-1}) + 1$
    end if
    if $f_u = t_{u-1} - t_u$ **then**
      $b(S_{u-1}) \leftarrow b(S_{u-1}) - 1$
    end if
  end for
end function

6.3.2 Résultats et conclusion

Nous avons appliqué le procédé décrit dans la sous-section 6.3.1 afin d'améliorer le décodage des codes $[9602, 4801, 90]$-QC-MDPC avec un poids d'erreur initial $t = 84$.

Premièrement, nous explicitons la fonction de seuil obtenue par la figure 6.11.

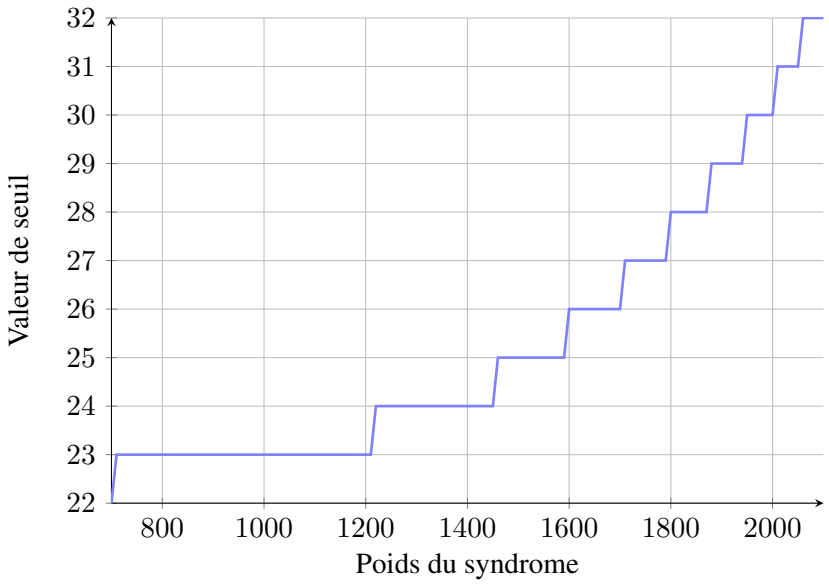


FIGURE 6.11 – Fonction de seuil trouvée grâce au processus heuristique d'amélioration du seuil, pour les codes $[9602, 4801, 90]$-QC-MDPC et un poids d'erreur initial de 84

Deuxièmement, la table 6.10 compare la probabilité d'échec au décodage après un nombre d'itérations fixé lorsque les seuils ont été :

(i) précalculés grâce à l'approche introduite dans la section 5.2, c'est-à-dire en utilisant les résultats de [Gal63],

(ii) définis comme une fonction du poids du syndrome, obtenue par le procédé heuristique décrit dans la sous-section 6.3.1.

Itération	[Gal63]	Seuil adaptatif
2	0.999	0.999
3	0.306	0.231
4	$0.241\ 10^{-3}$	$0.411\ 10^{-3}$
5	$0.900\ 10^{-6}$	$0.336\ 10^{-5}$
6	$0.300\ 10^{-6}$	$0.140\ 10^{-6}$
$\leqslant 7$	$0.300\ 10^{-6}$	0

TABLE 6.10 – Probabilité d'échec au décodage après un nombre fixé d'itérations

Remarque 38. *Nous comparons les résultats des simulations effectuées avec* 100 *matrices et* 10^5 *erreurs aléatoires par matrice, lorsque le seuil est précalculé grâce aux formules de [Gal63], c'est-à-*

dire lorsque des seuils fixes sont utilisés d'une part et que la fonction de seuil considérée est celle explicitée par la figure 6.11 d'autre part. De ce fait, une probabilité d'échec nulle signifie que nous n'avons pas rencontré ce cas lors de nos simulations et ne constitue pas une borne théorique.

En conclusion, ces résultats semblent indiquer que le procédé heuristique détaillé dans la sous-section 6.3.1 permet effectivement de diminuer la probabilité d'échec au décodage pour les paramètres considérés. Bien que cette démarche ne permette pas de garantir que la probabilité d'échec au décodage est de l'ordre de la sécurité du cryptosystème (ici, 2^{-80}), nous pouvons néanmoins penser que l'emploi de seuils adaptatifs (*i.e.* définis comme une fonction du poids du syndrome) est une bonne piste pour améliorer, en pratique, la probabilité d'échec au décodage.

7 Analyse théorique

7.1 Objectifs

Dans ce chapitre, nous nous intéresserons à l'analyse de l'algorithme de décodage *bit flipping* pour les codes MDPC. Cet algorithme fut étudié en détail dans [Gal63], où Gallager donne une borne sur la probabilité d'échec au décodage après I itérations de l'algorithme lorsque celui-ci est appliqué au décodage des codes LDPC. L'analyse de ces résultats, explicitée dans le chapitre 5, nous a permis de comprendre les limites de ce modèle. En effet, dans notre contexte, l'algorithme de décodage est employé sur des erreurs de poids fixe t, et non sur une erreur provenant d'un canal binaire symétrique. Ceci explique en partie pourquoi les résultats expérimentaux introduits dans ce chapitre indiquent que le modèle ne coïncide pas avec nos observations.

Nous devons donc trouver d'autres outils pour analyser l'efficacité du décodage. Une première remarque est que le comportement de l'algorithme est complètement déterminé par les seuils utilisés au cours des itérations. Plus particulièrement, lorsque le poids de l'erreur avant l'itération est connu, c'est la proportion de positions fausses corrigées et celle des positions justes modifiées (par erreur) au cours de cette itération, qui déterminent le poids de l'erreur à la fin de celle-ci. Malheureusement, lorsque le décodage s'effectue au cours d'une étape de déchiffrement, seul le poids de l'erreur avant la première itération est connu. L'enjeu est donc de trouver les distributions de ces deux événements de la manière la plus fine possible, voire d'en fournir une approximation pour une instance donnée. Ce qui permettrait d'effectuer un décodage optimal pour chaque matrice de parité et vecteur d'erreur.

Le principal objectif de ce chapitre est donc de fournir une analyse théorique de l'algorithme de décodage 2. Cette analyse nous permettrait, d'une part, de confirmer que la fonction de seuil explicitée dans le chapitre 6 est un choix judicieux et d'autre part, de donner une estimation précise de la probabilité d'échec au décodage après un nombre fixé d'itérations.

Nous commencerons donc par définir une série de grandeurs, que nous appellerons "grandeurs fondamentales". Ce sont celles dont les valeurs permettent d'exprimer entièrement le fonctionnement du décodage. La section 7.2 fournira la définition de ces grandeurs et les différentes formules qui explicitent le lien entre ces grandeurs et le comportement de l'algorithme. Nous donnerons, dans la section 7.3, une estimation de la distribution de ces grandeurs à la première itération.

7.2 Grandeurs fondamentales de l'algorithme

Dans cette section, nous introduirons les grandeurs fondamentales de l'algorithme, c'est-à-dire les valeurs qui permettent de comprendre le déroulement du décodage. Nous considérerons l'algorithme de *bit flipping* comme introduit dans le chapitre 5.

Nous noterons $\mathcal{H}(n, r, w)$ l'ensemble des matrices de taille $r \times n$, à valeur dans $\mathbb{F}_2$ et dont le poids des lignes est égal à w. Nous supposerons de plus que ces matrices ont un poids de colonne fixe et égal à d. Pour $H \in \mathcal{H}(n, r, w)$, les équations de parité définies par H seront notées $\boldsymbol{h}_i$ où $1 \leqslant i \leqslant r$.

Rappelons que le contexte dans lequel nous nous plaçons est celui de l'emploi de l'algorithme de *bit flipping* dans le cryptosystème de McEliece lors de l'étape de déchiffrement. Dans ce cas, nous cherchons à décoder un mot de code bruité $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ avec $\boldsymbol{c} \in \mathcal{C}$ où $\mathcal{C}$ est le code MDPC défini par la matrice de parité $H \in \mathcal{H}(n, r, w)$ connue (puisqu'elle forme la clé privée) et $\boldsymbol{e} \in \mathbb{F}_2^n$ une erreur aléatoire de poids t dont nous cherchons le support. L'algorithme de *bit flipping* est un algorithme itératif et probabiliste mais dont la procédure est la même pour chaque itération. Nous supposerons donc, sans perte de généralité, qu'à chaque itération, le problème consiste à trouver $\boldsymbol{e}^{(u)} \in \mathbb{F}_2^n$ telle que $(\boldsymbol{y} - \boldsymbol{e}^{(u)})H^t = 0$ (*i.e.* $\boldsymbol{y} - \boldsymbol{e}^{(u)} \in \mathcal{C}$) où $\boldsymbol{y}$ est mis à jour au cours des itérations.

Notons $E_\ell^{(u)}$ le nombre d'équations de parité faisant intervenir ℓ positions de l'erreur à la u-ème itération, c'est-à-dire :

$$E_\ell^{(u)} \stackrel{\text{def}}{=} |\{i \ : \ 1 \leqslant i \leqslant r \text{ et } |\boldsymbol{h}_i \cap \boldsymbol{e}^{(u)}| = \ell\}|$$

Nous allons voir que la connaissance de la valeur des $E_\ell^{(u)}$ suffit à décrire le comportement de l'algorithme.

Remarque 39. *Les valeurs des $E_\ell^{(u)}$ ne sont pas observables en pratique, puisqu'elles dépendent du support de l'erreur, qui n'est pas connu.*

Dans la section 5.2, nous avons décrit l'analyse de ce décodeur comme introduit dans [Gal63]. Bien que le contexte soit différent, nous pouvons remarquer que le comportement de l'algorithme est toujours déterminé par la connaissance du nombre d'équations de parité insatisfaites par une position $j \in \{1, \dots, n\}$ (que nous appellerons compteurs de la j-ème position), selon que $j \in \boldsymbol{e}$ ou $j \notin \boldsymbol{e}$. Plus particulièrement, c'est la différence entre ces deux quantités, qui nous permettra d'analyser l'efficacité du décodage.

Dans un premier temps, nous allons voir que les valeurs des $E_\ell^{(u)}$ sont fixées par les différents paramètres de l'algorithme.

Propriété 31. *Pour tout u, $\forall H \in \mathcal{H}(n, r, w)$ et pour $\boldsymbol{e}^{(u)}$ l'erreur obtenue après la u-ème itération du décodage, on a*

$$\sum_{\ell \geqslant 0} E_\ell^{(u)} = r \tag{7.1}$$

Démonstration. En effet, pour $\boldsymbol{e}^{(u)}$ et $i \in \{1, \dots, r\}$ fixés, $|\boldsymbol{h}_i \cap \boldsymbol{e}^{(u)}|$ est unique. □

Pour ne pas surcharger les écritures et comme nous considérons que les résultats donnés sont vrais pour toute itération u, nous écrirons plus succinctement E_ℓ au lieu de $E_\ell^{(u)}$. Il en sera de même pour toutes les données de l'algorithme (l'erreur, le syndrome etc.).

Rappelons qu'un nombre impair de positions de l'erreur impliqué dans une équation de parité donne un bit du syndrome non nul. Ainsi, le poids du syndrome peut aussi être exprimé en fonction de la valeur des E_ℓ.

Propriété 32. *Soient $H \in \mathcal{H}(n,r,w)$, $\boldsymbol{e} \in \mathbb{F}_2^n$ et $S = |\boldsymbol{e}H^t|$.*

$$\sum_{\ell \text{ impair}} E_\ell = S$$

Démonstration. En effet, on a $S = |\boldsymbol{e}H^t| = \sum_{i=1}^{r} \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$. Or $\langle \boldsymbol{h}, \boldsymbol{e} \rangle = 1 \Leftrightarrow |\boldsymbol{h} \cap \boldsymbol{e}| = \ell$ avec ℓ impair.

D'où $S = |\{i \ : \ 1 \leqslant i \leqslant r \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1\}| = |\bigsqcup_{\ell \text{ impair}} \{i \ : \ 1 \leqslant i \leqslant r \text{ et } |\boldsymbol{h}_i \cap \boldsymbol{e}| = \ell\}|$ □

Maintenant, nous souhaitons décrire le comportement des compteurs en fonction de la valeur des E_ℓ. Dans un premier temps, nous allons voir que ces valeurs sont déterminées par le poids du syndrome de l'instance considérée.

Propriété 33. *Soient $H \in \mathcal{H}(n,r,w)$, $\sigma_j = |\{i \ : \ 1 \leqslant i \leqslant r \text{ et } j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1\}|$ le nombre d'équations de parité non vérifiées par la position j de l'erreur et $S = |\boldsymbol{e}H^t|$. On a*

$$\sum_{j=1}^{n} \sigma_j = w \cdot S$$

Démonstration. Soit, pour $1 \leqslant i \leqslant r$, $s_i = \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$, le i-ième bit du syndrome.

On a

$$\sum_{j=1}^{n} \sigma_j = \sum_{j=1}^{n} \sum_{i=1}^{r} s_i h_{i,j} = \sum_{i=1}^{r} s_i \sum_{j=1}^{n} h_{i,j}$$

Or, $\forall i \in \{1, \ldots, r\}$, $\sum_{j=1}^{n} h_{i,j} = w$, puisque $H \in \mathcal{H}(n,r,w)$.

D'où $\sum_{j=1}^{n} \sigma_j = \sum_{i=1}^{r} s_i \cdot w = Sw$. □

La propriété suivante nous permet de caractériser, grâce à la valeur des E_ℓ, la somme des compteurs des positions du support de l'erreur.

Propriété 34. *Soient $H \in \mathcal{H}(n,r,w)$ et $\sigma_j = |\{i \ : \ 1 \leqslant i \leqslant r, \ j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1\}|$. En considérant les positions du support de l'erreur nous obtenons :*

$$\sum_{j \in \boldsymbol{e}} \sigma_j = \sum_{\ell \text{ impair}} \ell \cdot E_\ell$$

Démonstration. Cette preuve se décompose en deux parties.

— Premièrement, on a :

$$\sum_{j\in \boldsymbol{e}} \sigma_j = \sum_{j\in \boldsymbol{e}} \sum_{i\ :j\in \boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle,$$

par définition de σ_j qui est le nombre d'équations de parité insatisfaites impliquant j.

— Deuxièmement, on a :

$$\begin{aligned}\sum_{\ell \text{ impair}} \ell \cdot E_\ell &= \sum_{\ell \text{ impair}} \ell \cdot |\{i, 1 \leqslant i \leqslant r \ : |\boldsymbol{h_i} \cap \boldsymbol{e}| = \ell\}| \\ &= \sum_{i=1}^{r} |\boldsymbol{e} \cap \boldsymbol{h}_i| \cdot \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle\end{aligned}$$

Rappelons que pour $i \in \{1, \ldots, r\}$, $\langle \boldsymbol{h}_i, \boldsymbol{e}\rangle$ est le produit scalaire de $\boldsymbol{h}_i$ et $\boldsymbol{e}$ dans $\mathbb{F}_2$, donc

$$\langle \boldsymbol{h}_i, \boldsymbol{e}\rangle = 1 \Leftrightarrow |\boldsymbol{h}_i \cap \boldsymbol{e}| = \ell \text{ avec } \ell \text{ impair.}$$

De plus, on a pour $i \in \{1, \ldots, r\}$, $|\boldsymbol{h}_i \cap \boldsymbol{e}| = \sum_{j=1}^{n} h_j \cdot e_j$ par définition.
Or $h_j \cdot e_j = 1 \Leftrightarrow h_j = 1$ et $e_j = 1 \Leftrightarrow j \in \boldsymbol{e}$ et $j \in \boldsymbol{h}_i$
D'où

$$\begin{aligned}\sum_{i=1}^{r} |\boldsymbol{h}_i \cap \boldsymbol{e}| \cdot \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle &= \sum_{i=1}^{r} (\sum_{j=1}^{n} h_j e_j) \cdot \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle \\ &= \sum_{i=1}^{r} (\sum_{j\in \boldsymbol{e}} h_{i,j}) \cdot \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle \\ &= \sum_{j\in \boldsymbol{e}} \sum_{i=1}^{r} h_{i,j} \cdot \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle \\ &= \sum_{j\in \boldsymbol{e}} \sum_{i\ :j\in \boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle \\ &= \sum_{j\in \boldsymbol{e}} \sigma_j\end{aligned}$$

□

À présent, nous avons les outils pour caractériser la somme des valeurs des compteurs des positions fausses et justes, en fonction du poids du syndrome et de la valeur des E_ℓ.

Propriété 35. *Soient $H \in \mathcal{H}(n, r, w)$, $\sigma_j = |\{i \ : \ 1 \leqslant i \leqslant r, \ j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle = 1\}|$ le nombre d'équations de parité non vérifiées par la position j de l'erreur et $S = |\boldsymbol{e}H^t|$.*

$$\sum_{j\in \boldsymbol{e}} \sigma_j = S + \sum_{\ell \geqslant 0} 2\ell \cdot E_{2\ell+1}$$

Démonstration. La propriété 34 nous permet de déduire

$$\sum_{j\in \boldsymbol{e}} \sigma_j = \sum_{\ell \text{ impair}} \ell E_\ell = \sum_{\ell \text{ impair}} E_\ell + \sum_{\ell \text{ impair}} (\ell - 1) E_\ell$$

D'autre part, la propriété 32 nous donne $\sum_{\ell \text{ impair}} E_\ell = S$. D'où le résultat.

□

Propriété 36. *Soient $H \in \mathcal{H}(n,r,w)$, $\sigma_j = |\{i \ : \ 1 \leqslant i \leqslant r, \ j \in \boldsymbol{h}_i \text{ et } \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle = 1\}|$ le nombre d'équations de parité non vérifiées par la position j de l'erreur et $S = |\boldsymbol{e}H^t|$.*

$$\sum_{j \notin \boldsymbol{e}} \sigma_j = (w-1) \cdot S - \sum_{\ell \geqslant 0} 2\ell \cdot E_{2\ell+1}$$

Démonstration. On a $\sum\limits_{j=1}^{n} \sigma_j = \sum\limits_{j \in \boldsymbol{e}} \sigma_j + \sum\limits_{j \notin \boldsymbol{e}} \sigma_j$.

D'où $\sum\limits_{j \notin \boldsymbol{e}} \sigma_j = \sum\limits_{j=1}^{n} \sigma_j - \sum\limits_{j \in \boldsymbol{e}} \sigma_j$. On conclut en utilisant les résultats des propriétés 33 et 36. □

7.3 Distributions à la première itération

Les grandeurs fondamentales que nous avons introduites dans la section précédente caractérisent entièrement le comportement de l'algorithme *bit flipping* pour une instance fixée. En d'autres termes, pour $\boldsymbol{e} \in \mathbb{F}_2^n$ telle que $|\boldsymbol{e}| = t$ et $H \in \mathcal{H}(n,r,w)$ fixés, la connaissance de la valeur des E_ℓ nous permet de prédire précisément le déroulement du décodage à chaque itération.

Cependant, dans notre contexte, le support de l'erreur n'est pas connu. Donc nous ne pouvons pas déterminer la valeur des E_ℓ. Malgré tout, nous savons que l'erreur au début de la première itération a un poids fixe, égal à t et nous pouvons exploiter cette information pour décrire de manière plus précise le comportement de l'algorithme pour une instance fixée. En particulier, nous avons pu constater que la plupart des grandeurs fondamentales décrites ci-dessus dépendent du poids du syndrome obtenu.

Dans la suite de cette section, nous allons donc considérer que les valeurs de $H \in \mathcal{H}(n,r,w)$ et $S \stackrel{\text{def}}{=} |\boldsymbol{e}H^t|$, avec $\boldsymbol{e} \sim \mathcal{U}_t$, sont fixées. Dans un premier temps, nous avons besoin de trouver une estimation de la valeur des E_ℓ. Comme précédemment, pour $j \in \{1,\dots,n\}$ fixé, nous noterons $\mathcal{H}_0$ l'hypothèse $j \notin \boldsymbol{e}$ et $\mathcal{H}_1$ l'hypothèse $j \in \boldsymbol{e}$.

Propriété 37. *Soient $H \in \mathcal{H}(n,r,w)$, $S = |\boldsymbol{e}H^t|$ où $\boldsymbol{e} \sim \mathcal{U}_t$ et $\boldsymbol{h}_i$ pour $1 \leqslant i \leqslant r$ désignant les équations de parité définies par H. Nous considérons $E_\ell = |\{i \ : \ 1 \leqslant i \leqslant r \text{ et } |\boldsymbol{h}_i \cap \boldsymbol{e}| = \ell\}|$.*

$$\mathbb{E}[E_\ell] = r \, \frac{\binom{w}{\ell}\binom{n-w}{t-\ell}}{\binom{n}{t}}$$

Démonstration. On a, pour $i \in \{1,\dots,r\}$, $\mathbb{P}[|\boldsymbol{h}_i \cap \boldsymbol{e}| = \ell] = \frac{\binom{w}{\ell}\binom{n-w}{t-\ell}}{\binom{n}{t}}$, par des arguments de combinatoire.

D'où, comme nous considérons r équations de parité, nous pouvons estimer la valeur moyenne des E_ℓ grâce à $\mathbb{E}[E_\ell] = r\mathbb{P}[|\boldsymbol{h}_i \cap \boldsymbol{e}| = \ell]$. □

La propriété 37 permet d'estimer la valeur moyenne des E_ℓ pour une instance fixée, lorsque le poids de l'erreur et le poids du syndrome sont connus. Autrement dit, nous pouvons tirer parti de cette

propriété pour donner une estimation précise de la valeur des E_ℓ pour la première itération (lorsque le poids de l'erreur est connu) et pour une instance particulière, en fonction du poids de syndrome observé.

Remarque 40. *Bien que la propriété 37 ne peut être appliquée, en pratique, que pour estimer la valeur moyenne des E_ℓ pour la première itération, les résultats restent vrais pour toute itération, en supposant que la valeur du poids de l'erreur est connu et égal à t.*

Comme précédemment, nous définissons, pour $1 \leqslant j \leqslant n$, $\sigma_j \stackrel{\text{def}}{=} \sum_{i\,:\,j\in \boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle \in \{0,\dots,d\}$. La propriété 35 nous permet d'estimer la valeur moyenne de cette variable aléatoire, selon que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée.

Propriété 38. *Soient $H \in \mathcal{H}(n,r,w)$, $S \stackrel{def}{=} |\boldsymbol{e}H^t|$ où $\boldsymbol{e} \sim \mathcal{U}_t$ et $\boldsymbol{h}_i$ pour $1 \leqslant i \leqslant r$ désignant les équations de parité définies par H et $\sigma_j = \sum_{i\,:\,j\in \boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e}\rangle$. Alors,*

— *si $\mathcal{H}_1$ est vérifiée,*

$$\mathbb{E}[\sigma_j] = \frac{1}{t}S + \frac{1}{t}\sum_{\ell\geqslant 0} 2\ell \cdot E_{2\ell+1}$$

— *si $\mathcal{H}_0$ est vérifiée,*

$$\mathbb{E}[\sigma_j] = \frac{w-1}{n-t}S - \frac{1}{n-t}\sum_{\ell\geqslant 0} 2\ell \cdot E_{2\ell+1}$$

Démonstration. Premièrement, rappelons que la propriété 35, nous donne $\sum_{j\in \boldsymbol{e}} \sigma_j = S + \sum_{\ell\geqslant 0} 2\ell\cdot E_{2\ell+1}$. Or, $|\boldsymbol{e}| = t$ donc, si $\mathcal{H}_1$ est vérifiée, $\mathbb{E}[\sigma_j] = \frac{1}{t}\sum_{j\in \boldsymbol{e}} \sigma_j$.

Deuxièmement, la propriété 36 nous donne $\sum_{j\notin \boldsymbol{e}} \sigma_j = (w-1)S - \sum_{\ell\geqslant 0} 2\ell\cdot E_{2\ell+1}$ et comme $|\{j \,:\, j\notin \boldsymbol{e}\}| = n-t$, on obtiens, lorsque $\mathcal{H}_0$ est vérifiée $\mathbb{E}[\sigma_j] = \frac{1}{n-t}\sum_{j\notin \boldsymbol{e}} \sigma_j$. □

Remarque 41. *La propriété 38 montre que, lorsque le poids du syndrome augmente, l'espérance de la valeur des compteurs des positions justes comme des positions fausses augmente. Il semble donc approprié de choisir un seuil adaptatif en fonction du poids du syndrome, qui tiendrait compte de cette corrélation.*

Considérons maintenant, pour $1 \leqslant j \leqslant n$ fixé et pour $1 \leqslant i \leqslant r$ tel que $j \in \boldsymbol{h}_i$, avec $\boldsymbol{s} = \boldsymbol{e}H^t$ où $\boldsymbol{e} \sim \mathcal{U}_t$ et $H \in \mathcal{H}(n,r,w)$, les probabilités conditionnelles suivantes :

$$\begin{cases} C_0(S) \stackrel{\text{def}}{=} \mathbb{P}[\langle \boldsymbol{h}_i, \boldsymbol{e}\rangle = 1 |\, j\notin \boldsymbol{e} \text{ et } |\boldsymbol{s}| = S] \\ C_1(S) \stackrel{\text{def}}{=} \mathbb{P}[\langle \boldsymbol{h}_i, \boldsymbol{e}\rangle = 1 |\, j\in \boldsymbol{e} \text{ et } |\boldsymbol{s}| = S] \end{cases}$$

Pour donner une estimation de la distribution des σ_j, nous devons d'abord comprendre le comportement de ces probabilités.

Propriété 39. *Soient $H \in \mathcal{H}(n,r,w)$, $S = |\boldsymbol{e}H^t|$ où $\boldsymbol{e} \sim \mathcal{U}_t$ alors,*

$$C_0(S) = \frac{1}{d(n-t)} \left((w-1)S - \sum_{\ell \geqslant 0} 2\ell \cdot E_{2\ell+1} \right)$$

$$C_1(S) = \frac{1}{dt} \left(S + \sum_{\ell \geqslant 0} 2\ell \cdot E_{2\ell+1} \right)$$

Démonstration. Par définition, pour $1 \leqslant j \leqslant n$ fixé, nous avons $\sigma_j = \sum_{i \,:\, j \in \boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$. En supposant, que pour $1 \leqslant i \leqslant n$, les variables aléatoires $\langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$ sont indépendantes alors $\mathbb{E}[\sigma_j] = \mathbb{E}[\sum_{i \,:\, j \in \boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle]$. Or par hypothèse, nous avons fixé $|\{i \ : \ j \in \boldsymbol{h}_i\}| = d$, donc $\mathbb{E}[\sigma_j] = d\mathbb{E}[\langle \boldsymbol{h}_i, \boldsymbol{e} \rangle]$. Enfin comme $\mathbb{E}[\langle \boldsymbol{h}_i, \boldsymbol{e} \rangle] = \mathbb{P}[\langle \boldsymbol{h}_i, \boldsymbol{e} \rangle = 1]$ nous obtenons le résultat. □

La propriété 39 nous permet donc d'exprimer l'espérance des variables aléatoires σ_j pour $j \in \boldsymbol{e}$ et $j \notin \boldsymbol{e}$ respectivement, en fonction du poids du syndrome et de la valeur des E_ℓ de l'instance considérée.

Nous pouvons en déduire la distribution des σ_j, selon que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée, en fonction de ces mêmes grandeurs.

Propriété 40. *Soient $H \in \mathcal{H}(n,r,w)$ et $\boldsymbol{e} \sim \mathcal{U}_t$ alors, pour $1 \leqslant j \leqslant n$,*

$$\begin{cases} \mathbb{P}[\sigma_j = c | \{j \in \boldsymbol{e} \text{ et } |\boldsymbol{e}H^t| = S\}] &= \binom{d}{c} C_1(S)^c \cdot (1 - C_1(S))^{d-c} \\ \mathbb{P}[\sigma_j = c | \{j \notin \boldsymbol{e} \text{ et } |\boldsymbol{e}H^t| = S\}] &= \binom{d}{c} C_0(S)^c \cdot (1 - C_0(S))^{d-c} \end{cases}$$

Démonstration. On a $\sigma_j = \sum_{i \,:\, j \in \boldsymbol{h}_i} \langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$. Comme précédemment, on suppose que pour $i \in \{1, \ldots, r\}$, les variables aléatoires $\langle \boldsymbol{h}_i, \boldsymbol{e} \rangle$ sont iid. Donc σ_j est une somme de d variables aléatoires de Bernouilli iid de paramètre $C_b(S)$. □

En pratique, la distribution des E_ℓ n'est pas connue (puisque l'erreur est secrète). Cependant, nous considérons que l'estimation de la valeur des E_ℓ obtenue grâce à leur moyenne est assez précise pour fournir une estimation raisonnable de celles-ci. Autrement dit, nous utiliserons cette hypothèse pour analyser les résultats explicités ci-après.

Ces distributions nous permettent, comme dans la section 5.2, d'estimer le seuil optimal pour la première itération, en fonction du poids du syndrome. Ainsi, il est possible d'affiner le calcul du seuil en fonction de l'instance considérée à partir du poids du syndrome observé lors du décodage.

Les tables 7.1 et 7.2 indiquent que la distribution des compteurs varie en fonction du poids du syndrome et ceux pour les positions justes et fausses respectivement. Cela conforte l'idée qu'il faut adopter une fonction de seuil qui dépend du poids du syndrome pour optimiser le décodage de chaque instance.

Le choix du seuil optimal, en fonction du poids du syndrome, est illustré par les figures 7.1 et 7.2. En effet, la première figure représente la moyenne du nombre de positions, respectivement justes

	Poids du syndrome		
	$S = 1852$	$S = 1916$	$S = 1926$
$\mathbb{E}[\sigma_j]$	17.279	17.877	17.971
$\mathbb{V}[\sigma_j]$	10.644	10.775	10.794

TABLE 7.1 – Comparaison de l'espérance et de la variance des compteurs des positions justes à la première itération pour différents poids de syndrome

	Poids du syndrome		
	$S = 1892$	$S = 1916$	$S = 1926$
$\mathbb{E}[\sigma_j]$	26.890	27.175	27.295
$\mathbb{V}[\sigma_j]$	10.822	10.764	10.739

TABLE 7.2 – Comparaison de l'espérance et de la variance des compteurs des positions fausses à la première itération pour différents poids de syndrome

et fausses, ayant une valeur de compteurs fixe, pour les instances dont le syndrome a un poids égal à 1826. Ces courbes semblent indiquer qu'un choix de seuil optimal pour ce poids de syndrome et cette itération est 27. La seconde figure quant à elle, représente les mêmes données, pour un poids de syndrome égal à 1926. L'analyse de cette figure indique que le seuil optimal pour cette itération et ce poids de syndrome est 28. Cela confirme l'hypothèse selon laquelle le seuil adaptatif, en tant que fonction du poids du syndrome, est croissant.

Remarque 42. *Les valeurs des seuils optimaux théoriques ont été calculées pour optimiser le décodage de la première itération, contrairement aux valeurs de seuils adaptatifs trouvés expérimentalement grâce au processus explicité dans la section 6.3, qui correspondent à une optimisation globale de l'algorithme.*

À présent, nous pouvons déterminer, pour un poids de syndrome fixé, la distribution de l'erreur après la première itération. Plus précisément, nous allons expliciter la probabilité qu'une position soit modifiée, suivant qu'elle soit juste ou fausse, lorsque le seuil est calculé comme une fonction du poids du syndrome.

Propriété 41. *Soient* $H \in \mathcal{H}(n, r, w)$, $\boldsymbol{s} \stackrel{def}{=} \boldsymbol{e}H^t$ *où* $\boldsymbol{e} \sim \mathcal{U}_t$. *Nous noterons* $b(S)$ *la fonction de seuil et nous posons, pour* $1 \leqslant j \leqslant n$, $P_0(S, b(S)) \stackrel{def}{=} \mathbb{P}[e_j^{(1)} = 1 | \{j \notin \boldsymbol{e} \text{ et } |\boldsymbol{s}| = S\}]$ *et* $P_1(S, b(S)) \stackrel{def}{=} \mathbb{P}[e_j^{(1)} = 0 | \{j \in \boldsymbol{e} \text{ et } |\boldsymbol{s}| = S\}]$ *alors,*

$$\begin{aligned} P_1(S, b(S)) &= \sum_{c=b(S)}^{d} \binom{d}{c} C_1(S)^c \cdot (1 - C_1(S))^{d-c} \\ P_0(S, b(S)) &= \sum_{c=b(S)}^{d} \binom{d}{c} C_0(S)^c \cdot (1 - C_0(S))^{d-c} \end{aligned}$$

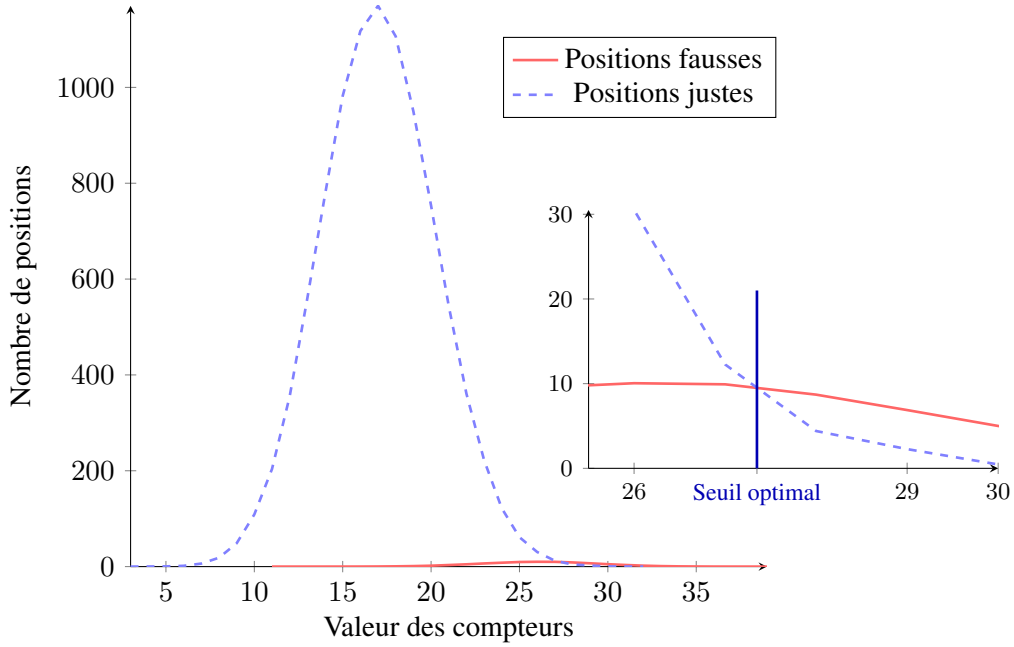


FIGURE 7.1 – Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la première itération pour un poids de syndrome égal à 1826

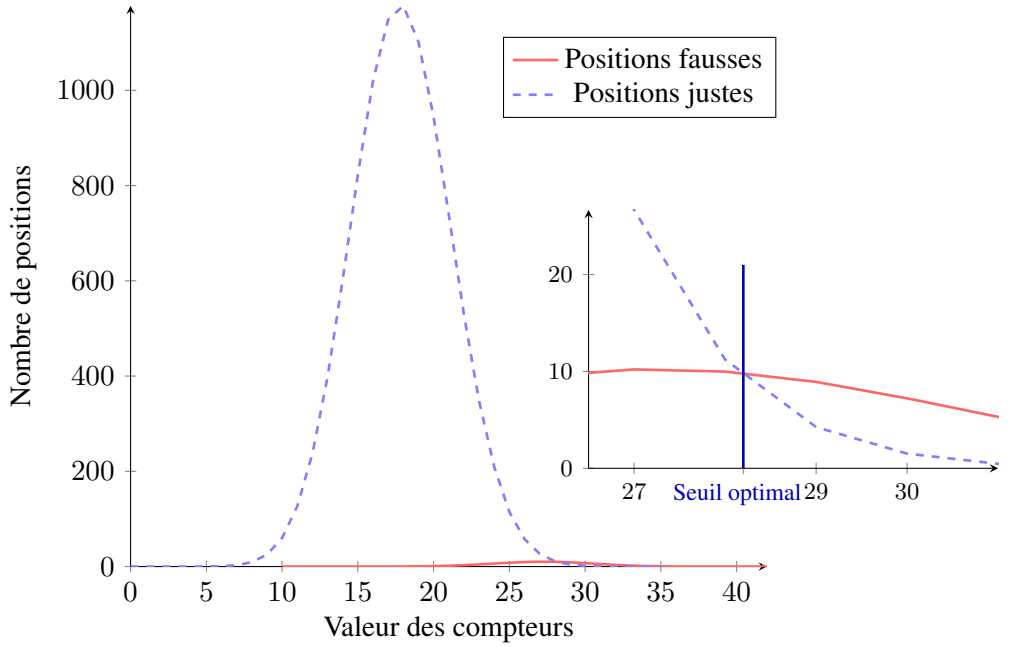


FIGURE 7.2 – Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la première itération pour un poids de syndrome égal à 1916

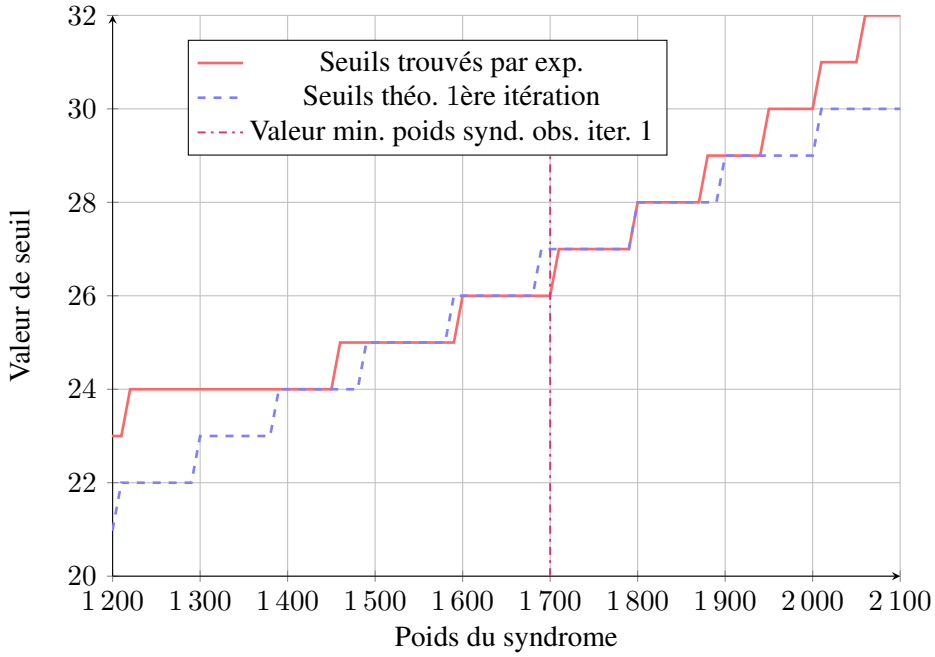


FIGURE 7.3 – Comparaison des valeurs théoriques et observées du seuil adaptatif (calculé en fonction du poids du syndrome)

Démonstration. Par définition,

$$\begin{aligned} P_0(S, b(S)) &= \mathbb{P}[e_j^{(1)} = 1 | \{j \notin \boldsymbol{e} \text{ et } |\boldsymbol{s}| = S\}] \\ &= \mathbb{P}[\sigma_j \geqslant b(S) | \{j \notin \boldsymbol{e} \text{ et } |\boldsymbol{s}| = S\}] \\ &= \sum_{c=b(S)}^{d} \mathbb{P}[\sigma_j = c | \{j \notin \boldsymbol{e} \text{ et } |\boldsymbol{s}| = S\}] \end{aligned}$$

□

Ceci nous permet aussi de donner la distribution du poids de l'erreur après une itération. En effet, il nous suffit de reprendre les résultats de la propriété 26 en fixant :

$$\mathbb{P}[e_j^{(1)} = 1 | j \notin \boldsymbol{e}] = \sum_{S=0}^{r} P^0(S, b_1(S)) \cdot \mathbb{P}[|\boldsymbol{s}| = S] \tag{7.2}$$

$$\mathbb{P}[e_j^{(1)} = 0 | j \in \boldsymbol{e}] = \sum_{S=0}^{r} P^1(S, b_1(S)) \cdot \mathbb{P}[|\boldsymbol{s}| = S] \tag{7.3}$$

La figure 7.4 illustre la différence entre les distributions du poids de l'erreur théorique à la fin de la première itération, pour différents poids de syndrome.

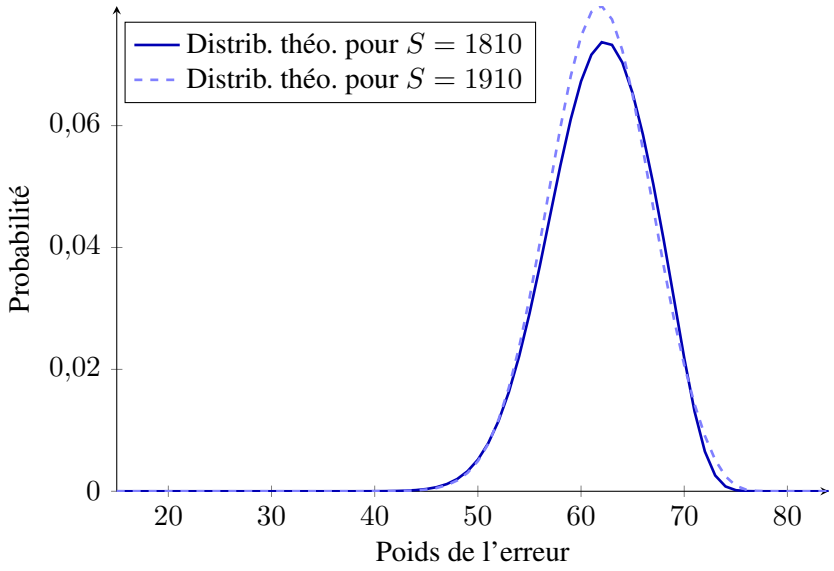


FIGURE 7.4 – Comparaison des distributions du poids de l'erreur à la fin de la première itération en fonction du poids du syndrome

7.4 Résultats observés pour différents paramètres

Nous avons comparé les modèles théoriques explicités dans la section précédente aux résultats observés lors de simulations de décodage pour deux jeux de paramètres :

(i) des codes $[9602, 4801, 90]$-QC-MDPC avec un poids d'erreur initial égal à 84,

(ii) des codes $[19714, 9857, 142]$-QC-MDPC avec un poids d'erreur initial égal à 134.

Ces paramètres correspondent aux propositions de [MTSB12] pour des niveaux de sécurité de respectivement 80 et 128 bits (voir la table 4.2 de la section 3.2).

Ainsi, nous pourrons comparer la précision des estimations fournies par l'analyse théorique présentée dans la section 7.3, pour différents paramètres.

Pour ce faire, nous avons utiliser un nombre de simulations de l'ordre de 10^7 pour tous les résultats présentés ci-après.

7.4.1 Pour une sécurité de 80 bits

La figure 7.5 illustre les distributions théoriques et observées de la distribution des valeurs des compteurs pour les positions justes pour différents poids de syndrome à la première itération. Nous pouvons remarquer que les résultats théoriques et observées coïncident bien. Donc le modèle théorique introduit dans la section 7.3 semble fournir une estimation fidèle du comportement de ces variables aléatoires.

La figure 7.6 représente quant à elle, les distributions théoriques et observées de la valeur des compteurs des positions fausses à la première itération pour différents poids de syndrome. Comme

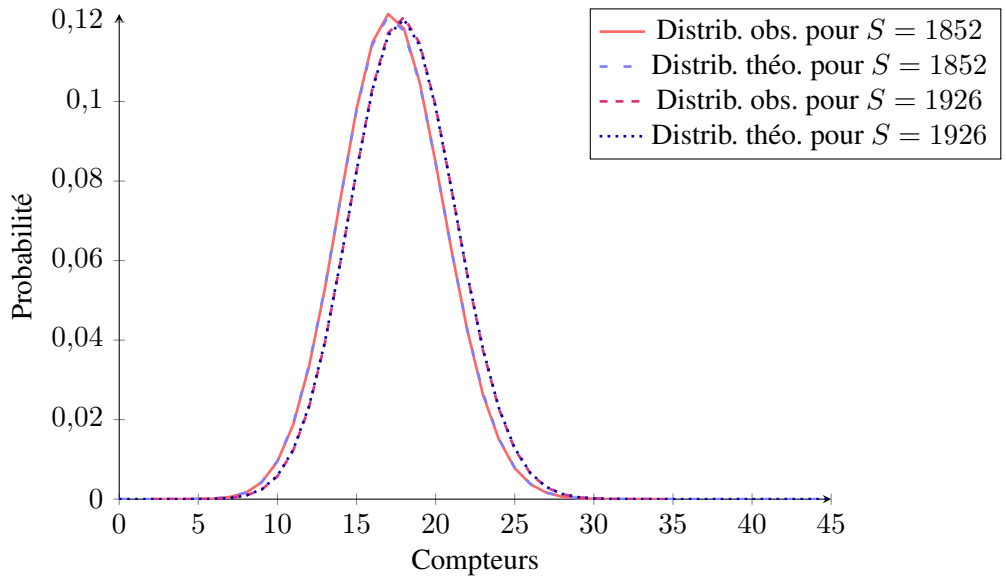


FIGURE 7.5 – Comparaison des valeurs théoriques et observées de la distribution de la valeur des compteurs pour les positions justes pour différents poids du syndrome pour des codes $[9602, 4801, 90]$-QC-MDPC avec un poids d'erreur initial égal à 84

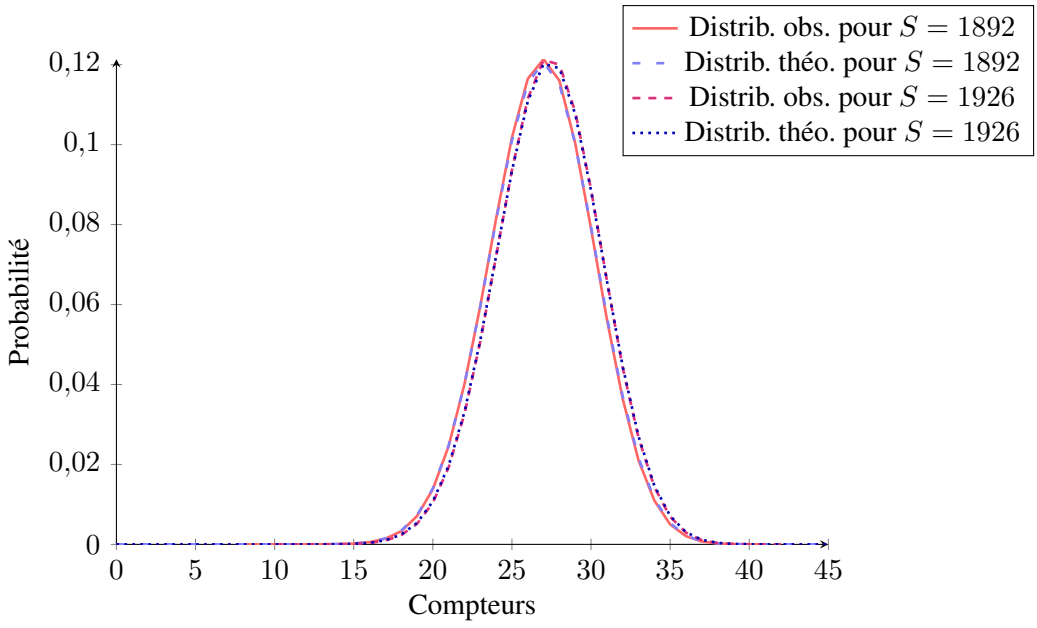


FIGURE 7.6 – Comparaison des valeurs théoriques et observées des distribution de la valeur des compteurs pour les positions fausses pour différents poids du syndrome pour des codes $[9602, 4801, 90]$-QC-MDPC avec un poids d'erreur initial égal à 84

pour les positions justes, nous pouvons constater que le modèle théorique paraît représentatif du comportement de l'algorithme.

Cependant, la figure 7.7 indique que la distribution du poids de l'erreur à la fin de la première itération ne suit pas fidèlement la distribution théorique prévue. Ainsi, l'étude des itérations suivantes semblent être compromise, puisque cette donnée est nécessaire pour fournir une estimation précise de la distribution des compteurs.

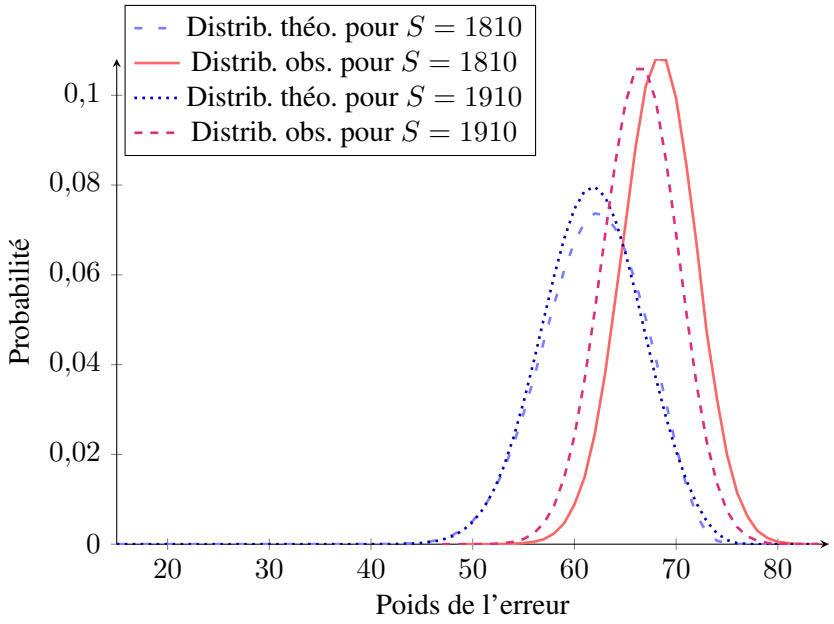


FIGURE 7.7 – Comparaison des valeurs théoriques et observées du poids de l'erreur en fonction du poids du syndrome pour des codes $[9602, 4801, 90]$-QC-MDPC avec un poids d'erreur initial égal à 84

7.4.2 Pour une sécurité de 128 bits

Afin de vérifier que les modèles présentés dans la section 7.3 sont cohérents quelque soient les paramètres considérés, nous avons effectué les mêmes simulations sur d'autres jeux de paramètres. En particulier, nous allons considérer le décodage d'une erreur aléatoire de poids $t = 134$ pour des codes $[19714, 9857, 142]$-QC-MDPC.

Premièrement, nous avons repris l'approche introduite dans la section 7.3 pour déterminer la fonction de seuil adaptative appropriée pour ces paramètres. Cette fonction est illustrée par la figure 7.8.

Deuxièmement, nous pouvons constater que, comme pour les codes $[9602, 4801, 90]$-QC-MDPC et un poids d'erreur initial égal à 84 (voir la figure 6.6 de la section 6.1), la distribution du poids du syndrome est plutôt mal estimée. Plus précisément, les variances théoriques et observées ne coïncident pas parfaitement. Mais ces résultats sont cohérents avec ceux que nous avons introduits dans la section 6.1 puisque nous avons vu que les positions du syndrome étaient corrélées. Autrement dit,

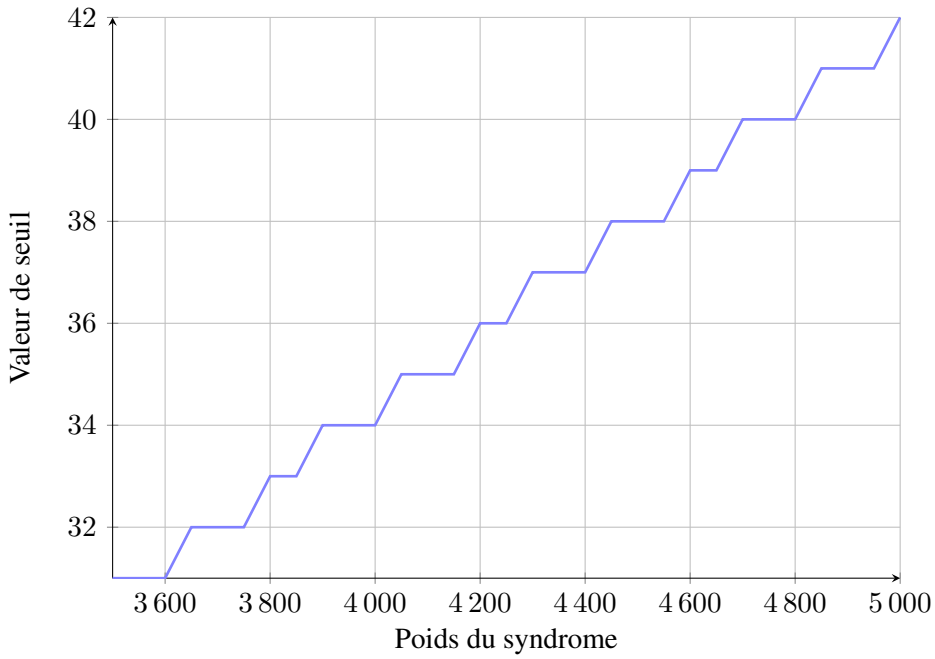


FIGURE 7.8 – Valeur de seuil adaptatif en fonction du poids du syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$

cela ne remet pas en cause les modèles théoriques proposés, mais confirme que, pour ces paramètres, les résultats de la section 6.1 sont aussi cohérents avec nos observations.

	Valeurs théoriques	Valeurs observées
$\mathbb{E}[S_0]$	4232.805	4232.827
$\mathbb{V}[S_0]$	2415.149	2217.240

TABLE 7.3 – Comparaison de l'espérance et de la variance du poids du syndrome observé et théorique au début de la première itération pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$

Ensuite, nous comparons les distributions théoriques et observées des valeurs des compteurs des positions justes et fausses pour différents poids de syndrome. Les tables 7.4 et 7.4 explicitent la valeur de l'espérance et de la variance des compteurs des positions justes au début de la première itération et pour différents poids de syndrome.

Comme précédemment, nous pouvons constater, d'une part que les modèles théoriques pour estimer la valeur des compteurs semblent bien coïncider avec nos observations. D'autre part, la valeur des compteurs augmente lorsque le poids du syndrome augmente, comme nous avons pu le constater pour d'autres paramètres.

	Poids du syndrome	
	$S = 4170$	$S = 4302$
$\mathbb{E}[\sigma_j]$ théorique	29.967	30.918
$\mathbb{E}[\sigma_j]$ observée	29.967	30.917
$\mathbb{V}[\sigma_j]$ théorique	17.319	17.454
$\mathbb{V}[\sigma_j]$ observée	17.182	17.324

TABLE 7.4 – Comparaison de l'espérance et de la variance des compteurs des positions justes à la première itération pour différents poids de syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$

	Poids du syndrome	
	$S = 4170$	$S = 4302$
$\mathbb{E}[\sigma_j]$ théorique	40.186	41.172
$\mathbb{E}[\sigma_j]$ observée	40.125	41.241
$\mathbb{V}[\sigma_j]$ théorique	17.441	17.297
$\mathbb{V}[\sigma_j]$ observée	17.278	17.121

TABLE 7.5 – Comparaison de l'espérance et de la variance des compteurs des positions fausses à la première itération pour différents poids de syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$

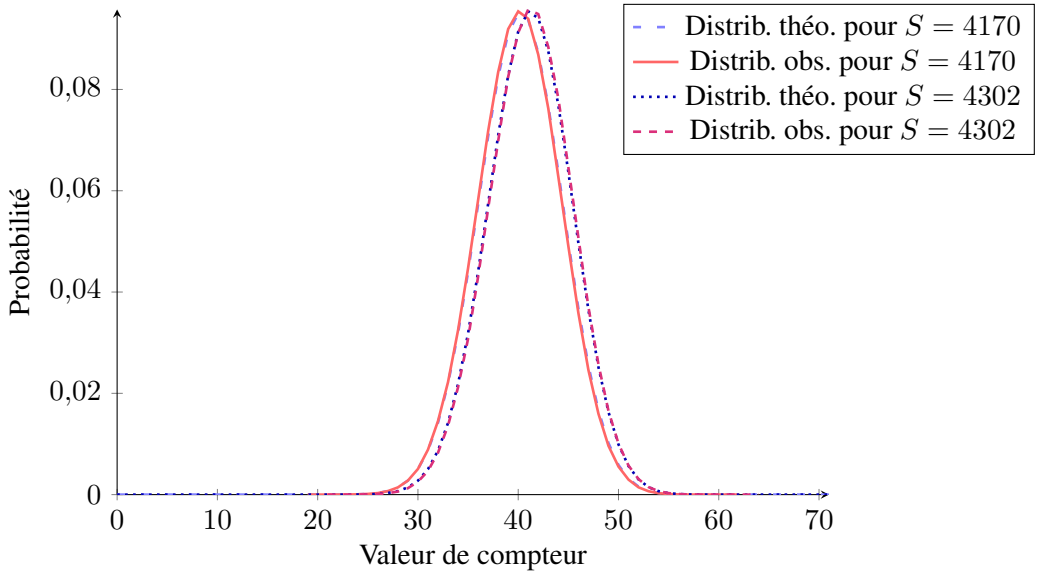


FIGURE 7.9 – Comparaison des valeurs théoriques et observées de la distribution des compteurs des positions justes en fonction du poids du syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$

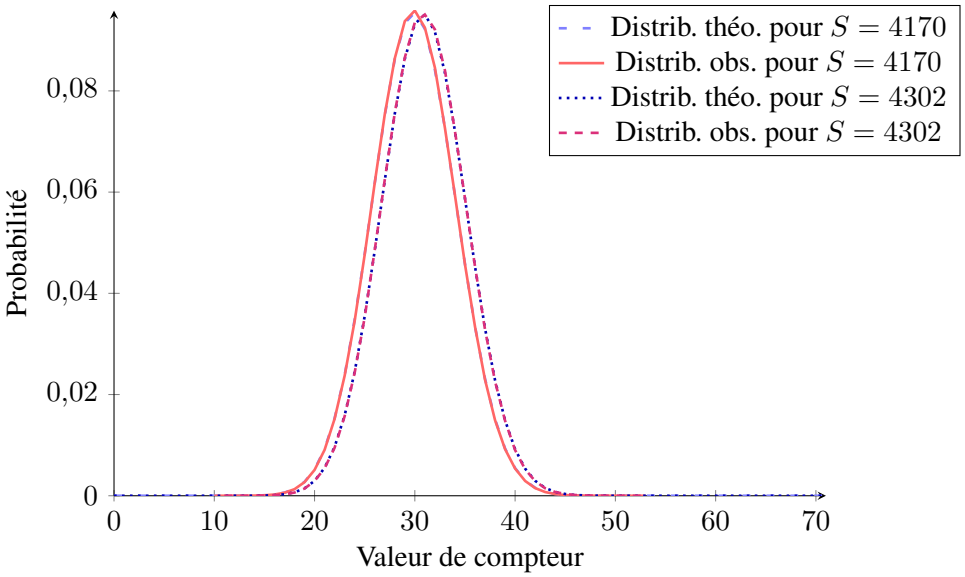


FIGURE 7.10 – Comparaison des valeurs théoriques et observées de la distribution des compteurs des positions fausses en fonction du poids du syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$

Conclusions et perspectives

Dans cette partie, nous avons, dans un premier temps, donné différentes méthodes permettant d'améliorer le calcul du seuil par rapport à l'approche fournie par [Gal63]. La première amélioration provient du fait que le modèle considéré est plus proche du modèle réel dans un contexte cryptographique. Cependant, nous avons pu constater que la précision des estimations des différentes grandeurs qui caractérisent le comportement de l'algorithme de décodage n'était pas assez précise, dès la deuxième itération. En effet, ces résultats ne nous permettent pas, par exemple, d'estimer de façon fidèle la distribution du poids de l'erreur à la fin de la première itération. Malheureusement, cette distribution est essentielle pour estimer le comportement de l'algorithme pour les itérations suivantes et notamment, la probabilité d'échec au décodage.

La deuxième amélioration proposée consiste à optimiser heuristiquement le décodage, en adoptant un seuil qui dépend du poids du syndrome de l'instance. Cette amélioration ayant pour objectif de diminuer la probabilité d'échec au décodage, en adaptant le décodage spécifiquement pour éviter des comportements divergents.

Dans un troisième temps, nous avons fourni une analyse théorique du comportement du décodage. Plus précisément, nous avons introduit différentes grandeurs fondamentales, dont les valeurs caractérisent entièrement le comportement d'une instance fixée (voir la section 7.2). La plupart de ces grandeurs n'étant pas observables en pratique (lorsque le décodage est utilisé par l'algorithme de déchiffrement), elles ne peuvent pas être manipulées directement au cours du décodage. Cependant, ces résultats nous indiquent que le poids du syndrome est, en effet, l'une des valeurs clés pour appréhender le comportement du décodage pour une instance fixée. Plus précisément, le poids du syndrome permet d'exprimer la valeur de la plupart des autres grandeurs fondamentales de l'algorithme et notamment la valeur des compteurs, respectivement pour les positions justes et fausses. Grâce à ces résultats, nous avons pu, entres autres, fournir une estimation de la distribution de ces compteurs à la première itération en fonction du poids du syndrome. Ceci explique en partie pourquoi l'ajustement du seuil en fonction du poids du syndrome permet de diminuer la probabilité d'échec au décodage. De plus, nous avons constaté que la fonction de seuil trouvée heuristiquement (section 6.3) était cohérente avec les résultats théoriques connus, (explicités dans la section 7.3) au moins pour la première itération.

Ces résultats fournissent donc de nouveaux outils pour comprendre et améliorer le décodage des codes MDPC dans un contexte cryptographique. Il reste cependant de nombreuses pistes à explorer.

En premier lieu, il faudrait pouvoir estimer de manière précise la distribution du poids de l'erreur après la première itération en fonction du poids du syndrome. Ceci nous permettra de trouver, théoriquement, le seuil en fonction du poids du syndrome pour la deuxième itération et de vérifier que cette fonction est cohérente avec celle obtenue pour la première itération. Aussi, cela pourra confirmer l'hypothèse selon laquelle un calcul adaptatif de seuil en fonction du poids du syndrome et non en fonction des itérations permet de diminuer la probabilité que l'algorithme diverge. Dans le même temps, il y a de forte chance pour que ces résultats puissent être étendus aux itérations suivantes et ainsi donner une bonne estimation de la probabilité d'échec au décodage.

En second lieu, nous pourrions tirer parti de ces résultats pour estimer la sécurité du cryptosystème, notamment contre les attaques par canaux cachés. En effet, dans un premier temps, nous pourrons fournir une estimation précise de la probabilité d'échec au décodage permettrait d'évaluer la complexité des attaques exploitant cette faiblesse. Dans un second temps, cette estimation nous permettra de définir les paramètres d'un cryptosystème dont la probabilité d'échec au décodage est négligeable et donc qui ne serait pas menacé par ces attaques. Enfin, nous pourrons déterminer les grandeurs observables sensibles, c'est-à-dire dont un adversaire pourrait disposer pour mener une attaque par canaux cachés. En effet, nous avons vu qu'un échec au décodage était susceptible de fournir des informations sur le secret (voir [GJS16]). De même, les articles [vMG14a, vMG14b] fournissent des techniques pour retrouver une partie du secret grâce à l'analyse de courant et du temps de l'étape de calcul du syndrome. Il est donc raisonnable de penser que d'autres grandeurs pourraient être exploitées afin de déduire des informations sur les instances de décodage. Identifier ces grandeurs nous permettrait, entre autre, d'adopter les contremesures nécessaires.

En dernier lieu, nous pouvons supposer que ces résultats fournissent assez d'éléments pour améliorer le coût du décodage. La première piste à explorer paraît être le choix du seuil. En effet, bien que l'approche qui consiste à considérer un seuil adaptatif (en fonction du poids du syndrome) offre déjà de bons résultats, il n'est pas exclu que d'autres méthodes amélioreront encore le décodage. La seconde piste à considérer pourra être l'utilisation d'information souple lors du décodage. Nous pourrons, par exemple, exploiter, lors d'une itération, les informations observées à l'itération précédente. Cela compliquera peut être l'analyse du décodage mais fournira un décodeur très efficace (dont le nombre maximal d'itérations est faible).

PARTIE IV

CRYPTANALYSE D'UN SCHÉMA DE CHIFFREMENT DE MCELIECE UTILISANT LES CODES POLAIRES

Introduction

Dans la section 2.4 nous avons vu que de nombreuses familles de codes avaient été proposées pour être utilisées dans le cryptosystème de McEliece. Cependant, dans de nombreux cas, la présence d'une structure algébrique identifiable a mené à des attaques, en retrouvant le code privé ou un décodeur efficace. En ce sens, les codes polaires semblent être une proposition raisonnable puisqu'ils ne sont pas, *a priori* dotés d'une structure algébrique trop forte. De plus, ces codes ont une capacité de correction proche des bornes théoriques issues de la théorie de l'information ([CT91]) et sont munis d'algorithmes de décodage très rapides, ce qui en fait un très bon candidat pour le cryptosystème de McEliece. En effet, d'une part, la sécurité de la clé ne semble pas être remise en cause par la présence d'une structure algébrique identifiable. D'autre part, la très bonne capacité de correction de ces codes permet d'utiliser des clés de tailles raisonnables. Enfin, les algorithmes de décodages étant très performants, l'étape de déchiffrement est peu coûteuse en pratique.

L'une des principales particularités de l'utilisation de ces codes en cryptographie est qu'ils sont, à paramètres fixés et à permutation sur les coordonnées près, uniques. Généralement, les familles de codes considérées dans ce contexte contiennent de nombreux codes, de sorte que l'ensemble des clés possibles soit large. Dans le contexte des codes polaires, comme dans celui des codes de Reed-Muller, c'est l'utilisation d'une permutation qui permet de générer la clé publique, à partir du code sélectionné (déterminé par le choix des paramètres). Dans ce cas, retrouver la clé privée revient à identifier la permutation qui a été utilisée, c'est-à-dire à résoudre une équivalence de codes. Or, ce problème, introduit dans [PR97], peut être résolu efficacement pour la plupart des familles de codes en utilisant l'algorithme de séparation du support (ou *support splitting algorithm*) explicité dans [Sen00]. Celui-ci nécessite que le code considéré admette un "petit" groupe de permutations et que son hull (l'intersection du code avec son dual) soit de petite dimension. Heuristiquement, nous avons pu constater que la dimension du hull d'un code polaire est souvent élevée (de l'ordre de la dimension du code) et que son groupe d'automorphisme est gros. Les codes polaires semblent donc être un bon candidat pour les cryptosystèmes fondés sur les codes correcteurs d'erreurs. Les codes de Reed-Muller, dont la définition est très proche de celle des codes polaires, ont aussi un gros groupe d'automorphisme et la dimension de leur hull n'est pas faible. Dans [Sid94], Sildenikov suggère l'utilisation des codes de Reed-Muller pour le cryptosystème de McEliece, en s'appuyant sur le fait que l'algorithme de séparation du support n'est pas efficace pour ces codes pour affirmer que la sécurité de la clé est assurée. Cependant, [MS07] propose une attaque de ce schéma, qui utilise les propriétés algébriques de ces codes pour retrouver la permutation secrète. Celles-ci n'étant pas

retrouvées dans les codes polaires, cette attaque ne peut être utilisée dans ce contexte. D'un autre côté, ces résultats indiquent que les codes polaires semblent avoir une structure algébrique assez proche de celle des codes de Reed-Muller. Notre idée fut donc d'analyser cette structure afin d'exploiter ses propriétés pour résoudre l'équivalence de codes.

Dans un premier temps, nous commencerons par définir les codes polaires et expliciter le cryptostystème de McEliece comme introduit dans [SK14]. Dans un second temps, nous définirons de manière formelle le problème d'équivalence de codes et nous donnerons les outils qui permettent d'en évaluer la difficulté dans certains contextes. Ensuite, nous introduirons une nouvelle famille de codes, appelée codes monomiaux décroissants, qui regroupent les codes polaires (pour certains paramètres) et les codes de Reed-Muller. Enfin, l'analyse des propriétés algébriques de cette famille de codes nous permettra de résoudre le problème d'équivalence de codes pour les codes polaires utilisés dans [SK14].

8 Codes polaires

Les codes polaires ont été introduits par Erdal Arikan en 2009 [Ari09]. Cette famille de codes a de très bonnes propriétés pour le décodage puisque leur capacité de correction est presque aussi bonne que la limite théorique donnée par la théorie de l'information (pour certains canaux) et leurs algorithmes de décodage sont très peu coûteux en mémoire et en temps.

8.1 Construction

La construction des codes polaires est très similaire à celle des codes de Reed-Muller. Toutes les deux reposent sur la donnée d'une matrice formée du produit de Kronecker d'une matrice particulière.

Définition 34 (Produit de Kronecker)**.** *Soient $A = (a_{i,j}) \in \mathbb{F}_2^{k_a \times n_a}$ et $B = (b_{i,j}) \in \mathbb{F}_2^{k_b \times n_b}$. Le produit de Kronecker de A et B, noté $A \otimes B$, est la matrice de $\mathbb{F}_2^{k_a k_b \times n_a n_b}$ suivante :*

$$A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,n_a}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,n_a}B \\ & & \ddots & \\ a_{k_a,1}B & a_{k_a,2}B & \dots & a_{k_a,n_a}B \end{pmatrix}$$

où pour $1 \leqslant i \leqslant k_a$ et $1 \leqslant j \leqslant n_a$,

$$a_{i,j}B = \begin{pmatrix} a_{i,j}b_{1,1} & a_{i,j}b_{1,2} & \dots & a_{i,j}b_{1,n_b} \\ a_{i,j}b_{2,1} & a_{i,j}b_{2,2} & \dots & a_{i,j}b_{2,n_b} \\ & & \ddots & \\ a_{i,j}b_{k_b,1} & a_{i,j}b_{k_b,2} & \dots & a_{i,j}b_{k_b,n_b} \end{pmatrix}$$

La construction d'une matrice génératrice d'un code polaire de longueur $n = 2^m$ repose sur la donnée de la matrice $G_m \in \mathbb{F}_2^{n \times n}$ avec

$$G_m \overset{\text{def}}{=} \underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_{m \text{ fois}}.$$

En effet, une matrice génératrice d'un code polaire de longueur n et de dimension k est construite grâce à k lignes de G_m. Les lignes à extraire seront choisies en fonction de l'algorithme de décodage.

Remarque 43. *La structure des codes polaires est définie par le choix des lignes à extraire, qui dépend de l'algorithme de décodage et donc n'est* a priori *pas relié à des propriétés algébriques particulières.*

Même si, pour les codes de Reed-Muller, le choix des lignes à extraire dépend directement de certaines propriétés algébriques, tandis que pour les codes polaires, ce choix ne dépend que des propriétés du décodeur, nous pouvons remarquer que leur construction est assez proche.

Exemple 2. *Pour $m = 3$, on a*

$$G_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Une matrice génératrice du $[2^3, 5]$-code polaire est

$$G_{pol} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Le code de Reed-Muller $\mathcal{R}(1, 3)$ admet pour matrice génératrice

$$G_{\mathcal{R}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

8.2 McEliece avec les codes polaires

L'utilisation des codes polaires pour le cryptosystème de McEliece a été proposée par Sujan R. Shrestha et Young-Sik Kim dans [SK14]. Les auteurs de [HSEA14], quant à eux, suggèrent de considérer des sous-codes de codes polaires. Dans cette partie, nous nous intéresserons plus particulièrement à la proposition [SK14]. Le cryptosystème proposé dans cet article est détaillé par la table 8.1. Dans le cas classique (voir la section 2.1), le code privé est un code choisi aléatoirement dans la famille de codes considérée. La structure de ce code privé, définie par la matrice génératrice

sélectionnée, est alors masquée en utilisant une matrice de permutation (et un changement de base). Ici, il est nécessaire d'utiliser une autre approche puisque les codes polaires sont définis de manière unique à paramètres fixés. La clé privée n'est donc pas le code (puisque celui-ci est connu) mais la permutation qui a été utilisée. Dans ce contexte, elle n'a pas pour vocation de cacher la structure du code, mais d'en mélanger les positions de sorte que le chiffré corresponde à la permutation d'un mot de code bruité. Celle-ci forme donc la clé privée, puisque le déchiffrement ne peut s'effectuer que si elle est connue.

Génération des paramètres
Soit $\mathcal{C}$ le $[n, k]$-code polaire muni d'un algorithme de décodage $\Psi_{\mathcal{C}}$ capable de corriger t erreurs.

Génération des clés

— Clé publique : $G' = SGP \in \mathbb{F}_2^{k\times n}$ où $G \in \mathbb{F}_2^{k\times n}$ est la matrice génératrice de $\mathcal{C}$ sous sa forme initiale, S est une matrice inversible de $\mathbb{F}_2^{k\times k}$ et P une matrice de permutation dans $\mathbb{F}_2^{n\times n}$

— Clé privée : $\Psi_{\mathcal{C}}$, $S \in \mathbb{F}_2^{k\times k}$ et $P \in \mathbb{F}_2^{n\times n}$ définis comme ci-dessus.

Chiffrement
Soit $\boldsymbol{m} \in \mathbb{F}_2^k$ le message clair et $G' \in \mathbb{F}_2^{k\times n}$ la clé publique.
On choisit aléatoirement $\boldsymbol{e} \in \mathbb{F}_2^n$ tel que $wt(\boldsymbol{e}) = t$.
Le chiffré est $\boldsymbol{c} = \boldsymbol{m}G' + \boldsymbol{e} \in \mathbb{F}_2^n$

Déchiffrement
Étant donné $\boldsymbol{c} \in \mathbb{F}_2^n$ un chiffré et $\Psi_{\mathcal{C}}$, S et P la clé secrète.
Calculer $\boldsymbol{c}P^{-1} = \boldsymbol{m}SG + \boldsymbol{e}P^{-1}$.
Retrouver $\boldsymbol{m}SG = \Psi_{\mathcal{C}}(\boldsymbol{c}P^{-1})$ puis en déduire $\boldsymbol{m}G = \boldsymbol{m}SGS^{-1}$.

TABLE 8.1 – Cryptosystème de McEliece utilisant les codes polaires

Dans cette partie, nous allons considérer que ce cryptosystème utilise le code polaire de longueur 2048 et de dimension 614. La très bonne capacité de correction de ces codes permet de décoder une erreur de poids $t = 200$, ce qui correspond à un niveau de sécurité de 105 bits, pour le décodage générique (voir la sous-section 2.3.3). En comparaison, lorsque le cryptosystème de McEliece est instancié avec des codes de Goppa de même taux de transmission, le poids de l'erreur utilisé est $t = 130$, ce qui donne une sécurité de 70 bits (toujours par rapport au décodage générique).

La sécurité de ce cryptosysème repose d'une part sur la difficulté de résoudre le problème du décodage générique (lorsque la clé privée n'est pas connue) et d'autre part sur la difficulté de résoudre le problème d'équivalence de codes, que nous explicitons ci-après.

9 Équivalence de codes

La sécurité du cryptosystème présenté dans la table 8.1 repose sur la difficulté de résoudre deux problèmes. Premièrement, en ce qui concerne la sécurité du message : l'adversaire est face à un problème de décodage dans un code dont il ne connaît pas la structure. En d'autres termes, il doit résoudre le problème du décodage générique. Deuxièmement, nous expliciterons la sécurité de la clé, qui revient à analyser la difficulté de retrouver la clé privée (une matrice de permutation) lorsque la clé publique est connue. Dans le contexte des codes polaires, l'attaquant connaît une matrice génératrice du code polaire permuté et du code polaire, son but est donc de trouver la permutation privée. En d'autres termes, il doit résoudre un problème d'équivalence de codes.

9.1 Code permuté

Dans un premier temps, nous allons définir les notions de vecteur et de code permuté. Le groupe symétrique d'indice n sera noté S_n.

Définition 35 (Permutation d'un mot ou d'un code)**.** *Soient $\boldsymbol{x} = (x_i)_{0 \leqslant i \leqslant n-1} \in \mathbb{F}_2^n$ et $\pi \in S_n$. Notons $\boldsymbol{x}^\pi \stackrel{def}{=} (x_{\pi(i)})_{0 \leqslant i \leqslant n-1}$, le vecteur $\boldsymbol{x}$ permuté par π.*
Pour $\mathcal{C}$ un code linéaire de longueur n,

$$\mathcal{C}^\pi \stackrel{def}{=} \{\boldsymbol{c}^\pi \ : \ \boldsymbol{c} \in \mathcal{C}\}$$

Exemple 3. *Soient $\boldsymbol{x} = (1,1,0,1,0,1) \in \mathbb{F}_2^6$ et $\pi = (0\ 1\ 3)(2\ 5\ 4) \in S_6$.*

On a $\boldsymbol{x}^\pi = (x_{\pi(0)}, x_{\pi(1)}, \ldots, x_{\pi(5)}) = (1,1,1,1,0,0)$.

Remarque 44. *Les permutations forment un groupe pour la composition de fonctions. Dans la suite et quand le contexte le permet, nous noterons multiplicativement cette opération. Autrement dit, nous désignerons, pour $(\pi, \pi') \in S_n^2$, $\pi\pi' \stackrel{def}{=} \pi \circ \pi'$.*

De manière équivalente, un code permuté est défini grâce à une matrice génératrice permutée.

Propriété 42. *Soient $\mathcal{C}$ un $[n,k]$-code linéaire et $G \in \mathbb{F}_2^{k \times n}$ une matrice génératrice de $\mathcal{C}$. De plus, soient $\pi \in S_n$ une permutation et $P \in \mathbb{F}_2^{n \times n}$ la matrice de permutation associée à π. Alors, $GP \in \mathbb{F}_2^{k \times n}$ est une matrice génératrice de $\mathcal{C}^\pi$.*

Remarque 45. *Soit $\pi \in S_n$ une permutation. La matrice de permutation associée à π est la matrice $P \in \mathbb{F}_2^{n\times n}$ définie par $P = (p_{i,j})_{0\leqslant i,j\leqslant n-1}$ avec $p_{i,j} = 1 \Leftrightarrow i = \pi(j)$ et $p_{i,j} = 0$ sinon.*

Il se peut que certaines permutations sur les positions d'un code laissent le code invariant. Celles-ci forment le groupe de permutations du code.

Définition 36 (Groupe de permutations d'un code)**.** *Le groupe de permutations d'un code $\mathcal{C}$ de longueur n est l'ensemble des permutations $\pi \in S_n$ telles que $\mathcal{C}^\pi = \mathcal{C}$.*

Nous allons voir que, dans certains contextes, il est utile de considérer l'ensemble des images par l'action du groupe de permutations d'un code sur l'un des mots de code qu'il définit.

Définition 37 (Orbite d'un mot de code)**.** *Soient $\mathcal{C}$ un code linéaire et $\mathcal{G}$ un sous-groupe du groupe de permutations de $\mathcal{C}$. L'orbite de $\boldsymbol{c} \in \mathcal{C}$ selon $\mathcal{G}$ est l'ensemble*

$$\mathcal{O}_{\boldsymbol{c}}^{\mathcal{G}} \stackrel{def}{=} \{\boldsymbol{c}' \in \mathcal{C} \ : \exists\, \pi \in \mathcal{G} \text{ tel que } \boldsymbol{c}^\pi = \boldsymbol{c}'\}$$

Comme dans le cas classique, nous définissons le produit de composition des permutations d'un vecteur.

Propriété 43. *Pour tout $\boldsymbol{x} = (x_i)_{0\leqslant i\leqslant n-1} \in \mathbb{F}_2^n$ et pour tout $(\pi, \pi') \in S_n^2$ on a*

$$(\boldsymbol{x}^\pi)^{\pi'} = \boldsymbol{x}^{\pi\pi'}.$$

Démonstration. Soient $\pi, \pi' \in S_n$ et $\boldsymbol{x} \in \mathbb{F}_2^n$. Posons $\boldsymbol{y} = \boldsymbol{x}^\pi \in \mathbb{F}_2^n$. Ainsi, $\forall\ 0 \leqslant i \leqslant n-1$, on a $x_i = y_{\pi^{-1}(i)}$, puisque $y_j = x_{\pi(j)}$ où $j = \pi^{-1}(i)$.

D'un autre côté, $(\boldsymbol{x}^\pi)^{\pi'} = \boldsymbol{y}^{\pi'}$ avec $\forall\ 0 \leqslant i \leqslant n-1$, $y_{\pi'(i)} = y_{\pi^{-1}(\pi(\pi'(i))} = x_{\pi(\pi'(i)))}$.

Ainsi, $(\boldsymbol{x}^\pi)^{\pi'} = \boldsymbol{x}^{\pi\pi'}$. □

Dans la suite, nous utiliserons la notation suivante : pour $\pi, \hat{\pi} \in S_n$ et $\mathcal{G} \subseteq S_n$,

$$\pi\mathcal{G}\hat{\pi} \stackrel{\text{def}}{=} \{\pi\sigma\hat{\pi} \ : \ \sigma \in \mathcal{G}\}$$

La description du groupe de permutations d'un code permet de définir le groupe de permutation de son permuté.

Propriété 44. *Soient $\mathcal{C}$ un code linéaire de longueur n dont le groupe de permutations est $\mathcal{G}$ et $\pi \in S_n$ alors*

(i) $\left\{\hat{\pi} \in S_n \ : \ \mathcal{C}^{\hat{\pi}} = \mathcal{C}^\pi\right\} = \mathcal{G}\pi$

(ii) *$\mathcal{C}^\pi$ a pour groupe de permutations $\pi^{-1}\mathcal{G}\pi$.*

Démonstration. Soient $\mathcal{C}$ un code de longueur n dont le groupe de permutations est $\mathcal{G}$ et $\pi \in S_n$.

(i) Nous voulons prouver que

$$\left\{\hat{\pi} \in S_n \ : \ \mathcal{C}^{\hat{\pi}} = \mathcal{C}^{\pi}\right\} = \{\sigma\pi \ : \ \sigma \in \mathcal{G}\}$$

Soit $\hat{\pi} \in S_n$ telle que $\mathcal{C}^{\hat{\pi}} = \mathcal{C}^{\pi}$. Dans ce cas, $\mathcal{C}^{\hat{\pi}\pi^{-1}} = \mathcal{C}$.
Donc $\hat{\pi}\pi^{-1} \in \mathcal{G}$, *i.e.* $\exists\, \sigma \in \mathcal{G}$ telle que $\hat{\pi}\pi^{-1} = \sigma$. Autrement dit, $\hat{\pi} = \sigma\pi$ avec $\sigma \in \mathcal{G}$. Donc $\hat{\pi} \in \mathcal{G}\pi$.
D'un autre côté, soit $\hat{\pi} \in \mathcal{G}\pi$, ainsi $\exists\, \sigma \in \mathcal{G}$ telle que $\hat{\pi} = \sigma\pi$. On a donc $\mathcal{C}^{\hat{\pi}} = \mathcal{C}^{\sigma\pi} = (\mathcal{C}^{\sigma})^{\pi}$ par la propriété 43.
Mais, comme $\sigma \in \mathcal{G}$ on a $(\mathcal{C}^{\sigma})^{\pi} = \mathcal{C}^{\pi}$. Donc $\mathcal{C}^{\hat{\pi}} = \mathcal{C}^{\pi}$.

(ii) Prouvons maintenant que

$$\{\sigma \ : \ \sigma \in S_n \text{ et } (\mathcal{C}^{\pi})^{\sigma} = \mathcal{C}^{\pi}\} = \{\pi^{-1}\sigma'\pi \ : \ \sigma' \in \mathcal{G}\}$$

D'un côté, soit $\sigma \in S_n$ telle que $(\mathcal{C}^{\pi})^{\sigma} = \mathcal{C}^{\pi}$. La propriété 43 nous donne $\mathcal{C}^{\pi\sigma} = \mathcal{C}^{\pi}$. D'où $\mathcal{C}^{\pi\sigma\pi^{-1}} = \mathcal{C}$ et $\pi\sigma\pi^{-1} \in \mathcal{G}$.
Réciproquement, soit $\sigma' \in \pi^{-1}\mathcal{G}\pi$. On a $(\mathcal{C}^{\pi})^{\sigma'} = \mathcal{C}^{\pi\pi^{-1}\sigma\pi} = \mathcal{C}^{\sigma\pi} = \mathcal{C}^{\pi}$ pour $\sigma \in \mathcal{G}$.

□

9.2 Résolution du problème d'équivalence de codes

La sécurité du cryptosystème de McEliece utilisant les codes polaires ou les codes de Reed-Muller repose en partie sur la difficulté de résoudre un problème d'équivalence de codes. En effet, la sécurité de la clé privée réside dans la difficulté de retrouver la permutation privée, lorsque les descriptions du code et de son permuté sont connues. Dans le cas classique, le problème considéré est l'indistinguabilité du code public, dont la difficulté dépend du code considéré (il n'existe pas d'algorithme générique pour le résoudre). Dans ce contexte, la problématique est la même, puisque la difficulté de résoudre le problème d'équivalence de codes dépend de la structure du code considéré (et notamment de son groupe de permutation).

Problème 8 (Problème d'équivalence de codes - Recherche)**.**
Paramètres : $n \in \mathbb{N}$ *et* S_n *le groupe symétrique d'indice* n*.*
Instance : $\mathcal{C}$ *et* $\mathcal{C}'$ *deux codes linéaires de longueur* n*.*
Question : *Trouver* $\pi \in S_n$ *telle que* $\mathcal{C}^{\pi} = \mathcal{C}'$ *si elle existe.*

Remarque 46. *Si le groupe de permutations du code considéré n'est pas trivial, ce problème peut avoir de nombreuses solutions.*

À l'heure actuelle, la méthode la plus efficace pour résoudre ce problème est l'algorithme de séparation du support, ou *support splitting algorithm*, introduit par [Sen00]. Cet algorithme est basé sur la notion de signature, qui est définie comme une fonction du code ou des positions du code qui est invariante par permutation. Ainsi, si la signature d'une position du code n'est pas égale à la signature d'une position du code permuté, c'est que cette dernière n'est pas l'image de la position considérée par la permutation. Cet algorithme est probabiliste, puisqu'il s'agit de rejeter l'hypothèse qu'une position soit l'image d'une autre par une permutation et sa complexité dépend en partie de la "taille" du groupe de permutations du code considéré.

Définition 38 (Invariant)**.** *Soit $\mathcal{C}$ un code de longueur n et $\pi \in S_n$ une permutation. Un invariant est une fonction* $\mathrm{Inv} : \mathcal{C} \longrightarrow \mathcal{F}$ *telle que* $\mathrm{Inv}(\mathcal{C}) = \mathrm{Inv}(\mathcal{C}^\pi)$.

Un invariant est une fonction dont la valeur est invariante par permutation du code. Les propriétés globales du code, comme par exemple sa longueur ou sa distance minimale, sont des invariants. Ce type de fonctions peut être utilisé pour réduire les dimensions du problème d'équivalence considéré.

Parallèlement, pour retrouver la permutation, nous avons besoin de trouver des propriétés qui sont propres aux positions du code, puisque la permutation est définie par son action sur les positions.

Définition 39 (Signature)**.** *Soit $\mathcal{C}$ un code linéaire de longueur n. Une signature $\Sigma : \mathcal{C} \times \{1, \ldots, n\} \longrightarrow F$ est une application qui vérifie*

$$\forall \pi \in S_n, \Sigma(\mathcal{C}, i) = \Sigma(\mathcal{C}^\pi, \pi(i)) \text{ où } i \in \{1, \ldots, n\}$$

Une signature est une spécification des propriétés particulières de chacune (ou d'un ensemble) des positions du code. Pour des raisons pratiques et puisqu'il faudra calculer une signature pour chacune des positions du code, il est souhaitable d'utiliser des fonctions peu coûteuses. Une autre technique est d'effectuer une (ou plusieurs) opérations invariantes par permutation sur le code afin de se ramener à un problème d'équivalence de codes dont les dimensions sont plus petites. Dans [Sen00], Sendrier suggère, par exemple, de considérer le hull du code (*i.e.* l'intersection du code avec son dual). En effet, avec une forte probabilité, la dimension du hull d'un code aléatoire est petite (comme le montre [Sen97]).

Définition 40 (hull)**.** *Soit $\mathcal{C}$ un code linéaire. Le hull de $\mathcal{C}$, noté $\mathcal{H}(\mathcal{C})$, est*

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

Nous avons vu qu'une signature était une fonction invariante par permutation qui permet d'identifier des positions du code. Cependant, il se peut que plusieurs positions aient la même signature et dans ce cas, il devient difficile de distinguer les positions à l'intérieur de cet ensemble. Une signature sera dite discriminante si elle prend plusieurs valeurs en fonction des positions considérées. Si de plus, une signature admet une valeur différente pour chacune des positions du code, elle sera dite totalement discriminante.

Définition 41. *Soient $\mathcal{C}$ un code de longueur n, $\Sigma : \mathcal{C} \longrightarrow F$ une signature. On dit que*

(i) *Σ est discriminante si $\exists\, (i, j) \in \{0, \ldots, n-1\}^2$ tel que*

$$\Sigma(\mathcal{C}, i) \neq \Sigma(\mathcal{C}, j)$$

(ii) *Σ est totalement discriminante, si $\forall (i, j) \in \{0, \ldots, n-1\}^2$ tels que $i \neq j$ on a*

$$\Sigma(\mathcal{C}, i) \neq \Sigma(\mathcal{C}, j)$$

D'autres versions de cet algorithme fonctionnent bien, même lorsque la signature considérée n'est pas totalement discriminante. Nous ne les expliciterons pas dans ce manuscrit puisque la plupart du temps, il est plus efficace d'utiliser une signature discriminante.

Algorithme 4 Algorithme de séparation du support

Entrées : $\mathcal{C}$ et $\mathcal{C}'$ deux codes linéaires équivalents de longueur n et Σ une signature totalement discriminante

Sortie : π telle que $\mathcal{C}' = \mathcal{C}^\pi$

for $(i, j) \in \{1, \ldots, n\}^2$ **do**
  if $\Sigma(\mathcal{C}, i) = \Sigma(\mathcal{C}', j)$ **then**
    $\pi(i) = j$
  end if
end for
return π

10 Codes monomiaux décroissants

Dans ce chapitre, nous allons introduire une nouvelle structure algébrique, qui nous permettra de mieux comprendre et d'appréhender la structure des codes polaires. Pour ce faire, nous donnerons la définition d'une nouvelle famille de codes : les codes monomiaux décroissants, dont les codes de Reed-Muller et certains codes polaires font partie.

La description de cette famille de codes et de sa structure nous permettront de comprendre, d'une part, les raisons pour lesquelles l'algorithme de séparation du support n'est pas efficace pour les codes de Reed-Muller et les codes polaires et d'autre part, de trouver des propriétés spécifiques à certains codes polaires (vus comme des codes monomiaux décroissants), qui nous aiderons à résoudre le problème d'équivalence de codes pour le code polaire utilisé dans [SK14].

Dans ce chapitre, nous donnerons, dans un premier temps, une définition équivalente des codes de Reed-Muller en utilisant un formalisme polynomiale. Cette description nous permettra d'appréhender le concept de code polynomial que nous définirons par la suite. Enfin, nous expliciterons différentes propriétés des codes monomiaux décroissants, notamment sur leur distance minimale, leur dual et leur groupe d'automorphisme, qui nous permettrons, entres autres, de mieux comprendre la structure algébrique des codes polaires.

10.1 Codes de Reed-Muller

Les codes de Reed-Muller de longueur 2^m peuvent être vus comme des codes d'évaluations de polynômes de $\mathbb{F}_2[x_0, \dots, x_{m-1}]$ et nous allons voir que ce formalisme s'applique aussi aux codes polaires.

Rappelons que nous considérons des codes binaires. Nous nous intéresserons donc à l'évaluation de ces polynômes dans $\mathbb{F}_2^m$. Nous identifierons x_i à x_i^2 et nous nous placerons dans l'anneau $\mathbb{R}_2[x_0, \dots, x_{m-1}] = \mathbb{F}_2[x_0, \dots, x_{m-1}]/(x_0^2 - x_0, \dots, x_{m-1}^2 - x_{m-1})$.

Nous associerons un polynôme $g \in \mathbb{R}_2[x_0, \dots, x_{m-1}]$ au vecteur binaire $\mathsf{ev}(g) \in \mathbb{F}_2^n$, où $n = 2^m$, qui est l'évaluation de g en chacun des éléments de $\mathbb{F}_2^m$. En d'autres termes,

$$\forall g \in \mathbb{R}_2[x_0, \dots, x_{m-1}], \;\; \mathsf{ev}(g) \stackrel{\text{def}}{=} \big(g(u_0, \dots, u_{m-1})\big)_{(u_0,\dots,u_{m-1}) \in \mathbb{F}_2^m}$$

Avec cette notation, les indices des coordonnées d'un vecteur correspondent aux éléments de $\mathbb{F}_2^m$. De plus, nous utiliserons l'ordre usuel en considérant que le vecteur $(u_0, u_1, \ldots, u_m) \in \mathbb{F}_2^m$ correspond à l'entier $\sum_{i=0}^{m-1} u_i 2^i$.

Exemple 4. *Soient $m = 3$ et $g(x_0, x_1, x_2) = x_0x_1 + x_2 \in \mathbb{R}_2[x_0, x_1, x_2]$. On a*

$$\begin{aligned} \mathsf{ev}(g) &= (g_0, g_1, \ldots, g_{2^3-1}) \\ &= (g(0,0,0), g(1,0,0), g(0,1,0), \ldots, g(0,1,1), g(1,1,1)) \\ &= (0,0,0,1,1,1,1,0) \end{aligned}$$

Définition 42 (Code de Reed-Muller)**.** *Les codes de Reed-Muller de paramètres r et m sont définis comme*

$$\mathcal{R}(r,m) \stackrel{def}{=} \{\mathsf{ev}(P) \ : \ P \in \mathbb{R}_2[x_0, \ldots, x_{m-1}] \text{ tel que } \deg P \leqslant r\}$$

Sachant que l'ensemble des polynômes de $\mathbb{R}_2[x_0, \ldots, x_{m-1}]$ est aussi décrit comme un espace vectoriel engendré par les monômes de $\mathbb{R}_2[x_0, \ldots, x_{m-1}]$, les codes de Reed-Muller sont aussi définis comme les codes engendrés par les évaluations d'une base de monômes en m variables et de degré inférieur ou égal à r.

Par la suite, l'ensemble des monômes en m variables sera noté

$$\mathcal{M} \stackrel{\text{def}}{=} \{1, x_0, \ldots, x_{m-1}, x_0x_1, \ldots, x_0 \cdots x_{m-1}\}$$

Remarque 47. *Soit $g = x_{i_1}x_{i_2}\ldots x_{i_s} \in \mathcal{M}$ un monôme de degré s quelconque. Le support de $\mathsf{ev}(g)$ correspond à l'espace affine défini par $x_{i_1} = x_{i_2} = \ldots = x_{i_s} = 1$. En d'autres termes, on a $\mathrm{Supp}(\mathsf{ev}(g)) = \Big\{ \sum_{i=0}^{m-1} u_i 2^i \ : \ \forall i \in \{i_1, \ldots, i_s\}, u_i = 1$ et $u_i \in \mathbb{F}_2$ sinon$\Big\}$.*

Pour $g \in \mathcal{M}$, le nombre de ses coefficients non nuls correspond à la dimension de l'espace affine défini par son support. Si le degré de g est d, alors

$$|\,\mathrm{Supp}(\mathsf{ev}(g))| = 2^{m-d}$$

Dans la suite et pour simplifier les notations, nous écrirons plus succinctement, pour g dans $\mathbb{R}_2[x_0, \ldots, x_{m-1}]$, $\mathrm{Supp}(g) \stackrel{\text{def}}{=} \mathrm{Supp}(\mathsf{ev}(g))$.

Le groupe de permutations des codes de Reed-Muller $\mathcal{R}(r,m)$ est l'ensemble des transformations affines $\boldsymbol{x} \to A\boldsymbol{x} + \boldsymbol{b}$ où $A \in \mathbb{F}_2^{m\times m}$ est une matrice inversible et $\boldsymbol{b} \in \mathbb{F}_2^m$.

Cela vient des deux constatations suivantes :

— toute transformation affine et bijective $\mathcal{A} \ : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2^m$ peut aussi être considérée comme une permutation sur les positions du code, telle que

$$(u_0, \ldots, u_{m-1}) \mapsto \mathcal{A}(u_0, \ldots, u_{m-1})$$

— cette permutation laisse le code invariant, puisque pour $P \in \mathbb{R}_2[x_0, \ldots, x_{m-1}]$, $P(\mathcal{A}(x_0, \ldots, x_{m-1}))$ est un polynôme de degré inférieur ou égal au degré de P. Ainsi,

$$\mathsf{ev}(P) \in \mathcal{R}(r,m) \Rightarrow \mathsf{ev}(P \circ \mathcal{A}) \in \mathcal{R}(r,m)$$

10.2 Définition

Dans la section précédente, nous avons décrit les codes de Reed-Muller comme des codes dont les mots sont des évaluations de polynômes. Grâce à ce formalisme, les codes de Reed-Muller sont aussi définis comme des codes polynomiaux.

Définition 43 (Code polynomial)**.** *Soit $I \subseteq \mathbb{R}_2[x_0, \ldots, x_{m-1}]$ un ensemble fini de polynômes. Le code polynomial associé à I est l'espace vectoriel $\mathcal{C}(I) \subseteq \mathbb{F}_2^n$ engendré par*

$$\{\mathsf{ev}(f) \ : \ f \in I\}$$

Remarque 48. *Pour $I \subseteq \mathbb{R}_2[x_0, \ldots, x_{m-1}]$, la longueur du code polynomial $\mathcal{C}(I)$ est 2^m.*

Les codes polaires et les codes de Reed-Muller sont des codes polynomiaux. En effet, les constructions des codes polaires et des codes de Reed-Muller sont très semblables et nous pouvons remarquer que la matrice G_m introduite dans la section 8.1 est définie, de manière équivalente, comme la matrice formée des évaluations des monômes de $\mathbb{F}_2^m$.

Exemple 5. *Pour $m = 3$, on a*

$$G_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathsf{ev}(1) \\ \mathsf{ev}(x_0) \\ \mathsf{ev}(x_1) \\ \mathsf{ev}(x_0x_1) \\ \mathsf{ev}(x_2) \\ \mathsf{ev}(x_0x_2) \\ \mathsf{ev}(x_1x_2) \\ \mathsf{ev}(x_0x_1x_2) \end{pmatrix}$$

Le choix des lignes utilisées pour former la matrice génératrice des codes de Reed-Muller $\mathcal{R}(r, m)$ consiste à sélectionner les monômes de degré inférieur ou égal à r. Pour les codes polaires, ce choix dépend des propriétés du décodeur et ne sera pas explicité dans ce manuscrit. Cependant, ce choix s'appuie sur l'ordre des monômes comme introduit ci-après.

Définition 44 (Ordre sur les monômes)**.** *Les monômes de même degré sont ordonnés de sorte que*

$$x_{i_1} \cdots x_{i_s} \preceq x_{j_1} \cdots x_{j_s} \Leftrightarrow \forall \ell \in \{1, \ldots, s\}, \ \ i_\ell \leqslant j_\ell$$

où $i_1 < \ldots < i_s$ et $j_1 < \ldots < j_s$.
Cet ordre s'étend aux monômes de degrés quelconques en utilisant la divisibilité.

Soient $f, g \in \mathcal{M}$.

$$\exists\, g^* \in \mathcal{M} \text{ tel que } g^* \text{ divise } g \text{ et } f \preceq g^* \Leftrightarrow f \preceq g$$

Remarque 49. *Cet ordre n'est pas total. Par exemple, pour $m = 3$, les monômes x_2 et x_0x_1 ne sont pas comparables.*

Dans la suite, pour $f, g \in \mathcal{M}$ tels que $f \preceq g$, nous noterons

$$[f; g] \stackrel{\text{def}}{=} \{h \ : h \in \mathcal{M} \text{ et } f \preceq h \preceq g\}$$

Exemple 6. *Pour $m = 3$, on a $[x_0; x_1x_2] = \{x_0, x_1, x_2, x_0x_1, x_1x_2\}$.*

Définition 45 (Ensemble décroissant)**.** *Un ensemble de monômes $I \subseteq \mathcal{M}$ est dit décroissant si, et seulement si,*

$$f \in I \text{ et } g \preceq f \Rightarrow g \in I$$

Exemple 7. *Pour $m = 3$, l'ensemble $[x_0; x_1x_2]$ n'est pas décroissant puisque $1 \notin [x_0; x_1x_2]$ et $1 \preceq x_0$.*

Remarque 50. *Si $I \subseteq \mathcal{M}$ est un ensemble décroissant tel que $x_0 \cdots x_{m-1} \in I$ alors $I = \mathcal{M}$.*

Définition 46 (Code monomial et code monomial décroissant)**.** *Soient $I \subseteq \mathbb{R}_2[x_0, \ldots, x_{m-1}]$ un ensemble fini de polynômes et $\mathcal{C}(I)$ le code polynomial associé à I.*

(i) *Si $I \subseteq \mathcal{M}$ alors on dit que $\mathcal{C}(I)$ est un code monomial.*

(ii) *Si $I \subseteq \mathcal{M}$ et I est un ensemble décroissant alors $\mathcal{C}(I)$ est un code monomial décroissant.*

La dimension d'un code monomial correspond donc à la taille de la base de monômes qui l'engendre.

Lemme 2. *Pour tout $I \subseteq \mathcal{M}$, la dimension du code monomial $\mathcal{C}(I)$ est $|I|$.*

Les codes polaires et les codes de Reed-Muller sont donc des codes monomiaux. Cependant, la structure de cette famille de codes n'explique pas l'ensemble des propriétés des codes polaires et de Reed-Muller que nous avons pu mettre en évidence.

Propriété 45. *Le code de Reed-Muller de paramètres r et m (d'ordre r et en m variables) est le code monomial décroissant défini par l'intervalle $[1; x_{m-r} \ldots x_{m-1}]$* i.e.

$$\mathcal{R}(r, m) = \mathcal{C}\left([1; x_{m-r} \ldots x_{m-1}]\right).$$

La dimension de ce code est $1 + m + \cdots + \binom{m}{r}$.

Démonstration. D'une part, l'ensemble $[1; x_{m-r} \ldots x_{m-1}]$ est bien décroissant. Montrons maintenant que $\mathcal{R}(r, m) = \mathcal{C}\left([1; x_{m-r} \ldots x_{m-1}]\right)$. Tout d'abord, nous savons que $\mathcal{R}(r, m) = \mathcal{C}(I)$ est un code monomial, engendré par l'ensemble I des monômes de $\mathbb{R}_2[x_0, \ldots, x_{m-1}]$ de degré inférieur ou égale à r. Nous voulons donc prouver que $I = [1; x_{m-r} \ldots x_{m-1}]$.

Premièrement, nous pouvons remarquer que $x_{m-r} \ldots x_{m-1} \in I$ et que tout monôme $f \in I$ satisfait $f \preceq x_{m-r} \ldots x_{m-1}$.

Deuxièmement, les monômes de degré supérieur à r ne peuvent pas être plus petit que $x_{m-r} \ldots x_{m-1}$. Donc on a bien $I = [1; x_{m-r} \ldots x_{m-1}]$.

Le résultat sur la dimension du code $\mathcal{R}(r, m)$ se déduit directement du lemme 2. □

Les codes de Reed-Muller sont donc des codes monomiaux décroissants. De plus, [BDOT16] prouve que les codes polaires sont des codes monomiaux décroissants, pour tout les canaux binaires symétriques. En particulier, les codes polaires considérés pour l'instanciation du cryptosystème de McEliece sont monomiaux décroissants.

10.3 Propriétés

10.3.1 Description du dual

Le dual d'un code monomial décroissant a une description très simple. De plus, sous certaines conditions, ces codes sont faiblement auto-duaux.

Le dual d'un code monomial est toujours un code polynomial, mais qui n'est pas nécessairement monomial. Cependant, le dual d'un code monomial décroissant est encore un code monomial décroissant comme nous allons le voir.

Définition 47 (Complémentaire)**.** *Pour tout $g \in \mathcal{M}$ le complémentaire de g est $\check{g} \in \mathcal{M}$ tel que*

$$\check{g} \stackrel{def}{=} \frac{x_0 \dots x_{m-1}}{g}$$

Pour $I \subseteq \mathcal{M}$ un ensemble de monômes, nous définissons

$$\check{I} \stackrel{\text{def}}{=} \{\check{g} \ : \ g \in I\}$$

Remarque 51. *Par définition, pour $I \subseteq \mathcal{M}$, $|I| = |\check{I}|$.*

Dans un premier temps, nous allons voir que l'ordre entre deux monômes peut s'étendre à leurs complémentaires.

Propriété 46. *Soient $f, g \in \mathcal{M}$, alors*

$$f \preceq g \Leftrightarrow \check{f} \succeq \check{g}$$

Démonstration. Soient $f = x_{i_1} \cdots x_{i_s}$ et $g = x_{j_1} \cdots x_{j_t}$ avec $s \leqslant t$, $i_1 < \ldots < i_s$ et $j_1 < \ldots < j_t$. Par hypothèse, nous considérons $f \preceq g$ et nous distinguons deux cas :

— si $s = t$.
Soit $\mathcal{I} \stackrel{\text{def}}{=} \{i_1, \ldots, i_s\}$ avec $i_1 < i_2 < \ldots < i_s$ et $\mathcal{J} \stackrel{\text{def}}{=} \{j_1, \ldots, j_s\}$ avec $j_1 < j_2 < \ldots < j_s$ les indices des monômes f et g respectivement.
La définition de l'ordre que nous avons introduit nous donne :

$$f \preceq g \Leftrightarrow \forall\, k \in \{1, \ldots, s\}, i_k \leqslant j_k$$

Considérons maintenant, les ensembles $\mathcal{I}' \stackrel{\text{def}}{=} \{i'_1, \ldots, i'_{s'}\}$ et $\mathcal{J}' \stackrel{\text{def}}{=} \{j'_1, \ldots, j'_{s'}\}$ qui correspondent aux indices des monômes $\check{f}$ et $\check{g}$ respectivement, où $s' \stackrel{\text{def}}{=} m - s$. Par définition, nous posons $i'_1 < i'_2 < \ldots < i'_{s'}$ et $j'_1 < j'_2 < \ldots < j'_{s'}$.
Nous voulons montrer que $\check{f} \succeq \check{g}$ *i.e.*

$$\forall k \in \{1, \ldots, s'\},\ i'_k \geqslant j'_k$$

Par définition, nous savons que $\mathcal{I} \sqcup \mathcal{I}' = \mathcal{J} \sqcup \mathcal{J}' = \{0, \ldots, m-1\}$.
Considérons maintenant les vecteurs $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_2^m$ tel que $u_\ell = 1 \Leftrightarrow \ell \in \mathcal{I}$ et $v_\ell = 1 \Leftrightarrow j \in \mathcal{I}$. Autrement dit, les vecteurs $\boldsymbol{u}$ et $\boldsymbol{v}$ sont les vecteurs d'incidence de $\mathcal{I}$ et $\mathcal{J}$ respectivement.
Nous pouvons alors remarquer que :

(i) les complémentaires du $\boldsymbol{u}$ et $\boldsymbol{v}$, que l'on notera $\overline{\boldsymbol{u}}$ et $\overline{\boldsymbol{v}}$, correspondent aux vecteurs d'incidence de $\mathcal{I}'$ et $\mathcal{J}'$ respectivement,

(ii) puisque $i_1 < i_2 < \ldots < i_s$, on a, pour tout $k \in \{1, \ldots, s\}$,

$$\{i_1, \ldots, i_s\} \cap \{0, \ldots, i_k\} = \{i_1, \ldots, i_k\}$$

et de même pour $j_1, \ldots, j_s$,

(iii) par définition, $\forall k \in \{1, \ldots, s\}$, $\sum_{\ell=0}^{i_k} u_\ell = \sum_{\ell \in \mathcal{I} \,:\, \ell \leqslant i_k} u_\ell = k$ et de même $\sum_{\ell=0}^{j_k} v_\ell = k$.

En utilisant le (iii) nous pouvons montrer que

$$\exists\, k \in \{1, \ldots, s\} \text{ tel que } i_k \leqslant j_k \Rightarrow \sum_{\ell=0}^{i_k} u_\ell \geqslant \sum_{\ell=0}^{i_k} v_\ell$$

puisque $k = \sum_{\ell=0}^{i_k} u_\ell = \sum_{\ell=0}^{j_k} v_\ell = \sum_{\ell=0}^{i_k} v_\ell + \sum_{\ell=i_k+1}^{j_k} v_\ell$

Donc, $\forall k \in \{1, \ldots, s\}$, $i_k \leqslant j_k \Rightarrow \forall t \in \{0, \ldots, m-1\}$, $\sum_{\ell=0}^{t} u_\ell \geqslant \sum_{\ell=0}^{t} v_\ell$.

Supposons maintenant par l'absurde, que

$$\exists\, k' \in \{1, \ldots, s'\} \text{ tel que } i'_{k'} < j'_{k'}$$

Posons $i' \stackrel{\text{def}}{=} i'_{k'}$ et $j' \stackrel{\text{def}}{=} j'_{k'}$.

Comme précédemment, nous avons $\sum_{\ell=0}^{i'} \overline{u}_\ell > \sum_{\ell=0}^{i'} \overline{v}_\ell$.

Mais, par définition, $\sum_{\ell=0}^{i'} \overline{u}_\ell = m - \sum_{\ell=0}^{i'} u_\ell$ et $\sum_{\ell=0}^{i'} \overline{v}_\ell = m - \sum_{\ell=0}^{i'} v_\ell$

Ainsi, si $\sum_{\ell=0}^{i'} \overline{u}_\ell > \sum_{\ell=0}^{i'} \overline{v}_\ell$ alors, $\sum_{\ell=0}^{i'} u_\ell < \sum_{\ell=0}^{i'} v_\ell$ ce qui est contradictoire.

— si $s < t$, alors par définition de l'ordre,

$$f \preceq g \Leftrightarrow \exists\, g_1 \in \mathcal{M} \text{ tel que } g = g_1 g_2 \text{ avec } \deg(g_1) = \deg(f) \text{ et } f \preceq g$$

Nous avons prouver que, pour $f, g_1 \in \mathcal{M}$ tels que $\deg(f) = \deg(g_1)$, $f \preceq g_1 \Leftrightarrow \check{f} \succeq \check{g_1}$.
De plus, comme g_1 divise g, on a que $\check{g}$ divise $\check{g_1}$ donc $\check{g_1} \succeq \check{g}$.
Ainsi, $\check{g} \preceq \check{g_1} \preceq \check{f}$.

□

Cette propriété nous permet de déduire la propriété suivante.

Propriété 47. *Si $I \subseteq \mathcal{M}$ est un ensemble décroissant alors l'ensemble $\mathcal{M} \setminus \check{I}$ est aussi un ensemble décroissant de monômes.*

Démonstration. Premièrement, puisque $\mathcal{M} \setminus \check{I} \subseteq \mathcal{M}$, nous savons déjà que c'est un ensemble de monômes.

Prouvons maintenant que cet ensemble est décroissant.

Soient $f \in \mathcal{M} \setminus \check{I}$ et $g \in \mathcal{M}$ tel que $g \preceq f$. Supposons par l'absurde que $g \notin \mathcal{M} \setminus \check{I}$.

On a alors $g \in \check{I}$, *i.e* $\exists\, h \in I$ tel que $g = \check{h}$.

Comme $g \preceq f$, on a $\check{h} \preceq f$. En utilisant la propriété 46, nous déduisons $\check{\check{h}} \succcurlyeq \check{f}$ *i.e.* $h \succcurlyeq \check{f}$.

Or $h \in I$ et I est un ensemble décroissant de monômes, donc $\check{f} \in I$, *i.e.* $f \in \check{I}$ ce qui est contradictoire. □

Les propriété 46 et 47 nous permettent de décrire le dual d'un code monomial décroissant.

Propriété 48. *Si $I \subseteq \mathcal{M}$ est un ensemble décroissant alors le dual de $\mathcal{C}(I)$ est*

$$\mathcal{C}(I)^{\perp} = \mathcal{C}(\mathcal{M} \setminus \check{I}).$$

Démonstration. Dans un premier temps, nous pouvons remarquer que

$$\dim(\mathcal{C}(I)^{\perp}) = |\mathcal{M}| - \dim(\mathcal{C}(I)) = |\mathcal{M}| - |I|$$

et

$$\dim(\mathcal{C}(\mathcal{M} \setminus \check{I})) = |\mathcal{M} \setminus \check{I}| = |\mathcal{M}| - |\check{I}| = |\mathcal{M}| - |I|$$

Donc $\mathcal{C}(I)^{\perp} = \mathcal{C}(\mathcal{M} \setminus \check{I}) \Leftrightarrow \mathcal{C}(\mathcal{M} \setminus \check{I}) \subseteq \mathcal{C}(I)^{\perp}$.

Soient $f \in \mathcal{C}(I)$ et $g \in \mathcal{C}(\mathcal{M} \setminus \check{I})$.

Supposons par l'absurde que $g \notin \mathcal{C}(I)^{\perp}$, autrement dit que $|\operatorname{Supp}(f) \cap \operatorname{Supp}(g)| = \ell$ avec ℓ impair.

Remarquons que $\operatorname{Supp}(f) \cap \operatorname{Supp}(g) \neq \emptyset$ si, et seulement si, $\exists\, h = x_{i_1} x_{i_2} \cdots x_{i_s}$ tel que h divise f et h divise g. Or $\operatorname{Supp}(h)$ est l'espace affine défini par $x_{i_1} = x_{i_2} = \ldots = x_{i_s} = 1$ et de dimension 2^{m-s}.

Donc $|\operatorname{Supp}(h)|$ est impair si, et seulement si, $m = s$. Autrement dit si $h = x_0 x_1 \cdots x_{m-1}$.

Mais, comme h divise f et h divise g, on a $h = f = g$ d'où $h \in I$ et $h \in \mathcal{M} \setminus \check{I}$.

Or, par hypothèse, I est un ensemble décroissant donc $x_0 x_1 \cdots x_{m-1} \in I \Rightarrow I = \mathcal{M}$. Ainsi, $\check{I} = \check{\mathcal{M}} = \mathcal{M}$ donc $\mathcal{M} \setminus \check{I} = \emptyset$ ce qui est contradictoire.

□

Les propriétés 47 et 48 montrent que le dual d'un code monomial décroissant est bien un code monomial décroissant.

Exemple 8. *Pour les codes de Reed-Muller, on a $\mathcal{R}(r,m) = \mathcal{C}\left([1; x_{m-r} \cdots x_{m-1}]\right)$ et*

$$\begin{aligned} \mathcal{R}(r,m)^{\perp} &= \mathcal{C}(\mathcal{M} \setminus [x_0 \cdots x_{m-r-1}; x_0 \cdots x_{m-1}]) \\ &= \mathcal{C}\left([1; x_{r+1} \cdots x_{m-1}]\right) \\ &= \mathcal{R}(m-r-1, m). \end{aligned}$$

Corollaire 1. *Soit $\mathcal{C}(I)$ un code monomial décroissant tel que $I \subseteq \mathcal{M}$. Si $|I| \leqslant 2^{m-1}$ alors*

$$\mathcal{C}(I) \subseteq \mathcal{C}(I)^{\perp} \Leftrightarrow \forall f \in I,\ \check{f} \notin I.$$

Les codes polaires de taux de transmission inférieur à $1/2$ satisfont, dans la plupart des cas, les hypothèses du corollaire 1 et sont donc faiblement auto-duaux. Pour les codes polaires de taux de transmission supérieur, c'est le dual de ces codes qui satisfait ces hypothèses.

Cela est dû au processus de polarisation utilisé pour choisir les monômes qui définissent une base du code. Nous ferons l'hypothèse que pour les codes polaires utilisés dans le cryptosystème de McEliece, le corollaire 1 peut s'appliquer. Ceci explique pourquoi les codes polaires considérés sont faiblement auto-duaux et aussi pourquoi l'algorithme de séparation du support comme introduit dans [Sen00] n'est pas efficace. En effet, si $\mathcal{C}$ est un code faiblement auto-dual alors $\mathcal{C} \cap \mathcal{C}^{\perp} \stackrel{\text{def}}{=} \mathcal{H}(\mathcal{C}) = \mathcal{C}$.

10.3.2 Groupe de permutation

Les codes polynomiaux et monomiaux peuvent avoir un groupe de permutations trivial. Dans le cas des codes monomiaux, l'application d'une permutation affine permet de construire un code polynomial qui n'est pas nécessairement monomial.

Dans un premier temps, pour comprendre l'action d'une permutation $\pi \in S_n$ qui définit aussi une transformation affine sur $\mathbb{F}_2^m$, nous remarquons que pour tout monôme $f \in \mathcal{M}$ on a

$$\mathsf{ev}(f)^{\pi} = \mathsf{ev}(f \circ \pi)$$

où, d'un coté, π est vue comme une permutation sur les coordonnées (considérées comme des éléments de $\mathbb{F}_2^m$) et de l'autre côté, π est vue comme une permutation affine.

Cette égalité explique pourquoi l'image d'un code monomial par une permutation affine peut ne pas être un code monomial.

Exemple 9. *Soient $m = 3$, $I = \{x_1, x_2, x_0x_2\}$ et $\mathcal{C}(I)$ le code monomial engendré par les évaluations des monômes de I. Maintenant, soit π la permutation affine définie par*

$$\pi : \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \longrightarrow \begin{pmatrix} x_0 \\ x_0 + x_1 \\ x_2 \end{pmatrix}$$

Le code $\mathcal{C}(I)^{\pi}$ est engendré par les évaluations de $\{\pi(x_1), \pi(x_2), \pi(x_0x_2)\} = \{x_0 + x_1, x_2, x_0x_2\}$. Ce code est un code polynomial qui n'est pas monomial.

Cependant, la famille des codes monomiaux décroissants admet un très grand groupe de permutations comme nous allons le voir.

Définition 48 (Groupe affine triangulaire inférieur)**.** *Le groupe affine triangulaire inférieur $\mathbb{LTA}_m$ de $\mathbb{F}_2^m$ est défini comme l'ensemble des transformations affines sur $\mathbb{F}_2^m$ de la forme $\boldsymbol{x} \mapsto A\boldsymbol{x} + \boldsymbol{b}$ où $A \in \mathbb{F}_2^{m \times m}$ est une matrice triangulaire inférieure inversible et $\boldsymbol{b} \in \mathbb{F}_2^m$.*

Théorème 1. *Le groupe de permutations d'un code monomial décroissant en m variables contient $\mathbb{LTA}_m$.*

Démonstration. Soient $I \subseteq \mathcal{M}$ un ensemble décroissant, $\pi \in \mathbb{LTA}_m$ et $\boldsymbol{x} \in \mathbb{F}_2^m$. Posons $\boldsymbol{x}' = \pi(\boldsymbol{x})$. Il existe alors a_{ij} et ε_i dans $\mathbb{F}_2$ tels que, pour $0 \leqslant i \leqslant m-1$,

$$x'_i = x_i + \sum_{j<i} a_{ij} x_j + \varepsilon_i$$

Toute permutation affine $\pi \in \mathbb{LTA}_m$ agit aussi de façon naturelle sur les monômes de sorte que $\pi(x_{i_1} \cdots x_{i_s}) \stackrel{\text{def}}{=} x'_{i_1} \cdots x'_{i_s}$.

En d'autres termes, l'action de π sur $f \in \mathcal{M}$ est décrite par $f \circ \pi$ et celle-ci est telle que $\mathsf{ev}(f)^\pi = \mathsf{ev}(f \circ \pi)$.

Nous pouvons alors trouver $f \circ \pi$ et vérifier que c'est une somme de monômes inférieurs ou égaux à f pour l'ordre que nous avons introduit.

Comme I est un ensemble décroissant, tous ces monômes sont dans I. L'évaluation de cette somme de monômes, qui correspond à $\mathsf{ev}(f \circ \pi)$ appartient donc à $\mathcal{C}(I)$ et $\mathcal{C}(I)$ est donc invariant par π. □

Ce théorème explique pourquoi les codes polaires ont un groupe de permutations non trivial, qui correspond à un sous-groupe du groupe de permutations des codes de Reed-Muller.

10.3.3 Distance minimale

Dans un premier temps, nous rappelons un résultat bien connu sur la distance minimale des codes de Reed-Muller.

Théorème 2. *La distance minimale du code de Reed-Muller $\mathcal{R}(r,m)$ est 2^{m-r}.*

Remarque 52. *Les mots de code de poids minimal d'un code de Reed-Muller $\mathcal{R}(r,m)$ sont obtenus comme $\mathsf{ev}(x'_0 \cdots x'_{r-1})$ où $x'_0, \ldots, x'_{r-1}$ correspondent aux images de $x_0, \ldots, x_{r-1}$ par un changement de coordonnées affine et bijectif (*i.e *tels que $\boldsymbol{x}' = A\boldsymbol{x} + \boldsymbol{b}$ où $A \in \mathbb{F}_2^m$ est inversible et $\boldsymbol{b} \in \mathbb{F}_2^m$ quelconque).*

En d'autres termes, à action du groupe de permutations près, les codes de Reed-Muller n'admettent qu'un seul mot de poids minimal. Ceci facilite l'attaque du cryptosystème de McEliece qui utilise les codes de Reed-Muller, présentée dans [MS07] et explique, en partie, pourquoi l'attaque ne fonctionne pas pour les codes polaires puisque ceux-ci ont plusieurs mots de poids minimal (modulo le groupe de permutations) comme nous allons le voir.

Pour comprendre le calcul de la distance minimale d'un code monomial décroissant et plus particulièrement des codes polaires, nous introduisons les notations suivantes.

Définition 49. *Soit $\mathcal{C}(I)$ un code monomial décroissant en m variables. On pose*

$$\begin{aligned} r_-(\mathcal{C}(I)) &\stackrel{\text{def}}{=} \max\{r \ : \ \mathcal{R}(r,m) \subseteq \mathcal{C}(I)\} \\ r_+(\mathcal{C}(I)) &\stackrel{\text{def}}{=} \min\{r \ : \ \mathcal{C}(I) \subseteq \mathcal{R}(r,m)\} \end{aligned}$$

Une autre façon de définir ces quantités est de voir r_- comme le plus grand r pour lequel le monôme $x_{m-r} \dots x_{m-1}$ est dans I. En effet, nous avons vu que $\mathcal{R}(r,m) = \mathcal{C}([1; x_{m-r} \cdots x_{m-1}])$ en tant que code monomial décroissant. Quant à r_+, cette valeur correspond au plus grand r tel que $x_0 \cdots x_{r-1}$ est dans I. En d'autres termes, nous pouvons écrire de façon équivalente :

$$\begin{aligned} r_-(\mathcal{C}(I)) &= \max\{r \ : \ x_{m-r} \cdots x_{m-1} \in I\} \\ r_+(\mathcal{C}(I)) &= \max\{r \ : \ x_0 \cdots x_{r-1} \in I\} \end{aligned}$$

Remarque 53. *Pour $I \subseteq \mathcal{M}$ un ensemble décroissant, si $x_{m-d} \cdots x_{m-1} \in I$ alors tout monôme de degré d appartient à I, puisque $x_{m-d} \cdots x_{m-1}$ est le plus grand monôme de degré d (pour l'ordre que nous avons introduit). De plus, dans ce cas, tout monôme de degré inférieur à d appartient aussi à I, comme diviseur d'un monôme de degré d.*

Ces quantités nous permettent de décrire la distance minimale d'un code monomial décroissant.

Propriété 49. *Soit $\mathcal{C}(I)$ un code monomial décroissant en m variables, on a :*

(i) *la distance minimale de $\mathcal{C}(I)$ est égale à $2^{m-r_+(\mathcal{C}(I))}$,*

(ii) *$r_-(\mathcal{C}(I)^\perp)$ et $r_+(\mathcal{C}(I)^\perp)$ satisfont les égalités suivantes*

$$\begin{aligned} r_-(\mathcal{C}(I)^\perp) &= m - 1 - r_+(\mathcal{C}(I)) \\ r_+(\mathcal{C}(I)^\perp) &= m - 1 - r_-(\mathcal{C}(I)) \end{aligned}$$

(iii) *la distance minimale de $\mathcal{C}(I)^\perp$ est $2^{r_-(\mathcal{C}(I))+1}$.*

Démonstration. Soit $I \subseteq \mathcal{M}$ un ensemble décroissant de monômes.

(i) Il nous faut prouver que la distance minimale de $\mathcal{C}(I)$ est égale à 2^{m-r_+} où $r_+ \stackrel{\text{def}}{=} r_+(\mathcal{C}(I))$. D'un côté, r_+ correspond aussi au degré du monôme de plus haut degré de I, que l'on notera $f_{r_+} \in I$. Donc la distance minimale de $\mathcal{C}(I)$ est inférieure ou égale à $|\operatorname{Supp}(f_{r_+})| = 2^{m-r_+}$. D'un autre côté, on sait que $\mathcal{C}(I) \subseteq \mathcal{R}(r_+, m)$ (par définition de r_+) donc la distance minimale de $\mathcal{C}(I)$ est supérieure ou égale à la distance minimale de $\mathcal{R}(r_+, m)$, *i.e.* 2^{m-r_+} par le théorème 2.

(ii) Prouvons maintenant que $r_+(\mathcal{C}(I)^\perp) = m - 1 - r_-(\mathcal{C}(I))$.
Posons

$$\begin{cases} f_+(r) \stackrel{\text{def}}{=} x_0 \cdots x_{r-1} \\ f_-(r) \stackrel{\text{def}}{=} x_{m-r} \cdots x_{m-1} \end{cases}$$

et

$$\begin{cases} \mathcal{R}_+(r) \stackrel{\text{def}}{=} \{f_+(r') \ : \ 0 \leqslant r' \leqslant r\} \\ \mathcal{R}_-(r) \stackrel{\text{def}}{=} \{f_-(r') \ : \ 0 \leqslant r' \leqslant r\} \end{cases}$$

Par définition, $r_+(\mathcal{C}(I)) = \max\{r \ : \ \mathcal{R}_+(r) \subseteq I\}$, car rappelons que I est un ensemble décroissant donc, si pour r fixé, $x_0 \cdots x_{r-1} \in I$ alors, pour tout $r' \leqslant r$, $x_0 \cdots x_{r'-1} \preceq x_0 \cdots x_{r-1}$ (par divisibilité) donc $x_0 \cdots x_{r'-1} \in I$.

Comme $f_+(r_+(\mathcal{C}(I)))$ est le monôme de plus haut degré de $\mathcal{R}_+(r)$ appartenant à I, nous déduisons que $f_+(r_+(\mathcal{C}(I))+1)$ est le monôme de plus bas degré de $\mathcal{R}_+(r)$ appartenant à $\mathcal{M}\setminus I$.

Soit $g \stackrel{\text{def}}{=} f_+(r_+(\mathcal{C}(I))+1) \in \mathcal{M}\setminus I$. On a alors

$$\check{g} = \frac{x_0\cdots x_{m-1}}{x_0\cdots x_{r_+(\mathcal{C}(I))}} = x_{r_+(\mathcal{C}(I))+1}\cdots x_{m-1} = f_-(m-1-r_+(\mathcal{C}(I))) \in \mathcal{M}\setminus \check{I}$$

est le monôme de $\mathcal{R}_-$ de plus haut degré appartenant à $\mathcal{M}\setminus\check{I}$.

Ainsi, $r_-(\mathcal{C}(\mathcal{M}\setminus\check{I})) = m-1-r_+(\mathcal{C}(I))$.

Enfin, la propriété 47 nous donne $\mathcal{C}(\mathcal{M}\setminus\check{I}) = \mathcal{C}(I)^\perp$ et nous permet de conclure.

La deuxième égalité se déduit du fait que $(\mathcal{C}(I)^\perp)^\perp = \mathcal{C}(I)$ et de l'égalité que nous venons de prouver.

(iii) De (i) nous pouvons déduire que la distance minimale de $\mathcal{C}(I)^\perp$ est égale à $2^{m-r_+(\mathcal{C}(I)^\perp)}$. et de (ii) nous savons que $r_+(\mathcal{C}(I)^\perp) = m-1-r_-(\mathcal{C}(I))$. D'où le résultat.

□

Cette propriété nous permet de conclure que la distance minimale d'un code polaire est toujours plus petite ou égale à la distance minimale du code de Reed-Muller de même dimension (si ce code existe). Ceci fournit une forte indication sur le fait que cette distance minimale est plutôt faible.

Pour le code polaire auquel nous nous intéressons, nous avons été capable de trouver les mots de poids minimal dans le code et son dual en utilisant les algorithmes standards de décodage par ensemble d'informations (voir la sous-section 2.3.3).

Dans le cas des codes de Reed-Muller, il n'y a qu'une orbite selon le groupe de permutations à l'intérieur de l'ensemble des mots de poids minimal, ce qui n'est pas forcément le cas pour les codes monomiaux décroissants. Le théorème suivant permet de simplifier la caractérisation de ces mots de poids minimal.

Théorème 3. *Soit $I \subseteq \mathcal{M}$ un ensemble décroissant. Tout mot de poids minimal $\boldsymbol{c}_{min} \in \mathcal{C}(I)$ s'écrit $\boldsymbol{c}_{min} \stackrel{def}{=} \mathsf{ev}(f)^\pi$ où $f \in I$ et $\pi \in \mathbb{LTA}_m$.*

Démonstration. Soit $I \subseteq \mathcal{M}$ un ensemble décroissant et posons $r_+ \stackrel{\text{def}}{=} r_+(\mathcal{C}(I))$. La propriété 49 nous permet de savoir que tout mot de poids minimal de $\mathcal{C}(I)$ est aussi un mot de poids minimal de $\mathcal{R}(r_+,m)$. Le théorème 2 montre que tout mot de poids minimal $\boldsymbol{c}_{\min}$ de $\mathcal{R}(r_+,m)$ peut s'écrire comme l'évaluation du produit de r_+ formes affines indépendantes définies par :

$$x'_0 \stackrel{\text{def}}{=} \sum_j a_{0j}x_j + \varepsilon_0,\ \ x'_1 \stackrel{\text{def}}{=} \sum_j a_{1j}x_j + \varepsilon_1\ ,\ \ldots\ ,\ \ x'_{r_+-1} \stackrel{\text{def}}{=} \sum_j a_{(r_+-1)j}x_j + \varepsilon_{r_+-1}$$

où $\forall\ 0 \leqslant j \leqslant m$ et $\forall\ 0 \leqslant i \leqslant r_+-1$, $(a_{ij},\varepsilon_i) \in \mathbb{F}_2^2$.

Nous supposons maintenant qu'il existe r_+ formes affines indépendantes $x''_0,\ldots,x''_{r_+-1}$ telles que

(i) $\mathsf{ev}(x'_0\cdots x'_{r_+-1}) = \mathsf{ev}(x''_0\cdots x''_{r_+-1})$

(ii) $\forall\, 0 \leqslant i \leqslant r_+ - 1$ on a

$$x''_i = \sum_{j<\varphi(i)} a'_{\varphi(i),j} x_j + \varepsilon'_i$$

où $\varphi \in S_m$ et $(a'_{ij}, \varepsilon'_i) \in \mathbb{F}_2^2$.

Ceci est facile à vérifier en utilisant la forme affine x'_i qui implique la "plus grande" variable x_j, c'est-à-dire la variable dont l'indice est le plus grand. Nous notons x_{j_0} cette variable et considérons, sans perte de généralités, que $x'_0 = x_{j_0}$. Nous pouvons maintenant écrire :

$$\mathsf{ev}(x'_0 x'_1 \cdots x'_{r_+-1}) = \mathsf{ev}(x'_0 x'''_1 \cdots x'''_{r_+-1})$$

où $x'''_i = x'_i - x'_0 - 1$ si x'_i implique la variable x_{j_0} et $x'''_i = x'_i$ sinon.

Nous pouvons remarquer que les formes affines $x'''_1, \ldots, x'''_{r_+-1}$ n'impliquent que des variables x_j telles que $j < j_0$.

Ce procédé peut être répété pour les formes affines indépendantes $x'''_1, \ldots, x'''_{r_+-1}$ en considérant la variable x_j qui est la plus grande parmi toutes les variables impliquées dans ces formes. Nous pouvons donc, en réitérant ce procédé, obtenir les r_+ formes affines $x''_0, \ldots, x''_{r_+-1}$ qui vérifient les conditions (i) et (ii).

Considérons maintenant le monôme $x_{j_0} \cdots x_{j_{r_+-1}}$ formé du produit des variables x_j les plus grandes de chacun de ces x''_i. Ce monôme appartient à I et $\mathsf{ev}(x''_0 \cdots x''_{r_+-1}) = \mathsf{ev}(\pi(x_{j_0} \cdots x_{j_{r_+-1}}))$ pour $\pi \in \mathbb{LTA}_m$. □

En d'autres termes, les mots de poids minimal d'un code monomial décroissant peuvent être décrits comme les évaluations des polynômes dans l'orbite des monômes de base qui ont le plus haut degré (dans l'ensemble des monômes de base).

11 Cryptanalyse du schéma de McEliece proposé dans [SK14]

Dans un premier temps, nous allons expliciter les propriétés que nous utiliserons pour résoudre le problème d'équivalence de codes pour les codes polaires utilisés dans le cryptosystème de McEliece proposé dans [SK14].

Rappelons que nous considérons le code polaire $\mathcal{C}$ de longueur $n = 2048 = 2^{11}$ et de dimension 614 et son permuté $\mathcal{C}^\pi$ où $\pi \in S_{2048}$.

Ce code peut être vu comme un code monomial décroissant $\mathcal{C}(I)$, où

$$I \subseteq \mathcal{M} \stackrel{\text{def}}{=} \{x_0, x_1, \ldots, x_{10}, x_0x_1, \ldots, x_0 \cdots x_{10}\}$$

est un ensemble décroissant de monômes.

Premièrement nous pouvons calculer la distance minimale de ces codes grâce à la propriété 49. Dans ce cas, nous obtenons $r_+ \stackrel{\text{def}}{=} r_+(\mathcal{C}(I)) = 6$ et $r_- \stackrel{\text{def}}{=} r_-(\mathcal{C}(I)) = 2$, donc la distance minimale de $\mathcal{C}(I)$ est $2^{m-r_+} = 2^5 = 32$ et la distance minimale de $\mathcal{C}(I)^\perp$ est $2^{r_-+1} = 2^3 = 8$.

Comme la distance minimale du code considéré est plutôt faible, nous avons de bonnes chances de trouver l'ensemble W_{min} des mots de poids minimal de $\mathcal{C}(I)$ et l'ensemble des mots de poids minimal W_{min}^π de $\mathcal{C}(I)^\pi$ grâce aux algorithmes de décodage par ensemble d'informations classiques (voir la sous-section 2.3.3). Dans la suite, nous considérerons

$$W_{\text{min}} \stackrel{\text{def}}{=} \{\boldsymbol{c} \in \mathcal{C}(I) \ : \ |\boldsymbol{c}| = 2^{m-r_+}\}$$

$$W_{\text{min}}^\pi \stackrel{\text{def}}{=} \{\boldsymbol{c} \in \mathcal{C}(I)^\pi \ : \ |\boldsymbol{c}| = 2^{m-r_+}\}$$

De plus, le mot de code défini par $\boldsymbol{c}_{\text{min}} \stackrel{\text{def}}{=} \mathsf{ev}(x_0 \cdots x_{r_+-1})$ appartient à W_{min} puisque le code considéré est un code monomial décroissant de distance minimale 2^{m-r_+}. Le théorème 3, nous permet de déduire qu'il existe un monôme de degré r_+ dans l'ensemble des monômes de base de ce code. Puisque ce code est décroissant, l'ensemble des monômes de base est un ensemble décroissant et comme $x_0x_1 \cdots x_{r_+-1}$ est le plus petit monôme de degré r_+, il appartient aussi à cet ensemble et donc au code.

Deuxièmement, la connaissance de l'action du groupe de permutations du code polaire sur ces mots de poids minimal, va nous permettre de déterminer l'orbite de $\boldsymbol{c}_{\min}$ et de $\boldsymbol{c}^{\pi}_{\min}$.

En effet, d'une part, la propriété 44 nous permet de déterminer le groupe de permutations de $\mathcal{C}(I)^{\pi}$ en fonction du groupe de permutations de $\mathcal{C}(I)$ et d'autre part, le théorème 3 nous permet de décrire les mots de poids minimal de $\mathcal{C}(I)^{\pi}$ en fonction de l'action de son groupe de permutations sur les monômes de base.

Propriété 50. *Soient $\boldsymbol{c}_{min} \overset{def}{=} \mathsf{ev}(x_0 \cdots x_{r_+-1})$ un mot de poids minimal de $\mathcal{C}(I)$ un code monomial décroissant et $\pi \in S_n$.*

*(*i*)* $\mathcal{O}^{\mathbb{LTA}_m}_{\boldsymbol{c}_{min}} = \Big\{ \prod_{i=0}^{r_+-1} (x_i + \varepsilon_i) \ : \ \forall\, 0 \leqslant i \leqslant r_+ - 1,\ \varepsilon_i \in \mathbb{F}_2 \Big\}$

*(*ii*)* $\mathcal{O}^{\pi^{-1}\mathbb{LTA}_m\pi}_{\boldsymbol{c}^{\pi}_{min}} = \Big\{ g^{\pi} \ : \ g \in \mathcal{O}^{\mathbb{LTA}_m}_{\boldsymbol{c}_{min}} \Big\}$

Démonstration. Soient $\boldsymbol{c}_{\min} \overset{\text{def}}{=} \mathsf{ev}(x_0 \cdots x_{r_+-1}) \in \mathcal{C}(I)$ un code monomial décroissant et $\pi \in \mathbb{LTA}_m$. L'image par π du monôme $x_0 \cdots x_{r_+-1}$ est de la forme $x'_0 \cdots x'_{r_+-1}$ où $x'_i = \sum_{j<i} a_{ij} + \varepsilon_i$ pour $0 \leqslant i \leqslant x_{r_+-1}$ avec a_{ij} et ε_i dans $\mathbb{F}_2$.

Le support de chacun de ces polynômes est défini par l'espace affine $x'_0 = 1,\ \ldots\ ,\ x'_{r_+-1} = 1$. Cet espace affine peut aussi être vu comme $x_0 = \varepsilon'_0,\ \ldots\ ,\ x_{r_+-1} = \varepsilon'_{r_+-1}$ où les $\varepsilon'_i \in \mathbb{F}_2$.

Autrement dit, l'orbite de $\boldsymbol{c}_{\min}$ sous l'action de $\mathbb{LTA}_m$ sont bien issus des évaluations de monôme, dont les supports sont définis comme ci-dessus.

Pour trouver l'orbite de $\boldsymbol{c}^{\pi}_{\min}$ sous l'action de $\pi^{-1}\mathbb{LTA}_m\pi$, il suffit de remarquer que $\forall \sigma \in \mathbb{LTA}_m$, $(\boldsymbol{c}^{\pi}_{\min})^{\pi^{-1}\sigma\pi} = (\boldsymbol{c}^{\sigma}_{\min})^{\pi}$. □

Remarque 54. *On a donc* $|\mathcal{O}^{\mathbb{LTA}_m}_{\boldsymbol{c}_{min}}| = |\mathcal{O}^{\pi^{-1}\mathbb{LTA}_m\pi}_{\boldsymbol{c}^{\pi}_{min}}| = 2^{r_+}$.

De plus, les éléments de l'orbite de $\boldsymbol{c}_{\min}$ (respectivement $\boldsymbol{c}^{\pi}_{\min}$) ont un support disjoint. En effet, notons $A(\varepsilon_0, \ldots, \varepsilon_{r_+-1})$ l'espace affine défini par $x_0 = \varepsilon_0, x_1 = \varepsilon_1, \ldots, x_{r_+-1} = \varepsilon_{r_+-1}$. Ces espaces affines correspondent au support des mots de code obtenus comme l'évaluation des monômes de l'orbite de $\boldsymbol{c}_{\min}$. En considérant les espaces affines $A(\varepsilon_0, \ldots, \varepsilon_{r_+-1})$, pour $(\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{r_+-1}) \in \mathbb{F}_2^{r_+}$ nous pouvons remarquer que ceux-ci forment des ensembles disjoints. Les supports des mots de l'orbite de $\boldsymbol{c}_{\min}$ sont donc aussi disjoints. En appliquant le même raisonnement avec $\boldsymbol{c}^{\pi}_{\min}$, nous obtenons que les supports des éléments de l'orbite de $\boldsymbol{c}^{\pi}_{\min}$ sont aussi disjoints et correspondent aux versions permutées $A(\varepsilon_0, \ldots, \varepsilon_{r_+-1})^{\pi} \overset{\text{def}}{=} \{\pi^{-1}(i) \ : \ i \in A(\varepsilon_0, \ldots, \varepsilon_{r_+-1})\}$ des espaces affines $A(\varepsilon_0, \ldots, \varepsilon_{r_+-1})$.

L'idée utilisée dans cette attaque est de construire une signature qui nous permettrait d'identifier ces espaces affines, de sorte à résoudre une équivalence de codes sur les positions qu'ils définissent et de reconstruire la permutation sur toutes les positions du code pas à pas.

Dans un premier temps, nous aurons donc besoin de distinguer l'orbite de $\boldsymbol{c}_{\min}$ et $\boldsymbol{c}^{\pi}_{\min}$. Pour ce faire, nous allons introduire la notion de signature pour un ensemble de mots de code (qui forment une orbite).

Définition 50 (Signature pour une orbite)**.** *Soient $\mathcal{C}$ un code linéaire de longueur n et $\mathcal{G}$ un sous groupe du groupe de permutations de $\mathcal{C}$.*

*Notons W un sous-ensemble de $\mathcal{C}$ globalement invariant par $\mathcal{G}$ (*i.e. $\forall \boldsymbol{c} \in W$, $\forall \sigma \in \mathcal{G}$, $\boldsymbol{c}^\sigma \in W$*).*

On dit que la fonction $\Sigma(\boldsymbol{c}, \mathcal{C})$ où $\boldsymbol{c} \in \mathcal{C}$ est une signature pour l'action de $\mathcal{G}$ sur W si, et seulement si,

*(*i) $\Sigma(\boldsymbol{c}, \mathcal{C}) = \Sigma(\boldsymbol{c}^\pi, \mathcal{C}^\pi)$ *pour* $\pi \in S_n$ *(*i.e. Σ *est invariant par permutation),*

*(*ii) *si* $\boldsymbol{c}, \boldsymbol{c}' \in W$ *et* $\boldsymbol{c}' \notin \mathcal{O}_{\boldsymbol{c}}^{\mathcal{G}}$ *alors* $\Sigma(\boldsymbol{c}, \mathcal{C}) \neq \Sigma(\boldsymbol{c}', \mathcal{C})$

Remarque 55. *Avec cette définition de signature, les mots d'une orbite selon $\mathcal{G}$ auront la même signature. En effet, pour $\boldsymbol{c} \in W$ et $\gamma \in \mathcal{G}$ alors $\Sigma(\boldsymbol{c}, \mathcal{C}) = \Sigma(\boldsymbol{c}^\gamma, \mathcal{C}^\gamma) = \Sigma(\boldsymbol{c}^\gamma, \mathcal{C})$.*

11.1 Description de l'attaque

11.1.1 Recherche des mots de poids minimal

La distance minimale du code considéré étant assez faible, les mots de poids minimal de $\mathcal{C}(I)^\pi$ peuvent être trouvés en utilisant l'algorithme de Dumer [Dum91] ou tout autre algorithme de décodage par ensemble d'informations. D'un autre côté, tous les mots de poids minimal de $\mathcal{C}(I)$ peuvent être obtenus grâce au théorème 3. En effet, les mots de W_{min} se décomposent en orbites selon l'action de $\mathbb{LTA}_m$ où chaque orbite contient un des monômes de I de degré r_+.

11.1.2 Signature des orbites dans l'ensemble des mots de poids minimal du code

Dans la sous-section 10.3.3, nous avons introduit le théorème suivant, qui nous permet de décrire l'ensemble des mots de poids faible d'un code monomial décroissant.

Théorème 3. *Soit $I \subseteq \mathcal{M}$ un ensemble décroissant. Tout mot de poids minimal $\boldsymbol{c}_{min} \in \mathcal{C}(I)$ s'écrit $\boldsymbol{c}_{min} \overset{def}{=} \mathsf{ev}(f)^\pi$ où $f \in I$ et $\pi \in \mathbb{LTA}_m$.*

Pour distinguer les mots de code à l'intérieur de W_{min}, il faut dans un premier temps choisir un monôme dans chacune des orbites selon $\mathbb{LTA}_m$ (qui forment une partition de W_{min}). Nous noterons ces monômes $g_{\min}$ *i.e* on a $g_{\min} \in I$ et $\mathsf{ev}(g_{\min}) \in W_{\text{min}}$. Pour chacun de ces monômes $g_{\min}$, nous avons calculé le dual du code raccourci $S \overset{\text{def}}{=} \left(\mathcal{S}_{\mathcal{J}}\left(\mathcal{C}(I)\right)\right)^\perp$ où $\mathcal{J}$ correspond au support de $\mathsf{ev}(g_{\min})$.

Il est apparu que, pour le code polaire considéré, la donnée du nombre de mots de code de poids 2^{r_-} de S et la dimension de S est assez discriminante pour former une signature de l'orbite.

Théorème 4. *Soit $g = x_{i_1} \dots x_{i_{r_+}}$ un monôme de degré r_+ dans I. La distance minimale de $S \overset{def}{=} \left(\mathcal{S}_{\text{Supp}(g)}\left(\mathcal{C}(I)\right)\right)^\perp$ est égale à 2^{r_-} si, et seulement si, il existe un monôme $h \in \mathcal{M} \setminus \check{I}$ tel que :*

*(*i) *le nombre de variables de h qui sont aussi des variables de g est égal à $r_+ - 1$,*

*(*ii) *le nombre de variables de h qui sont aussi des variables de $\check{g}$ est $m - r_- - r_+$.*

Pour prouver ce théorème, nous aurons besoin des résultats suivants. Premièrement, nous introduirons la notation suivante :

$$E(\mathcal{S}_J(\mathcal{C})) \stackrel{\text{def}}{=} \{(c_i)_{0\leqslant i\leqslant n-1} \ : \ (c_j)_{j\notin J} \in \mathcal{S}_J(\mathcal{C}) \text{ et } c_j = 0 \text{ pour } j \in J\}$$

Autrement dit, $E(\mathcal{S}_J(\mathcal{C}))$ correspond au raccourci de $\mathcal{C}$ sur J que l'on a étendu aux positions sur lesquelles le code a été raccourci, en ajoutant des zéros.

Lemme 3. *Soient $I \subseteq \mathcal{M}$ un ensemble décroissant et $g \in I$. On a*

$$E(\mathcal{S}_{\mathrm{Supp}(g)}(\mathcal{C}(I)))^\perp = \{\mathsf{ev}((1+g)f) \ : \ f \in \mathcal{M} \setminus \check{I}\}$$

Démonstration. Rappelons que

$$(\mathcal{S}_{\mathrm{Supp}(g)}(\mathcal{C}(I)))^\perp = \mathcal{P}_{\mathrm{Supp}(g)}\left(\mathcal{C}(I)^\perp\right)$$

(voir la propriété 4). De plus, on a $\mathcal{C}(I)^\perp = \mathcal{C}(\mathcal{M} \setminus \check{I})$. Remarquons aussi que le support de $\mathsf{ev}(1+g)$ correspond au complémentaire du support de $\mathsf{ev}(g)$. D'où le résultat. □

Dans la suite, pour $g = x_{i_1}\cdots x_{i_s} \in \mathcal{M}$, $\mathrm{Ind}(g) \stackrel{\text{def}}{=} \{i_1, \ldots, i_s\}$ désigne l'ensemble de ses indices. De plus, pour $h \in \mathcal{M}$, nous noterons

$$g \wedge h \stackrel{\text{def}}{=} \prod_{i\in \mathrm{Ind}(g)\cap \mathrm{Ind}(h)} x_i$$

En d'autres termes, $g \wedge h$ est le plus grand diviseur commun de h et g.

Lemme 4. *Soient $g \in \mathcal{M}$ de degré $s \geqslant 1$ et $I \subseteq \mathcal{M}$ un ensemble décroissant. La distance minimale de $(\mathcal{S}_{\mathrm{Supp}(g)}(\mathcal{C}(I)))^\perp$ est $\geqslant 2^{r_-}$. De plus, si sa distance minimale est égale à 2^{r_-} alors $\exists\, h \in \mathcal{M} \setminus \check{I}$ tel que*

(i) *le nombre de variables de $h \wedge g$ est égal à $s-1$,*

(ii) *le nombre de variables de $h \wedge \check{g}$ est égal à $m - r_- - s$.*

Démonstration. Soit $\boldsymbol{c} \stackrel{\text{def}}{=} \mathsf{ev}(f) \in \mathcal{C}(I)^\perp$ un mot de code non nul tel que $f \in \mathbb{R}_2[x_0, \ldots, x_{m-1}]$. Le degré de f est donc au plus $m-1-r_-$ (par la propriété 49) et nous supposerons que $f = \sum_j m_j$ où $\forall j, m_j \in \mathcal{M}$.

Soit $g \in \mathbb{R}_2[x_0, \ldots, x_{m-1}]$, nous définissons $f_g \stackrel{\text{def}}{=} \sum_{j \,:\, g \text{ ne divise pas } m_j} m_j$. Autrement dit, f_g est le polynôme défini comme la somme des monômes de f qui ne sont pas divisibles par g. Comme $(\mathcal{S}_{\mathrm{Supp}(g)}(\mathcal{C}(I)))^\perp = \mathcal{P}_{\mathrm{Supp}(g)}\left(\mathcal{C}(I)^\perp\right)$ nous voulons montrer que l'évaluation de f sur $\mathbb{F}_2^n \setminus \mathrm{Supp}(g)$ est soit nulle, soit de poids $\leqslant 2^{r_-}$. De plus, nous pouvons remarquer que, sur $\mathbb{F}_2^n \setminus \mathrm{Supp}(g)$, l'évaluation de f est égale à celle de f_g.

Supposons que $g = x_0 \cdots x_{s-1}$ et choisissons f_g de degré maximum en $x_s, \ldots, x_{m-1}$ que l'on notera d. Dans ce cas,

$$f_g = m' \cdot u(x_0, \ldots, x_{s-1}) + v(x_0, \ldots, x_{m-1})$$

où m' est un monôme de degré d en $x_s, \ldots, x_{m-1}$, $u \stackrel{\text{def}}{=} \sum_{j\,:\,m' \text{ divise } m_j} m_j$ (rappelons que $f = \sum_j m_j$) et v est constitué du reste des monômes de f_g. Soit d' le degré de u alors $d' \leqslant s$ puisque f_g ne contient aucun monôme divisible par g et que u est formée des variables x_i pour $i \leqslant s-1$.

Remarquons que l'évaluation de $u(x_0, \ldots, x_{s-1})$ est non nulle sur au moins $2^{s-d'} - 1$ positions (si l'on ne compte pas la position qui correspond à $(1, \ldots, 1)$, puisque $\mathsf{ev}(u) \in \mathcal{R}(d', s)$ et que la distance minimale de $\mathcal{R}(d', s)$ est $2^{s-d'}$. Dans la suite, nous désignerons par "bloc", un ensemble de points $(x_0, \ldots, x_{m-1})$ dont la valeur sur $x_0, \ldots, x_{s-1}$ est fixée. Le support de g correspond au bloc défini par $x_0 = 1, \ldots, x_{s-1} = 1$. Dans ce cas, le poids de la restriction de $\mathsf{ev}(f_g)$ à n'importe lequel des blocs, sauf $x_0 = 1, \ldots, x_{s-1} = 1$, est au moins 2^{m-s-d}, puisque cette restriction correspond à un mot de code de $\mathcal{R}(d, m-s)$. En d'autres termes,

$$|\operatorname{Supp}(f_g(1+g))| \geqslant 2^{m-s-d}(2^{s-d'} - 1) \geqslant 2^{m-d-d'-1}$$

Comme, par définition, $d + d' \leqslant m - r_- - 1$ nous obtenons finalement

$$|\operatorname{Supp}(f_g)| \leqslant 2^{m-(m-r_--1)-1} = 2^{r_-}$$

ce qui nous permet de déduire que, dans le cas où $g = x_0 \cdots x_{s-1}$, la distance minimale de $(\mathcal{S}_{\operatorname{Supp}(g)}(\mathcal{C}(I)))^\perp$ est supérieure ou égale à 2^{r_-}. Il nous suffit à présent de remarquer que la seule condition que nous avons utilisée sur g est $g \neq 1$. Nous pouvons donc déduire que cet énoncé reste vrai pour tout monôme g de degré s.

Supposons maintenant que la distance minimale de $(\mathcal{S}_{\operatorname{Supp}(g)}(\mathcal{C}(I)))^\perp$ est égale à 2^{r_-}. En reprenant les notations précédentes, nous pouvons voir que ceci implique que $\deg(u) = s-1$ et $\deg(m') = m - r_- - 1 - (s-1) = m - r_- - s$. Nous pouvons aussi écrire u comme une somme de monômes, *i.e.* $u = \sum_j n'_j$ et choisir $n' = n'_j \in \mathcal{M}$ tel que $\deg(n') = s-1$. Dans ce cas, $h \stackrel{\text{def}}{=} m'n'$ est un monôme de degré $(s-1) + (m - r_- - s) = m - r_- - 1$ tel que $\exists\, j, m_j = h$ avec $f = \sum_j m_j$. Autrement dit, $h \in \mathcal{M} \setminus \check{I}$ et son degré est égal à $m - r_- - 1$.

□

Nous pouvons maintenant donner la preuve du théorème 4.

Démonstration : théorème 4. Dans un premier temps, nous pouvons remarquer que, par définition, la distance minimale de $\mathcal{E} \stackrel{\text{def}}{=} E(\mathcal{S}_{\operatorname{Supp}(g)}(\mathcal{C}(I)))^\perp$ est égale à la distance minimale de $\mathcal{S}_{\operatorname{Supp}(g)}(\mathcal{C}(I))^\perp$. De plus, le lemme 3 nous permet de déduire que tout mot de code de $\mathcal{E}$ peut s'écrire comme $\mathsf{ev}((1+g)f)$ où f est un polynôme formé d'une combinaison linéaire de monômes de $\mathcal{M} \setminus \check{I}$. Considérons maintenant qu'il existe $h \in \mathcal{M} \setminus \check{I}$ qui satisfait les conditions du théorème et prouvons que le poids de $\mathsf{ev}((1+g)h)$ est égal à 2^{r_-}.

Soit $i_0 \in \operatorname{Ind}(g) \setminus \operatorname{Ind}(g \wedge h)$, on a

$$\begin{aligned}
(1+g)h &= (1 + x_{i_1} \cdots x_{i_r})\Big(\prod_{i \in \operatorname{Ind}(g \wedge h)} x_i\Big)\Big(\prod_{i \in \operatorname{Ind}(\check{g} \wedge h)} x_i\Big) \\
&= (1 + x_{i_0})\Big(\prod_{i \in \operatorname{Ind}(g \wedge h)} x_i\Big)\Big(\prod_{i \in \operatorname{Ind}(\check{g} \wedge h)} x_i\Big) \\
&= (1 + x_{i_0})h
\end{aligned}$$

Ainsi,

$$|\operatorname{Supp}((1+g)h)| = \operatorname{Supp}((1+x_{i_0})h) = 2^{m-(m-r_--1r=+0)} = 2^{r_-}$$

Nous pouvons en déduire, en utilisant la borne inférieure donnée par le lemme 4 que la distance minimale de $\mathcal{S}_{\operatorname{Supp}(g)}\left(\mathcal{C}(I)\right)^{\perp}$ est égale à 2^{r_-}. Ainsi, nous pouvons utiliser les résultats du lemme 4 pour conclure que h est bien de la forme souhaitée.

□

11.1.3 Calcul des orbites dans l'ensemble des mots de poids minimal du code permuté

La signature Σ, explicitée dans l'étape précédente, est maintenant appliquée à W_{min}^{π}. Ceci permet de trouver les orbites de W_{min}^{π} par rapport au groupe de permutations $\pi^{-1}\mathcal{G}\pi$.

Propriété 51. *W_{min}^{π} est invariant sous l'action de $\pi^{-1}\mathbb{LTA}_m\pi$ et si Σ est une signature pour W_{min} sous l'action de $\mathbb{LTA}_m$, alors c'est aussi une signature sous l'action de $\pi^{-1}\mathbb{LTA}_m\pi$ pour W_{min}^{π}.*

Démonstration. Premièrement, le fait que W_{min}^{π} est invariant vient de :

(i) $\mathbb{LTA}_m$ est un sous groupe du groupe de permutations de $\mathcal{C}(I)$ (par le théorème 1),

(ii) $\pi^{-1}\mathbb{LTA}_m\pi$ est donc un sous groupe du groupe de permutations de $\mathcal{C}(I)^{\pi}$ (de (i) et la propriété 44)

La seconde partie se déduit du fait que Σ prend différentes valeurs suivant l'orbite (selon $\pi^{-1}\mathbb{LTA}_m\pi$) de W_{min}^{π}. Considérons $\boldsymbol{x}^{\pi}$ et $\boldsymbol{y}^{\pi}$ deux éléments appartenant à deux orbites distinctes de W_{min}^{π}. Par définition, ce sont deux versions permutées de $\boldsymbol{x}$ et $\boldsymbol{y}$ dont les orbites (dans W_{min}) sont aussi distinctes. En effet, dans le cas contraire, il existerait $\sigma \in \mathbb{LTA}_m$ telle que $\boldsymbol{x} = \boldsymbol{y}^{\sigma}$ donc ceci impliquerait que $\boldsymbol{x}^{\pi} = \boldsymbol{y}^{\sigma\pi} = \boldsymbol{y}^{\pi\pi^{-1}\sigma\pi} = (\boldsymbol{y}^{\pi})^{\pi^{-1}\sigma\pi}$. En d'autres termes, $\boldsymbol{x}^{\pi}$ et $\boldsymbol{y}^{\pi}$ seraient dans la même orbite selon $\pi^{-1}\mathbb{LTA}_m\pi$.

La dernière partie de la preuve se déduit de

$$\begin{aligned}\Sigma(\boldsymbol{x}^{\pi}, \mathcal{C}(I)^{\pi}) &= \Sigma(\boldsymbol{x}, \mathcal{C}(I))\\ \Sigma(\boldsymbol{y}^{\pi}, \mathcal{C}(I)^{\pi}) &= \Sigma(\boldsymbol{y}, \mathcal{C}(I))\end{aligned}$$

□

Nous utiliserons donc cette signature pour trouver les orbites de $\boldsymbol{c}_{\text{min}}^{\pi}$. En d'autres termes, trouver les orbites dans W_{min}^{π} et considérer le support des mots de code que nous avons trouvé de cette manière, nous permet de décrire le support des versions permutées $A(\varepsilon_0, \ldots, \varepsilon_{r_+-1})^{\pi}$ des espaces affines $A(\varepsilon_0, \ldots, \varepsilon_{r_+-1})$.

11.1.4 Identification des espaces affines

Il existe plusieurs techniques pour identifier les versions permutées des espaces affines considérés. Une des façons les plus simples, qui fonctionne pour le code polaire de longueur 2048 et de dimension 614 étudié, est de calculer la dimension d'espaces particuliers.

Soient $\mathcal{I}_0$ le support de $\boldsymbol{c}_{\min}$ et $\mathcal{I}'_0$ le support du mot de code $\mathsf{ev}\left(x_0x_1\cdots x_{r_+-2}(x_{r_+-1}-1)\right)$ (qui est aussi un mot de poids minimal).

Remarque 56. *$\mathcal{I} \overset{def}{=} \mathcal{I}_0 \cup \mathcal{I}'_0$ est le support du mot de code* $\mathsf{ev}(x_0\cdots x_{r_+-2})$.

Ensuite, nous calculons la dimension du code $\mathcal{P}_{\mathcal{I}}\left(\mathcal{C}(I)\right)$.

Maintenant, considérons tous les mots de code dans l'orbite de $\boldsymbol{c}^{\pi}_{\min}$. Ceux-ci sont de la forme $\boldsymbol{c}^{\pi\gamma}_{\min}$ où γ est une permutation qui laisse $\mathcal{C}(I)$ invariant. En d'autres termes, à action du groupe de permutations près, nous pouvons désigner n'importe lequel de ces mots de code comme $\boldsymbol{c}^{\pi}_{\min}$. Soient $\mathcal{J}_0,\ldots,\mathcal{J}_{2^{r_+}-1}$ les supports des mots de code qui appartiennent à l'orbite de $\boldsymbol{c}^{\pi}_{\min}$, avec $\mathcal{J}_0$ correspondant au support du mot de code $\boldsymbol{c}^{\pi}_{\min}$. Nous calculons les dimensions du code $\mathcal{P}_{\mathcal{J}_0\cup\mathcal{J}_i}\left(\mathcal{C}(I)^{\pi}\right)$ pour $1 \leqslant i \leqslant 2^{r_+}-1$. Il s'avère qu'il n'existe généralement qu'un seul espace $\mathcal{J}_i$ tel que $\dim\left(\mathcal{P}_{\mathcal{J}_0\cup\mathcal{J}_i}\left(\mathcal{C}(I)^{\pi}\right)\right) = \dim\left(\mathcal{P}_{\mathcal{I}}\left(\mathcal{C}(I)\right)\right)$.

Pour γ un élément du groupe de permutations de $\mathcal{C}(I)$, nous remarquons que ce procédé peut être employé afin de former les couples de tous les espaces $A(\varepsilon_0,\ldots,\varepsilon_{r_+-2},0)^{\gamma\pi}$ et $A(\varepsilon_0,\ldots,\varepsilon_{r_+-2},1)^{\gamma\pi}$ en associant les ensembles $\mathcal{J}_i$ et $\mathcal{J}_j$ quand $\mathcal{J}_j$ est le seul ensemble qui, pour un i donné, est tel que

$$\dim\left(\mathcal{P}_{\mathcal{J}_i\cup\mathcal{J}_j}\left(\mathcal{C}(I)^{\pi}\right)\right) = \dim\left(\mathcal{P}_{\mathcal{I}}\left(\mathcal{C}(I)\right)\right).$$

Dans ce cas, $\mathcal{J}_i$ et $\mathcal{J}_j$ correspondent nécessairement à $A(\varepsilon_0,\ldots,\varepsilon_{r_+-2},0)^{\gamma\pi}$ et $A(\varepsilon_0,\ldots,\varepsilon_{r_+-2},1)^{\gamma\pi}$ pour un certain $(\varepsilon_0,\ldots,\varepsilon_{r_+-2}) \in \mathbb{F}_2^{r_+-1}$.

En d'autres termes, après cette étape, les espaces $A(\varepsilon_0,\ldots,\varepsilon_{r_+-2})^{\gamma\pi} = A(\varepsilon_0,\ldots,\varepsilon_{r_+-2},0)^{\gamma\pi} \cup A(\varepsilon_0,\ldots,\varepsilon_{r_+-2},1)^{\gamma\pi}$ sont connus.

Ce procédé est aussi applicable aux mots de code $\boldsymbol{c} = \mathsf{ev}(x_0\cdots x_{r_+-1})$ (au lieu de $\boldsymbol{c}_{\min}$) et permet ainsi de retrouver tous les espaces affines permutés

$$A(1)^{\gamma\pi}, A(1,1)^{\gamma\pi},\ldots,A(\underbrace{1,1,\ldots,1}_{r_+ \text{ fois}})^{\gamma\pi}$$

pour les permutations γ qui laissent $\mathcal{C}(I)$ invariant.

11.1.5 Problème d'équivalence de codes pour un petit code monomial décroissant

Soit $\mathcal{K}$ le complémentaire du support de $\boldsymbol{c}_{\min}$ *i.e* $\mathcal{K} \overset{\text{def}}{=} \{0,\ldots,n-1\} \setminus \mathcal{I}_0$. Nous pouvons remarquer que le complémentaire du support de $\boldsymbol{c}^{\pi}_{\min}$ est alors $\mathcal{K}^{\pi}$. Nous définissons $\mathcal{D} \overset{\text{def}}{=} \mathcal{P}_{\mathcal{K}}\left(\mathcal{C}\right)$ et $\mathcal{D}^{\pi} \overset{\text{def}}{=} \mathcal{P}_{\mathcal{K}^{\pi}}\left(\mathcal{C}^{\pi}\right)$.

Il faut maintenant résoudre l'équivalence de codes pour $\mathcal{D}$ qui est un code de longueur 2^{m-r_+}, c'est-à-dire pour un code de longueur beaucoup plus petite que le code considéré initialement.

Remarque 57. *$\mathcal{D}$ est aussi un code monomial décroissant.*

Une solution consiste à répéter le processus décrit précédemment. Pour le code polaire de longueur 2048 et de dimension 614 considéré, il est même possible de retrouver le groupe de permutations du

code en entier, qui est plus proche du groupe affine. Dans ce cas, c'est un code de longueur 32 qui contient le code de Reed-Muller $\mathcal{R}(2,5)$ et qui est contenu dans le code de Reed-Muller $\mathcal{R}(3,5)$.

Cette étape n'est pas détaillée ici, mais il existe de nombreuses manières de résoudre ce problème.

11.1.6 Étape d'induction

Ici, l'idée est de reconstruire la permutation $\hat{\pi} \in S_n$, sachant que nous connaissons son action sur le support de $\boldsymbol{c}_{\text{min}}$. Plus précisément, le problème d'équivalence de codes considéré peut être décrit comme ci-dessous.

Problème 9 (Équivalence de codes avec promesses - Recherche)**.**
Paramètres : $t, n \in \mathbb{N}$
Entrées : $(\mathcal{C}, \mathcal{C}')$ *un couple de codes linéaires de longueur* n *et* t *couples de positions* $((i_s, j_s))_{0 \leqslant s \leqslant t-1}$
Promesses : $\exists\, \pi \in S_n$ *telle que* $\mathcal{C}' = \mathcal{C}^\pi$ *et* $\forall s < t$, $\pi(i_s) = j_s$
Problème : *trouver* $\hat{\pi} \in S_n$ *telle que*

$$\mathcal{C}^{\hat{\pi}} = \mathcal{C}' \text{ et } \hat{\pi}(i_s) = j_s \ \forall s \in \{0, 1, \ldots, t-1\}$$

Soient $\boldsymbol{c}^i \stackrel{\text{def}}{=} \mathsf{ev}(x_0 \cdots x_{i-1})$ avec $\boldsymbol{c}^0 \stackrel{\text{def}}{=} \mathsf{ev}(1)$ par convention. Avec cette notation, on a $\boldsymbol{c}_{\text{min}} = \boldsymbol{c}^{r_+}$.

Pour résoudre ce problème, nous avons utilisé l'algorithme suivant.

Posons $\mathcal{I} \stackrel{\text{def}}{=} \{i_0, \ldots, i_{t-1}\}$ et $\mathcal{J} \stackrel{\text{def}}{=} \{j_0, \ldots, j_{t-1}\}$.

(i) Choisir ℓ mots de code de $\mathcal{C}$, que nous noterons $\boldsymbol{c}(0), \ldots, \boldsymbol{c}(\ell-1)$.

(ii) Soit $\mathcal{C}(j)$ l'ensemble des mots de code de $\mathcal{C}$ qui coïncident avec $\boldsymbol{c}(j)$ sur les positions qui appartiennent à $\mathcal{J}$ et soit $\mathcal{C}(i)^\pi$ l'ensemble des mots de code de $\mathcal{C}^\pi$ qui coïncident avec $\boldsymbol{c}(i)^\pi$ sur $\mathcal{I}$.

(iii) Calculer, pour tout $0 \leqslant i \leqslant \ell - 1$ et pour toute position j qui n'appartient pas à $\mathcal{J}$, le nombre $\Sigma(i,j)$ de mots de poids minimal dans $\mathcal{P}_j\left(\mathcal{C}(i)\right)$. De la même façon, calculer le nombre de mots de poids minimal dans $\mathcal{P}_j\left(\mathcal{C}(i)^\pi\right)$, que l'on notera $\Sigma^\pi(i,j)$, pour toutes les positions j qui ne sont pas dans $\mathcal{I}$.

(iv) On considère, pour u qui n'appartient pas à $\mathcal{I}$ que $\hat{\pi}(u) = v$ s'il existe un unique v qui n'appartient pas à $\mathcal{J}$ tel que $\Sigma(i,v) = \Sigma^\pi(i,u)$ pour tout $0 \leqslant i \leqslant \ell - 1$.

Dans ce contexte, l'algorithme renvoie bien une permutation $\hat{\pi}$ solution du problème 9. Nous avons aussi rencontré des cas où les informations dont nous disposons sur $\hat{\pi}$ nous permettent de trouver plusieurs solutions. Dans ce cas, nous avons pu déterminer le nombre de solutions possibles et ajouter les couples de positions déduits à la suite des (i_s, j_s) pour trouver une unique solution.

11.2 Implémentation de l'attaque

Nous avons implémenté cette attaque sur le code polaire $\mathcal{C}(I)$ de longueur 2048 et de dimension 614, avec $I \subseteq \{x_0, x_1, \ldots, x_{10}, x_0x_1, \ldots, x_0x_1 \cdots x_{10}\}$. La première étape fut de vérifier que ce code et son dual sont bien des codes monomiaux décroissants. Ensuite, nous avons pu déterminer l'ensemble des mots de poids minimal de ces codes en utilisant le théorème 3. De plus, comme, dans ce cas, les conditions du corollaire 1 sont respectées, nous avons pu vérifier que ce code était faiblement auto-dual, *i.e.* $\mathcal{C}(I) \subseteq \mathcal{C}(I)^{\perp}$.

La distance minimale de $\mathcal{C}(I)$ est ici égale à 32 et nous avons pu trouver les 42176 mots de ce poids dans ce code. La distance minimale de son dual, elle, est égale à 8 et celui-ci admet 6912 mots de poids minimal. Ainsi, les versions permutés de ces mots de poids minimal de $\mathcal{C}(I)^{\pi}$ et $(\mathcal{C}(I)^{\pi})^{\perp}$ ont pu être retrouvés très rapidement, grâce à l'algorithme de Dumer ([Dum91]). Plus précisément, en utilisant un processeur 8-core XEON E3-1240 de 3.40 GHz, nous avons trouvé tous les mots de poids minimal du code permuté en 27 secondes et ceux de son dual en 3 secondes.

L'étape la plus coûteuse de l'attaque est en fait l'étape d'induction, puisque nous avons dû calculer de nombreuses signatures. Bien que celles-ci furent aussi calculées grâce à l'algorithme de Dumer, dans ce cas le nombre de mots de poids fixe cherchés est inconnu. En effet, dans un cas, le théorème 3 nous permet de connaître, *a priori*, le nombre de mots de poids minimal dans le code considéré. Tandis que dans ce cas, nous voulons calculer le nombre de mots de poids minimal dans $\mathcal{P}_j\left(\mathcal{C}(i)\right)$ pour chaque couple de positions (i, j). Nous avons donc dû utiliser une procédure probabiliste basée sur le problème du collectionneur de coupons (ou *coupon collector problem*) : la recherche de mots de poids minimal n'est arrêtée qu'à partir du moment où nous avons trouvé N mots de poids minimal et que chaque mot apparaît environ $\alpha \ln N$ fois (avec $\alpha \geqslant 1$). Pour s'assurer d'avoir une bonne probabilité d'obtenir tous les mots de poids minimal, nous avons choisi d'utiliser cette approche avec $\alpha = 3$. Afin d'accélérer la recherche de ces mots de poids minimal, nous avons décidé d'utiliser les mots de poids minimal de $\mathcal{C}(I)$ pour fixer la valeur des $c(i)$.

En fin de compte, l'étape qui consiste à retrouver la permutation sur l'ensemble des 2048 positions du code, connaissant son action sur 1024 positions, compte pour plus de 80% du coût total de l'attaque. En effet, cette partie nécessite environ 227 heures de calcul, tandis que le coût total de l'attaque est d'environ 280 heures. Cependant, nous avons de bonnes raisons de penser que le coût de cette partie peut largement être diminuée si besoin.

Conclusions et perspectives

Les résultats exposés dans cette partie ont été publiés dans [BCD+16]. En premier lieu, nous pouvons conclure que l'utilisation des codes polaires pour le cryptosystème de McEliece, comme introduit dans [SK14], n'est pas sûre. En effet, nous avons explicité une méthode permettant de résoudre le problème d'équivalence de codes efficacement, pour certains paramètres et notamment ceux proposés dans [SK14]. Cette attaque repose sur plusieurs propriétés algébriques des codes polaires que nous avons mis en évidence. D'une part la présence de mots de poids faible dans le code et son dual, d'autre part, le fait que nous connaissons la structure de ces mots de poids faible et enfin que nous avons identifié l'action du groupe d'automorphisme sur ces mots.

En second lieu, nous avons défini une nouvelle famille de codes, les codes monomiaux décroissants, dont les codes polaires et les codes de Reed-Muller font partie. En plus de fournir de nouveaux résultats sur les propriétés algébriques de ces codes, ces résultats sont, de manière indépendante, pertinents dans le domaine de la théorie des codes. Récemment, [BDOT16] exhibent de nouvelles propriétés de cette famille de codes.

Cependant, il est important de noter que, bien que l'utilisation des codes polaires dans le cryptosystème de McEliece soit compromise, ces codes sont tout de même très intéressants pour la cryptographie. En effet, ils ont une très bonne capacité de correction et leurs algorithmes de décodage sont très peu coûteux.

Bibliographie

[Ale11] Michael Alekhnovich. More on average case vs approximation complexity. *Computational Complexity*, 20(4) :755–786, 2011.

[Ari09] Erdal Arikan. Channel polarization : a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans. Inform. Theory*, 55(7) :3051–3073, 2009.

[BBC08] Marco Baldi, Marco Bodrato, and Franco Chiaraluce. A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In *Proceedings of the 6th international conference on Security and Cryptography for Networks*, SCN '08, pages 246–262, Berlin, Heidelberg, 2008. Springer-Verlag.

[BC07] Marco Baldi and Franco Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2591–2595, Nice, France, June 2007.

[BCD+16] Magali Bardet, Julia Chaulet, Vlad Dragoi, Ayoub Otmani, and Jean-Pierre Tillich. Cryptanalysis of the mceliece public key cryptosystem based on polar codes. In *Post-Quantum Cryptography 2016*, pages 118–143, 2016.

[BCG06] Marco Baldi, Franco Chiaraluce, and Roberto Garello. On the usage of quasi-cyclic low-density parity-check codes in the McEliece cryptosystem. In *Proceedings of the First International Conference on Communication and Electronics (ICEE'06)*, pages 305–310, October 2006.

[BCGM07] Marco Baldi, Franco Chiaraluce, Roberto Garello, and Francesco Mininni. Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem. In *Communications, 2007. ICC '07. IEEE International Conference on*, pages 951–956, June 2007.

[BCGO09] Thierry P. Berger, Pierre-Louis Cayrel, Philippe Gaborit, and Ayoub Otmani. Reducing key length of the McEliece cryptosystem. In Bart Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Comput. Sci.*, pages 77–97, Gammarth, Tunisia, June 21-25 2009.

[BCS13] Daniel J. Bernstein, Tung Chou, and Peter Schwabe. Mcbits : Fast constant-time code-based cryptography. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *LNCS*, pages 250–272. Springer, 2013.

[BDOT16] Magali Bardet, Vlad Dragoi, Ayoub Otmani, and Jean-Pierre Tillich. Algebraic properties of polar codes from a new polynomial formalism. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2016*, pages 230–234. IEEE, 2016.

[Ber10] Daniel J. Bernstein. Grover vs. McEliece. In Nicolas Sendrier, editor, *Post-Quantum Cryptography 2010*, volume 6061 of *Lecture Notes in Comput. Sci.*, pages 73–80. Springer, 2010.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, Lecture Notes in Comput. Sci. Springer, 2012.

[BMvT78] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3) :384–386, May 1978.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymetric encryption. In A. De Santis, editor, *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *LNCS*, pages 92–111. Springer, 1995.

[CEvMS15] Cong Chen, Thomas Eisenbarth, Ingo von Maurich, and Rainer Steinwandt. Differential power analysis of a mceliece cryptosystem. In *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, pages 538–556, 2015.

[Cho16] Tung Chou. Qcbits : Constant-time small-key code-based cryptography. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 280–300, 2016.

[CMCP14] Alain Couvreur, Irene Márquez-Corbella, and Ruud Pellikaan. A polynomial time attack against algebraic geometry code based public key cryptosystems. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2014*, pages 1446–1450, June 2014.

[CS98] Anne Canteaut and Nicolas Sendrier. Cryptanalysis of the original McEliece cryptosystem. In *Advances in Cryptology - ASIACRYPT 1998*, volume 1514 of *Lecture Notes in Comput. Sci.*, pages 187–199. Springer, 1998.

[CS16] Julia Chaulet and Nicolas Sendrier. Worst case QC-MDPC decoder for mceliece cryptosystem. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 1366–1370, 2016.

[CT91] Thomas M. Cover and Joy A. Thomas. *Information Theory*. Wiley Series in Telecommunications. Wiley, 1991.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, 22(6) :644–654, November 1976.

[Dum91] Ilya Dumer. On minimum distance decoding of linear codes. In *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, pages 50–52, Moscow, 1991.

[FGO$^+$11] Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high rate McEliece cryptosystems. In *Proc. IEEE Inf. Theory Workshop- ITW 2011*, pages 282–286, Paraty, Brasil, October 2011.

[FL08] Pierre-Alain Fouque and Gaëtan Leurent. Cryptanalysis of a hash function based on quasi-cyclic codes. In *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008. Proceedings*, volume 4964 of *Lecture Notes in Comput. Sci.*, pages 19–35. Springer, 2008.

[FM08] Cédric Faure and Lorenz Minder. Cryptanalysis of the McEliece cryptosystem over hyperelliptic curves. In *Proceedings of the eleventh International Workshop on Algebraic and Combinatorial Coding Theory*, pages 99–107, Pamporovo, Bulgaria, June 2008.

[FOPS04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. *J. Cryptology*, 17(2) :81–104, 2004.

[FOPT10] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of McEliece variants with compact keys. In *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Comput. Sci.*, pages 279–298, 2010.

[Gab05] Philippe Gaborit. Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, Bergen, Norway, March 2005.

[Gal63] Robert G. Gallager. *Low Density Parity Check Codes*. M.I.T. Press, Cambridge, Massachusetts, 1963.

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. *IACR Cryptology ePrint Archive*, 2016 :858, 2016.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.

[HSEA14] R Hooshmand, M Koochak Shooshtari, T Eghlidos, and MR Aref. Reducing the key length of McEliece cryptosystem using polar codes. In *2014 11th International ISC Conference on Information Security and Cryptology (ISCISC)*, pages 104–108. IEEE, 2014.

[HvMG13] Stefan Heyse, Ingo von Maurich, and Tim Güneysu. Smaller keys for code-based cryptography : QC-MDPC McEliece implementations on embedded devices. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013*, volume 8086 of *Lecture Notes in Comput. Sci.*, pages 273–292. Springer, 2013.

[Jab01] Abdulrahman Al Jabri. A statistical decoding algorithm for general linear block codes. In Bahram Honary, editor, *Cryptography and coding. Proceedings of the 8th IMA International Conference*, volume 2260 of *Lecture Notes in Comput. Sci.*, pages 1–8, Cirencester, UK, December 2001. Springer.

[JM96] Heeralal Janwa and Oscar Moreno. McEliece public key cryptosystems using algebraic-geometric codes. *Des. Codes Cryptogr.*, 8(3) :293–307, 1996.

[Kac16] Ghazal Kachigar. Étude et conception d'algorithmes quantiques pour le décodage de codes linéaires . Master's thesis, Université de Rennes 1, France, September 2016.

[LB88] Pil J. Lee and Ernest F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Advances in Cryptology - EUROCRYPT'88*, volume 330 of *Lecture Notes in Comput. Sci.*, pages 275–280. Springer, 1988.

[LDW94] Yuan Xing Li, Robert H. Deng, and Xin Mei Wang. On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. *IEEE Trans. Inform. Theory*, 40(1) :271–273, 1994.

[LJ12] Carl Löndahl and Thomas Johansson. A new version of McEliece PKC based on convolutional codes. In *Information and Communications Security, ICICS*, volume 7168 of *Lecture Notes in Comput. Sci.*, pages 461–470. Springer, 2012.

[LJKS+16] Carl Löndahl, Thomas Johansson, Masoumeh Koochak Shooshtari, Mahmoud Ahmadian-Attari, and Mohammad Reza Aref. Squaring attacks on mceliece public-key cryptosystems using quasi-cyclic codes of even dimension. *Des. Codes Cryptography*, 80(2) :359–377, August 2016.

[LT13] Grégory Landais and Jean-Pierre Tillich. An efficient attack of a McEliece cryptosystem variant based on convolutional codes. In P. Gaborit, editor, *Post-Quantum Cryptography'13*, volume 7932 of *LNCS*, pages 102–117. Springer, June 2013.

[Mac95] David J. C. MacKay. Good codes based on very sparse matrices. In *Cryptography and Coding, 5th IMA Conference, Cirencester, UK, December 18-20, 1995, Proceedings*, pages 100–111, 1995.

[MB09] Rafael Misoczki and Paulo Barreto. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography*, Calgary, Canada, August 13-14 2009.

[McE78] Robert J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 107–124. Springer, 2011.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 203–228. Springer, 2015.

[MOG15] Ingo Von Maurich, Tobias Oder, and Tim Güneysu. Implementing QC-MDPC McEliece encryption. *ACM Trans. Embed. Comput. Syst.*, 14(3) :44 :1–44 :27, April 2015.

[MRAS00] Chris Monico, Joachim Rosenthal, and Amin A. Shokrollahi. Using low density parity check codes in the McEliece cryptosystem. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, page 215, Sorrento, Italy, 2000.

[MS07] Lorenz Minder and Amin Shokrollahi. Cryptanalysis of the Sidelnikov cryptosystem. In *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Comput. Sci.*, pages 347–360, Barcelona, Spain, 2007.

[MTSB12] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece : New McEliece variants from moderate density parity-check codes. *IACR Cryptology ePrint Archive, Report2012/409*, 2012, 2012.

[Nie86] Harald Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, 15(2) :159–166, 1986.

[OTD08] Ayoub Otmani, Jean-Pierre Tillich, and Léonard Dallot. Cryptanalysis of McEliece cryptosystem based on quasi-cyclic LDPC codes. In *Proceedings of First International Conference on Symbolic Computation and Cryptography*, pages 69–81, Beijing, China, April 28-30 2008. LMIB Beihang University.

[Ove06] Raphael Overbeck. Statistical decoding revisited. In Reihaneh Safavi-Naini Lynn Batten, editor, *Information security and privacy : 11th Australasian conference, ACISP 2006*, volume 4058 of *Lecture Notes in Comput. Sci.*, pages 283–294. Springer, 2006.

[PR97] Erez Petrank and RonM. Roth. Is code equivalence easy to decide ? *IEEE Trans. Inform. Theory*, 43(5) :1602–1604, 1997.

[Pra62] Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5) :5–9, 1962.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126, 1978.

[Sch72] J. Pieter M. Schalkwijk. An algorithm for source coding. *IEEE Trans. Information Theory*, 18(3) :395–399, 1972.

[Sen97] Nicolas Sendrier. On the dimension of the hull. In *SIAM J. Discrete Math.*, volume 10, pages 282–293, 1997.

[Sen00] Nicolas Sendrier. Finding the permutation between equivalent linear codes : The support splitting algorithm. *IEEE Trans. Inform. Theory*, 46(4) :1193–1203, 2000.

[Sen05] Nicolas Sendrier. Encoding information into constant weight words. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2005*, pages 435–438, 2005.

[Sen10] Nicolas Sendrier. On the use of structured codes in code based cryptography. In L. Storme S. Nikova, B. Preneel, editor, *Coding Theory and Cryptography III*, pages 59–68. The Royal Flemish Academy of Belgium for Science and the Arts, 2010.

[Sen11] Nicolas Sendrier. Decoding one out of many. In *Post-Quantum Cryptography 2011*, volume 7071 of *Lecture Notes in Comput. Sci.*, pages 51–67, 2011.

[Sho94] P.W. Shor. Algorithms for quantum computation : Discrete logarithms and factoring. In S. Goldwasser, editor, *FOCS*, pages 124–134, 1994.

[Sid94] Vladimir Michilovich Sidelnikov. A public-key cryptosytem based on Reed-Muller codes. *Discrete Math. Appl.*, 4(3) :191–207, 1994.

[SK14] Sujan Raj Shrestha and Young-Sik Kim. New McEliece cryptosystem based on polar codes as a candidate for post-quantum cryptography. In *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*, pages 368–372. IEEE, 2014.

[Ste88] Jacques Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Comput. Sci.*, pages 106–113. Springer, 1988.

[TS16] Rodolfo Canto Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, pages 144–161, 2016.

[vMG14a] Ingo von Maurich and Tim Güneysu. Lightweight code-based cryptography : QC-MDPC McEliece encryption on reconfigurable devices. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, March 24-28, 2014*, pages 1–6, 2014.

[vMG14b] Ingo von Maurich and Tim Güneysu. Towards side-channel resistant implementations of QC-MDPC mceliece encryption on constrained devices. In *Post-Quantum Cryptography 2014*, volume 8772 of *LNCS*, pages 266–282. Springer, 2014.

Table des figures

Code correcteur d'erreurs **17**

1.1 Canal binaire symétrique de probabilité d'erreur p 18

Cryptographie fondée sur les codes linéaires **27**

2.1 Illustration du fonctionnement d'un schéma de chiffrement symétrique 27

2.2 Illustration du fonctionnement d'un schéma de chiffrement asymétrique 28

2.3 Fonctionnement de l'OAEP 35

Primitives cryptographiques utilisant les codes QC-MDPC **49**

4.1 Protocole d'échange de clé utilisant les codes QC-MDPC 57

Algorithme de *bit flipping* **59**

5.1 Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la première itération 67

5.2 Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la deuxième itération 68

5.3 Comparaison des distributions théoriques et observées des compteurs des positions justes au début de la première itération 70

5.4 Comparaison des distributions théoriques et observées des compteurs des positions fausses au début de la première itération 70

5.5 Comparaison des distributions théoriques et observées des compteurs des positions justes au début de la deuxième itération 71

5.6 Comparaison des distributions théoriques et observées des compteurs des positions fausses au début de la deuxième itération 71

5.7 Comparaison des distributions théoriques et observées du poids de l'erreur à la fin de la première itération 72

Amélioration du calcul du seuil **81**

Calcul du seuil **81**

6.1 Comparaison des distributions théoriques et observées des compteurs des positions justes au début de la première itération 85

6.2 Comparaison des distributions théoriques et observées des compteurs des positions fausses au début de la première itération 85

6.3 Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la première itération 87

6.4 Comparaison des distributions du poids de l'erreur après une itération du *bit flipping* observées pour un seuil fixe (29) et les valeurs théoriques pour différents seuils . . . 89

6.5 Comparaison des distributions du poids de l'erreur après deux itérations du *bit flipping* observées pour les seuils $b_1 = 29$ et $b_2 = 27$ et les valeurs théoriques pour différents seuils 91

6.6 Comparaison de la distribution théorique et observée du poids du syndrome à la première itération 92

Pires cas pour le décodage **95**

6.7 Distribution du poids du syndrome et poids du syndrome des pires cas observés au début de la première itération 98

6.8 Distribution du poids du syndrome et poids du syndrome des pires cas observés au début de la seconde itération 98

6.9 Distribution du nombre moyen de positions modifiées et nombre de positions modifiées pour les pires cas au cours de la première itération 99

6.10 Distribution du nombre de positions modifiées moyen et nombre de positions modifiées pour les pires cas au cours de la deuxième itération 99

Calcul adaptatif du seuil **101**

6.11 Fonction de seuil trouvée grâce au processus heuristique d'amélioration du seuil, pour les codes $[9602, 4801, 90]$-QC-MDPC et un poids d'erreur initial de 84 105

Analyse théorique **107**

Distributions à la première itération **111**

7.1 Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la première itération pour un poids de syndrome égal à 1826 115

7.2 Comparaison du nombre de positions justes et fausses ayant une valeur de compteur donnée à la première itération pour un poids de syndrome égal à 1916 115

7.3 Comparaison des valeurs théoriques et observées du seuil adaptatif (calculé en fonction du poids du syndrome) 116

7.4 Comparaison des distributions du poids de l'erreur à la fin de la première itération en fonction du poids du syndrome 117

Résultats observés pour différents paramètres **117**

7.5 Comparaison des valeurs théoriques et observées de la distribution de la valeur des compteurs pour les positions justes pour différents poids du syndrome pour des codes $[9602, 4801, 90]$-QC-MDPC avec un poids d'erreur initial égal à 84 118

7.6 Comparaison des valeurs théoriques et observées des distribution de la valeur des compteurs pour les positions fausses pour différents poids du syndrome pour des codes $[9602, 4801, 90]$-QC-MDPC avec un poids d'erreur initial égal à 84 118

7.7 Comparaison des valeurs théoriques et observées du poids de l'erreur en fonction du poids du syndrome pour des codes $[9602, 4801, 90]$-QC-MDPC avec un poids d'erreur initial égal à 84 119

7.8 Valeur de seuil adaptatif en fonction du poids du syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$ 120

7.9 Comparaison des valeurs théoriques et observées de la distribution des compteurs des positions justes en fonction du poids du syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$ 121

7.10 Comparaison des valeurs théoriques et observées de la distribution des compteurs des positions fausses en fonction du poids du syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$ 122

Liste des tableaux

Cryptographie fondée sur les codes linéaires **27**

2.1 Cryptosystème de McEliece 30

2.2 Cryptosystème de Niederreiter 31

2.3 Cryptosystème de McEliece avec conversion sémantiquement sûre 35

Primitives cryptographiques utilisant les codes QC-MDPC **49**

4.1 Version polynomiale du cryptosystème de McEliece utilisant les codes QC-MDPC, avec conversion sémantiquement sûre 51

4.2 Choix des paramètres des codes $[2r, r, w]$-QC-MDPC pour le cryptosystème de McEliece avec un poids d'erreur initial t 55

4.3 Description polynomiale du cryptosystème de Niederreiter avec conversion sémantiquement sûre utilisant les codes QC-MDPC 56

Algorithme de *bit flipping* **59**

5.1 Récapitulatif des résultats suivant que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée 64

5.2 Récapitulatif des résultats suivant que $\mathcal{H}_0^{(u)}$ ou $\mathcal{H}_1^{(u)}$ soit vérifiée 66

5.3 Comparaison de l'espérance et de la variance théoriques et observées de la valeur des compteurs pour les positions justes et fausses au début de la première itération 69

5.4 Comparaison de l'espérance et de la variance théoriques et observées de la valeur des compteurs pour les positions justes et fausses au début de la deuxième itération . . . 69

5.5 Comparaison de l'espérance et de la variance du poids de l'erreur à la fin de la première itération 72

Amélioration du calcul du seuil **81**

Calcul du seuil **81**

6.1 Récapitulatif des résultats suivant que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée 84

6.2 Comparaison de l'espérance et de la variance des compteurs théoriques et observées, suivant que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée 86

6.3 Comparaison de l'espérance et de la variance du poids de l'erreur à la fin de la première itération, pour différentes valeurs de seuils 89

6.4 Comparaison de l'espérance et de la variance du poids du syndrome observé et théorique au début de la première itération 92

Pires cas pour le décodage **95**

6.5 Comparaison de la moyenne du poids de l'erreur pour les cas moyens avec le poids de l'erreur après les 5 premières itérations pour les pires cas 96

6.6 Comparaison de la probabilité d'échec au décodage après un nombre d'itérations fixé pour les cas moyens et la matrice de parité $H_{\mathcal{W}_0}$ 100

6.7 Comparaison de la probabilité d'échec au décodage après un nombre d'itérations fixé pour une erreur fixe "moyenne" et le pire cas $\mathcal{W}_0$ 100

6.8 Comparaison de l'espérance et de la variance de différentes données monitorables des cas moyens et des simulations avec $H_{\mathcal{W}_0}$ 100

6.9 Comparaison de l'espérance et de la variance de différentes données monitorables pour des erreurs "moyennes" et $\mathcal{W}_0$ 100

Calcul adaptatif du seuil **101**

6.10 Probabilité d'échec au décodage après un nombre fixé d'itérations 105

Analyse théorique **107**

Distributions à la première itération **111**

7.1 Comparaison de l'espérance et de la variance des compteurs des positions justes à la première itération pour différents poids de syndrome 114

7.2 Comparaison de l'espérance et de la variance des compteurs des positions fausses à la première itération pour différents poids de syndrome 114

Résultats observés pour différents paramètres **117**

7.3 Comparaison de l'espérance et de la variance du poids du syndrome observé et théorique au début de la première itération pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$ 120

7.4 Comparaison de l'espérance et de la variance des compteurs des positions justes à la première itération pour différents poids de syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$ 121

7.5 Comparaison de l'espérance et de la variance des compteurs des positions fausses à la première itération pour différents poids de syndrome pour les codes $[19714, 9857, 142]$-QC-MDPC et un poids d'erreur initial $t = 134$ 121

Codes polaires **131**

8.1 Cryptosystème de McEliece utilisant les codes polaires 133