Cryptographie fondée sur les codes : nouvelles approches pour constructions et preuves ; contribution en cryptanalyse

Thomas Debris-Alazard

Sorbonne Université

École doctorale Informatique, Télécommunications et Électronique (Paris)

Inria de Paris / Équipe-projet SECRET

Cryptographie fondée sur les codes : nouvelles approches pour constructions et preuves ; contribution en cryptanalyse

Thèse de doctorat d'informatique

présentée par

Thomas Debris-Alazard

dirigée par Jean-Pierre Tillich

soutenue publiquement le 17 décembre 2019

devant le jury composé de :

Jean-Pierre TILLICH	Inria	Directeur
Léo DUCAS	CWI	Rapporteur
Philippe GABORIT	XLIM	Rapporteur
Damien STEHLÉ	ENS Lyon	Rapporteur
Daniel AUGOT	Inria	Président
Nicolas SENDRIER	Inria	Examinateur
Annick VALIBOUZE	LIP6, Sorbonne Université	Examinatrice
Gilles ZÉMOR	Institut Mathématiques de Bordeaux	Examinateur
Ayoub OTMANI	Université de Rouen	Invité

Équipe-Projet SECRET
Inria de Paris,
2 rue Simone IFF,
75012 Paris

Remerciements

Jean-Pierre, mes premières pensées sont pour toi. Je tiens à t'exprimer ma profonde gratitude pour ces trois ans. Cela est malheureusement bien peu après tout ce que tu m'as transmis, appris. Je ne serais tout simplement pas la même personne sans toi, tu vas me manquer. J'espère que dans les années qui viennent tu seras fier de moi.

Nicolas, comment te dire, sans toi ma thèse aurait été différente sur bien des aspects et je ne serais pas le même scientifique. Merci pour ta droiture, ton calme, ton humour et ton temps. J'imagine que tu t'y attends en lisant ces lignes mais merci pour toutes nos discussions politiques bien que rarement nous ayons été d'accord. En tout cas sache que j'ai toujours le secret espoir de te convaincre un jour.

André, je ne peux pas te dire que tu es un prof génial car j'ai séché tes cours. En revanche ce n'est pas le cas de tes qualités de chercheur et d'humain. Tu m'as appris beaucoup. Bon, entre nous, déjà mon chat est plus beau que le tien. Ensuite, ne t'inquiète pas je sais ce que je dirai à mes futurs étudiants : comme mon père, grand-père etc... Merci André.

Je tiens à remercier ma mère et mon père. Vous m'avez appris la liberté et la curiosité sans lesquelles il n'y aurait de créativité. Sans votre éducation, jamais je ne me serais battu, jamais je n'y aurais toujours cru. Il va falloir continuer, je sais, comptez sur moi.

Matthieu lors de la rédaction de ces lignes tu viens d'être reçu au concours de la magistrature. Ton travail et ton abnégation ont payé. Je suis fier de toi. Je raconte maintenant à tout le monde que mon frère est magistrat même si te connaissant cela t'énerverait ! Mes pensées se dirigent aussi vers toi Pierre. Tu reliras sûrement avec malice ces lignes dans quelques années. Aujourd'hui tout t'attend, juste devant toi, et je suis sur que tu trouveras ta voie pour être heureux même s'il va falloir être un peu moins feignant !

Je tiens à te remercier Bénédicte Dezirat pour tout ce que tu m'as donné. Tu n'es malheureusement plus avec nous depuis un certain temps maintenant. Tu me manques mais tu sais il y a bien une chose que j'imagine aujourd'hui, tu aurais été d'une immense fierté ce mardi 17 décembre 2019 et tu nous aurais tous fait bouger lors du pot.

Je tiens à te remercier Pierre pour tout ce que tu m'as apporté, ton soutien et ton intérêt pour mes travaux. J'étais très insupportable lors de nos premières rencontres, j'espère que cela a changé ! En tout cas sache que j'ai hâte d'avancer avec toi sur tes projets en lesquels je crois.

Merci aux membres du jury, Léo Ducas, Philippe Gaborit, Damien Stehlé, Daniel Augot, Annick Valibouze, Gilles Zémor et Ayoub Otmani. J'ai une pensée toute particulière pour mes rapporteurs (au delà de la thèse bien trop longue que j'ai pu leur faire relire), Léo, Philippe et Damien. Léo notre rencontre a été bien hasardeuse et j'en garde un souvenir clair. Sans ton intérêt, Wave mais aussi le template de mon CV et de site web (je sais le plagiat c'est mal) n'auraient pas été les mêmes. Merci Léo. Nos discussions scientifiques n'en sont qu'à leurs débuts, j'en suis sûr. Merci Damien. Cela fut bref lors de notre première rencontre à l'ENS Lyon mais tu as su m'écouter et tout saisir plus que rapidement. Sans toi aussi Wave ne serait pas ce qu'elle est. Merci et encore une fois je n'ai que ma gratitude à te donner (et un .pdf avec cette thèse pour tes déménagements). Philippe, merci. Je t'ai

pas mal embêté ces trois dernières années et j'espère continuer si tu veux bien !

Mes remerciements se dirigent maintenant vers les membres l'équipe SECRET, pardon COSMIQ, Gaëtan, André, Maria, Anthony, Anne, Nicolas, Jean-Pierre, Pascale, Léo, Christina, Christelle, Matthieu, Valentin, Ferdinand, André, Daniel, Xavier, Kévin et Rémi. Vous permettez que l'Inria soit un endroit parfait pour s'épanouir (it is your motto), que ce soient par vos qualités scientifiques et humaines. Merci. Mes pensées vont tout d'abord vers notre fraiche nouvelle ancienne responsable d'équipe Anne. Merci Anne d'avoir permis que tout tienne debout. Merci de m'avoir obtenu le financement de cette thèse, merci pour nos discussions football. Un jour nous fêterons ensemble la victoire du PSG en ligue des champions ! Merci Pascale. Merci pour toutes nos discussions, ton écoute, ta vision des choses que je partage souvent. Heureusement que tu as été là dans les moments difficiles. Merci Christelle. J'ai souvent été plus qu'embêtant avec toi mais tu as toujours été là, merci. Plus généralement, l'équipe ne serait pas la même sans toi. Je tiens aussi à remercier Antoine, Fodil et Julien. Grâce à vous mes lundis ont toujours été plaisants. J'espère juste que vous ne m'en voulez pas trop. En tout cas je vous avais prévenus, c'est plus que difficile de me battre à MPG.

Je tiens aussi à remercier tout particulièrement Kevin, Matthieu, Xavier, André, Valentin, Rémi. Kévin, merci de m'avoir supporté dans ton bureau pendant trois ans (et pendant les conférences). Nos destins semblent maintenant se séparer mais j'espère que l'on continuera à se voir et bien-sûr arpenter Paris ensemble. Merci Matthieu pour tous nos échanges même si je pense qu'ils n'ont pas été assez nombreux ! Tu peux me faire confiance je vais tout faire pour que cela change. Xavier, merci pour ces trois ans, que ce soit par ton soutien et nos divers échanges. Tu sais, comme tu me le rappelles souvent, l'important ce sont les valeurs. Merci André, ta passion pour la recherche m'impressionne et me sert aujourd'hui d'exemple. Merci Valentin, bien que ce soit difficile de te voir tôt le matin j'ai toujours eu le plaisir de t'embêter. Merci Rémi pour nos travaux en commun. J'espère que tu me pardonnes de n'être pas assez rigoureux au tableau.

Merci aux anciens de Secret Julia et Yann ! Merci Yann pour ta bonne humeur, nos nombreuses pauses. Merci Julia pour ton accueil lors de mon arrivée, j'en garde un beau souvenir. J'ai un peu grandi depuis mais je reste ton petit scarabée, je sais.

Je tiens aussi à te remercier Alain. Déjà pour m'avoir présenté Jean-Pierre mais aussi pour ton sens de l'écoute. Tu as toujours été là. J'espère maintenant une chose, que nos collaborations soient plus que nombreuses.

Merci Dahmun et Thomas. Dahmun, ces trois années nous ont permis de mieux nous connaître sur le plan scientifique et amical. Merci pour ton travail sur Surf, tu as finalement été le premier à nous soutenir. Merci d'avoir partagé avec André et moi ton envie de voir l'Iran aller en finale de la coupe du monde. Bon, on se rappelle bien comment cela s'est terminé, sacré portugais ! Merci Thomas. Déjà parce que je t'ai bien embêté avec des histoires de politique. Excuse-moi d'avance mais cela devrait continuer ! Merci pour ton soutien sur Wave, merci pour nos nombreuses discussions scientifiques qui j'espère vont continuer.

Je tiens à remercier ma famille, mes tantes, mes grands-parents, Christine. J'ai une pensée toute particulière pour ma taty Carole (et son compagnon Guillaume) ainsi que ma tâtâ Sophie. Merci.

Merci mes amis, Jean-Baptiste, Pablo, Timothee, Enzo, Arthur, Félix, Antoine, Miloch, Théo, Toufik, Lucas, Thibault, Audrey. Vous m'avez aidé à traverser les bonnes périodes, heureusement vous avez *toujours* été là. Mais il n'y a pas que les moments difficiles, il ne faut pas oublier, nombre de nos bons moments c'est ensemble que nous les avons vécus et ça va continuer !

Merci Sylvie, Pierre-Louis, Philippe, Valérie, Gwenaëlle. Sylvie et Pierre-Louis je n'imaginais pas que cela se terminerait comme ça (surtout après quelques cours avec toi Pierre-Louis). Sylvie et Philippe, Pierre-Louis peut vous remercier. Pierre-Louis, je suis fier de toi

(ainsi que de t'avoir aidé). J'espère maintenant que chaque été nous nous retrouverons à Gilly. Je n'ai pas dit mon dernier mot au scrabble.

Florence, je ne m'étendrai pas ici, je sais que tu n'aimerais pas. De toute façon nous savons ce qui nous relie, nous savons ce qui nous anime. Laisse moi simplement terminer là-dessus : crois-moi, nous avons encore un long chemin à parcourir ensemble.

À Jean-Pierre,
Nicolas,
André,
Florence.

Table des matières

Introduction

Cryptographie post-quantique

L'année 1978 naquit le premier schéma de chiffrement à clef publique avec la proposition de Rivest, Shamir et Adleman [RSA78]. La sécurité de ce cryptosystème repose sur le problème de la factorisation issu de la théorie des nombres : étant donné $n = pq$ où p et q sont premiers, les retrouver. Cette même année 1978 une seconde proposition fut faite mais reposant quant à elle sur l'utilisation de codes correcteurs d'erreurs : le schéma de McEliece [McE78]. Ce système, bien que jouissant de certains avantages comparativement à RSA (par exemple la rapidité du chiffrement et déchiffrement), ne fut pas utilisé en pratique. En effet, la clef publique nécessaire au chiffrement RSA est bien plus petite ce qui fut jugé à l'époque plus performant. De nos jours, ce système ainsi que le protocole d'échange de clefs de Diffie-Hellman [DH76] sont les plus utilisés en pratique. Le protocole de [DH76] est entre autres lié au problème dit du logarithme discret : étant donné un groupe $\mathbb{G}$ de générateur g et g^x, retrouver x.

La cryptographie à clef publique repose donc uniquement sur deux problèmes. Or ces-derniers ne sont pas si éloignés, ils sont tout deux une instance du problème dit du sous-groupe caché [Joz01] dans un groupe abélien. Il n'existe pas à ce jour d'attaque classique contre ce problème. En revanche la situation est bien différente dans un monde où l'ordinateur quantique existe depuis la découverte par Shor [Sho94] d'un algorithme quantique résolvant le problème du sous-groupe caché dans un groupe abélien. Les protocoles [RSA78] et [DH76] seront donc caducs une fois que les premiers ordinateurs quantiques de taille suffisante auront été construits. Face à ce danger il fut alors durant plusieurs années invoqué l'infaisabilité d'un tel ordinateur. C'était sans compter sur les récents progrès techniques qui aujourd'hui rendent l'existence d'un ordinateur quantique efficace probable dans les prochaines décennies. Google a même récemment prétendu avoir obtenu la suprématie quantique [Aru+19], c'est à dire être en mesure de résoudre en temps raisonnable avec un ordinateur quantique un problème hors d'atteinte pour des ordinateurs classiques. Ce résultat est cependant contesté par IBM [1]. Quoiqu'il en soit le temps se fait pressant pour trouver et étudier de nouveaux paradigmes mathématiques pour lesquels la cryptographie à clef publique sera sûre quantiquement. En effet, certaines données ont vocation à être confidentielles pour les cinquante prochaines années.

Fort heureusement nous connaissons des solutions cryptographiques fonctionnant sur ordinateur classique et ayant des chances d'être quantiquement sûres. On parle usuellement de *cryptographie post-quantique*. Le cryptosystème de McEliece est par exemple revenu sur le devant de la scène après la découverte de Shor. De nos jours, les seules attaques quantiques connues contre ce système requièrent une complexité exponentielle en sa taille. Il existe actuellement bien d'autres propositions, nous pouvons citer parmi les plus prometteuses celles reposant sur les réseaux euclidiens, les fonctions de hachage, les systèmes multi-variés ou plus récemment les isogénies. Nous changeons donc d'ère cryptographique, les problèmes sur lesquels peuvent reposer la sécurité numérique sont

1. https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/

tout aussi bien variés que de nature différente. Néanmoins ces domaines cryptographiques ont été bien moins étudiés que ne le furent [RSA78] et [DH76]. Or il y a aujourd'hui urgence à proposer des systèmes sûrs aussi bien classiquement que quantiquement.

C'est dans ce contexte que le *National Institute of Standard Technology* (NIST) du gouvernement américain lança en 2017 [2] un appel pour la standardisation de systèmes à clef publique. L'objectif était alors de lancer un processus de sélection visant à stimuler la recherche pour trouver les meilleures alternatives à [RSA78] et [DH76]. Nous sommes au moment de l'écriture de ce document au deuxième tour de sélection. Les tables 1 et 2 comparent alors les différentes propositions entre les deux tours de l'appel du NIST.

Table 1 – Comparaison des soumissions au NIST du premier tour.

	Proportion	Échange de clefs	Signature
Réseaux euclidiens	38%	22	5
Codes	31%	19	3
Fonctions de hachage	3%	0	2
Multi-varié	15.5%	2	9
Isogénies	1,5%	1	0
Preuves ZK	1,5%	0	1
Autres	8.5%	5	1

Table 2 – Comparaison des soumissions au NIST du second tour.

	Proportion	Échange de clefs	Signature
Réseaux	46%	9	3
Codes	28%	7	0
Fonctions de hachage	4%	0	1
Multi-varié	15%	0	4
Isogénies	4%	1	0
Preuves ZK	4%	0	1

Cette thèse se déroula dans ce cadre d'étude de la cryptographie post-quantique à clef publique avec comme tâche de fond la standardisation du NIST. Nous nous sommes tout particulièrement intéressés à la branche cryptographique née de la proposition de McEliece reposant sur le problème du décodage.

Cryptographie utilisant des codes correcteurs d'erreurs

Un code correcteur d'erreurs linéaire $\mathcal{C}$ est défini comme un sous-espace vectoriel de $\mathbb{F}_q^n$. Ce concept fut initialement introduit dans le contexte des télécommunications et de la sauvegarde des données numériques. En effet, de façon inhérente au processus d'enregistrement ou de transmission de l'information, des erreurs peuvent venir corrompre les données numériques. Le problème algorithmique central, dit de *décodage*, est alors : étant donné un mot de code bruité $\mathbf{c} + \mathbf{e}$ ($\mathbf{c} \in \mathcal{C}$ représente nos données) où $\mathbf{e}$ est de petit poids de Hamming w, retrouver efficacement $\mathbf{c}$. L'objectif de la théorie des codes depuis maintenant près de soixante-dix ans fut alors de proposer des codes munis d'une structure

2. https://csrc.nist.gov/projects/post-quantum-cryptography/

forte permettant un décodage efficace. Si tous ces efforts furent faits c'est que le décodage d'un code quelconque semble par nature être difficile comme son caractère NP-complet l'indique. De plus, après de nombreuses années de recherche, les meilleurs algorithmes connus de décodage de codes aléatoires sont tous de complexité $2^{c \cdot w}$ pour une certaine constante $c > 0$.

Dans ce contexte McEliece introduisit son chiffrement à clef publique avec pour objectif de faire reposer la sécurité sur le problème du décodage d'un code aléatoire, dit *décodage générique*, ce qui donna naissance à la cryptographie utilisant des codes. L'idée est la suivante : nous nous donnons un code de longueur n que l'on sait décoder à une certaine distance, disons w, puis nous rendons publique une représentation du code où la structure est cachée. Maintenant pour chiffrer un message, étant un mot de code, nous lui ajoutons une erreur de poids de Hamming w. Le déchiffrement consiste alors à utiliser la structure du code pour décoder et retrouver le message envoyé. Ce schéma est donc correct mais comment s'assurer que seul le détenteur du secret est en mesure de déchiffrer ? Une réponse est donnée avec la notion de *réduction de sécurité*. De façon générale ce concept permet de prouver mathématiquement que casser un système cryptographique, c'est à dire pour simplifier retrouver le secret, implique résoudre tel ou tel problème que l'on pense difficile. Dans le cas de McEliece nous savons qu'attaquer le chiffrement revient à résoudre le problème du décodage générique *ou* distinguer un code obtenu avec la structure cachée d'un code quelconque. Le choix du code est donc ici crucial et McEliece proposa dans sa première instanciation d'utiliser des codes de Goppa. Ces-derniers ont jusque-là été robustes à la cryptanalyse, aucune attaque pour les distinguer de codes quelconques n'a été trouvé pour des paramètres raisonnables. De plus, ils permettent un décodage à une très bonne distance, de l'ordre de $n/\log_2(n)$. Depuis, de nombreux codes cryptographiques furent proposés pour des chiffrements *à la McEliece* comme les codes MDPC [Mis+12], permettant un décodage à distance $\sqrt{n}$, ou encore les codes LRPC [Gab+13], ces-derniers utilisant une autre métrique que celle de Hamming, la métrique rang.

En revanche, en terme de primitives cryptographiques plus avancées, comme les signatures, la situation est en comparaison mauvaise. Toutes les signatures proposées ont été cassées ou sont de taille extrêmement importante. Cette situation est illustrée par le second tour du processus de standardisation post-quantique du NIST où aucune signature utilisant des codes n'est présente (voir la table 2). Si nous faisons l'exception des signatures obtenues à partir de la transformation de Fiat-Shamir [FS87] avec un protocole à divulgation nulle de connaissance (par exemple avec des codes le protocole de Stern [Ste93a]), l'histoire des signatures utilisant des codes a essentiellement consisté en deux approches. La première est celle de [KKS97]. Le problème de ce schéma est que les signatures font fuiter de l'information sur la clef secrète comme montré par les nombreuses attaques [COV07 ; OT11 ; Hue+18] à son encontre. Nous ne pouvons actuellement qu'au mieux considérer cette primitive comme une signature unique. La seconde approche, de type hache et signe, fut celle de CFS [CFS01]. Il s'agissait ici de trouver une famille de codes telle que l'on soit en mesure de décoder à distance bornée $\leqslant w$ au moins presque tout mot de l'espace ambiant. Nous désignons ici par décodage, retrouver à partir d'un vecteur l'unique mot de code à distance $\leqslant w$. Les auteurs de [CFS01] remarquèrent que les codes de Goppa avec leur algorithme de décodage permettaient d'accomplir cette tâche mais en revanche pour des paramètres extrêmes : leur dimension est très proche de leur longueur. Ces choix de dimensions impliquent alors une mauvaise mise à l'échelle du système. Les clefs sont par exemple, pour 128 bits de sécurité, de l'ordre de plusieurs Gigabytes. En revanche, contrairement à KKS il n'y a pas de fuite d'information dans CFS étant donné que pour un mot quelconque la signature est *l'unique* mot de code à distance $\leqslant w$.

Les travaux de cette thèse se sont alors principalement inscrits dans ce contexte de signature utilisant des codes correcteurs. Nous proposons entres autres une telle primitive, Wave.

Contributions

Résumé des travaux de recherche

Nos contributions s'organisent selon trois axes.

Construction de primitives [DST19a]

- i (Chapitre 5) : Nous construisons une nouvelle primitive de signature, baptisée Wave, utilisant des codes correcteurs et de type hache et signe. Ce schéma est fondé sur une nouvelle construction de codes à trappe pour le problème du décodage : les codes $(U, U+V)$-généralisés permutés. Nous montrons comment décoder ces codes en des distances telles (i) qu'il existe un grand nombre de solutions au problème du décodage et (ii) pour lesquelles il est difficile d'en retrouver ne serait-ce qu'une générique ment, c'est à dire lorsque le code est aléatoire. Nous montrons ensuite que pour cette famille de codes et son décodeur la stratégie de Gentry Peikert et Vaikuntanathan [GPV08b], qui s'est avérée fructueuse en cryptographie utilisant des réseaux euclidiens, peut être suivie. Nous avons pour cela introduit une méthode de rejet très efficace nous permettant de prouver que les signatures de notre algorithme évitent toute fuite d'information sur la clef secrète.
- ii (Chapitre 6) : Nous montrons que Wave est sûr dans le modèle de l'oracle aléatoire sous les hypothèses qu'une variante multi-instances du décodage générique en grande distance est difficile ainsi que distinguer un code $(U, U+V)$-généralisé permuté d'un code aléatoire.
- iii (Chapitre 7) : Nous proposons des algorithmes pour distinguer un code $(U, U+V)$-généralisé permuté d'un code aléatoire. Nos techniques sont proches de celles utilisées pour décoder un code aléatoire. De plus, nous montrons que décider si un code est un $(U, U+V)$-généralisé permuté ou non est NP-complet.

Analyse de sécurité [Bri+19 ; DT17]

- i (Chapitre 3) : Nous généralisons les algorithmes de décodage générique *en grande distance*. Cette étude est essentielle pour déterminer la difficulté de ce problème sur lequel repose entres autres Wave. De plus nous montrons que ce problème est au moins aussi difficile que son pendant en petite distance.
- ii (Chapitre 4) : Nous étudions une des rares alternatives du décodage par ensemble d'information : le décodage statistique [Jab01]. Dans un premier temps nous améliorons les techniques utilisées pour trouver des équations de parité de poids faible ou modéré puis nous donnons la première étude asymptotique de ce décodeur. Nous montrons alors, contrairement à ce qui avait pu être affirmé [FKI07], que le décodage statistique n'est pas compétitif avec les décodeurs par ensemble d'information. Cette étude repose sur de nouveaux résultats portant sur les polynômes de Krawtchouk.

Cryptanalyse [DT18c]

- i (Chapitre 8) : `RankSign` [Gab+14a] était la dernière primitive de signature utilisant des codes encore présente au premier tour de standardisation du NIST. Nous avons monté une attaque polynomiale contre ce schéma en montrant (et utilisant) une nouvelle propriété surprenante des codes LRPC utilisés dans le schéma : ceux-ci contiennent des mots de poids rang faible révélant le secret. L'existence de ces mots est due aux contraintes particulières imposées par `RankSign` qui est de type hache et signe.
- ii (Chapitre 9) : Nous montons une attaque contre les paramètres du premier et unique *Identity-Based-Encryption* (IBE) [Gab+16] utilisant des codes correcteurs en métrique rang. Nous montrons qu'il est possible de choisir les paramètres de façon à éviter cette attaque contrairement à l'adaptation de l'IBE en métrique de Hamming.

Construction du schéma de signature Wave

Le schéma de signature Wave utilise des codes et est de type hache et signe au même titre que CFS. En revanche notre approche fut différente. Nous n'avons pas cherché une famille de codes avec un algorithme de décodage stricto sensu. Dans le cas de Wave nous avons plutôt cherché à proposer une famille de codes permettant de trouver pour un mot quelconque *un des mots de code* à distance w. En particulier, notre distance de décodage w est bien plus grande que dans CFS, ce qui nous assure *typiquement* de l'existence d'un mot de code à distance w. En revanche, le mot de code n'est plus unique et il en existe même un très grand nombre ce qui pose deux problèmes. Premièrement il existe peu de codes permettant un tel décodage (on parle usuellement de distorsion en théorie des codes) et deuxièmement, il se peut que notre choix de mot de code lors du "décodage" à distance w soit biaisé et fasse ainsi fuiter de l'information. Ce dernier point est problématique. Nous avons alors pallié ces problèmes en proposant le schéma Surf [DST17a] qui n'apparaît pas explicitement dans cette thèse. Nous avons proposé une nouvelle famille de codes à trappe en cryptographie : les $(U, U+V)$-binaires permutés. L'algorithme que nous avons proposé et qui leur est associé vient avec une méthode de rejet permettant essentiellement de prouver que les solutions produites ne font pas fuiter d'information.

Nous avons malheureusement trouvé un attaque contre les codes $(U, U+V)$-binaires permutés, c'est à dire que leur structure ne peut se cacher et ainsi mis à mal notre approche. Cependant notre attaque est profondément liée au corps binaire. Nous avons donc naturellement généralisé nos travaux en augmentant la taille de corps $q > 2$ sur lequel sont construits nos codes $(U, U+V)$. Cela nous permit d'éviter l'attaque mais aussi "d'augmenter" la famille de codes considérés en définissant les codes $(U, U+V)$-généralisés. En revanche, afin que notre algorithme de décodage ne fasse pas fuiter d'information il est nécessaire que nous décodions en très grande distance, de l'ordre de 95% de la longueur du code. De plus, les codes U et V doivent être choisis de façon aléatoire (le code $(U, U+V)$-généralisé consiste en un mélange de U et V). Le décodage de Wave utilise alors de façon cruciale la structure des corps de taille > 2, l'aléa fournit par les codes U et V ainsi que le plus simple des algorithmes par ensemble d'information [Pra62]. Ceci nous permit de nous inscrire dans l'approche de Gentry, Peikert et Vaikuntanathan en proposant une méthode de rejet (extrêmement efficace) permettant de prouver que notre décodage ne fait pas fuiter d'information.

Nous avons alors donné une réduction de sécurité de Wave aux problèmes du décodage générique multi-instances (on cherche à décoder non pas un mot mais un parmi une grande liste de candidats) ou de distinguer un code $(U, U+V)$-généralisé permuté d'un code aléatoire. Nous prouvons dans ce document que le second problème est NP-complet ce qui fait de Wave, au mieux de notre connaissance, le premier schéma utilisant des codes à trappe et dont les deux problèmes sur lesquels il repose sont NP-complets. En revanche notre réduction du problème de décider si un code est un $(U, U+V)$-généralisé permuté ou non est faite dans un régime de paramètres pour lequel notre décodage ne fonctionne pas. Cependant la réduction n'utilise pas notre structure généralisée des codes $(U, U+V)$.

Nous proposons dans ce document le meilleur algorithme connu à ce jour pour distinguer les codes $(U, U+V)$-généralisés permutés de codes aléatoires. Ce-dernier est de complexité exponentielle en la longueur du code. Il est à noter que nos algorithmes sont très proches de ceux utilisés pour le décodage générique. Ceci nous laisse aujourd'hui penser que ce problème de distinction se réduit justement au décodage générique mais cette question est hors du propos de cette thèse.

Fondements de sécurité

Le décodage en grande distance pour des corps autres que binaire est pour la première fois considéré dans cette thèse. Bien que ce problème ne soit pas naturel, la sécurité de Wave

s'y reposant, il est essentiel de l'étudier. Nous proposons de généraliser les algorithmes par ensemble d'information, dits ISD (*Information Set Decoding*), à ces nouveaux paramètres. La famille d'algorithmes ISD est aujourd'hui la plus efficace pour résoudre le décodage générique. Notre généralisation de ces algorithmes non triviaux nous permit de jauger finement la difficulté du problème dans l'état de l'art algorithmique et ainsi de proposer des paramètres concrets pour Wave. De plus, il s'est avéré de façon surprenante que le décodage générique en grande distance est plus difficile (dans l'état de l'art) que son pendant en petite distance. Ce résultat ouvre donc la perspective d'utiliser le problème du décodage générique en grande distance pour construire de nouvelles primitives cryptographiques. Les tailles des clefs en cryptographie utilisant des codes se verraient alors réduites.

Enfin nous nous intéressons à l'une des rares alternatives du décodage par ensemble d'information : le décodage statistique [Jab01]. L'idée de cet algorithme pour décoder un mot de code bruité consiste à utiliser des vecteurs de petits poids du code orthogonal. Ces-derniers servent à "annuler" le mot de code du mot de code bruité tout en espérant révéler l'erreur. Cette approche existe dans le contexte de la cryptographie utilisant des réseaux euclidiens et est même parfois l'une des meilleures solutions en terme de complexité. En revanche, dans le cas du décodage de codes binaires, bien que prometteuse, la complexité du décodage statistique était jusqu'à maintenant très mal comprise faute d'étude asymptotique. Nous avons alors dans un premier temps donné la complexité asymptotique de cet algorithme (à l'aide de nouveaux résultats portant sur les polynômes de Krawtchouk) puis dans un deuxième temps nous avons amélioré les techniques utilisées pour trouver des équations de parité de poids faible ou modéré. Nous montrons alors (sous des hypothèses naturelles), contrairement à ce qui avait pu être affirmé [FKI07], que le décodage statistique n'est pas compétitif avec les décodeurs par ensemble d'information et ce même pour le plus simple d'entre eux [Pra62].

Cryptanalyse

Il existe en cryptographie utilisant des codes une autre distance que celle de Hamming : la métrique rang. Les codes sont ici vus comme des sous-espaces matriciels équipés de la distance rang, *i.e* : la distance entre deux matrices est le rang de leur différence. L'avantage de travailler avec de tels codes est que la plupart du temps les tailles de clef sont extrêmement petites. Cela s'explique par le fait qu'il est classiquement considéré des codes linéaires construits sur une extension de corps $\mathbb{F}_{q^m}$ que l'on peut alors voir comme des codes matriciels sur le corps de base $\mathbb{F}_q$. Cette structure donne alors de manière naturelle des codes avec un grand groupe d'automorphismes.

Les codes munis de la métrique rang sont aujourd'hui des solutions prometteuses (deux chiffrements, ROLLO [Ara+19b] et RQC [Agu+19], utilisant cette métrique sont au second tour du NIST) et de nombreuses primitives cryptographiques sont proposées comme des chiffrements [Gab+13 ; Bet+19 ; Ara+19b], des signatures [Gab+14a ; Ara+19a], un IBE [Gab+16], un générateur pseudo-aléatoire [GHT16] etc.

Nous présentons dans ce document deux attaques contre de tels schémas. La première concerne la signature de type hache et signe, `RankSign` [Gab+14a]. Cette primitive jouissait de petites tailles de clefs et était candidate au processus de standardisation post-quantique du NIST. Nous avons cependant monté une attaque polynomiale contre `RankSign`. Celle-ci exploite le fait que l'algorithme de signature, qui utilise une procédure de décodage avec effacement de codes LRPC, ne peut fonctionner que pour des zones de paramètres très particulières du code public. Nous montrons alors que les codes publics utilisés contiennent des mots de petit poids révélant le secret. Ces mots peuvent être retrouvés avec des techniques de type base de Gröbner, ce que nous avons vérifié en pratique. Notre deuxième attaque est contre les paramètres de l'unique et premier IBE [Gab+16], inspiré de celui de [GPV08a] proposé avec des réseaux euclidiens, dont la sécurité repose sur des problèmes

de code. Nous utilisons des techniques similaires à celles nous ayant permis d'attaquer `RankSign`. En revanche, nous montrons qu'il est toujours possible de sélectionner des paramètres de façon à éviter notre attaque. Ce résultat s'oppose alors à notre troisième cryptanalyse. Nous avons en effet généralisé l'IBE de [Gab+16] à la métrique de Hamming et donné une attaque polynomiale contre ce-dernier. Notre résultat montre essentiellement qu'appliquer les techniques de l'IBE [GPV08b ; Gab+16] aux codes binaires en métrique de Hamming se fait difficilement.

Organisation de la thèse

Ce document est organisé en quatre parties précédées de quelques notations. Dans la partie I nous donnons un état de l'art de la cryptographie utilisant des codes en nous focalisant sur divers cryptosystèmes ainsi que le problème du décodage générique. En revanche nous nous focaliserons peu sur les attaques structurelles (en particulier algébriques) contre les codes proposés en cryptographie. Nous nous intéressons ensuite dans un premier temps de la partie II au problème du décodage générique en grande distance. Dans un second temps nous étudions la complexité du décodage statistique ainsi que donnons quelques améliorations. La partie III est elle consacrée à la conception du schéma de signature Wave. Nous décrivons dans le premier chapitre le système. Dans le second chapitre nous montrons que Wave est sûr dans le modèle de l'oracle aléatoire sous deux hypothèses calculatoires. Nous terminons ensuite cette partie en proposant des algorithmes distinguant des codes $(U, U+V)$-généralisés permutés de codes aléatoires. Ce document se conclut par deux cryptanalyses en métrique rang dans la partie IV. La première concerne le schéma de signature `RankSign` alors que la seconde s'attaque aux paramètres du premier et seul IBE utilisant des codes.

Notations

Nous fixons ici quelques notations qui seront utilisées tout au long de ce document.

Notations générales

Généralités. La notation $x \stackrel{\triangle}{=} y$ signifie que x est défini comme étant égal à y. Nous dénoterons par $\mathbb{F}_q = \{0, 1, \cdots, q-1\}$ le corps fini à q éléments. Pour deux entiers a et b tels que $a \leqslant b$, nous désignerons par $[\![a, b]\!]$ l'ensemble des entiers $\{a, a+1, \ldots, b\}$. De plus, pour un ensemble $\mathcal{E}$, la notation $\#\mathcal{E}$ ou $|\mathcal{E}|$ désignera son cardinal.

Une fonction $f(n)$ est sera dite négligeable, ce que l'on notera :

$$f \in \text{negl}(n)$$

si pour tout polynôme $p(n)$, $|f(n)| < 1/p(n)$ pour tout entier n suffisamment grand.

Notations matrices,vecteurs. Les vecteurs seront notés avec des lettres grasses minuscules (telles que $\mathbf{e}$) et les matrices avec des majuscules (telles que $\mathbf{H}$). Les vecteurs seront en *notation ligne* et nous noterons $\mathbb{F}_q^{m \times n}$ l'ensemble des matrices à coefficients dans $\mathbb{F}_q$ ayant m lignes et n colonnes.

Pour un vecteur $\mathbf{x}$ nous pourrons noter x_i ou $\mathbf{x}(i)$ sa i-ème composante. De même pour une matrice $\mathbf{H}$, nous désignerons par $h_{i,j}$ ou $\mathbf{H}(i,j)$ sa composante de coordonnée (i,j). Nous noterons $\mathbf{x}_{\mathcal{I}}$ le vecteur dont les coordonnées sont celles de $\mathbf{x} = (x_i)_{1 \leqslant i \leqslant n}$ mais restreintes à l'ensemble $\mathcal{I} \subseteq [\![1, n]\!]$, *i.e :* $\mathbf{x}_{\mathcal{I}} = (x_i)_{i \in \mathcal{I}}$. De même, $\mathbf{H}_{\mathcal{I}}$ désignera la matrice dont les colonnes sont celles de $\mathbf{H} \in \mathbb{F}_q^{m \times n}$ restreintes à $\mathcal{I}$, *i.e :* $\mathbf{H}_{\mathcal{I}} = (h_{i,j})_{\substack{1 \leqslant i \leqslant m \\ j \in \mathcal{I}}}$. De plus, la notation $\text{Rang}(\mathbf{H})$ désignera le rang de la matrice $\mathbf{H}$ tandis que pour un vecteur $\mathbf{x} \in \mathbb{F}_q^n$ nous dénoterons par $\mathbf{Diag}(\mathbf{x})$ la matrice diagonale de $\mathbb{F}_q^{n \times n}$ définie comme :

$$\mathbf{Diag}(\mathbf{x})(i,i) \stackrel{\triangle}{=} x_i \quad \text{et si } i \neq j, \quad \mathbf{Diag}(\mathbf{x})(i,j) \stackrel{\triangle}{=} 0.$$

Le support d'un vecteur $\mathbf{x}$ de longueur n est quant à lui défini comme

$$\text{Supp}(\mathbf{x}) \stackrel{\triangle}{=} \{1 \leqslant i \leqslant n : x_i \neq 0\}$$

tandis que le support d'un ensemble $\mathcal{E} \subseteq \mathbb{F}_q^n$ de vecteurs se définit de la façon suivante :

$$\text{Supp}(\mathcal{E}) \stackrel{\triangle}{=} \bigcup_{\mathbf{x} \in \mathcal{E}} \text{Supp}(\mathbf{x}).$$

Nous définissons le poids de Hamming de $\mathbf{x} \in \mathbb{F}_q^n$ comme,

$$|\mathbf{x}| \stackrel{\triangle}{=} \# \,\text{Supp}(\mathbf{x}).$$

Lorsque le contexte le permettra nous parlerons tout simplement de poids de $\mathbf{x}$. De plus, nous utiliserons très souvent la notation $S_{w,n}$ (ou S_w si le contexte le permet) pour désigner l'ensemble des mots de $\mathbb{F}_q^n$ de poids de Hamming w, *i.e* :

$$S_{w,n} \stackrel{\triangle}{=} \{\mathbf{x} \in \mathbb{F}_q^n : |\mathbf{x}| = w\}.$$

Nous dénoterons par $(\mathbf{x}, \mathbf{y})$ la concaténation des deux vecteurs $\mathbf{x}$ et $\mathbf{y}$. On désignera pour $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ leur produit scalaire par :

$$\langle \mathbf{x}, \mathbf{y} \rangle \stackrel{\triangle}{=} \sum_{i=1}^{n} x_i y_i.$$

Maintenant, nous noterons $\mathbf{x} \odot \mathbf{y}$ le produit par composantes, aussi appelé produit de Hadamard ou de Shur, étant défini comme :

$$\mathbf{x} \odot \mathbf{y} \stackrel{\triangle}{=} (x_1 y_1, \cdots, x_n y_n).$$

Notations probabilistes. Soit S un ensemble fini, alors $x \hookleftarrow S$ dénote le fait que x est tirée uniformément dans S. Pour deux variables aléatoires X et Y, $X \sim Y$ signifiera que X et Y sont équi-distribuées. Nous utiliserons la même notation pour une variable aléatoire X et une distribution $\mathcal{D}$, la notation $X \sim \mathcal{D}$ signifiant que X est distribuée selon $\mathcal{D}$. De plus, la notation $x \hookleftarrow \mathcal{D}$ (resp. $x \hookleftarrow X$) dénote le fait que x est tirée selon la distribution $\mathcal{D}$ (resp. la variable aléatoire X). De plus, si $\mathcal{E}$ désigne un ensemble fini alors, sauf mention du contraire, $x \hookleftarrow \mathcal{E}$ signifiera que x est tiré de façon uniforme dans $\mathcal{E}$.

Nous souhaiterons de temps en temps expliciter l'espace probabiliste sur lequel les probabilités ou espérances sont considérées. Pour cela nous indiquerons en indice de la probabilité la variable aléatoire ou la distribution sur laquelle cette dernière est calculée. A titre d'exemple, si $\mathcal{D}$ désigne une distribution et $\mathbf{x} \hookleftarrow \mathcal{D}$, la probabilité de l'évènement "$\mathbf{x} \in \mathcal{E}$" sera notée $\mathbb{P}_{\mathbf{x}}(\mathbf{x} \in \mathcal{E})$.

Distances statistique et calculatoire. La distance statistique entre deux distributions discrètes $\mathcal{D}_0$ et $\mathcal{D}_1$ sur un même espace $\mathcal{E}$ est définie comme :

$$\rho(\mathcal{D}_0, \mathcal{D}_1) \stackrel{\triangle}{=} \frac{1}{2} \sum_{x \in \mathcal{E}} |\mathcal{D}_0(x) - \mathcal{D}_1(x)|.$$

Pour deux variables aléatoires X et Y à valeurs dans le même espace, nous noterons aussi $\rho(X, Y)$ la distance statistique entre la distribution $\mathcal{D}_X$ de X et la distribution $\mathcal{D}_Y$ de Y : $\rho(X, Y) \stackrel{\triangle}{=} \rho(\mathcal{D}_X, \mathcal{D}_Y)$.

La distance calculatoire entre deux distributions est définie de la façon suivante. Nous commençons par introduire la notion de *distingueur* entre deux distributions $\mathcal{D}_0$ et $\mathcal{D}_1$ sur un même espace $\mathcal{E}$. Il s'agit d'un algorithme $\mathcal{A}$ probabiliste prenant en entrée un élément de $\mathcal{E}$ distribué selon $\mathcal{D}_0$ ou $\mathcal{D}_1$ et qui renvoie un bit $b \in \{0, 1\}$. Ce dernier est alors caractérisé par son avantage :

$$Adv^{\mathcal{D}_0, \mathcal{D}_1}(\mathcal{A}) \stackrel{\triangle}{=} \mathbb{P}_{X_0, c}\left(\mathcal{A}(X_0) = 1\right) - \mathbb{P}_{X_1, c}\left(\mathcal{A}(X_1) = 1\right)$$

où $X_i \sim \mathcal{D}_i$ et c l'aléa interne de $\mathcal{A}$. La distance calculatoire entre les deux distributions $\mathcal{D}_0$ et $\mathcal{D}_1$ est alors définie comme :

$$\forall t \geqslant 0, \quad \rho_c(\mathcal{D}_0, \mathcal{D}_1)(t) \stackrel{\triangle}{=} \max_{\mathcal{A}:|\mathcal{A}| \leqslant t} \left(Adv^{\mathcal{D}0, \mathcal{D}_1}(\mathcal{A})\right)$$

avec $|\mathcal{A}|$ désignant le temps de calcul de $\mathcal{A}$ sur ses entrées.

Notation de Bachmann-Landau. Nous utiliserons tout au long de ce document la famille de notations de Bachmann-Landau. Soient deux fonctions $f, g : \mathbb{N} \to \mathbb{R}$. On notera :

$$f(n) \stackrel{\triangle}{=} o(g(n)) \iff \lim_{n \to +\infty} \frac{f(n)}{g(n)} = 0.$$

$$f(n) \stackrel{\triangle}{=} O(g(n)) \iff \exists M > 0,\ \forall n \in \mathbb{N} : |f(n)| \leqslant M \cdot |g(n)|$$

$$f(n) \stackrel{\triangle}{=} \Omega(g(n)) \iff \exists m > 0,\ \forall n \in \mathbb{N} : |f(n)| \geqslant m \cdot |g(n)|$$

$$f(n) \stackrel{\triangle}{=} \Theta(g(n)) \iff f(n) = O(g(n)) \text{ et } f(n) = \Omega(g(n))$$

et

$$f(n) \stackrel{\triangle}{=} \tilde{O}(g(n)) \iff \exists k \in \mathbb{N},\ f(n) = \log^k |g(n)| \cdot O\left(g(n)\right).$$

Théorie des codes

Un code linéaire de dimension k et de longueur n sur un corps fini $\mathbb{F}_q$, ce que l'on abrègera par $[n,k]_q$-code, est un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k. Nous désignerons, pour un $[n,k]_q$-code, par *rendement* le rapport k/n. Il est classique d'appeler un vecteur d'un code *un mot de code*. Un $[n,k]_q$-code $\mathcal{C}$ peut être spécifié par une matrice génératrice, c'est à dire une matrice $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ dont les k lignes forment l'une de ses bases en tant qu'espace vectoriel :

$$\mathcal{C} = \left\{ \mathbf{mG} : \mathbf{m} \in \mathbb{F}_q^k \right\}.$$

L'orthogonal (ou le dual) $\mathcal{C}^\perp$ d'un $[n,k]_q$-code $\mathcal{C}$ se définit comme :

$$\mathcal{C}^\perp \stackrel{\triangle}{=} \left\{ \mathbf{h} \in \mathbb{F}_q^n : \forall \mathbf{c} \in \mathcal{C}, \langle \mathbf{c}, \mathbf{h} \rangle = 0 \right\}$$

On appelle *matrice de parité* de $\mathcal{C}$, toute matrice génératrice $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$ de $\mathcal{C}^\perp$ (qui est bien de dimension $n-k$). De cette manière,

$$\mathcal{C} = \{ \mathbf{c} \in \mathbb{F}_q^n : \mathbf{cH}^\intercal = \mathbf{0} \}.$$

On a coutume d'appeler équation de parité de $\mathcal{C}$ tout mot de $\mathcal{C}^\perp$. On dira qu'un vecteur $\mathbf{x} \in \mathbb{F}_q^n$ vérifie une équation de parité $\mathbf{h} \in \mathcal{C}^\perp$ si $\langle \mathbf{x}, \mathbf{h} \rangle = 0$.

De plus, pour une matrice $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ (resp. $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$) de rang k (resp. $n-k$) nous pourrons noter $\langle \mathbf{G} \rangle$ (resp. $\langle \mathbf{H} \rangle^\perp$) le $[n,k]_q$-code de matrice génératrice $\mathbf{G}$ (resp. de parité $\mathbf{H}$).

Tout au long de ce document il nous arrivera de choisir la matrice génératrice ou de parité d'un code $\mathbf{M} \in \mathbb{F}_q^{m \times n}$ de rang $m \leqslant n$ sous forme systématique, *i.e :* $\mathbf{M}$ est égale à une permutation près de ses colonnes à la matrice $\begin{pmatrix} \mathbf{1}_m & \mathbf{M}' \end{pmatrix}$ où $\mathbf{M}' \in \mathbb{F}_q^{m \times (n-m)}$ et $\mathbf{1}_m \in \mathbb{F}_q^{m \times m}$ est la matrice identité.

Première partie

Codes correcteurs et cryptographie : un bref état de l'art

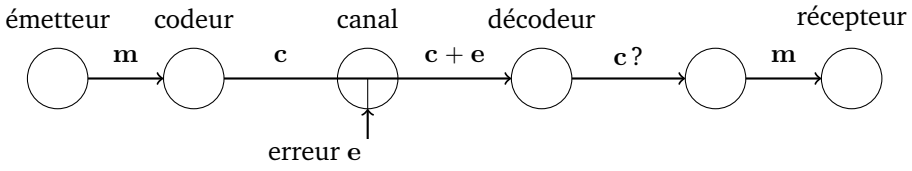
Chapitre 1

Introduction aux codes correcteurs en cryptographie

Une brève histoire des codes correcteurs en cryptographie

L'un des grands tournants de notre époque fut la numérisation de l'information. Cette dernière ouvrit la possibilité de la conservation illimitée de l'information tout comme notre capacité à la transmettre quasi-instantanément. En revanche ceci ne fut possible qu'au prix de la protection contre les erreurs. En effet toute donnée enregistrée sur un support (pensons à nos vieux CD-ROM) ou téléchargée d'un serveur à l'autre bout du monde est susceptible d'être altérée. Le principe pour s'en prémunir est alors simple et naturel : adjoindre de la redondance à chaque symbole que l'on souhaite transmettre ou sauvegarder. Une illustration banale est lorsque nous cherchons à épeler notre nom au téléphone : T comme Thierry, I comme Inès, L comme Léo etc...

Dans un contexte numérique l'idée est la suivante : partons d'un message que l'on souhaite transmettre, à savoir une suite $\mathbf{m}$ formée de k symboles. On commence par se fixer un code $\mathcal{C}$ linéaire de $\mathbb{F}_q^n$ de dimension k ($n \geqslant k$) spécifié par une matrice génératrice $\mathbf{G} \in \mathbb{F}_q^{k\times n}$. On commence alors par calculer l'encodé du message $\mathbf{m}$: $\mathbf{c} \stackrel{\triangle}{=} \mathbf{mG}$ avec $n-k$ symboles de redondance. Le mot de code $\mathbf{c}$ est ensuite transmis à travers un canal pouvant induire des erreurs. On obtient alors un mot de la forme $\mathbf{c}+\mathbf{e}$. Il s'agit maintenant de retrouver $\mathbf{c}$, et donc $\mathbf{m}$, à partir de $\mathbf{c}+\mathbf{e}$. Cette opération est appelée décodage. La figure qui suit résume la situation.



Le seul instant non-maitrisé ici est l'action du canal sur le mot de code. Une première modélisation simple et plutôt réaliste dans le cas où $q = 2$ est celle du canal binaire symétrique : tout bit de $\mathbf{c}$ est indépendamment modifié en son opposé avec une probabilité $0 \leqslant p < 1/2$. Dans ce modèle une donnée naturelle quantifiant les erreurs possibles est le poids de Hamming. On vérifie facilement qu'une erreur donnée $\mathbf{e}$ de poids de Hamming w a pour probabilité d'apparition $p^w(1-p)^{n-w}$. Du fait que $p < 1/2$ cette probabilité est décroissante en w. On en déduit que le mot de code transmis est d'autant plus probable qu'il est proche, *au sens de la métrique de Hamming*, du mot reçu. Le décodage consiste alors

à retrouver le mot de code à la plus petite distance de Hamming du mot reçu. On parle usuellement de décodage par maximum de vraisemblance. La procédure naïve consistant à énumérer les 2^k mots de code possibles et sélectionner le plus proche est évidemment à écarter. Il nous faut donc ajouter une structure sur le code dont on pourra ensuite tirer parti.

L'ambition de la théorie des codes est de proposer des familles de codes avec une structure permettant un algorithme de décodage efficace. Historiquement deux grandes familles furent proposées : (i) les codes munis d'une forte structure algébrique et d'un décodage déterministe comme les codes de Reed-Solomon [MS86, Chapitre 10] ou les codes de Goppa [MS86, Chapitre 12] et (ii) ceux avec des algorithmes de décodage probabiliste comme les codes convolutifs [Eli55], les codes LDPC [Gal63] ou plus récemment les codes polaires [Arı09]. Si toutes ces structures ont été introduites c'est que le problème de décodage d'un code quelconque semble de nature exponentielle, ce que confirment comme nous le verrons près de soixante années de recherche.

Le décodage en cryptographie

McEliece et son chiffrement. L'idée de McEliece [McE78] a alors été d'utiliser ce problème de décodage pour un code quelconque, que l'on nommera *décodage générique*, dans un contexte de cryptographie asymétrique. Le fait est d'autant plus remarquable que sa proposition est légèrement postérieure à celle de RSA [RSA78]. L'idée pour faire du chiffrement est la suivante : le détenteur de la clef secrète dispose d'un code linéaire sur $\mathbb{F}_q$ de longueur n et de dimension k spécifié par une matrice génératrice $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ pour laquelle il connaît une structure sous-jacente secrète et cachée. Cette dernière lui permet de décoder efficacement des erreurs de poids inférieur à une certaine quantité, notée w_{dec}. La clef publique est alors constituée de w_{dec} et de la matrice $\mathbf{G}$. Le chiffré d'un message $\mathbf{m}$ est $\mathbf{mG} + \mathbf{e}$ où $|\mathbf{e}| = w_{\text{dec}}$ avec $|\cdot|$ le poids de Hamming. Le déchiffrement consiste alors tout simplement à décoder le chiffré grâce à la structure cachée pour en déduire $\mathbf{m}$.

A priori dans ce schéma, à moins qu'un attaquant réussisse à se donner un avantage avec la structure tapie dans $\mathbf{G}$, la sécurité repose sur le décodage générique que l'on peut donner comme suit :

Problème 1.1. (Décodage générique.) *Étant donné* $\mathbf{G} \in \mathbb{F}_q^{k\times n}$*,* w *et* $\mathbf{y} \in \mathbb{F}_q^n$*, trouver* $\mathbf{e}$ *de poids de Hamming* exactement w *tel que* $\mathbf{y} - \mathbf{e} = \mathbf{mG}$ *pour un certain* $\mathbf{m} \in \mathbb{F}_q^k$*.*

Ce problème est NP-complet [BMT78] et comme nous le verrons dans le chapitre 2, la complexité moyenne des meilleurs algorithmes (classiques ou quantiques) pour le résoudre est donnée par :

$$2^{cw(1+o(1))}$$

où c est une constante dépendant de la dimension du code et de l'algorithme utilisé. Avec l'approche de McEliece nous ne pouvons donc pas faire mieux en terme de taille de clef que quadratique en le paramètre de sécurité. Par exemple pour atteindre une sécurité de 2^λ, la taille de la clef publique est de l'ordre de $\Theta(\lambda^2)$ si $w = \Theta(n)$.

Néanmoins dans le schéma de McEliece une question cruciale est de savoir quelle structure utiliser, comment bien la cacher et s'en assurer. La théorie des codes est d'une grande richesse. Il y aujourd'hui pléthore de familles de codes avec un bon algorithme de décodage (c'est à dire en mesure de décoder beaucoup d'erreurs). Nous pouvons donc apparemment instancier ce schéma de multiples façons. L'histoire des codes en cryptographie a en revanche démontré à quel point nous devons nous méfier. En effet, l'immense majorité des propositions fut brisée. De nos jours les familles robustes à la cryptanalyse ne sont plus légion. Nous pouvons par exemple citer les codes de Goppa (proposition originelle de McEliece) et les codes MDPC [Mis+12] avec respectivement un algorithme de décodage à distance $n/\log_2(n)$ et $\sqrt{n}$.

Revenons un instant sur l'introduction des codes MDPC, ces derniers étant symptomatiques d'une idée profonde en théorie des codes et cryptographie. Nous savons depuis les codes LDPC que des mots de petits poids dans le dual du code sont d'une grande aide pour le décodage. Les codes LDPC sont aujourd'hui très utilisés en télécommunication et nous pourrions être tenté de les utiliser en cryptographie. Malheureusement, il est trop facile de retrouver les mots de petits poids dans leur dual. Ces derniers sont asymptotiquement en leur longueur n de poids $O(1)$. Le rationnel des codes MDPC consiste alors à augmenter le poids de ces mots sans pour autant trop détériorer l'algorithme de décodage. Ceci peut-être fait jusqu'à des poids de l'ordre de $\sqrt{n}$ tout en ayant une distance de décodage avoisinant $\sqrt{n}$. Ces codes ont ainsi un bien moins bon pouvoir de correction et il serait idiot de les utiliser dans un contexte de télécommunication. En revanche, ces derniers semblent bien adaptés à la cryptographie car retrouver les mots de poids $\sqrt{n}$ est un problème difficile.

Et les signatures ? Concernant les signatures avec des codes correcteurs, si l'on souhaite adopter la stratégie *du hache et signe* reposant sur le problème du décodage l'idée est la suivante :

— *Le signataire* dispose d'une matrice $\mathbf{G}$ pour laquelle il connaît un algorithme de décodage à une distance w_{sgn}.
La signature d'un message $\mathbf{x}$ est alors une solution $\mathbf{e}$ du problème de décodage à distance w_{sgn} avec comme entrée $\mathbf{G}$ et $\mathcal{H}(\mathbf{m})$ où $\mathcal{H}(\cdot)$ est une fonction de hachage cryptographique.

— *La clef publique* (de vérification) est encore une fois $\mathbf{G}$ et le poids w_{sgn}.

— *La vérification* d'un message $\mathbf{x}$ consiste alors à s'assurer que $\mathcal{H}(\mathbf{x}) - \mathbf{e} = \mathbf{mG}$ pour un certain $\mathbf{m} \in \mathbb{F}_q^k$ et $|\mathbf{e}| = w_{\text{sgn}}$.

Malheureusement, *décoder* pour signer dans ce paradigme est très contraignant. En effet, du fait que le mot à décoder est un certain haché, la condition sur le décodage devient : presque tout mot de l'espace peut se décoder efficacement. Comme nous le verrons, le nombre attendu de mots de code à distance w d'un vecteur $\mathbf{y}$ passe selon w de (i) exponentiellement élevé à (ii) exponentiellement faible dans un tout petit intervalle. Plus précisément, c'est dans une fenêtre très étroite autour du poids dit de "Gilbert-Vashamov" $w_{\text{GV}} = \Theta(n)$ que nous nous attendons typiquement à une unique solution tandis que pour des poids inférieurs à $(1-\varepsilon)w_{\text{GV}}$ pour $\varepsilon > 0$ une solution existe avec une probabilité exponentiellement faible. Or c'est précisément dans ces zones de poids faible qu'ont été développés les algorithmes de décodage. Il semble donc difficile de suivre une approche à la McEliece avec des codes correcteurs pour fabriquer une signature dans le paradigme du hache et signe.

Une des contributions de [CFS01] a été de remarquer qu'en tordant suffisamment les paramètres des codes de Goppa, plus précisément en choisissant leur dimension proche de leur longueur, ces derniers sont en mesure de décoder à une distance légèrement inférieure à w_{GV}. La densité des mots "décodables" est alors suffisamment proche de 1 pour tenter de décoder $\mathcal{H}(\mathbf{m}, i)$ et incrémenter i si cela n'a pas fonctionné (la signature est alors $(\mathbf{e}, i)$). Malheureusement le système de [CFS01] souffre d'un problème de passage à l'échelle. De plus, comme prouvé dans [Fau+10a] un code de Goppa de grande dimension est distinguable d'un code quelconque provoquant un problème de structure. D'autres propositions de codes [Bal+13 ; Gli+14 ; Lee+17] furent alors faites pour instancier le schéma de CFS mais toutes furent cassées (voir par exemple [PT16 ; MP16]).

L'approche consistant à chercher des codes correcteurs avec un poids de décodage suffisamment proche de w_{GV} semble donc difficile à faire aboutir. L'idée naturelle pour se défaire d'un tel carcan est de relâcher la contrainte d'un algorithme de décodage stricto sensu et donc l'existence d'un unique mot de code à la bonne distance comme cela fut décrit pour la première fois dans [OT12]. Le paradigme n'est plus le même, on

se place dans des zones de paramètres où pour tout mot de l'espace ambiant il existe un nombre exponentiel de mots de code à une distance fixée w. On cherche alors à retrouver efficacement pour tout mot de l'espace ambiant *l'un des mots de codes* à distance w. Il ne s'agit plus formellement de décodage mais plutôt de distorsion comme décrit en théorie des codes. Le souci étant maintenant que peu de familles de codes viennent avec un algorithme efficace pour résoudre ce problème. Nous pouvons citer les codes convolutifs ou encore les codes LDGM [WMM10] mais ces derniers ne résistent pas à la cryptanalyse tout comme la suggestion des codes polaires dans [OT12]. Bien que non-sure cette dernière fut le prémisse de la proposition des codes $(U, U+V)$ généralisés en signature, contribution de cette thèse décrite dans la partie III.

Dans ce document nous discuterons peu des questions de structure avec l'approche de McEliece mais quoiqu'il en soit nous souhaiterions nous passer de cette épine en proposant un schéma reposant uniquement sur le problème du décodage générique.

Le décodage et rien que le décodage

De nos jours, trois grandes approches ont mené à différentes primitives reposant uniquement sur le décodage générique : (i) les chiffrements d'Alekhnovich [Ale11], (ii) la fonction de hachage FSB [Aug+08] et (iii) le protocole d'identification de Stern [Ste93b] puis de Véron [Vér96] qui, associés à la transformation de Fiat-Shamir [FS87] et diverses optimisations [AGS11], donnent un schéma de signature. Penchons-nous un peu plus en détail sur l'idée d'Alekhnovich.

Afin de se réduire uniquement au problème du décodage générique, Alekhnovich propose contrairement à McEliece de partir d'un code aléatoire $\mathcal{C}$ pour lequel on ne connaît pas d'algorithme de décodage. Soit $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ une matrice génératrice de $\mathcal{C}$. On commence par calculer un mot de code bruité à une certaine distance w : $\mathbf{y} \stackrel{\triangle}{=} \mathbf{mG} + \mathbf{e}$ où $|\mathbf{e}| = w$. Le poids w est cependant choisi faible :

$$w = o(\sqrt{n})$$

L'idée consiste alors à rendre publique la matrice génératrice

$$\begin{pmatrix} \mathbf{G} \\ \mathbf{y} \end{pmatrix}$$

du code engendré par $\mathcal{C}$ et $\mathbf{y}$, noté $\mathcal{C}_{\text{pub}}$, tandis que la clef privée est tout simplement $\mathbf{e}$. Remarquons ici que le code $\mathcal{C}_{\text{pub}}$ est engendré par $\mathcal{C}$ et $\mathbf{e}$. Il est clair que retrouver la clef privée à partir de la clef publique correspond exactement à résoudre le problème du décodage générique à distance w. Le chiffrement, de bits, avec une telle construction se fait de la façon suivante :

Chiffrement. Pour chiffrer 1 on renvoie $\text{Enc}(1) \stackrel{\triangle}{=} \mathbf{u}$ un vecteur aléatoire de longueur n tandis que pour chiffrer 0 on calcule $\text{Enc}(0) \stackrel{\triangle}{=} \mathbf{c}^\perp + \mathbf{e}'$ où $\mathbf{c}^\perp$ est un mot de $\mathcal{C}_{\text{pub}}^\perp$ et $\mathbf{e}'$ est une erreur de poids w.

Déchiffrement. Le déchiffrement du bit $\text{Enc}(m)$ correspond tout simplement à calculer $\langle \mathbf{e}, \text{Enc}(m)\rangle$. La correction d'une telle procédure repose alors sur la remarque suivante :

$$\langle \mathbf{e}, \text{Enc}(0)\rangle = \langle \mathbf{e}, \mathbf{c}^\perp + \mathbf{e}'\rangle = \langle \mathbf{e}, \mathbf{e}'\rangle$$

car $\mathbf{e} \in \mathcal{C}_{\text{pub}}$ et $\mathbf{c}^\perp \in \mathcal{C}_{\text{pub}}^\perp$. Nous pouvons alors démontrer que :

$$\mathbb{P}\left(\langle \mathbf{e}, \mathbf{e}'\rangle = 0\right) = 1 - \frac{1}{2}\left(1 - 2^{-c\cdot w^2/n(1+o(1))}\right)$$

pour une certaine constante $c > 0$. Donc étant donné que w a été choisi en $o(\sqrt{n})$, le déchiffrement de 0 réussira avec une probabilité $1 - o(1)$. Le déchiffrement de 1 donnera quant à lui un bit aléatoire. Il suffit donc pour chiffrer un bit de recommencer l'opération à plusieurs reprises.

L'avantage d'un tel schéma est que sa sécurité repose uniquement sur le problème du décodage générique. En revanche la situation est plutôt dramatique concernant la taille de clef qui sera de l'ordre de λ^4 pour une sécurité de 2^λ du fait que $w = o(\sqrt{n})$. La situation est un peu plus enviable, sans être totalement satisfaisante, du côté du système de McEliece où les clefs sont quand même au mieux quadratique en le paramètre de sécurité $O(\lambda^2)$. Fort heureusement, des stratégies pour éviter cet écueil ont été développées.

Réduction de taille de clef : des codes avec un grand groupe d'automorphismes

Les codes avec un grand groupe d'automorphismes. Les résultats de [Gab05] furent le commencement d'une longue série de travaux concernant une réduction de la taille des clefs. Il est proposé dans [Gab05] de choisir des codes $\mathcal{C}$ sur le corps $\mathbb{F}_q$ avec un suffisamment grand groupe d'automorphismes $\mathrm{Aut}(\mathcal{C})$ défini comme :

$$\mathrm{Aut}(\mathcal{C}) \stackrel{\triangle}{=} \left\{\sigma : \mathbb{F}_q^n \to \mathbb{F}_q^n \text{ isométrie linéaire telle que } \sigma(\mathcal{C}) = \mathcal{C}\right\}$$

où l'isométrie est ici au sens de la métrique de Hamming. L'idée est de pouvoir représenter un code de dimension k avec peu de vecteurs (par exemple un si $\mathrm{Aut}(\mathcal{C})$ est cyclique) au lieu de k. Plus généralement on s'attend à réduire la représentation du code d'un facteur donné par l'ordre du groupe d'automorphismes.

Étant donné qu'un code quelconque a typiquement un groupe d'automorphismes trivial il nous faut choisir une structure supplémentaire à la linéarité. Deux grandes familles de codes furent proposées en cryptographie :

— les codes quasi-cycliques [Gab05 ; BC07 ; Ber+09 ; Mis+12], *i.e :* les codes ayant un sous-groupe d'automorphismes cyclique,

— les codes quasi-dyadiques, [MB09], *i.e :* les codes ayant un sous-groupe d'automorphismes isomorphe à $(\mathbb{Z}/2\mathbb{Z})^t$.

A titre d'exemple, un code $\mathcal{C}$ sur $\mathbb{F}_q^n$ pour lequel il existe un $s \mid n$ tel que :

$$(c_1, \cdots, c_n) \in \mathcal{C}, \text{ alors } \quad (c_s, c_1, \cdots, c_{s-1}, \cdots, c_n, c_{n-s+1} \cdots, c_{n-1}) \in \mathcal{C}$$

est un code quasi-cyclique d'ordre s. Cette solution, bien qu'élégante, doit cependant être maniée avec parcimonie. En effet, la proposition de sous-codes BCH [MS86, Chapitre 6] dans [Gab05] comme codes quasi-cycliques a été complètement cassée dans [OTD10]. En revanche, seuls les paramètres de [Ber+09 ; MB09] utilisant des codes de Goppa avec une structure respectivement quasi-cyclique et quasi-dyadique ont été attaqués par [Fau+10b]. La cryptanalyse fut donc sévère à l'encontre des codes algébriques auxquels un grand groupe d'automorphismes est ajouté. Inversement les codes MDPC [Mis+12] quasi-cycliques ou leur ancêtre [BC07] (légèrement plus structuré) n'utilisant pas de structure algébrique ont jusque-là résisté aux attaques.

Décodage générique des codes au grand groupe d'automorphismes. Au-delà de gagner sur la taille de représentation, un autre intérêt des codes avec un grand groupe d'automorphismes en cryptographie est que leur décodage générique ne semble pas plus facile que le décodage des codes aléatoires. En général les algorithmes ne gagnent qu'un facteur polynomial de l'ordre de la taille du groupe. Il y a donc un véritable gain en terme de taille de clef. Seul le récent travail de [CT19] a permis un gain exponentiel pour

certaines zones de paramètres en utilisant le concept de repliement évoqué dans [Leg11]. L'idée étant pour un code $\mathcal{C}$ de groupe d'automorphismes $\mathbb{G}$ de chercher à décoder non pas directement dans $\mathcal{C}$ mais dans le code dit "replié". Ce code est formé des $\mathbf{c}^{\mathbb{G}}$ définis comme les mots obtenus à partir des $\mathbf{c} \in \mathcal{C}$ où l'on a sommé toutes les coordonnées appartenant à une même orbite sous l'action de $\mathbb{G}$. Décoder $\mathbf{c} + \mathbf{e}$ revient alors à décoder $\mathbf{c}^{\mathbb{G}} + \mathbf{e}^{\mathbb{G}}$. Avec le repliement on divise la longueur par le nombre d'orbites tout en conservant le même rapport de dimension sur longueur mais en augmentant le poids de l'erreur. Ceci donne un nouvel équilibre de paramètres pouvant être utilisé à bon escient. Quoiqu'il en soit le gain étant a priori faible il y a un véritable intérêt à trouver les bonnes structures de codes avec un grand groupe d'automorphismes.

Réduction de taille de clef : changer la métrique, la métrique rang

Les codes matriciels et leur contexte. Le choix de codes avec un grand groupe d'automorphismes est de façon surprenante et détournée apparue dans un autre contexte. Comme nous l'avons évoqué un peu plus haut, dans un modèle classique de transmission de symboles, chercher le mot envoyé revient à trouver le mot de code le plus proche du mot reçu au sens de la métrique de Hamming. Il existe cependant des contextes où l'on cherche, non pas à transmettre une suite de symboles, mais des $\mathbb{F}_q$-espaces vectoriels. Autrement dit, on transmet une matrice, disons de $\mathbb{F}_q^{m\times n}$, qui est ensuite perturbée. On considère donc des codes matriciels, c'est à dire des sous-espaces vectoriels de $\mathbb{F}_q^{m\times n}$. Ainsi, la métrique naturelle pour retrouver le mot de code envoyé est la métrique rang : la distance entre deux matrices est le rang de leur différence. De cette manière, décoder une matrice signifie retrouver la matrice du code la plus proche pour la métrique rang.

Décoder en métrique rang ? Les codes $\mathbb{F}_{q^m}$-linéaires. Il s'agit donc maintenant de trouver des codes matriciels munis d'un bon algorithme de décodage. Trois familles de codes furent proposées : les codes de Gabidulin [Gab85] similaires aux codes de Reed-Solomon, les codes simples [SKK10] et les codes LRPC [Gab+13]. Fait remarquable, à l'exception des codes simples, ces derniers ont un grand groupe d'automorphismes, *i.e :* un large groupe d'isométries linéaires pour la métrique rang. Au lieu de partir d'un code matriciel de $\mathbb{F}_q^{m\times n}$ on part d'un code $\mathbb{F}_{q^m}$-linéaire sur $\mathbb{F}_{q^m}^n$. Le corps $\mathbb{F}_{q^m}$ étant un extension de $\mathbb{F}_q$ de degré m, on peut écrire tout élément $x \in \mathbb{F}_{q^m}$ comme un vecteur colonne de $\mathbb{F}_q^{m\times 1}$ (évidemment à base fixée). Il est alors clair que tout mot $\mathbf{x} \in \mathbb{F}_{q^m}^n$ peut s'écrire comme une matrice sur $\mathbb{F}_q^{m\times n}$, notée $\mathrm{Mat}(\mathbf{x})$. Un code $\mathbb{F}_{q^m}$-linéaire sur $\mathbb{F}_{q^m}^n$ peut donc se voir comme un code matriciel sur $\mathbb{F}_q^{m\times n}$. On munit alors ces codes de la métrique rang en les écrivant comme codes matriciels. En revanche le fait qu'un code $\mathbb{F}_{q^m}$-linéaire $\mathcal{C}$ ait un groupe d'automorphismes non trivial n'est pas ici transparent. Cela découle de la $\mathbb{F}_{q^m}$-linéarité :

$$\text{si } \mathbf{c} \in \mathcal{C}, \text{ alors pour tout } \alpha \in \mathbb{F}_{q^m},\ \alpha \cdot \mathbf{c} \in \mathcal{C}.$$

L'opérateur $\mathrm{Mat}(\mathbf{c}) \mapsto \mathrm{Mat}(\alpha \cdot \mathbf{c})$ est linéaire et isométrique si $\alpha \neq 0$, $\mathrm{Rang}(\mathrm{Mat}(\alpha \cdot \mathbf{c})) = \mathrm{Rang}(\mathrm{Mat}(\mathbf{c}))$. Ce large groupe d'automorphismes pour les codes matriciels issus de codes $\mathbb{F}_{q^m}$-linéaire permet alors de les représenter de façon bien plus réduite (d'un facteur m).

Nous pouvons donc dans ce contexte naturellement instancier le schéma de McEliece avec des codes $\mathbb{F}_{q^m}$-linéaires. Cela fut pour la première fois fait dans [GPT91] où il a été proposé d'utiliser des codes de Gabidulin. Les paramètres proposés étaient alors très attrayants. Malheureusement l'attaque de [Ove05] montra que ces codes souffrent d'une structure trop importante. Les résultats de [Ove05] mirent ainsi un coup d'arrêt temporaire à l'utilisation de la métrique rang en cryptographie.

Décodage générique des codes $\mathbb{F}_{q^m}$-linéaires. Quoiqu'il en soit, si nous passons outre ces questions de structure sous-jacente, le système de McEliece avec des codes $\mathbb{F}_{q^m}$-linéaire se réduit au problème suivant :

Problème 1.2 (Décodage générique d'un code $\mathbb{F}_{q^m}$-linéaire en métrique rang)**.** *Étant donné* $\mathbf{G} \in \mathbb{F}_{q^m}^{k\times n}$*,* w *et* $\mathbf{y} \in \mathbb{F}_{q^m}^n$*, trouver* $\mathbf{e} \in \mathbb{F}_{q^m}^n$ *de poids rang* w *tel que* $\mathbf{y} - \mathbf{e} = \mathbf{mG}$ *pour un certain* $\mathbf{m} \in \mathbb{F}_{q^m}^k$*.*

Ce problème de décodage en métrique rang peut être vu comme instanciation particulière du problème MinRank :

Problème 1.3 (MinRank)**.** *Étant donné* $\mathbf{M}_1, \cdots, \mathbf{M}_k \in \mathbb{F}_q^{m\times n}$*, un poids* w *et* $\mathbf{S} \in \mathbb{F}_q^{m\times n}$*, trouver* $\lambda_1, \cdots, \lambda_k \in \mathbb{F}_q$ *tels que* $\mathrm{Rang}(\sum_{i=1}^k \lambda_i \mathbf{M}_i - \mathbf{S}) = w$*.*

En effet, pour réduire une instance du décodage en poids rang à MinRank il suffit d'écrire chaque ligne de $\mathbf{G}$ ainsi que $\mathbf{y}$ comme une matrice de $\mathbb{F}_q^{m\times n}$. Le problème revient alors exactement à trouver une $\mathbb{F}_q$-combinaison linéaire des matrices obtenues à partir des lignes de $\mathbf{G}$ à une distance rang w de la matrice définie par $\mathbf{y}$: le problème MinRank. Cependant permettons-nous d'insister, le problème 1.2 n'est pas équivalent au problème 1.3, les matrices obtenues à partir de $\mathbf{G}$ forment un code de $\mathbb{F}_q^{m\times n}$ structuré du fait de la $\mathbb{F}_{q^m}$-linéarité du code défini par $\mathbf{G}$. Une première différence fondamentale apparait sur le caractère NP-complet : Minrank l'est [Cou01] alors que pour le problème 1.2 on ne connait qu'une réduction probabiliste polynomiale au problème du décodage en métrique de Hamming [GZ16].

En revanche, de façon similaire au problème du décodage en métrique de Hamming, l'existence d'un grand groupe d'automorphismes en métrique rang ne semble pas aider pour le décodage. Il est a priori difficile d'utiliser la $\mathbb{F}_{q^m}$-linéarité pour mieux résoudre le problème 1.2 que MinRank. Les premiers algorithmes [OJ02] résolvant le décodage générique en métrique rang reviennent à décoder un code matriciel sur $\mathbb{F}_q^{m\times n}$ à l'aide d'algèbre linéaire. Une telle approche donne alors des complexités pour décoder à distance w des codes de dimension k de l'ordre de :

$$q^{wk(1+o(1))}$$

L'utilisation des codes $\mathbb{F}_{q^m}$-linéaires en cryptographie donne donc un véritable avantage sur les codes matriciels en terme de taille de clef. De nombreuses investigations furent faites pour essayer de tirer profit de la $\mathbb{F}_{q^m}$-linéarité. Par exemple [GRS16] améliora les résultats de [OJ02] dans le cas où le degré de l'extension m est inférieur à n. Un des algorithmes proposés dans [GRS16] peut-être vu comme une adaptation du plus simple des algorithmes résolvant le problème du décodage en métrique de Hamming [Pra62]. En revanche, toutes les tentatives pour adapter les algorithmes de décodage améliorant [Pra62] au cas des codes $\mathbb{F}_{q^m}$-linéaires s'avérèrent infructueuses. Signalons cependant l'amélioration récente de [Ara+17b] où un algorithme d'une autre nature et de complexité donnée dans le cas standard $m = n$ par $q^{(w(k+1)-m)(1+o(1))}$ fut introduit.

Il existe cependant une autre approche que l'algèbre linéaire. Repartons du problème MinRank, une résolution classique consiste à utiliser des techniques de bases de Gröbner [KS99 ; FLP08 ; Spa12]. On peut dès-lors appliquer cette méthode au décodage d'un code $\mathbb{F}_{q^m}$-linéaire. Il n'existe pas à ce jour de travaux tirant profit de cette linéarité à l'exception de la mise en équation du problème en système bilinéaire dans [Ara+17c]. En revanche, l'étude de la complexité de la résolution de ce système ne tient pas compte de la structure sous-jacente de la $\mathbb{F}_{q^m}$-linéarité.

Dans tous les cas, l'exposant dominant des attaques génériques reste de l'ordre de nm, longueur du code matriciel obtenu à partir du code $\mathbb{F}_{q^m}$-linéaire, quand la distance de décodage et la dimension du code sont linéaires en $\min(n, m)$.

Les codes LRPC. Une question cruciale dans ce contexte est quelle structure introduire pour instancier le système de McEliece ? Une réponse fut donnée avec l'introduction des codes LRPC [Gab+13]. Ces derniers ont d'une certaine façon suivi l'idée des codes MDPC. Le décodage des codes LRPC tire parti de mots de petits poids dans le dual. L'idée est la suivante : on part d'un code $\mathbb{F}_{q^m}$-linéaire $\mathcal{C}$ de dimension k et de longueur n tel que son dual admette une base $(\mathbf{c}_1^\perp, \cdots, \mathbf{c}_{n-k}^\perp)$ pour laquelle il existe un sous $\mathbb{F}_q$-espace vectoriel $F \subset \mathbb{F}_{q^m}$ de petite dimension d telles que toutes les coordonnées des $\mathbf{c}_i^\perp$ sont dans F, *i.e* : pour tout $i \in [\![1, n-k]\!]$ et $j \in [\![1, n]\!]$ nous avons, :

$$\mathbf{c}_i^\perp(j) \in F.$$

Cette structure permet alors un décodage efficace fonctionnant avec probabilité $1_{,,} - q^{-(n-k+1-wd)}$ pour tous w et d tels que $wd \leqslant \min(m, n-k)$. Nous retrouvons d'un certaine façon la condition de décodage des codes MDPC ou du chiffrement d'Alekhnovich où le poids de décodage est en racine carré de la longueur. Le rationnel du décodage des codes LRPC est le suivant. Pour un mot de code bruité $\mathbf{y} \stackrel{\triangle}{=} \mathbf{c} + \mathbf{e}$ à distance w, on commence par calculer pour tout $i \in [\![1, n-k]\!]$:

$$\langle \mathbf{c}_i^\perp, \mathbf{y} \rangle = \langle \mathbf{c}_i^\perp, \mathbf{e} \rangle.$$

Sous la condition $wd \leqslant \min(m, n-k)$ (d est petit), du fait que les coordonnées des $\mathbf{c}_i^\perp$ *appartiennent toutes au même $\mathbb{F}_q$-espace vectoriel F*, ces produits scalaires engendrent, avec une très bonne probabilité, le $\mathbb{F}_q$-espace produit $E \cdot F$. Ce dernier est engendré sur $\mathbb{F}_q$ par $\{ef : e \in E, f \in F\}$ où E est le $\mathbb{F}_q$-espace engendré par $e_1, \cdots, e_n$ qui est de dimension w. Maintenant, connaissant F on en déduit E qui donne facilement $\mathbf{e}$ en résolvant un système linéaire sur $\mathbb{F}_q$.

Au-delà de venir avec un bon algorithme de décodage, les codes LRPC semblent bien adaptés à la cryptographie. En effet, l'idée naturelle pour retrouver leur structure cachée est de chercher des mots de petit rang dans leur dual. La meilleure solution est alors a priori d'utiliser les algorithmes de décodage générique. Les codes LRPC ont jusque-là résisté à la cryptanalyse. A ce jour il n'existe qu'une seule attaque contre ces codes en cryptographie et de surcroît pour des paramètres très particuliers uniquement utilisés avec la signature RankSign [Gab+14b]. Il s'agit d'une contribution de cette thèse décrite dans la partie IV.

Quoiqu'il en soit, l'idée des codes LRPC de choisir comme secret un espace sous-jacent de petite dimension ouvrit la voie à de nombreuses solutions cryptographiques avec l'introduction dans [Gab+17b] du problème *Rank Support Learning* (RSL). Ce-dernier consiste à se donner N mots de codes bruités où les erreurs partagent une structure commune, toutes les coordonnées des erreurs appartiennent à un même espace vectoriel. Plus formellement,

Problème 1.4 (RSL)**.** *Étant donné un code $\mathcal{C}$ sur $\mathbb{F}_{q^m}$ de dimension k, w, N, un sous-espace vectoriel $F \subseteq \mathbb{F}_{q^m}$ de dimension w et $\mathbf{c}_1 + \mathbf{e}_1, \cdots, \mathbf{c}_N + \mathbf{e}_N$ où les $\mathbf{c}_i \in \mathcal{C}$ et les $\mathbf{e}_i \in \mathbb{F}_{q^m}^n$ sont telles que $\mathbf{e}_i(j) \in F$ pour tout i, j, trouver F.*

La difficulté de ce problème décroit clairement avec N. Nous retrouvons dans le cas où $N = 1$ le décodage générique d'un code $\mathbb{F}_{q^m}$-linéaire à distance $w = \dim(F)$ alors que pour $N = wk$ il existe une attaque polynomiale décrite dans [Gab+17b]. La question d'ordre cryptographique est alors jusqu'à quel point pouvons-nous faire grossir N tout en nous assurant que le problème reste difficile ? Dans l'état de l'art actuel nous ne savons pas faire mieux que $N = wk$. La situation est alors favorable si nous comparons à la métrique de Hamming. En effet, l'équivalent du problème RSL, que nous noterons HSL, correspond dans ce cas à générer des erreurs dans $\mathbb{F}_q^n$ partageant toutes le même support de positions. Il existe une attaque polynomiale dès lors que $N = w$ [Gab+17b, §4.2]. De plus, ce problème est directement utilisé dans les schémas de signature en métrique de Hamming [KKS97 ;

KKS05] et ses variantes [BMS11 ; GS12 ; Fuk+17]. Ces-derniers peuvent cependant être au mieux considérés comme des signatures uniques à la lumière des attaques [COV07 ; OT11 ; Hue+18]. Ces-dernières utilisent alors de façon cruciale la structure induite par le problème HSL. Il semble donc qu'il y ait une véritable différence à ce niveau entre la métrique rang et de Hamming.

Le problème RSL permit de concevoir des primitives cryptographiques en métrique rang : la signature Durandal [Ara+19a] adaptant la technique de Schnorr-Lyubashevsky [Sch91 ; Lyu09a] ou encore le premier IBE [Gab+17b] utilisant des codes et inspiré de celui de [GPV08b]. En revanche, comme montré dans la partie IV de ce document, les paramètres de l'IBE [Gab+17b] sont aujourd'hui cassés.

Le problème du décodage version syndrome

Revenons maintenant un instant sur le problème du décodage générique. Notre vision classique de ce problème se fera tout au long de ce document avec une matrice de parité $\mathbf{H}$ du code (et non une matrice génératrice).

Problème 1.5. *Étant donné* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}, w$ *et* $\mathbf{s} \in \mathbb{F}_q^{n-k}$*, trouver* $\mathbf{e} \in \mathbb{F}_q^n$ *de poids* exactement w *tel que* $\mathbf{eH}^\intercal = \mathbf{s}$.

Le poids désigne ici la métrique de Hamming ou la métrique rang que nous définirons formellement dans §1.2.1. On a l'habitude de nommer syndrome tout vecteur de $\mathbb{F}_q^{n-k}$. Les problèmes 1.1 et 1.2 sont alors équivalents à 1.5 selon le poids utilisé : si l'on sait résoudre l'un alors on sait résoudre l'autre en essentiellement le même temps et réciproquement. Pour un code de matrice génératrice $\mathbf{G}$ et de parité $\mathbf{H}$, cela se prouve en remarquant (i) que $\mathbf{y} - \mathbf{e} = \mathbf{mG}$ pour un certain $\mathbf{m}$ si et seulement si $\mathbf{yH}^\intercal = \mathbf{eH}^\intercal$ et (ii) que pour tout syndrome $\mathbf{s}$ on peut calculer à l'aide d'algèbre linéaire un mot $\mathbf{x}$ tel que $\mathbf{xH}^\intercal = \mathbf{s}$.

Dans la suite nous considèrerons uniquement le problème de décodage sous la forme matrice de parité et syndrome. De plus, nous appellerons *algorithme de décodage* ou *décodeur* tout algorithme permettant de résoudre le problème du décodage et nous y adjoindrons le terme *générique* dès lors qu'aucune hypothèse n'aura été faite sur le code.

1.1 Le décodage générique en cryptographie avec la métrique de Hamming

1.1.1 Difficulté dans le pire cas

La version décisionnelle du problème de décodage en métrique de Hamming a été prouvée NP-complète dans [BMT78]. Ceci donne l'existence d'un pire cas pouvant se résumer de la façon suivante : il existe une suite (de taille croissante) de matrices de parité, de syndromes et de poids pour lequel tout algorithme polynomial en la taille des instances échouera sous l'hypothèse que P $\neq$ NP. Cependant remarquons que le poids de décodage fait partie de l'instance du problème. Une des premières contributions de cette thèse est de montrer que le problème reste NP-complet même si le poids est seulement supposé linéaire en la taille du code.

Proposition 1.1 (ε-SD)**.** *Soit* $\varepsilon \in]0,1[$*. Le problème de décider si pour* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ *et* $\mathbf{s} \in \mathbb{F}_2^{n-k}$ *il existe* $\mathbf{e} \in \mathbb{F}_2^n$ *de poids de Hamming* $\lfloor \varepsilon n \rfloor$ *tel que* $\mathbf{eH}^\intercal = \mathbf{s}$ *est* NP*-complet.*

La NP-complétude de ce problème peut sembler intrigante. En effet, considérons le cas où $\varepsilon = \frac{1}{2}$. Si nous calculons une solution à l'aide d'algèbre linéaire cette-dernière sera avec une probabilité non-négligeable de poids $\lfloor \frac{1}{2} n \rfloor$ ce qui semble contredire le caractère

NP-complet du problème. En d'autres termes, pour $\varepsilon = \frac{1}{2}$ le problème est facile en moyenne tout en restant NP-complet ce qui n'est en rien une contradiction. En effet, la probabilité est faite sur la matrice tandis que le pire cas signifie qu'il existe une matrice.

La preuve de la proposition 1.1 (tout comme la NP-complétude du décodage) repose sur le problème NP-complet qui suit.

Problème 1.6 (Mariage Tri-dimensionnel (MTD))**.**

— Instance : *un ensemble fini T et un sous-ensemble $U \subseteq T \times T \times T$.*

— Décision : *il existe $V \subseteq U$ tel que $|V| = |T|$ et pour tout $(x_1, y_1, z_1), (x_2, y_2, z_2) \in V$ nous avons $x_1 \neq x_2, y_1 \neq y_2$ et $z_1 \neq z_2$.*

Le formalisme de ce problème est alors très lié au décodage. En effet, nous pouvons l'écrire à l'aide d'une matrice d'incidence. L'idée est la suivante : on commence par prendre la première coordonnée de chaque élément de U pour construire la matrice d'incidence relativement à T de taille $|T| \times |U|$. On effectue cette opération pour chaque coordonnée ce qui nous donne trois matrices d'incidence que l'on concatène ensuite verticalement. On obtient alors en temps polynomial sur les entrées une matrice de taille $3|T| \times |U|$. Cette dernière est appelée la matrice d'incidence de l'instance de MTD considérée. Il est maintenant clair qu'il existe une solution au problème MTD associé si et seulement s'il existe $|T|$ colonnes de cette matrice se sommant dans $\mathbb{N}$ au vecteur tout à 1. Afin d'illustrer cette discussion, donnons un exemple.

Exemple 1.1. Considérons $T = \{1, 2, 3\}$ et $U = \{u_1, u_2, u_3, u_4, u_5\}$ avec :

$$u_1 = (1, 1, 2), \quad u_2 = (2, 3, 1), \quad u_3 = (1, 2, 3)$$

$$u_4 = (3, 1, 2) \quad \text{et} \quad u_5 = (2, 2, 2).$$

Notre construction donne la matrice suivante :

	112	231	123	312	222
1	1	0	1	0	0
2	0	1	0	0	1
3	0	0	0	1	0
1	1	0	0	1	0
2	0	0	1	0	1
3	0	1	0	0	0
1	0	1	0	0	0
2	1	0	0	1	1
3	0	0	1	0	0

Les colonnes $2, 3$ et 4 se somment dans $\mathbb{N}$ au vecteur tout à un. Une solution est alors donnée par $V = \{u_2, u_3, u_4\}$.

Cette vision du problème MTD avec une matrice d'incidence est finalement particulièrement utile avec le lemme suivant.

Lemme 1.1. *Soient T et $U \subseteq T \times T \times T$ une instance du problème* MTD*. Notons $\mathbf{H}_{\text{MTD}} \in \mathbb{F}_2^{3|T| \times |U|}$ la matrice d'incidence associée à cette instance. Nous avons :*

$$\textit{l'instance } T, U \textit{ admet une solution} \iff \exists \mathbf{e} \in \mathbb{F}_2^{|U|} \ : \ |\mathbf{e}| = |T| \textit{ et } \mathbf{e}\mathbf{H}_{\text{3MTD}}^{\intercal} = \mathbf{1}.$$

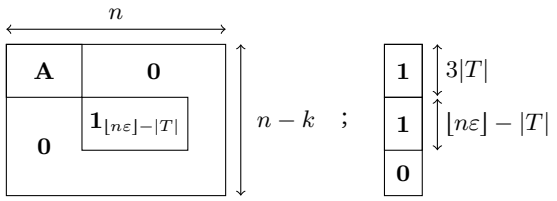
Démonstration du lemme 1.1.

Supposons que l'instance T, U admette une solution. Par construction de la matrice d'incidence, cela implique qu'il existe $|T|$ vecteurs colonnes dont les supports sont deux à deux disjoints. Or chacune de ces colonnes a un poids 3. Donc si nous les sommons dans $\mathbb{N}$ nous obtenons un vecteur de poids $3|T|$. De plus, les vecteurs colonnes sont de taille $3|T|$. La somme est donc le vecteur tout à 1 dans $\mathbb{N}$ et donc dans $\mathbb{F}_2$.

Inversement, supposons qu'il existe une somme de $|T|$ colonnes de la matrice $\mathbf{H}_{\text{3MTD}}$ dans $\mathbb{F}_2$ égale au vecteur tout à 1. Par définition de la matrice d'incidence, chaque colonne est un vecteur de longueur $3|T|$ et contient exactement trois coordonnées à 1. Il est alors clair que la seule façon pour que la somme de $|T|$ colonnes donne un vecteur de poids $3|T|$ dans $\mathbb{F}_2$ est que ces-dernières soient de support disjoint ce qui donne bien une solution du problème 3MTD considéré. □

Démonstration de la Proposition 1.1.

Soit $\varepsilon \in]0,1[$ et notons ε-SD le problème considéré dans la proposition. Il est clair que ε-SD$\in$NP. Nous allons montrer que le problème MTD s'y réduit en temps polynomial. Soit U, T une instance de MTD. Commençons par calculer (en temps polynomial) la matrice $\mathbf{A}$ d'incidence associée de taille $3|T| \times |U|$. Considérons maintenant la matrice et le syndrome suivant que nous noterons $\mathbf{H}$ et $\mathbf{s}$:



Cette construction est valide sous les conditions :

$$n \geqslant |U| + \lfloor n\varepsilon \rfloor - |T|, \quad k \leqslant n, \quad \lfloor n\varepsilon \rfloor - |T| \leqslant n - k - 3|T| \quad \text{et} \quad \lfloor n\varepsilon \rfloor - |T| \geqslant 0$$

Pour cela il suffit de choisir $n = 1 + \lceil \max\left(\frac{|T|}{\varepsilon}, \frac{|U|-|T|}{1-\varepsilon}, \frac{2|T|}{1-\varepsilon}\right) \rceil$ et ensuite un k qui convient. La matrice et le syndrome obtenus sont bien de taille polynomiale en $|T|$ et $|U|$.

Il est maintenant clair que l'existence d'une solution pour le problème MTD implique l'existence d'une solution pour l'instance de ε-SD considéré avec $\mathbf{H}$ et $\mathbf{s}$. La réciproque se fait en observant que les colonnes de $\mathbf{A}$ sont de poids 3. □

1.1.2 Le décodage générique en métrique de Hamming

Les résultats que nous venons d'énoncer donnent la difficulté dans le pire cas du décodage. Comme nous l'avons évoqué en introduction, ce problème est un bon candidat pour être utilisé en cryptographie et afin de l'étudier plus en détail formalisons-le.

Problème 1.7. (Problème du décodage générique de syndromes - SD(n, q, R, ω)) *Soient les entiers* $k \triangleq \lfloor Rn \rfloor$ *et* $w \triangleq \lfloor \omega n \rfloor$.

— Instance : *une matrice de parité* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *de rang* $n-k$*, un syndrome* $\mathbf{s} \in \mathbb{F}_q^{n-k}$.

— Recherche : *un vecteur d'erreur* $\mathbf{e} \in \mathbb{F}_q^n$ *de poids de Hamming* w *tel que* $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$.

Ce problème est paramétrisé par la taille q du corps, le rendement $R \in]0,1[$ du code considéré et le poids relatif de décodage ω qui dans les diverses applications sera une constante dans $]0,1[$ ou une fonction de n tendant vers 0. Nous nous intéresserons dans cette thèse à la *complexité asymptotique* de ce problème, c'est à dire pour un algorithme donné $\mathcal{A}$ résolvant SD(n, q, R, ω) au coefficient $\alpha(q, R, \omega)$ (pouvant être une fonction de n), que l'on nomme usuellement *l'exposant asymptotique*, telle que la complexité de $\mathcal{A}$ est donnée pour $n \to +\infty$ par :

$$2^{n \cdot \alpha(q,R,\omega)(1+o(1))}.$$

L'étude de ce problème sera faite *en moyenne*. Par là nous entendrons que pour un jeu de paramètres donné (n, q, R, ω), la complexité des algorithmes étudiés le résolvant sera faite en moyenne sur $\mathbf{H}$ distribuée uniformément sur $\mathbb{F}_q^{(n-k)\times n}$ et $\mathbf{s} \triangleq \mathbf{e}\mathbf{H}^\intercal$ où $\mathbf{e} \hookleftarrow S_{w,n}$. La condition sur les syndromes permet ici de promettre l'existence d'une solution au problème. Cependant, comme nous le montrerons un peu plus loin nous pourrons la relaxer dans certains cas, les syndromes pouvant être en fonction du poids indistinguables *statistiquement* de l'uniforme. Quoiqu'il en soit, nous nous efforcerons tout au long de ce document à toujours spécifier les distributions utilisées.

La difficulté de SD(n, q, R, ω) varie grandement en fonction des paramètres. Une des premières propriétés impactant cette dernière est le nombre attendu de solutions.

1.1.3 Nombre attendu de solutions

Nous rappelons ici que l'espérance du nombre de solutions pour le décodage à poids w est, en moyennant sur les codes, donnée par $\binom{n}{w}(q-1)^w/q^{n-k}$. Cette quantité est alors selon les rapports fixés w/n et k/n asymptotiquement en n exponentiellement élevée ou faible à l'exception de deux valeurs ω^- (usuellement appelée distance relative de Gilbert-Varshamov) et $\omega^+ > \omega^-$ en lesquelles nous attendons une solution. La figure 1.1 résume la situation en traçant le logarithme en base 3 de ce nombre attendu de solutions pour $q = 3$.

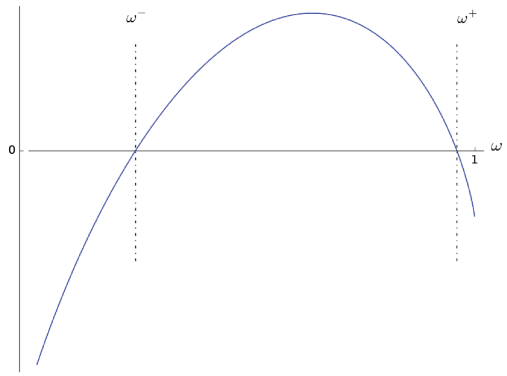


Figure 1.1 – Limite pour $n \to +\infty$ de la division par n du logarithme en base 3 du nombre attendu de solutions au problème du décodage pour $q = 3$ et $R = 1/4$.

Commençons par le fait suivant,

Fait 1. *Soient n, k, w des entiers où $w, k \leqslant n$ et $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$. Supposons que toute équation $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ d'inconnue $\mathbf{e}$ de poids de Hamming w admette une solution, alors nécessairement $w \in [\![w^-, w^+]\!]$ où :*

$$w^- \triangleq \min\left\{ w \in [\![0, n]\!], \quad \binom{n}{w}(q-1)^w \geqslant q^{n-k} \right\} \tag{1.1}$$

$$w^+ \triangleq \max\left\{ w \in [\![0, n]\!], \quad \binom{n}{w}(q-1)^w \geqslant q^{n-k} \right\} \tag{1.2}$$

Remarque 1.1. Pour $q = 2$ on a $w^+ = n - w^-$ mais ceci n'est plus le cas dès que $q \geqslant 3$.

Les quantités w^- et w^+ correspondent au plus petit et plus grand rayon de la sphère centrée en 0 de volume plus grand que celui de l'espace ambiant des syndromes. Ces dernières sont intimement liées à la distance de Gilbert-Varshamov [OS09].

Définition 1.1 (Distance de Gilbert-Varshamov)**.** *Soient $k \leqslant n$ et q des entiers. La distance de Gilbert-Varshamov $w_{\text{GV}}(q,n,k)$ (ou simplement w_{GV} si le contexte le permet) est définie comme le plus grand entier tel que :*

$$\sum_{j=0}^{w_{\text{GV}}(q,n,k)} \binom{n}{j}(q-1)^j \leqslant q^{n-k}$$

Cette distance pour les paramètres n et k donne donc le plus petit rayon à partir duquel une boule couvre $\mathbb{F}_q^{n-k}$. Cependant dans notre cas les boules étant de taille exponentielle, leur volume est donné asymptotiquement par celui de leur sphère. Ainsi bien que w^- et w_{GV} soient proches mais distincts, leurs comportements asymptotiques coïncident. Ces derniers sont fonction de l'entropie q-aire.

Définition 1.2 (Fonction entropie)**.** *Soit un entier q. L'entropie q-aire h_q est définie sur $[0,1]$ comme :*

$$h_q(x) \stackrel{\triangle}{=} -(1-x)\log_q(1-x) - x\log_q\left(\frac{x}{q-1}\right). \tag{1.3}$$

On désignera par g_q^- (resp. g_q^+) son inverse à valeurs dans $[0, \frac{q-1}{q}]$ (resp. $[\frac{q-1}{q}, 1]$) et définie sur $[0,1]$.

Pour $R \stackrel{\triangle}{=} \frac{k}{n}$ fixé et n tendant vers l'infini on a :

$$\frac{w^-}{n} = g_q^-(1-R) + o(1) \tag{1.4}$$

$$\frac{w^+}{n} = \begin{cases} g_q^+(1-R) + o(1) & \text{si } R \leqslant 1 - \log_q(q-1) \\ 1 & \text{sinon.} \end{cases} \tag{1.5}$$

et

$$\frac{w_{\text{GV}}}{n} - \frac{w^-}{n} = o(1). \tag{1.6}$$

Ceci motive l'introduction des quantités relatives ω^- et ω^+ comme :

$$\omega^- \stackrel{\triangle}{=} g_q^-(1-R) \tag{1.7}$$

$$\omega^+ \stackrel{\triangle}{=} \begin{cases} g_q^+(1-R) & \text{si } R \leqslant 1 - \log_q(q-1) \\ 1 & \text{sinon.} \end{cases} \tag{1.8}$$

Dans la suite nous confondrons régulièrement ω^- et la distance relative de Gilbert-Varshamov définie comme $\lim_{n\to+\infty} w_{\text{GV}}/n$.

Toutes ces égalités découlent du lemme suivant (conséquence immédiate de la formule de Stirling) qui sera très régulièrement utilisé tout au long de cette thèse.

Lemme 1.2. *Soient les entiers q, n et $w \stackrel{\triangle}{=} \omega n$ avec $\omega \in]0,1[$. Alors asymptotiquement en n,*

$$\binom{n}{w}(q-1)^w \sim \frac{1}{\sqrt{2\pi n\omega(1-\omega)}} q^{nh_q(\omega)} \tag{1.9}$$

Pour résumer, la condition sur le poids $w \in [\![w^-, w^+]\!]$ est nécessaire pour assurer l'existence d'une solution au problème du décodage pour tout syndrome. De plus, nous connaissons le comportement asymptotique de w^- et w^+ comme fonction de l'entropie. Au delà de donner des bornes nécessaires pour l'existence de solutions, les quantités w^- et w^+ sont étroitement associées au nombre de solutions attendu au problème du décodage pour un code aléatoire. En effet, rappelons la proposition suivante bien connue et pour laquelle nous allons rappeler une démonstration.

Proposition 1.2. *Soient les entiers n, k, w avec $w, k \leqslant n$ et $\mathbf{s} \in \mathbb{F}_q^{n-k}$ un syndrome. Le nombre moyen de solutions $\mathbf{e}$ de poids w de l'équation $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ quand $\mathbf{H}$ est tirée uniformément parmi $\mathbb{F}_q^{(n-k)\times n}$ est donné par :*

$$\frac{\binom{n}{w}(q-1)^w}{q^{n-k}}.$$

Quand n tend vers l'infini mais que w et k sont tels que $w = \omega n$ et $k = Rn$ avec $\omega \in]\omega^-, \omega^+[$ et $R \in]0,1[$, le nombre moyen de solutions est égal à

$$2^{\alpha n(1+o(1))} \quad \text{où} \quad \alpha \stackrel{\triangle}{=} h_q(\omega) - 1 + R > 0.$$

Dans le cas où ω est égal à ω^- ce nombre attendu est donné par $P(n)(1+o(1))$ pour un certain polynôme P.

Le lemme suivant, que nous utiliserons à de nombreuses reprises tout au long de ce document, nous permet de prouver cette proposition.

Lemme 1.3. *Soient les entiers $k \leqslant n$, un vecteur non nul $\mathbf{y} \in \mathbb{F}_q^n$ et un syndrome quelconque $\mathbf{s} \in \mathbb{F}_q^{n-k}$. Nous avons dans le cas où $\mathbf{H}$ est tirée uniformément parmi les matrices de taille $(n-k) \times n$ à coefficients dans $\mathbb{F}_q$:*

$$\mathbb{P}_{\mathbf{H}}\left(\mathbf{y}\mathbf{H}^\intercal = \mathbf{s}\right) = \frac{1}{q^{n-k}}$$

Démonstration du lemme 1.3.

Notons $h_{i,j}$ le coefficient de $\mathbf{H}$ en position (i,j). Quitte à réordonner le vecteur $\mathbf{y}$ et le multiplier par un scalaire nous pouvons supposer que $y_1 = 1$. L'évènement dont nous cherchons la probabilité est donc donné par :

$$\forall i \in [\![1, n-k]\!], \quad h_{i,1} = s_i - \sum_{j=2}^{n} h_{i,j} y_j$$

Du fait de l'uniformité de $\mathbf{H}$, les coefficients $h_{i,j}$ sont indépendants et uniformément distribués ce qui donne le résultat. □

Démonstration de la proposition 1.2.

Soit $\chi_{\mathbf{e}}$ la fonction indicatrice de l'évènement "$\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$". Le nombre attendu de solutions $\mathbf{e}$ de poids w de $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ lors d'un tirage sur $\mathbf{H}$ est donné par $\mathbb{E}_{\mathbf{H}}\left(\sum_{\mathbf{e}\in S_w} \chi_{\mathbf{e}}\right)$. Par linéarité de l'espérance et le lemme 1.3 nous obtenons alors la première partie de la proposition. La seconde partie découle quant à elle directement de la définition de ω^-. □

Comme nous le verrons ce nombre exponentiel de solutions pour des poids relatifs entre ω^- et ω^+ aura un véritable impact sur la complexité des décodeurs génériques. La figure 1.2 résume la situation pour $n \to +\infty$ en fonction des paramètres fixés k/n et w/n du problème de décodage.

1.1.4 Distribution uniforme des syndromes

Les résultats que nous allons évoquer dans cette sous-section font partie des contributions de cette thèse.

Comme tout problème cryptographique, le décodage générique est étudié au regard d'une certaine distribution sur ses entrées. Il s'agit ici de $(\mathbf{H}, \mathbf{e}\mathbf{H}^\intercal)$ pour $\mathbf{H}$ tirée uniformément parmi les matrices de parité et $\mathbf{e}$ au sein des erreurs de poids w. Nous allons

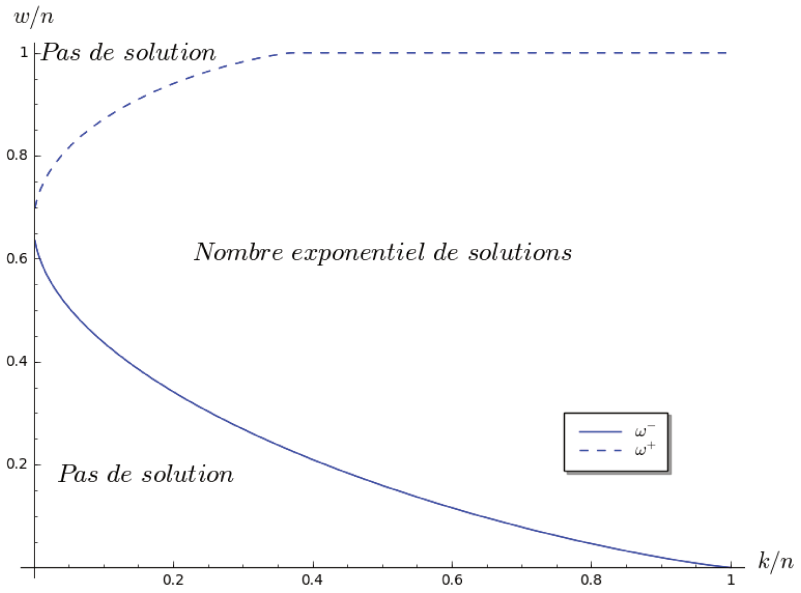


Figure 1.2 – Nombre attendu de solutions au problème du décodage générique pour $q = 3$.

dans ce qui suit montrer que $\mathbf{e}\mathbf{H}^\intercal$ est en moyenne sur $\mathbf{H}$, sous la condition $w/n \in]\omega^-, \omega^+[$, indistinguable de l'uniforme au sens de la distance statistique.

Commençons par la proposition suivante montrant que nous aurions pu tout aussi bien nous intéresser à la distribution des mots de code bruités.

Proposition 1.3. *Soit $\mathcal{C}$ un $[n,k]_q$-code de matrice de parité $\mathbf{H}$. Considérons les variables aléatoires :*

- $\mathbf{e}$ *uniformément distribuée dans les mots de poids w de $\mathbb{F}_q^n$,*
- $\mathbf{c}$ *tiré uniformément dans $\mathcal{C}$,*
- $\mathbf{s}_{\text{unif}}$ *(respectivement $\mathbf{y}_{\text{unif}}$) uniformément distribué dans $\mathbb{F}_q^{n-k}$ (respectivement $\mathbb{F}_q^n$).*

Alors,

$$\rho\left(\mathbf{e}\mathbf{H}^\intercal, \mathbf{s}_{\text{unif}}\right) = \rho\left(\mathbf{c}+\mathbf{e}, \mathbf{y}_{\text{unif}}\right).$$

Démonstration de la proposition 1.3.

Notons $\mathbf{y}(\mathbf{s})$ un vecteur quelconque de $\mathbb{F}_q^{n-k}$ tel que $\mathbf{y}(\mathbf{s})\mathbf{H}^\intercal = \mathbf{s}$ pour $\mathbf{s} \in \mathbb{F}_q^{n-k}$. Nous avons,

$$\begin{aligned}
\rho\left(\mathbf{c}+\mathbf{e}, \mathbf{y}_{\text{unif}}\right) &= \sum_{\mathbf{y}\in\mathbb{F}_q^n} \left| \mathbb{P}_{\mathbf{e},\mathbf{c}}\left(\mathbf{c}+\mathbf{e}=\mathbf{y}\right) - \frac{1}{q^n} \right| \\
&= \sum_{\mathbf{s}\in\mathbb{F}_q^{n-k}} \sum_{\mathbf{c}'\in\mathcal{C}} \left| \mathbb{P}_{\mathbf{e},\mathbf{c}}\left(\mathbf{c}+\mathbf{e}=\mathbf{y}(\mathbf{s})+\mathbf{c}'\right) - \frac{1}{q^n} \right| \qquad (1.10) \\
&= \sum_{\mathbf{s}\in\mathbb{F}_q^{n-k}} \sum_{\mathbf{c}'\in\mathcal{C}} \left| \mathbb{P}_{\mathbf{e}}\left(\mathbf{e}\mathbf{H}^\intercal = \mathbf{y}(\mathbf{s})\mathbf{H}^\intercal\right) \mathbb{P}_{\mathbf{c}}\left(\mathbf{c}=\mathbf{c}'\right) - \frac{1}{q^n} \right| \qquad (1.11) \\
&= \rho\left(\mathbf{e}\mathbf{H}^\intercal, \mathbf{s}_{\text{unif}}\right)
\end{aligned}$$

où nous avons utilisé dans (1.10) le fait que $\mathbb{F}_q^n = \bigsqcup_{\mathbf{s}} \{\mathbf{y}(s)+\mathbf{c} : \mathbf{c}\in\mathcal{C}\}$. La ligne (1.11) est obtenue car les variables aléatoires $\mathbf{c}$ et $\mathbf{e}$ sont indépendantes tandis que la dernière est une conséquence de $\mathbb{P}_{\mathbf{c}}\left(\mathbf{c}=\mathbf{c}'\right) = \frac{1}{q^k}$. □

Venons-en maintenant à la proposition suivante bornant la distance statistique moyenne (sur les codes) entre syndromes respectivement aléatoires et produits à partir d'erreurs uniformes de poids w fixé.

Proposition 1.4. *Soient les entiers $w, k \leqslant n$, les variables aléatoires $\mathbf{H}$, $\mathbf{s}$ et $\mathbf{e}$ distribuées respectivement uniformément sur $\mathbb{F}_q^{(n-k)\times n}$, $\mathbb{F}_q^{n-k}$ et S_w. Nous avons,*

$$\mathbb{E}_{\mathbf{H}}\left(\rho\left(\mathbf{e}\mathbf{H}^{\intercal}, \mathbf{s}_{\text{unif}}\right)\right) \leqslant \frac{1}{2}\sqrt{\frac{q^{n-k}-1}{\binom{n}{w}(q-1)^w}}. \tag{1.12}$$

Quand n tend vers l'infini mais que w, k sont tels que $w = \omega n$ et $k = Rn$ avec $\omega \in]\omega^-, \omega^+[$ et $R \in]0,1[$ nous obtenons,

$$\mathbb{E}_{\mathbf{H}}\left(\rho\left(\mathbf{e}\mathbf{H}^{\intercal}, \mathbf{s}_{\text{unif}}\right)\right) = q^{\alpha n(1+o(1))} \quad \textit{pour} \quad \alpha = \frac{1}{2}\left(1 - R - h_q(\omega)\right) < 0.$$

Remarque 1.2. Nous retrouvons dans la borne de l'équation (1.12) l'inverse du nombre attendu de solutions de poids w au problème du décodage générique.

Remarque 1.3. Ce résultat peut-être vu comme une version plus faible du paramètre de lissage [MR07] en réseaux euclidiens. Il est en effet donné dans [MR07, lemme 4.1] directement une borne sur la distance statistique entre la distribution uniforme et celle d'un mot bruité d'un réseau *fixé* Λ (et non en moyenne sur tous les réseaux). La majoration dépend du bruit mais aussi du plus petit poids des mots du dual de Λ.

La proposition 1.4 est une conséquence directe du *left-over hash lemma* [Bar+11]. Moralement ce lemme ramène la question de l'uniformité des sorties d'une famille de fonctions à la probabilité de collision sur les sorties. Cependant nous allons utiliser ici une variation de ce lemme qui nous sera particulièrement utile dans la partie III dédiée à la description du schéma de signature Wave.

Lemme 1.4. *Soit une famille finie $\mathcal{H} = (h_i)_{i\in I}$ d'applications d'un ensemble fini E dans un ensemble fini F. Notons ε le biais de collision,*

$$\mathbb{P}_{h,e,e'}(h(e) = h(e')) = \frac{1}{|F|}(1+\varepsilon)$$

où h est tirée uniformément dans $\mathcal{H}$, e et e' sont uniformément distribuées dans E. Soit $\mathcal{U}$ la distribution uniforme sur F et $\mathcal{D}(h)$ la distribution $h(e)$ pour e choisi uniformément dans E. Nous avons,

$$\mathbb{E}_h\left(\rho(\mathcal{D}(h), \mathcal{U})\right) \leqslant \frac{1}{2}\sqrt{\varepsilon}.$$

Démonstration du Lemme 1.4.

Définissons pour h_0 et e tirés uniformément dans $\mathcal{H}$ et E :

$$q_{h,f} \triangleq \mathbb{P}_{h_0,e}(h_0 = h, h_0(e) = f).$$

Par définition de la distance statistique et du tirage uniforme de h_0,

$$\begin{aligned}
\mathbb{E}_h\left(\rho(\mathcal{D}(h), \mathcal{U})\right) &= \sum_{h\in\mathcal{H}} \frac{1}{|\mathcal{H}|}\rho(\mathcal{D}(h), \mathcal{U}) \\
&= \sum_{h\in\mathcal{H}} \frac{1}{2|\mathcal{H}|} \sum_{f\in F} \left|\mathbb{P}_e(h(e) = f) - \frac{1}{|F|}\right| \\
&= \frac{1}{2} \sum_{(h,f)\in\mathcal{H}\times F} \left|\mathbb{P}_{h_0,e}(h_0 = h, h_0(e) = f) - \frac{1}{|\mathcal{H}|\cdot|F|}\right| \\
&= \frac{1}{2} \sum_{(h,f)\in\mathcal{H}\times F} \left|q_{h,f} - \frac{1}{|\mathcal{H}|\cdot|F|}\right|. \qquad (1.13)
\end{aligned}$$

ce qui donne en utilisant l'inégalité de Cauchy-Schwartz,

$$\sum_{(h,f)\in\mathcal{H}\times F}\left|q_{h,f}-\frac{1}{|\mathcal{H}|\cdot|F|}\right|\leqslant\sqrt{\sum_{(h,f)\in\mathcal{H}\times F}\left(q_{h,f}-\frac{1}{|\mathcal{H}|\cdot|F|}\right)^2}\cdot\sqrt{|\mathcal{H}|\cdot|F|}. \quad (1.14)$$

Nous observons désormais que,

$$\begin{aligned}\sum_{(h,f)\in\mathcal{H}\times F}\left(q_{h,f}-\frac{1}{|\mathcal{H}|\cdot|F|}\right)^2 &= \sum_{h,f}\left(q_{h,f}^2-2\frac{q_{h,f}}{|\mathcal{H}|\cdot|F|}+\frac{1}{|\mathcal{H}|^2\cdot|F|^2}\right)\\ &= \sum_{h,f}q_{h,f}^2-2\frac{\sum_{h,f}q_{h,f}}{|\mathcal{H}|\cdot|F|}+\frac{1}{|\mathcal{H}|\cdot|F|}\\ &= \sum_{h,f}q_{h,f}^2-\frac{1}{|\mathcal{H}|\cdot|F|}. \quad (1.15)\end{aligned}$$

Considérons pour $i\in\{0,1\}$ les variables aléatoires indépendantes h_i et e_i distribuées uniformément dans $\mathcal{H}$ et E respectivement. Continuons notre calcul en remarquant tout d'abord que

$$\begin{aligned}\sum_{h,f}q_{h,f}^2 &= \sum_{h,f}\mathbb{P}_{h_0,e_0}(h_0=h,h_0(e_0)=f)\mathbb{P}_{h_1,e_1}(h_1=h,h_1(e_1)=f)\\ &= \mathbb{P}_{h_0,h_1,e_0,e_1}\left(h_0=h_1,h_0(e_0)=h_1(e_1)\right)\\ &= \frac{\mathbb{P}_{h_0,e_0,e_1}\left(h_0(e_0)=h_0(e_1)\right)}{|\mathcal{H}|}\\ &= \frac{1+\varepsilon}{|\mathcal{H}|\cdot|F|}. \quad (1.16)\end{aligned}$$

En substituant $\sum_{h,f}q_{h,f}^2$ donné par (1.16) dans (1.15) puis en utilisant l'équation (1.14) nous obtenons finalement,

$$\begin{aligned}\sum_{(h,f)\in\mathcal{H}\times F}\left|q_{h,f}-\frac{1}{|\mathcal{H}|\cdot|F|}\right| &\leqslant \sqrt{\frac{1+\varepsilon}{|\mathcal{H}|\cdot|F|}-\frac{1}{|\mathcal{H}|\cdot|F|}}\sqrt{|\mathcal{H}|\cdot|F|}\\ &= \sqrt{\frac{\varepsilon}{|\mathcal{H}|\cdot|F|}}\sqrt{|\mathcal{H}|\cdot|F|}.\end{aligned}$$

ce qui termine la preuve du lemme. □

Afin de démontrer la proposition 1.4 nous allons utiliser ce lemme avec $E\triangleq S_w$, $F\triangleq\mathbb{F}_q^{n-k}$ et $\mathcal{H}$ la famille de fonctions $\mathbf{e}\mapsto\mathbf{e}\mathbf{H}^\intercal$ indexée par *toutes* les matrices de parité $\mathbf{H}\in\mathbb{F}_q^{(n-k)\times n}$. Cependant remarquons que ce lemme s'appliquerait tout aussi bien à un sous-ensemble de matrices de parité pour lequel on saurait calculer le biais de collision.

Démonstration de la proposition 1.4.

Calculons le biais de collision,

$$\begin{aligned}\mathbb{P}_{\mathbf{H},\mathbf{e},\mathbf{e}'}\left(\mathbf{e}\mathbf{H}^\intercal=\mathbf{e}'\mathbf{H}^\intercal\right) &= \mathbb{P}_{\mathbf{H},\mathbf{e},\mathbf{e}'}\left((\mathbf{e}-\mathbf{e}')\mathbf{H}^\intercal=\mathbf{0}\right)\\ &= \mathbb{P}_{\mathbf{H},\mathbf{e},\mathbf{e}'}\left((\mathbf{e}-\mathbf{e}')\mathbf{H}^\intercal=\mathbf{0}\mid\mathbf{e}\neq\mathbf{e}'\right)\mathbb{P}\left(\mathbf{e}\neq\mathbf{e}'\right)+\mathbb{P}_{\mathbf{e},\mathbf{e}'}(\mathbf{e}=\mathbf{e}')\\ &= \frac{1}{q^{n-k}}\left(1-\frac{1}{\binom{n}{w}}\right)+\frac{1}{\binom{n}{w}} \quad \text{(cf Lemme 1.3.)}\\ &= \frac{1}{q^{n-k}}(1+\varepsilon) \quad \text{où} \quad \varepsilon\triangleq\frac{q^{n-k}-1}{\binom{n}{w}}\end{aligned}$$

ce qui termine la première partie de la preuve. La seconde partie découle quant à elle directement du lemme 1.2 et des définitions de ω^- et ω^+. □

1.1.5 Réduction de recherche à décision

Il est usuel en cryptographie de considérer pour un même problème calculatoire deux variantes : décisionnelle ou de recherche. Grossièrement la situation est la suivante, on commence par se donner une fonction à sens unique f. Le problème de recherche a pour but de retrouver $\mathbf{x}$ étant donné $(f, f(\mathbf{x}))$. Le problème décisionnel consiste quant à lui à distinguer une véritable instance d'une instance aléatoire, c'est à dire étant donné $(f, \mathbf{y})$ à décider si $\mathbf{y}$ est aléatoire ou obtenu comme un certain $f(\mathbf{x})$. S'appuyer sur la difficulté de la variante décisionnelle, plutôt que de recherche, d'un problème calculatoire est a priori une hypothèse plus forte. Il s'avère que des constructions cryptographiques comme celles de Diffie-Hellman [DH76] ou encore d'El Gamal [ElG84] reposent sur une telle assomption. La situation est différente pour le problème du décodage où il a été prouvé dans [FS96] puis repris dans [Döt14] que la variante décisionnelle est aussi dure que son pendant de recherche dans le sens où à partir d'un algorithme résolvant la version décisionnelle avec probabilité ε en temps t on peut en déduire un algorithme résolvant la version de recherche en un temps essentiellement donné par t/ε^2. On parle alors de réduction de recherche à décision.

Une idée naturelle pour prouver un tel résultat est la suivante, partons d'un adversaire $\mathcal{A}$ distinguant $\mathbf{y}$ aléatoire de $f(\mathbf{x})$. Étant donné $f(\mathbf{x})$ que l'on souhaite inverser, on souhaiterait utiliser $\mathcal{A}$ pour glaner une information partielle sur $\mathbf{x}$, par exemple en perturbant $f(\mathbf{x})$, pour ensuite en déduire $\mathbf{x}$ en recommençant l'opération un certain nombre de fois. Le point crucial ici est la signification d'information partielle avec laquelle nous devons prendre quelques précautions et être plus précis. Il est par exemple vain de chercher à obtenir un énoncé comme, si à partir de $\mathcal{A}$ on en déduit un bit de $\mathbf{x}$ connaissant $f(\mathbf{x})$ alors on peut inverser f. En effet, si f est à sens unique, il est clair que $g(x_1, \mathbf{x}) = (x_1, f(\mathbf{x}))$ l'est aussi alors que le bit x_1 est révélé. Nous avons donc besoin d'une étape intermédiaire entre $\mathcal{A}$ et l'inversion de f. Autrement dit, quelle information doit-on tirer de $\mathcal{A}$ pour ensuite pouvoir inverser f ? Une première réponse à cette question a été donnée dans [Gol01, Proposition 2.5.4] où il est prouvé, comme énoncé dans la proposition qui suit, que si $\mathcal{A}$ permet de retrouver avec une meilleure probabilité que $1/2$ la quantité $\langle \mathbf{x}, \mathbf{r} \rangle$ à partir de $f(\mathbf{x})$ et $\mathbf{r}$ uniforme alors c'est gagné, on sait inverser f.

Proposition 1.5 ([Gol01])**.** *Soient $f : \{0,1\}^* \to \{0,1\}^*$, $\mathcal{A}$ un algorithme probabiliste en temps $t : \mathbb{N} \to \mathbb{N}$ et $\varepsilon : \mathbb{N} \to [0,1]$ tels que*

$$\mathbb{P}\left(\mathcal{A}(f(\mathbf{x}_n), \mathbf{r}_n) = \langle \mathbf{x}_n, \mathbf{r}_n \rangle\right) = \frac{1}{2} + \varepsilon(n)$$

où la probabilité est calculée sur l'aléa interne de $\mathcal{A}$ et $\mathbf{x}_n, \mathbf{r}_n$ eux-mêmes uniformément distribués sur $\{0,1\}^n$. Définissons $\ell(n) \triangleq \log(1/\varepsilon(n))$. Alors il existe un algorithme $\mathcal{A}'$ fonctionnant en temps $O(n^2 \ell(n)^3) t(n)$ qui satisfait :

$$\mathbb{P}\left(\mathcal{A}'(f(\mathbf{x}_n) = \mathbf{x}_n)\right) = \Omega\left(\varepsilon(n)^2\right)$$

où la probabilité est calculée sur l'aléa interne de $\mathcal{A}'$ et $\mathbf{x}_n$.

Remarque 1.4. Fait intéressant, la preuve d'un tel résultat repose sur l'utilisation des codes de Reed-Muller d'ordre 1 [MS86, Chapitre 13] et de leur algorithme de décodage.

Cette proposition sera la brique élémentaire de la preuve que le problème du décodage se réduit à sa variante décisionnelle. Commençons par définir la version décisionnelle du problème de décodage.

Problème 1.8 (Problème décisionnel du décodage générique - DSD(n, q, R, ω))**.** *Soient les entiers $k \triangleq \lfloor Rn \rfloor$ et $w \triangleq \lfloor \omega n \rfloor$.*

— *Distributions :*

· $\mathcal{D}_0$: $(\mathbf{H}, \mathbf{s}^{\text{unif}})$ *distribuée uniformément sur* $\mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$.

· $\mathcal{D}_1$: $(\mathbf{H}, \mathbf{e}\mathbf{H}^\intercal)$ *où* $\mathbf{H}$ *et* $\mathbf{e}$ *uniformément distribués sur* $\mathbb{F}_q^{(n-k)\times n} \times S_w$.

— *Entrée :* $(\mathbf{H}, \mathbf{s})$ *pour* $(\mathbf{H}, \mathbf{s}) \hookleftarrow \mathcal{D}_b$ *avec* $b \hookleftarrow \{0, 1\}$

— *Décision :* $b' \in \{0, 1\}$.

Une première solution pour résoudre DSD consiste à renvoyer pour chaque instance un bit b' aléatoire. On retrouve alors la distribution utilisée en entrée avec probabilité $1/2$ ce qui n'est pas très intéressant. L'efficacité d'un algorithme $\mathcal{A}$ résolvant DSD est mesurée par l'écart entre sa probabilité de trouver la bonne solution avec $1/2$. La quantité capturant cette notion est l'avantage DSD défini comme :

Définition 1.3. *L'avantage* DSD *d'un algorithme* $\mathcal{A}$ *est défini comme :*

$$Adv^{\text{DSD}(n,q,R,\omega)}(\mathcal{A}) \triangleq \frac{1}{2}(\mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}) = 1 \mid b = 1\right) \quad - \quad \mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}) = 1 \mid b = 0\right)) \tag{1.17}$$

où les probabilités sont calculées sur l'aléa interne de $\mathcal{A}$*,* b *et l'entrée tirée selon* $\mathcal{D}_b$ *définie dans le problème 1.8 de paramètres* (n, q, R, ω)*. Nous définissons alors le succès* DSD *calculatoire en temps* t *:*

$$Succ^{\text{DSD}(n,q,R,\omega)}(t) \triangleq \max_{\mathcal{A}:|\mathcal{A}|\leqslant t} \left(Adv^{\text{DSD}}(\mathcal{A})\right).$$

Il nous arrivera d'omettre la dépendance en les paramètres (n, q, R, ω)*.*

On vérifie alors facilement que :

$$\mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}) = b\right) = \frac{1}{2} + Adv^{\text{DSD}}(\mathcal{A})$$

Cette égalité motive l'introduction de la notion d'avantage : plus ce dernier est éloigné de 0, plus la probabilité de résoudre le problème décisionnel est proche de 1.

Notre objectif est de démontrer le théorème qui suit reliant la difficulté de DSD à SD. Pour cela nous allons rappeler comment ramener $\mathcal{A}$ d'avantage $Adv^{\text{DSD}}(\mathcal{A})$ à un algorithme calculant $\langle \mathbf{e}, \mathbf{r} \rangle$ avec probabilité $1/2 + Adv^{\text{DSD}}(\mathcal{A})$ étant donné $\mathbf{e}\mathbf{H}^\intercal$ et $\mathbf{r}$. Il suffira ensuite d'appliquer la proposition 1.5.

Théorème 1.1. *Soient* n, R *et* ω *des paramètres de* SD*. Soit* $\mathcal{A}$ *un algorithme probabiliste fonctionnant en temps* $t : \mathbb{N} \to \mathbb{N}$ *et de* DSD*-avantage* $\varepsilon : \mathbb{N} \to \left[-\frac{1}{2}, \frac{1}{2}\right]$ *pour les paramètres* $(n, 2, R, \omega)$*. Définissons* $\ell(n) \triangleq \log(1/\varepsilon(n))$*.*

Alors il existe un algorithme $\mathcal{A}'$ *qui résout* $\text{SD}(n, 2, R, \omega)$ *en temps* $O(n^2\ell(n)^3)t(n)$ *et avec probabilité* $\Omega(\varepsilon(n)^2)$*.*

Démonstration du théorème 1.1.

Soit $\mathcal{A}$ un algorithme d'avantage DSD ε. Nous allons construire un algorithme $\mathcal{A}'$ qui étant donné $(\mathbf{H}, \mathbf{e}\mathbf{H}^\intercal)$ et $\mathbf{r}$ calcule $\langle \mathbf{r}, \mathbf{e} \rangle$ avec probabilité $1/2 + \varepsilon$. Pour terminer la preuve, il suffira alors d'appliquer la proposition 1.5.

Algorithme $\mathcal{A}'$:
Entrée : $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^n$ et $\mathbf{r} \in \mathbb{F}_2^n$
$\mathbf{u} \hookleftarrow \mathbb{F}_2^{n-k}$
$\mathbf{H}' \leftarrow \mathbf{H} - \mathbf{u}^\intercal\mathbf{r}$
$b \leftarrow \mathcal{A}(\mathbf{H}', \mathbf{s})$
Sortie : b

La matrice $\mathbf{H}$ étant uniformément distribuée il en est de même de $\mathbf{H}'$. En effet, celle-ci est une translation de $\mathbf{H}$. Remarquons maintenant que,

$$\mathbf{s} = \mathbf{e}\mathbf{H}^\intercal = \mathbf{e}\mathbf{H}'^\intercal + \mathbf{u}\langle \mathbf{e}, \mathbf{r}\rangle.$$

Définissons,

$$\mathbf{s}' \stackrel{\triangle}{=} \mathbf{e}\mathbf{H}'^\intercal + \mathbf{u}.$$

Il est alors clair que $\mathbf{s}'$ est uniformément distribuée. Ainsi, selon que $\langle \mathbf{e}, \mathbf{r}\rangle$ est égal à 0 ou 1 on retrouve les distributions de DSD. La probabilité que $\mathcal{A}'$ renvoie $\langle \mathbf{e}, \mathbf{r}\rangle$ est donnée par :

$$\begin{aligned}
&\mathbb{P}(\mathcal{A}'(\mathbf{H},\mathbf{e}\mathbf{H}^\intercal, \mathbf{r}) = \langle \mathbf{r}, \mathbf{e}\rangle) \\
&\quad = \frac{1}{2}\mathbb{P}\left(\mathcal{A}'(\mathbf{H}, \mathbf{e}\mathbf{H}^\intercal, \mathbf{r}) = 0 \mid \langle \mathbf{r}, \mathbf{e}\rangle = 0\right) \\
&\qquad\qquad + \frac{1}{2}\mathbb{P}\left(\mathcal{A}'(\mathbf{H}, \mathbf{e}\mathbf{H}^\intercal, \mathbf{r}) = 1 \mid \langle \mathbf{r}, \mathbf{e}\rangle = 1\right) \qquad (1.18) \\
&\quad = \frac{1}{2}\left(\mathbb{P}\left(\mathcal{A}(\mathbf{H}', \mathbf{e}\mathbf{H}'^\intercal) = 0\right) + \mathbb{P}\left(\mathcal{A}(\mathbf{H}', \mathbf{s}') = 1\right)\right) \\
&\quad = \frac{1}{2} + \frac{1}{2}\left(\mathbb{P}\left(\mathcal{A}(\mathbf{H}', \mathbf{s}') = 1\right) - \mathbb{P}\left(\mathcal{A}(\mathbf{H}', \mathbf{e}\mathbf{H}'^\intercal) = 1\right)\right) \\
&\quad = \frac{1}{2} + \varepsilon \qquad (1.19)
\end{aligned}$$

où nous avons utilisé dans (1.18) le fait que $\mathbf{r}$ est uniformément distribuée et dans (1.19) la définition de l'avantage DSD. □

Le théorème 1.1 est démontré pour le corps binaire. Nous pouvons l'étendre au cas q-aire en utilisant la généralisation de la proposition 1.5 [Gol01] démontrée dans [GRS00].

Nous souhaiterions étendre cette réduction de recherche à décision au problème du décodage où en entrée est donné un code avec un grand groupe d'automorphismes. Cela s'appliquerait au chiffrement utilisant des codes MDPC [Mis+13] ou encore HQC [AM+18]. Une piste intéressante pour la preuve d'un tel résultat est le travail récent de [PRS17] donnant une réduction de recherche à décision des problèmes génériques avec des réseaux euclidiens idéaux et non quelconques.

1.2 Le décodage générique en cryptographie avec la métrique rang

Nous allons maintenant suivre le fil de la section précédente en nous intéressant au décodage générique pour la métrique rang. Commençons cependant par définir formellement cette métrique ainsi que quelques concepts associés.

1.2.1 Généralités sur la métrique rang, codes matriciels et $\mathbb{F}_{q^m}$-linéaires

Nous regardons dans cette section l'espace ambiant $\mathbb{F}_q^N$ où $N = m \times n$ comme l'espace des matrices à coefficients dans $\mathbb{F}_q$ ayant m lignes et n colonnes. On munit alors naturellement cet espace d'une structure métrique $d(\cdot, \cdot)$ avec l'opérateur rang sur les matrices comme,

$$\forall \mathbf{X}, \mathbf{Y} \in \mathbb{F}_q^{m\times n},\ d(\mathbf{X}, \mathbf{Y}) \stackrel{\triangle}{=} \mathrm{Rang}(\mathbf{X} - \mathbf{Y}).$$

Dans ce paradigme il est usuel de nommer code matriciel tout $[m \times n, K]_q$-code et de le munir de la métrique rang. Cependant l'habitude en cryptographie fondée sur les codes équipés de la métrique rang n'est pas de s'intéresser aux codes matriciels mais à une sous-classe de ces derniers ayant la propriété d'être spécifiés de manière bien plus compacte : les codes $\mathbb{F}_{q^m}$-linéaires.

Le corps fini $\mathbb{F}_{q^m}$ est une extension algébrique de degré m de $\mathbb{F}_q$ ce qui en fait un $\mathbb{F}_q$-espace vectoriel de dimension m. Étant donné des mots $x_1, \cdots, x_N \in \mathbb{F}_{q^m}$, la notation suivante désignera le $\mathbb{F}_q$-espace vectoriel engendré par cs-derniers.

Notation 1.

$$\text{Vect}_{\mathbb{F}_q}(x_1, \cdots, x_N) \stackrel{\triangle}{=} \left\{ \sum_{i=1}^{N} \lambda_i x_i \ : \ \lambda_1, \cdots, \lambda_N \in \mathbb{F}_q \right\}.$$

Considérons une base $\mathcal{B} = (\alpha_1, \cdots, \alpha_m)$ de $\mathbb{F}_{q^m}$ vu comme un espace $\mathbb{F}_q$-espace vectoriel. Il est alors possible d'exprimer tout élément de $\mathbb{F}_{q^m}$ avec m coordonnées à l'aide de $\mathcal{B}$ dans $\mathbb{F}_q$ et donc d'écrire tout mot $(x_1, \cdots, x_n)$ de $\mathbb{F}_{q^m}^n$ comme une matrice de taille $m \times n$ en remplaçant chaque entrée x_i par le vecteur colonne de ses coordonnées dans la base $\mathcal{B}$. En conséquence nous pouvons voir tout $[n,k]_{q^m}$-code de base $(\mathbf{c}_1, \cdots, \mathbf{c}_k)$ comme un code matriciel de longueur $n \times m$ sur $\mathbb{F}_q$ et de dimension $k \times m$, engendré par l'écriture matricielle de $(\alpha_j \mathbf{c}_i)_{\substack{1 \leqslant j \leqslant m \\ 1 \leqslant i \leqslant k}}$.

Définition 1.4 (Code matriciel associé à un code $\mathbb{F}_{q^m}$-linéaire et métrique rang associée)**.** *Soit $\mathcal{C}$ un $[n,k]_{q^m}$-code et $(\alpha_1, \cdots, \alpha_m)$ une base de $\mathbb{F}_{q^m}$ en tant que $\mathbb{F}_q$-espace vectoriel. Tout mot de code $\mathbf{c} \in \mathcal{C}$ peut s'exprimer comme :*

$$\text{Mat}(\mathbf{c}) \stackrel{\triangle}{=} (M_{i,j}) \in \mathbb{F}_q^{m \times n} \quad \textit{où} \quad c_j = \sum_{i=1}^{m} M_{i,j} \alpha_i.$$

L'ensemble $\{\text{Mat}(\mathbf{c}) : \mathbf{c} \in \mathcal{C}\}$ est le $[n \times m, k \times m]_q$-code associé au code $\mathbb{F}_{q^m}$-linéaire $\mathcal{C}$. Le poids (rang) de $\mathbf{c} \in \mathcal{C}$ est alors défini comme :

$$|\mathbf{c}|_{\text{Rang}} \stackrel{\triangle}{=} \text{Rang}(\text{Mat}(\mathbf{c})).$$

Remarque 1.5. La représentation matricielle d'un vecteur dépend évidemment de la base de $\mathbb{F}_{q^m}$ utilisée ce qui n'est pas le cas de $|\cdot|_{\text{Rang}}$.

Lorsque le contexte s'y prêtera, nous omettrons l'indice "Rang" dans l'écriture $|\mathbf{c}|_{\text{Rang}}$ et nous parlerons tout simplement de poids. Revenons maintenant un instant sur la vision matricielle des codes $\mathbb{F}_{q^m}$-linéaires, celle-ci n'étant pas toujours la plus adaptée. Avec le poids de Hamming nous comptions les éléments du support. En métrique rang ce concept doit être quelque peu ajusté comme il a été décrit dans [GRS13 ; GRS16].

Définition 1.5 ($\mathbb{F}_q$-support)**.** *Soit $\mathbf{x} \in \mathbb{F}_{q^m}^n$, son $\mathbb{F}_q$-support est défini comme*

$$\text{Supp}(\mathbf{x})_{\mathbb{F}_q} \stackrel{\triangle}{=} \text{Vect}_{\mathbb{F}_q}(x_1, \cdots, x_n) \subseteq \mathbb{F}_{q^m}.$$

Cette notion capture foncièrement le concept de métrique rang pour les codes $\mathbb{F}_{q^m}$-linéaires. Le poids $|\mathbf{x}|$ est égal à la dimension de $\text{Supp}(\mathbf{x})_{\mathbb{F}_q}$. Cette relation entre dimension et poids sera régulièrement utilisée car offrant une plus grande maniabilité.

1.2.2 Difficulté dans le pire cas

Les codes usuellement considérés en métrique rang sont une sous-classe des codes matriciels : les codes $\mathbb{F}_{q^m}$-linéaires. Avant de nous intéresser au problème du décodage de ces derniers, commençons par le cas des codes matriciels équipés de la métrique rang. Soient $\mathcal{C}$ un $[m \times n, K]_q$ code matriciel de base $\mathbf{C}_1, \cdots, \mathbf{C}_K$ et le mot $\mathbf{Y}$ à décoder à distance w. Nous cherchons ici à trouver $\lambda_1, \cdots, \lambda_k \in \mathbb{F}_q$ tels que :

$$\text{Rang}\left(\sum_{i=1}^{K} \lambda_i \mathbf{C}_i - \mathbf{Y} \right) = w.$$

Le décodage d'un code matriciel correspond donc exactement au problème MinRank (voir le problème 1.3) dont la version décisionnelle a été prouvé NP-complète par une réduction au problème de décodage en métrique de Hamming [Cou01]. L'idée est la suivante : partant d'un code sur $\mathbb{F}_q^n$ muni de la métrique de Hamming, on transforme tout mot de longueur n en une matrice diagonale de taille $n \times n$ dont les n coefficients diagonaux sont les coordonnées du mot. Ainsi avec cette transformation, décoder en métrique de Hamming est équivalent à décoder en métrique rang. En revanche, bien que tout code $\mathbb{F}_{q^m}$-linéaire puisse s'écrire comme un code matriciel, le problème de décodage de ces derniers n'est a priori pas NP-complet.

Il a néanmoins été donné dans [GZ16] une réduction probabiliste et polynomial au problème de décodage en métrique de Hamming. Cependant cette réduction est faite dans le cas où le degré de l'extension de corps du code est quadratique en sa longueur (*ie :* $m > n^2$). Il est à noter que cette condition impacte fortement la qualité de la réduction car dans la pratique cryptographique m est choisi de l'ordre de n. L'ajout de la $\mathbb{F}_{q^m}$-linéarité semble donc être une entrave à la NP-complétude du problème de décodage en métrique rang.

1.2.3 Le décodage générique en métrique rang

Le problème MinRank ainsi que le décodage d'un code $\mathbb{F}_{q^m}$-linéaire, bien que semblant proches, sont intrinsèquement différents. Dans cette thèse nous nous intéresserons au décodage des codes $\mathbb{F}_{q^m}$-linéaires, commençons par le formaliser.

Problème 1.9 (Problème du décodage générique des codes $\mathbb{F}_{q^m}$-linéaires en métrique rang à poids fixé - RSD(n, q, R, ω, μ))**.** *Soient les entiers* $m \triangleq \mu n$, $k \triangleq \lfloor Rn \rfloor$ *et* $w \triangleq \lfloor \omega n \rfloor$.

— Instance : *une matrice de parité* $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ *de rang* $n-k$*, un syndrome* $\mathbf{s} \in \mathbb{F}_{q^m}^{n-k}$.

— Recherche : *un vecteur d'erreur* $\mathbf{e} \in \mathbb{F}_{q^m}^n$ *de poids rang* w *tel que* $\mathbf{eH}^\intercal = \mathbf{s}$.

Ce problème est entre autres paramétré par le poids relatif de décodage ω qui généralement sera une constante dans $]0, 1[$ ou une fonction négligeable de n et par μ qui dans les diverses applications sera une constante proche de 1.

L'analyse de RSD(n, q, R, ω, μ) se fera de la même façon que pour SD(n, q, R, ω). De plus, comme dans le cas de la métrique de Hamming nous allons étudier le nombre attendu de solutions et la distribution des syndromes produits à partir d'erreur de poids fixé. Les résultats que nous allons exposer (ainsi que les définitions) sont similaires à la différence près que le degré d'extension m est un nouveau degré de liberté avec lequel nous pourrons jouer. En revanche, contrairement au cas de la métrique de Hamming il n'est pas connu à ce jour une réduction du problème de décodage des codes $\mathbb{F}_{q^m}$-linéaires à sa variante décisionnelle.

1.2.4 Nombre attendu de solutions

Nous nous intéressons dans cette sous-section au nombre attendu de solutions du problème de décodage en métrique rang. Tout comme en métrique de Hamming, cette quantité sera exponentiellement élevée ou faible selon le rapport w/n mais cette fois-ci à l'exception d'une valeur et non de deux.

Le dénombrement d'erreurs en métrique de Hamming faisait intervenir des binomiales car nous comptions les différents supports possibles. En métrique rang, afin de compter les erreurs de poids w nous devons énumérer les $\mathbb{F}_q$-espaces vectoriels de $\mathbb{F}_{q^m}$ de dimension w, c'est à dire dénombrer les $\mathbb{F}_q$-supports (définition 1.5) de dimension w ou encore compter les matrices de $\mathbb{F}_q^{m\times n}$ de rang w.

Proposition 1.6. *Soient les entiers n, m et q une taille de corps. Le nombre de $\mathbb{F}_q$-supports de dimension $w \leqslant m$ est donné par :*

$$\begin{bmatrix} m \\ w \end{bmatrix}_q \triangleq \prod_{\ell=0}^{w-1} \frac{q^m - q^\ell}{q^w - q^\ell}. \tag{1.20}$$

Quant au volume d'une sphère de rayon $w \leqslant \min(m,n)$ de $\mathbb{F}_{q^m}^n$:

$$\prod_{\ell=0}^{w-1} (q^n - q^\ell) \begin{bmatrix} m \\ w \end{bmatrix}_q. \tag{1.21}$$

Pour $\max(m,n)$ tendant vers l'infini on a :

$$\begin{bmatrix} m \\ w \end{bmatrix}_q = \Theta\left(q^{w(m-w)}\right) \quad \textit{et} \quad \prod_{\ell=0}^{w-1} (q^n - q^\ell) \begin{bmatrix} m \\ w \end{bmatrix}_q = \Theta(q^{w(m+n-w)}). \tag{1.22}$$

On trouvera une preuve de ces résultats dans [Loi06]. Cependant une façon rapide de retrouver une bonne estimation du nombre de sous $\mathbb{F}_q$-espace vectoriel de $\mathbb{F}_{q^m}$ dimension w est le raisonnement suivant : une proportion constante de sous-espaces vectoriel de dimension w est déterminée par une matrice génératrice sous la forme $\begin{pmatrix} \mathbf{1}_w & \mathbf{G} \end{pmatrix} \in \mathbb{F}_q^{w \times m}$. Il y a $q^{w(m-w)}$ telles matrices ce qui donne l'estimation.

Nous sommes maintenant en mesure de définir la distance de Gilbert-Varshamov pour les codes $\mathbb{F}_{q^m}$-linéaires en métrique rang.

Définition 1.6 (Distance de Gilbert-Varshamov pour la métrique rang)**.** *Soient $k \leqslant n$ et q des entiers. La distance de Gilbert-Varshamov pour la métrique rang $w_{\mathrm{rGV}}(q,m,n,k)$ (ou simplement w_{rGV} si le contexte le permet) est définie comme le plus grand entier tel que :*

$$\sum_{j=0}^{w_{\mathrm{rGV}}(q,m,n,k)} \left(\prod_{\ell=0}^{j-1} (q^n - q^\ell) \right) \begin{bmatrix} m \\ j \end{bmatrix}_q \leqslant q^{m(n-k)}$$

Comme en métrique de Hamming, la distance de Gilbert-Varshamov est le plus petit rayon à partir duquel une boule couvre $\mathbb{F}_{q^m}^{n-k}$. La proposition qui suit donne son développement asymptotique (voir [Loi06, Théorème 2]).

Proposition 1.7. *Pour $R = \frac{k}{n}$ fixé, nous avons lorsque $\max(m,n)$ tend vers l'infini :*

$$w_{\mathrm{rGV}}(q,m,n,k) = \frac{m + n - \sqrt{(m-n)^2 + 4km}}{2}(1 + o(1)). \tag{1.23}$$

On en déduit que pour $R = \frac{k}{n}$ fixé et $n \to +\infty$ on a :

— Si $m = n$:

$$w_{\mathrm{rGV}}(q,m,n,k) \sim n(1 - \sqrt{R}). \tag{1.24}$$

— Si $m = n^2$:

$$w_{\mathrm{rGV}}(q,m,n,k) \sim n(1 - R). \tag{1.25}$$

Proposition 1.8. *Soient les entiers n, k, w avec $w, k \leqslant n$ et $\mathbf{s} \in \mathbb{F}_{q^m}^{n-k}$ un syndrome. Le nombre moyen de solutions $\mathbf{e}$ de poids rang w de l'équation $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ quand $\mathbf{H}$ est tirée uniformément parmi $\mathbb{F}_{q^m}^{(n-k)\times n}$ est donnée par :*

$$\frac{\prod_{\ell=0}^{w-1}(q^n - q^\ell) \begin{bmatrix} m \\ w \end{bmatrix}_q}{q^{m(n-k)}} \underset{\max(m,n)\to+\infty}{=} \Theta\left(q^{w(m+n-w)-m(n-k)}\right).$$

Si $w = w_{\mathrm{rGV}}(q,m,n,k)$ alors ce nombre de solutions est un $\Theta(1)$.

La démonstration de cette proposition est similaire à celle de la proposition 1.2. La figure 1.3 résume la situation quand $m = n$ en fonction des paramètres fixés k/n et w/n où w_{rGV} est asymptotiquement donnée dans (1.24).

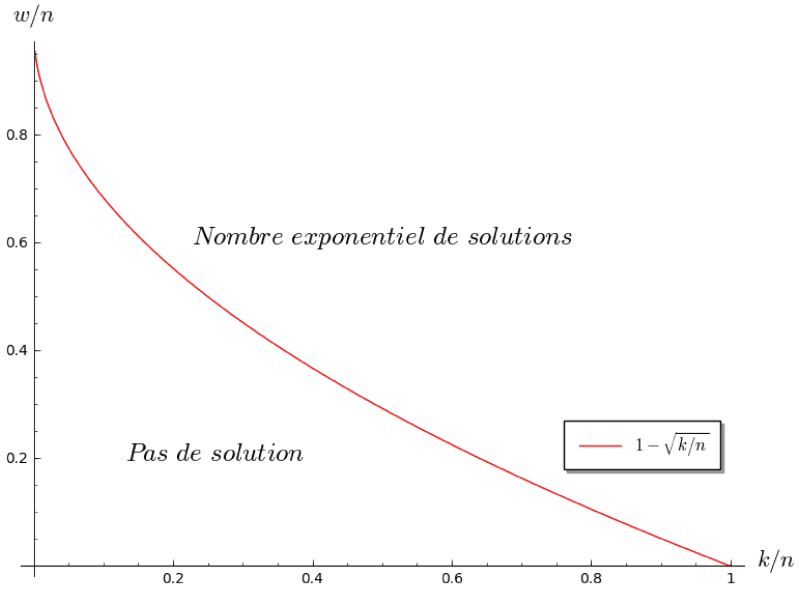


Figure 1.3 – Nombre attendu de solutions au problème du décodage générique en métrique rang pour $m = n$.

1.2.5 Distribution uniforme des syndromes

Le problème du décodage en métrique rang pose lui aussi la question de la distribution de ses entrées : $(\mathbf{H}, \mathbf{eH}^\intercal)$ pour $\mathbf{H}$ uniformément distribuée sur $\mathbb{F}_{q^m}^{(n-k)\times n}$ et $\mathbf{e}$ uniformément tiré dans les mots de rang w. Le comportement des syndromes $\mathbf{eH}^\intercal$ est alors semblable au cas de la métrique de Hamming. Ils sont en moyenne sur les codes statistiquement indistinguables de l'uniforme si w est choisi dans des zones où le nombre de solutions est exponentiel comme énoncé dans la proposition qui suit.

Proposition 1.9. *Soient les entiers m et $w, k \leqslant n$, la variable aléatoire $\mathbf{H}$ (resp. $\mathbf{e}$) distribuée uniformément sur $\mathbb{F}_{q^m}^{(n-k)\times n}$ (resp. sur les mots de poids rang w). Nous avons,*

$$\mathbb{E}_{\mathbf{H}}\left(\rho\left(\mathbf{eH}^\intercal, \mathbf{s}_{\text{unif}}\right)\right) \leqslant \frac{1}{2}\sqrt{\frac{q^{m(n-k)}-1}{\prod_{\ell=0}^{w-1}(q^n - q^\ell)\begin{bmatrix} m \\ w \end{bmatrix}_q}}. \tag{1.26}$$

Quand $\max(m, n)$ tend vers l'infini on a :

$$\mathbb{E}_{\mathbf{H}}\left(\rho\left(\mathbf{eH}^\intercal, \mathbf{s}_{\text{unif}}\right)\right) = \Theta\left(q^{m(n-k)-w(m+n-w)}\right). \tag{1.27}$$

La démonstration d'une telle proposition est similaire à la preuve de la proposition 1.4 où le lemme 1.4 est utilisé de façon cruciale.

1.3 Chiffrement à clef publique et codes correcteurs

Historiquement il y eut deux grandes approches pour faire du chiffrement reposant sur les problèmes SD et RSD selon la métrique utilisée : (i) le schéma de McEliece [McE78] et de Niederreiter [Nie86] réclamant des codes pour lesquels on connaît un bon algorithme de décodage contrairement au (ii) chiffrement d'Alekhnovich [Ale11] ne faisant reposer la sécurité que sur le décodage des codes aléatoires.

Nous commencerons ici par présenter sommairement les systèmes de (i) ainsi que les familles des codes de Goppa et MDPC proposés lors de leurs instanciations. Nous exposerons ensuite une version duale du chiffrement d'Alekhnovich introduite dans [Gab+17b] et nous terminerons cette discussion par l'instanciation de ces schémas avec des codes $\mathbb{F}_{q^m}$-linéaires munis de la métrique rang.

Commençons tout d'abord par donner la définition formelle de chiffrement à clef publique ainsi que les modèles de sécurité que nous considèrerons.

Définition 1.7. *Un chiffrement asymétrique est donné par un triplet d'algorithmes* $(\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ *et un couple de fonctions* $(m(\lambda), n(\lambda))$ *polynomiales en le paramètre de sécurité* λ *tels que :*

- `KeyGen`*,* la génération de clef*, est un algorithme probabiliste polynomial d'entrée* 1^λ [1] *et renvoyant une paire de clef publique et secrète* $(\mathrm{pk}, \mathrm{sk})$*,*
- `Enc`*,* le chiffrement*, est un algorithme probabiliste polynomial prenant en entrée* pk*,* $\mathbf{x} \in \{0,1\}^{m(\lambda)}$ *et renvoyant un chiffré* $\mathbf{y} \in \{0,1\}^{n(\lambda)}$*,*
- `Dec`*,* le déchiffrement*, est un algorithme probabiliste polynomial prenant en entrée* sk*,* $\mathbf{y} \in \{0,1\}^{n(\lambda)}$ *et renvoyant* $\mathbf{x} \in \{0,1\}^{m(\lambda)}$ *ou un symbole* $\perp$ *en cas d'échec.*

De plus, nous avons la relation que pour tout couple $(\mathrm{pk}, \mathrm{sk})$ *obtenu à partir de* 1^λ *et tout* $\mathbf{x} \in \{0,1\}^{m(\lambda)}$ *:*

$$\mathbb{P}\left(\texttt{Dec}^{\mathrm{sk}}\left(\texttt{Enc}^{\mathrm{pk}}(\mathbf{x})\right) \neq \mathbf{x}\right) \in \mathsf{negl}(\lambda)$$

où la probabilité est calculée pour $\mathbf{x} \hookleftarrow \{0,1\}^{m(\lambda)}$*.*

Définir la sécurité d'un chiffrement à clef publique n'est pas chose aisée. Cette difficulté trouve son origine dans le pouvoir que l'on peut potentiellement donner à un attaquant cherchant à déchiffrer sans la clef secrète. On peut par exemple supposer qu'un adversaire dispose de $\texttt{Enc}^{\mathrm{pk}}(\mathbf{x})$ avec le doute que $\mathbf{x}$ a été chiffré. Ce dernier ne cherche alors qu'à en avoir la confirmation. Si le chiffrement est déterministe, il lui suffit de calculer $\texttt{Enc}^{\mathrm{pk}}(\mathbf{x})$ pour vérifier. Au contraire, si $\texttt{Enc}^{\mathrm{pk}}(\cdot)$ est probabiliste et que l'on s'assure que $\texttt{Enc}^{\mathrm{pk}}(\mathbf{y})$ et $\texttt{Enc}^{\mathrm{pk}}(\mathbf{z})$ sont calculatoirement indistinguables avec la connaissance de $\mathbf{y}$ et $\mathbf{z}$, notre adversaire ne pourra rien faire. Nous pouvons encore augmenter son pouvoir en supposant qu'il peut déchiffrer des messages de son choix afin de tenter de glaner quelque information que ce soit sur la clef secrète. Néanmoins nous ne considèrerons pas ici ce modèle de sécurité.

Commençons par rappeler la définition du modèle où l'adversaire a uniquement accès à la clef publique, un chiffré $\texttt{Enc}^{\mathrm{pk}}(\mathbf{x})$ et pour but de retrouver $\mathbf{x}$. On mesure alors son efficacité comme le succès OW-CPA (One-Wayness Chosen Plaintext Attack) :

Définition 1.8 (succès OW-CPA)**.** *Le succès* OW-CPA *d'un algorithme* $\mathcal{A}$ *contre un chiffrement à clef publique* $\mathcal{E} \triangleq (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ *est défini comme :*

$$Succ_{\mathcal{E}}^{\text{OW-CPA}}(\mathcal{A}) \triangleq \mathbb{P}\left(\mathcal{A}(\mathbf{y}, \mathrm{pk}) = \mathbf{x}\right)$$

où la probabilité est calculée sur l'aléa interne de $\mathcal{A}$*,* `KeyGen`*,* `Enc` *avec :*

$$(\mathrm{pk}, \mathrm{sk}) \hookleftarrow \texttt{KeyGen}(1^\lambda), \mathbf{x} \hookleftarrow \{0,1\}^{m(\lambda)} \quad \textit{et} \quad \mathbf{y} \hookleftarrow \texttt{Enc}(\mathrm{sk}, \mathbf{x}).$$

1. 1^λ est un vecteur de *taille* λ. La complexité des algorithmes est mesurée en fonction de la *taille* des entrées.

Nous définissons alors le succès OW-CPA *calculatoire de casser $\mathcal{E}$ en temps t :*

$$Succ_{\mathcal{E}}^{\text{OW-CPA}}(t) \stackrel{\triangle}{=} \max_{\mathcal{A}:|\mathcal{A}|\leqslant t} \left(Succ_{\mathcal{E}}^{\text{OW-CPA}}(\mathcal{A})\right)$$

avec $|\mathcal{A}|$ dénotant le temps d'exécution de l'algorithme $\mathcal{A}$.

Dans le second modèle de sécurité que nous considèrerons ici, l'objectif d'un adversaire est de distinguer parmi le chiffrement de deux messages de son choix. Nous parlons de succès IND-CPA (Indistinguishability Chosen Plaintext Attack) :

Définition 1.9 (succès IND-CPA)**.** *Soient $\mathcal{A}_1, \mathcal{A}_2$ deux algorithmes et $\mathcal{A} \stackrel{\triangle}{=} (\mathcal{A}_1, \mathcal{A}_2)$. Le succès* IND-CPA *de $\mathcal{A}$ contre un chiffrement à clef publique $\mathcal{E} \stackrel{\triangle}{=} (\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ est défini comme :*

$$Succ_{\mathcal{E}}^{\text{IND-CPA}}(\mathcal{A}) \stackrel{\triangle}{=} 2 \times \mathbb{P}\left(\mathcal{A}_2(\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}, \text{pk}) = b\right) - 1$$

où la probabilité est calculée sur l'aléa interne de $\mathcal{A}$, $\texttt{KeyGen}$ et $\texttt{Enc}$ avec :

$$(\text{pk}, \text{sk}) \hookleftarrow \texttt{KeyGen}(1^\lambda), b \hookleftarrow \{0,1\}, (\mathbf{x}_0, \mathbf{x}_1) \leftarrow \mathcal{A}_1(\text{pk}) \quad \textit{et} \quad \mathbf{y} \hookleftarrow \texttt{Enc}(\text{sk}, \mathbf{x}_b).$$

Nous définissons alors le succès IND-CPA *calculatoire de $\mathcal{E}$ en temps t :*

$$Succ_{\mathcal{E}}^{\text{IND-CPA}}(t) \stackrel{\triangle}{=} \max_{\mathcal{A}_1,\mathcal{A}_2:|\mathcal{A}_1|+|\mathcal{A}_2|\leqslant t} \left(Succ_{\mathcal{E}}^{\text{IND-CPA}}(\mathcal{A}_1, \mathcal{A}_2)\right)$$

avec $|\mathcal{A}_i|$ dénotant le temps d'exécution de l'algorithme $\mathcal{A}_i$.

Dans ce qui suit nous dirons d'un chiffrement à clef publique $\mathcal{E}$ qu'il est sûr OW-CPA (resp. IND-CPA) si pour toute fonction t du paramètre de sécurité λ :

$$\frac{t}{Succ_{\mathcal{E}}^{\text{OW-CPA}}(t)} \in \text{negl}(\lambda) \quad \left(\frac{t}{Succ_{\mathcal{E}}^{\text{IND-CPA}}(t)} \in \text{negl}(\lambda)\right).$$

Il est alors classique que [Poi19],

Proposition 1.10. *Tout schéma de chiffrement sûr* IND-CPA *l'est dans le modèle* OW-CPA*.*

1.3.1 Les systèmes de McEliece et Niederreiter

Généralités. Les chiffrements de McEliece et Niederreiter reposent sur la notion de codes à trappe.

Définition 1.10 (Codes à trappe à sens unique)**.** *Il s'agit d'une paire d'algorithmes probabilistes et polynomiaux $(\texttt{KeyTrap}, \texttt{InvDec})$ avec un triplet de fonctions $(n(\lambda), k(\lambda), w(\lambda))$ polynomiales en le paramètre de sécurité λ tels que (on omet la dépendance en λ) :*

- $\texttt{KeyTrap}$ *prend en entrée 1^λ et renvoie en temps polynomial $(\mathbf{H}, T)$ où $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ de rang $n-k$ et T sa trappe associée,*
- $\texttt{InvDec}$ *est un algorithme probabiliste polynomial, dit décodeur, prenant en entrée T, $\mathbf{s} \in \mathbb{F}_q^{n-k}$ et renvoyant $\mathbf{e} \in \mathbb{F}_q^n$ de poids de Hamming w tel que $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ ou un symbole $\perp$ en cas d'échec mais sous la condition que :*

  $$\mathbb{P}\left(\texttt{InvDec}(T, \mathbf{e}\mathbf{H}^\intercal) \neq \mathbf{e}\right) \in \text{negl}(\lambda)$$

  où la probabilité est calculée sur l'aléa interne de $\texttt{InvDec}$ et $\mathbf{e}$ uniformément tirée parmi les mots de poids w.

De plus :

— *Sens unique sans la trappe : pour tout algorithme probabiliste polynomial $\mathcal{A}$:*

$$\mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}) = \mathbf{e}\right) \in \mathrm{negl}(\lambda)$$

où la probabilité est calculée sur l'aléa interne de $\mathcal{A}$, `KeyTrap` *avec :*

$$(\mathbf{H}, T) \hookleftarrow \mathtt{KeyTrap}(1^\lambda), \mathbf{e} \hookleftarrow S_w, \mathbf{s} = \mathbf{e}\mathbf{H}^\intercal.$$

A titre d'exemple, si nous considérons la famille des codes de Reed-Solomon $\mathrm{RS}_k(\mathbf{x}) \triangleq \{(f(x_1), \cdots, f(x_n)) : f \in \mathbb{F}_q[X]_{<k}\}$ la trappe T associée est le vecteur $\mathbf{x}$ tandis que $\mathbf{H}$ est une matrice de parité aléatoire de ce code. En revanche comme nous le verrons un peu plus loin, bien que ces codes viennent avec un bon algorithme de décodage, ces derniers ne peuvent pas être considérés comme des codes à trappe à sens unique.

Une fois que l'on dispose de tels codes, l'idée de McEliece pour faire du chiffrement est la suivante. La clef publique est une matrice génératrice $\mathbf{G}$ du code de matrice de parité $\mathbf{H}$ donnée par `KeyTrap`. Le chiffré d'un message $\mathbf{m}$ est $\mathbf{m}\mathbf{G} + \mathbf{e}$ où $|\mathbf{e}| = w$. Le déchiffrement consiste alors à appliquer `InvDec`. Nous résumons la situation dans la table 1.1.

Table 1.1 – Chiffrement de McEliece

`KeyGen`(1^λ)
- $(\mathbf{H}, T) \leftarrow$ `KeyTrap`(1^λ)
- $\mathrm{sk} \leftarrow T$
- $\mathrm{pk} \leftarrow \mathbf{G} \in \mathbb{F}_q^{k\times n} : \mathbf{G}\mathbf{H}^\intercal = \mathbf{0}$
- `renvoie` $(\mathrm{pk}, \mathrm{sk})$

`Enc`$(\mathrm{pk}, \mathbf{m})$
- $\mathbf{e} \hookleftarrow S_w$
- $\mathbf{y} \leftarrow \mathbf{m}\mathbf{G} + \mathbf{e}$
- `renvoie` $\mathbf{y}$

`Dec`$(\mathrm{sk}, \mathbf{y})$
- $\mathbf{e} \leftarrow$ `InvDec`$(T, \mathbf{y}\mathbf{H}^\intercal)$
- $\mathcal{J} \subset [\![1, n]\!] : G_{\mathcal{J}}$ est inversible
- $\mathbf{m} \leftarrow (\mathbf{y} - \mathbf{e})_{\mathcal{J}}(\mathbf{G}_{\mathcal{J}})^{-1}$
- `renvoie` $\mathbf{m}$

Niederreiter a ensuite proposé une variante duale [Nie86] du chiffrement de McEliece. L'idée est de directement raisonner avec des syndromes et non avec des mots de codes bruités. La différence est qu'au lieu d'encoder le message à chiffrer on lui associe une erreur de poids w avec une fonction publique :

$$\varphi : \{0,1\}^m \to S_w$$

injective et facile à inverser. La table 1.2 décrit ce chiffrement à clef publique.

Une des différences entre ces deux schémas est la taille de clef. Dans le schéma de Niederreiter nous pouvons supposer que la matrice publique $\mathbf{H}$ est sous forme systématique alors que ce n'est évidemment pas le cas pour McEliece (à moins de rajouter une surcouche). Nous obtenons donc des clefs avec :

McEliece	Niederreiter
$kn\log_2(q)$ bits	$k(n-k)\log_2(q)$ bits

En revanche l'ajout de la fonction φ dans Niederreiter ralentit le temps du chiffrement et du déchiffrement.

La question cruciale est maintenant quelle trappe associer à un code pour être en mesure de décoder efficacement ? Depuis l'établissement du schéma de McEliece, une réponse à

Table 1.2 – Chiffrement de Niederreiter

$\texttt{KeyGen}(1^\lambda)$:
$(\mathbf{H}, T) \leftarrow \texttt{KeyTrap}(1^\lambda)$
$\mathrm{sk} \leftarrow T$
$\mathrm{pk} \leftarrow \mathbf{H}$
`renvoie` $(\mathrm{pk}, \mathrm{sk})$

$\texttt{Enc}(\mathrm{pk}, \mathbf{m})$:	$\texttt{Dec}(\mathrm{sk}, \mathbf{y})$:
$\mathbf{e} \leftarrow \varphi(\mathbf{m})$	$\mathbf{e} \leftarrow \texttt{InvDec}(T, \mathbf{s})$
$\mathbf{s} \leftarrow \mathbf{e}\mathbf{H}^\intercal$	$\mathbf{m} \leftarrow \varphi^{-1}(\mathbf{e})$
`renvoie` $\mathbf{s}$	`renvoie` $\mathbf{m}$

cette question fut donnée en proposant d'utiliser des codes structurés et non quelconques. Dans ce cas il faut se méfier, supposer la condition de sens unique sans la trappe de la définition 1.10 ne revient pas à supposer que tout adversaire cherchant à résoudre *le décodage générique* en temps polynomial réussit avec une probabilité négligeable, La proposition qui suit montre alors que le schéma est sûr dans le modèle OW-CPA (ce qui revient à vérifier la condition de sens unique sans la trappe) si le problème de distinguer le code défini par la clef publique d'un code aléatoire *et* le décodage générique (Problème 1.7) sont calculatoirement difficiles ($\rho_c(\cdot,\cdot)$ désigne la distance calculatoire).

Proposition 1.11 ([Sen11b])**.** *Soient* $R \in]0,1[$ *et* $\omega \in]0,1[$ *une fonction de* n*. Notons* $k \triangleq \lfloor Rn \rfloor$ *et* $w \triangleq \lfloor \omega n \rfloor$*. Soient* $\mathcal{E}$ *le chiffrement de McEliece ou de Neiderreiter et* $\mathcal{D}_{\text{alea}}$ *(resp.* $\mathcal{D}_{\text{pub}}$*) la distribution uniforme sur* $\mathbb{F}_q^{(n-k)\times n}$ *(resp. les matrices de parité du code défini par* pk *dans* $\mathcal{E}$*). Nous avons :*

$$Succ_{\mathcal{E}}^{\text{OW-CPA}}(t) \leqslant Succ^{\text{SD}(n,q,\omega,R)}(t) + \rho_c\left(\mathcal{D}_{\text{alea}}, \mathcal{D}_{\text{pub}}\right)(t)$$

où :

$$Succ^{\text{SD}(n,q,R,\omega)}(t) \triangleq \max_{\mathcal{A}:|\mathcal{A}|\leqslant t} \left(\mathbb{P}\big(\mathcal{A}(\mathbf{H},\mathbf{s}) = \mathbf{e}\big)\right)$$

avec :

$$\mathbf{H} \hookleftarrow \mathbb{F}_q^{(n-k)\times n}, \quad \mathbf{e} \hookleftarrow S_w \quad \textit{et} \quad \mathbf{s} \triangleq \mathbf{e}\mathbf{H}^\intercal.$$

Comme nous le verrons au chapitre 2, les meilleurs algorithmes pour résoudre SD en moyenne ont une complexité asymptotique donnée par :

$$2^{c\cdot w(1+o(1))}.$$

pour une constante c dépendant des rapports k/n et w/n fixés. On ne peut donc pas s'attendre à mieux en terme de taille de clefs que $\Theta(\lambda^2)$ et ce quand $w = \Theta(n)$, c'est à dire pour un décodeur très performant. Plus précisément, étant donné le contexte de chiffrement ($\mathbf{e} \mapsto \mathbf{e}\mathbf{H}^\intercal$ doit être injective) la meilleure distance que l'on peut atteindre asymptotiquement en n ici est donnée par la borne de Gilbert-Varshamov :

$$w/n \underset{n\to+\infty}{=} \omega^{-}(1 + o(1)).$$

En effet, il s'agit du poids limite où l'on s'attend à une solution typiquement unique au problème du décodage alors qu'au delà les solutions sont en nombre exponentiel.

Afin d'instancier son schéma, McEliece proposa d'utiliser des codes de Goppa binaires. Il s'avère que ces codes ont jusqu'à aujourd'hui résisté aux différentes cryptanalyses. Seul

le résultat de [Fau+10a] a montré comment distinguer ces derniers de codes quelconques dans le régime particulier où leur dimension est proche de leur longueur. Quoiqu'il en soit les codes de Goppa semblent être une bonne solution cryptographique. De plus, ces derniers peuvent décoder jusqu'à $\Theta(n/\log_2(n))$ erreurs pour k/n fixé.

Comme nous l'avons évoqué lors de l'introduction de ce chapitre, de nombreux codes furent proposés pour instancier le schéma de McEliece mais la plupart d'entre eux ne résistèrent pas à diverses attaques à l'exception d'un tout petit nombre de familles de codes comme les codes MDPC [Mis+12].

Nous allons maintenant décrire succinctement les codes de Goppa et MDPC ainsi que leurs algorithmes de décodage.

Les codes de Goppa et leur décodage. Les codes de Goppa proviennent de la famille des codes dits alternants eux-mêmes définis à partir des codes de Reed-Solomon généralisés.

Définition 1.11 (Code de Reed-Solomon généralisé)**.** *Soient $\mathbf{x} \in \mathbb{F}_q^n$ de coordonnées deux à deux disjointes, $\mathbf{y} \in (\mathbb{F}_q^*)^n$ et k un entier. On définit le code de Reed-Solomon généralisé de support $\mathbf{x}$ et coordonnée $\mathbf{y}$ comme :*

$$\mathrm{GRS}_k(\mathbf{x},\mathbf{y}) \triangleq \{(y_1 f(x_1), \cdots, y_n f(x_n)) \mid f \in \mathbb{F}_q[X]_{<k}\}.$$

On vérifie alors facilement la relation suivante :

$$\mathrm{GRS}_k(\mathbf{x},\mathbf{y})^\perp = \mathrm{GRS}_{n-k}(\mathbf{x},\mathbf{y}') \quad \text{où} \quad y'_i \triangleq \frac{1}{y_i \prod_{j\neq i}(x_i - x_j)} \quad \text{pour} \quad 1 \leqslant i \leqslant n.$$

De cette façon, $\mathrm{GRS}_k(\mathbf{x},\mathbf{y}')$ est un $[n,k]_q$-code de matrice de parité :

$$\mathbf{H} \triangleq \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{n-k-1} & x_2^{n-k-1} & \cdots & x_n^{n-k-1} \end{pmatrix} \begin{pmatrix} y'_1 & & & 0 \\ & y'_2 & & \\ & & \ddots & \\ 0 & & & y'_n \end{pmatrix} \tag{1.28}$$

L'intérêt de tels codes est qu'ils viennent avec l'algorithme déterministe de décodage de Berlekamp-Welch pouvant décoder jusqu'à $\lfloor \frac{n-k}{2} \rfloor$ erreurs. Décrivons-le rapidement.

Décodage des Reed-Solomon généralisés[Cou19, Ch. 8]. Raisonnons directement sur un mot bruité que l'on cherche à décoder :

$$\mathbf{z} \triangleq \mathbf{c} + \mathbf{e}$$

où $\mathbf{c} = (y_1 f(x_1), \cdots, y_n f(x_n)) \in \mathrm{GRS}_k(\mathbf{x},\mathbf{y})$ avec $f \in \mathbb{F}_q[X]_{<k}$ et $\mathbf{e} \in \mathbb{F}_q^n$ de poids de Hamming $\lfloor \frac{n-k}{2} \rfloor$. Considérons le cas simplifié où $y_1 = \cdots = y_n = 1$, le cas général s'en déduisant en multipliant les coordonnées de $\mathbf{z}$ par y_i^{-1}. Introduisons le polynôme inconnu :

$$E(X) \triangleq \prod_{i:e_i\neq 0} (X - e_i).$$

La clef de l'algorithme est alors le fait que :

$$\forall i \in [\![1,n]\!], \quad z_i E(x_i) = f(x_i) E(x_i). \tag{1.29}$$

Les coordonnées z_i et x_i sont connues par le décodeur contrairement aux coefficients de f et E. Le système (1.29) n'est alors pas linéaire. L'idée est donc de procéder à une linéarisation :

$$N \triangleq Ef$$

Ainsi, (1.29) devient :

$$\forall i \in [\![1, n]\!], \quad z_i E(x_i) = N(x_i)$$

où les coefficients du polynôme $N \in \mathbb{F}_q[X]_{<k+\lfloor (n-k)/2 \rfloor}$ sont inconnus. On a donc un système linéaire sur-déterminé avec n équations et $k + \lfloor \frac{n-k}{2} \rfloor + \lfloor \frac{n-k}{2} \rfloor \leqslant n$ inconnues du fait des degrés de E et f. Ce système a alors (E, Ef) comme solution et toute autre solution permet de retrouver f grâce à l'identité $\frac{N_1}{E_1} = \frac{N_2}{E_2}$ pour tout couple $(E_1, N_1), (E_2, N_2)$ de solutions. Le décodeur trouve donc une solution (E, N) pour en déduire le mot de code solution $\mathbf{c} = (f(x_1), \cdots, f(x_n))$ avec $f \stackrel{\triangle}{=} E/N$.

Les Reed-Solomon généralisés peuvent donc sembler être de bon candidats pour le sytème de McEliece ou de Niederreiter. Malheureusement ces derniers sont caducs dans ce contexte depuis l'attaque de [SS92] ou encore [Cou+14] où un distingueur est donné. De plus il est à noter, cette fois-ci du côté de la théorie des codes, que leurs paramètres sont extrêmement contraints, leur longueur est nécessairement inférieure à la taille du corps :

$$n \leqslant q$$

du fait que $x_i \neq x_j$ pour $i \neq j$. L'idée naturelle pour se défaire de ce carcan tout en conservant les qualités du décodage a alors été d'utiliser une extension de corps $\mathbb{F}_{q^m}$ sur laquelle définir le Reed-Solomon généralisé de longueur $n \leqslant q^m$ que l'on restreint ensuite au petit corps $\mathbb{F}_q$. Ainsi en choisissant $m = \log_q n$ on obtient un sous-code de longueur n sur $\mathbb{F}_q$. Ceci correspond exactement à la définition des codes alternants :

Définition 1.12 (Code alternant)**.** *Soient les entiers n, m tels que $n \leqslant q^m$, $\mathbf{x} \in \mathbb{F}_{q^m}^n$ un mot dont les coordonnées sont deux à deux disjointes, $\mathbf{y} \in (\mathbb{F}_{q^m}^*)^n$ et k un entier. On définit le code alternant $\mathcal{A}_k(\mathbf{x}, \mathbf{y})$ de support $\mathbf{x}$ et coordonnée $\mathbf{y}$ comme :*

$$\mathcal{A}_k(\mathbf{x}, \mathbf{y}) \stackrel{\triangle}{=} \mathsf{GRS}_k(\mathbf{x}, \mathbf{y}) \cap \mathbb{F}_q^n.$$

Les codes alternants définissent une large classe de codes dits algébriques : les BCH [MS86, Chapitre 9], les Srivastava [MS86, Chapitre 12], les Goppa [MS86, Chapitre 12] etc... Leurs propriétés sont aujourd'hui bien connues et en particulier :

— $\tilde{\mathbf{H}}$ où l'on décompose chaque colonne dans $\mathbb{F}_q$ de $\mathbf{H}$ (voir (1.28)) est une matrice de parité de $\mathcal{A}_k(\mathbf{x}, \mathbf{y})$,

— Nous avons :

$$\dim(\mathcal{A}_k(\mathbf{x}, \mathbf{y})) \geqslant n - m(n-k) \tag{1.30}$$

De plus on sait les décoder efficacement. Une solution consiste à appliquer directement le décodage de Berlekamp-Welch. On peut donc décoder $\mathcal{A}_k(\mathbf{x}, \mathbf{y})$ à distance $\lfloor \frac{n-k}{2} \rfloor$ qui est égale avec une bonne probabilité à :

$$\left\lfloor \frac{n - \dim(\mathcal{A}_k(\mathbf{x}, \mathbf{y}))}{2m} \right\rfloor \tag{1.31}$$

Les codes de Goppa sont maintenant définis comme une sous-famille des codes alternants $\mathcal{A}_k(\mathbf{x}, \mathbf{y})$ où une relation particulière entre $\mathbf{x}$ et $\mathbf{y}$ est imposée.

Définition 1.13 (Codes de Goppa)**.** *Soient un entier m, un polynôme $G \in \mathbb{F}_{q^m}[X]$, $k \stackrel{\triangle}{=} \deg(G)$ et $\mathbf{x} \in \mathbb{F}_{q^m}^n$ de coordonnées deux à deux distinctes. On définit le code de Goppa $\Gamma(G, \mathbf{x})$:*

$$\Gamma(G, \mathbf{x}) \stackrel{\triangle}{=} \mathcal{A}_k(\mathbf{x}, \mathbf{y}) \quad \textit{où} \quad \forall i \in [\![1, n]\!],\ y_i = \frac{G(x_i)}{\Pi_{j \neq i}(x_i - x_j)}.$$

McEliece a proposé d'utiliser ces codes avec $q = 2$ (on parle de Goppa binaire), $m = \log_2(n)$ et G un polynôme sans facteur multiple. L'intérêt d'un tel choix est qu'avec de tels polynômes nous pouvons multiplier la distance de décodage par deux (voir entre autres [Sen02]). Ces codes de Goppa ont donc un avantage par rapport aux codes alternants généraux. La trappe associée à une matrice quelconque de parité ou génératrice de $\Gamma(G, \mathbf{x})$ (pas besoin de permuter) consiste alors tout simplement en G et $\mathbf{x}$.

Les codes MDPC et leur décodage. Les codes MDPC [Mis+12] appartiennent contrairement aux Goppa à la famille des codes munis d'un décodage probabiliste. Commençons par rappeler la définition de ces codes :

Définition 1.14 (Codes MDPC[Mis+12])**.** *Un $[n, k]_2$-code est dit* MDPC *s'il admet une matrice de parité $\mathbf{H} = (h_{i,j})_{\substack{1 \leqslant i \leqslant n-k \\ 1 \leqslant j \leqslant n}}$ dont les lignes sont de poids de Hamming en $O(\sqrt{n})$:*

$$\forall i \in [\![1, n-k]\!], \quad |(h_{i,j})_{1 \leqslant j \leqslant n}| = O(\sqrt{n}).$$

Cette base du dual de poids *modéré*, qui est la trappe associée aux MDPC, permet un algorithme de décodage efficace à distance $\Omega(\sqrt{n})$. Cet algorithme est inspiré du décodage des codes LDPC [Gal63] définis comme les codes MDPC mais où le poids des vecteurs d'une base du dual est en $O(1)$. Décrivons sommairement l'idée de cet algorithme dans le contexte des codes MDPC.

Un décodage des codes MDPC. On pourra voir [Mis+12 ; CS16b ; Cha17 ; SV18 ; Til18a] pour plus de détails. Fixons une matrice de parité $\mathbf{H}$ d'un MDPC dont les $n-k$ lignes $\mathbf{h}_i$ sont de poids $t = O(\sqrt{n})$. Soit $\mathbf{s} \triangleq \mathbf{e}\mathbf{H}^\intercal$ un syndrome que l'on cherche à décoder où $w \triangleq |\mathbf{e}|$ vérifiant :

$$w = O(\sqrt{n}).$$

Notre objectif est de retrouver $\mathbf{e}$ avec la connaissance de $\mathbf{s}$ et des équations de parités $\mathbf{h}_i$. Il est clair que :

$$\forall j \in [\![1, n-k]\!], \quad \langle \mathbf{h}_j, \mathbf{e} \rangle = s_j$$

Le point de départ pour retrouver $\mathbf{e}$ est alors la remarque que ces produits scalaires sont biaisés du fait du poids modéré des $\mathbf{h}_j$ et de $|\mathbf{e}|$. Plus particulièrement, si $\mathbf{h}_j(i) = 1$, la probabilité que $\langle \mathbf{e}, \mathbf{h}_j \rangle$ soit égal à 1 n'est pas la même selon que $\mathbf{e}(i) = 1$ ou 0 comme montré dans le lemme qui suit,

Lemme 1.5 ([Til18b, lemme 2])**.** *Soit $\mathbf{e}$ une variable aléatoire uniformément distribuée sur les mots de $\mathbb{F}_2^n$ de poids w et $\mathbf{h} \in \mathbb{F}_2^n$ de poids t. Supposons que $w, t = O(\sqrt{n})$ et $h_i = 1$ pour $i \in [\![1, n]\!]$. Nous avons pour $b \in \{0, 1\}$,*

$$\mathbb{P}_{\mathbf{e}}\left(\langle \mathbf{e}, \mathbf{h} \rangle = 1 \mid e_i = b\right) = \frac{1}{2} - (-1)^b \varepsilon \left(\frac{1}{2} + O\left(\frac{1}{\sqrt{n}}\right)\right)$$

où $\varepsilon \triangleq e^{\frac{-wt}{n}}$.

Le rationnel du décodage des MDPC consiste alors à faire des tests statistiques avec un vote majoritaire pour décider si $e_i = 1$ ou non. L'échantillon est ici donné par les équations de parité $\mathbf{h}_j$ telles que $\mathbf{h}_j(i) = 1$. Nous nous attendons lors du décodage pour une position $i \in [\![1, n]\!]$ telle que $e_i = 1$ à ce que les bits $\langle \mathbf{h}_j, \mathbf{e} \rangle = s_j$ soient majoritairement égaux à 1 en les positions j données par les équations de parité $\mathbf{h}_j$ vérifiant $\mathbf{h}_j(i) = 1$. L'algorithme consiste alors tout simplement à calculer pour une position $i \in [\![1, n]\!]$:

$$N_i \triangleq \#\left\{j \in [\![1, n-k]\!] : \mathbf{h}_j(i) = 1\right\} \quad \text{(taille de l'échantillon)}$$

$$C_i \triangleq \#\{j \in [\![1, n-k]\!] : \mathbf{h}_j(i) = s_j = 1\} \quad \text{(le compteur)}$$

et si $C_i > N_i/2$ on ajoute la i-ème colonne de $\mathbf{H}$ à $\mathbf{s}$ (on dit que le bit i a été flippé). On effectue alors cette opération pour tous les bits de position $i \in [\![1, n]\!]$. On espère alors avoir diminué le poids de l'erreur. On itère ensuite ce procédé jusqu'à obtenir le syndrome nul. L'erreur recherchée est alors donnée par les positions des colonnes que l'on a ajouté au syndrome.

Malheureusement, les échantillons étant de taille assez faible, l'algorithme a peu de chance de s'arrêter, il y a une probabilité non nulle d'échec. Une des idées de Gallager [Gal63] pour le décodage des LDPC est de ne flipper que les bits i pour lesquels les compteurs C_i sont maximaux. Il est alors démontré [Gal63] que l'algorithme terminera et renverra la bonne erreur avec une très bonne probabilité. Les auteurs de [Mis+12] ont proposé de procéder d'une façon similaire en ne flippant que les bits ayant un compteur compris entre $M - \delta$ et M où M dénote la valeur du compteur maximal et δ un paramètre à optimiser. Dans les travaux de [CS16b ; Cha17 ; SV19] il a été proposé pour améliorer la probabilité de réussite du décodage d'adapter à chaque itération la valeur de δ en fonction du poids du syndrome.

Si nous utilisons les codes MDPC dans les systèmes de McEliece ou Niederreiter la taille de clef sera pour le paramètre de sécurité λ en λ^4 étant donné que leur décodage est à distance $\sqrt{n}$. Il a alors été proposé dans [Mis+12] de leur adjoindre une structure quasi-cyclique que nous décrivons rapidement dans ce qui suit.

Remarque 1.6. La probabilité d'échec au décodage est cependant un problème pour l'utilisation de codes MDPC en chiffrement. En effet, une attaque contre les premiers paramètres a été donnée [Fab+17] dans un modèle de sécurité où l'adversaire a accès à un oracle de déchiffrement. Les récents travaux de [Til18a] ont cependant montré comment choisir les paramètres afin de rendre cette probabilité d'échec au décodage asymptotiquement négligeable.

Les codes MDPC doublement-circulants. Commençons par rappeler la définition des codes doublement-circulants (cas particulier des codes quasi-cycliques).

Définition 1.15 (Codes doublement-circulant)**.** *Un $[n, n/2]_2$-code est dit doublement-circulant si tout décalage de $n/2$ positions d'un mot de code est encore un mot de code.*

Une matrice de parité d'un tel code est alors donnée par deux blocs de taille $n/2 \times n/2$ où chaque ligne est obtenue comme le décalage à droite de la précédente, ce que l'on note :

$$\begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_1 \\ \circlearrowright & \circlearrowright \end{pmatrix}$$

où $\mathbf{h}_0, \mathbf{h}_1 \in \mathbb{F}_2^{n/2}$. On peut donc représenter ces codes avec les deux vecteurs $\mathbf{h}_0$ et $\mathbf{h}_1$ et non toute une matrice. Il est classique d'identifier ces codes à des polynômes grâce au fait suivant,

Fait 2. *L'application,*

$$\begin{array}{rccc} \varphi : & \mathbb{F}_2^{p\times p} & \longrightarrow & \mathbb{F}_2[X]/(X^p-1) \\ & \begin{pmatrix} m_0 & m_1 & \cdots & m_{p-1} \\ m_{p-1} & m_1 & \cdots & m_{p-2} \\ \vdots & \ddots & \ddots & \vdots \\ m_1 & m_2 & \cdots & m_0 \end{pmatrix} & \longmapsto & \sum_{i=0}^{p-1} m_i X^i \end{array}$$

est un isomorphisme de $\mathbb{F}_2$-algèbres. Il s'étend aux vecteurs $\mathbf{x} \in \mathbb{F}_2^p$ comme :

$$\varphi(\mathbf{x}) = \sum_{i=0}^{p-1} x_{i+1} X^i \in \mathbb{F}_2[X]/(X^p - 1)$$

et alors,

$$\varphi(\mathbf{xM}) = \varphi(\mathbf{x})\varphi(\mathbf{M}).$$

Un code doublement-circulant $\mathcal{C}$ peut s'écrire avec deux polynômes $h_0, h_1 \in \mathbb{F}_2[X]/(X^{n/2}-1)$ comme :

$$\mathcal{C} = \left\{(c_1, c_2) : c_1 h_1 + c_2 h_2 = 0 \mod (X^{n/2}-1)\right\}.$$

On suppose dans la suite que h_1 est inversible modulo $X^{n/2}-1$. Le code peut donc s'écrire de la façon suivante (il s'agit ni plus ni moins d'une mise sous forme systématique) :

$$\mathcal{C} = \left\{(c_1, c_2) : c_1 + c_2 h_2 / h_1^{-1} = 0 \mod (X^{n/2}-1)\right\}.$$

Le problème du décodage générique des codes doublement-circulants peut maintenant s'exprimer comme (où le poids d'un polynôme dénote son nombre de coefficients non nuls) :

Problème 1.10. *Étant donné $h, s \in \mathbb{F}_2[X]/(X^{n/2}-1)$ et w, trouver $e_1, e_2 \in \mathbb{F}_2[X]/(X^{n/2}-1)$ de poids w tels que $e_1 + e_2 h = s \mod (X^{n/2}-1)$.*

Comme évoqué lors de l'introduction, les meilleurs algorithmes pour résoudre ce problème ont la même complexité, à un facteur polynomial près, que ceux résolvant le problème du décodage générique. Seul le récent travail de [CT19] a montré que dans certaines zones de paramètres il était possible d'obtenir un gain exponentiel.

Dans ce contexte un QC-MDPC se définit [Mis+12] tout simplement comme un $[n, n/2]_2$-code MDPC auquel on adjoint une structure doublement-circulante :

$$\mathcal{C}_{\text{QC-MCDPC}} \triangleq \left\{(c_1, c_2) : c_1 + c_2 h_2 / h_1^{-1} = 0 \mod (X^{n/2}-1) \text{ et } |h_2| = |h_1| = w\right\}$$

où $|\cdot|$ est le poids d'un polynôme et :

$$w = O(\sqrt{n}).$$

Le décodage de ces codes est alors le même que celui des MDPC. La clef publique dans le système de Niederreiter est tout simplement h_2/h_1^{-1}, donc formée de $n/2$ bits. La trappe associée est le couple de polynômes creux (h_1, h_2). Le chiffrement d'un message $\mathbf{m}$ consiste à lui associer un couple de polynômes (e_1, e_2) de poids w puis de calculer $e_1 + e_2 h_2 / h_1^{-1}$.

Il est alors démontré [Ara+17a] que la sécurité repose, non pas sur la difficulté du problème SD ou la difficulté de distinguer le code utilisé d'un code aléatoire (voir la proposition 1.11) mais sur la difficulté des problèmes 1.10 *ou* :

Problème 1.11. *Étant donné $h \in \mathbb{F}_2[X]/(X^{n/2}-1)$ et w existe t-il $c_1, c_2 \in \mathbb{F}_2[X]/(X^{n/2}-1)$ de poids w tels que $c_1 + c_2 h = 0 \mod (X^{n/2}-1)$?*

Ce problème est exactement la version décisionnelle du problème 1.10 pour $s = 0$. Bien que les meilleurs algorithmes pour les résoudre sont les mêmes, il n'existe pas à ce jour de réduction entre les deux.

Il est de plus à noter que le problème 1.11 peut être vu comme le pendant du problème NTRU [HPS98] mais en métrique de Hamming. Ainsi, avec des codes MDPC doublement-circulants nous avons une instanciation en métrique de Hamming proche des chiffrements utilisant NTRU.

1.3.2 La version duale du chiffrement d'Alekhnovich

Alekhnovich a proposé [Ale11] deux systèmes de chiffrement qui, contrairement aux schémas de McEliece et Niederreiter, ne font reposer la sécurité que sur le problème du décodage d'un code aléatoire. Nous avons donné en introduction le premier des deux mais qui ne permet que de chiffrer des bits. Nous présentons dans ce qui suit une version duale de son deuxième chiffrement. Ce schéma a été introduit sous le nom de `RankPKE` dans [Gab+17b] et dans le contexte de la métrique rang. Ce système s'étend alors naturellement pour la métrique de Hamming comme nous l'avons décrit dans [DT18b]. Les problèmes sur lesquels il repose en métrique de Hamming sont SD et la version décisionnelle du problème DOOM (Decoding One Out of Many) qui est SD où au lieu de un est donné $N > 1$ syndromes tandis qu'il est demandé d'en décoder uniquement l'un d'entre-eux. Il y a ici une différence avec la métrique rang où comme nous le verrons dans §1.3.5, la sécurité repose de même sur RSD avec plusieurs syndromes mais en revanche tous produits à partir d'erreurs ayant en plus du même poids un $\mathbb{F}_q$-support identique.

Définissons le problème DOOM dans sa variante de recherche.

Problème 1.12 (Problème DOOM - DOOM(n, q, R, ω, N))**.** *Soient les entiers* $k \triangleq \lfloor Rn \rfloor$ *et* $w \triangleq \lfloor \omega n \rfloor$.

- Instance : *une matrice de parité* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *de rang* $n-k$*,* N *syndromes* $\mathbf{s}_1, \cdots, \mathbf{s}_N \in \mathbb{F}_q^{n-k}$.
- Recherche : *un vecteur d'erreur* $\mathbf{e} \in \mathbb{F}_q^n$ *de poids de Hamming* w *et* $i \in [\![1, N]\!]$ *tels que* $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}_i$.

Ce problème apparaît naturellement comme nous le verrons plus tard dans les réductions de sécurité des signatures utilisant des codes dans le paradigme du hache et signe. De plus nous étudierons les meilleurs algorithmes le résolvant dans les chapitres 2 de cette partie ainsi que dans la partie II.

La version décisionnelle de ce problème se définit comme :

Problème 1.13 (Problème décisionnel DOOM - DDOOM(n, q, R, ω, N))**.** *Soient les entiers* $k \triangleq \lfloor Rn \rfloor$ *et* $w \triangleq \lfloor \omega n \rfloor$.

- *Distributions :*
  - $\mathcal{D}_0$ *:* $(\mathbf{H}, \mathbf{s}_1^{\text{unif}}, \cdots, \mathbf{s}_N^{\text{unif}})$ *distribuée uniformément sur* $\mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k} \times \cdots \times \mathbb{F}_q^{n-k}$*,*
  - $\mathcal{D}_1$ *:* $(\mathbf{H}, \mathbf{e}_1\mathbf{H}^\intercal, \cdots, \mathbf{e}_N\mathbf{H}^\intercal)$ *où* $\mathbf{H}$ *et* $(\mathbf{e}_1, \cdots, \mathbf{e}_N)$ *uniformément distribués sur* $\mathbb{F}_q^{(n-k)\times n}$ *et* $S_w \times \cdots \times S_w$.
- Entrée : $(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N)$ *pour* $(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N) \hookleftarrow \mathcal{D}_b$ *avec* $b \hookleftarrow \{0, 1\}$
- Décision : $b' \in \{0, 1\}$.

On définit alors le succès de résoudre ce problème :

Définition 1.16. *L'avantage* DDOOM *d'un algorithme* $\mathcal{A}$ *est défini comme :*

$$Adv^{\text{DDOOM}(n,q,R,\omega,N)}(\mathcal{A}) \triangleq \frac{1}{2}\big(\mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N) = 1 \mid b = 1\right) - \mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N) = 1 \mid b = 0\right)\big) \quad (1.32)$$

où les probabilités sont calculées sur l'aléa interne de $\mathcal{A}$*,* b *et l'entrée tirée selon* $\mathcal{D}_b$ *définie dans le problème 1.13 de paramètres* (q, R, ω, N)*. Nous définissons alors le succès* DDOOM *calculatoire en temps* t *:*

$$Succ^{\text{DDOOM}(n,q,R,\omega,N)}(t) \triangleq \max_{\mathcal{A}:|\mathcal{A}|\leqslant t} \left(Adv^{\text{DDOOM}(n,q,R,\omega,N)}(\mathcal{A})\right)$$

Présentons maintenant le schéma `RankPKE` [Gab+17b] en métrique de Hamming. Le lemme suivant sera fondamental pour la correction de la construction.

Lemme 1.6 ([Til18b, Appendice B, lemme 6])**.** *Soient* $\mathbf{e}_1$ *une variable aléatoire uniformément distribuée sur les mots de* $\mathbb{F}_2^n$ *de poids* w *et* $\mathbf{e}_2 \in \mathbb{F}_2^n$ *de poids* w*. Supposons que* $w^2 = O(n)$*, on a :*

$$\mathbb{P}_{\mathbf{e}_1}\left(\langle \mathbf{e}_1, \mathbf{e}_2 \rangle = 1\right) = \frac{1}{2}\left(1 - 2^{-c\frac{w^2}{n}}\left(1 + O\left(\frac{1}{\sqrt{n}}\right)\right)\right)$$

pour la constante $c = 2\log_2(e)$.

Le schéma. Commençons par fixer les paramètres du système. Soient $n \in \mathbb{N}$ une fonction du paramètre de sécurité λ dont on omet la dépendance. Les premiers paramètres sont :

$$N \in \mathbb{N} \text{ (une fonction de } n) \quad ; \quad R \in]0,1[\quad ; \quad \omega \in]0,1[\tag{1.33}$$

où

$$\omega = O\left(1/\sqrt{n}\right). \tag{1.34}$$

On définit :

$$k \triangleq Rn \quad \text{et} \quad w \triangleq \omega n$$

Soit $w_{\text{dec}} \in [\![1, N]\!]$ une fonctions de n telle que :

$$1 - 2^{-c\frac{w^2}{n}} \leqslant (1-\varepsilon)\frac{w_{\text{dec}}}{N} \tag{1.35}$$

où c est la constante du lemme 1.6 et $\varepsilon > 0$. De plus, soient $k_{\text{dec}} \in [\![1, N]\!]$ et $\mathbf{G}_{\text{dec}} \in \mathbb{F}_2^{k_{\text{dec}} \times N}$ la matrice génératrice d'un code pour lequel on connaît un algorithme $\mathcal{A}$ en temps polynomial de décodage à distance w_{dec} :

$$\mathcal{A}\left(\mathbf{m}\mathbf{G}_{\text{dec}} + \mathbf{e}\right) = \mathbf{e}$$

avec une bonne probabilité pour toute erreur $\mathbf{e} \in \mathbb{F}_2^N$ de poids $\leqslant w_{\text{dec}}$. Décrivons maintenant le schéma.

`KeyGen`$(\mathbf{1}^{\boldsymbol{\lambda}})$ **:**

- $\mathbf{H} \hookleftarrow \mathbb{F}_2^{(n-k)\times n}$,
- $\mathbf{x} \hookleftarrow \mathbb{F}_2^{n-k}$ et $\mathbf{e} \hookleftarrow S_{w,n}$,
- $\text{pk} \leftarrow (\mathbf{H}, \mathbf{x}\mathbf{H} + \mathbf{e}, \mathbf{G}_{\text{dec}})$,
- $\text{sk} \leftarrow \mathbf{x}$,
- `renvoie` (pk, sk).

`Enc`$(\mathbf{pk}, \mathbf{m})$ **:** Pour chiffrer un message $\mathbf{m} \in \mathbb{F}_2^{k_{\text{dec}}}$ on commence par générer N erreurs de poids w :

$$\forall i \in [\![1, N]\!], \quad \mathbf{e}_i \hookleftarrow S_{w,n}$$

et soit $\mathbf{E}$ la matrice obtenue par concaténation des N erreurs $\mathbf{e}_i$:

$$\mathbf{E} \triangleq \begin{pmatrix} \mathbf{e}_1^\intercal & \cdots & \mathbf{e}_N^\intercal \end{pmatrix} \in \mathbb{F}_2^{n\times N}.$$

On calcule maintenant le chiffré $\begin{pmatrix}\mathbf{C}\\ \mathbf{y}\end{pmatrix} \in \mathbb{F}_2^{(n-k+1)\times N}$ comme :

$$\begin{pmatrix}\mathbf{C}\\ \mathbf{y}\end{pmatrix} \triangleq \begin{pmatrix} \mathbf{H}\mathbf{E} \\ (\mathbf{x}\mathbf{H} + \mathbf{e})\,\mathbf{E} + \mathbf{m}\mathbf{G}_{\text{dec}} \end{pmatrix}.$$

`Dec`$(\mathbf{sk}, \mathbf{y})$ **:** On utilise la clef secrète $\mathbf{x}$ pour calculer :

$$\begin{aligned}(\mathbf{x}, -\mathbf{1})\begin{pmatrix}\mathbf{C}\\ \mathbf{y}\end{pmatrix} &= \mathbf{xC} - \mathbf{y}\\ &= \mathbf{xHE} - (\mathbf{xH}+\mathbf{e})\mathbf{E} - \mathbf{mG}_{\text{dec}}\\ &= -\mathbf{eE} - \mathbf{mG}_{\text{dec}}\\ &= -(\mathbf{mG}_{\text{dec}} + (\langle \mathbf{e}, \mathbf{e}_i\rangle)_{1\leqslant i\leqslant N})\end{aligned}$$

On applique ensuite l'algorithme de décodage $\mathcal{A}$ à ce mot. Or w et w_{dec} ont été choisis comme $w^2 = O(n)$ et w_{dec} de façon à ce que :

$$1 - 2^{-c\frac{w^2}{n}} \leqslant (1-\varepsilon)\frac{w_{\text{dec}}}{N}.$$

pour $\varepsilon > 0$. Donc d'après le lemme 1.6, le poids typique de $(\langle \mathbf{e}, \mathbf{e}_i\rangle)_{1\leqslant i\leqslant N}$ sera asymptotiquement en n inférieur à $(1-\varepsilon)w_{\text{dec}}$ et l'algorithme $\mathcal{A}$ renverra $\mathbf{mG}_{\text{dec}}$ (dont on déduit facilement $\mathbf{m}$) avec une probabilité tendant exponentiellement vers 1 en n.

La réduction de sécurité. Le schéma que nous venons de décrire est, comme montré dans théorème qui suit (énoncé dans [Gab+17b]), sûr IND-CPA sous l'hypothèse que les problèmes DSD et DDOOM sont calculatoirement difficiles.

Théorème 1.2. *Soit $\mathcal{E}$ le schéma de chiffrement* `RankPKE` *en métrique de Hamming. On a :*

$$Succ_{\mathcal{E}}^{\text{IND-CPA}}(t) \leqslant Succ^{\text{DSD}(n,2,1-R,\omega)}(t) + Succ^{\text{DDOOM}(n,2,R,\omega,N)}(t)$$

pour les paramètres du système $R \in]0,1[$, $N \in \mathbb{N}$ et $\omega = O(1/\sqrt{n})$.

Démonstration du Théorème 1.2.

Il suffit de reprendre la démonstration de [Gab+17b, Théorème 1 dans §3.3]. Intuitivement la sécurité sur DSD provient de la clef publique $(\mathbf{H}, \mathbf{xH}+\mathbf{e})$ et celle de DDOOM du chiffrement où $(\mathbf{H}, \mathbf{HE})$ est connu. □

Comme nous l'avons vu avec les MDPC, des mots de poids en $O(\sqrt{n})$ dans le dual d'un code permettent un algorithme de décodage à distance $O(\sqrt{n})$. Dans le chiffrement `RankPKE` et plus généralement dans ceux d'Alekhnovich [Ale11] des mots de poids encore en $O(\sqrt{n})$ servent de trappe. La remarque fondamentale faisant fonctionner ces schémas est que le produit scalaire de deux mots de longueur n et de *poids de Hamming* en $O(\sqrt{n})$ donnera 1 avec une probabilité $< 1/2$.

1.3.3 Le chiffrement HQC [AM+18]

Le chiffrement `RankPKE` utilise de façon cruciale le fait que n produits scalaires de mots de poids $\sqrt{n}$ donnent un mot de $\mathbb{F}_2^n$ de petits poids. Il fut proposé dans [AM+18] un schéma reposant sur une idée similaire mais cette fois-ci utilisant des codes quasi-cycliques aléatoires. L'idée est comme nous allons le décrire succinctement d'identifier $\mathbb{F}_2^n$ aux polynômes de $\mathbb{F}_2[X]/(X^n+1)$ et d'utiliser maintenant le fait que le produit de deux polynômes creux est encore un polynôme creux. Ceci permet alors de concevoir un chiffrement proche de `RankPKE`. En revanche. la sécurité IND-CPA du chiffrement [AM+18] ne repose que sur une hypothèse calculatoire, la difficulté de la version décisionnelle du problème de décodage d'un code aléatoire quasi-cyclique.

Commençons par considérer,

$$\begin{array}{rccc}\varphi : & \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2[X]/(X^n+1)\\ & \mathbf{x} & \longmapsto & \sum_{i=0}^{n-1} x_{i+1}X^i\end{array}$$

Cette application est linéaire, bijective ce qui nous permet d'identifier les deux espaces vectoriels. De plus, nous définissons la multiplication de deux vecteurs $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ comme le vecteur obtenu par multiplication de leur polynôme associé :

$$\mathbf{xy} \triangleq \varphi^{-1}\left(\varphi(\mathbf{x})\varphi(\mathbf{y})\right)$$

Fait 3. *Nous avons,*

$$|\mathbf{xy}| \leqslant |\mathbf{x}| \cdot |\mathbf{y}|.$$

Le schéma. Ici $n \in \mathbb{N}$ désigne une fonction du paramètre de sécurité λ. Nous introduisons,

$$w \triangleq \omega n \quad \text{avec} \quad \omega = O(1/\sqrt{n}). \tag{1.36}$$

Considérons maintenant un $[n,k]_2$-code $\mathcal{C}$, spécifié par une matrice génératrice $\mathbf{G}$, que l'on sait décoder à distance w_{dec} où $w_{\text{dec}} = \Omega(n)$.

$\texttt{KeyGen}(\mathbf{1^\lambda})$ **:**

- $\mathbf{x}, \mathbf{y} \hookleftarrow S_{w,n}$,
- $\mathbf{h} \hookleftarrow \mathbb{F}_2^n$,
- $\mathrm{pk} \leftarrow (\mathbf{h}, \mathbf{x} + \mathbf{hy})$,
- $\mathrm{sk} \leftarrow (\mathbf{x}, \mathbf{y})$
- `renvoie` $(\mathrm{pk}, \mathrm{sk})$.

$\texttt{Enc}(\mathbf{pk}, \mathbf{m})$ **:** Pour chiffrer un message $\mathbf{m} \in \mathbb{F}_2^k$ nous commençons par générer $\mathbf{r}_1, \mathbf{r}_2, \mathbf{e} \hookleftarrow S_{w,n}$ puis nous renvoyons :

$$(\mathbf{u}, \mathbf{v}) \triangleq (\mathbf{r}_1 + \mathbf{hr}_2, \mathbf{mG} + (\mathbf{x} + \mathbf{hy})\mathbf{r}_2 + \mathbf{e})$$

$\texttt{Dec}(\mathbf{sk}, (\mathbf{u}, \mathbf{v}))$ **:** Nous utilisons une partie de la clef secrète $\mathbf{y}$ pour calculer :

$$\begin{aligned}\mathbf{v} - \mathbf{uy} &= \mathbf{mG} + (\mathbf{x} + \mathbf{hy})\mathbf{r}_2 + \mathbf{e} - \mathbf{r}_1\mathbf{y} - \mathbf{hr}_2\mathbf{y} \\ &= \mathbf{mG} + \mathbf{xr}_2 - \mathbf{r}_1\mathbf{y} + \mathbf{e}\end{aligned}$$

Nous terminons alors le déchiffrement en décodant dans le code $\mathcal{C}$ de matrice génératrice $\mathbf{G}$.

La correction du déchiffrement repose tout simplement sur le raisonnement suivant. D'après le fait 3, le poids du terme d'erreur lors du décodage vérifie,

$$|\mathbf{xr}_2 - \mathbf{r}_1\mathbf{y} + \mathbf{e}| \leqslant |\mathbf{x}| \cdot |\mathbf{r}_2| + |\mathbf{r}_1| \cdot |\mathbf{y}| + |\mathbf{e}|$$

Or ici $|\mathbf{x}|, |\mathbf{r}_2|, |\mathbf{r}_1|, |\mathbf{y}| = w$ avec $w^2 = O(n)$. La capacité de correction du code est cependant choisie comme $\Omega(n)$. Il suffit alors de choisir correctement les constantes pour que le déchiffrement fonctionne. Il nous faut cependant utiliser une famille de codes avec une très bonne capacité de correction. Les auteurs [AM+18] du schéma proposèrent les codes BCH [MS86]. L'avantage de ces codes est qu'ils peuvent se représenter de façon très compacte avec une matrice génératrice ce qui permet de facto d'avoir une taille de clef publique extrêmement petite.

1.3.4 Instanciation des schémas de McEliece et Niederreiter avec des codes $\mathbb{F}_{q^m}$-linéaires en métrique rang : les LRPC

Nous nous intéressons désormais dans cette sous-section à l'extension naturelle des schémas de McEliece et Niederreiter avec des codes matriciels munis de la métrique rang. Cependant comme nous l'avons évoqué lors de l'introduction, cette dernière se fait avec une sous-classe des codes matriciels : les codes $\mathbb{F}_{q^m}$-linéaires. Or comme nous allons ici le voir, les avantages de tels codes sur les codes matriciels sont :

— une représentation bien plus compacte,

— une famille de codes munis d'un algorithme de décodage : les codes LRPC.

En revanche, comme nous le montrerons, les codes $\mathbb{F}_{q^m}$-linéaires sont des codes matriciels avec une structure particulière.

Codes matriciels $\mathbb{F}_{q^m}$-linéaires : réduction de la taille de clef. Tout $[n \times m, k \times m]_q$-code peut se représenter avec :

$$km \times (nm - km) \times \log_2(q) = k(n-k)m^2 \log_2(q)$$

bits. En revanche, un code $\mathbb{F}_{q^m}$-linéaire de longueur n et de dimension k (qui peut s'écrire sous la forme d'un $[n \times m, k \times m]_q$-code matriciel) se spécifie avec une matrice génératrice sous forme systématique sur $\mathbb{F}_{q^m}$ comme $\begin{pmatrix}\mathbf{1}_k & \mathbf{G}'\end{pmatrix}$ où $\mathbf{G}' \in \mathbb{F}_{q^m}^{k\times(n-k)}$ et donc avec :

$$k \times (n-k)\log_2(q^m) = k \times (n-k)m\log_2(q)$$

bits. Le gain en terme de représentation à considérer des codes $\mathbb{F}_{q^m}$-linéaires, plutôt que les codes matriciels, est donc de m ce qui est très avantageux quant on sait que m est dans la pratique cryptographique de l'ordre de 100. Il y a cependant quelques contre-parties à considérer une pareille construction. Les codes $\mathbb{F}_{q^m}$-linéaires forment une sous-classe structurée des codes matriciels.

Codes $\mathbb{F}_{q^m}$-linéaires : des codes matriciels particuliers. Le corps $\mathbb{F}_{q^m}$ étant une extension de $\mathbb{F}_q$, soit le polynôme de rupture unitaire ayant pour racine $x \in \mathbb{F}_{q^m}$, $R_x(X) \triangleq \sum_{\ell=0}^{m-1} a_i X^i + X^m$. En particulier, $\mathcal{B} \triangleq (1, x, \cdots, x^{m-1})$ est une base de $\mathbb{F}_{q^m}$ en tant que $\mathbb{F}_q$-espace vectoriel. Soit maintenant l'application $\mathbb{F}_q$-linéaire $\varphi : y \in \mathbb{F}_{q^m} \mapsto x \cdot y \in \mathbb{F}_{q^m}$. Sa représentation dans la base $\mathcal{B}$ est donnée par :

$$\mathsf{Comp}(R_x) \triangleq \begin{pmatrix} 0 & 0 & \cdots & \cdots & -a_0 \\ 1 & \ddots & \cdots & \cdots & -a_1 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 & -a_{m-1} \end{pmatrix} \in \mathbb{F}_q^{m\times m}$$

qui n'est autre que la matrice compagnon de R_x. Soit maintenant $\mathcal{C}$ un $[n,k]_{q^m}$-code de base $\mathbf{c}_1, \cdots, \mathbf{c}_k$. Du fait que $\mathcal{C}$ est $\mathbb{F}_{q^m}$-linéaire, son code matriciel associé dans la base $\mathcal{B}$ est engendré pour $0 \leqslant j \leqslant m-1$ et $1 \leqslant i \leqslant k$ par :

$$\mathsf{Mat}(x^j \mathbf{c}_i) = \mathsf{Comp}(R_x)^j \mathsf{Mat}(\mathbf{c}_i)$$

où l'égalité s'obtient par linéarité et composition de φ. Notons maintenant $\mathcal{A}$ le $\mathbb{F}_q$-espace vectoriel engendré par $(\mathsf{Comp}(R_x)^j)_{0\leqslant j\leqslant m-1}$ qui n'est autre qu'une algèbre de dimension m. Le code matriciel associé à $\mathcal{C}$ est donc stable par multiplication à gauche des éléments de $\mathcal{A}$, propriété qui n'a aucune raison d'être vérifiée par un code matriciel quelconque.

Les codes LRPC et leur algorithme de décodage. Venons-en maintenant à une famille de codes $\mathbb{F}_{q^m}$-linéaires utilisée en cryptographie : les codes LRPC.

Définition 1.17 (Matrice homogène)**.** *Une matrice* $\mathbf{H} = (h_{ij})_{\substack{1 \leqslant i \leqslant n-k \\ 1 \leqslant j \leqslant n}}$ *sur* $\mathbb{F}_{q^m}$ *est dite homogène de poids* d *si tous ses coefficients génèrent un* $\mathbb{F}_q$*-espace de* $\mathbb{F}_{q^m}$ *de dimension* d *:*

$$\dim\left(\mathsf{Vect}_{\mathbb{F}_q}\left(h_{ij} : 1 \leqslant i \leqslant n-k,\ 1 \leqslant j \leqslant n\right)\right) = d$$

Un code LRPC (pour *Low Rank Parity Check*) de poids d est défini à partir d'une matrice homogène de poids d comme :

Définition 1.18 (Codes LRPC)**.** *Un code* LRPC *sur* $\mathbb{F}_{q^m}$ *de poids* d *est un code admettant une matrice de parité* $\mathbf{H}$ *sur* $\mathbb{F}_{q^m}$ *homogène de poids* d*.*

Ces codes sont en quelque sorte le pendant des codes MDPC/LDPC (définition 1.14) en métrique rang : il existe une base du dual de petit poids. En revanche, en métrique de Hamming les mots de cette base sont certes de petit poids mais ils ne partagent pas le même support.

Ces codes ont l'avantage de venir avec un algorithme de décodage probabiliste pouvant décoder jusqu'à une distance de :

$$\frac{n-k}{d}.$$

Décrivons rapidement cet algorithme et pour cela rappelons la définition du produits d'espaces.

Définition 1.19. *Soient* U, V *deux sous-espaces de* $\mathbb{F}_{q^m}$*, on définit leur produit comme :*

$$U \cdot V \triangleq \mathsf{Vect}_{\mathbb{F}_q}\left(uv : u \in U, v \in V\right).$$

La proposition qui suit montre que typiquement la dimension d'un produit d'espaces est égale au produit des dimensions si celui-ci ne dépasse pas $m = \dim_{\mathbb{F}_q}(\mathbb{F}_{q^m})$.

Proposition 1.12 ([Gab+13, Proposition 1])**.** *Soit* U *un sous-espace de* $\mathbb{F}_{q^m}$ *de dimension* d*. Soient* w *éléments de* $\mathbb{F}_{q^m}$ *tirés aléatoirement formant une famille libre et dénotons par* V *l'espace vectoriel qu'ils engendrent. On a si* $wd < m$ *:*

$$\mathbb{P}\left(\dim(U \cdot V) = wd\right) \geqslant 1 - w\frac{q^{wd}}{q^m}.$$

Décodage des codes LRPC [Gab+13]. Soit $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ la matrice de parité homogène d'un $[n,k]_{q^m}$-code LRPC de poids d. Nous supposons que les paramètres sont choisis tels que :

$$wd < \min(m, n-k). \tag{1.37}$$

Notons,

$$U \triangleq \mathsf{Vect}_{\mathbb{F}_q}\left(h_{ij} : 1 \leqslant i \leqslant n-k,\ 1 \leqslant j \leqslant n\right)$$

Par définition $\dim_{\mathbb{F}_q}(U) = d$ et soit $\{f_1, \cdots, f_d\}$ l'une de ses bases. On cherche à résoudre le système d'inconnue $\mathbf{e} \in \mathbb{F}_{q^m}^n$ de poids w :

$$\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}.$$

Le point de départ de l'algorithme est de calculer le $\mathbb{F}_q$-espace :

$$W \triangleq \mathrm{Supp}(\mathbf{s})_{\mathbb{F}_q}. \tag{1.38}$$

De plus notons,

$$V \triangleq \mathrm{Supp}(\mathbf{e})_{\mathbb{F}_q} \tag{1.39}$$

qui est inconnu du décodeur et de dimension w. Il est maintenant clair que $W \subseteq U \cdot V$. Or comme énoncé par la proposition qui suit on s'attend typiquement à avoir une égalité.

Proposition 1.13 ([Gab+13, Proposition 4]). *Soit* $\mathbf{e}$ *uniformément distribuée sur les mots de* $\mathbb{F}_{q^m}^n$ *de poids* w*. Alors :*

$$\mathbb{P}_{\mathbf{e}}\left(W \neq U \cdot V\right) \leqslant q^{-(1+(n-k)-wd)}.$$

Nous pouvons donc supposer dans ce qui suit que :

$$W = U \cdot V. \tag{1.40}$$

Il est donc clair par définition de U que :

$$V \subseteq f_1^{-1}W \cap \cdots \cap f_d^{-1}W.$$

Avec le choix des paramètres comme dans (1.37), il est montré dans [Gab+13] que l'inclusion précédente est en réalité une égalité avec une bonne probabilité. De cette manière l'algorithme retrouve V et donc une de ses bases $\{v_1, \cdots, v_w\}$ en calculant l'intersection $\cap_i f_i^{-1}W$.

La dernière étape consiste à retrouver $\mathbf{e}$ avec la connaissance de son support V. On écrit :

$$\forall i \in [\![1, n]\!], \quad e_i = \sum_{j=1}^{w} x_j(i) v_j$$

où les $x_j(i)$ sont inconnues. Donc en écrivant le système $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ dans une base de $U \cdot V$, qui est typiquement de dimension wd d'après (1.37) et la proposition 1.12, nous obtenons un système linéaire avec $(n-k)wd$ équations et nw inconnues que l'on résout ce qui termine le décodage.

Instanciation avec des codes LRPC des schémas de McEliece et Niederreiter. Il serait idiot dans les schémas de McEliece et Niederreiter de renvoyer comme clef publique la matrice de parité homogène associée à un code LRPC. L'idée de la trappe consiste à rendre publique une base aléatoire du code LRPC.

Soit $\mathbf{H}_{\text{LRPC}} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ la matrice de parité homogène associée à un $[n,k]_{q^m}$-code LRPC $\mathcal{C}_{\text{LRPC}}$ de poids d. On tire alors uniformément une matrice inversible $\mathbf{S} \in \mathbb{F}_{q^m}^{(n-k)\times(n-k)}$. La clef publique dans Niederreiter et sa trappe associée T sont alors données par :

$$\mathbf{H}_{\text{pk}} \leftarrow \mathbf{S}\mathbf{H}_{\text{LRPC}} \quad \text{et} \quad T \leftarrow \mathbf{H}_{\text{LRPC}}.$$

La matrice $\mathbf{H}_{\text{pk}}$ est une matrice de parité aléatoire du code $\mathcal{C}_{\text{LRPC}}$ du fait de la multiplication par $\mathbf{S}$. Dans McEliece la clef publique est une matrice génératrice $\mathbf{G}_{\text{pk}} \in \mathbb{F}_{q^m}^{k\times n}$ aléatoire de $\mathcal{C}_{\text{LRPC}}$. Quoiqu'il en soit pour déchiffrer il faut décoder le syndrome,

$$\mathbf{s} \stackrel{\triangle}{=} \mathbf{e}\mathbf{H}_{\text{pk}}^\intercal \quad \text{où} \quad |\mathbf{e}| = w < \frac{1}{d}\min(m, n-k).$$

On commence par calculer,

$$\begin{aligned}\mathbf{s}' &\stackrel{\triangle}{=} \mathbf{s}{\mathbf{S}^{-1}}^\intercal \\ &= \mathbf{e}\mathbf{H}_{\text{LRPC}}^\intercal \mathbf{S}^\intercal {\mathbf{S}^{-1}}^\intercal \\ &= \mathbf{e}\mathbf{H}_{\text{LRPC}}^\intercal\end{aligned}$$

On décode ensuite ce syndrome avec la trappe T ce qui nous donne $\mathbf{e}$ (les paramètres sont choisis de façon à avoir unicité de la solution) de poids w.

De façon identique à la proposition 1.11 le schéma est sûr OW-CPA sous réserve de la difficulté du problème RSD et celui de distinguer un code LRPC d'un code aléatoire :

Problème 1.14. *Étant donné* $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$*,* $d \in \mathbb{N}$*, le code de matrice de parité* $\mathbf{H}$ *est-il un* $[n,k]$*-code* LRPC *de poids* d*?*

Les codes LRPC présentent un grand intérêt car à ce jour les meilleurs algorithmes pour résoudre ce problème consistent à rechercher des mots de poids faible dans le dual du code défini par la matrice de parité donnée en entrée. Or on utilise pour cela les algorithmes résolvant RSD.

1.3.5 Alekhnovich en métrique rang : RankPKE et le problème RSL

Nous avons décrit dans §1.3.2 le schéma `RankPKE` en métrique de Hamming. Ce système fut initialement proposé en métrique rang [Gab+17b]. La première différence avec ce que nous avons décrit est qu'en métrique rang il faut remplacer le corps $\mathbb{F}_2$ par $\mathbb{F}_{q^m}$ pour $m \triangleq \mu n$ avec le paramètre $\mu \in]0,1[$. De plus, il faut un $[N, k_{\text{dec}}]_{q^m}$-code de matrice génératrice $\mathbf{G}_{\text{dec}}$ que l'on sait décoder en métrique rang à distance w_{dec}. Il a alors été proposé d'utiliser les codes simples [SKK10] dans [Gab+17b] car ayant un bien meilleur pouvoir de correction que les codes LRPC.

En revanche, la différence fondamentale entre les deux schémas réside dans la partie chiffrement.

$\texttt{Enc}(\mathbf{pk}, \mathbf{m})$ **:** Pour chiffrer un message $\mathbf{m} \in \mathbb{F}_{q^m}^{k_{\text{dec}}}$ on commence par générer aléatoirement :

$$U \subseteq \mathbb{F}_{q^m} \text{sous-espace} \quad \text{et} \quad \dim_{\mathbb{F}_q}(U) = w.$$

On génère alors aléatoirement N erreurs de *même* $\mathbb{F}_q$-support F :

$$\forall i \in [\![1, N]\!], \quad \mathbf{e}_i \hookleftarrow \mathbb{F}_{q^m}^n \,:\, \mathrm{Supp}(\mathbf{e}_i)_{\mathbb{F}_q} = U.$$

et soit $\mathbf{E}$ la matrice obtenue par concaténation des N erreurs $\mathbf{e}_i$:

$$\mathbf{E} \triangleq \begin{pmatrix} \mathbf{e}_1^\intercal & \cdots & \mathbf{e}_N^\intercal \end{pmatrix} \in \mathbb{F}_{q^m}^{n\times N}.$$

Le chiffré $\begin{pmatrix} \mathbf{C} \\ \mathbf{y} \end{pmatrix} \in \mathbb{F}_{q^m}^{(n-k+1)\times N}$ se calcule comme précédemment :

$$\begin{pmatrix} \mathbf{C} \\ \mathbf{y} \end{pmatrix} \triangleq \begin{pmatrix} \mathbf{HE} \\ (\mathbf{xH} + \mathbf{e})\,\mathbf{E} + \mathbf{mG}_{\text{dec}} \end{pmatrix}$$

où $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ et $\mathbf{xH} + \mathbf{e}$ forment la clef publique avec $|\mathbf{e}| = w$.

Le fait que la matrice $\mathbf{E}$ soit la concaténation de N erreurs $\mathbf{e}_i$ de même $\mathbb{F}_q$-support est alors crucial dans le déchiffrement et plus particulièrement pour le décodage. En effet après avoir utilisée la clef secrète $\mathbf{x}$ sur un chiffré on obtient :

$$-\left(\mathbf{mG}_{\text{dec}} + \left(\langle \mathbf{e}, \mathbf{e}_i \rangle_{1\leqslant i\leqslant N}\right)\right).$$

Or si on note,

$$V \triangleq \mathrm{Supp}(\mathbf{e})$$

où $\dim_{\mathbb{F}_q}(V) = \dim_{\mathbb{F}_q}(U) = w$. Nous avons du fait que les $\mathbf{e}_i$ ont tous le même $\mathbb{F}_q$-support U par définition du produit d'espaces (définition 1.19) :

$$\mathrm{Supp}\left(\langle \mathbf{e}, \mathbf{e}_i \rangle_{1\leqslant i\leqslant N}\right)_{\mathbb{F}_q} \subseteq U \cdot V.$$

Le poids de l'erreur à décoder est donc sous l'hypothèse

$$w^2 \leqslant m \tag{1.41}$$

inférieur ou égal à w^2 d'après la proposition 1.12 avec probabilité $1 - w\frac{q^{w^2}}{q^m}$. Le paramètre w_{dec} de RankPKE est donc choisi comme :

$$w_{\text{dec}} = w^2 \tag{1.42}$$

avec w vérifiant (1.41).

Notons que sans la condition sur le support des $\mathbf{e}_i$ il n'y aurait pas eu ce contrôle sur le poids de l'erreur à décoder lors du déchiffrement.

La réduction de sécurité. La sécurité de `RankPKE` repose entre autres sur la difficulté du problème RSL (et non l'équivalent de DOOM en métrique rang) que nous rappelons [Gab+17b].

Problème 1.15 (Problème RSL - RSL$(n, q, R, \omega, \mu, N)$)**.** *Soient les entiers* $m \triangleq \mu n$, $k \triangleq \lfloor Rn \rfloor$ *et* $w \triangleq \lfloor \omega n \rfloor$.

- Instance : *une matrice de parité* $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ *de rang* $n-k$, N *syndromes* $\mathbf{e}_1\mathbf{H}^\intercal, \cdots, \mathbf{e}_N\mathbf{H}^\intercal \in \mathbb{F}_{q^m}^{n-k}$ *où les* $\mathbf{e}_i$*'s sont tous de même* $\mathbb{F}_q$*-support* U *de dimension* w.
- Recherche : U.

La version décisionnelle de ce problème se définit alors comme :

Problème 1.16 (Problème décisionnel RSL - DRSL$(n, q, R, \omega, \mu, N)$)**.** *Soient les entiers* $m \triangleq \mu n$, $k \triangleq \lfloor Rn \rfloor$ *et* $w \triangleq \lfloor \omega n \rfloor$.

- *Distributions :*
  - $\mathcal{D}_0$ *:* $(\mathbf{H}, \mathbf{s}_1^{\text{unif}}, \cdots, \mathbf{s}_N^{\text{unif}}) \hookleftarrow \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{n-k} \times \cdots \times \mathbb{F}_{q^m}^{n-k}$,
  - $\mathcal{D}_1$ *:* $(\mathbf{H}, \mathbf{e}_1\mathbf{H}^\intercal, \cdots, \mathbf{e}_N\mathbf{H}^\intercal)$ *où* $\mathbf{H} \hookleftarrow \mathbb{F}_{q^m}^{(n-k)\times n}$, $U \subseteq \mathbb{F}_{q^m}$ *aléatoire de dimension* w *et les* $\mathbf{e}_i$ *uniformément distribuées sur les mots de* $\mathbb{F}_{q^m}^n$ *de support* U.
- Entrée : $(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N)$ *pour* $(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N) \hookleftarrow \mathcal{D}_b$ *avec* $b \hookleftarrow \{0, 1\}$
- Décision : $b' \in \{0, 1\}$.

On définit alors le succès de résoudre ce problème :

Définition 1.20. *Le succès* DRSL *d'un algorithme* $\mathcal{A}$ *est défini comme :*

$$Adv^{\text{DRSL}(n,q,R,\omega,\mu,N)}(\mathcal{A}) \triangleq \frac{1}{2}\Big(\mathbb{P}\big(\mathcal{A}(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N) = 1 \mid b = 1\big) - \mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N) = 1 \mid b = 0\right)\Big) \tag{1.43}$$

où les probabilités sont calculées sur l'aléa interne de $\mathcal{A}$, b *et l'entrée tirée selon* $\mathcal{D}_b$ *définie dans le problème 1.16 de paramètres* $(n, q, R, \omega, \mu, N)$. *Nous définissons alors le succès DRSL calculatoire en temps* t *:*

$$Succ^{\text{DRSL}(n,q,R,\omega,\mu,N)}(t) \triangleq \max_{\mathcal{A}:|\mathcal{A}|\leqslant t} \left(Adv^{\text{DRSL}(n,q,R,\omega,\mu,N)}(\mathcal{A})\right).$$

Nous discuterons de la difficulté du problème RSL dans la partie IV (voir [Gab+17b, §4] et [DT18c ; Ara+18]) mais pour résumer rapidement cette dernière dépend fortement du nombre N de syndromes donnés en entrée. Les meilleurs algorithmes pour le résoudre sont :

- exponentiels si N est suffisamment proche de 1,
- sous-exponentiels si $N \geqslant wk$,

— polynomiaux si $N \geqslant nw$

Il est maintenant donné dans [Gab+17b] une réduction de sécurité IND-CPA de RankPKE aux problèmes RSL et RSD.

Théorème 1.3 ([Gab+17b]). *Soit $\mathcal{E}$ le schéma de chiffrement* RankPKE *en métrique rang. On a :*

$$Succ_{\mathcal{E}}^{\text{IND-CPA}}(t) \leqslant Succ^{\text{DRSD}(n,2,1-R,\omega,\mu)}(t) + Succ^{\text{DRSL}(n,2,R,\omega,\mu,N)}(t)$$

pour les paramètres du système $R \in]0,1[$, $N \in \mathbb{N}$ et $\omega = O(\mu/\sqrt{n})$.

Terminons cette sous-section par mentionner le chiffrement en métrique rang RQC [Agu+19]. Ce-dernier suit la même approche que HQC (voir §1.3.3) vis à vis de RankPKE en métrique de Hamming mais en métrique rang. L'idée est cette fois-ci d'utiliser comme secret des polynômes dont tous les coefficients appartiennent à un même espace vectoriel de petite dimension. La sécurité se réduit alors uniquement à la difficulté de la version décisionnelle du décodage en métrique rang.

1.4 Codes et signatures

Contrairement aux chiffrements, il y eut peu de propositions pour des schémas de signature dont la sécurité repose sur le décodage générique. La conception de ces primitives avec la théorie des codes est un problème ouvert depuis maintenant de nombreuses années.

Nous commencerons dans cette section par présenter les schémas de signature de type *hache et signe* CFS [CFS01] ainsi que RankSign [Gab+14b]. Nous présenterons de plus le premier *Identity-Based-Encryption* (IBE) [Sha84] utilisant des codes [Gab+17b] avec comme briques élémentaires RankSign et RankPKE (voir §1.3.5). Enfin, nous terminerons cette section par la présentation du schéma de signature KKS [KKS97] ainsi que le protocole d'identification de Stern [Ste93b] à divulgation nulle de connaissance et sa conversion en schéma de signature à l'aide de la transformation de Fiat-Shamir [FS87].

Commençons cependant par donner la définition formelle de signature à clef publique ainsi qu'une discussion informelle sur le modèle de sécurité EUF-CMA que nous considèrerons dans la suite.

Généralités sur les signatures numériques.

Définition 1.21. *Une signature à clef publique est donnée par un triplet d'algorithmes* $(\texttt{KeyGen}, \texttt{Sgn}, \texttt{Vrfy})$ *et un couple de fonctions* $(m(\lambda), n(\lambda))$ *polynomiales en le paramètre de sécurité λ tels que :*

— KeyGen*, la* génération de clef*, est un algorithme probabiliste polynomial d'entrée 1^λ et renvoyant une paire de clefs publique et secrète* $(\mathrm{pk}, \mathrm{sk})$*,*

— Sgn*, la* signature*, est un algorithme probabiliste polynomial prenant en entrée* sk*, un message $\mathbf{m} \in \{0,1\}^{m(\lambda)}$ et renvoyant une signature $\mathbf{y} \in \{0,1\}^{n(\lambda)}$,*

— Vrfy*, est un algorithme à valeurs dans $\{0,1\}$ et prenant comme entrée* pk*, $\mathbf{m} \in \{0,1\}^{m(\lambda)}$ et un élément de $\{0,1\}^{n(\lambda)}$ tels que :*

$$\texttt{Vrfy}^{\mathrm{pk}}(\mathbf{m}, \texttt{Sgn}(\mathbf{m})) = 1.$$

Il y a alors trois façons classiques de concevoir des signatures : (i) utiliser le paradigme de type hache et signe à l'aide d'une trappe, (ii) transformer un schéma d'authentification à divulgation nulle de connaissance en signature à l'aide de la transformation de Fiat-Shamir [FS87] ou encore (iii) à partir de primitives symétriques. Dans ce document nous nous intéresserons tout particulièrement à la première solution. Décrivons-la rapidement.

Signatures de type *hache et signe*. L'idée est de tout d'abord se donner une fonction à trappe à sens unique $f : \mathcal{A} \to \mathcal{D}$, c'est à dire :

— pour toute entrée $x \in \mathcal{A}$, $f(x)$ se calcule facilement,

— la probabilité de tout algorithme ne connaissant pas la trappe de trouver un élément de $f^{-1}(f(x))$ est négligeable,

— on calcule facilement un élément de $f^{-1}(f(x))$ avec la trappe.

Considérons maintenant une fonction de hachage cryptographique $\mathcal{H}$. La signature d'un message $\mathbf{m}$ et sa vérification consistent tout simplement en :

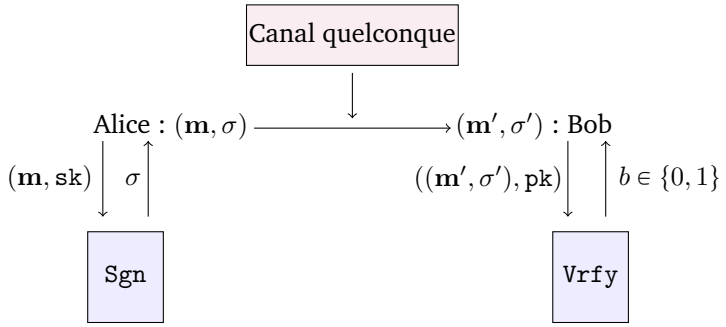
— Signature : $\sigma \in f^{-1}(\mathcal{H}(\mathbf{m}))$ calculée grâce à la trappe,

— Vérification : $f(\sigma) = \mathcal{H}(\mathbf{m})$.

Les réductions de sécurité de ces types de signature se font souvent dans le cas où la fonction de hachage $\mathcal{H}$ est supposée aléatoire [BR93]. On parle alors de modèle de l'oracle aléatoire (ROM).

Une remarque fondamentale est que pour espérer obtenir de telles signatures il nous faut être en mesure de pouvoir inverser f en presque toute entrée

Les signatures et leur modèle de sécurité : un bref aperçu. Nous résumons la définition de signature dans la figure qui suit où Alice détentrice d'une clef secrète envoie des signatures à Bob disposant quant à lui de la clef publique associée. L'objectif d'Alice est d'assurer à Bob que les messages viennent bien d'elle et que ces derniers n'ont pas été altérés lors de leur transmission.



L'objectif de toute signature cryptographique est qu'il soit impossible pour un adversaire de signer un message (on pourra aussi dire *forger une signature*) à la place du détenteur de la clef secrète. D'un point de vue sécurité il y a alors deux possibilités. Nous pouvons demander à ce que tout adversaire ne puisse pas signer :

— le message de son choix : contrefaçon *existentielle*,

— tout message : contrefaçon *universelle*.

Il est alors clair que la notion la plus forte est celle assurant l'impossibilité des contrefaçons existentielles. Nous considèrerons donc dans la suite le modèle existentiel.

En revanche, quel pouvoir donner à un attaquant faisant face à un schéma de signature ? Ce dernier dispose évidemment de la clef publique. La difficulté réside dans le fait suivant. Quand Alice qui dispose de la clef secrète souhaite donner une signature à Bob, cette dernière l'envoie à travers un canal non sur. Nous pouvons donc supposer que tout couple message-signature peut être à la disposition de tous. Or ces derniers sont produits directement à partir de la clef secrète. Il se peut donc qu'il y ait *des fuites d'informations* sur la clef. Autrement dit, un adversaire peut collectionner des signatures pour finalement en déduire la clef secrète. Il est donc naturel dans le cas des signatures de supposer que

l'adversaire dispose de couples message-signatures. Dans la suite nous nous placerons dans le modèle le plus fort où tout adversaire a en main des signatures de messages *de son choix* et son objectif est de signer un message pour lequel il ne dispose pas déjà d'une signature.

Nous définirons formellement ce modèle de sécurité, dit *Existential UnforgEability under Chosen Message Attacks* (EUF-CMA), dans la partie III, chapitre 2 de ce document lors de l'étude de la signature Wave [DST19b].

Notons que pour les signatures la situation est inversée par rapport aux schémas de chiffrement. En effet, n'importe qui peut produire des chiffrés à l'aide de la clef publique. L'opération de déchiffrement utilisant la clef secrète est alors transparente à moins d'être dans un modèle d'attaques fort par canaux cachés ou dans le cas où les adversaires ont accès à un oracle de déchiffrement. Il est donc plus difficile de concevoir des signatures car il faut pour ces dernières que la clef publique *et* le fait de signer ne révèlent rien sur la clef secrète alors que pour des chiffrements il est a priori seulement demandé que la clef publique ne donne par d'information sur la clef secrète.

1.4.1 L'approche CFS

Suite aux propositions de chiffrement de McEliece [McE78] et Niederreiter [Nie85] il fallut attendre la proposition de [CFS01] pour obtenir la première signature de type hache et signe dont la sécurité repose en partie sur le décodage générique. Cependant, comme nous allons le voir ici ce schéma (i) souffre d'un problème de mise à l'échelle de ses paramètres car sa sécurité est polynomiale en la taille de clef publique et (ii) d'un problème de structure les matrices de parité publiques étant distinguables de matrices aléatoires.

Les auteurs de [CFS01] proposèrent de choisir comme fonction à sens unique celle définie pour les paramètres entiers n, k, w et une matrice de parité $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ par :

$$\begin{aligned} f_{w,\mathbf{H}} : \quad \left\{\mathbf{x} \in \mathbb{F}_q^n : |\mathbf{x}| = w\right\} &\longrightarrow \mathbb{F}_q^{n-k} \\ \mathbf{e} &\longmapsto \mathbf{e}\mathbf{H}^\intercal \end{aligned}$$

Il est alors clair que cette fonction peut être supposée à sens unique car l'inverser revient à résoudre le problème du décodage à distance w. Choisir une trappe consiste ici à prendre $\mathbf{H}$ (qui est la clef publique) comme une matrice de parité d'un code pour lequel on connaît un algorithme de décodage. Il fut alors proposé dans [CFS01] de choisir comme dans l'instanciation de McEliece [McE78] des codes de Goppa binaire obtenus à partir de polynômes sans facteurs multiples (voir §1.3.1). La longueur n du code de Goppa est choisie telle que

$$n \triangleq 2^m \iff m = \log_2(n) \tag{1.44}$$

et sa dimension k,

$$k \triangleq n - wm \iff w = \frac{n-k}{m} \tag{1.45}$$

où $m \in \mathbb{N}$ et $w \in [\![0, n]\!]$ est la distance de décodage.

Dans le contexte d'une signature de type hache et signe, signer un message $\mathbf{m}$ revient à inverser $\mathcal{H}(\mathbf{m}) \in \mathbb{F}_2^{n-k}$, *i.e* : trouver un élément de $f_{w,\mathbf{H}}^{-1}(\mathcal{H}(\mathbf{m}))$ où $\mathcal{H}$ est une fonction de hachage cryptographique que l'on suppose aléatoire. On cherche donc à décoder un syndrome $\mathbf{s}$ aléatoire. Or dans le cas où k/n est fixé les codes de Goppa et leur algorithme de décodage ne permettent de décoder qu'une proportion exponentiellement faible de syndromes. En effet, la distance de décodage w est bien inférieure à la borne de Gilbert-Varshamov $w_{\text{GV}}(n, k)$ étant donné que $w = o(n)$. En revanche, comme remarqué dans [CFS01] w est proche de $w_{\text{GV}}(n, k)$ lorsque asymptotiquement avec n, le rendement du code est telle que,

$$\frac{k}{n} = 1 - o(1) \tag{1.46}$$

La proportion des syndromes que l'on peut décoder, dits "décodables", est alors donnée par :

$$\frac{\binom{n}{w}}{2^{n-k}} = \frac{\binom{2^m}{w}}{2^{wm}} \approx \frac{1}{w!} \tag{1.47}$$

Il fut proposé dans [CFS01] de chercher à inverser $\mathcal{H}(\mathbf{m}, i)$ où i est un compteur que l'on incrémente tant que le décodage échoue. La signature de $\mathbf{m}$ correspond alors au résultat du décodage $\mathbf{e}$ et le compteur i. La vérification consiste à s'assurer que $\mathbf{e}\mathbf{H}^\intercal = \mathcal{H}(\mathbf{m}, i)$. Les travaux de [Dal10] proposèrent de remplacer le compteur par un sel $\mathbf{r}$ tiré aléatoirement à chaque essai. Ceci permit de donner une réduction de sécurité [Dal10] dans le modèle EUF-CMA aux mêmes hypothèses calculatoires que le chiffrement originel de McEliece.

En revanche, il nous faut ici tenir compte du nombre d'essais réalisés avant de réussir le décodage. Il est clair d'après (1.47) qu'il faut faire en moyenne environ

$$w!$$

tentatives avant de réussir. Il est donc essentiel pour que l'algorithme ne fasse pas trop d'essais que w soit suffisamment petit. En pratique

$$w \leqslant 12 \tag{1.48}$$

Les ennuis commencent alors pour CFS. En effet, une clef publique est donnée par une matrice de parité aléatoire mise sous forme systématique du code de Goppa considéré. Cette dernière est donc formée $K \triangleq k\,(n-k)$ bits. De cette façon, d'après (1.44), (1.45) et (1.46) nous avons :

$$K \approx mw2^m.$$

Au même moment, l'algorithme de Dumer que nous décrirons dans §2.2.1 résout dans le régime de rendement (1.46) le problème du décodage générique (c'est à dire ici sans connaître la trappe) en temps $T \approx 2^{(n-k)/2}$ (voir la proposition 2.1). Donc d'après (1.45), une attaque contre CFS par décodage coûte $T \approx 2^{wm/2}$. Or w est petit (1.48) et majoré par une constante, ce qui donne,

$$T \approx K^{w/2}.$$

Autrement dit, la sécurité du schéma est une fonction polynomiale de la taille de clef. Les auteurs de [CFS01] proposèrent tout de même des paramètres pour 80 bits de sécurité dans l'état de l'art ainsi qu'une implémentation. Le temps pour générer une signature est de l'ordre de quelques secondes tandis que le temps vérification est inférieur à une seconde. La taille de clef publique est quant à elle de l'ordre d'un Mbyte. Il est à noter que les tailles de signature sont extrêmement petites, environ une centaine de bits. Cela provient du fait que w, le poids de l'erreur décodée est petit. Cependant une attaque non publiée de Bleichenbacher vint mettre à mal ces paramètres. L'idée de cette attaque est de ne pas chercher à inverser $f_{w,\mathbf{H}}$ pour un syndrome obtenu comme $\mathcal{H}(\mathbf{m})$ mais pour de nombreux syndromes $\mathcal{H}(\mathbf{m}_1), \cdots, \mathcal{H}(\mathbf{m}_N)$ hachés de messages du choix de l'adversaire. Autrement dit Bleichenbacher attaqua CFS en considérant le problème DOOM plutôt que SD ce qui est naturel dans un contexte de signature. En effet, un adversaire peut choisir n'importe quel message à signer et le hacher. Or comme nous le verrons dans le prochain chapitre il existe des algorithmes de décodage générique exponentiellement meilleurs pour résoudre DOOM que SD typiquement lorsque w est de l'ordre de la borne de Gilbert-Varshamov, ce qui est le cas ici. Il fut cependant proposé avec le schéma parallèle-CFS [Fin10] une méthode pour éviter cette attaque. L'idée étant pour signer un message $\mathbf{m}$ de la hacher i fois (typiquement $i = 2$) avec i différentes fonctions de hachage $\mathcal{H}_1, \cdots, \mathcal{H}_i$ et de renvoyer comme signatures les i inverses selon $f_{w,\mathbf{H}}$ de $\mathcal{H}_1(\mathbf{m}), \cdots, \mathcal{H}_i(\mathbf{m})$.

Quoiqu'il en soit, si nous avions à choisir pour CFS ou parallèle-CFS des paramètres donnant 128 bits de sécurité nous aurions des tailles de clefs publiques de l'ordre de

plusieurs gigabytes et des temps de signatures de plusieurs dizaines de secondes. Le problème s'aggravent encore plus en montant à des terabytes et heures si nous cherchons à donner des paramètres pour 128 bits de sécurité quantique. Cela fait donc de CFS un schéma impraticable.

De plus, le schéma CFS connaît aujourd'hui un autre problème venu invalider sa preuve de sécurité. Rappelons que cette dernière se réduit aux problèmes (i) du décodage générique et (ii) de distinguer un code de Goppa utilisé comme clef publique d'un code aléatoire. Concernant le problème (ii), les travaux de [Fau+13] montrèrent qu'il faut être précautionneux avec le choix des paramètres pour une utilisation cryptographique. En effet, il est donné dans [Fau+13] un distingueur d'un code de Goppa de rendement $k/n \approx 1$ d'un code aléatoire de mêmes paramètres. Or dans CFS le code de Goppa utilisé est *nécessairement* dans ce régime de paramètres. Il est cependant à noter que ce dernier ne permet pas à l'heure actuelle de retrouver la structure du code de Goppa sous-jacente.

1.4.2 Une signature en métrique rang : RankSign

Il s'avère que les codes LRPC (voir §1.3.4) peuvent être utilisés pour une signature en métrique rang mais avec une stratégie différente de celle de CFS [CFS01] comme il le fut remarqué dans [Gab+14b]. Les auteurs de [Gab+14b] ont montré qu'en choisissant bien les paramètres d'un code LRPC de matrice de parité $\mathbf{H}_{\text{LRPC}}$ il est possible de résoudre efficacement :

$$\text{pour presque tout } \mathbf{s} \in \mathbb{F}_{q^m}^{n-k} \text{ et } T \text{ sous-espace de } \mathbb{F}_{q^m} \text{ de dimension } t \text{ trouver}$$

$$\mathbf{e} \text{ tel que } \mathbf{e}\mathbf{H}_{\text{LRPC}}^{\intercal} = \mathbf{s} \text{ avec } |\mathbf{e}| = w \text{ et } T \subseteq \text{Supp}(\mathbf{e})_{\mathbb{F}_q}.$$

Autrement dit, l'algorithme de [Gab+14b] permet de résoudre avec des codes LRPC bien choisis le problème du décodage pour des syndromes aléatoires avec une erreur dont on fixe arbitrairement une partie du support. Cette liberté implique clairement que pour un syndrome donné il n'y a plus unicité de la solution ici de rang w. La distance de décodage w est donc, contrairement à CFS, nécessairement supérieure à la borne de Gilbert-Varshamov w_{rGV} (définition 1.6).

Nous allons dans ce qui suit rappeler cet algorithme de décodage et montrer comment ce dernier permet d'instancier `RankSign`.

Commençons par fixer dans tout ce qui suit pour les paramètres $n, m, k, w, d, t \in \mathbb{N}$, et une matrice de parité $\mathbf{H}_{\text{LRPC}} = (h_{i,j}) \in \mathbb{F}_{q^m}^{(n-k)\times n}$ d'un code LRPC telle que

$$F \triangleq \text{Vect}_{\mathbb{F}_q}\left(h_{i,j} : 1 \leqslant i \leqslant n-k, 1 \leqslant j \leqslant n\right).$$

L'espace F est de dimension d et $f_1, \cdots, f_d$ désignera l'une des ses bases. De plus, nous noterons $T \subseteq \mathbb{F}_{q^m}$ un sous-espace de $\mathbb{F}_{q^m}$ de dimension t. La signature `RankSign` a été étudié par les auteurs de [Gab+14b] dans le cas où les paramètres vérifient :

$$w - t \leqslant \frac{n-k}{d}. \tag{1.49}$$

L'algorithme de décodage. L'algorithme de décodage présenté dans [Gab+14b] fonctionne pour une certaine classe de syndromes.

Définition 1.22 ([Gab+14b, Définition 5])**.** *Soient F, T deux sous-espaces de $\mathbb{F}_{q^m}$ et $f_1, f_2 \in F$ formant une famille libre. Un syndrome $\mathbf{s} \in \mathbb{F}_{q^m}^{n-k}$ est dit (w, T)-décodable s'il existe un espace $E \subseteq \mathbb{F}_{q^m}$ de dimension w contenant T tel que :*

1. $\dim_{\mathbb{F}_q}(F \cdot E) = \dim_{\mathbb{F}_q}(F)\dim_{\mathbb{F}_q}(E)$,

2. $\dim_{\mathbb{F}_q}\left(f_1^{-1}F\cdot E\cap f_2^{-1}F\cdot E\right)=\dim_{\mathbb{F}_q}(E)$,
3. $s_1,\cdots,s_{n-k}\in F\cdot E$ *et*

$$\text{Vect}_{\mathbb{F}_q}\left(s_1,\cdots,s_{n-k},uv : u\in F \text{ et } v\in T\right)=F\cdot E.$$

Soit $\mathbf{s}\in\mathbb{F}_{q^m}$ un syndrome (w,T)-décodable d'espace E associé. Il est alors possible de retrouver $\mathbf{e}\in\mathbb{F}_{q^m}^n$ tel que $\mathbf{eH}_{\text{LRPC}}^{\intercal}=\mathbf{s}$ et $\text{Supp}(\mathbf{e})_{\mathbb{F}_q}\subseteq E$ en procédant de la façon suivante :

1. D'après la condition 3 : on calcule avec $\mathbf{s}$ et les espaces F et T l'espace $F\cdot E$,
2. D'après les conditions 1 et 2 : on en déduit E en calculant l'espace $f_1^{-1}F\cdot E\cap f_2^{-1}F\cdot E=E$
3. On résout dans $\mathbb{F}_q$ le système $\mathbf{eH}_{\text{LRPC}}^{\intercal}=\mathbf{s}$ où $e_i=\sum_{j=1}^{w}x_j^{(i)}\alpha_j$ et $\alpha_1,\cdots,\alpha_w$ est une base de E.

Par définition de la (w,T)-décodabilité seule la troisième étape de cet algorithme peut échouer, le système n'ayant pas toujours de solution de support E. En revanche, les coordonnées de la matrice $\mathbf{H}_{\text{LRPC}}$ étant dans F et celles de $\mathbf{s}$ dans $F\cdot E$, il suffit que l'application suivante :

$$\begin{array}{rcl} E^n & \longrightarrow & (F\cdot E)^{n-k} \\ \mathbf{e} & \longmapsto & \mathbf{eH}_{\text{LRPC}}^{\intercal}\end{array}$$

soit surjective. Il est donc nécessaire pour que cela soit le cas avec une suffisamment bonne probabilité que les paramètres n, k, $w=\dim_{\mathbb{F}_q}(E)$ et $d=\dim_{\mathbb{F}_q}(F)$ soient tels que :

$$\begin{aligned}\dim E^n=\dim(F\cdot E)^{n-k} &\iff n\dim E=(n-k)\dim E\cdot F\\ &\iff nw=(n-k)wd \quad \text{(point 1 de la définition 1.22)}\end{aligned}$$

D'où l'on en déduit la contrainte sur les paramètres :

$$n=(n-k)d.$$

Cette condition est alors essentielle pour que le décodage puisse fonctionner. Malheureusement cette contrainte implique une faiblesse fatale de `RankSign` que nous avons exploité [DT18c] pour monter une attaque comme nous le verrons dans la partie IV de ce document.

En résumé, nous avons la proposition suivante :

Proposition 1.14. *Soit $\mathbf{s}\in\mathbb{F}_{q^m}^{n-k}$ un syndrome (w,T)-décodable. Alors sous la condition*

$$n=(n-k)d \tag{1.50}$$

nous pouvons résoudre :

$$\mathbf{eH}_{\text{LRPC}}^{\intercal}=\mathbf{s},\quad |\mathbf{e}|\leqslant w \quad \textit{et} \quad T\subseteq\text{Supp}(\mathbf{e})_{\mathbb{F}_q}$$

avec une probabilité proche de 1.

Le décodeur de cette proposition peut vraiment être vu comme *un décodeur par effacement*. En métrique de Hamming l'objectif des algorithmes de ce type est de pouvoir retrouver un mot de code $\mathbf{c}$ à partir $\mathbf{c}+\mathbf{e}$ mais où *certaines coordonnées ont été effacées*. On peut alors voir les effacements comme une partie arbitraire du support de l'erreur recherchée. L'équivalent de ce problème en métrique rang ne consiste pas à effacer des positions, la notion de support étant différente. En effet, retrouver une erreur $\mathbf{e}\in\mathbb{F}_{q^m}^n$ revient essentiellement à chercher un espace vectoriel. Un effacement correspond donc à fixer aléatoirement une partie T du support $\text{Supp}(\mathbf{e})_{\mathbb{F}_q}$. Le décodeur que nous venons de présenter permet donc bien de retrouver avec des codes LRPC des erreurs de ce type.

Désormais, étant donné que nous sommes dans un contexte de signature, il est nécessaire d'être en mesure de décoder presque tous les syndromes. L'algorithme que nous venons de présenter rapidement le fait pour des syndromes (w, T)-décodables. Il est alors démontré dans [Gab+14b] (sous quelques hypothèses supplémentaires) que ces derniers sont de densité dans $\mathbb{F}_{q^m}^{n-k}$:

$$O\left(q^{(w-t)(m-w)+(n-k)(wd-m)}\right).$$

Les auteurs de [Gab+14b] proposent donc de choisir les paramètres tels que cette proportion soit un $O(1)$, *i.e* :

$$(w-t)(m-w)+(n-k)(wd-m)=0.$$

Avec les différentes contraintes, une solution de cette équation est donnée par :

$$m = w(d+1) \quad \text{et} \quad w = t + \frac{n-k}{d}. \tag{1.51}$$

La signature `RankSign`**.** Afin de tirer profit du décodage des syndromes (w, T)-décodables il est introduit dans [Gab+14b] les codes LRPC-augmentés.

Définition 1.23. *Un code* LRPC-augmenté *de type* (d, t) *sur* $\mathbb{F}_{q^m}$ *est un code admettant une matrice de parité* $\begin{pmatrix}\mathbf{H}_{\text{LRPC}} & \mathbf{R}\end{pmatrix}\mathbf{P}$ *où* $\mathbf{H}_{\text{LRPC}}$ *est homogène de poids* d*,* $\mathbf{R}$ *est une matrice avec* t *colonnes quelconques et* $\mathbf{P}$ *est une matrice à coefficients dans* $\mathbb{F}_q$ *et inversible.*

Ces codes ont été introduits pour deux raisons. La première est que la relation sur les paramètres $n = (n-k)d$ du décodeur de la proposition 1.14 est très contraignante d'un point de vue cryptographique. En effet, dans le cas du rendement classiquement considéré $k/n = 1/2$, nous avons $d = 2$. Il existe donc des mots de poids 2 dans le dual du code LRPC considéré. Ces derniers sont alors facilement accessibles. L'ajout de la matrice $\mathbf{R}$ permet donc d'augmenter typiquement de t le poids minimal des mots dans le code. De plus, cette matrice ne change pas le "pouvoir du décodeur". En effet, nous allons voir que c'est la combinaison linéaire de ses colonnes qui donnera l'espace T du décodeur par effacement.

La signature `RankSign` est spécifiée de la façon suivante :

Les paramètres : n, k, w, d, t et m sont choisis tels que les équations (1.50) et (1.51) sont vérifiées. De plus, on se donne une fonction de hachage $\mathcal{H} : \{0,1\}^* \to \mathbb{F}_{q^m}^{n-k}$ cryptographique.

Clef secrète : une matrice de parité de $\mathbf{H}_{\text{LRPC}} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ homogène de poids d, une matrice $\mathbf{R} \in \mathbb{F}_{q^m}^{(n-k)\times t}$ aléatoire, $\mathbf{P} \in \mathbb{F}_q^{(n+t)\times(n+t)}$ et $\mathbf{S} \in \mathbb{F}_{q^m}^{(n-k)\times(n-k)}$ inversibles.

Clef publique : La distance de signature w, la matrice $\mathbf{H}_{\text{pk}} \triangleq \mathbf{S}\begin{pmatrix}\mathbf{H}_{\text{LRPC}} & \mathbf{R}\end{pmatrix}\mathbf{P}$ qui est une matrice de parité aléatoire d'un LRPC augmenté de type (d, t).

Signature d'un message $\mathbf{m}$: on procède de la façon suivante :

1. $\mathbf{x} \hookleftarrow \mathbb{F}_{q^m}^t$ et $\mathbf{r} \hookleftarrow \{0,1\}^\ell$,
2. On utilise l'algorithme décrit précédemment pour décoder avec la matrice $\mathbf{H}_{\text{LRPC}}$ le syndrome
$$\mathbf{s}' \triangleq \mathbf{s}\mathbf{S}^{-1^\intercal} - \mathbf{x}\mathbf{R}^\intercal \quad \text{où} \quad \mathbf{s} \triangleq \mathcal{H}(\mathbf{m}, \mathbf{r})$$
avec pour espace $T \triangleq \text{Supp}(\mathbf{x})_{\mathbb{F}_q}$. Si l'algorithme échoue on retourne à l'étape 1. Sinon, l'algorithme renvoie $\mathbf{e}$ de poids $\leqslant w$ et telle que $T \subseteq \text{Supp}(\mathbf{e})$. On retourne alors comme signature
$$(\mathbf{e}, \mathbf{x})\mathbf{P}^{-1^\intercal}, \mathbf{r}$$

De cette façon, $\mathbf{P}$ étant une une matrice inversible sur $\mathbb{F}_q$ donc une isométrie pour la métrique rang, nous avons :

$$|(\mathbf{e}, \mathbf{x})\mathbf{P}^{-1^\intercal}| = |(\mathbf{e}, \mathbf{x})|.$$

Mais maintenant du fait que $\mathrm{Supp}(\mathbf{x}) = T \subseteq \mathrm{Supp}(\mathbf{e})$, nous obtenons :

$$|(\mathbf{e}, \mathbf{x})| = |\mathbf{e}|$$

qui est inférieur ou égal à w par définition de l'algorithme de décodage des syndromes (w, T)-décodables.

Vérification de $(\mathbf{m}, \mathbf{e}', \mathbf{r})$: on s'assure que $\mathbf{e}'\mathbf{H}_{\text{pk}}^\intercal = \mathcal{H}(\mathbf{m}, \mathbf{r})$ et $|\mathbf{e}'| \leqslant w$.

Les contraintes sur les paramètres sont alors essentielles pour que l'algorithme de signature puisse fonctionner :

— (1.50) permet d'assurer qu'un syndrome (w, T)-décodable se décode,

— avec (1.51) la densité des syndromes (w, T)-décodable est en $O(1)$.

Quoiqu'il en soit, `RankSign` a été la première signature reposant sur des problèmes de décodage en métrique rang. De plus, les auteurs [Gab+17b] ont démontré que les signatures produites ne faisaient pas fuiter d'information sur la clef dans le sens suivant.

Théorème 1.4 ([Gab+17b])**.** *Pour tout algorithme $\mathcal{A}$ forgeant une signature avec la clef publique et $N \leqslant q/2$ signatures authentiques, il existe un algorithme $\mathcal{A}'$ de même complexité forgeant une signature avec uniquement la clef publique.*

1.4.3 Un IBE utilisant des codes

Les IBE en cryptographie. Un *Identity-Based-Encryption* (IBE) est une primitive cryptographique fournissant une alternative crédible aux chiffrements à clef publique introduit en 1984 par Shamir [Sha84]. Dans un IBE, la clef publique associée à un utilisateur est une chaîne arbitraire reliée à son identité comme par exemple son adresse e-mail. De cette façon, tout utilisateur peut chiffrer des messages en dérivant la clef publique de l'identité du destinataire. La principale différence entre un IBE et une infrastructure de chiffrement à clef publique (PKE) concerne donc la gestion des clefs, en particulier la façon dont elles sont générées et vérifiées. Dans un PKE la vérification de clefs se fait à l'aide de certificats ce qui n'est pas le cas dans un IBE. En revanche, dans ce type d'infrastructure il y a une partie tiers de confiance, dite "Centre de dérivation de clef". Cette dernière possède alors une clef secrète maître notée MSK et une clef publique associée MPK. La clef secrète MSK permet alors de calculer à partir de n'importe quelle identité id une clef secrète sk_{id}. On en déduit alors une paire de clef public/privée :

$$(id, \text{MPK}), \mathrm{sk}_{id}.$$

permettant de chiffrer des messages au protagoniste "d'identité" id. Par exemple supposons qu'Alice souhaite envoyer un message à Bob, elle le chiffre à l'aide de l'adresse mail de Bob `bob@bob.com` et la clef publique MPK (qu'elle utilise aussi pour envoyer des messages aux amis de Bob) fourni par le tiers de confiance. En revanche Bob est incapable de connaître par lui-même sa clef privée sk_{id}, c'est le centre dérivation de clef qui lui fournit.

Le problème maintenant étant que presque toute identité id doit être reliée à une clef secrète sk_{id}. Il fallut alors attendre près de 20 ans pour voir apparaitre les premiers IBE avec les propositions de [SOK00] et [BF01] reposant sur la méthode des "couplages" (*pairing*) [Jou00] avec des courbes elliptiques. D'autres propositions s'ensuivirent et se fondant

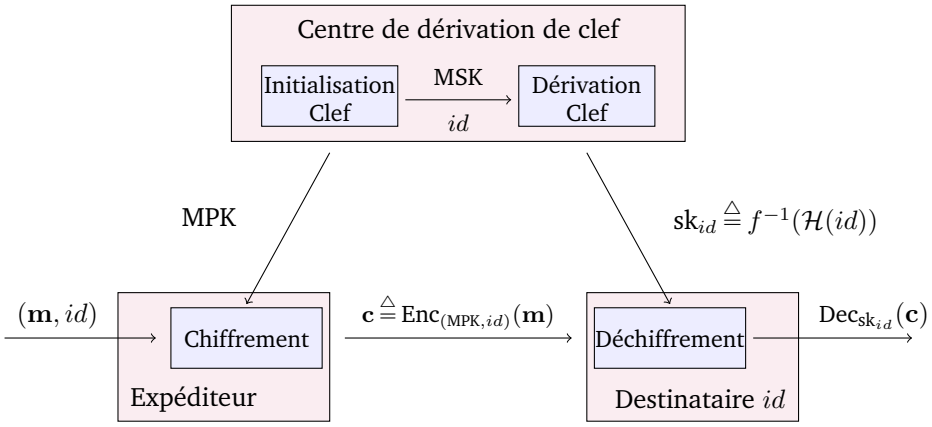


Figure 1.4 – IBE de GPV [GPV08a]

sur différents paradigmes mathématiques comme par exemple le problème des résidus quadratiques [Coc01] ou le problème dit de Diffie-Hellman (retrouver g^{ab} à partir de g^a, g^b et g un générateur d'un groupe public) [DG17b]. Il est à noter que de récents progrès ont été fait concernant des méthodes générales pour de la conception d'IBE. Les travaux de [DG17a] ont en effet montré comment construire des IBE prouvés surs avec l'utilisation d'une nouvelle primitive cryptographique : les signatures uniques avec chiffrement (OTSE). Cette primitive fut alors instanciée dans [Döt+18] à l'aide de LWE ou encore du problème *Low Parity with Noise* (LPN).

Nous présentons ici la proposition de [Gab+17b] permettant d'obtenir le premier IBE dont la sécurité repose sur des hypothèses issues de la théorie des codes correcteurs. Dans un certain sens ce schéma peut-être vu comme une adaptation de l'IBE introduit par [GPV08b] dans le paradigme des réseaux euclidiens. L'approche de [GPV08b] utilise comme brique élémentaire pour l'IBE un schéma de signature de type hache et signe de la façon suivante. On se donne une fonction de hachage cryptographique $\mathcal{H}$ et une fonction à sens unique f admettant comme trappe MSK. Le centre de dérivation de clef calcule alors tout simplement avec MSK :

$$\mathrm{sk}_{id} \in f^{-1}(\mathcal{H}(id)).$$

Nous résumons ce schéma dans la figure 1.4. Dans l'instanciation de [GPV08b], la signature tire des vecteurs courts qui additionnés avec le haché du message donne un point d'un réseau Euclidien public. Ces vecteurs courts sont alors utilisés dans un schéma de chiffrement dont la sécurité repose [GPV08b, §7.1, p26]) sur le problème LWE [Reg05].

Les auteurs de [Gab+17b] ont alors proposé d'utiliser la signature `RankSign` et le chiffrement `RankPKE` que nous avons vu dans §1.3.5. Ces derniers ont montré comment réduire la sécurité de leur IBE (dans un modèle que nous ne préciserons pas) aux problèmes :

— distinguer une clef publique de `RankSign` d'une matrice de parité aléatoire de même taille,

— le problème RSL (voir le problème 1.15 dans §1.3.5).

Le schéma. Nous donnons maintenant le cadre général de [Gab+17b] pour obtenir un IBE reposant sur les codes. Nous présentons ici une version légèrement plus générale (contribution de cette thèse, [DT18a]) englobant métrique rang [Gab+17b] et de Hamming. Les auteurs de [Gab+17b] proposèrent d'utiliser `RankSign` comme signature mais dans notre description cette-dernière sera vue comme une boîte noire.

Dans toute la suite $\mathbb{F}$ désignera un corps fini sur lequel on considèrera les codes. Le cas $\mathbb{F} = \mathbb{F}_2$ correspondra à la métrique de Hamming tandis que $\mathbb{F} = \mathbb{F}_{q^m}$ sera pour la métrique rang et $|\cdot|$ sera la norme associée.

Soit $\mathcal{C}_{\text{signe}}$ un $[n_{\text{signe}}, k_{\text{signe}}]$-code pour lequel on connaisse une trappe permettant étant donné $\mathbf{y} \in \mathbb{F}^{n_{\text{signe}}}$ de trouver facilement un mot de code $\mathbf{c}_{\mathbf{y}} \in \mathcal{C}_{\text{signe}}$ à distance w_{signe}

Soit maintenant $\mathcal{C}_{\text{dec}}$ un $[n_{\text{dec}}, k_{\text{dec}}]$-code tel qu'il existe un algorithme polynomial de décodage à distance $w_{\text{dec}} \in [\![0, n_{\text{dec}}]\!]$.

Considérons maintenant les matrices génératrices $\mathbf{G}_{\text{signe}}$ et $\mathbf{G}_{\text{dec}}$ de respectivement $\mathcal{C}_{\text{signe}}$ et $\mathcal{C}_{\text{dec}}$. Il est alors proposé dans [Gab+17b] d'utiliser comme clef maître publique et secrète :

— MSK la trappe de la signature permettant de décoder à distance w_{signe} de $\mathcal{C}_{\text{signe}}$,

— MPK $\triangleq (\mathcal{C}_{\text{signe}}, \mathcal{C}_{\text{dec}})$.

Considérons $\mathcal{H}$ une fonction de hachage cryptographique à valeurs dans $\mathbb{F}^{n_{\text{signe}}}$ et id l'identité d'un destinataire dans l'IBE. Le centre de dérivation de clefs calcule pour id un vecteur $\mathbf{u}_{id}$ tel que :

$$|\mathbf{u}_{id}\mathbf{G}_{\text{signe}} - \mathcal{H}(id)| = w_{\text{signe}} \tag{1.52}$$

qui est alors utilisé comme la clef secrète associée à id :

— $\text{sk}_{id} \triangleq \mathbf{u}_{id}$.

Le chiffrement utilisé dans l'IBE de [Gab+17b] est alors `RankPKE` que nous avons décrit pour la métrique de Hamming dans §1.3.5 et pour la métrique rang dans §1.3.2. La paire de clefs publique et privée est donnée par,

$$(\mathbf{G}_{\text{signe}}, \mathbf{G}_{\text{dec}}, id), \mathbf{u}_{id}.$$

Rappelons rapidement dans ce contexte le fonctionnement de ce schéma.

— **Chiffrement.** Considérons un message $\mathbf{m}$ à chiffrer. Les auteurs de [Gab+17b] ont introduit la fonction à trappe à sens unique suivante :

$$\begin{array}{rccc} g_{\mathbf{G}_{\text{signe}}, \mathbf{G}_{\text{dec}}, id} : & \mathbb{F}^{k_{\text{dec}}} & \longrightarrow & \mathbb{F}^{(k_{\text{signe}}+1)\times n_{\text{dec}}} \\ & \mathbf{m} & \longmapsto & \begin{bmatrix} \mathbf{G}_{\text{signe}}\mathbf{E} \\ \mathcal{H}(id)\mathbf{E} + \mathbf{m}\mathbf{G}_{\text{dec}} \end{bmatrix} \end{array}$$

où $\mathbf{E} \in \mathbb{F}^{n_{\text{signe}} \times n_{\text{dec}}}$. Pour la métrique rang $\mathbf{E}$ est homogène de poids w_{dec} tandis que pour la métrique de Hamming, les colonnes de $\mathbf{E}$ sont seulement choisies de poids w_{dec}.

— **Déchiffrement.** La clef secrète $\mathbf{u}_{id}$ est utilisée comme :

$$\begin{aligned} (\mathbf{u}_{id}, -1) g_{\mathbf{G}_{\text{signe}}, \mathbf{G}_{\text{dec}}, id}(\mathbf{m}) &= (\mathbf{u}_{id}, -1) \begin{bmatrix} \mathbf{G}_{\text{signe}}\mathbf{E} \\ \mathcal{H}(id)\mathbf{E} + \mathbf{m}\mathbf{G}_{\text{dec}} \end{bmatrix} \\ &= (\mathbf{u}_{id}\mathbf{G}_{\text{signe}} - \mathcal{H}(id))\,\mathbf{E} - \mathbf{m}\mathbf{G}_{\text{dec}} \end{aligned}$$

Il est alors démontré dans [Gab+17b] la proposition suivante (voir [Gab+17b, théorème 4]) :

Proposition 1.15. *La sécurité de l'IBE*[2] *en métrique rang se réduit aux problèmes :*

— *la version décisionnelle de* $\text{RSD}(n_{\text{signe}}, q, R_{\text{signe}}, \omega_{\text{signe}}, \mu)$ *où*

$$R_{\text{signe}} \triangleq \frac{k_{\text{signe}}}{n_{\text{signe}}} \quad \textit{et} \quad \omega_{\text{signe}} \triangleq \frac{w_{\text{signe}}}{n_{\text{signe}}}.$$

2. dans le modèle IND-CPA des IBE

— *le problème de distinguer un code aléatoire du code* LRPC*-augmenté de longueur* n_{signe} *et dimension* k_{signe} *utilisé dans* `RankSign`*,*

— *la version décisionnelle de* $\text{RSL}(n_{\text{signe}}, q, R_{\text{RSL}}, \omega_{\text{RSL}}, \mu, N)$ *où*

$$R_{\text{RSL}} \triangleq \frac{n_{\text{signe}} - k_{\text{signe}}}{n_{\text{signe}}}, \quad \omega_{\text{RSL}} \triangleq \frac{w_{\text{dec}}}{n_{\text{signe}}} \quad \textit{et} \quad N \triangleq n_{\text{dec}}.$$

En reprenant la démonstration de [Gab+17b, théorème 4] nous pouvons tout aussi bien obtenir la proposition suivante :

Proposition 1.16. *La sécurité de l'IBE en métrique de Hamming se réduit aux problèmes :*

— *la version décisionnelle de* $\text{SD}(n, q, R_{\text{signe}}, \omega_{\text{signe}})$ *où*

$$R_{\text{signe}} \triangleq \frac{k_{\text{signe}}}{n_{\text{signe}}} \quad \textit{et} \quad \omega_{\text{signe}} \triangleq \frac{w_{\text{signe}}}{n_{\text{signe}}}.$$

— *le problème de distinguer un code aléatoire du code utilisé pour la signature de type hache et signe,*

— *la version décisionnelle de* $\text{DOOM}(n_{\text{signe}}, q, R_{\text{DOOM}}, \omega_{\text{DOOM}}, N)$ *où*

$$R_{\text{DOOM}} \triangleq \frac{n_{\text{signe}} - k_{\text{signe}}}{n_{\text{signe}}}, \quad \omega_{\text{DOOM}} \triangleq \frac{w_{\text{dec}}}{n_{\text{signe}}} \quad \textit{et} \quad N \triangleq n_{\text{dec}}.$$

Cependant, il nous faut certaines conditions sur w_{signe} et w_{dec} pour que lors de la phase de déchiffrement le vecteur $(\mathbf{u}_{id}\mathbf{G}_{\text{signe}} - \mathcal{H}(id))\,\mathbf{E}$ soit de poids suffisamment faible pour pouvoir être décodé dans $\mathcal{C}_{\text{dec}}$ et ainsi retrouver $\mathbf{m}$. La proposition qui suit résume alors la contrainte sur les paramètres pour que ce décodage puisse en principe être possible.

Proposition 1.17 ([DT18c])**.** *Afin de pouvoir asymptotiquement décoder à rendement constant* $R \triangleq k_{\text{dec}}/n_{\text{dec}}$ *il doit exister* $\varepsilon(R) > 0$ *tel que les paramètres* $n_{\text{signe}}, w_{\text{signe}}$ *et* w_{dec} *vérifient nécessairement,*

— *pour la métrique rang :*

$$w_{\text{signe}} w_{\text{dec}} = (1 - \varepsilon(R)) \min(m, n_{\text{dec}}). \tag{1.53}$$

— *pour la métrique de Hamming :*

$$w_{\text{signe}} w_{\text{dec}} = O(n_{\text{signe}}). \tag{1.54}$$

Démonstration de la proposition 1.17.

Nous séparons la preuve en deux parties selon la norme utilisée.

Métrique rang. Dans ce cas, comme prouvé dans [Gab+17b, §3.2] le poids rang de $\left(\mathbf{u}_{id}\mathbf{G}_{\mathcal{C}_{\text{signe}}} - \mathcal{H}(id)\right)\mathbf{E}$ est avec une probabilité élevée $w_{\text{signe}} w_{\text{dec}}$. Rappelons que le code considéré pour décoder $\mathcal{C}_{\text{dec}}$ est sur l'alphabet $\mathbb{F}_{q^m}$. Une condition alors nécessaire pour être en mesure de décoder à rendement constant est alors la suivante. Le poids w de l'erreur doit être au plus une fraction constante du degré de l'extension m de $\mathbb{F}_{q^m}$ et de la longueur n_{dec} du code considéré pour décoder, *i.e* : pour un certain $0 < \varepsilon(R) < 1$:

$$\frac{w}{\min(m, n_{\text{dec}})} \leqslant 1 - \varepsilon(R).$$

Métrique de Hamming. Par définition le poids du vecteur $\mathbf{u}_{id}\mathbf{G}_{\mathcal{C}_{\text{signe}}} - \mathcal{H}(id)$ est w_{signe}. Les colonnes de la matrice $\mathbf{E}$ étant de poids w_{dec}, la probabilité (sur le tirage d'une colonne de $\mathbf{E}$) qu'un bit de

$$\left(\mathbf{u}_{id}\mathbf{G}_{\mathcal{C}_{\text{signe}}} - \mathcal{H}(id)\right)\mathbf{E}$$

soit égal à un est donnée par $\frac{1}{2}\left(1-2^{-cw_{\text{signe}}w_{\text{dec}}/n_{\text{signe}}(1+o(1))}\right)$ d'après le lemme 1.6. Le poids relatif de l'erreur à décoder est donc de cet ordre. Une condition nécessaire pour être en mesure de décoder asymptotiquement est alors que ce dernier soit $< 1/2$ d'où l'on tire la condition de la proposition.

□

Cette contrainte sur les paramètres de l'IBE est alors cruciale pour son fonctionnement. Malheureusement, comme nous le verrons dans la partie IV de ce document, cette dernière en métrique de Hamming implique une faiblesse fatale. L'IBE en métrique de Hamming est cassé [DT18c] avec le plus simple des algorithmes de décodage générique. Concernant la métrique rang, nous présentons aussi une attaque [DT18c] brisant les paramètres proposés dans [Gab+17b]. En revanche, contrairement au cas de la métrique de Hamming, nous montrons qu'il existe toujours une petite zone de paramètres pour contrer notre attaque. L'origine de cette différence est la liberté sur le paramètre m de l'extension de corps.

1.4.4 La signature KKS

Nous allons maintenant présenter rapidement la signature KKS [KKS97 ; KKS05] dont la sécurité repose en partie sur le problème du décodage générique en métrique de Hamming et plus particulièrement sur l'équivalent de RSL dans cette métrique (et non DOOM). La primitive KKS, contrairement à CFS et `RankSign` ne nécessite pas de code que l'on sait décoder. Cette-dernière est essentiellement spécifiée de la façon suivante :

Les paramètres : des entiers n, k, N, K où $n, K \leqslant N$ et typiquement $n \ll N$.

Clef secrète : des vecteurs $\mathbf{e}_1, \cdots, \mathbf{e}_k \in \mathbb{F}_q^N$ dont *les supports sont tous inclus dans un même ensemble* $\mathcal{J} \subseteq [\![1, N]\!]$ de taille n, *i.e* :

$$\forall i \in [\![1, k]\!], \quad \mathrm{Supp}(\mathbf{e}_i) \subseteq \mathcal{J}.$$

Clef publique : une matrice aléatoire $\mathbf{H} \in \mathbb{F}_q^{(N-K)\times N}$ et les syndromes

$$\mathbf{e}_1\mathbf{H}^\intercal, \cdots, \mathbf{e}_k\mathbf{H}^\intercal.$$

Signature d'un message $\mathbf{m} \in \mathbb{F}_q^k$: la signature est tout simplement

$$\mathbf{x} \triangleq \sum_{i=1}^{k} m_i \mathbf{e}_i. \tag{1.55}$$

Ici le fait que tous les $\mathrm{Supp}(\mathbf{e}_i)$ soient inclus dans un même ensemble $\mathcal{J}$ de taille n permet d'affirmer que $|\mathbf{x}| \leqslant n$.

Vérification de $(\mathbf{m}, \mathbf{x})$: on s'assure que $\sum_{i=1}^{k} m_i \mathbf{s}_i = \mathbf{x}\mathbf{H}^\intercal$ et $|\mathbf{x}| \leqslant n$.

Cette primitive de signature fait clairement fuiter de l'information, le vecteur $\mathbf{x}$ révèle à chaque signature une partie du support des erreurs $\mathbf{e}_i$ qui sont secrètes. Ce schéma ne peut donc qu'au mieux être considéré comme une signature unique, *i.e* : pour une clef secrète donnée on ne peut signer qu'une seule fois. De plus, il s'avère que le choix des paramètres doit être fait avec une grande attention à la lumière des attaques [COV07] retrouvant à partir de $O(\log_2(n))$ signatures le support $\mathcal{J}$ et celle de [OT11] retrouvant le secret seulement à partir des données publiques. Il est à noter que l'attaque de [OT11] est de nature exponentielle.

Le problème provient de l'équivalent de RSL en métrique de Hamming et est le suivant. Un attaquant contre KKS commence par décoder de façon quelconque les syndromes publics, *i.e* : il détermine $\mathbf{y}_1, \cdots, \mathbf{y}_N \in \mathbb{F}_q^N$,

$$\mathbf{y}_1\mathbf{H}^\intercal = \mathbf{e}_1\mathbf{H}^\intercal, \cdots, \mathbf{y}_k\mathbf{H}^\intercal = \mathbf{e}_k\mathbf{H}^\intercal$$

Les vecteurs $\mathbf{y}_i$ peuvent s'écrire comme $\mathbf{c}_i + \mathbf{e}_i$ où les $\mathbf{c}_i$ sont des mots du $[N, K]_q$-code $\mathcal{C}$ de matrice de parité $\mathbf{H}$. Il est donc clair que le code $\mathcal{C}'$ défini comme le code $\mathcal{C}$ auquel on ajoute l'espace engendré par les $\mathbf{y}_i$ vérifie :

$$\mathcal{C}' = \mathcal{C} + \text{Vect}_{\mathbb{F}_q}(\mathbf{y}_1, \cdots, \mathbf{y}_k) = \mathcal{C} + \text{Vect}_{\mathbb{F}_q}(\mathbf{e}_1, \cdots, \mathbf{e}_k)$$

Le code $\mathcal{C}'$ s'obtient facilement à partir des données publiques. Or il y a dans ce dernier des mots $\mathbf{e}_i$ de petit poids ($n \ll N$) *et* dont les supports sont tous inclus dans un même ensemble. Les mots de poids faible dans $\mathcal{C}'$ sont donc de l'ordre de q^k. Ils peuvent alors être retrouvés avec des algorithmes de décodage générique dits par ensemble d'information que nous allons décrire dans le prochain chapitre. Ces algorithmes de complexité exponentielle en le poids de l'erreur recherchée sont probabilistes et font essentiellement le pari de tirer un ensemble contenant le support d'une solution. Ici le fait que les erreurs recherchés de poids faible sont en nombre exponentiel q^k et partagent le même support améliore exponentiellement, comparativement au cas générique, les probabilités de réussite des algorithmes de décodage par ensemble d'information.

1.4.5 Le protocole d'identification de Stern à divulgation nulle de connaissance et l'approche de Fiat-Shamir

Nous allons ici présenter rapidement (sans entrer dans le détail des définitions et modèle de sécurité) le protocole d'identification à divulgation nulle de connaissance de Stern [Ste93b]. La sécurité de ce dernier repose sur le décodage générique. Commençons par rappeler qu'un tel protocole est entre deux protagonistes, un **Prouveur** et un **Vérifieur**. L'objectif du **Prouveur** est alors de *prouver* au **Vérifieur** qu'il possède un secret en ne révélant aucune information sur ce dernier. Le **Vérifieur** dispose quant à lui de données publiques. Dans ce que nous allons présenter il y a trois phases de communication pour établir le protocole,

1. Le **Prouveur** commence par faire *une mise en gage* sur des valeurs auprès du **Vérifieur**,
2. Le **Vérifieur** pose un défi au **Prouveur** en lien avec les mises en gage,
3. Le **Prouveur** répond au défi avec le secret et le **Vérifieur** s'assure de la réponse à l'aide des données publiques.

L'échange entre le **Prouveur** et le **Vérifieur** est alors appelé une *preuve de connaissance*. Le protocole de Stern utilisant des codes est maintenant donné comme suit.

Paramètres : $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, un poids $w \in [\![0, n]\!]$ et une fonction de hachage cryptographique $\mathcal{H}$,

Clef secrète : $\mathbf{e} \in \mathbb{F}_q^n$ de poids w,

Clef publique : $\mathbf{s} \triangleq \mathbf{e}\mathbf{H}^\intercal$.

Le protocole : commençons par noter $\mathcal{S}_n$ l'ensemble des permutations de $\mathbb{F}_2^n$. Les mises en gage sont définies pour $\mathbf{y} \in \mathbb{F}_2^n$ et $\sigma \in \mathcal{S}_n$ comme :

$$c_0 \triangleq \mathcal{H}(\sigma(\mathbf{y})), \quad c_1 \triangleq \mathcal{H}(\mathbf{y}\mathbf{H}^\intercal, \sigma) \quad \text{et} \quad c_2 \triangleq \mathcal{H}(\sigma(\mathbf{y} + \mathbf{e})).$$

Le défi sera dans le protocole $b \in \{0,1,2\}$. La réponse du **Prouveur** est alors selon b :

$$A(0) \triangleq (\mathbf{y}+\mathbf{e}, \sigma), \quad A(1) \triangleq (\sigma(\mathbf{y}), \sigma(\mathbf{e})) \quad \text{et} \quad A(2) \triangleq (\mathbf{y}, \sigma).$$

En résumé le protocole [Ste93b] est défini comme suit,

	Prouveur		**Vérifieur**
Mise en gage	$\sigma \leftarrow \mathcal{S}_n$ $\mathbf{y} \hookleftarrow \mathbb{F}_2^n$	$\xrightarrow{c_0, c_1, c_1}$	
Défi		$\xleftarrow{b}$	$b \hookleftarrow \{0,1,2\}$
Réponse		$\xrightarrow{A(b)}$	Vérification

L'étape de vérification consiste quant à elle à retrouver deux des mises en gage sur trois à l'aide de $A(b)$ et de la clef publique. En effet, le **Vérifieur** peut facilement s'assurer que :

— $\mathcal{H}\left(\mathbf{y}\mathbf{H}^\intercal, \sigma\right), \mathcal{H}(\sigma(\mathbf{y}+\mathbf{e})) \stackrel{?}{=} c_1, c_2$ s'il dispose de $A(0)$,

— $\mathcal{H}(\sigma(\mathbf{y})), \mathcal{H}(\sigma(\mathbf{y}+\mathbf{e})) \stackrel{?}{=} c_0, c_2$ *et* $|\sigma(\mathbf{e})| \stackrel{?}{=} w$ s'il dispose de $A(1)$,

— $\mathcal{H}(\sigma(\mathbf{y})), \mathcal{H}(\mathbf{y}\mathbf{H}^\intercal, \sigma) \stackrel{?}{=} c_0, c_1$ s'il dispose de $A(2)$.

Ce schéma bénéficie alors des trois propriétés suivantes :

— *Complétude* : toute preuve de connaissance d'un **Prouveur** connaissant le secret est acceptée par le **Vérifieur**,

— *Sans fuite d'information* : les échanges entre le **Prouveur** et le **Vérifieur** ne révèlent aucune information sur le secret dans le sens suivant. A partir d'une exécution valide on peut construire sans le secret une autre exécution valide avec le même défi et qui est indistingable de la précédente,

— *Correction* : toute preuve de connaissance d'un **Prouveur** ne disposant pas du secret est acceptée par le **vérifieur** avec probabilité $\leqslant 2/3$.

La probabilité $2/3$ de la propriété de correction est optimale dans le sens suivant. Tout d'abord, un **Prouveur** ne connaissant pas le secret peut toujours fabriquer les mises en gage de façon à ce qu'il puisse répondre à deux défis sur trois. De plus, il est démontré dans [AGS11, théorème IV.1] que si un **Prouveur** ne connaissant pas le secret (un adversaire) réussit à faire accepter au **Vérifieur** une preuve de connaissance avec une probabilité $> 2/3 + \varepsilon$ alors nous pouvons trouver avec une probabilité $> \varepsilon$ une erreur $\mathbf{e}' \in \mathbb{F}_q^n$ de poids w telle que $\mathbf{e}'\mathbf{H}^\intercal = \mathbf{s}$ ou une collision sur la fonction de hachage.

De cette façon la sécurité de ce protocole se réduit dans le modèle de l'oracle aléatoire au problème SD(n, q, R, ω) où $R \triangleq k/n$ et $\omega \triangleq w/n$. Nous avons dès-lors tout intérêt à choisir les paramètres du problème SD pour lesquels ce dernier est le plus difficile dans l'état de l'art algorithmique.

Notons qu'il existe une variante de ce protocole avec des matrices génératrices [Vér96]. Cette-dernière repose aussi sur le problème du décodage mais mène à des équilibres différents dans les mises en gage et réponses apportés.

Transformation de Fiat-Shamir [FS87]. Il fut alors proposé dans [Ste93b] de construire avec ce protocole un schéma de signature. L'idée est d'utiliser la transformation générique de Fiat-Shamir permettant à partir d'un protocole à divulgation nulle de connaissance d'obtenir une signature. Cette-dernière consiste dans notre cas à procéder comme suit. Intuitivement une signature avec cette transformation correspond à un certain nombre

de traces de l'exécution du protocole (où le signataire joue le rôle du **Prouveur** et du **Vérifieur**) tandis que la vérification consiste tout simplement à s'assurer que ces traces sont toutes valides.

Plus concrètement, dans notre cas le signataire dispose comme clef secrète d'un vecteur $\mathbf{e} \in S_{w,n}$ tandis que la clef publique est $\mathbf{s} \stackrel{\triangle}{=} \mathbf{e}\mathbf{H}^\intercal$ et $\mathbf{H}$. De plus notons $\mathtt{GPA}(\cdot)$ un générateur pseudo-aléatoire public à valeurs dans $\{0,1,2\}$. Pour signer un message $\mathbf{m}$ on commence par générer N instances de mise en gage du **Prouveur** :

$$(\sigma_1, \mathbf{y}_1), \cdots, (\sigma_N, \mathbf{y}_N)$$

et calculer les mises en gage de ces valeurs :

$$(c_0^1, c_1^1, c_2^1), \cdots, (c_0^N, c_1^N, c_2^N).$$

puis on les hache avec en plus le message,

$$h_1 \stackrel{\triangle}{=} \mathcal{H}\left(c_0^1, c_1^1, c_2^1, \mathbf{m}\right), \cdots, h_N \stackrel{\triangle}{=} \mathcal{H}\left(c_0^N, c_1^N, c_2^N, \mathbf{m}\right)$$

Une fois que cela est fait on calcule les défis du vérifieur à l'aide de $\mathtt{GPA}$ utilisant comme graines ces hachés,

$$b^1 \stackrel{\triangle}{=} \mathtt{GPA}(h_1), \cdots, b^N \stackrel{\triangle}{=} \mathtt{GPA}(h_N).$$

Finalement la signature de $\mathbf{m}$ est,

$$\left(A(b^1), c_0^1, c_1^1, c_2^1\right), \cdots, \left(A(b^N), c_0^N, c_1^N, c_2^N\right)$$

qui peut facilement être vérifiée en s'assurant avec les données publiques que chaque preuve de connaissance est valide.

Le protocole de Stern est ici répété N fois pour la raison suivante. Il est possible de faire une preuve de connaissance sans le secret avec une probabilité au plus de $2/3$ (à moins d'attaquer l'une des hypothèses calculatoires du schéma). De cette façon, sans la clef secrète on peut produire avec au mieux une probabilité $(2/3)^N$ une signature valide. Donc pour s'assurer que le schéma est sûr pour λ bits de sécurité, il suffit de prendre N tel que

$$\left(\frac{2}{3}\right)^N = 2^{-\lambda} \iff N = \frac{\lambda}{\log_2\left(\frac{3}{2}\right)}.$$

L'avantage d'une telle signature est alors que les tailles de clef publique sont très petites. En effet, nous pouvons choisir une matrice de parité $\mathbf{H}$ aléatoire et donc donner à la place une graine et un générateur pseudo-aléatoire associé. En revanche, l'inconvénient est ici que la longueur de signature sera grande. Une signature correspondant à $\lambda/\log_2(3/2)$ traces d'exécution, elle meme de taille un $O(\lambda)$, du protocole de Stern pour λ bits de sécurité. Il fut alors proposé dans [AGS11] de nombreuses astuces afin réduire la longueur d'une trace d'exécution comme par exemple remplacer de façon maline certaines des quantités aléatoires

$$\sigma_i, \mathbf{y}_i, \sigma(\mathbf{y}_i), \sigma(\mathbf{e}), \mathbf{y}_i + \mathbf{e}$$

par des graines servant à les générer. Quoiqu'il en soit dans l'état de l'art actuel, la longueur d'une signature est de l'ordre de plusieurs centaines de kilobits pour 128 bits de sécurité.

Chapitre 2

Décodage par ensemble d'information en métrique de Hamming

Introduction

L'algorithme de Prange. La vocation de tout crypto-système utilisant des codes correcteurs comme brique élémentaire est de faire reposer la sécurité sur entre autres la difficulté du décodage d'un code aléatoire. Il est donc essentiel de chercher et étudier les algorithmes, dits de décodage générique, le résolvant. Les meilleures solutions, que nous allons décrire dans ce chapitre, sont à ce jour toutes des améliorations d'un vieil algorithme [Pra62] introduit par Prange. On parle d'algorithmes par ensemble d'information (ISD pour *Information Set Decoding*). Dans le contexte de la métrique de Hamming et du corps $\mathbb{F}_q$, partons d'un $[n,k]_q$-code $\mathcal{C}$ et $\mathbf{y}$ un mot de code bruité à distance w. Notre objectif est de retrouver $\mathbf{c} \in \mathcal{C}$ tel que $|\mathbf{y}-\mathbf{c}| = w$. Le code $\mathcal{C}$ étant un sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k, il existe par définition (raisonner sur la matrice génératrice du code de rang k) un ensemble $\mathcal{I} \subseteq [\![1,n]\!]$, dit *ensemble d'information*, de taille k déterminant uniquement tout mot de code :

$$\forall \mathbf{x} \in \mathbb{F}_q^k \quad : \quad \exists! \mathbf{c} \in \mathcal{C} \text{ que l'on calcule facilement tel que } \mathbf{c}_{\mathcal{I}} = \mathbf{x}.$$

L'idée de Prange pour trouver un $\mathbf{c}^{\text{sol}} \in \mathcal{C}$ qui convient où par définition $\mathbf{y} = \mathbf{c}^{\text{sol}} + \mathbf{e}^{\text{sol}}$ avec $|\mathbf{e}^{\text{sol}}| = w$ est la suivante. Nous commençons par tirer aléatoirement un ensemble d'information $\mathcal{I}$ et si nous avons eu de la chance, à savoir :

$$\forall i \in \mathcal{I}, \quad e_i^{\text{sol}} = 0 \tag{2.1}$$

alors il nous suffit de calculer le mot de code $\mathbf{c}$ tel que $\mathbf{c}_{\mathcal{I}} = \mathbf{y}_{\mathcal{I}}$ et par unicité $\mathbf{c} = \mathbf{c}^{\text{sol}}$. L'algorithme de Prange consiste donc à tirer un ensemble d'information $\mathcal{I}$, calculer l'unique mot de code égal à $\mathbf{y}$ en les positions données par $\mathcal{I}$ puis vérifier que le mot obtenu est à distance w. Nous ferons donc en moyenne $1/p$ tirages avant de trouver un mot de code à distance w où p est la probabilité que la contrainte (2.1) soit vérifiée pour au moins l'une des solutions. Comme nous le verrons cette probabilité p dépend fortement de w. Plus précisément, nous verrons que $1/p$ est avec $n \to +\infty$, $R \triangleq k/n$ fixé et $\omega \triangleq w/n$:

— polynomiale pour $\omega = \left(\frac{q-1}{q}\right)(1-R)$,

— exponentielle pour tout $\omega \in \left]0, \left(\frac{q-1}{q}\right)(1-R)\right[$,

— sous-exponentielle pour $\omega = o(1)$.

La majorité des solutions proposées pour améliorer l'algorithme de Prange sont ensuite parties d'une même idée. Cependant, afin d'en comprendre son origine revenons un peu en arrière.

L'approche de Dumer : une recherche de collisions. La plus simple des idées pour retrouver un mot de code à distance w de $\mathbf{y}$ est d'énumérer *toutes* les erreurs $\mathbf{e}$ de poids w jusqu'à en trouver une telle que $\mathbf{y} - \mathbf{e} \in \mathcal{C}$. Cette approche, peu maline, coûte alors :

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

Dumer [Dum89] proposa pour améliorer cette énumération exhaustive de se ramener à un problème de collisions pour tirer parti du paradoxe des anniversaires. Son idée fut la suivante. Commençons par nous donner une matrice de parité $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ du code $\mathcal{C}$ que nous décomposons en deux comme $\mathbf{H} = \begin{pmatrix}\mathbf{H}_1 & \mathbf{H}_2\end{pmatrix}$. Découpons de même $\mathbf{y}$ en deux, $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$. Nous calculons alors les deux listes incluses dans $\mathbb{F}_q^{n-k}$:

$$\mathcal{L}_1 \stackrel{\triangle}{=} \left\{(\mathbf{y}_1 - \mathbf{e}_1)\mathbf{H}_1^\intercal : |\mathbf{e}_1| = w/2\right\} \quad \text{et} \quad \mathcal{L}_2 \stackrel{\triangle}{=} \left\{(\mathbf{e}_2 - \mathbf{y}_2)\mathbf{H}_2^\intercal : |\mathbf{e}_2| = w/2\right\}.$$

Le temps de construction de ces deux listes est alors en racine carrée de l'énumération exhaustive :

$$\binom{n/2}{w/2}(q-1)^{w/2} = \tilde{O}\left(\sqrt{\binom{n}{w}(q-1)^w}\right) \quad \text{(voir le lemme 1.2).} \tag{2.2}$$

La remarque essentielle de Dumer fut que par définition tout couple :

$$((\mathbf{y}_1 - \mathbf{e}_1)\mathbf{H}_1^\intercal, (\mathbf{e}_2 - \mathbf{y}_2)\mathbf{H}_2^\intercal) \in \mathcal{L}_1 \times \mathcal{L}_2 \text{ tel que } (\mathbf{y}_1 - \mathbf{e}_1)\mathbf{H}_1^\intercal = (\mathbf{e}_2 - \mathbf{y}_2)\mathbf{H}_2^\intercal$$

donne $(\mathbf{e}_1, \mathbf{e}_2)$ comme solution. Il suffit donc de trouver les collisions entre $\mathcal{L}_1$ et $\mathcal{L}_2$ pour obtenir des solutions. Ces deux listes étant dans notre cas aléatoires (le code considéré pour le décodage générique est aléatoire) et de taille donnée par (2.2), nous nous attendons à trouver d'après une version plus faible du paradoxe des anniversaires :

$$\frac{\binom{n/2}{w/2}^2(q-1)^w}{q^{n-k}} = \tilde{O}\left(\frac{\binom{n}{w}(q-1)^w}{q^{n-k}}\right)$$

solutions. En utilisant des techniques classiques de type table de hachage, tri de listes, le coût de cette opération est donné par le nombre attendu de solutions.

En résumé, Dumer avec son algorithme trouve à un facteur polynomial près :

$$\frac{\binom{n}{w}(q-1)^w}{q^{n-k}} \quad \text{solutions en temps} \quad \sqrt{\binom{n}{w}(q-1)^w} + \frac{\binom{n}{w}(q-1)^w}{q^{n-k}} \tag{2.3}$$

Dans le cas où w est égal à la distance de Gilbert-Varshamov, c'est à dire essentiellement $\binom{n}{w}(q-1)^w = q^{n-k}$, l'algorithme de Dumer gagne donc un facteur quadratique par rapport à la recherche exhaustive. Cependant nous avons mieux que cela comme montré dans la proposition qui suit,

Proposition 2.1. *La complexité de l'algorithme de Prange pour résoudre* $\mathrm{SD}(n, q, R, \omega)$ *pour* $\omega = o(1)$ *et* $R = 1 - h_q(\omega)$ *est de la forme :*

$$q^{n\cdot(1-R)(1+o(1))}$$

alors que celle de l'approche de Dumer est :

$$q^{n\cdot\frac{1-R}{2}(1+o(1))}.$$

L'algorithme de Dumer a donc un gain quadratique par rapport à celui de Prange pour décoder, à la distance de Gilbert-Varshamov, des codes *au rendement tendant vers* 1.

Cependant l'intérêt premier de cette approche n'est pas ici. En effet, une remarque fondamentale est que l'algorithme trouve *toutes* les solutions au problème du décodage considéré. De plus, nous pouvons choisir le paramètre w au dessus de la borne de Gilbert-Varshamov pour que l'algorithme trouve toutes les solutions (en nombre exponentiel) en temps amorti un, *i.e* :

$$\sqrt{\binom{n}{w}(q-1)^w} = \frac{\binom{n}{w}(q-1)^w}{q^{n-k}} \iff \binom{n}{w}(q-1)^w = \left(q^{n-k}\right)^2.$$

Comme nous allons l'expliquer, l'idée des améliorations de l'algorithme de Prange a alors justement été de combiner ces deux remarques sur l'algorithme de Dumer [Dum89]. Le problème de décodage initial est pour cela ramené à celui de trouver de nombreuses solutions au problème de décodage d'un sous-code au rendement proche de un en temps amorti un.

Une approche combinée : les codes poinçonnés. L'idée fondamentale des algorithmes ISD pour améliorer l'algorithme de Prange n'est plus pour un mot $\mathbf{y}$ de calculer l'unique mot de code égal à ce dernier sur k positions. Il s'agit désormais de calculer *un ensemble* de mots de codes à faible distance p de $\mathbf{y}$ sur un ensemble de positions un peu plus grand :

$$\mathcal{S} \subseteq \{\mathbf{c} \in \mathcal{C} : |\mathbf{c}_{\mathcal{J}} - \mathbf{y}_{\mathcal{J}}| = p\} \quad \text{où} \quad \mathcal{J} \subseteq [\![1, n]\!] \quad \text{de taille} \quad k+\ell \quad \text{avec} \quad \ell > 0.$$

L'ensemble $\mathcal{S}$ correspond exactement à des solutions du problème de décodage de $\mathbf{y}_{\mathcal{J}}$ à distance p du code dit poinçonné en les positions de $\overline{\mathcal{J}}_\ell$:

$$\mathcal{C}_{\text{poinc}} \stackrel{\triangle}{=} \left\{\mathbf{x} \in \mathbb{F}_q^{k+\ell} : \exists \mathbf{c} \in \mathcal{C}, \mathbf{c}_{\mathcal{J}} = \mathbf{x}\right\}$$

qui est un code longueur $k+\ell$ et de dimension k si $\mathcal{J}$ contient un ensemble d'information, ce que l'on suppose dans la suite. Désormais $\mathcal{S}$ contiendra une solution $\mathbf{c}^{\text{sol}}$ si $\mathbf{y} = \mathbf{c}^{\text{sol}} + \mathbf{e}^{\text{sol}}$ pour $\mathbf{e}^{\text{sol}}$ de poids w vérifiant :

$$|(\mathbf{e}^{\text{sol}})_{\mathcal{J}}| = p. \tag{2.4}$$

Le décodage d'un code de dimension k et de longueur n est donc ramené au décodage à distance p d'un "sous-code" de longueur $k+\ell$, de dimension k et ainsi de rendement $k/(k+\ell) \approx 1$. Nous pouvons donc utiliser l'approche de Dumer pour résoudre ce problème. La probabilité de réussite de l'algorithme est ensuite donnée par la probabilité que l'une des solutions retrouvée lors du décodage du code poinçonné donne une solution. Dénotons, pour un ensemble $\mathcal{J}$ fixé, par $\alpha_{p,\ell}$ la probabilité sur les erreurs de poids w que (2.4) soit vérifiée. Comme nous le verrons dans la proposition 2.7 nous avons :

$$\alpha_{p,\ell} = \tilde{O}\left(\frac{q^\ell \binom{n-k-\ell}{w-p}(q-1)^{w-p}}{\min\left(q^{n-k}, \binom{n}{w}(q-1)^w\right)}\right). \tag{2.5}$$

De cette façon, si l'algorithme du décodage de code poinçonné trouve un ensemble de solutions $\mathcal{S}$, la probabilité de réussite du décodage est à un facteur constant près donnée par $\min(1, |\mathcal{S}| \cdot \alpha_{p,\ell})$. Nous avons alors la proposition suivante dont nous discuterons dans §2.3 :

Proposition 2.2. *Supposons que l'on dispose d'un algorithme trouvant un ensemble de solutions $\mathcal{S}$ en temps moyen T du décodage à distance p d'un code aléatoire de longueur $k+\ell$ et de dimension k. Alors, le problème du décodage à distance w d'un code aléatoire*

de dimension k et de longueur n, pour les paramètres w, k, ℓ et p, peut être résolu en temps moyen :

$$\tilde{O}\left(T \cdot \max\left(1, \frac{1}{|\mathcal{S}| \cdot \alpha_{p,\ell}}\right)\right).$$

L'algorithme utilisé pour le décodage du code poinçonné est donc une paramétrisation essentielle des ISD. Une stratégie optimale pour minimiser le coût des ISD est de trouver un décodeur du code poinçonné en temps amorti polynomial, *ie :*

$$\frac{T}{|\mathcal{S}|} = P(n) \tag{2.6}$$

pour un certain $P \in \mathbb{R}[X]$. Au même instant nous souhaiterions choisir (p, ℓ) pour que la probabilité $\alpha_{p,\ell}$ soit la plus proche de 1. Il s'agit cependant de deux requêtes contradictoires. En effet, nous pouvons tout d'abord vérifier avec (2.5) que pour le paramètre ℓ fixé :

$$p \mapsto \alpha_{p,\ell} \text{ est décroissante}$$

où nous supposons $w \leqslant \left(\frac{q-1}{q}\right)(n-k-\ell)$. Nous devons donc diminuer p pour rapprocher la probabilité $\alpha_{p,\ell}$ de 1. En revanche, le nombre de solutions du décodage du code poinçonné est croissant avec p à ℓ fixé étant donné que ce-dernier est donné par :

$$\frac{\binom{k+\ell}{p}(q-1)^p}{q^\ell} \tag{2.7}$$

Dans ce cas, plus p est petit, plus il est difficile de trouver des solutions et de surcroît avec des algorithmes en temps amorti $P(n)$. Par exemple, avec l'algorithme de Dumer que nous avons décrit précédemment (voir (2.3)), pour trouver toutes les solutions (2.7) en temps amorti polynomial, il nous faut choisir (p, ℓ) tel que :

$$\frac{\binom{k+\ell}{p}(q-1)^p}{q^\ell} = \sqrt{\binom{k+\ell}{p}(q-1)^p}.$$

La distance p dépend de ℓ et nous ne pouvons pas la diminuer arbitrairement. A ce stade les améliorations de l'algorithme de Dumer dans le contexte des ISD ont consisté en des solutions reposant toujours sur une recherche de collisions mais permettant essentiellement de faire diminuer p tout en obtenant des solutions en temps amorti polynomial. Les auteurs de [MMT11 ; Bec+12] ont montré dans le cas où $q = 2$ comment adapter des méthodes de résolution du problème de sac à dos [HJ10 ; BCJ11] à partir de techniques comme celles des représentations, $1 + 1 = 0$ ou encore utilisant l'approche de [Wag02].

Les récents travaux de [MO15 ; BM17 ; BM18] ont cependant proposé de se ramener à la résolution d'un problème dit du voisin le plus proche, c'est à dire trouver des couples "proches" à partir de deux listes au lieu de faire une recherche de collisions exactes. Les algorithmes proposés ont alors le meilleur exposant asymptotique mais en revanche ces-derniers souffrent d'un facteur super-polynomial dans leur complexité. Nous ne rentrerons pas ici dans les détails et nous nous restreindrons aux algorithmes [Pra62 ; Dum91 ; Bec+12].

Le problème étudié. Dans tout ce chapitre nous considèrerons la version matrice de parité et syndrome du problème du décodage, c'est à dire :

Problème 1.7. (Problème du décodage générique de syndromes - SD(n, q, R, ω)) *Soient les entiers $k \triangleq \lfloor Rn \rfloor$ et $w \triangleq \lfloor \omega n \rfloor$.*

— Instance : *une matrice de parité* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *de rang* $n-k$*, un syndrome* $\mathbf{s} \in \mathbb{F}_q^{n-k}$.
— Recherche : *un vecteur d'erreur* $\mathbf{e} \in \mathbb{F}_q^n$ *de poids de Hamming* w *tel que* $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$.

De plus, nous supposerons dans toute la suite qu'une instance $(\mathbf{H}, \mathbf{s})$ pour les paramètres (n, q, R, ω) est un couple de variables aléatoires distribuées comme (voir §1.1.3 pour la définition de ω^-) :

$$\mathbf{H} \hookleftarrow \mathbb{F}_q^{(n-k)\times n}$$

$$(i) \text{ si } \omega \leqslant \omega^- \ : \ \mathbf{s} \stackrel{\triangle}{=} \mathbf{e}\mathbf{H}^\intercal \text{ où } \mathbf{e} \hookleftarrow S_w \quad ; \quad (ii) \text{ si } \omega > \omega^- \ : \ \mathbf{s} \hookleftarrow \mathbb{F}_q^{n-k}$$

pour $w \stackrel{\triangle}{=} \lfloor \omega n \rfloor$ et $k \stackrel{\triangle}{=} \lfloor Rn \rfloor$. La condition (i) est là pour promettre l'existence d'une solution (ce qui est le cas dans un contexte cryptographique) lorsque pour ces paramètres il existe une solution avec une probabilité exponentiellement faible tandis que (ii) illustre le fait que $\mathbf{e}\mathbf{H}^\intercal$ est statistiquement proche de l'uniforme pour $|\mathbf{e}|/n$ fixé et $> \omega^-$ (Proposition 1.4 dans §1.1.4).

Fixons maintenant pour ce qui suit la définition d'ensemble d'information.

Définition 2.1 (Ensemble d'information)**.** *Soient* $\mathcal{C}$ *un* $[n,k]_q$*-code et* $\mathcal{I} \subseteq [\![1, n]\!]$*. On dit que* $\mathcal{I}$ *est ensemble d'information de* $\mathcal{C}$ *si :*

$$\forall \mathbf{x} \in \mathbb{F}_q^k, \quad \exists! \mathbf{c} \in \mathcal{C} \quad \textit{tel que} \quad \mathbf{x} = \mathbf{c}_{\mathcal{I}}.$$

On vérifie alors facilement que si $\mathbf{H}$ désigne une matrice de parité d'un $[n,k]_q$-code $\mathcal{C}$ alors :

Fait 4. *$\mathcal{I}$ est un ensemble d'information de $\mathcal{C}$ $\iff$ $\mathbf{H}_{\overline{\mathcal{I}}}$ est inversible.*

Définissons désormais un code poinçonné :

Définition 2.2 (Code poinçonné)**.** *Soient* $\mathcal{C}$ *un* $[n,k]_q$*-code et* $\mathcal{J} \subseteq [\![1, n]\!]$ *de cardinal* $L \in [\![0, n]\!]$*. Nous définissons* $\mathcal{C}_{\text{poinc}}(\mathcal{J})$ *le code poinçonné en les positions* $\mathcal{J}$ *comme :*

$$\mathcal{C}_{\text{poinc}}(\mathcal{J}) \stackrel{\triangle}{=} \left\{ \mathbf{x} \in \mathbb{F}_q^{n-L} \ : \ \exists \mathbf{c} \in \mathcal{C}, \ \mathbf{c}_{\overline{\mathcal{J}}} = \mathbf{x} \right\}.$$

2.1 Algèbre linéaire : le pari de Prange

Fixons $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$ une instance du décodage que nous souhaitons résoudre à distance w.

L'algorithme. Prange se déroule en trois étapes que nous décrivons :

1. *Tirage de l'ensemble d'information* : On tire aléatoirement $\mathcal{I} \subseteq [\![1, n]\!]$ de taille k. Si $\mathbf{H}_{\overline{\mathcal{I}}}$ n'est pas inversible on recommence.
   Soit $\mathbf{P}$ une matrice de permutation renvoyant les k coordonnées de $\mathcal{I}$ sur $[\![n-k+1, n]\!]$.
2. *Algèbre linéaire* : On effectue une élimination Gaussienne sur les lignes de $\mathbf{HP}$. Soit $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ la matrice inversible correspondant à cette opération. Nous obtenons $\mathbf{H}' \in \mathbb{F}_q^{(n-k)\times k}$ telle que :
   $$\mathbf{SHP} = \begin{pmatrix} \mathbf{1}_{n-k} & \mathbf{H}' \end{pmatrix}.$$
   Une solution du système linéaire est donnée par :
   $$\mathbf{e}^{\text{sol}} \stackrel{\triangle}{=} (\mathbf{s}\mathbf{S}^\intercal, \mathbf{0}_k)\mathbf{P}^\intercal \tag{2.8}$$
   où $\mathbf{0}_k$ est le vecteur de taille k tout à 0.
3. *Test* : Il ne reste plus qu'à vérifier que $|\mathbf{s}\mathbf{S}^\intercal| = w$ la matrice $\mathbf{P}$ étant une permutation. Si ce n'est pas le cas on retourne à l'étape 1.

Analyse de l'algorithme. Une itération de l'algorithme de Prange est dite réussie si lors du tirage de $\mathcal{I}$ il s'agit bien d'un ensemble d'information et si $\mathbf{e}^{\text{sol}}$ est bien de poids w. Introduisons la probabilité de succès d'une itération comme :

$$p_w \triangleq \mathbb{P}\left(\text{une itération de l'algorithme de Prange est réussie}\right).$$

où la probabilité est sur $\mathbf{H}$, $\mathbf{s}$ et $\mathcal{I}$. Nous pouvons vérifier que :

$$p_w = \tilde{O}\left(\frac{\binom{n-k}{w}(q-1)^w}{\min\left(\binom{n}{w}(q-1)^w, q^{n-k}\right)}\right)$$

On en déduit alors la proposition qui suit :

Proposition 2.3. *La complexité de l'algorithme de Prange est donnée pour tout $n \in \mathbb{N}$ et w, k des fonctions de n par :*

$$\tilde{O}\left(\frac{\min\left(\binom{n}{w}(q-1)^w, q^{n-k}\right)}{\binom{n-k}{w}(q-1)^w}\right). \tag{2.9}$$

Quand n tend vers l'infini mais que w et k sont tels que $w = \lfloor \omega n \rfloor$ et $k = \lfloor Rn \rfloor$ pour les constantes $R \in]0,1[$ et $\omega \in \left]0, \left(\frac{q-1}{q}\right)(1-R)\right[$, la complexité est donnée par :

$$q^{n\cdot\left(\min(1-R, h_q(\omega)) - (1-R)h_q\left(\frac{\omega}{1-R}\right)\right)(1+o(1))}$$

et lorsque $\omega = \left(\frac{q-1}{q}\right)(1-R)$ par :

$$P(n)(1+o(1))$$

pour un certain polynôme P. Tandis que pour $\omega = o(1)$ la complexité s'écrit comme :

$$2^{n\cdot(-\omega\log_2(1-R)(1+o(1)))}.$$

Remarque 2.1. La quantité $\min\left(\binom{n}{w}(q-1)^w, q^{n-k}\right)$ marque ici la différence entre w/n fixé $> \omega^-$ où $\binom{n}{w}(q-1)^w > q^{n-k}$ lorsque l'on s'attend typiquement à un grand nombre de solutions contrairement à w/n fixé $\leqslant \omega^-$ où la solution est a priori unique.

La complexité de l'algorithme de Prange est donc exponentielle en n pour tout rendement $k/n = R \in]0,1[$ et $w/n \in [1, \left(\frac{q-1}{q}\right)(1-R)[$ fixés alors qu'elle devient polynomiale pour $w/n = \left(\frac{q-1}{q}\right)(1-R)$. En effet, le vecteur dont on teste le poids lors d'une itération s'écrit comme :

$$\mathbf{e}^{\text{sol}} \triangleq (\mathbf{s}\mathbf{S}^\intercal, \mathbf{0}_k)\mathbf{P}^\intercal$$

Or lorsque $w/n = \left(\frac{q-1}{q}(1-R)\right)$ nous sommes dans la zone où le syndrome $\mathbf{s}$ de $n-k$ symboles dans $\mathbb{F}_q$ est uniformément distribué. Le poids relatif typique de $\mathbf{e}^{\text{sol}}$ sera donc de l'ordre de $\left(\frac{q-1}{q}\right)(1-R)$.

De plus, on vérifie facilement que la complexité de l'algorithme de Prange est maximale en la distance relative de Gilbert-Varshamov ω^- où typiquement nous nous attendons à une solution.

Nous résumons la situation en traçant dans la figure 2.1 l'exposant asymptotique de l'algorithme pour $q = 2$, le rendement fixé $R = 1/2$ en fonction de ω et dans la figure 2.2 ce même exposant en fonction de R pour ω fixé à ω^-.

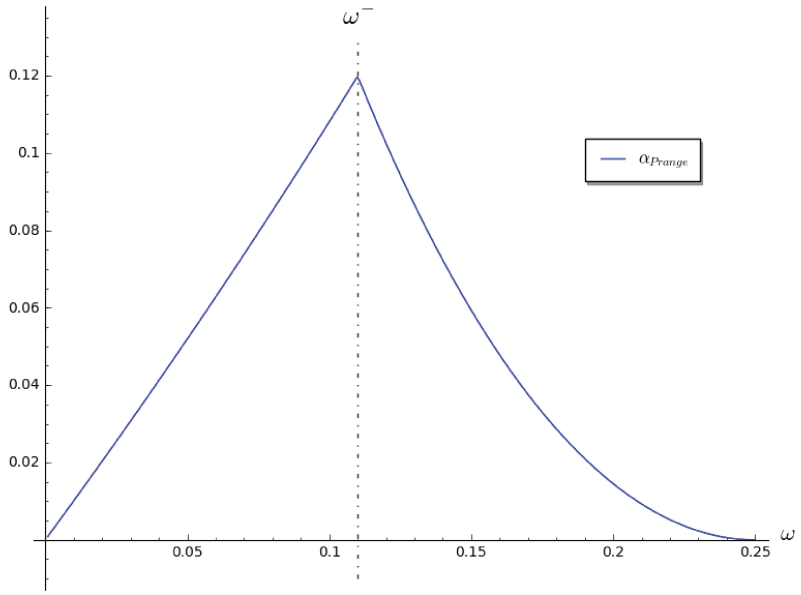


Figure 2.1 – Exposant asymptotique α_{Prange} de la complexité de l'algorithme de Prange en base 2 pour $q = 2$ et $R = 1/2$ en fonction de $\omega = w/n$.

Dans la section 2.3 nous présenterons diverses améliorations de l'algorithme de Prange. Comme nous le verrons, ces algorithmes diminuent l'exposant asymptotique de Prange en $\omega = \omega^-$. En revanche, ces améliorations s'amoindrissent avec ω s'écartant de ω^- jusqu'à relativement rapidement donner le même exposant que Prange. De plus, les divers algorithmes proposés dans le cas où $q = 2$ sont tous de complexité polynomiale lorsque la distance relative ω de décodage est égale à $(1 - R)/2$.

Cependant avant de présenter les ISD nous rappelons dans la section qui suit une autre approche qui sera utilisée dans la suite pour résoudre le problème du décodage.

2.2 Recherche de collisions

L'approche que nous présentons ici n'utilise pas d'algèbre linéaire pour résoudre le problème du décodage mais des collisions exactes. Cette méthode fut introduite pour la première fois par Dumer [Dum89] et comme nous le verrons cette dernière est plus efficace que l'algorithme de Prange pour des rendements k/n tendant vers 1.

2.2.1 L'approche de Dumer

Il est clair que nous pouvons trouver une solution $\mathbf{e}$ de poids w de $\mathbf{eH}^\intercal = \mathbf{s}$ par une énumération des $\binom{n}{w}(q-1)^w$ possibilités. Néanmoins nous pouvons être plus malin et utiliser une version légèrement plus faible du paradoxe des anniversaires :

Lemme 2.1. *Soit $\mathcal{L}_1, \mathcal{L}_2$ deux listes de taille L d'éléments de $\mathbb{F}_q^n$ uniformément tirés. Nous avons :*

$$\mathbb{E}\left(|\mathcal{L}_1 \cap \mathcal{L}_2|\right) = \frac{L^2}{q^n}$$

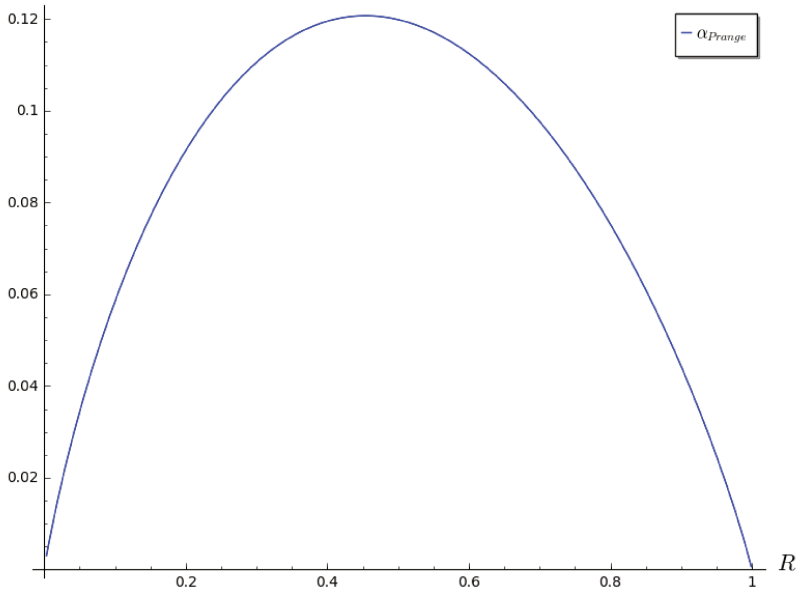


Figure 2.2 – Exposant asymptotique α_{Prange} de la complexité de l'algorithme de Prange en base 2 pour $q = 2$ et $w/n = \omega^-$ en fonction de R.

L'algorithme de Dumer et son analyse. Étant donné une matrice de parité $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ nous pouvons la décomposer (arbitrairement) en deux comme :

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 \end{pmatrix} \quad \text{où} \quad \mathbf{H}_1, \mathbf{H}_2 \in \mathbb{F}_q^{(n-k)\times n/2}.$$

L'idée étant que si nous trouvons une collision entre les deux listes :

$$\mathcal{L}_1 \stackrel{\triangle}{=} \{\mathbf{e}_1\mathbf{H}_1^\intercal : |\mathbf{e}_1| = w/2\} \quad \text{et} \quad \mathcal{L}_2 \stackrel{\triangle}{=} \{\mathbf{e}_2\mathbf{H}_2^\intercal + \mathbf{s} : |\mathbf{e}_2| = w/2\} \tag{2.10}$$

alors nous aurons une solution.

Le coût maintenant pour construire la collision de ces deux listes aléatoires ($\mathbf{H}$ est aléatoire) à l'aide de techniques classiques, de type table de hachage ou tri de liste, est majoré par une constante multipliant la taille de leur intersection. On en déduit alors facilement à l'aide des lemmes 2.1 et 1.2 la proposition suivante :

Proposition 2.4. *Soit $n \in \mathbb{N}$ et les paramètres w, k des fonctions de n. Toute instance $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$ du problème* SD(n, q, R, ω) *où $\omega \stackrel{\triangle}{=} w/n$ et $R \stackrel{\triangle}{=} k/n$ peut être résolu à un facteur polynomial près en temps moyen :*

$$L + \frac{L^2}{q^{n-k}} \quad \text{où} \quad L \stackrel{\triangle}{=} \sqrt{\binom{n}{w}(q-1)^w}$$

et renvoyant en moyenne $\max(L, 1)$ *solutions.*

Nous en déduisons la proposition suivante (en appliquant le lemme 1.2) comparant les complexités des algorithmes de Prange et de Dumer pour un décodage à distance w_{GV} avec $R \to 1$.

Proposition 2.1. *La complexité de l'algorithme de Prange pour résoudre* SD(n, q, R, ω) *pour $\omega = o(1)$ et $R = 1 - h_q(\omega)$ est de la forme :*

$$q^{n\cdot(1-R)(1+o(1))}$$

alors que celle de l'approche de Dumer est :

$$q^{n\cdot\frac{1-R}{2}(1+o(1))}.$$

Cette proposition avec le fait que l'algorithme de Dumer trouve toutes les solutions (possiblement en temps amorti 1 si w est bien choisi) est alors le point de départ des algorithmes ISD comme nous l'avons évoqué lors de l'introduction. Cependant avant de décrire ces algorithmes, présentons l'algorithme de Wagner [Wag02] qui peut-être vu dans notre cas comme une généralisation de l'approche de Dumer.

2.2.2 L'approche de Wagner

Il est naturel de chercher à étendre le procédé de Dumer en divisant non en deux mais en quatre voir en une puissance de deux arbitraire la matrice de parité pour obtenir une solution. Il s'agit essentiellement de l'approche introduite dans [Wag02] (puis étendu dans [MS09]) appliquée au décodage [CJ04] que nous allons décrire dans ce qui suit.

Nous commençons par la proposition donnant la complexité de l'algorithme de Wagner. Il y a comme nous le verrons deux stratégies et donc deux complexités selon que l'on souhaite des solutions en temps amorti en $O(1)$ ou une solution. De plus, l'analyse donnée est dans un cas où nous supposons l'instance $(\mathbf{H},\mathbf{s})$ du décodage uniformément distribuée.

Proposition 2.5 ([Wag02])**.** *Soit $n\in\mathbb{N}$ et les paramètres w,k des fonctions de n. Nous pouvons trouver pour une instance $(\mathbf{H},\mathbf{s})\in\mathbb{F}_q^{(n-k)\times n}\times\mathbb{F}_q^{n-k}$ uniformément distribuée du décodage* SD(n,q,R,ω) *où $\omega\triangleq w/n$ et $R\triangleq k/n$:*

1. *une solution en temps et espace moyen*

$$O\left(q^{\frac{n-k}{a+1}}\right)$$

où a est le plus grand entier tel que

$$\frac{1-R}{h_q(\omega)}\leqslant\frac{a+1}{2^a}.$$

2. $O\left(q^{\frac{n-k}{a}}\right)$ *solutions en temps amorti $O(1)$, ie : en temps*

$$O\left(q^{\frac{n-k}{a}}\right)$$

où a est le plus grand entier tel que

$$\frac{1-R}{h_q(\omega)}\leqslant\frac{a}{2^a}$$

L'algorithme de Wagner. La première étape de l'algorithme de Wagner consiste à séparer de façon arbitraire la matrice $\mathbf{H}$ en 2^a parties pour un paramètre $a\in\mathbb{N}$ qui sera dit *profondeur* :

$$\mathbf{H}=\begin{pmatrix}\mathbf{H}_1 & \cdots\mathbf{H}_{2^a}\end{pmatrix}\quad\text{avec}\quad\forall i\in[\![1,2^a]\!],\quad\mathbf{H}_i\in\mathbb{F}_q^{(n-k)\times\frac{n}{2^a}}.$$

Il est ensuite proposé de calculer les 2^a listes $\mathcal{L}_i$ de taille un certain paramètre L que nous fixerons plus tard :

$$\forall i\in[\![1,2^a]\!],\quad\mathcal{L}_i\subseteq\left\{\mathbf{e}\mathbf{H}_i^\intercal\ :\ \mathbf{e}\in\mathbb{F}_q^{n/2^a},|\mathbf{e}|=\frac{w}{2^a}\right\}\quad\text{et}\quad|\mathcal{L}_i|=L.\tag{2.11}$$

où par construction nous avons la contrainte :

$$L \leqslant \binom{n/2^a}{w/2^a}(q-1)^{w/2^a}. \tag{2.12}$$

La matrice $\mathbf{H}$ étant aléatoire nous supposerons que les vecteurs des listes $\mathcal{L}_i$ sont uniformément distribués. De plus, nous nous assurons par construction avoir accès pour $\mathbf{s}_i \in \mathcal{L}_i$ au vecteur $\mathbf{e}$ tel que $\mathbf{e}\mathbf{H}_i^\intercal = \mathbf{s}_i$.

Le temps pour construire les listes $\mathcal{L}_i$ (2.11) est en $O(L)$. Wagner propose ensuite de les *fusionner* de la façon suivante, pour tout $p \in \{1, 3, \cdots, 2^a - 3\}$ on crée la liste $\mathcal{L}_{p,p+1}$ comme :

$$\mathcal{L}_{p,p+1} \triangleq \{\mathbf{s}_p + \mathbf{s}_{p+1} \ : \ \mathbf{s}_i \in \mathcal{L}_i \text{ et } \textit{les derniers } \ell \textit{ symboles} \text{ de } \mathbf{s}_p + \mathbf{s}_{p+1} \text{ sont } 0\}$$

où par construction nous avons accès aux erreurs $\mathbf{e}_p$ et $\mathbf{e}_{p+1}$ donnant $\mathbf{s}_p$ et $\mathbf{s}_{p+1}$ avec $\mathbf{H}_p$ et $\mathbf{H}_{p+1}$. La liste $\mathcal{L}_{2^a-1,2^a}$ est elle crée à partir de $\mathcal{L}_{2^a-1}$ et $\mathcal{L}_{2^a}$ en fusionnant cette fois en fonction des ℓ derniers symboles de $\mathbf{s}$:

$$\mathcal{L}_{2^a-1,2^a} \triangleq \{\mathbf{s}_{2^a-1} + \mathbf{s}_{2^a} \ : \ \mathbf{s}_i \in \mathcal{L}_i \text{ et les derniers } \ell \text{ symboles de } \mathbf{s}_{2^a-1} + \mathbf{s}_{2^a} \text{ sont égaux à ceux de } \mathbf{s}\}.$$

Les vecteurs des listes $\mathcal{L}_i$ étant uniformément distribués sur $\mathbb{F}_q^{n-k}$, nous obtenons (lemme 2.1) en moyenne après fusion sur ℓ symboles 2^{a-1} nouvelles listes de taille :

$$\frac{L^2}{q^\ell}$$

pour un coût moyen total en temps et mémoire en $O\left(L + \frac{L^2}{q^\ell}\right)$. On recommence ensuite ce procédé $a-2$ fois en fusionnant sur ℓ nouveaux symboles à chaque étape. Wagner propose alors de choisir L de telle sorte *à ce qu'à chaque étape le coût de la fusion soit le même*, c'est à dire :

$$L = q^\ell. \tag{2.13}$$

Ceci implique d'après la contrainte (2.12) que le paramètre ℓ est choisi tel que :

$$q^\ell \leqslant \binom{n/2^a}{w/2^a}(q-1)^{w/2^a}. \tag{2.14}$$

Il est naturel de se demander si ce choix de paramètres impliquant un même coût moyen à chaque étape de l'algorithme est judicieux. Il est entre autres démontré dans [MS09] que oui, cette stratégie est essentiellement optimale.

Nous obtenons maintenant à l'étape $a-1$ deux listes $\mathcal{S}_1$ et $\mathcal{S}_2$ telles que :

$$\mathcal{S}_1 = \{\mathbf{s}_1 + \cdots + \mathbf{s}_{2^{a-1}} \ : \ \mathbf{s}_i \in \mathcal{L}_i \text{ et les } (a-1)\ell \text{ derniers symboles de } \mathbf{s}_1 + \cdots + \mathbf{s}_{2^{a-1}} \text{ sont égaux à } 0\}$$

$$\mathcal{S}_2 = \{\mathbf{s}_{2^{a-1}+1} + \cdots + \mathbf{s}_{2^a} \ : \ \mathbf{s}_i \in \mathcal{L}_i \text{ et les } (a-1)\ell \text{ derniers symboles de } \mathbf{s}_1 + \cdots + \mathbf{s}_{2^{a-1}} \text{ sont égaux à ceux de } \mathbf{s}\}$$

où en moyenne,

$$|\mathcal{S}_1| = |\mathcal{S}_2| = L = q^\ell$$

Il ne reste donc plus qu'à les fusionner en fonction des $(n-k)-(a-1)\ell$ premiers symboles de $\mathbf{s}$. Nous obtenons alors une liste de taille moyenne :

$$\frac{q^{2\ell}}{q^{(n-k)-(a-1)\ell}} = \frac{q^{\ell(a+1)}}{q^{(n-k)}} \tag{2.15}$$

en temps et espace moyen en $O\left(\frac{q^{\ell(a+1)}}{q^{(n-k)}}\right)$. La stratégie diverge ici selon que l'on souhaite obtenir une solution au problème ou des solutions en temps amorti $O(1)$.

Wagner pour obtenir une solution. Il suffit pour cela de choisir les paramètres de façon a avoir égalité dans (2.15) :

$$\ell = \frac{n-k}{a+1}. \tag{2.16}$$

Toutes les étapes ayant le même coût en $L = 2^\ell$ nous obtenons un algorithme qui trouve *une solution* en temps et espace moyen :

$$O\left(q^{\frac{n-k}{a+1}}\right). \tag{2.17}$$

En revanche nous devons faire attention au choix du paramètre a. Nous aimerions le faire croître mais ce dernier est contraint. En effet d'après (2.12),(2.13),(2.14) et (2.16) :

$$q^{\frac{n-k}{a+1}} \leqslant \binom{n/2^a}{w/2^a}(q-1)^{w/2^a}.$$

Ce qui donne comme contrainte asymptotique avec $n \to +\infty$ d'après le lemme 1.2 pour $\omega \stackrel{\triangle}{=} w/n$ et $R \stackrel{\triangle}{=} k/n$:

$$\frac{1-R}{a+1} \leqslant \frac{1}{2^a} h_q(\omega) \iff \frac{1-R}{h_q(\omega)} \leqslant \frac{a+1}{2^a}. \tag{2.18}$$

ce qui conclut le point 1 de la proposition 2.5.

Wagner pour obtenir des solutions en temps amorti $O(1)$. Si nous souhaitons obtenir des solutions en temps amorti 1 nous devons choisir le paramètre ℓ tel que :

$$\frac{q^{2\ell}}{q^{(n-k)-(a-1)\ell}} = \frac{q^{\ell(a+1)}}{q^{(n-k)}} = q^\ell$$

du fait que le nombre de solutions à la dernière fusion est donné par (2.15) et que le coût de l'algorithme était jusque-là en $O(L)$ où $L = q^\ell$. On en déduit alors facilement le deuxième point de la proposition 2.5.

Nous résumons dans la figure 2.3 l'algorithme de Wagner dans le cas où nous cherchons une solution pour $a = 2$.

Nous traçons dans la figure 2.4 l'exposant asymptotique de l'algorithme de Wagner pour trouver une solution au problème du décodage en fonction de $\omega \geqslant \omega^-$. Cette condition apparaît car dans notre analyse nous supposions $(\mathbf{H}, \mathbf{s})$ uniformément distribuée. Dans ce cas, une condition nécessaire pour que l'algorithme trouve en moyenne une solution est que cette dernière existe en moyenne, ce qui est équivalent à $\omega \geqslant \omega^-$.

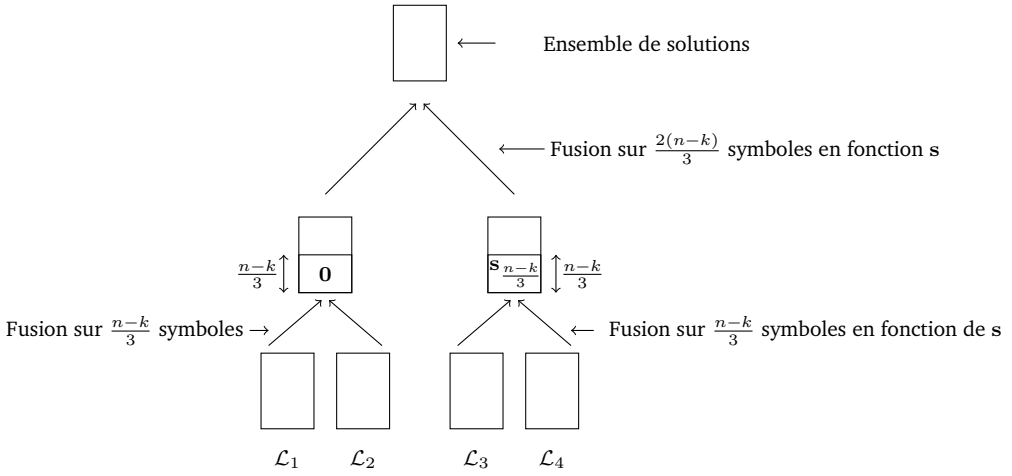


Figure 2.3 – Algorithme de Wagner de profondeur $a = 2$ appliqué au décodage.

Nous observons une discontinuité dans l'exposant. Cela provient du fait que la complexité est donnée par $q^{(n-k)/(a+1)}$ pour le plus grand entier a vérifiant (2.18). Nous avons cette condition car l'algorithme *décime* les solutions avec la profondeur a. En effet comme nous l'avons vu, les premières listes construites à partir d'erreurs $\mathbf{e} \in \mathbb{F}_q^{n/2^a}$ de poids $w/2^a$ sont de taille : $q^{(n-k)/(a+1)}$. Il est donc nécessaire que $q^{(n-k)/(a+1)} \leqslant \binom{n/2^a}{w/2^a}(q-1)^{w/2^a}$ ce qui essentiellement signifie d'après le développement asymptotique du lemme 1.2 que :

$$q^{(n-k)/(a+1)} \leqslant \left(\binom{n}{w}(q-1)^w\right)^{1/2^a}. \tag{2.19}$$

Or le nombre moyen de solutions au problème du décodage est donné par $\binom{n}{w}(q-1)^w/q^{n-k}$ qui grossit exponentiellement avec $w/n \leqslant (q-1)/q$. Nous voyons donc avec (2.19) que la profondeur de l'algorithme de Wagner augmente de façon discontinue avec le nombre typique de solutions du décodage considéré tout en étant rapidement limité.

Nous montrons cependant dans le prochain paragraphe comment atténuer ce phénomène de discontinuité.

Un algorithme de Wagner lissé. La proposition qui suit est une contribution de cette thèse que nous avons décrite dans [Bri+19].

Proposition 2.6. *Soit $n \in \mathbb{N}$ et les paramètres w, k des fonctions de de n. Nous pouvons trouver pour une instance $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$ uniformément distribuée du décodage* $\mathrm{SD}(n, q, R, \omega)$ *où $\omega \triangleq w/n$ et $R \triangleq k/n$:*

1. *une solution en temps et espace moyen*

$$O(q^\lambda) \quad \text{où} \quad \lambda \triangleq \frac{n-k}{a-1} - \frac{nh_q(\omega)}{(a-1)2^{a-1}}$$

si le plus petit entier a tel que

$$\frac{1-R}{h_q(\omega)} \geqslant \frac{a+1}{2^a}$$

est $\geqslant 2$.

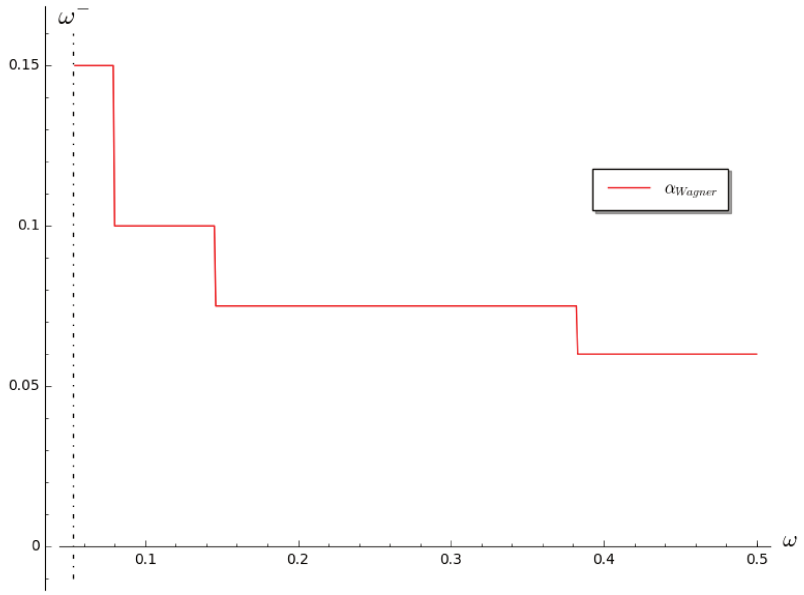


Figure 2.4 – Exposant asymptotique α_{Wagner} en base 2 de la complexité de l'algorithme de Wagner pour trouver une solution du problème de décodage avec $q = 2$ et $R = 0.7$ en fonction de $\omega = w/n$ fixé.

2. *$O(q^\lambda)$ solutions en temps amorti $O(1)$ où*

$$\lambda \stackrel{\triangle}{=} \frac{n-k}{a-2} - \frac{nh_q(\omega)}{(a-2)2^{a-1}}$$

si le plus petit entier a tel que

$$\frac{1-R}{h_q(\omega)} \geqslant \frac{a}{2^a}$$

est $\geqslant 2$.

Nous retrouvons bien les résultats de la proposition 2.5 quand $\frac{1-R}{h_q(\omega)} = \frac{a}{2^a}$.

Démonstration de la proposition 2.6.

Nous allons prouver le point 2, la démonstration du point 1 étant similaire. Soit a le plus petit entier tel que $\frac{1-R}{h_q(\omega)} \geqslant \frac{a}{2^a}$ que nous supposons $\geqslant 2$. Nous considérons l'algorithme de Wagner décrit dans la sous-section précédente avec une profondeur de a. En revanche, nous allons changer le nombre de symboles sur lesquels nous opérons les fusions. Supposons que nous construisons au départ de l'algorithme des listes de taille maximale :

$$\binom{n/2^a}{w/2^a}(q-1)^{w/2^a}.$$

Nous faisons alors une fusion sur m bits. Afin d'obtenir des listes de taille $O(q^\lambda)$ en temps $O(q^\lambda)$ nous choisissons m tel que :

$$\frac{\left(\binom{n/2^a}{w/2^a}(q-1)^{w/2^a}\right)^2}{q^m} = \tilde{O}\left(q^\lambda\right) \iff \frac{2n}{2^a}h_q(\omega) - m = \lambda. \tag{2.20}$$

Nous continuons ensuite les $(a-1)$ autres fusions de façon à obtenir à chaque étape q^λ solutions en temps amorti 1. Nous opérons donc les fusions sur λ symboles. Cependant,

à la fin de l'algorithme pour obtenir des solutions il nous faut avoir fusionné sur tous les symboles (au nombre de $n-k$). Ainsi m et λ doivent vérifier :

$$m + (a-1)\lambda = n - k. \tag{2.21}$$

En combinant les équations (2.20) et (2.21) nous obtenons finalement :

$$\lambda = \frac{n-k}{a-2} - \frac{nh_q(\omega)}{(a-2)2^{a-1}}.$$

Il est alors facile de vérifier que sous les conditions $\frac{1-R}{h_q(\omega)} \geqslant \frac{a}{2^a}$ et $a \geqslant 2$ que λ et m sont positifs ce qui conclut la preuve. □

Nous comparons dans la figure 2.5 les exposants asymptotiques de l'algorithme de Wagner non lissé et lissé pour trouver une solution au problème du décodage en fonction de $\omega \geqslant \omega^-$. Chaque discontinuité dans l'exposant de l'algorithme de Wagner non lissé provient d'un changement de profondeur. Comme nous pouvons le constater, notre proposition de lissage rend alors l'exposant affine par morceaux, en particulier ce dernier est le même que celui de l'algorithme originel en les poids relatifs où il y avait changement de profondeur.

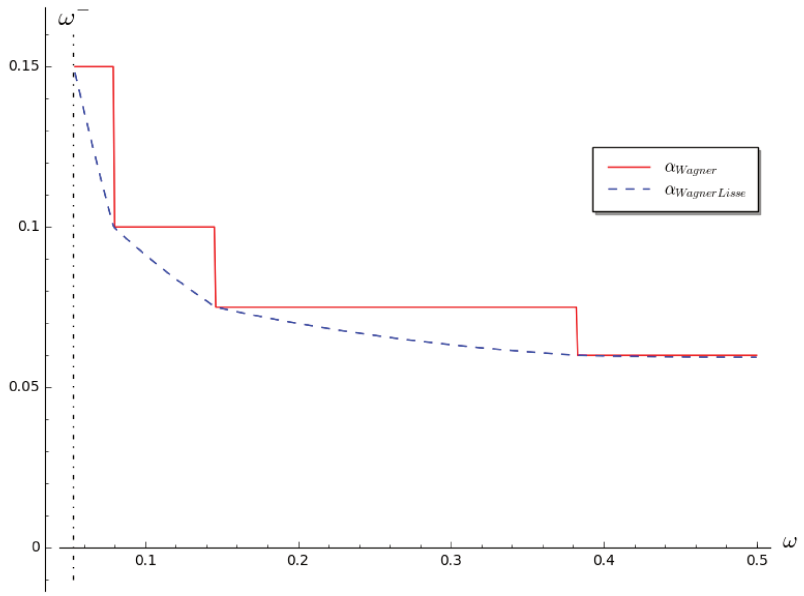


Figure 2.5 – Exposants asymptotiques α_{Wagner} et $\alpha_{WagnerLisse}$ de la complexité de l'algorithme en base 2 des algorithmes de Wagner non lissé et lissé pour trouver une solution du problème de décodage avec $q = 2$ et $R = 0.7$ en fonction de $\omega = w/n$ fixé.

2.3 Une approche combinée : les codes poinçonnés

Nous allons désormais présenter deux algorithmes de la famille des ISD [Dum91 ; Bec+12]. L'idée de ces algorithmes étant qu'ils combinent l'algorithme de Prange avec la résolution d'un petit problème de décodage pour des codes de rendement proche de 1 à l'aide de techniques comme décrites dans la section précédente. Stern [Ste88] est le premier à avoir eu l'idée de cette combinaison.

Le cadre général dans lequel s'inscrivent les ISD a été introduit dans [FS09], décrivons-le pour des matrices et vecteurs à coefficients dans $\mathbb{F}_q$. Fixons une instance $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$ du problème de décodage que nous cherchons à résoudre à distance w.

L'algorithme. Nous introduisons les deux paramètres :

$$\ell \in [\![0, n-k]\!] \quad \text{et} \quad p \in [\![0, \min(w, k+\ell)]\!].$$

Les algorithmes ISD se déroulent en 4 étapes :

1. *Tirage d'un ensemble contenant un ensemble d'information* : On tire aléatoirement $\mathcal{J} \subseteq [\![1, n]\!]$ de taille $k+\ell$. Si $\mathbf{H}_{\overline{\mathcal{J}}}$ n'est pas inversible, autrement dit si $\mathcal{J}$ ne contient pas d'ensemble d'information, on recommence.
   Soit $\mathbf{P}$ une matrice de permutation renvoyant les $k+\ell$ coordonnées de $\mathcal{J}$ sur $[\![n-k-\ell+1, n]\!]$.
2. *Algèbre linéaire* : On effectue une élimination Gaussienne *partielle* sur les $n-k-\ell$ premières lignes de $\mathbf{HP}$. Soit $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ la matrice inversible correspondant à cette opération. Nous obtenons $\mathbf{H}' \in \mathbb{F}_q^{(n-k-\ell)\times(k+\ell)}$ et $\mathbf{H}'' \in \mathbb{F}_q^{\ell\times(k+\ell)}$ telles que :
   $$\mathbf{SHP} = \begin{pmatrix} \mathbf{1}_{n-k-\ell} & \mathbf{H}' \\ \mathbf{0} & \mathbf{H}'' \end{pmatrix}. \tag{2.22}$$
   De plus, notons :
   $$\mathbf{s}\mathbf{S}^\intercal = (\mathbf{s}', \mathbf{s}'') \quad \text{avec} \quad \mathbf{s}' \in \mathbb{F}_q^{n-k-\ell} \quad \text{et} \quad \mathbf{s}'' \in \mathbb{F}_q^{\ell}. \tag{2.23}$$
3. *Décodage du code poinçonné* : On calcule un ensemble,
   $$\mathcal{S} \subseteq \left\{ \mathbf{e}'' \in \mathbb{F}_q^{k+\ell} : \mathbf{e}''\mathbf{H}''^\intercal = \mathbf{s}'' \text{ et } |\mathbf{e}''| = p \right\}. \tag{2.24}$$
4. *Test* : Pour tout vecteur $\mathbf{e}'' \in \mathcal{S}$ on teste si $\left|\mathbf{s}' - \mathbf{e}''\mathbf{H}''^\intercal\right| = w - p$. Si ce n'est pas le cas on retourne à l'étape 1 sinon on envoie le vecteur
   $$(\mathbf{s}' - \mathbf{e}''\mathbf{H}'^\intercal, \mathbf{e}'')\mathbf{P}^\intercal.$$

Remarquons que si $\mathcal{C}$ désigne le code de matrice de parité $\mathbf{H}$, alors le code poinçonné $\mathcal{C}_{\text{poinc}}(\mathcal{J})$ admet comme matrice de parité $\mathbf{H}''$ et dans ce cas l'étape 3 de l'algorithme consiste à décoder $\mathcal{C}_{\text{poinc}}(\mathcal{J})$ à distance p.

Correction de l'algorithme. Vérifions que le vecteur renvoyé $\mathbf{e} \triangleq (\mathbf{s}' - \mathbf{e}''\mathbf{H}'^\intercal, \mathbf{e}'')\mathbf{P}^\intercal$ est bien une solution. La matrice $\mathbf{P}$ étant une permutation l'étape 4 assure bien un poids w. Posons :

$$\mathbf{e}' \triangleq \mathbf{s}' - \mathbf{e}''\mathbf{H}'^\intercal \tag{2.25}$$

Nous avons,

$$\begin{aligned}
\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}^\intercal &\iff \mathbf{HP}(\mathbf{e}', \mathbf{e}'')^\intercal = \mathbf{s}^\intercal \\
&\iff \mathbf{SHP}(\mathbf{e}', \mathbf{e}'')^\intercal = \mathbf{S}\mathbf{s}^\intercal \\
&\iff \begin{pmatrix} \mathbf{1}_{n-k-\ell} & \mathbf{H}' \\ \mathbf{0} & \mathbf{H}'' \end{pmatrix} \begin{pmatrix} \mathbf{e}'^\intercal \\ \mathbf{e}''^\intercal \end{pmatrix} = \begin{pmatrix} \mathbf{s}'^\intercal \\ \mathbf{s}''^\intercal \end{pmatrix} \quad \text{(voir (2.22) et (2.23))} \\
&\iff \begin{cases} \mathbf{e}'^\intercal + \mathbf{H}'\mathbf{e}''^\intercal = \mathbf{s}'^\intercal \\ \mathbf{H}''\mathbf{e}''^\intercal = \mathbf{s}''^\intercal \end{cases}
\end{aligned}$$

Les deux équations sont bien vérifiées par définition de $\mathbf{e}'$ (2.25) et $\mathbf{e}'' \in \mathcal{S}$ (2.24).

Analyse de l'agorithme. Commençons par fixer la notation suivante dont on se servira dans toute la suite :

Notation 2. *Une quantité importante pour comprendre la complexité des* ISD *est la probabilité de succès de l'étape* 4*. Pour le couple de variables aléatoires* $(\mathbf{H}, \mathbf{s})$*, supposons que l'on dispose d'une solution de l'étape* 3*, un vecteur* $\mathbf{e}''$ *tel que* $\mathbf{e}''\mathbf{H}''^{\intercal} = \mathbf{s}''$ *et* $|\mathbf{e}''| = p$*. Soit* $\mathbf{e}' \triangleq \mathbf{s}' - \mathbf{e}''\mathbf{H}''^{\intercal}$*. On notera :*

$$\alpha_{p,\ell} \triangleq \mathbb{P}\left(|\mathbf{e}'| = w - p \mid |\mathbf{e}''| = p\right)$$

où la probabilité est calculée sur $\mathbf{H}, \mathbf{s}$ *et le tirage de l'ensemble* $\mathcal{J}$ *dans l'étape* 1*.*

Proposition 2.7. *Nous avons pour les paramètres* w, k, ℓ *et* p *:*

$$\alpha_{p,\ell} = \tilde{O}\left(\frac{\binom{n-k-\ell}{w-p}(q-1)^{w-p}}{\min\left(q^{n-k-\ell}, \binom{n}{w}(q-1)^w q^{-\ell}\right)}\right).$$

Démonstration de la proposition 2.7.

La preuve de cette proposition est un simple calcul combinatoire. Le numérateur correspond aux nombre de vecteurs $\mathbf{e}'$ de poids $w - p$. Le dénominateur est quant à lui le nombre de syndromes possible en entrée $\mathbf{s}$ divisé par q^ℓ car $\mathbf{s}'' \in \mathbb{F}_q^\ell$ est fixée, autrement dit : $\min\left(q^{n-k-\ell}, \binom{n}{w}(q-1)^w q^{-\ell}\right)$. Ce $\min$ provient du fait qu'en fonction de w et du nombre attendu de solutions $\binom{n}{w}(q-1)^w / q^{n-k-\ell}$ on tire $\mathbf{s} \hookleftarrow \mathbb{F}_q^{n-k-\ell}$ ou $\mathbf{s} = \mathbf{e}\mathbf{H}^{\intercal}$ avec $\mathbf{e} \hookleftarrow S_w$. □

La proposition qui suit donne la complexité des ISD. Cette dernière dépend alors de l'algorithme utilisé durant l'étape 3.

Proposition 2.2. *Supposons que l'on dispose d'un algorithme trouvant un ensemble de solutions* $\mathcal{S}$ *en temps moyen* T *du décodage à distance* p *d'un code aléatoire de longueur* $k + \ell$ *et de dimension* k*. Alors, le problème du décodage à distance* w *d'un code aléatoire de dimension* k *et de longueur* n*, pour les paramètres* w, k, ℓ *et* p*, peut être résolu en temps moyen :*

$$\tilde{O}\left(T \cdot \max\left(1, \frac{1}{|\mathcal{S}| \cdot \alpha_{p,\ell}}\right)\right).$$

Dans ce qui suit nous exposerons les propositions de Dumer [Dum91] et BJMM [Bec+12] pour résoudre l'étape 3 des algorithmes ISD. Nous n'avons pas vocation à être ici exhaustif. Ce choix de présentation est fait car il nous semble donner les grandes idées ayant fait avancer les techniques algorithmiques pour résoudre le problème du décodage. Néanmoins nous pouvons citer les algorithmes de Leon [Leo88] ou de Lee et Brickell [LB88] qui fait une recherche exhaustive dans le code poinçonné et ne gagne rien dans l'exposant de Prange. Nous ne parlerons pas non plus de l'algorithme de MMT [MMT11] qui est le premier à avoir utilisé la technique des représentations. Il y a aussi l'algorithme de Stern [Ste88] antérieur à celui de Dumer [Dum91] et lui étant proche.

Commençons cependant par une discussion sur la complexité des ISD pour résoudre $\mathrm{SD}(n, q, R, \omega)$ lorsque $\omega = o(1)$ ou $q \to +\infty$.

2.3.1 Les ISD pour $q \to +\infty$ et $\omega = o(1)$

Il a été démontré dans [CS16a] la proposition qui suit :

Proposition 2.8 ([CS16a])**.** *La complexité de tous les algorithmes* ISD *proposés, en particulier* [Pra62 ; Ste88 ; Dum91 ; MMT11 ; Bec+12 ; MO15 ; BM18] *pour résoudre le problème* $\mathrm{SD}(n, 2, R, \omega)$ *où :*

$$R \in]0, 1[\quad \textit{et} \quad \omega = o(1)$$

est donnée pour $n \to +\infty$ *par :*

$$2^{n \cdot (-\omega \log_2(1-R)(1+o(1)))}.$$

Autrement dit, tous les algorithmes ISD proposés n'ont pas amélioré la complexité de l'algorithme de Prange dans le cas où le poids de décodage est sous-linéaire.

De plus, il a de même été démontré dans [Can17] une proposition similaire : dans le cas où $q \to +\infty$, tous les ISD proposés ont la même complexité asymptotique que celle de l'algorithme de Prange.

Les différentes propositions d'algorithmes ISD furent faites pour des codes binaires. C'est pour cela que dans toute la suite de chapitre nous supposerons que :

$$q = 2.$$

2.3.2 L'algorithme ISD de Dumer [Dum91]

L'algorithme ISD de Dumer consiste exactement à utiliser l'approche que nous avons décrite dans §2.2.1 pour résoudre l'étape 3 des ISD. Fixons donc une instance du problème de décodage à distance $p \in [\![0, \min(w, k+\ell)]\!]$ que l'on souhaite résoudre $(\mathbf{H}'', \mathbf{s}'') \in \mathbb{F}_2^{\ell \times (k+\ell)} \times \mathbb{F}_2^{\ell}$. D'après la proposition 2.4 , l'algorithme de Dumer permet de trouver à un facteur polynomial près :

$$\frac{\binom{k+\ell}{p}}{2^\ell}$$

solutions en temps moyen :

$$\sqrt{\binom{k+\ell}{p}} + \frac{\binom{k+\ell}{p}}{2^\ell}.$$

On en déduit alors facilement à l'aide de la proposition 2.2 :

Proposition 2.9. *La complexité de l'*ISD *de Dumer est donnée pour tout $n \in \mathbb{N}$ et pour les paramètres w, k, ℓ, p des fonctions de n à un facteur polynomial près par :*

$$C_{\text{Dumer}} \triangleq \left(\sqrt{\binom{k+\ell}{p}} + \frac{\binom{k+\ell}{p}}{2^\ell} \right) \cdot \max\left(1, \frac{\min(2^{n-k}, \binom{n}{w})}{\binom{n-k-\ell}{w-p} \cdot \binom{k+\ell}{p}} \right) \tag{2.26}$$

La complexité C_{Dumer} de l'algorithme de Dumer est entre autres paramétrée par p et ℓ. Notre objectif dans ce qui suit est de la minimiser. Fixons pour cela en fonction des paramètres k, w, ℓ et p les quantités relatives :

$$R \triangleq \frac{k}{n} \quad ; \quad \omega \triangleq \frac{w}{n} \quad ; \quad \lambda \triangleq \frac{\ell}{n} \quad ; \quad \pi \triangleq \frac{p}{n} \tag{2.27}$$

Comme démontré dans [FS09] il est nécessaire pour optimiser la complexité d'avoir (à moins de cas extrêmes où l'égalité n'est pas possible) :

$$\sqrt{\binom{k+\ell}{p}} = \frac{\binom{k+\ell}{p}}{2^\ell} \iff 2^\ell = \sqrt{\binom{k+\ell}{p}} \tag{2.28}$$

L'optimisation de l'algorithme de Dumer fait donc qu'il trouve les solutions du décodage du code poinçonné en temps amorti polynomial comme nous l'avions évoqué lors de l'introduction. Cela donne asymptotiquement avec n en terme d'exposant en base 2 d'après le lemme 1.2 :

$$\lambda = \frac{R+\lambda}{2} h_2\left(\frac{\pi}{R+\ell}\right) \iff \pi = (R+\lambda) h_2^{-1}\left(\frac{2\lambda}{R+\lambda}\right) \tag{2.29}$$

Notons π_λ le paramètre π vérifiant l'égalité qui précède. Nous pouvons maintenant vérifier avec les équations (2.26) et (2.29) que pour $n \to +\infty$:

$$\log_2(C_{\text{Dumer}}) = f(\lambda)(1+o(1))$$

où :

$$f(\lambda) \stackrel{\triangle}{=} \big(\lambda + \max\left(0, \min\left(1-R, h_2(\omega)\right)\right) - (1 - R - \lambda) h_2\big(\frac{\omega - \pi_\lambda}{1-R-\lambda}\big) - 2\lambda\big).$$

La fonction $\lambda \mapsto f(\lambda)$ est alors unimodale et sa minimisation s'obtient facilement avec par exemple la méthode du nombre d'or (voir https://en.wikipedia.org/wiki/Golden-section_search). Intuitivement, avec $\lambda = \ell/n$ grossissant nous améliorons la probabilité de succès d'une itération de l'ISD et donc nous diminuons le nombre de répétitions mais nous augmentons le coût d'une itération.

Nous comparons maintenant après minimisation les complexités asymptotiques des algorithmes de Prange et de Dumer en fonction de ω pour $R = 1/2$ dans la figure 2.6 et en fonction de R pour $\omega = \omega^-$ dans la figure 2.7.

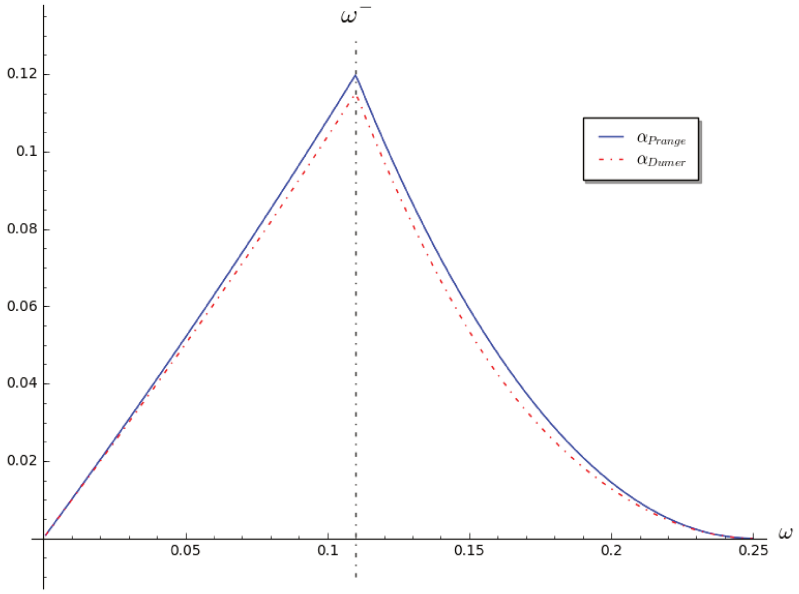


Figure 2.6 – Exposants asymptotiques α_{Prange} et α_{Dumer} de la complexité de l'algorithme de Prange et Dumer pour $R = 1/2$ en fonction de $\omega = w/n$ fixé.

Comme nous pouvons l'observer l'algorithme de Dumer est exponentiellement meilleur pour des poids relatifs autour de ω^-, zone où le décodage générique est le plus difficile. En revanche, ce gain exponentiel tend à s'amoindrir pour finalement disparaître avec ω s'écartant de ω^-.

2.3.3 L'algorithme BJMM [Bec+12]

Nous allons désormais présenter l'amélioration de [Bec+12]. Cet algorithme applique au décodage des méthodes introduites pour résoudre le problème du sac à dos : les

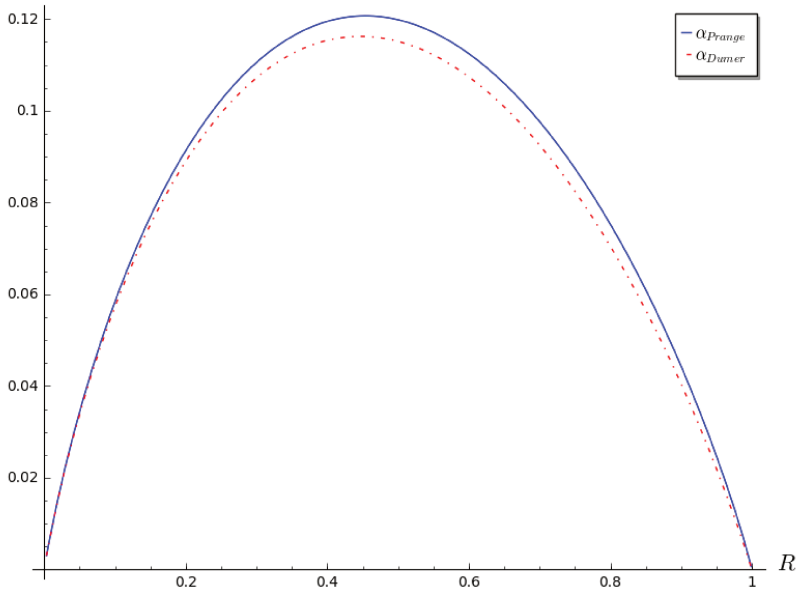


Figure 2.7 – Exposants asymptotiques α_{Prange} et α_{Dumer} de la complexité de l'algorithme de Prange et Dumer pour $w/n = \omega^-$ en fonction de R.

techniques dites des représentations introduite par [HJ10] et de $1+1=0$ [BCJ11]. Notons néanmoins que le premier algorithme appliquant la technique des représentations au décodage est celui de [MMT11].

Nous verrons dans ce qui suit :

Proposition 2.10 ([Bec+12])**.** *Soit $n, k, \ell, p, r, \varepsilon \in \mathbb{N}$. Le problème du décodage d'instance $(\mathbf{H}'', \mathbf{s}'') \in \mathbb{F}_2^{\ell \times (k+\ell)} \times \mathbb{F}_2^{\ell}$ à distance p peut-être résolu à un facteur polynomial près en temps :*

$$\sqrt{L} + \frac{L}{2^r} + \frac{1}{2^r}\frac{L^2}{2^\ell}$$

avec $\mu \frac{1}{2^r}\frac{L^2}{2^\ell}$ le nombre de solutions trouvées où :

$$L \triangleq \binom{k+\ell}{p/2+\varepsilon} \quad \text{et} \quad \mu \triangleq \frac{\binom{p/2+\varepsilon}{\varepsilon}\binom{k+\ell-p/2-\varepsilon}{p/2}}{\binom{k+\ell}{p/2+\varepsilon}}$$

et $r \in \mathbb{N}$ choisi tel que :

$$\frac{\binom{p}{p/2}\binom{k+\ell-p}{\varepsilon}}{2^r} = 1.$$

Considérons une instance $(\mathbf{H}'', \mathbf{s}'') \in \mathbb{F}_2^{\ell \times (k+\ell)} \times \mathbb{F}_2^{\ell}$ du problème de décodage à distance p que l'on souhaite résoudre. Dans l'algorithme de Dumer nous avions coupé la matrice $\mathbf{H}''$ en deux et cherché une collision entre les deux listes :

$$\mathcal{L}_1 = \{\mathbf{e}_1 \mathbf{H}_1^\intercal \ : \ \mathbf{e}_1 \in \mathbb{F}_2^{\frac{k+\ell}{2}} \text{ et } |\mathbf{e}_1| = p/2\},$$

$$\mathcal{L}_2 = \{\mathbf{e}_2 \mathbf{H}_2^\intercal + \mathbf{s}'' \ : \ \mathbf{e}_2 \in \mathbb{F}_2^{\frac{k+\ell}{2}} \text{ et } |\mathbf{e}_2| = p/2\}.$$

Une solution est alors facilement déduite d'une collision trouvée à partir de $\mathbf{e}_1$ et $\mathbf{e}_2$ comme :

$$\mathbf{e} = \left(\mathbf{e}_1, \mathbf{0}_{\frac{k+\ell}{2}}\right) + \left(\mathbf{0}_{\frac{k+\ell}{2}}, \mathbf{e}_2\right).$$

L'algorithme de Wagner quant à lui itère ce procédé de décomposition en deux tout en opérant étape après étape des collisions sur des sous-ensembles de positions ce qui a pour effet de décimer les solutions.

L'idée de la technique des représentations est de ne plus chercher à écrire une solution obtenue par découpage en deux. Un même vecteur de poids p peut se représenter de multiples façons comme somme de deux vecteurs de même poids $p/2+\varepsilon$. Il y a alors beaucoup plus de manières d'écrire un mot de poids p comme une somme de mots de poids $p/2+\varepsilon$ (on parle de représentations) comparé aux $\binom{(k+\ell)/2}{p/2}$ façons lors du découpage en deux dans l'algorithme de Dumer. Nous illustrons la situation dans la figure 2.8. Nous augmentons donc artificiellement le nombre de vecteurs à partir desquels nous pouvons obtenir des solutions. Le rationnel étant que dans l'algorithme de Dumer nous choisissions p de façon à obtenir des solutions en temps amorti polynomial, ce qui correspondait à :

$$2^\ell = \sqrt{\binom{k+\ell}{p}} = \tilde{O}\left(\binom{(k+\ell)/2}{p/2}\right).$$

Or comme nous en avions discuté lors de l'introduction nous cherchons à diminuer p le plus possible. Avec la technique des représentations nous aurons naturellement un terme plus gros que $\binom{(k+\ell)/2}{p/2}$ de vecteurs construits et nous pourrons alors diminuer p par rapport à Dumer et donc gagner dans l'exposant.

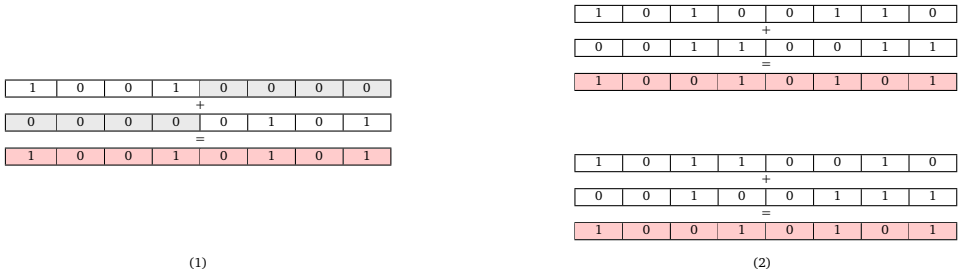


Figure 2.8 – Le même vecteur (1) écrit par découpage en deux et (2) utilisant les représentations.

Le terme ε est ici ajouté car il permet de contrôler la probabilité que la somme de deux mots de poids $p/2+\varepsilon$ soit de poids p tout en pouvant augmenter ou diminuer le nombre de représentations. Cette technique consistant à ajouter ε dans le poids est dite de $1+1=0$.

Définissons maintenant formellement les représentations binaires d'un vecteur.

Définition 2.3. *Soient $\mathbf{e} \in \mathbb{F}_2^n$ de poids $p \in [\![0,n]\!]$ et $\varepsilon \in \mathbb{N}$. On appelle $(p/2+\varepsilon)$-représentation de $\mathbf{e}$ tout couple $(\mathbf{e}_1,\mathbf{e}_2) \in \mathbb{F}_2^n$ tel que $|\mathbf{e}_1| = |\mathbf{e}_2| = p/2+\varepsilon$ et :*

$$\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2.$$

Rappelons que dans notre contexte nous cherchons à calculer un sous-ensemble de :

$$\left\{\mathbf{e} \ : \ \mathbf{e}\mathbf{H}''^{\intercal} = \mathbf{s}'' \text{ et } |\mathbf{e}| = p\right\}.$$

Avec la technique des représentations nous pouvons obtenir cette liste grâce à la collision des deux listes :

$$\mathcal{L}_1 = \{\mathbf{e}_1\mathbf{H}''^{\intercal} \ : \ \mathbf{e}_1 \in \mathbb{F}_2^{k+\ell} \text{ et } |\mathbf{e}_1| = p/2+\varepsilon\}.$$

$$\mathcal{L}_2 = \{\mathbf{e}_2\mathbf{H}''^{\intercal} + \mathbf{s}'' \ : \ \mathbf{e}_2 \in \mathbb{F}_2^{k+\ell} \text{ et } |\mathbf{e}_2| = p/2+\varepsilon\}.$$

Cette collision soulève alors deux points :

1. *De multiples représentants* : toute solution $\mathbf{e}$ de poids p de $\mathbf{e}\mathbf{H}''^{\mathsf{T}} = \mathbf{s}$ admet
$$\binom{p}{p/2}\binom{k+\ell-p}{\varepsilon} \tag{2.30}$$
$(p/2+\varepsilon)$-représentations.
2. *Des solutions de poids* $\neq p$: Lorsque nous extrayons à partir d'une collision les vecteurs $\mathbf{e}_1$ et $\mathbf{e}_2$ leur somme n'est pas nécessairement de poids p.

Concernant le point 1 il serait inutile et contre-productif de calculer pour une même solution toutes ses représentations. Il nous faut donc les éliminer et construire deux sous-listes de $\mathcal{L}_1$ et $\mathcal{L}_2$. L'idée est alors d'utiliser la même technique que dans l'algorithme de Wagner en les calculant à partir de collisions sur un petit sous-ensemble de positions. On décime alors les représentations mais pas les solutions.

Pour le point 2, du fait que $1+1=0$ il est possible que deux mots de poids $p/2+\varepsilon$ donnent un mot de poids p. La proportion de mots que l'on conservera est alors donnée par la probabilité de cet évènement. C'est l'objet du lemme qui suit.

Lemme 2.2. *Soient* $\mathbf{e}_1$ *et* $\mathbf{e}_2$ *uniformément distribués parmi les mots de* $\mathbb{F}_2^{k+\ell}$ *de poids* $p/2+\varepsilon$. *Nous avons :*
$$\mathbb{P}\left(|\mathbf{e}_1+\mathbf{e}_2|=p\right) = \frac{\binom{p/2+\varepsilon}{\varepsilon}\binom{k+\ell-p/2-\varepsilon}{p/2}}{\binom{k+\ell}{p/2+\varepsilon}} \tag{2.31}$$

Décrivons maintenant l'algorithme utilisant les techniques des représentations et $1+1=0$ pour construire une liste de solutions.

L'algorithme. Nous commençons par nous donner un nouveau paramètre :
$$r \in [\![0,\ell]\!] \tag{2.32}$$
et décomposer le syndrome $\mathbf{s}'' \in \mathbb{F}_2^{\ell}$ et la matrice $\mathbf{H}'' \in \mathbb{F}_2^{\ell\times(k+\ell)}$ comme :
$$\mathbf{s}'' = (\mathbf{s}_1,\mathbf{s}_2) \quad \text{où} \quad \mathbf{s}_1 \in \mathbb{F}_2^{\ell-r} \quad \text{et} \quad \mathbf{s}_2 \in \mathbb{F}_2^{r}$$
$$\mathbf{H}'' = \begin{pmatrix}\mathbf{H}_1\\ \mathbf{H}_2\end{pmatrix} \quad \text{où} \quad \mathbf{H}_1 \in \mathbb{F}_2^{(\ell-r)\times(k+\ell)} \quad \text{et} \quad \mathbf{H}_2 \in \mathbb{F}_2^{r\times(k+\ell)}.$$
Nous décomposons maintenant en deux la matrice $\mathbf{H}_2$:
$$\mathbf{H}_2 = \begin{pmatrix}\mathbf{H}_{2,1} & \mathbf{H}_{2,2}\end{pmatrix}.$$

1. *Collisions à la Dumer* : On calcule les quatre listes pour $i \in \{1,2\}$,
$$\mathcal{U}_{1,i} \triangleq \left\{\mathbf{e}_{1,i}\mathbf{H}_{2,i}^{\mathsf{T}} : |\mathbf{e}_{1,i}| = p/4+\varepsilon/2\right\}, \tag{2.33}$$
$$\mathcal{U}_{2,i} \triangleq \left\{\mathbf{e}_{2,i}\mathbf{H}_{2,2}^{\mathsf{T}} : |\mathbf{e}_{2,i}| = p/4+\varepsilon/2\right\}. \tag{2.34}$$
On effectue ensuite les collisions de $\mathcal{U}_{1,1},\mathcal{U}_{1,2}$ et $\mathcal{U}_{2,1},\mathcal{U}_{2,2}$ dont l'on en déduit :
$$\mathcal{U}_1 \triangleq \left\{\mathbf{e}_1\mathbf{H}''^{\mathsf{T}} : \mathbf{e}_1 \in \mathbb{F}_2^{k+\ell}, |\mathbf{e}_1| = p/2+\varepsilon \text{ et } \mathbf{e}_1\mathbf{H}_2^{\mathsf{T}} = \mathbf{0}_r\right\}, \tag{2.35}$$
$$\mathcal{U}_2 \triangleq \left\{\mathbf{s}'' + \mathbf{e}_2\mathbf{H}''^{\mathsf{T}} : \mathbf{e}_2 \in \mathbb{F}_2^{k+\ell}, |\mathbf{e}_2| = p/2 \text{ et } \mathbf{e}_2\mathbf{H}_2^{\mathsf{T}} = \mathbf{s}_2\right\}. \tag{2.36}$$
2. *Collisions avec représentations et* $1+1=0$: On en déduit alors de la collision des listes $\mathcal{U}_1$ et $\mathcal{U}_2$ des vecteurs $\mathbf{e}_1$ et $\mathbf{e}_2$ tels que
$$(\mathbf{e}_1+\mathbf{e}_2)\mathbf{H}''^{\mathsf{T}} = \mathbf{s}''$$
dont on ne conserve que ceux tels que $|\mathbf{e}_1+\mathbf{e}_2| = p$.

Analyse de l'algorithme. Notons pour ce qui suit :

$$L \triangleq \binom{k+\ell}{p/2+\varepsilon}$$

L'étape 1 construit, à partir des listes $\mathcal{U}_{1,i}, \mathcal{U}_{2,i}$ de taille $\sqrt{L}$, en moyenne les listes $\mathcal{U}_1$ et $\mathcal{U}_2$ de taille $L/2^r$ en temps :

$$\sqrt{L} + \frac{L}{2^r}$$

Dans l'étape 2 nous opérons une collision selon les $\ell - r$ premiers bits de $\mathbf{s}$. Malheureusement nous pouvons avoir des doublons dans les solutions. En effet, soit $(\mathbf{e}_1, \mathbf{e}_2)$ une représentation d'une solution $\mathbf{e} \in \mathbb{F}_2^{k+\ell}$ de poids p de $\mathbf{e}\mathbf{H}''^{\intercal} = \mathbf{s}''$, *ie :* $\mathbf{e} = \mathbf{e}_1 + \mathbf{e}_2$ où $|\mathbf{e}_1| = |\mathbf{e}_2| = p/2 + \varepsilon$. Une telle représentation sera alors par construction obtenue à partir de $\mathcal{U}_1$ et $\mathcal{U}_2$ si et seulement si :

$$\mathbf{e}_1 \mathbf{H}_2^{\intercal} = \mathbf{0}_r.$$

La matrice $\mathbf{H}_2$ étant uniformément distribuée car obtenue à partir de $\mathbf{H}$, cet évènement se produit avec probabilité :

$$\frac{1}{2^r}.$$

Nous pouvons donc conserver en moyenne une représentation par solution en choisissant r tel que :

$$\frac{\binom{p}{p/2}\binom{k+\ell-p}{\varepsilon}}{2^r} = 1. \tag{2.37}$$

Avec un tel choix de r nous obtenons donc après collision de $\mathcal{U}_1$ et $\mathcal{U}_2$ lors de l'étape 2 un ensemble de taille moyenne :

$$\left(\frac{L}{2^r}\right)^2 \cdot \frac{1}{2^{\ell-r}} = \frac{1}{2^r} \cdot \frac{L^2}{2^\ell}.$$

Or on ne conserve ensuite que les mots de poids p, nous obtenons donc à l'aide du lemme 2.2 en moyenne une liste de solutions de taille :

$$\mu \frac{1}{2^r} \frac{L^2}{2^\ell} \quad \text{où} \quad \mu \triangleq \frac{\binom{p/2+\varepsilon}{\varepsilon}\binom{k+\ell-p/2-\varepsilon}{p/2}}{\binom{k+\ell}{p/2+\varepsilon}}$$

en temps :

$$\sqrt{L} + \frac{L}{2^r} + \frac{1}{2^r} \cdot \frac{L^2}{2^\ell}$$

et nous retrouvons la proposition 2.10.

L'algorithme que nous venons de présenter est dit à un niveau du fait que les deux listes $\mathcal{U}_1$ (2.35) et $\mathcal{U}_2$ (2.36) sont calculées à partir des 4 listes (2.34),(2.34) avec la technique du paradoxe des anniversaires et donc en partant d'erreurs à supports disjoints. Nous pouvons dès lors augmenter la profondeur de l'algorithme en calculant chacune des listes $\mathcal{U}_1$ et $\mathcal{U}_2$ avec la technique que nous venons de décrire. Dans BJMM [Bec+12] il est alors proposé d'utiliser justement cet algorithme à deux niveaux. Il suffit alors de reprendre le raisonnement que nous venons de faire et la proposition 2.7 pour en déduire :

Proposition 2.11 ([Bec+12])**.** *Soient $n \in \mathbb{N}$ et les paramètres $w, k, \ell, p, \varepsilon_1$ et ε_2 des fonctions de n et r_1, r_2 choisis tels que :*

$$\frac{\binom{p/4+\varepsilon_2}{\frac{1}{2}(p/4-\varepsilon_1)+\varepsilon_2}\binom{k+\ell-p/4-\varepsilon_2}{\frac{1}{2}(p/4+\varepsilon_1)}}{2^{r_2}} = 1 \quad \textit{et} \quad \frac{\binom{p}{p/2}\binom{k+\ell-p}{\varepsilon}}{2^{r_1}} = 1 \tag{2.38}$$

La complexité de l'algorithme BJMM *est donnée à une facteur polynomial près par :*

$$C_{\text{BJMM}} \triangleq \left(\sqrt{L_2} + \frac{1}{2^{r_2}}\frac{L_2}{2^{r_1}} + \mu_2^2 \frac{L_2^4}{2^{r_1+2r_2+\ell}}\right) \cdot \max\left(1, 2^{r_1+2r_2}\frac{\min(2^{n-k}, \binom{n}{w})}{\binom{n-k-\ell}{w-p}\cdot \mu_1 \cdot \mu_2^2 \cdot L_2^4}\right) \quad (2.39)$$

où :

$$\mu_1 \triangleq \frac{\binom{p/2+\varepsilon}{\varepsilon}\binom{k+\ell-p/2-\varepsilon}{p/2}}{\binom{k+\ell}{p/2+\varepsilon}}, \quad \mu_2 \triangleq \frac{\binom{p/4+\varepsilon_2}{\varepsilon_2-\varepsilon_1/2}\binom{k+\ell-p/4-\varepsilon_2}{p/2+\varepsilon_1/2}}{\binom{k+\ell}{p/4+\varepsilon_1}}$$

$$L_2 \triangleq \sqrt{\binom{k+\ell}{p/4+\varepsilon_1}}.$$

Il y a donc quatre paramètres à optimiser dans l'algorithme de BJMM : ℓ, p, ε_1 et ε_2. Nous comparons maintenant après minimisation obtenue avec le logiciel Cawof [1] les complexités asymptotiques des algorithmes de Prange, Dumer et BJMM en fonction de ω pour $R = 1/2$ dans la figure 2.9 et en fonction de R pour $\omega = \omega^-$ dans la figure 2.10.

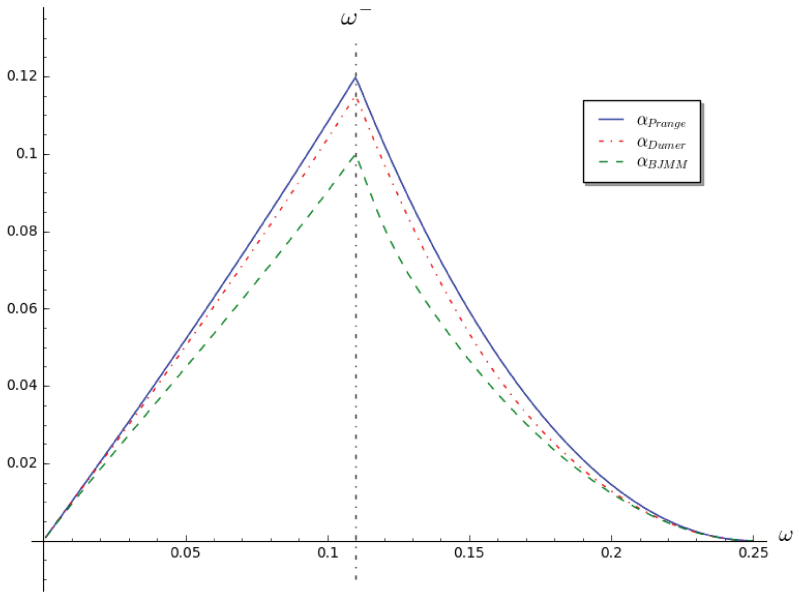


Figure 2.9 – Exposants asymptotiques $\alpha_{Prange}, \alpha_{Dumer}$ et α_{BJMM} de la complexité des algorithmes de Prange, Dumer et BJMM pour $R = 1/2$ en fonction de $\omega = w/n$ fixé.

Le tableau 2.1 compare les exposants asymptotiques des trois algorithmes pour $R = 1/2$ et $\omega = \omega^-$, poids relatif où les exposants sont les plus élevés.

1. https://gforge.inria.fr/scm/?group_id=9344

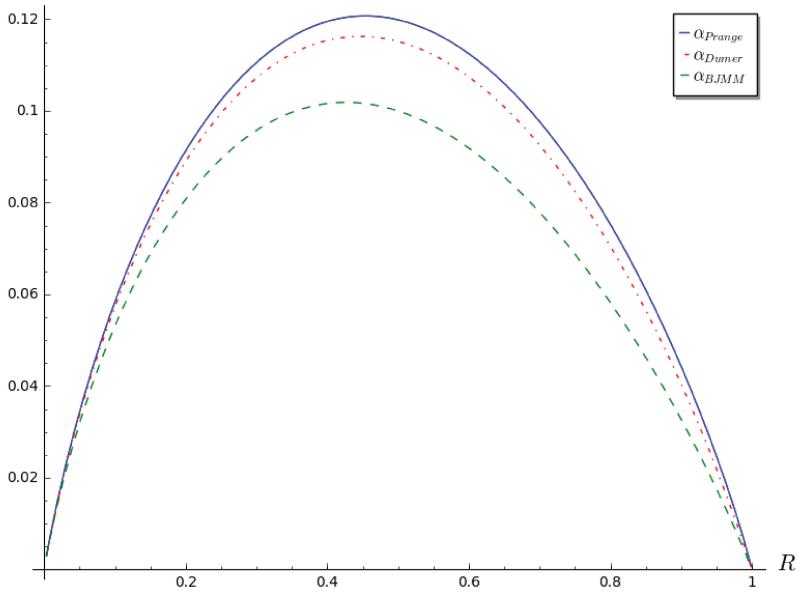


Figure 2.10 – Exposants asymptotiques $\alpha_{Prange}, \alpha_{Dumer}$ et α_{BJMM} de la complexité des algorithmes de Prange, Dumer et BJMM pour $w/n = \omega^-$ en fonction de R.

Table 2.1 – Exposants asymptotiques des algorithmes pour $R = 1/2$ et $\omega = \omega^-$.

Algorithme	Exposant
Prange	0.120
Dumer	0.115
BJMM	0.100

2.4 Les algorithmes par ensemble d'information pour résoudre DOOM

Nous nous intéressons désormais au problème du décodage où au lieu de un syndrome il nous en est donné N avec comme objectif de décoder l'un d'entre eux. Il s'agit exactement du problème DOOM que nous avons introduit dans §1.3.2. Ce problème fut considéré pour la première fois lors d'une attaque non publiée de Bleichenbacher contre le schéma de signature CFS [CFS01]. Bleichenbauer proposa pour résoudre DOOM d'utiliser le cadre des ISD. Les travaux de [FS09 ; Sen11a] généralisèrent ensuite cette approche algorithmique. Nous présentons dans ce qui suit l'approche de [Sen11a] s'inspirant de l'algorithme de Dumer [Dum91].

Considérons pour $N \in \mathbb{N}$ une instance $(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N) \in \mathbb{F}_2^{(n-k)\times n} \times \mathbb{F}_2^{n-k} \times \cdots \times \mathbb{F}_2^{n-k}$ du problème DOOM que l'on souhaite résoudre à distance w. Donnons-nous les matrices $\mathbf{H}'' \in \mathbb{F}_2^{\ell\times(k+\ell)}$ et $\mathbf{S} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ obtenues lors de l'étape 2 des ISD. Nous écrivons alors :

$$\forall i \in [\![1, N]\!], \quad \mathbf{s}_i\mathbf{S}^\intercal = (\mathbf{s}'_i, \mathbf{s}''_i) \quad \text{où} \quad \mathbf{s}'_i \in \mathbb{F}_2^{n-k-\ell} \quad \text{et} \quad \mathbf{s}''_i \in \mathbb{F}_2^{\ell}.$$

Il est proposé dans [Sen11a] de trouver tous les vecteurs $\mathbf{e}'' \in \mathbb{F}_2^{k+\ell}$ tels que :

$$\exists i \in [\![1, N]\!], \quad \mathbf{e}''\mathbf{H}''^\intercal = \mathbf{s}''_i \quad \text{et} \quad |\mathbf{e}''| = p. \tag{2.40}$$

Nous pouvons alors supposer que

$$N \leqslant \min\left(2^{\ell}, \binom{k+\ell}{p}\right) \tag{2.41}$$

de façon à nous assurer qu'il n'y a pas en moyenne de redondances inutiles dans les syndromes $\mathbf{s}''_i$ et que ces derniers peuvent tous s'obtenir d'une erreur de poids p. Une fois les

$$N \cdot \frac{\binom{k+\ell}{p}}{2^{\ell}}$$

solutions calculées on teste les vecteurs en fonction des $\mathbf{s}_i$ comme lors de la dernière étape des ISD.

Rappelons maintenant qu'une solution $\mathbf{e}''$ de (2.40) donne une solution de poids w avec probabilité donnée à un facteur polynomial près par :

$$\frac{\binom{n-k-\ell}{w-p}}{\min\left(2^{n-k-\ell}, 2^{-\ell}\right)}.$$

Nous testons alors que cette erreur donne une solution pour l'un des N syndromes $\mathbf{s}_i$. La probabilité de succès de l'algorithme est donc donnée à un facteur polynomial près par :

$$\min\left(1, N \cdot \frac{\binom{k+\ell}{p}}{2^{\ell}} \frac{\binom{n-k-\ell}{w-p}}{\min\left(2^{n-k-\ell}, \binom{n}{w}2^{-\ell}\right)}\right) \tag{2.42}$$

Le problème étant maintenant de trouver un moyen de calculer toutes les solutions de poids p de $\mathbf{e}\mathbf{H}''^{\mathsf{T}} = \mathbf{s}_i$ pour un certain i. Il est proposé dans [Sen11a] de s'inspirer de l'approche de Dumer.

Nous commençons par décomposer (pas forcément en deux parties égales) la matrice $\mathbf{H}'' \in \mathbb{F}_2^{\ell \times (k+\ell)}$:

$$\mathbf{H}'' = \begin{pmatrix} \mathbf{H}_1 & \mathbf{H}_2 \end{pmatrix} \quad \text{où} \quad \mathbf{H}_1 \in \mathbb{F}_2^{\ell \times n_1} \quad \text{et } \mathbf{H}_2 \in \mathbb{F}_2^{\ell \times n_2} \quad \text{avec} \quad n_1 + n_2 = k + \ell.$$

On construit alors les deux listes :

$$\mathcal{L}_1 \stackrel{\triangle}{=} \left\{ \mathbf{e}_1 \mathbf{H}_1^{\mathsf{T}} \quad : \quad \mathbf{e}_1 \in \mathbb{F}_2^{n_1} \quad \text{et} \quad |\mathbf{e}_1| = p_1 \right\},$$

$$\mathcal{L}_2 \stackrel{\triangle}{=} \left\{ \mathbf{s}_i + \mathbf{e}_2 \mathbf{H}_2^{\mathsf{T}} \quad : \quad \mathbf{e}_2 \in \mathbb{F}_2^{n_2}, \quad |\mathbf{e}_2| = p_2 \quad \text{et} \quad 1 \leqslant i \leqslant N \right\}.$$

où les paramètres p_1 et p_2 vérifient :

$$p_1 + p_2 = p.$$

On va alors opérer une collision entre ces dernières. La liste de solutions obtenues est alors de taille $\binom{n_1}{p_1} \cdot N\binom{n_2}{p_2}/2^{\ell}$. Il est alors proposé dans [Sen11a] pour obtenir toutes les solutions au problème de choisir les paramètres p_1, n_1, p_2 et n_2 comme :

$$\binom{n_1}{p_1} = N\binom{n_2}{p_2} \quad \text{et} \quad \frac{p_1}{n_1} = \frac{p_2}{n_2}.$$

En effet, d'après le lemme 1.2 que dans ce cas le nombre de solutions est :

$$\frac{\binom{n_1}{p_1} \cdot N\binom{n_2}{p_2}}{2^{\ell}} = \tilde{O}\left(\frac{N \cdot \binom{k+\ell}{p}}{2^{\ell}}\right).$$

Le temps de construction de la collision de ces deux listes est donc donné à un facteur polynomial près par :

$$\sqrt{N \cdot \binom{k+\ell}{p}} + \frac{N \cdot \binom{k+\ell}{p}}{2^\ell}. \tag{2.43}$$

Il est ensuite proposé dans [Sen11a] de choisir ℓ et p tels que :

$$\frac{N \cdot \binom{k+\ell}{p}}{2^\ell} = \sqrt{N \cdot \binom{k+\ell}{p}} \iff 2^\ell = \sqrt{N \cdot \binom{k+\ell}{p}}. \tag{2.44}$$

Le coût de l'algorithme pour trouver une solution est donc donné par combinaison des équations (2.42),(2.43) et (2.44) pour N vérifiant (2.41) à un facteur polynomial près par :

$$\sqrt{N \cdot \binom{k+\ell}{p}} \max\left(1, \frac{\min\left(2^{n-k}, \binom{n}{w}\right)}{N \cdot \binom{k+\ell}{p}\binom{n-k-\ell}{w-p}}\right).$$

Si nous comparons avec la complexité de l'algorithme de Dumer §2.3.2 pour résoudre le problème du décodage il semble que nous gagnions un facteur qui est à peu près de l'ordre de $\sqrt{N}$. Dans la pratique l'équilibre des paramètres n'est pas le même du fait des équations (2.44) et (2.28) (voir §2.3.2). De plus le nombre de syndromes que l'on peut considérer N est majoré dans (2.41). Quoiqu'il en soit nous obtenons des gains exponentiels entre l'algorithme que nous venons de présenter pour résoudre DOOM et les algorithmes résolvant le problème du décodage. Nous comparons dans la table 1 ces exposants.

Table 2.2 – Exposants asymptotiques des algorithmes résolvant DOOM en fonction du nombre de syndromes pour $R = 1/2$ et $\omega = \omega^-$.

Algorithme	Exposant	Nombre de syndromes
Prange	0.120	1
Dumer	0.115	1
BJMM	0.100	1
[Sen11a]	0.0872	$2^{0.0872n}$

Le gain est ici significatif. En revanche, comme nous l'avons montré dans [DST17c], ce dernier s'amoindrit avec ω croissant au dessus de ω^-. Dans ces zones de paramètres un syndrome admet typiquement un nombre exponentiel de solutions ce dont les ISD tirent profit. L'avantage que confère DOOM en permettant de faire grossir artificiellement les listes où l'on cherche une collision semble ne plus rien apporter. La question d'utiliser d'autres ISD que celui proposé par Dumer est ajourd'hui ouverte. De plus, la question se pose de savoir s'il n'existe pas une autre solution que d'adapter les ISD pour résoudre DOOM.

Deuxième partie

Décodage(s) générique(s) pour tous les poids

Chapitre 3

Décodage par ensemble d'information en grandes distances

Introduction

Le décodage en petites distances dans $\mathbb{F}_2^n$. Depuis leur introduction, les crypto-systèmes utilisant des codes correcteurs ont pour objectif de faire reposer leur sécurité sur le décodage générique : étant donné un $[n,k]_q$-code $\mathcal{C}$ quelconque, un mot bruité $\mathbf{y} \stackrel{\triangle}{=} \mathbf{c} + \mathbf{e}$ de $\mathbb{F}_q^n$ où $\mathbf{c} \in \mathcal{C}$ et $|\mathbf{e}| = w$, trouver $\mathbf{c}' \in \mathcal{C}$ à distance w de $\mathbf{y}$. Ce problème est étudié depuis désormais près de 60 ans. Historiquement, l'immense majorité des études fut faite dans le cas où :

$$w \leqslant w_{\text{GV}} \quad \text{et} \quad q = 2.$$

Ce choix est tout sauf anodin. En effet, le crypto-système de McEliece réclame un code $\mathcal{C}$ que l'on sait décoder à distance w. Maintenant sous l'hypothèse que $\mathcal{C}$ peut se masquer, la sécurité se réduit au décodage générique à distance w. Le problème étant, quelles distances w pouvons-nous considérer et pour quels codes ? La partie I de ce document fut entre autre dédiée à cette question. Or, comme nous l'avons vu, les solutions ne courent pas les rues et dans le contexte de McEliece en métrique de Hamming il n'existe à ce jour essentiellement que les codes MDPC ou de Goppa. Ces deux familles ont alors été proposées dans le cas binaire, *ie :* $q = 2$. Les distances w pour lesquelles nous savons les décoder vérifient alors grossièrement à rendement constant :

$$w = o(n).$$

Ainsi, le décodage générique fut largement étudié pour des distances dites *petites*. Les crypto-systèmes d'Alekhnovich sont ensuite allés dans le même sens : bien que ces derniers utilisent directement des codes aléatoires, leur sécurité repose sur le décodage de mots de code bruité à distance $\sqrt{n}$.

En revanche, même si les propositions de McEliece et Alekhnovich ne se font dans ce cas, l'étude du décodage s'est principalement centrée autour de la borne de Gilbert-Varshamov :

$$w_{\text{GV}}(n,k,2) = \Theta(n) \quad (\text{par exemple } w_{\text{GV}}(n,n/2,2) \approx 0.11n).$$

Il s'agit du paramètre où le problème semble le plus dur et où nous nous attendons typiquement à une solution. Ces analyses, au-delà de leur intérêt algorithmique, ont un véritable sens cryptographique. En effet, par exemple la sécurité dans le protocole de Stern

repose sur le décodage à la distance w que l'on peut librement choisir. Nous avons dès-lors tout intérêt à considérer le w où le problème est le plus difficile.

Au-delà de la borne de Gilbert-Varshamov, comme nous l'avons précédemment vu, le nombre de solutions du décodage devient exponentiel et croît avec w jusqu'à $n/2$. Les meilleurs algorithmes de décodage arrivent alors à tirer profit de cette situation. Leur complexité reste néanmoins exponentielle mais diminue avec w croissant pour devenir polynomiale à $w = \frac{n-k}{2}$.

Nous pouvons maintenant aller plus loin en nous posant la question des poids encore plus élevés, *ie :* $w > \frac{n-k}{2}$. Dans le cas du corps $\mathbb{F}_2$ la réponse est plutôt triviale, la difficulté du problème est la même en les distances w et $n-w$. En effet, notons $\mathbf{1}$ le vecteur de $\mathbb{F}_2^n$ tout à un, décoder $\mathbf{y} = \mathbf{c}+\mathbf{e}$ où $|\mathbf{e}| = w$ est équivalent à décoder $\mathbf{c}+\mathbf{e}+\mathbf{1}$ et $|\mathbf{e}+\mathbf{1}| = n-w$. Cependant cette symétrie se brise dès que nous considérons des corps au cardinal plus élevé. Comme nous le verrons tout au long de ce chapitre, le problème du décodage à grande distance prend alors tout son sens pour des corps $\mathbb{F}_q$ où $q \geqslant 3$. Quoiqu'il en soit, avant de nous poser cette question revenons rapidement sur l'état de l'art du décodage pour $q \geqslant 3$.

Bref état de l'art du décodage dans $\mathbb{F}_q^n$ pour $q \geqslant 3$. Le problème du décodage générique pour $q \geqslant 3$ a jusqu'à ce jour suscité un moins grand intérêt. Une première explication à cela est l'absence de crypto-systèmes pour ces tailles de corps. Dans l'état de l'art actuel nous pouvons citer les études de [CG90 ; Pet10 ; Meu17 ; Hir16 ; GKH17 ; Int+18] reprenant l'idée des algorithmes par ensemble d'information et [Can17] montrant que toute adaptation des ISD donne la même complexité que l'algorithme de Prange pour $q \to +\infty$. Les différents algorithmes proposés ont alors principalement été étudiés pour des poids w autour de :

$$w_{\mathrm{GV}}(n,k,q)$$

où la difficulté du décodage semble être la plus élevée. Les résultats qui furent présentés donnèrent de surcroît une autre raison du faible intérêt pour le décodage dans $\mathbb{F}_q^n$. Notons $\alpha(q,R,\omega)$ l'exposant asymptotique *en base* 2 du meilleur algorithme résolvant le problème de décodage pour les paramètres $R \triangleq k/n$ et $\omega \triangleq w/n$. Supposons maintenant que l'on souhaite représenter le code considéré lors du décodage par une matrice de parité sous forme systématique. Cette dernière sera de taille (*en bits*) :

$$T(q,R,\omega) \triangleq \log_2(q) \cdot n^2 R(1-R) \quad \text{où} \quad n \triangleq \left\lceil \frac{128}{\alpha(q,R,\omega)} \right\rceil$$

pour 128 bits de sécurité. La question est alors quelle taille de clef optimale espérer ? Jusqu'aux récents travaux présentés dans ce chapitre nous avions :

$$\forall q \geqslant 3 \text{ (taille de corps)}, \quad \min_{R,\omega} T(q,R,\omega) > \min_{R,\omega} T(2,R,\omega).$$

Le décodage à taille de clef fixée semble donc plus facile dans $\mathbb{F}_q^n$ pour $q \geqslant 3$ que pour $q = 2$. Néanmoins, jusqu'à ce jour le décodage à distance élevée ne fut pas étudié dans le cas où $q \geqslant 3$. Or comme nous l'avons évoqué un peu plus haut, ce problème semble digne d'intérêt sa difficulté n'étant pas directement liée au décodage à petite distance.

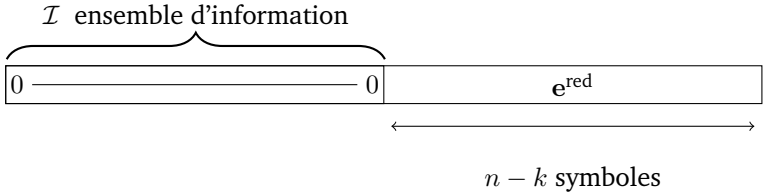
Le décodage à grande distance. Dans un contexte cryptographique, où il est souvent question de retrouver le mot *le plus proche*, le problème de trouver le mot *le plus éloigné* peut sembler surprenant. A titre d'exemple, par décodage à grande distance nous entendrons : à partir de $\mathbf{y} \in \mathbb{F}_q^n$, trouver pour un code $\mathcal{C}$ sur $\mathbb{F}_q$ un mot $\mathbf{c} \in \mathcal{C}$ tel que $|\mathbf{y}-\mathbf{c}| \approx n$.

Afin de résoudre ce problème de décodage à grande distance nous proposons dans un premier temps d'étendre l'algorithme de Prange. Reprenons-le tout d'abord comme nous

l'avions décrit dans §2.1. Étant donné $\mathcal{C}$ un $[n,k]_q$-code, un mot $\mathbf{y} \in \mathbb{F}_q^n$ que l'on souhaite décoder, l'algorithme tire un ensemble d'information $\mathcal{I}$ du code et calcule l'unique mot de code $\mathbf{c}$ vérifiant :

$$\forall i \in \mathcal{I}, \quad c_i = y_i.$$

Nous espérons ainsi trouver un mot de code proche. Autrement dit, avec l'algorithme de Prange nous avons calculé l'unique erreur $\mathbf{e} \in \mathbb{F}_q^n$ telle que $\mathbf{e}_{\mathcal{I}} = \mathbf{0}$ et $\mathbf{y} - \mathbf{e} \in \mathcal{C}$. Le code $\mathcal{C}$ étant aléatoire, la forme de l'erreur obtenue est alors donnée à une permutation près par :



où $\mathbf{e}^{\text{red}} \in \mathbb{F}_q^{n-k}$ est essentiellement uniformément distribuée. Ainsi le poids moyen de l'erreur obtenue sera :

$$\frac{q-1}{q}(n-k).$$

Maintenant si l'on souhaite un erreur de petit poids $\leqslant (1-\varepsilon)\frac{(q-1)}{q}(n-k)$ où $\varepsilon > 0$ il faudra répéter l'algorithme un nombre exponentiel de fois. Notons que toutes les améliorations [LB88 ; Ste88 ; Dum91 ; MMT11 ; Bec+12 ; MO15 ; BM17] de l'algorithme de Prange (ainsi que celles ne s'en inspirant pas) sont de même nature en terme de complexité asymptotique : polynomiale en le poids $(n-k)/2$ et exponentielle en deçà. La complexité moyenne du décodage semble donc être bien mesurée par l'algorithme de Prange.

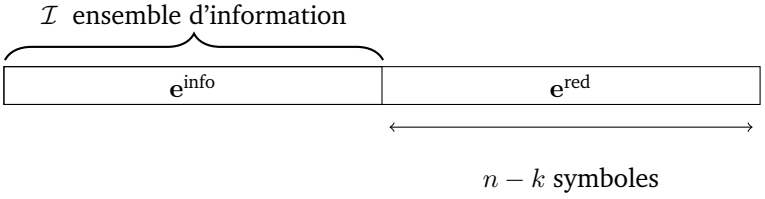
Supposons désormais que l'on cherche un mot de code à grande distance. Si nous reprenons l'algorithme de Prange, l'idée pour le généraliser aux grandes distances est cette fois pour $\mathcal{I}$ de commencer par choisir $(z_i)_{i\in\mathcal{I}}$ telle que :

$$\forall i \in \mathcal{I}, \quad z_i \neq y_i.$$

et de calculer ensuite l'unique mot $\mathbf{c} \in \mathcal{C}$ tel que :

$$\forall i \in \mathcal{I}, \quad c_i = z_i.$$

Plus généralement, cette procédure revient à calculer l'unique erreur $\mathbf{e} \in \mathbb{F}_q^n$ telle que $\mathbf{y} - \mathbf{e} \in \mathcal{C}$ mais où l'on a choisi $\mathbf{e}_{\mathcal{I}}$ avec non nécessairement des symboles nuls. La forme de l'erreur obtenue est cette fois-ci :



où $\mathbf{e}^{\text{info}} \in \mathbb{F}_q^k$ est librement choisie et $\mathbf{e}^{\text{red}}$ est essentiellement distribuée uniformément. L'algorithme de Prange généralisé renvoie donc une erreur de poids moyen :

$$\mathbb{E}\left(\mathbf{e}^{\text{info}}\right) + \frac{q-1}{q}(n-k).$$

Ainsi, en choisissant correctement $\mathbf{e}^{\text{info}} \in \mathbb{F}_q^k$ nous pouvons atteindre en temps polynomial tout poids de l'intervalle :

$$\left[\!\left[\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k) \right]\!\right].$$

Si nous souhaitons obtenir une erreur de poids $\geqslant (1+\varepsilon)\left(k + \frac{q-1}{q}(n-k)\right)$ où $\varepsilon > 0$ il faudra répéter l'algorithme un nombre exponentiel de fois. Dans ce cas la meilleure stratégie est de choisir à chaque itération $\mathbf{e}^{\text{info}} \in \mathbb{F}_q^k$ de poids plein. Ceci suggère que le problème de décodage à distance $w \geqslant (1+\varepsilon)\left(k + \frac{q-1}{q}(n-k)\right)$ est exponentiellement difficile.

Des instances plus difficiles et un crypto-système pour le décodage à grande distance. Nous présenterons précisément dans la première cette section de ce chapitre cette généralisation de l'algorithme de Prange. Nous comparons dans la figure 3.1 les exposant asymptotiques de cette généralisation selon $q = 2$ et $q = 3$ pour $R = 1/2$ en fonction de $\omega \triangleq w/n$.

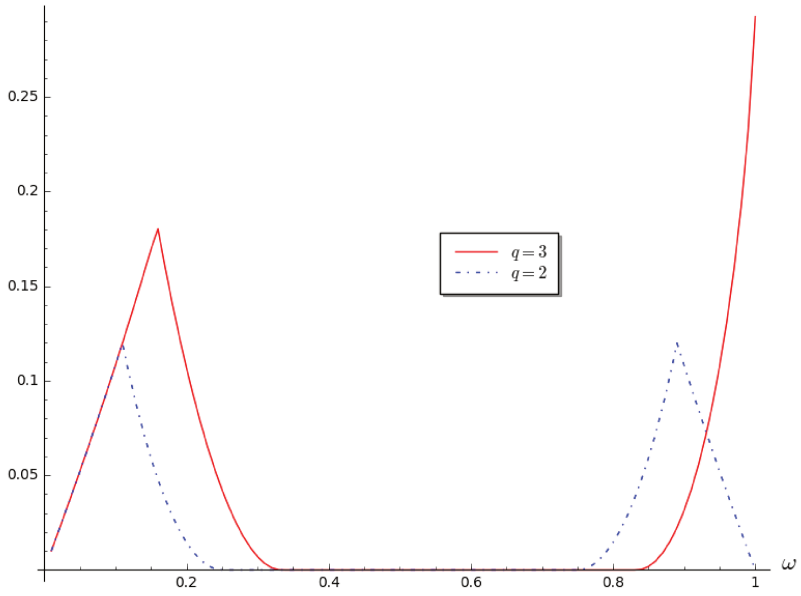


Figure 3.1 – Exposants asymptotiques de la complexité de l'algorithme de Prange pour $R = 1/2$ en fonction de ω pour $q = 2$ et $q = 3$.

Nous observons bien une symétrie entre grandes et petites distances pour $q = 2$ contrairement au cas de $q = 3$. De plus, nous constatons que le pire exposant de l'algorithme de Prange est pour $q = 3$ dans la zone de distance élevée.

Nos travaux avec [Bri+19], dont nous discuterons dans les sections 3.2 à 3.5 de ce chapitre, ont montré comment utiliser des approches similaires à celle de Wagner [Wag02] et de BJMM [Bec+12] pour améliorer l'algorithme de Prange pour $q = 3$. Plus précisément nous avons suivi la même approche que les ISD. En revanche, contrairement à ces derniers nous ne nous sommes pas réduits au problème du décodage d'un code poinçonné mais un

problème de sac à dos binaire modulo 3. Nous avons alors obtenu des gains exponentiels significatifs. Néanmoins, malgré nos améliorations non-triviales, le problème du décodage à grande distance dans $\mathbb{F}_3$ est, comme nous le décrirons dans la section 5, bien plus difficile à taille de clef fixée que le décodage binaire. Nous résumons la situation dans la table 3.1.

Table 3.1 – Taille de clef minimale (en kbits) pour une complexité de résolution en temps de 2^{128}.

Algorithme	$q = 2$	$q = 3$ et $w/n > 1/2$
Prange	275	44
Dumer/Wagner	295	83
BJMM/Notre algorithme	374	99

Le décodage à grande distance a de plus été directement utilisé pour la première fois dans le schéma de signature Wave [DST19b] (que nous décrirons dans la partie III de ce document) utilisant des codes $(U, U + V)$-généralisés. Comme nous le verrons, la signature d'un message est grosso modo un mot de code *très éloigné* du message. La sécurité de ce schéma repose donc de façon cruciale sur notre compréhension du décodage à grande distance. Nos travaux [Bri+19] ont alors permis de donner des paramètres plus précis au schéma Wave.

Le décodage à grande distance en métrique rang ? Le décodage générique semblant exponentiellement difficile à grande distance en métrique de Hamming il est légitime de se poser la question concernant la métrique rang. La réponse est négative. En effet, notre extension de l'algorithme de Prange renverra pour le décodage générique en métrique rang une erreur de la forme (à une permutation près) :

$$(\mathbf{e}^{\text{info}}, \mathbf{e}^{\text{red}}), \quad \mathbf{e}^{\text{info}} \in \mathbb{F}_{q^m}^{k} \quad \text{et} \quad \mathbf{e}^{\text{red}} \in \mathbb{F}_{q^m}^{n-k}.$$

où $\mathbf{e}^{\text{red}}$ est essentiellement uniformément distribuée, donc typiquement de rang plein, *i.e :* $\min(m, n - k)$. Il suffit alors de choisir $\mathbf{e}^{\text{info}}$ de poids plein pour s'attendre à une erreur de rang maximal.

Quelques notations et rappels

Dans ce chapitre $(\mathbf{H}, \mathbf{s}) \in \mathbb{F}_q^{(n-k)\times n} \times \mathbb{F}_q^{n-k}$ désignera une instance du problème de décodage que nous souhaitons résoudre à grande distance :

$$w \in [\![0, n]\!]$$

où nous supposerons dans tout ce qui suit que w/n et k/n sont fixés. La matrice $\mathbf{H}$ désignera la variable aléatoire :

$$\mathbf{H} \hookleftarrow \mathbb{F}_q^{(n-k)\times n}.$$

La distribution de $\mathbf{s}$ dépendra quant à elle du nombre attendu de solutions au problème du décodage considéré. Rappelons que ce nombre est donné par :

$$\frac{\binom{n}{w}(q-1)^w}{q^{n-k}}$$

qui est exponentiellement élevé, faible ou polynomial selon les rapports fixés w/n et k/n. Plus précisément, ce nombre de solutions est (voir §1.1.3)

— exponentiel pour :

$$w/n \in]\omega^-, 1] \quad \text{et} \quad k/n > 1 - \log_q(q-1)$$

ou

$$w/n \in]\omega^-, \omega^+[\quad \text{et} \quad k/n \leqslant 1 - \log_q(q-1)$$

— polynomial si :

$$w/n = \omega^- \quad \text{et} \quad k/n > 1 - \log_q(q-1)$$

ou

$$w/n \in \{\omega^-, \omega^+\} \quad \text{et} \quad k/n \leqslant 1 - \log_q(q-1)$$

— exponentiellement faible sinon.

Les deux quantités fondamentales ω^- et ω^+ ont été introduites dans §1.1.3. Rappelons que ces dernières sont définies pour $R \triangleq k/n$ comme [1] :

$$\begin{aligned} \omega^- &\triangleq g_q^-(1-R) \\ \omega^+ &\triangleq \begin{cases} g_q^+(1-R) & \text{si } R \leqslant 1 - \log_q(q-1) \\ 1 & \text{sinon.} \end{cases} \end{aligned}$$

Remarque 3.1. Dans le cas binaire nous avons $\omega^+ = 1 - \omega^-$. Cette symétrie n'est plus vérifiée dès que $q \geqslant 3$.

A moins que nous mentionnons le contraire, nous considèrerons alors $\mathbf{s}$ comme une variable aléatoire uniformément distribuée si le nombre de solutions attendu est exponentiel. Autrement cette dernière sera tirée telle que $\mathbf{s} \triangleq \mathbf{e}\mathbf{H}^\intercal$ où $\mathbf{e} \hookleftarrow S_w$.

3.1 Généralisation de l'algorithme de Prange

Nous présentons dans cette section notre généralisation de l'algorithme de Prange. Rappelons que cet algorithme est itératif. Commençons par généraliser ses itérations.

3.1.1 Une itération de la généralisation de l'algorithme de Prange

Notre généralisation consiste une fois les matrices $\mathbf{S}$ et $\mathbf{P}$ calculées telles que :

$$\mathbf{SHP} = \begin{pmatrix} \mathbf{1}_{n-k} & \mathbf{H}' \end{pmatrix}$$

à renvoyer comme solution :

$$\mathbf{e}^{\text{sol}} = (\mathbf{s}\mathbf{S}^\intercal - \mathbf{e}^{\text{info}}\mathbf{H}'^\intercal, \mathbf{e}^{\text{info}})\mathbf{P}^\intercal.$$

où $\mathbf{e}^{\text{info}} \in \mathbb{F}_q^k$ est choisi à notre guise (possiblement à $\mathbf{0}_k$ où l'on retrouve l'itération de l'algorithme de Prange originel). Ce vecteur nous offre alors un degré de liberté, en particulier nous pouvons choisir son poids avec une certaine distribution. En revanche, pour un syndrome $\mathbf{s}$ uniformément distribué, l'autre partie de l'erreur sera uniforme.

Nous donnons dans l'algorithme 1 la version complète et précise de notre généralisation.

La proposition qui suit donne alors la distribution de poids de sortie de l'algorithme 1 quand en entrée est donné un syndrome $\mathbf{s}$ uniformément distribué. Un des intérêt de notre algorithme est que sa distribution de poids est paramétrée par une distribution $\mathcal{D}$.

1. g_q^- (resp. g_{q^+}) dénote l'inverse de l'entropie q-aire à défini sur $[0, \frac{q-1}{q}]$ (resp. $[\frac{q-1}{q}, 1]$)

Algorithme 1 ITÉRATIONPRANGEGEN($\mathbf{H}, \mathbf{s}$) — Une itération de la généralisation de l'algorithme de Prange

Paramètre : q, n, k, $\mathcal{D}$ une distribution sur $[\![0, k]\!]$

Require: $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$
Ensure: $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$
1: $t \hookleftarrow \mathcal{D}$
2: $\mathcal{I} \leftarrow$ INFOSET($\mathbf{H}$) ▷ INFOSET*(*$\mathbf{H}$*) retourne un ensemble d'information du code de matrice de parité* $\mathbf{H}$
3: $\mathbf{x} \hookleftarrow \{\mathbf{y} \in \mathbb{F}_q^n \mid |\mathbf{y}_\mathcal{I}| = t\}$
4: $\mathbf{e} \leftarrow$ PRANGEALGÈBRELINÉAIRE($\mathbf{H}, \mathbf{s}, \mathcal{I}, \mathbf{x}$)
5: **return** $\mathbf{e}$

function PRANGEALGÈBRELINÉAIRE($\mathbf{H}, \mathbf{s}, \mathcal{I}, \mathbf{x}$)

Require: $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_q^{n-k}$, $\mathcal{I}$ un ensemble d'information du code de matrice de parité $\mathbf{H}$, $\mathbf{x} \in \mathbb{F}_q^n$
Ensure: $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ et $\mathbf{e}_\mathcal{I} = \mathbf{x}_\mathcal{I}$
$\mathbf{P} \leftarrow$ une $n \times n$ permutation renvoyant $\mathcal{I}$ sur les k dernières coordonnées
$(\mathbf{A} \mid \mathbf{B}) \leftarrow \mathbf{H}\mathbf{P}$ ▷ $\mathbf{A} \in \mathbb{F}_q^{(n-k)\times(n-k)}$
$(\mathbf{0} \mid \mathbf{e}') \leftarrow \mathbf{x}$ ▷ $\mathbf{e}' \in \mathbb{F}_q^k$
$\mathbf{e} \leftarrow \left((\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal)\left(\mathbf{A}^{-1}\right)^\intercal, \mathbf{e}'\right)\mathbf{P}^\intercal$
return $\mathbf{e}$

Proposition 3.1. *Pour* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ fixée *et* $\mathbf{s}$ *uniformément distribué sur* $\mathbb{F}_q^{n-k}$*, nous avons pour la sortie* $\mathbf{e}$ *de* ITÉRATIONPRANGEGEN($\mathbf{H}, \mathbf{s}$) *:*

$$|\mathbf{e}| = S + T$$

où $S \in [\![0, n-k]\!]$ *et* $T \in [\![0, k]\!]$ *sont des variables aléatoires indépendantes. Ici* S *désigne le poids de Hamming d'un vecteur uniformément distribué sur* $\mathbb{F}_q^{n-k}$ *tandis que* $\mathbb{P}(T = t) = \mathcal{D}(t)$*. La distribution de* $|\mathbf{e}|$ *est alors donnée par :*

$$\mathbb{P}_{\mathbf{s},c}\left(|\mathbf{e}| = w\right) \quad = \quad \sum_{t=0}^{w} \frac{\binom{n-k}{w-t}(q-1)^{w-t}}{q^{n-k}}\mathcal{D}(t), \quad \mathbb{E}_{\mathbf{s},c}(|\mathbf{e}|) = \overline{\mathcal{D}} + \tfrac{q-1}{q}(n-k)$$

où la probabilité est calculée sur $\mathbf{s}$ *uniforme, l'aléa interne* c *de l'algorithme et* $\overline{\mathcal{D}} = \sum_{t=0}^{k} t\mathcal{D}(t)$*.*

Démonstration de la proposition 3.1.

Par définition toute sortie de l'algorithme 1 s'écrit comme :

$$\left((\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal)\left(\mathbf{A}^{-1}\right)^\intercal, \mathbf{e}'\right)\mathbf{P}^\intercal$$

où $\mathbf{P}$ est une matrice de permutation. De plus, la matrice $\mathbf{A}$ étant une bijection, le poids de ce vecteur est distribué comme $\left|(\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal, \mathbf{e}')\right|$. D'où :

$$\begin{aligned}\mathbb{P}_{\mathbf{s},c}(|\mathbf{e}| = w) &= \sum_{t=0}^{w} \mathbb{P}_{\mathbf{s},c}\left(\left|(\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal, \mathbf{e}')\right| = w - t \mid |\mathbf{e}'| = t\right)\mathbb{P}\left(|\mathbf{e}'| = t\right) \\ &= \sum_{t=0}^{w} \mathbb{P}_{\mathbf{s},c}\left(\left|(\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal, \mathbf{e}')\right| = w - t \mid |\mathbf{e}'| = t\right)\mathcal{D}(t)\end{aligned}$$

Maintenant $\mathbf{s} \in \mathbb{F}_q^{n-k}$ étant uniformément distribuée, il en est de même pour $\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal$ quelques soient $\mathbf{e}'$ et $\mathbf{B}$ qui sont indépendants de $\mathbf{s}$. Donc nous avons :

$$\mathbb{P}_{\mathbf{s},c}\left(\left|(\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal, \mathbf{e}')\right| = w - t \mid |\mathbf{e}'| = t\right) = \frac{\binom{n-k}{w-t}(q-1)^{w-t}}{q^{n-k}}$$

d'où le résultat. □

3.1.2 L'algorithme de Prange pour toutes distances

L'algorithme généralisé de Prange consiste alors tout simplement à répéter ITÉRATIONPRANGEGEN($\mathbf{H}, \mathbf{s}$) en étant précautionneux sur le choix de $\mathcal{D}$. Plus précisément :

— si $w < \frac{q-1}{q}(n-k)$, $\mathcal{D}$ est choisie comme étant la distribution nulle,

— si $w \in [\![\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k)]\!]$, $\mathcal{D}$ est la distribution constante égale à $w - \frac{q-1}{q}(n-k)$,

— si $w > k + \frac{q-1}{q}(n-k)$, $\mathcal{D}$ est est choisie comme étant la distribution constante égale à k.

Nous obtenons alors comme probabilité (sur $\mathbf{H}, \mathbf{s}$) de succès de l'algorithme :

$$\tilde{O}\left(\frac{\binom{n-k}{w-j}(q-1)^{w-j}}{\min\left(\binom{n}{w}(q-1)^w, q^{n-k}\right)}\right)$$

$$\text{où} \begin{cases} j = 0 & \text{si } w \leqslant \frac{q-1}{q}(n-k) \\ j = k & \text{si } w \geqslant k + \frac{q-1}{q}(n-k) \\ j = w - \frac{q-1}{q}(n-k) & \text{sinon.} \end{cases}$$

Remarque 3.2. Nous retrouvons, à travers le facteur $\min\left(\binom{n}{w}(q-1)^w, q^{n-k}\right)$, l'incidence du nombre de solutions attendu dans la complexité de l'algorithme.

Nous traçons l'exposant asymptotique de la généralisation de l'algorithme de Prange pour $q = 3$, $R = 1/5$ en fonction du poids relatif $\omega \triangleq w/n$ dans la figure 3.2. Le pire exposant est atteint en la distance relative élevée ω^+. Au delà la complexité diminue. Nous observons ce comportement du fait que $\omega^+ < 1$. En effet, dans ce cas pour $\omega^+ < w/n \leqslant 1$ il existe une solution au problème de décodage avec une probabilité exponentiellement faible comme dans le cas où $w/n < \omega^-$.

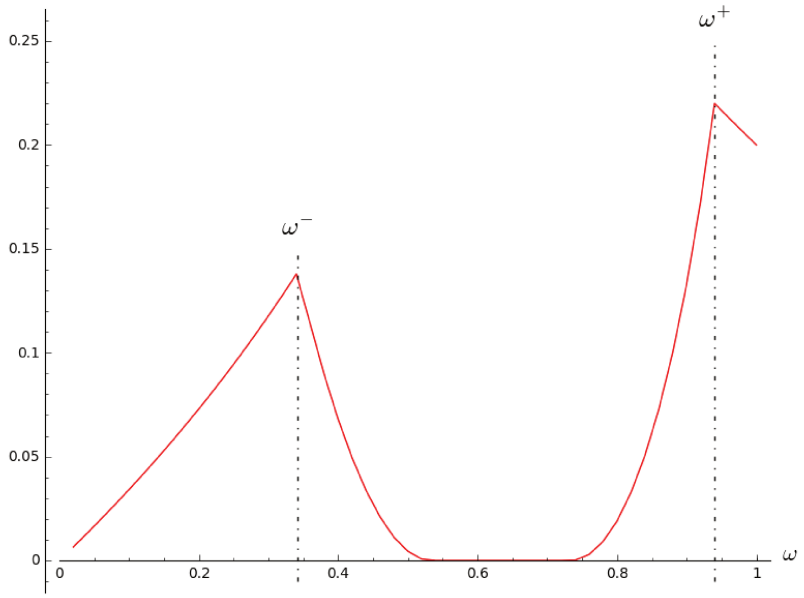


Figure 3.2 – Exposant asymptotique de la complexité de l'algorithme de Prange pour $R = 1/5$ en fonction de $\omega \triangleq w/n$ et avec $q = 3$.

Dans la figure 3.3 nous traçons l'exposant asymptotique de Prange pour $q = 3$, $w/n = \omega^+$ et cette fois-ci en fonction de R. Nous constatons que le plus grand exposant est atteint en $R = 1 - \log_3(2)$.

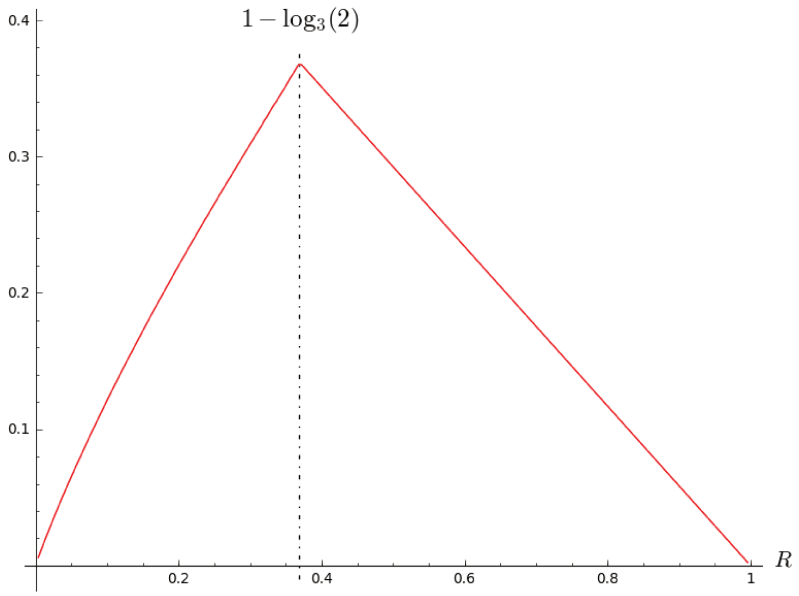


Figure 3.3 – Exposant asymptotique l'algorithme de Prange pour $q = 3$ et $w/n = \omega^+$ en fonction de R.

Plus généralement, nous avons la proposition suivante :

Proposition 3.2. *La pire complexité de la généralisation de l'algorithme de Prange pour $q \geqslant 3$ est atteinte en les paramètres :*

$$w/n = 1 \quad \textit{et} \quad R_0 \stackrel{\triangle}{=} 1 - \log_q(q-1).$$

Cette-dernière est alors donnée à un facteur polynomiale près par :

$$(q-1)^{\lfloor R_0 n \rfloor},$$

La quantité R_0 est exactement le rendement où en $w/n = 1$ nous attendons une unique solution au problème du décodage : pour $R > R_0$ le nombre de solutions est exponentiellement élevé en $\omega = 1$ alors qu'il est exponentiellement faible en ce poids relatif pour $R < R_0$.

Nous allons maintenant présenter nos améliorations [Bri+19] de l'algorithme de Prange pour :

$$q = 3 \quad \text{et} \quad w \geqslant k + \frac{2}{3}(n-k). \tag{3.1}$$

Nous avons proposé de suivre l'approche algorithmique de type ISD.

3.2 Les algorithmes ISD en grandes distances

Nous reprenons dans cette section la description des algorithmes ISD ainsi que les résultats associés (voir §2.3).

3.2.1 Le choix des paramètres

Commençons par rappeler que les algorithmes ISD ont génériquement deux paramètres :

$$\ell \in [\![0, n]\!] \quad \text{et} \quad p \in [\![0, \min(w, k+\ell)]\!].$$

Par définition, un algorithme ISD dans $\mathbb{F}_3$ trouve un ensemble de solutions du problème de décodage à distance p avec pour entrée (obtenue à partir d'algèbre linéaire sur $\mathbf{H}$) :

$$(\mathbf{H}'', \mathbf{s}'') \in \mathbb{F}_3^{\ell \times (k+\ell)} \times \mathbb{F}_3^{\ell}.$$

L'algorithme complète ensuite de façon unique les solutions obtenues sur $n-k-\ell$ symboles en espérant obtenir sur cette partie un poids $w-p$. La probabilité :

$$\alpha_{p,\ell}$$

qu'une solution de poids p sur $\mathbb{F}_q^{\ell}$ se complète en une solution de poids w sur $\mathbb{F}_q^n$ est alors donnée (proposition 2.7, $q=3$) par :

$$\alpha_{p,\ell} = \tilde{O}\left(\frac{\binom{n-k-\ell}{w-p} 2^{w-p}}{\min\left(3^{n-k-\ell}, \binom{n}{w} 2^w 3^{-\ell}\right)} \right). \tag{3.2}$$

Dans notre cas, du fait que $w \geqslant k + \frac{2}{3}(n-k)$, nous proposons de commencer par choisir p maximal afin d'améliorer la probabilité de succès de l'algorithme :

$$p = k + \ell \tag{3.3}$$

Ce choix de p implique asymptotiquement pour le second paramètre ℓ :

$$\ell = \Theta(n) \tag{3.4}$$

autrement l'algorithme ISD serait de même complexité à un facteur polynomial près que celui de Prange. En effet nous pouvons vérifier avec le lemme 1.2 et (3.2) que $\alpha_{k+\ell,\ell}$ est donnée quand $\ell = o(n)$ à un facteur polynomial près par (k/n et w/n sont fixés) :

$$\frac{\binom{n-k}{w-k} 2^{w-k}}{\min\left(\binom{n}{w} 2^w, 3^{n-k}\right)}$$

qui n'est autre que la probabilité de succès de la généralisation de l'algorithme de Prange en grande distance.

3.2.2 Une réduction au problème du sac à dos

Nous allons désormais montrer que les ISD avec les choix de paramètres que nous avons fait pour résoudre le décodage à grande se réduisent à la résolution d'un problème de sac à doc et non de décodage. Pour cela revenons un instant sur le cas général. Les ISD réduisent la résolution d'une instance du décodage au problème suivant (pour $m = k + \ell$) :

Problème 3.1 (Sub-SD(q, m, ℓ, p, L))**.**

- Instance : m *vecteurs* $\mathbf{h}_i \in \mathbb{F}_q^{\ell}$ *pour* $1 \leqslant i \leqslant m$ *et un vecteur cible* $\mathbf{s} \in \mathbb{F}_q^{\ell}$,
- Recherche : L *solutions* $\mathbf{b}^{(j)} = (b_1^{(j)}, \cdots, b_m^{(j)}) \in \mathbb{F}_q^m$ *pour* $1 \leqslant j \leqslant L$ *de* $\sum_{i=1}^m b_i^{(j)} \mathbf{h}_i = \mathbf{s}$ *avec* $|\mathbf{b}^{(j)}| = p$.

La contrainte de poids sur les solutions $\mathbf{b}^{(j)}$ fait alors de ce problème un décodage. Sans cette contrainte et en restreignant ces vecteurs à coefficients dans $\{0, 1\}$ nous aurions le problème du sac à dos modulaire :

Problème 3.2 (Sac à dos modulaire SDM(q, m, ℓ, L))**.**

— Instance : m *vecteurs* $\mathbf{h}_i \in \mathbb{F}_q^\ell$ *pour* $1 \leqslant i \leqslant m$ *et un vecteur cible* $\mathbf{s} \in \mathbb{F}_q^\ell$,

— Recherche : L *solutions* $\mathbf{b}^{(j)} = (b_1, \cdots, b_m^{(j)}) \in \{0,1\}^m$ *pour* $1 \leqslant j \leqslant L$ *de* $\sum_{i=1}^m b_i^{(j)} \mathbf{h}_i = \mathbf{s}$.

Ce problème est aujourd'hui bien compris et il a largement été étudié [CFG89 ; GM91 ; FP05 ; Lyu05 ; HJ10 ; BCJ11] dans le cas où :

$$L = 1 \quad ; \quad q = 2^m \quad ; \quad \ell = 1$$

Il s'agit des paramètres pour lesquels la difficulté est maximale et où en moyenne sur les entrées il existe une unique solution (typiquement nous nous attendons à $2^m/q = 1$ solutions). Il y a bien d'autres régimes de paramètres, chacun menant à des algorithmes aux ordres de complexité différents. Par exemple, dans le cas où :

$$q = O(2^{m^\varepsilon})$$

avec $0 < \varepsilon < 1$, le problème admet en moyenne un nombre exponentiel de solutions, *ie :* $2^{m(1-m^{\varepsilon-1})}$. Dans ce cas on parle usuellement de problème du sac à dos à *grande densité*. Il existe alors un algorithme [Lyu05] sous-exponentiel pour le résoudre.

Nous résumons les différentes complexités de résolution du problème de sac à dos en fonction des paramètres dans la table 3.2 mais dans le cas où seulement une solution est requise $(L = 1)$ et pour $\ell = 1$. Nous constatons que la difficulté du problème décroit avec le nombre de solutions $2^m/q$ au problème.

Table 3.2 – Complexité des meilleurs algorithmes pour résoudre $\mathrm{SDM}(q, m, 1, 1)$.

Valeur de q	Complexité	Référence
$O(m)$	poly(m)	[GM91 ; CFG89]
$O(m^2)$	poly(m)	[FP05]
$O(2^{m^\varepsilon})$ pour $\varepsilon < 1$	$2^{O\left(\frac{m^\varepsilon}{\log(m)}\right)}$	[Lyu05]
$O(2^m)$	$2^{O(m)}$	[HJ10 ; BCJ11]

Revenons maintenant au décodage en grandes distances. Nous avons réduit dans la sous-section qui précède les ISD à la résolution de :

$$\text{Sub-SD}(3, k+\ell, \ell, k+\ell, L).$$

Dans ce cas particulier, les solutions recherchées étant de poids plein ($p = k + \ell$), *ie* : à coefficients dans $\mathbb{F}_3^* = \{1, 2\}$, ce problème du décodage est équivalent au problème du sac à dos comme montré par le lemme suivant.

Lemme 3.1. *Si nous avons un algorithme qui résout* $\mathrm{SDM}(3, k+\ell, \ell, L)$ *alors nous disposons d'un algorithme qui résout* Sub-SD$(3, k+\ell, \ell, k+\ell, L)$ *avec la même complexité et réciproquement.*

Démonstration lemme 3.1.

Soient $\mathcal{A}$ un algorithme résolvant $\mathrm{SDM}(3, k+\ell, \ell, L)$ et une instance $(\mathbf{h}_1, \ldots, \mathbf{h}_{k+\ell}, \mathbf{s})$ de Sub-SD$(3, k+\ell, \ell, k+\ell, L)$. Nous cherchons $b_1, \ldots, b_{k+\ell} \in \{1, 2\}$ ($\mathbb{F}_3 = \{0, 1, 2\}$) tels que $\sum_{i=1}^{k+\ell} b_i \mathbf{h}_i = \mathbf{s}$. Considérons :

$$\mathbf{s}' = 2\mathbf{s} - \sum_{i=1}^{k+\ell} \mathbf{h}_i$$

et donnons comme entrée à $\mathcal{A}$: $(\mathbf{h}_1, \dots, \mathbf{h}_{k+\ell}, \mathbf{s}')$. Nous obtenons $b'_1, \dots, b'_{k+\ell} \in \{0, 1\}$ tels que $\sum_{i=1}^{k+\ell} b'_i \mathbf{h}_i = \mathbf{s}'$. Il suffit alors de renvoyer $(b_1, \cdots, b_{k+\ell})$ où :

$$\forall i \in [\![1, k+\ell]\!], \quad b_i \triangleq \frac{b'_i + 1}{2}.$$

Nous obtenons bien une solution du problème Sub-SD considéré. En effet, les éléments b_i sont dans $\{1, 2\}$ et nous avons :

$$\sum_{i=1}^{k+\ell} b_i \mathbf{h}_i = \sum_{i=1}^{k+\ell} \frac{b'_i + 1}{2} \mathbf{h}_i = \frac{\mathbf{s}'}{2} + \sum_{i=1}^{k+\ell} \frac{\mathbf{h}_i}{2} = \mathbf{s}.$$

La réciproque se fait tout aussi simplement. □

D'après ce lemme nous réduisons donc nos ISD pour le décodage en grande distance à la résolution de :

$$\text{SDM}(3, k+\ell, \ell, L).$$

Or dans ce contexte, d'après (3.4) :

$$\ell = \Theta(k) \quad (k/n \text{ est fixé}).$$

Nous serons cependant dans ce qui suit en mesure de choisir ℓ comme une petite fraction de k. Le nombre de solutions :

$$\frac{2^{k+\ell}}{3^\ell} \tag{3.5}$$

sera exponentiel. Cette situation est similaire au régime de grande densité mais pour autant les meilleurs algorithmes sont eux de complexité exponentielle. Nous sommes dans un régime de densité modérée.

De plus, dans notre cas $L \gg 1$. En effet, rappelons (proposition 2.2) que si nous disposons d'un algorithme trouvant L solutions en temps T de SDM$(3, k+\ell, \ell, L)$ alors la complexité de résolution du problème de décodage considéré sera de l'ordre de :

$$T \cdot \max\left(1, \frac{1}{L \cdot \alpha_{k+\ell,\ell}}\right)$$

où $\min(1, L \cdot \alpha_{k+\ell,\ell})$ est essentiellement la probabilité de succès d'une itération de l'algorithme et $\alpha_{k+\ell,\ell}$ est donnée dans (3.2). Il y a donc un véritable compromis entre le temps amorti

$$\frac{T}{L}$$

pour trouver L solutions du problème de sac à dos et la probabilité de succès. Or nous serons dans un cas où la complexité des algorithmes les plus efficaces pour trouver une solution est exponentielle. Nous proposons [Bri+19] alors dans la prochaine section d'utiliser une approche à la Dumer et Wagner [Wag02] dans un régime permettant de trouver un nombre exponentiel $\tilde{O}(L)$ de solutions en temps $\tilde{O}(L)$. L'algorithme que nous donnons est le pendant de celui de Dumer [Dum91] dans le cas binaire $q = 2$. En revanche, du fait du grand nombre de solutions (3.5), la profondeur de notre algorithme ne sera pas en général deux. A titre d'exemple, pour les paramètres de la signature Wave nous aurons 7 étages.

Enfin, dans la section 4 nous montrerons comment utiliser la technique des représentations (que nous avons décrite dans §2.3.3) dans le cas particulier où $q = 3$. Notre algorithme sera l'équivalent de celui de BJMM [Bec+12] où $q = 2$. L'optimisation de l'approche sera quant à elle totalement différente.

3.3 Approche de Wagner

Notre objectif dans ce qui suit est de résoudre le problème du sac à dos suivant :

$$\mathrm{SDM}(3, k+\ell, \ell, L). \tag{3.6}$$

Ce problème admet en moyenne sur ses entrées $\frac{2^{k+\ell}}{3^\ell}$ solutions. Nous sommes ici dans un contexte où k/n est par exemple égal à $1/2$ alors que ℓ est une petite fraction constante de n. Le nombre de solutions est donc extrêmement grand ce dont on peut tirer profit pour optimiser le coût de l'algorithme ISD. Analysons rapidement l'algorithme de Dumer (voir §2.2.1) pour comprendre une telle assertion. Restreignons-nous aux paramètres :

$$\frac{k}{n} = \frac{1}{2} \quad \text{et} \quad \frac{w}{n} = 0.9. \tag{3.7}$$

Avec l'approche de Dumer nous pouvons trouver L solutions de (3.6) en temps L à la condition :

$$L = 3^\ell \iff \frac{L^2}{3^\ell} = L.$$

où L la taille des listes construites, à partir d'erreurs de poids plein sur $\mathbb{F}_3^{(k+\ell)/2}$. Cette quantité L doit alors vérifier :

$$L = 3^\ell \leqslant 2^{(k+\ell)/2} \quad \text{(taille de liste maximale)} \tag{3.8}$$

La complexité de l'algorithme ISD sera donc à un facteur polynomial près :

$$f(\ell) \triangleq 3^\ell \cdot \max\left(1, \frac{1}{3^\ell \alpha_{k+\ell,\ell}}\right)$$

Nous pouvons alors vérifier avec (3.2) que $\ell \mapsto f(\ell)$ est décroissante jusqu'à un point ℓ_0 puis croissante où :

$$3^{\ell_0} = \frac{1}{p_{k+\ell_0,\ell_0}}. \tag{3.9}$$

Ce paramètre ℓ_0 est alors admissible d'après (3.8) si :

$$3^{\ell_0} \leqslant 2^{(k+\ell_0)/2}$$

ce qui est vérifié car d'après les paramètres donnés dans (3.7) et (3.9) nous avons :

$$\frac{\ell_0}{n} \approx 0.017 \quad \text{et} \quad \frac{1}{n}\log_3\left(2^{(k+\ell)/2}\right) \approx 0.16.$$

De plus, pour le paramètre ℓ_0 l'exposant en base 3 du nombre de solutions au problème 3.6 est donné par :

$$\frac{1}{n}\log_3\left(\frac{2^{k+\ell}}{3^\ell}\right) \approx 0.31$$

Nous remarquons alors que l'ensemble des 3^{ℓ_0} solutions trouvées par l'algorithme de Dumer est bien plus petit que le nombre de solutions $\frac{2^{k+\ell_0}}{3^{\ell_0}}$ du problème de sac à dos considéré. Ce régime est complètement différent du cas binaire où l'algorithme de Dumer trouve toutes les solutions du problème considéré (voir dans §2.3.2 l'équation (2.28)).

L'algorithme de Wagner prend alors pour ces jeux de paramètres tout son sens. Ce dernier décime l'espace de recherche des solutions tout en permettant de les retrouver plus efficacement. C'est exactement cette remarque qui nous amena [Bri+19] à utiliser cet algorithme pour résoudre 3.6.

Dans toute la suite $\mathbf{h}_1, \cdots, \mathbf{h}_{k+\ell}, \mathbf{s} \in \mathbb{F}_3^\ell$ désignera une instance uniformément distribuée du problème 3.6.

3.3.1 Une brève description de l'algorithme de Wagner

Nous allons voir ici qu'avec l'algorithme de Wagner [Wag02] :

Proposition 3.3. *Le problème* SDM$(3, k+\ell, \ell, 3^{\ell/a})$ *peut être résolu en temps moyen* $O(3^{\ell/a})$ *où* a *est le plus grand entier tel que :*

$$3^{\ell/a} \leqslant 2^{(k+\ell)/2^a}$$

On en déduit d'après §3.2.2, la proposition 2.2 et (3.2) avec $p = k + \ell$ le résultat suivant.

Proposition 3.4. *Considérons les paramètres* n, k, w, ℓ *du problème* SD$(n, 3, R, \omega)$ *où* $\omega \stackrel{\triangle}{=} w/n$ *et* $R \stackrel{\triangle}{=} k/n$*. Ce-dernier peut être résolu à un facteur polynomial près en temps,*

$$3^{\ell/a} \cdot \max\left(1, \frac{\min\left(3^{n-k}, \binom{n}{w}2^w\right)}{3^{\ell/a} \cdot 3^{\ell}\binom{n-k-\ell}{w-k-\ell}2^{w-k-\ell}}\right)$$

où a *est le plus grand entier vérifiant,*

$$3^{\ell/a} \leqslant 2^{(k+\ell)/2^a}.$$

Une brève description de l'algorithme. Notre objectif est de trouver L vecteurs $\mathbf{b}^{(j)} = (b_1^{(j)}, \cdots, b_{k+\ell}^{(j)}) \in \{0,1\}^{k+\ell}$ tels que,

$$\forall j \in [\![1, L]\!], \quad \sum_{i=1}^{k+\ell} b_i^{(j)} \mathbf{h}_i = \mathbf{s}.$$

Nous proposons alors de trouver ces L solutions *en temps amorti* $O(1)$, donc en temps $O(L)$ à l'aide de l'algorithme de Wagner [Wag02].

Rappelons que pour appliquer cet algorithme il nous faut introduire le paramètre dit de profondeur :

$$a \in \mathbb{N}^*.$$

Nous définissons ensuite la partition de $[\![1, k+\ell]\!]$:

$$\forall i \in [\![1, 2^a]\!], \quad \mathcal{I}_i \stackrel{\triangle}{=} \left[\!\!\left[1 + \frac{(i-1)(k+l)}{2^a}, \frac{i(k+l)}{2^a}\right]\!\!\right].$$

La première étape de l'algorithme de Wagner consiste à calculer les 2^a listes $(\mathcal{L}_i)_{1 \leqslant i \leqslant 2^a}$ de taille L telles que :

$$\forall i \in [\![1, 2^a]\!],\ \mathcal{L}_i \subseteq \left\{ \sum_{j \in \mathcal{I}_i} b_j \mathbf{h}_j \ :\ \forall j \in \mathcal{I}_i,\ b_j \in \{0,1\} \right\} \quad \text{et} \quad |\mathcal{L}_i| = L.$$

Il est maintenant clair que :

$$L \leqslant 2^{(k+\ell)/2^a} \quad (|\mathcal{I}_i| = (k+\ell)/2^a) \tag{3.10}$$

Nous allons ici faire a étapes de fusions deux à deux des listes sur ℓ/a symboles comme décrit dans §2.2.2. Afin d'assurer un temps amorti 1 il nous faut cependant choisir L tel que :

$$\frac{L^2}{3^{\ell/a}} = L \iff L = 3^{\ell/a}.$$

En combinant cette équation avec (3.10) nous obtenons la contrainte :

$$3^{\ell/a} \leqslant 2^{(k+\ell)/2^a} \tag{3.11}$$

et nous retrouvons la proposition 3.3.

Nous comparons dans la figure 3.4 les exposants asymptotiques de Prange et l'ISD où nous utilisons l'algorithme de Wagner (après minimisation sur le paramètre ℓ) pour $R = 1/2$ en fonction de w/n. Comme nous pouvons l'observer le gain sur l'exposant de Prange est significatif.

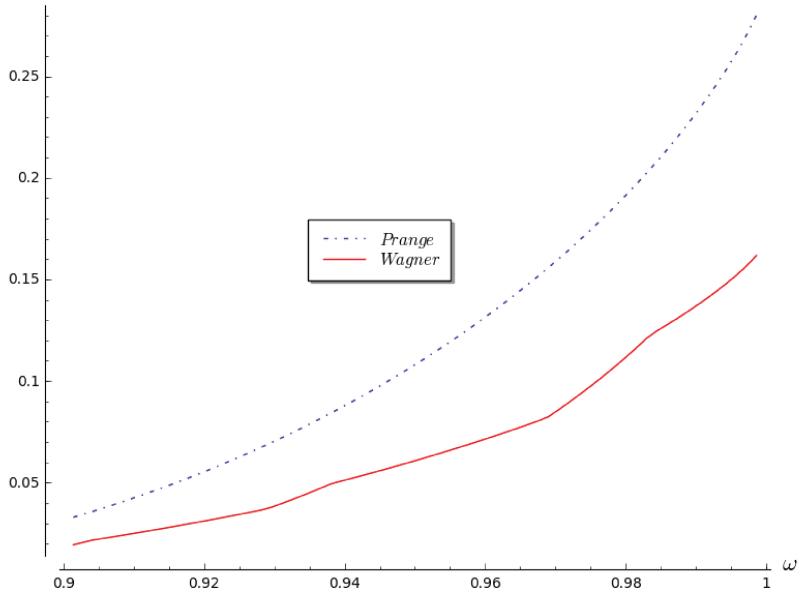


Figure 3.4 – Exposants asymptotiques de la complexité de l'algorithme de Prange et l'ISD où est utilisé Wagner pour $R = 1/2$ en fonction de $\omega \stackrel{\triangle}{=} w/n$ avec $q = 3$.

3.3.2 Un lissage de l'algorithme de Wagner

Nous venons de montrer comment utiliser l'algorithme de Wagner pour trouver L solutions de SDM$(3, k+\ell, \ell, L)$ en temps amorti $O(1)$ pour $L = 3^{\ell/a}$. Supposons maintenant que nous souhaitions plus de L solutions, nous pouvons répéter l'algorithme et toujours les trouver en temps amorti $O(1)$. De cette façon, plus L est petit, meilleur sera l'algorithme. C'est pour cette raison que nous choisissions dans la proposition 3.3 le plus grand entier a tel que $3^{\ell/a} < 2^{(k+\ell)/2^a}$. En revanche, cela induit une discontinuité. La valeur optimale de a qui doit être entière évolue de façon discontinue. C'est ce que nous observons dans la figure 3.4. Les ruptures de pente correspondent à des changements de l'entier a optimal. Nous montrons ici comment atténuer la discontinuité de a comme nous l'avions fait dans §2.2.2.

Proposition 3.5. *Soit a le plus grand entier tel que :*

$$3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}.$$

Si $a \geqslant 3$, le problème SDM$(3, k+\ell, \ell, 2^\lambda)$ *peut être résolu en temps $O(2^\lambda)$ où*

$$\lambda \triangleq \frac{\ell \log_2(3)}{a-2} - \frac{k+\ell}{(a-2)2^{a-1}}.$$

Nous retrouvons bien le résultat de la proposition 3.3 lorsque $3^{\ell/a} = 2^{(k+\ell)/2^a}$.

Démonstration de la prosposition 3.5.

Soit a le plus grand entier tel que $3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}$ que nous supposons $\geqslant 3$. Nous considérons l'algorithme de Wagner décrit dans la sous-section précédente avec une profondeur de a. En revanche, nous allons changer le nombre de symboles sur lesquels nous opérons les fusions. Supposons que nous construisons au départ de l'algorithme des listes de taille maximale :

$$2^{\frac{k+\ell}{2^a}}.$$

Nous faisons alors une fusion sur m bits. Afin d'obtenir des listes de taille 2^λ en temps $O(2^\lambda)$ nous choisissons m tel que :

$$\frac{\left(2^{(k+\ell)/2^a}\right)^2}{3^m} = 2^\lambda \iff \frac{2(k+\ell)}{2^a} - m\log_2(3) = \lambda. \tag{3.12}$$

Nous continuons ensuite les $(a-1)$ autres fusions de façon à obtenir à chaque étape 2^λ solutions en temps amorti 1. Nous opérons donc les fusions sur $\lambda/\log_2(3)$ symboles. Cependant, à la fin de l'algorithme pour obtenir des solutions il nous faut avoir fusionné sur tous les symboles (au nombre de ℓ). Ainsi m et λ doivent vérifier :

$$m + (a-1)\frac{\lambda}{\log_2(3)} = \ell. \tag{3.13}$$

En combinant les équations (3.12) et (3.13) nous obtenons finalement :

$$\lambda = \frac{\ell \log_2(3)}{a-2} - \frac{k+\ell}{(a-2)2^{a-1}}.$$

Il est alors facile de vérifier que sous les conditions $3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}$ et $a \geqslant 3$ que λ et m sont positifs ce qui conclut la preuve. □

On en déduit alors la proposition suivante.

Proposition 3.6. *Considérons pour les paramètres w, k, ℓ le problème* SD$(n, 3, R, \omega)$ *où $\omega \triangleq w/n$ et $R \triangleq k/n$. Ce dernier peut être résolu à un facteur polynomial près en temps :*

$$2^\lambda \cdot \max\left(1, \frac{\min\left(3^{n-k}, \binom{n}{w}2^w\right)}{2^\lambda \cdot 3^\ell \binom{n-k-\ell}{w-k-\ell} 2^{w-k-\ell}}\right)$$

où

$$\lambda = \frac{\ell \log_2(3)}{a-2} - \frac{k+\ell}{(a-2)2^{a-1}}$$

et a est le plus grand entier tel que :

$$3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}$$

que l'on suppose $\geqslant 3$.

Nous comparons dans la figure 3.5 les exposants asymptotiques de Prange et l'ISD où nous avons utilisé Wagner lissé et non lissé (après minimisation sur le paramètre ℓ) pour $R = 1/2$ en fonction de w/n.

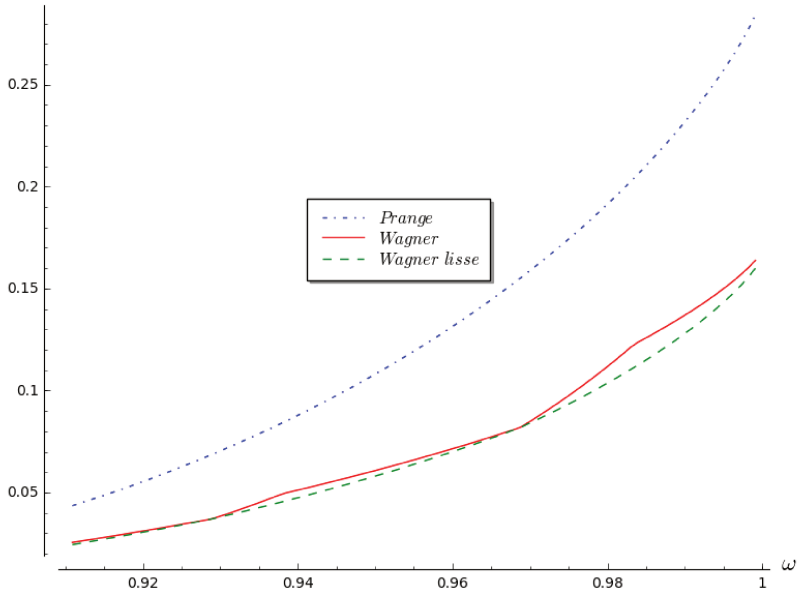


Figure 3.5 – Exposant asymptotiques de la complexité de l'algorithme de Prange et les ISD où est utilisé Wagner et Wagner lissé pour $R = 1/2$ en fonction de $\omega \triangleq w/n$ avec $q = 3$.

3.3.3 Et le problème DOOM ?

Nous pouvons parfaitement adapter l'algorithme de Wagner pour résoudre le problème DOOM. L'idée est la suivante, partant des paramètres a et ℓ nous prenons $N = 3^{\ell/a}$ syndromes. Nous formons alors la liste de ces syndromes en bas à droite de la structure arborescente de l'algorithme. De cette façon, nous n'avons plus qu'à construire $2^a - 1$ listes. La contrainte $3^{\ell/a} \leqslant 2^{(k+\ell)/2^a}$ de la proposition 3.4 sur les paramètres de l'algorithme devient alors :

$$3^{\ell/a} \leqslant 2^{(k+\ell)/(2^a-1)}.$$

Pour nos régimes de paramètres nous avons $a = 6$ ou 7. Le changement de 2^a à $2^a - 1$ a donc un impact négligeable sur la complexité.

3.4 Utilisation de la méthode des représentations

Nous présentons dans cette section notre algorithme pour améliorer l'ISD utilisant Wagner. Nous reprenons dans le contexte de $\mathbb{F}_3$ et du décodage en grande distance l'idée de la technique des représentations [HJ10 ; BCJ11 ; Bec+12] que nous avons décrite dans §2.3.3.

Notre algorithme a la même structure arborescente que celui de Wagner : nous commençons par créer 2^a listes (les feuilles de l'arbre) puis nous les fusionnons deux à deux sur un certain nombre de symboles. Nous recommençons ensuite ces fusions $a - 1$ fois. La différence étant que nous n'allons pas toujours construire les listes à partir de vecteurs à supports disjoints mais avec la technique des représentations. Nous détaillerons dans la sous-section §3.4.4 les différents paramètres de notre algorithme pour résoudre :

$$\mathrm{SD}(n, 3, R, \omega) \quad \text{où} \quad R \triangleq 0.676 \quad \text{et} \quad \omega \triangleq 0.948366.$$

De plus, nous traçons dans la figure 3.6 les exposants asymptotiques de Prange et des

ISD pour $R = 1/2$ en grande distance avec les algorithmes de Dumer, Wagner non lissé, lissé et notre algorithme utilisant la technique des représentations.

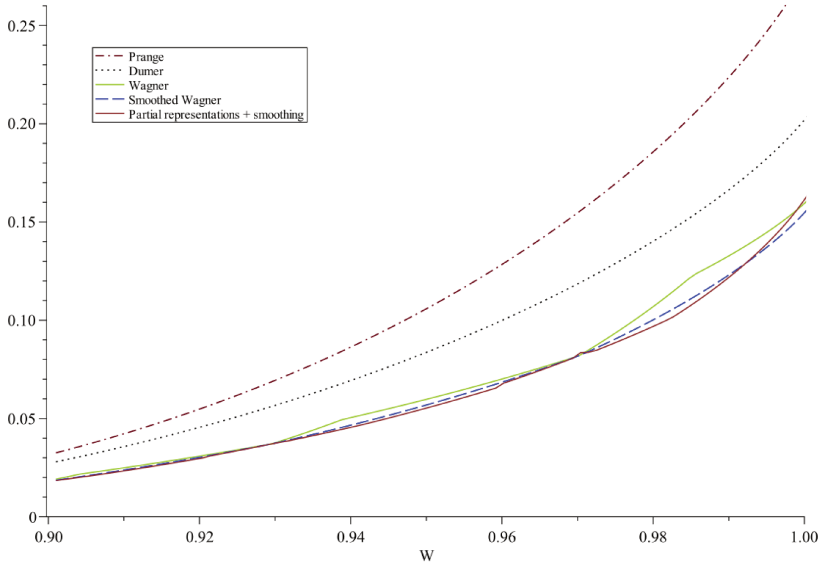


Figure 3.6 – Comparaisons des différents exposants asymptotiques des algorithmes pour résoudre le problème du décodage ternaire en fonction de $\omega \stackrel{\triangle}{=} w/n$ pour $R = 1/2$.

3.4.1 Les représentations modulo 3

Dans l'algorithme de Wagner tout comme celui de Dumer, nous fusionnions deux à deux des listes selon le principe *du découpage en deux gauche-droite*.

Nous obtenions par définition à chaque fusion une liste de la forme :

$$\mathcal{L} \subseteq \left\{ \sum_{j \in [\![A,B]\!]} b_j \mathbf{h}_j : b_j \in \{0,1\} \text{ et } |\mathbf{b}| = p \right\} \tag{3.14}$$

Contrairement à l'algorithme que nous avons décrit précédemment il y a ici une contrainte p sur le poids des vecteurs $\mathbf{b}$. Cela ne change pas notre analyse si p est choisi comme $\frac{B-A+1}{2}$ car nous pouvons atteindre tous les vecteurs sauf une proportion polynomiale. Nous ajoutons cette contrainte car elle nous permettra d'utiliser la méthode des représentations que nous présentons un peu plus loin.

La liste $\mathcal{L}$ était elle obtenue à partir des deux listes $\mathcal{L}_1$ et $\mathcal{L}_2$:

$$\mathcal{L}_1 \subseteq \left\{ \sum_{j \in [\![A,\lfloor \frac{B+A}{2} \rfloor]\!]} b_j \mathbf{h}_j : b_j \in \{0,1\} \text{ et } |\mathbf{b}| = p/2 \right\}$$

$$\mathcal{L}_2 \subseteq \left\{ \sum_{j \in [\![\lfloor \frac{B+A}{2} \rfloor + 1,B]\!]} b_j \mathbf{h}_j : b_j \in \{0,1\} \text{ et } |\mathbf{b}| = p/2 \right\}.$$

comme $\mathcal{L} \subseteq \{\mathbf{s}_1 + \mathbf{s}_2 : \mathbf{s}_i \in \mathcal{L}_i\}$ en faisant une collision sur un sous-ensemble de symboles. De plus, par construction si :

$$\mathbf{s}_1 \stackrel{\triangle}{=} \sum_{j \in [\![A,\lfloor \frac{B+A}{2} \rfloor]\!]} b_j^{(1)} \mathbf{h}_j \quad \text{et} \quad \mathbf{s}_2 \stackrel{\triangle}{=} \sum_{j \in [\![\lfloor \frac{B+A}{2} \rfloor + 1,B]\!]} b_j^{(2)} \mathbf{h}_j$$

nous déduisons $\mathbf{s} = \mathbf{s}_1 + \mathbf{s}_2$ à partir du vecteur :

$$\mathbf{b} \stackrel{\triangle}{=} (\mathbf{b}_1, \mathbf{0}) + (\mathbf{0}, \mathbf{b}_2).$$

qui est bien de poids p et dont les composantes ne sont que des 0 et 1.

L'idée de la technique des représentations est alors de suivre la méthode de Wagner de fusion de listes *mais* en permettant plus de possibilités dans l'écriture des vecteurs $\mathbf{b}$ de poids p. Nous allons écrire $\mathbf{b}$ comme une somme de vecteurs $\mathbf{b}_1 + \mathbf{b}_2$ où cette fois aucune plage de 0 n'est imposée. En revanche, nous remplaçons l'ancienne contrainte par une autre, moins restrictive : nous fixons p_1 le nombre de 1 et p_2 le nombre de 2 ($\mathbb{F}_3 = \{0, 1, 2\}$). Nous illustrons la situation dans la figure 3.7 où $p = 4$, $p_1 = 3$ et $p_2 = 1$. Formellement, nous définissons les représentations ternaires d'un vecteur comme :

Définition 3.1 (Représentations ternaires d'un vecteur)**.** *Soient* $\mathbf{b} \in \mathbb{F}_3^n$ *et* $p_1, p_2 \in [\![1, n]\!]$. *On appelle* (p_1, p_2)*-représentation de* $\mathbf{b}$ *tout couple de* $(\mathbf{b}_1, \mathbf{b}_2) \in \mathbb{F}_3^n$ *tel que :*

$$\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$$

où

$$\#\{i : \mathbf{b}_1(i) = 1\} = \#\{i : \mathbf{b}_2(i) = 1\} = p_1$$

et

$$\#\{i : \mathbf{b}_1(i) = 2\} = \#\{i : \mathbf{b}_2(i) = 2\} = p_2$$

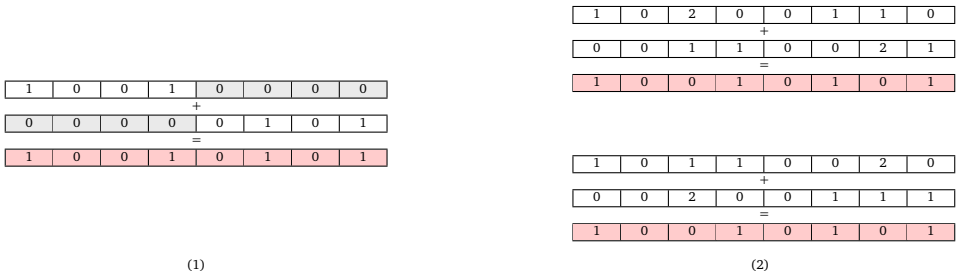


Figure 3.7 – Le même vecteur (1) écrit par découpage en deux et (2) utilisant des représentations ternaires.

Dans ce qui suit la notation suivante nous sera utile :

Notation 3. *On désignera par* $T(n, \alpha_1, \alpha_2)$ *l'ensemble des vecteurs* $\mathbf{b} \in \mathbb{F}_3^n$ *tels que :*

$$\#\{i : b_i = 1\} = \alpha_1/n \quad \textit{et} \quad \#\{i : b_i = 2\} = \alpha_2/n.$$

Avec les représentations nous considèrerons maintenant pour construire $\mathcal{L}$ (voir (3.14)) deux listes :

$$\mathcal{L}'_1, \mathcal{L}'_2 \subseteq \left\{ \sum_{j \in [\![A,B]\!]} b_j \mathbf{x}_j : \#\{j : b_j = 1\} = p_1, \#\{j : b_j = 2\} = p_2 \right\} \tag{3.15}$$

pour deux entiers p_1 et p_2. Nous les fusionnerons ensuite sur des sous-ensembles de symboles comme dans l'algorithme de Wagner.

Cette approche peut sembler au premier abord surprenante et inhabituelle. En effet, pour des valeurs p_1 et p_2 le vecteur $\mathbf{s}_1 + \mathbf{s}_2$ solution de la collision ne sera pas toujours obtenu à partir d'un vecteur de poids désiré p. De plus, ses composantes peuvent être différents de 0 et de 1. Nous parlons alors *d'éléments mal formés*. Ces éléments ne peuvent alors pas être utilisés pour la suite de l'algorithme et doivent donc être retirés de la liste de solutions. Cependant, le point positif est qu'un vecteur donné admet de nombreuses représentations. Plus précisément nous avons la proposition suivante :

Proposition 3.7. *Soit* $\mathbf{b} \in T(n, \alpha_0, \beta_0)$. *Nous désignerons par :*

$$\mathrm{Nrep}(\alpha_0, \beta_0, \alpha_1, \beta_1)$$

le nombre de $(n\alpha_1, n\beta_1)$*-représentations de* $\mathbf{b}$. *Nous avons :*

$$\mathrm{Nrep}(\alpha_0, \beta_0, \alpha_1, \beta_1) = \tilde{O}\left(2^{n(g(1-\alpha_0-\beta_0,\overline{x}_{12},\overline{x}_{12})+g(\alpha_0,\overline{x}_{01},\overline{x}_{01})+g(\beta_0,\overline{x}_{02},\overline{x}_{02}))}\right),$$

où :

$$g(n, k_1, k_2) \mapsto n\log_2(n) - k_1\log_2(k_1) - k_2\log_2(k_2) - (n-k_1-k_2)\log_2(n-k_1-k_2)$$

et

$$\begin{aligned}\overline{x}_{01} &= \frac{2\alpha_0+\beta_0-\alpha_1-2\beta_1}{3} + z\\ \overline{x}_{02} &= \frac{\alpha_0+2\beta_0-2\alpha_1-\beta_1}{3} + z\\ \overline{x}_{12} &= z\end{aligned}$$

avec z *la racine réelle de :*

$$\frac{(2\alpha_0+\beta_0-\alpha_1-2\beta_1+3z)(\alpha_0+2\beta_0-2\alpha_1-\beta_1+3z)z}{(1-\alpha_0-\beta_0-2z)(-2\alpha_0-\beta_0+4\alpha_1+2\beta_1-6z)(-\alpha_0-2\beta_0+2\alpha_1+4\beta_1-6z)} = 1.$$

Afin de démontrer cette proposition, rappelons la définition de multinomial.

Définition 3.2. *Nous définissons la multinomiale* $\binom{n}{k_1,\ldots,k_i}$ *comme :*

$$\binom{n}{k_1,\ldots,k_i} \triangleq \frac{n!}{k_1!\ldots k_i!}$$

où $n = k_1 + \cdots + k_i$.

Il est alors clair que :

$$T(n, \alpha_1, \alpha_2) = \binom{n}{\alpha_1 n, \alpha_2 n, (1-\alpha_1-\alpha_2)n}. \tag{3.16}$$

Le lemme suivant (que l'on prouve facilement avec la formule de Stirling) nous sera alors utile.

Lemme 3.2. *Pour* $n \to +\infty$ *et la fonction* g *définie dans la proposition 3.7 :*

$$\binom{n}{k_1, k_2, n-k_1-k_2} = \tilde{O}\left(2^{g(n,k_1,k_2)}\right).$$

On en déduit alors avec (3.16) que,

$$T(n, \alpha_1, \alpha_2) = \tilde{O}\left(2^{ng(1,\alpha_1,\alpha_2)}\right) \tag{3.17}$$

Nous sommes maintenant prêts pour la démonstration de la proposition 3.7.

Démonstration de la proposition 3.7.

Soit $\mathbf{b} \in T(n, \alpha_0, \beta_0)$. Notre objectif est de calculer le nombre de décompositions de $\mathbf{b}$ en deux vecteurs de $T(n, \alpha_1, \beta_1)$. Désignons par $x_{00}, \ldots, x_{22}$ la densité des neufs cas $(0+0, 0+1, \ldots, 2+2)$ comme montré dans la table :

	0	1	2
0	x_{00}	x_{01}	x_{02}
1	x_{10}	x_{11}	x_{12}
2	x_{20}	x_{21}	x_{22}

Désignons par $\mathcal{A}$ l'ensemble des possibilités $(x_{00}, \ldots, x_{22})$ valides.

De plus pour $\mathbf{x} \triangleq (x_{00}, \cdots, x_{22})$, nous noterons :

$$\mathrm{NrepDens}(\mathbf{x})$$

le nombre de décomposer $\mathbf{b}$ comme somme de deux vecteurs aux densités $(x_{00}, \ldots, x_{22})$. Nous avons :

$$\mathrm{NrepDens}(\mathbf{x}) = \binom{n(1-\alpha_0-\beta_0)}{nx_{00}, nx_{12}, nx_{21}} \binom{n\alpha_0}{nx_{01}, nx_{10}, nx_{22}} \binom{n\beta_0}{nx_{02}, nx_{11}, nx_{20}}.$$

En effet, un 0 dans $\mathbf{b}$ peut être décomposé comme $0+0$ (ce qui se produit nx_{00} fois), $1+2$ (nx_{12} fois) ou $2+1$ (nx_{21} fois). Maintenant comme le nombre de 0 dans $\mathbf{b}$ est $n(1-\alpha_0-\beta_0)$, il y a

$$\binom{n(1-\alpha_0-\beta_0)}{nx_{00}, nx_{12}, nx_{21}}$$

façons de choisir la décomposition de chaque 0 de $\mathbf{b}$. Les choix de décomposition de 1 et 2 donnent alors deux autres facteurs similaires.

On en déduit qu'étant donné $\alpha_0, \beta_0, \alpha_1$ et β_1, le nombre de décompositions possibles est :

$$\sum_{(x_{00},\ldots,x_{22})\in\mathcal{A}} \binom{n(1-\alpha_0-\beta_0)}{nx_{00}, nx_{12}, nx_{21}} \binom{n\alpha_0}{nx_{01}, nx_{10}, nx_{22}} \binom{n\beta_0}{nx_{02}, nx_{11}, nx_{20}}.$$

qui est égal à un facteur polynomial près à :

$$\sum_{(x_{00},\ldots,x_{22})\in\mathcal{A}} 2^{n(g(1-\alpha_0-\beta_0,x_{21},x_{12})+g(\alpha_0,x_{01},x_{10})+g(\beta_0,x_{02},x_{20}))}.$$

Ainsi, du fait que le nombre de répartitions $(nx_{00}, \ldots, nx_{22})$ des $0, 1$ et 2 est polynomial en n, cette somme est égale à un facteur polynomial près à :

$$2^{n(g(1-\alpha_0-\beta_0,\overline{x}_{21},\overline{x}_{12})+g(\alpha_0,\overline{x}_{01},\overline{x}_{10})+g(\beta_0,\overline{x}_{02},\overline{x}_{20}))} \tag{3.18}$$

où $(\overline{x}_{00}, \ldots, \overline{x}_{22})$ donne le plus grand terme de la somme, ce que nous nommerons dans ce qui suit le cas typique.

Calcul du cas typique. Une fois donné $\alpha_0, \beta_0, \alpha_1$ et β_1, nous avons les contraintes suivantes pour $x_{00}, \ldots, x_{22}$:

$$\begin{aligned}
x_{00}+x_{01}+x_{02} &= 1-\alpha_1-\beta_1\\
x_{10}+x_{11}+x_{12} &= \alpha_1\\
x_{20}+x_{21}+x_{22} &= \beta_1
\end{aligned}$$

$$\begin{aligned}
x_{00}+x_{10}+x_{20} &= 1-\alpha_1-\beta_1\\
x_{01}+x_{11}+x_{21} &= \alpha_1\\
x_{02}+x_{12}+x_{22} &= \beta_1
\end{aligned}$$

$$\begin{aligned}
x_{00}+x_{12}+x_{21} &= 1-\alpha_0-\beta_0\\
x_{01}+x_{10}+x_{22} &= \alpha_0\\
x_{02}+x_{11}+x_{20} &= \beta_0
\end{aligned}$$

Cependant ces équations ne sont pas indépendantes. Chaque groupe de trois équations implique que $x_{00} + \cdots + x_{22} = 1$. Nous avons finalement deux degrés de liberté et toute solution peut-être écrite comme,

$$\begin{array}{rcccccc}
x_{00} &=& 1-\alpha_0-\beta_0 & - & 2z & & \\
x_{01} &=& \frac{2\alpha_0+\beta_0-\alpha_1-2\beta_1}{3} & + & z & + & w \\
x_{02} &=& \frac{\alpha_0+2\beta_0-2\alpha_1-\beta_1}{3} & + & z & - & w \\
x_{10} &=& \frac{2\alpha_0+\beta_0-\alpha_1-2\beta_1}{3} & + & z & - & w \\
x_{11} &=& \frac{-2\alpha_0-\beta_0+4\alpha_1+2\beta_1}{3} & - & 2z & & \\
x_{12} &=& 0 & + & z & + & w \\
x_{20} &=& \frac{\alpha_0+2\beta_0-2\alpha_1-\beta_1}{3} & + & z & + & w \\
x_{21} &=& 0 & + & z & - & w \\
x_{22} &=& \frac{-\alpha_0-2\beta_0+2\alpha_1+4\beta_1}{3} & - & 2z. & &
\end{array}$$

Ainsi,

$$\mathcal{A} = \{(x_{00}(w,z), \ldots, x_{22}(w,z)) \mid \forall (i,j), x_{ij} \geqslant 0\}.$$

Déterminons w. Nous allons ici démontrer que dans le cas typique les densités sont symétriques,

$$\overline{x}_{01} = \overline{x}_{10}, \quad \overline{x}_{02} = \overline{x}_{20} \quad \text{et} \quad \overline{x}_{12} = \overline{x}_{21}$$

ce qui signifie que,

$$w = 0.$$

Pour cela considérons une paire (w,z) telle que les densités associées $(x_{00}, \ldots, x_{22})$ soient dans $\mathcal{A}$. De plus, notons $(\tilde{x}_{00}, \ldots, \tilde{x}_{22})$ ces densités pour $(0,z)$.

Du fait que $(x_{00}, \ldots, x_{22}) \in \mathcal{A}$, tous les x_{ij} sont positifs ou nuls. Ceci implique que les $\tilde{x}_{ij}$ sont eux-mêmes positifs ou nuls. Par exemple, pour $\tilde{x}_{01}$ nous avons :

$$0 \leqslant \min(x_{01}, x_{10}) = \tilde{x}_{01} - |w| \leqslant \tilde{x}_{01}.$$

Ainsi, $(\tilde{x}_{00}, \ldots, \tilde{x}_{22}) \in \mathcal{A}$ et nous obtenons :

$$\frac{\mathrm{NrepDens}(\tilde{\mathbf{x}})}{\mathrm{NrepDens}(\mathbf{x})} = \frac{2^{n(g(1-\alpha_0-\beta_0,\tilde{x}_{21},\tilde{x}_{12})+g(\alpha_0,\tilde{x}_{01},\tilde{x}_{10})+g(\beta_0,\tilde{x}_{02},\tilde{x}_{20}))}}{2^{n(g(1-\alpha_0-\beta_0,x_{21},x_{12})+g(\alpha_0,x_{01},x_{10})+g(\beta_0,x_{02},x_{20}))}} \tag{3.19}$$

Nous avons l'égalité suivante par définition de g,

$$\begin{aligned}
g(1-\alpha_0-\beta_0,&x_{21},x_{12}) = g(1-\alpha_0-\beta_0, \tilde{x}_{12}+w, \tilde{x}_{12}-w) \\
&= g(1-\alpha_0-\beta_0, \tilde{x}_{12}, \tilde{x}_{12}) + 2\tilde{x}_{21}\left(h_2(1/2+w/\tilde{x}_{12}) - h_2(1/2)\right).
\end{aligned}$$

De même,

$$g(\alpha_0, x_{01}, x_{10}) = g(\alpha_0, \tilde{x}_{01}, \tilde{x}_{10}) + 2\tilde{x}_{01}\left(h_2(1/2+w/\tilde{x}_{01}) - h_2(1/2)\right),$$

$$g(\beta_0, x_{02}, x_{20}) = g(\beta_0, \tilde{x}_{02}, \tilde{x}_{20}) + 2\tilde{x}_{02}\left(h_2(1/2+w/\tilde{x}_{02}) - h_2(1/2)\right).$$

De cette façon, nous pouvons réduire l'équation (3.19) à :

$$\frac{\mathrm{NrepDens}(\tilde{\mathbf{x}})}{\mathrm{NrepDens}(\mathbf{x})} = 2^{2n\left(\tilde{x}_{01}\left(1-h_2\left(\frac{1}{2}+\frac{w}{\tilde{x}_{01}}\right)\right)+\tilde{x}_{02}\left(1-h_2\left(\frac{1}{2}+\frac{w}{\tilde{x}_{02}}\right)\right)+\tilde{x}_{12}\left(1-h_2\left(\frac{1}{2}+\frac{w}{\tilde{x}_{12}}\right)\right)\right)}.$$

D'où :

$$\mathrm{NrepDens}(\mathbf{x}) \leqslant \mathrm{NrepDens}(\tilde{\mathbf{x}})$$

et ces deux quantités sont égales si et seulement si $w = 0$, *i.e* : $\mathbf{x} = \tilde{\mathbf{x}}$.

Déterminons z. Nous devons maintenant trouver z pour terminer le calcul du cas typique. Nous savons que $w = 0$ et donc d'après (3.18), la quantité z maximise l'expression :

$$g(1-\alpha_0-\beta_0, x_{21}, x_{12}) + g(\alpha_0, x_{01}, x_{10}) + g(\beta_0, x_{02}, x_{20}).$$

Remarquons que cette-dernière est, à un facteur additif près, équivalente à

$$-\sum_{i,j} x_{ij} \log_2(x_{ij}).$$

Cette fonction est strictement concave et donc admet un maximum unique. Sa dérivée en fonction de z est donnée par,

$$-2\log_2\left(\frac{\left(\frac{2\alpha_0+\beta_0-\alpha_1-2\beta_1}{3}+z\right)\left(\frac{\alpha_0+2\beta_0-2\alpha_1-\beta_1}{3}+z\right)z}{(1-\alpha_0-\beta_0-2z)\left(\frac{-2\alpha_0-\beta_0+4\alpha_1+2\beta_1}{3}-2z\right)\left(\frac{-\alpha_0-2\beta_0+2\alpha_1+4\beta_1}{3}-2z\right)}\right)$$

qui est égale à 0 si et seulement si :

$$\frac{(2\alpha_0+\beta_0-\alpha_1-2\beta_1+3z)(\alpha_0+2\beta_0-2\alpha_1-\beta_1+3z)z}{(1-\alpha_0-\beta_0-2z)(-2\alpha_0-\beta_0+4\alpha_1+2\beta_1-6z)(-\alpha_0-2\beta_0+2\alpha_1+4\beta_1-6z)} = 1.$$

ce qui conclut la preuve de la proposition. □

Les résultats de [HJ10 ; BCJ11 ; Bec+12] montrent alors dans le cas binaire comment un grand nombre de représentations peut compenser le fait que la plupart des décompositions n'appartiennent pas à la liste de solutions. Plus précisément, lors de la construction d'une liste de solutions avec les représentations, le nombre de symboles sur lequel est opéré la collision est fait de façon à ne garder pour une solution en moyenne qu'une de ses représentations.

3.4.2 Les représentations partielles

Il se peut dans certains cas que le grand nombre de représentations oblige pour ne conserver qu'un représentant par solution à fusionner les listes sur un trop grand nombre de symboles. Dans ce cas, nous proposons pour diminuer le nombre de représentants d'utiliser des *représentations partielles*. Les listes sont cette fois construites de façon intermédiaire entre le découpage en deux utilisé dans l'algorithme de Wagner et les représentations comme illustré dans la figure 3.8.

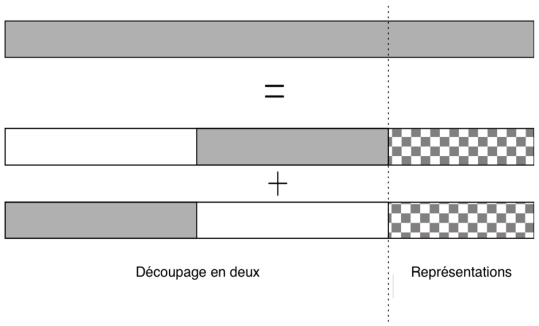


Figure 3.8 – Décomposition d'un vecteur avec des représentations partielles.

3.4.3 Présentation de notre algorithme

L'instanciation de l'algorithme de Wagner avec des représentations peut être fait de multiples façons et soulèvent principalement deux questions : quelles proportions de symboles à 1, 2 et à quel niveau de l'algorithme ? Sur combien de symboles opérer les collisions ? La proposition que nous faisons est alors le résultats de nombreux essais et erreurs. Nous présentons ici les principales caractéristiques de notre algorithme :

— Dans le régime que nous considérons le nombre d'étages de l'algorithme varie entre 5 et 7. Une si grande profondeur est plutôt inhabituelle et est dû au fait que nous avons beaucoup de solutions à notre problème de sac à dos modulaire.

— Du fait que nous cherchons un grand nombre de solutions en temps amorti $O(1)$, la technique des représentations devient moins efficace que le cas où seule une solution est demandée. En effet, il est difficile de choisir les paramètres de façon à ce que les éléments mal formés soient en proportion négligeable.

— Nous allons montrer que les représentations peuvent donner un gain exponentiel. Cependant, afin d'avoir une amélioration dans la plupart des régimes que nous avons considéré il suffit de seulement injecter des représentations dans 2 étages de l'algorithme.

La figure 3.9 illustre un exemple pour une profondeur de $a = 7$. Lorsque nous augmentons le nombre d'étages, nous ajoutons simplement des découpages gauche-droite.

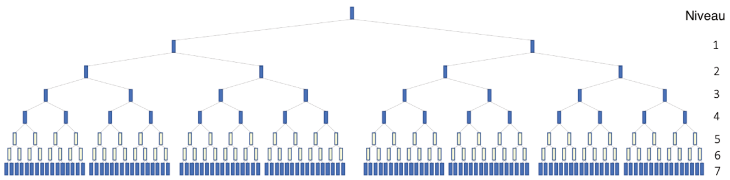


Figure 3.9 – Arbre de Wagner pour une profondeur $a = 7$. Les listes jaunes correspondent à la technique des représentations quant aux bleus à des découpages gauche-droite.

Nous donnons dans la prochaine sous-section les paramètres exactes de cet algorithme dans le contexte du décodage.

3.4.4 Application au décodage générique pour $R = 0.676$ et $\omega = 0.95$

Nous présentons ici les détails de notre algorithme avec les représentations pour résoudre :

$$\text{SD}(n, 3, R, \omega) \quad \text{où} \quad R = 0.676 \quad \text{et} \quad \omega = 0.948366$$

Il s'agit d'un des jeux de paramètre de la signature Wave que nous présentons dans la partie III de ce document. Nous obtenons comme complexité,

$$\tilde{O}\left(2^{0.0176n}\right). \tag{3.20}$$

Nous avons obtenu cette dernière en choisissant entre autres pour les paramètres génériques des ISD,

$$\ell \triangleq 0.060835n \quad \text{et} \quad p \triangleq k + \ell \quad \text{où} \quad k \triangleq \lfloor Rn \rceil \tag{3.21}$$

et nous résolvons le sac à dos modulaire qui suit,

$$\text{SDM}(3, k + \ell, 2^{0.0176n}) \tag{3.22}$$

en temps amorti $O(1)$. Autrement dit, d'après (3.20), l'ISD n'a besoin que d'une itération pour trouver une solution au décodage considéré.

Nous avons avec notre algorithme $a = 7$ étages, ce qui signifie que nous commençons dans la structure arborescente de l'algorithme de Wagner par construire $2^7 = 128$ listes aux feuilles (figure 3.9). Du niveau 0 à 6 les listes sont toutes de même taille L, *i.e :* la complexité pour résoudre le problème de sac à dos considéré, où

$$L \triangleq 2^{0.0176n}.$$

Au niveau 7 de notre algorithme nous construisons 128 listes de taille,

$$2^{0.0139n}.$$

Nous décrivons maintenant en détail comment les listes sont construites à chaque étage de l'algorithme.

— Les niveaux 1 à 4 correspondent à des fusions de listes comme dans l'algorithme de Wagner par découpage en deux. Par exemple, au niveau 4 nous avons $2^4 = 16$ listes de la forme :

$$\forall i \in [\![1, 16]\!],\ \mathcal{L}_i \subseteq \left\{ \sum_{j \in \mathcal{I}_i} b_j \mathbf{h}_j \ :\ \forall j \in \mathcal{I}_i,\ b_j \in \{0, 1\} \right\} \text{ et } |\mathcal{L}_i| = L.$$

avec $\mathcal{I}_i \triangleq [\![1 + \frac{(i-1)(k+\ell)}{16}, \frac{i(k+\ell)}{16}]\!]$.

— La figure 3.10 illustre la discussion qui suit. Au niveaux 5 et 6 nous utilisons des représentations partielles. Entre les niveaux 4 et 5 nous avons utilisé sur une proportion

$$\lambda_1 \triangleq 0.7252 \tag{3.23}$$

de symboles des représentations que nous avons obtenu par découpage gauche-droite à partir du niveau 6. En revanche, sur l'autre proportion de symboles,

$$\lambda_2 \triangleq 1 - \lambda_1 \tag{3.24}$$

nous avons utilisé des représentations sur les deux niveaux. Plus précisément, pour les proportions λ_1 et λ_2 des vecteurs nous avons construit les listes (3.15) avec pour densité de $0, 1$ et 2 :

  — pour la partie avec un niveau de représentation, la densité ρ_1 correspond à 74.8% de 0, 25.1% de 1 et 0.1% de 2,

  — pour l'autre partie avec deux niveaux de représentations, la densité ρ_2 au niveau 5 correspond à 74.2% de 0, 25.4% de 1, 0.4% de 2 et la densité ρ_3 du niveau 6 est de 86.9% de 0, 13.1% de 1 et 0.0% de 2.

— Les listes du niveau 6 sont obtenues par un découpage gauche-droite à partir des listes du niveau 7 de façon à obtenir les densité décrites dans la figure 3.10.

Le choix des différentes densités ainsi que les calculs associés peuvent s'avérer complexes. Nous avons alors utilisé de façon cruciale la proposition 3.7.

Comme nous l'avons expliqué précédemment avec l'utilisation des représentations nous pouvons produire après fusion des listes avec des éléments mal formés. C'est à dire des vecteurs n'ayant pas les bonnes densités de poids. Nous supprimons ces éléments après la création des listes. Le lemme qui suit permet de donner le nombre attendu d'éléments bien formés après la fusion de deux listes avec la technique des représentations.

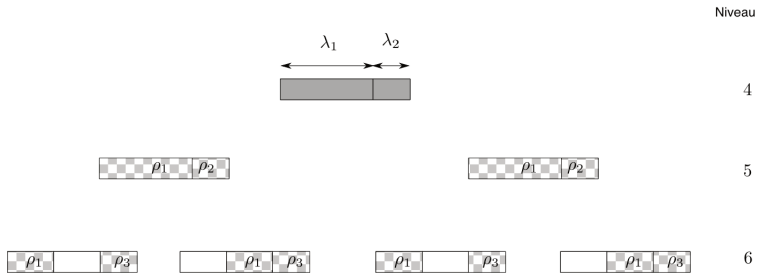


Figure 3.10 – Détail des niveaux où nous utilisons les représentations partielles.

Lemme 3.3. *Soient* $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{F}_3^\ell$ *et les deux listes,*

$$\mathcal{L}_1, \mathcal{L}_2 \subseteq \left\{ \sum_{j=1}^{n} b_j \mathbf{x}_j : \mathbf{b} \in T(n, \alpha_1, \beta_1) \right\} \quad \textit{où} \quad |\mathcal{L}_1| = |\mathcal{L}_2| = L.$$

Notons $\mathcal{L}$ *la collision de ces deux listes sur* $\log_3(L)$ *symboles et* $\mathcal{L}'$ *la liste de ses éléments bien formés selon* $T(n, \alpha_0, \beta_0)$ *:*

$$\mathcal{L}' \stackrel{\triangle}{=} \left\{ \sum_{j=1}^{n} b_j \mathbf{x}_j \in \mathcal{L} : \mathbf{b} \in T(n, \alpha_0, \beta_0) \right\}$$

Alors en moyenne $\mathcal{L}'$ *est de taille :*

$$\tilde{O}\left(L \cdot \frac{\mathrm{Nrep}(\alpha_0, \beta_0, \alpha_1, \beta_1) 2^{ng(1,\alpha_0,\beta_0)}}{2^{2ng(1,\alpha_1,\beta_1)}} \right)$$

où $\mathrm{Nrep}(\alpha_0, \beta_0, \alpha_1, \beta_1)$ *et la fonction* g *sont donnés dans la proposition 3.7.*

Démonstration du lemme 3.3.

Il y a $\tilde{O}\left(2^{ng(1,\alpha_0,\beta_0)}\right)$ vecteurs dans $T(n, \alpha_0, \beta_0)$ d'après le lemme (3.17). Chacun de ces vecteurs admet par définition :

$$\mathrm{Nrep}(\alpha_0, \beta_0, \alpha_1, \beta_1)$$

façons d'être décomposé comme somme d'éléments de $T(n, \alpha_1, \beta_1)$. De plus le nombre de vecteurs dans $T(n, \alpha_1, \beta_1)$ est $\tilde{O}\left(2^{ng(1,\alpha_1,\beta_1)}\right)$. Il y a donc $\tilde{O}\left(2^{2ng(1,\alpha_1,\beta_1)}\right)$ paires de vecteurs de $T(n, \alpha_1, \beta_1)$ mais seulement $\tilde{O}\left(\mathrm{Nrep}(\alpha_0, \beta_0, \alpha_1, \beta_1) 2^{ng(1,\alpha_0,\beta_0)}\right)$ paires de vecteurs de $T(n, \alpha_1, \beta_1)$ se sommant dans $T(n, \alpha_0, \beta_0)$. Toutes les autres paires forment des éléments mal formés. Ainsi, si nous fusionnons deux listes de $T(n, \alpha_1, \beta_1)$ de taille L sur $\log_3(L)$ symboles nous obtiendrons en moyenne :

$$\tilde{O}\left(L \cdot \frac{\mathrm{Nrep}(\alpha_0, \beta_0, \alpha_1, \beta_1) 2^{ng(1,\alpha_0,\beta_0)}}{2^{2ng(1,\alpha_1,\beta_1)}} \right)$$

vecteurs de $T(n, \alpha_0, \beta_0)$, le reste étant des éléments mal formés. □

De cette façon, du fait des éléments mal formés, afin d'obtenir à la fin de l'algorithme une liste de taille $2^{0.0176n}$ nous diminuons le nombre de symboles sur lesquels nous opérons la collision. Dans notre cas, au niveau 4 de l'algorithme le nombre d'éléments bien formés est $2^{0.0116n}$, nous faisons alors la collision sur $\log_3\left(2^{0.0055n}\right)$ symboles au lieu de $\log_3\left(2^{0.0176}\right)$ pour obtenir les listes du niveau 3. Le nombre d'éléments bien formés au niveau 5 est donné par $2^{0.0174n}$. Nous opérons donc au niveau 5 la collision sur

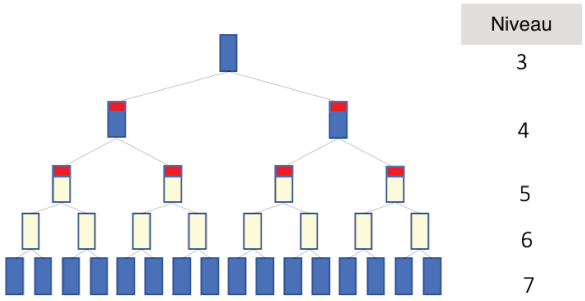


Figure 3.11 – Détail des niveaux 3 à 7 de l'algorithme où les éléments en rouge sont ceux mal formés.

$\log_3\left(2^{0.0173n}\right)$ symboles pour obtenir les listes du niveau 4. Nous illustrons cette discussion avec la figure 3.11.

Au niveau 7, pour obtenir les listes du niveau 6 nous faisons une fusion sur $\log_3\left(2^{0.0032n}\right)$ symboles par découpage en deux avec des listes de taille $2^{0.01039n}$.

En résumé, le nombre d'éléments bien formés par liste du niveau 0 à 7 est :

$$2^{0.0176n}, 2^{0.0176n}, 2^{0.0176n}, 2^{0.0176n}, 2^{0.0116n}, 2^{0.0174n}, 2^{0.0176n}, 2^{0.01039n}$$

et le nombre de symboles sur lesquels nous avons opéré une collision du niveau 1 à 6 est :

$$\log_3\left(2^{0.0176n}\right), \log_3\left(2^{0.0176n}\right), \log_3\left(2^{0.0176n}\right), \log_3\left(2^{0.0055n}\right),$$

$$\log_3\left(2^{0.0173n}\right), \log_3\left(2^{0.0176n}\right), \log_3\left(2^{0.0032n}\right).$$

Nous avons ici utilisé de façon cruciale le lemme 3.3 au niveau des étages 5 et 6 pour obtenir ces tailles de listes en fonction du nombre de symboles sur lesquels nous opérons les collisions. De plus, on peut vérifier que le nombre de symboles sur lesquels nous avons opéré les fusions (somme de ce qui précède) est bien ℓ. Nous obtenons donc bien à la fin de l'algorithme des solutions du problème de sac à doc considéré.

Le nombre de solutions du sac à dos est :

$$\frac{2^{k+\ell}}{3^\ell} = 2^{0.6404n}.$$

Une solution admet quant à elle $2^{0.4915n}$ représentations d'après la proposition 3.7 ce qui donne

$$2^{0.6404n} \times 2^{0.4915n} = 2^{1.1319n}$$

possibilités. On peut de plus vérifier qu'avec les fusions nous filtrons $2^{1.1143n}$ vecteurs solutions. Nous obtenons donc à la fin de l'algorithme :

$$\frac{2^{1.1319n}}{2^{1.1143n}} = 2^{0.0176n}$$

solutions.

3.5 Les instances les plus difficiles du décodage ternaire

Nous avons présenté dans les sections précédentes nos algorithmes pour le décodage à grande distance. Nous allons désormais nous intéresser aux paramètres du décodage où le problème est le plus difficile avec ces algorithmes.

Le problème du décodage à grande distance dans $\mathbb{F}_3$ est bien plus difficile que son pendant binaire en petit poids comme nous l'avons mentionné dans l'introduction. Dans l'exemple que nous avons considéré, à savoir $R = 0.676$, les algorithmes étaient le moins efficaces pour $\omega = 1$. En revanche pour des rendement plus faibles ceci n'est plus le cas. Notons :

$$R_0 \triangleq 1 - \log_3(2) \approx 0.36907.$$

Par définition de ω^+ nous avons si et seulement si $R \in [0, R_0]$:

— SD$(n, 3, R, \omega^+)$ admet typiquement une solution,

— SD$(n, 3, R, \omega)$ admet en moyenne un nombre exponentiellement faible de solutions où $\omega \in]\omega^+, 1]$.

La quantité R_0 est le rendement limite à partir duquel (strictement) nous nous attendons toujours à un nombre exponentiel de solutions en ω^+ qui vaut alors 1. Ceci explique pourquoi dans la figure 3.6 la complexité ne diminue pas avec ω à partir d'un certain seuil. Notons que pour le rendement $R = 1/5 < R_0$, où nous avons tracé dans la figure 3.2 l'exposant de notre généralisation de l'algorithme de Prange, nous observons bien cette diminution de l'exposant à partir de ω^+.

Nous savons d'après la proposition 3.2 que l'algorithme de Prange est le moins efficace en $\omega = 1$ et $R = R_0$ dès que $q \geqslant 3$. C'est aussi ce que nous avons observé numériquement pour nos deux algorithmes. Nous comparons dans la table 3.3 les pires exposants de ces derniers avec ceux des algorithmes de Prange [Pra62], Dumer [Dum91] et [Bec+12] dans le cas du décodage binaire à petite distance. Dans tous les cas nous notons pour quels rendements et distances ces exposants sont atteints. Dans le cas binaire la distance relative est donnée par la borne relative de Gilbert-Varshamov.

Table 3.3 – Pires exposants en base 2 avec rendement et distance associés.

Algorithme	$q = 2$ et $\omega = \omega^-$	$q = 3$, $R = 0.369$ et $\omega = 1$
Prange	0.121 ($R = 0.454$)	0.369
Dumer/Wagner	0.116 ($R = 0.447$)	0.269
BJMM/notre algorithme	0.102 ($R = 0.427$)	0.247

Que ce soit dans le cas binaire ou ternaire, nous constatons que l'algorithme de Prange est peu efficace. L'algorithme de Dumer donne quant à lui une première amélioration exponentielle mais c'est l'algorithme de BJMM qui donne le meilleur gain relatif. Cet algorithme utilise les méthodes de l'état de l'art pour résoudre le problème du sac à dos, la technique des représentations et $1 + 1 = 0$ dans le cas binaire. L'analyse des algorithmes de Prange et Dumer pour $q = 3$ se fait tout aussi facilement que dans le cas binaire. Concernant l'équivalent de BJMM en ternaire, (*i.e :* l'algorithme de Wagner avec des représentations), nous avons obtenu l'exposant 0.247 avec une profondeur de 2 et un niveau de représentations. Nous avons essayé un arbre plus profond avec un autre équilibre de représentations mais cela donna toujours de moins bons exposants.

Le décodage ternaire apparaît donc bien plus difficile que son pendant binaire. Cela était attendu, les exposants de décodage étant moins bon dans le cas ternaire (même à ω^-). Cela peut s'expliquer entre autre car une matrice de parité (ou génératrice) d'un code sur $\mathbb{F}_3$ contient bien plus d'informations qu'un code sur $\mathbb{F}_2$ de même dimension. En revanche, pour une sécurité donnée, les tailles de clef dans le cas ternaire étaient jusqu'à nos récents travaux bien plus élevées. Notons :

$$\alpha(q, R, \omega)$$

l'exposant asymptotique en base 2 d'un algorithme $\mathcal{A}$ résolvant SD(n, q, R, ω). Si nous représentons maintenant une matrice de parité donnée en entrée de SD(n, q, R, ω) sous forme systématique, cette dernière sera de taille :

$$T(q, R, \omega) \triangleq \log_2(q) \cdot n^2 R(1-R) \quad \text{où} \quad n \triangleq \left\lceil \frac{128}{\alpha(q, R, \omega)} \right\rceil$$

pour une sécurité de 128 bits relativement à $\mathcal{A}$. Nous cherchons alors à trouver les paramètres (R, ω) minimisant pour q :

$$T(q, R, \omega).$$

ce qui donnera la plus petite taille de clef espérée pour crypto-système dont la sécurité repose sur SD dans le corps $\mathbb{F}_q$. Nous donnons dans la table 3.4 le résultat de cette minimisation pour $q = 2$ et $q = 3$ en fonction de nos algorithmes et ceux de Dumer, Prange et BJMM.

Table 3.4 – Taille de clef minimale (en Kbits) pour une complexité de résolution en temps 2^{128}.

Algorithme	$q = 2$ et $\omega = \omega^-$	$q = 3$, $R = 0.369$ et $\omega = 1$
Prange	275 ($R = 0.384$)	44
Dumer/Wagner	295 ($R = 0.369$)	83
BJMM/our algorithm	374 ($R = 0.326$)	99

Notons que contrairement au cas ternaire dans le cas binaire les plus petites tailles de clef ne sont pas obtenues pour les rendements où le décodage est le plus difficile (voir table 3.3).

Ce travail, bien que préliminaire, ouvre donc de nouvelles perspectives pour la cryptographie utilisant des codes correcteurs en métrique de Hamming. Il semble que nous aurions tout intérêt en terme de taille de clef à trouver des primitives, allant des chiffrements aux signatures et les fonctions de hachage, dont la sécurité repose sur le décodage générique à grande distance dans $\mathbb{F}_q$ où $q > 2$.

Chapitre 4

Décodage statistique

Introduction

Le problème du décodage générique est étudié depuis maintenant près de 60 ans. Les efforts furent nombreux et nous pouvons citer à titre d'exemple les algorithmes proposés dans [Pra62 ; Ste88 ; Leo88 ; LB88 ; Dum91 ; MMT11 ; Bec+12 ; MO15 ; BM18]. Toutes ces solutions reposent néanmoins sur une même idée issue de l'algorithme de Prange [Pra62] et de Dumer [Dum89] comme nous l'avons décrit dans la partie I, chapitre 2 : tirage de positions contenant un ensemble d'information puis décodage d'un code poinçonné avec des techniques plus ou moins sophistiquées (paradoxe des anniversaires, représentations, voisin le plus proche). Nous parlons usuellement d'algorithmes ISD pour *Information Set Decoding*.

Il existe cependant un algorithme n'appartenant pas à la famille des ISD : le décodage statistique proposé par [Jab01] puis légèrement amélioré dans [Ove06]. Une version itérative de l'algorithme fut même proposée dans [FKI07].

L'idée du décodage statistique est la suivante. Donnons-nous un $[n,k]_2$-code $\mathcal{C}$, un mot de code bruité $\mathbf{y} \stackrel{\triangle}{=} \mathbf{c} + \mathbf{e}$ où $\mathbf{c} \in \mathcal{C}$, $|\mathbf{e}| = w$ et une équation de parité $\mathbf{h} \in \mathcal{C}^\perp$. Par définition :

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{c} + \mathbf{e}, \mathbf{h} \rangle = \langle \mathbf{e}, \mathbf{h} \rangle.$$

Supposons maintenant que $h_{i_0} \neq 0$, nous avons :

— Si $e_{i_0} = 1$:

$$\langle \mathbf{e}, \mathbf{h} \rangle = 1 \iff \#\operatorname{Supp}(\mathbf{e}) \cap \operatorname{Supp}(\mathbf{h}) \backslash \{i_0\} \text{ est } \textit{pair}$$

— Si $e_{i_0} = 0$:

$$\langle \mathbf{e}, \mathbf{h} \rangle = 1 \iff \#\operatorname{Supp}(\mathbf{e}) \cap \operatorname{Supp}(\mathbf{h}) \backslash \{i_0\} \text{ est } \textit{impair}$$

La probabilité avec $\mathbf{h}$ que le produit scalaire $\langle \mathbf{e}, \mathbf{h} \rangle = \langle \mathbf{y}, \mathbf{h} \rangle$ vaille 1 est donc dépendante de la présence ou non d'une erreur en le bit de position i_0. Cette remarque est à l'origine de la proposition de [Jab01]. Le décodage statistique est alors un algorithme en deux étapes opérant un test statistique par vote majoritaire sur la valeur de $\langle \mathbf{h}, \mathbf{y} \rangle$:

1. On commence par calculer un ensemble d'équations de parité d'un même poids t :

$$\mathcal{S} \subseteq \left\{ \mathbf{h} \in \mathcal{C}^\perp : |\mathbf{h}| = t \text{ et } h_{i_0} = 1 \right\}.$$

2. Selon la valeur du compteur :

$$C \stackrel{\triangle}{=} \sum_{\mathbf{h} \in \mathcal{S}} \langle \mathbf{h}, \mathbf{e} \rangle \tag{4.1}$$

on décide que e_{i_0} égale 0 ou non.

La question est alors comment prendre la bonne décision dans l'étape 2 ? Pour cela notons X_b la variable aléatoire :

$$X_b \stackrel{\triangle}{=} \langle \mathbf{h}, \mathbf{e} \rangle \quad \text{selon l'hypothèse que} \quad e_{i_0} = b$$

l'aléa étant sur les équations $\mathbf{h} \in \mathcal{C}^\perp$ *que l'on suppose uniformément distribuées parmi les mots de poids* t *dont le bit en position* i_0 *est égal à* 1. Ce modèle n'aura de sens que si le poids de l'erreur w du décodage considéré vérifie :

$$w \leqslant w_{\text{GV}} \tag{4.2}$$

de façon à nous assurer que pour une entrée $\mathbf{y} \in \mathbb{F}_2^n$ il existe avec une suffisamment bonne probabilité une unique décomposition $\mathbf{y} = \mathbf{c} + \mathbf{e}$ où $\mathbf{c} \in \mathcal{C}$ et $|\mathbf{e}| = w$. Notons maintenant pour $b \in \{0, 1\}$ la quantité ε_b définie comme :

$$\mathbb{E}(X_b) = 1/2 + \varepsilon_b.$$

Les espérances $1/2 + \varepsilon_0$ et $1/2 + \varepsilon_1$ selon que $e_{i_0} = 0$ ou 1 sont alors *différentes*. La valeur attendue du compteur $C = \sum X_b$ défini dans (4.1) diffèrera donc avec la présence d'une erreur ou non en position i_0 : $|\mathcal{S}| \times (1/2 + \varepsilon_b)$ si $e_{i_0} = b$. La décision se fera alors selon que C est plus proche de $|\mathcal{S}| \times (1/2 + \varepsilon_0)$ ou $|\mathcal{S}| \times (1/2 + \varepsilon_1)$. Dans ce contexte, la borne de Chernoff nous apprend que la taille de l'échantillon d'équations de parité :

$$P_t \stackrel{\triangle}{=} |\mathcal{S}|$$

doit être suffisamment grande, de l'ordre de :

$$P_t = \tilde{O}\left(\frac{1}{(\varepsilon_0 - \varepsilon_1)^2}\right) \tag{4.3}$$

pour prendre la bonne décision avec une probabilité exponentiellement proche de 1.

En résumé, le décodage statistique consiste tout simplement à calculer P_t (vérifiant (4.3)) équations de parité de poids t puis d'en déduire C pour prendre une décision. Plus précisément, nous aurons la proposition suivante (nous définissons les hypothèses 1 et 2 dans §4.1 et §4.1.1).

Proposition 4.1. *Supposons que pour tout* $i_0 \in [\![1, n]\!]$ *nous disposons d'un algorithme calculant* N_t *équations de parité* $\mathbf{h}$ *de poids* t *avec* $h_{i_0} = 1$ *en temps* T_t*. Alors, sous les hypothèses 1 et 2, le problème du décodage binaire à distance* $w \leqslant w_{\text{GV}}$ *peut-être résolu en temps :*

$$\tilde{O}\left(P_t + T_t \cdot \max\left(1, \frac{P_t}{N_t}\right)\right).$$

où $P_t = 1/(\varepsilon_0 - \varepsilon_1)^2$.

Cette proposition soulève alors deux questions si nous souhaitons déterminer la complexité asymptotique du décodage statistique :

1. Quelle est l'expression asymptotique de P_t ?
2. Comment calculer des équations de parité de poids t ?

Bien qu'étant essentielle, les travaux [Jab01 ; Ove06] ne donnèrent pas la formule asymptotique du nombre P_t d'équations de parité nécessaires au décodage statistique. En revanche, [FKI07] donna une telle formule mais dans un modèle simplifié où les équations de parité $\mathbf{h} \in \mathbb{F}_2^n$ sont des variables aléatoires telles que :

$$\text{les bits } h_i \text{ sont indépendants} \quad \text{et} \quad \mathbb{P}(h_i = 1) = \frac{t}{n}.$$

Ce modèle peut sembler raisonnable du fait que le code est supposé aléatoire et que $\mathbb{E}(|\mathbf{h}|) = t$. En revanche, comme nous le verrons §4.3 ce dernier ne capture pas la complexité du décodage statistique. Concernant le second point, les papiers [Jab01 ; Ove06] proposèrent d'utiliser une variation l'algorithme de Stern [Ste88].

Notre contribution [DT17] fut de clarifier ces questions en donnant trois résultats. Nous avons tout d'abord déterminé l'expression asymptotique de P_t en utilisant le développement asymptotique des polynômes de Krawtchouk [MS86, Ch. 2. §2] déterminé dans [IS98]. Notre expression de P_t s'avéra être d'une remarquable simplicité pour une large classe de paramètres. Nous avons en particulier prouvé que le nombre d'équations de parité de poids $t \triangleq \lfloor \tau n \rfloor$ où τ est constant, nécessaires au décodage statistique pour retrouver une erreur de poids $w \triangleq \lfloor \omega n \rfloor$ avec ω constant est dès-lors que $\tau \geqslant \frac{1}{2} - \sqrt{\omega(1-\omega)}$ exponentiel et donné par :

$$\tilde{O}\left(2^{n(h_2(\omega)+h_2(\tau)-1)}\right).$$

Cette formule montre entre autres que lorsque nous cherchons à décoder un $[n,k]_2$-code aléatoire à la distance relative de Gilbert-Varshmov, *ie* : $h_2(\omega) = 1 - k/n$, le décodage statistique réclame à un facteur polynomial près toutes les $\binom{n}{t}/2^k$ équations de parité de poids t existant typiquement dans un code aléatoire.

Avec la formule asymptotique de P_t en poche, il ne nous reste plus qu'à décrire des algorithmes permettant de trouver des équations de parité de poids t pour en déduire avec la proposition 4.1 une complexité du décodage statistique. Nous avons proposé d'utiliser comme décrit dans §4.5 les techniques algorithmiques des ISD. Nous proposons en particulier l'algorithme de Prange permettant de calculer $\tilde{O}(P_t)$ équations de parité en temps $\tilde{O}(P_t)$ pour $t = 1 + k/2$. Nous en avons déduit les complexités suivantes du décodage statistique en fonction de la distance relative de décodage $\omega \triangleq w/n$ et du rendement $R \triangleq k/n$ du code :

1. Si $\omega = \omega^-$, la complexité est donnée par :
$$2^{n\cdot(h_2(R/2)-R)(1+o(1))}.$$
2. Si $\omega = o(1)$, la complexité est donnée par :
$$2^{-2w(\log_2(1-R)+o(1))}.$$

Dans le premier cas nous pouvons vérifier que le décodage statistique est de complexité supérieure à celle de l'algorithme de Prange. L'idée pour l'améliorer est alors de chercher à calculer des équations de parité de poids relatif $< R/2$. En effet, on peut vérifier que le nombre d'équations nécessaires pour prendre une bonne décision diminue avec t décroissant. Nous avons alors proposé d'utiliser une variation de l'algorithme de Dumer [Dum91] permettant de calculer des équations de parité de poids relatif fixé $< R/2$ en temps amorti polynomial. Concernant le second point, rappelons que tous les ISD sont de même complexité [CS16a] pour $\omega = o(1)$,

$$2^{-w\log_2(1-R)(1+o(1))}.$$

Le décodage statistique utilisant l'algorithme de Prange pour trouver des équations de parité fait donc moins bien que tous les ISD pour $\omega = o(1)$. Nous n'avons malheureusement pas réussi à l'améliorer. En effet, le problème étant que nous devons trouver un algorithme calculant une équation de poids relatif $< R/2$ en temps sous-exponentiel ou tout du moins un nombre sous-exponentiel en temps amorti polynomial. Au mieux de nos connaissances nous ne connaissons pas d'algorithmes réalisant ceci.

Finalement, bien que le décodage statistique dépende de l'algorithme calculant des équations de parité, nous donnons comme troisième résultat une borne inférieure de

sa complexité. Nous avons alors montré que ce décodage ne peut pas être plus efficace que l'algorithme de Prange pour le décodage à la distance de Gilbert-Varshamov. Cette borne provient du fait que la complexité du décodage statistique sera toujours supérieure au nombre P_t d'équations de parité de poids t nécessaires. Nous prouvons alors que le minimum de la fonction $t \mapsto P_t$ est toujours supérieur à la complexité de l'algorithme de Prange quand nous cherchons à décoder à w_{GV}.

Le problème étudié et quelques notations

Dans tout ce chapitre nous considèrerons la version duale du décodage par syndrome (équivalente en terme de complexité) qui rappelons-le est défini comme,

Problème 1.1. (Décodage générique.) *Étant donné* $\mathbf{G} \in \mathbb{F}_q^{k \times n}$*,* w *et* $\mathbf{y} \in \mathbb{F}_q^n$*, trouver* $\mathbf{e}$ *de poids de Hamming* exactement w *tel que* $\mathbf{y} - \mathbf{e} = \mathbf{mG}$ *pour un certain* $\mathbf{m} \in \mathbb{F}_q^k$*.*

Nous nous restreindrons dans notre étude au cas binaire,

$$q = 2.$$

La distance de décodage w vérifiera quant à elle,

$$w \leqslant w_{\text{GV}}(2, n, k). \tag{4.4}$$

De plus nous supposerons dans toute la suite que $\mathbf{G}$ désigne la variable aléatoire,

$$\mathbf{G} \hookleftarrow \mathbb{F}_2^{k \times n}.$$

Implicitement $\mathcal{C}$ dénotera le code de matrice génératrice $\mathbf{G}$ et réciproquement. La variable aléatoire $\mathbf{y}$ sera distribuée quant à elle comme :

$$\mathbf{y} \stackrel{\triangle}{=} \mathbf{mG} + \mathbf{e} \quad \text{où} \quad \mathbf{m} \hookleftarrow \mathbb{F}_2^k \quad \text{et} \quad \mathbf{e} \hookleftarrow S_{w,n}.$$

Les notations suivantes nous seront utiles :

Notation 4. *Soient* $n \in \mathbb{N}$*,* $t \in [\![0, n]\!]$*,* $p \in [0, 1]$ *et* $\mathcal{C}$ *un code binaire de longueur* n *:*

- $S_{t,i} \stackrel{\triangle}{=} \{\mathbf{x} \in S_t : x_i = 1\}$*,*
- $\mathcal{H}_t(\mathcal{C}) \stackrel{\triangle}{=} \mathcal{C}^\perp \cap S_t$*,*
- $\mathcal{H}_{t,i}(\mathcal{C}) \stackrel{\triangle}{=} \mathcal{C}^\perp \cap S_{t,i}$*,*
- $\text{Ber}(p)$ *la loi de Bernoulli de paramètre* p*.*

4.1 Le décodage statistique

L'analyse du décodage statistique se fera dans tout ce chapitre comme dans [Ove06] sous l'hypothèse qui suit.

Hypothèse 1. *La distribution du produit scalaire* $\langle \mathbf{h}, \mathbf{y} \rangle$ *pour* $\mathbf{h} \hookleftarrow \mathcal{H}_{t,i}(\mathcal{C})$ *est approchée par la distribution de* $\langle \mathbf{h}, \mathbf{y} \rangle$ *pour* $\mathbf{h} \hookleftarrow S_{t,i}$*.*

Un modèle plus élémentaire est donné dans [FKI07] : les produits scalaires $\langle \mathbf{h}, \mathbf{y} \rangle$ sont supposés tels que les bits de $\mathbf{h}$ sont indépendants et distribuées comme des Bernoulli de paramètre t/n. Ceci présente l'avantage de rendre l'analyse du décodage statistique bien plus simple. En revanche, comme nous le verrons dans §4.3, ce modèle mène à une estimation de complexité sous-évaluée.

4.1.1 Un biais obtenu avec des équations de parité

Nous commençons l'analyse du décodage statistique par introduire les notations suivantes.

Notation 5. *Soient $n \in \mathbb{N}$, $w, t \in [\![0, n]\!]$ et $\mathbf{e} \in \mathbb{F}_2^n$ de poids w. Nous notons les probabilité pour $\mathbf{h} \sim S_{t,i}$,*

$$q_1(\mathbf{e}, t, i) = \mathbb{P}_{\mathbf{h}}\left(\langle \mathbf{e}, \mathbf{h} \rangle = 1\right) \text{ quand } e_i = 1$$

$$q_0(\mathbf{e}, t, i) = \mathbb{P}_{\mathbf{h}}\left(\langle \mathbf{e}, \mathbf{h} \rangle = 1\right) \text{ quand } e_i = 0$$

Il s'agit des probabilités approximant sous l'hypothèse 1 celles auxquelles nous nous intéressons (quand les $\mathbf{h}$ sont tirées dans le dual du code et non uniformément). Il est alors clair que :

$$q_1(\mathbf{e}, t, i) = \frac{\sum\limits_{j \text{ pair}}^{t-1} \binom{w-1}{j}\binom{n-w}{t-1-j}}{\binom{n-1}{t-1}} \quad \text{et} \quad q_0(\mathbf{e}, t, i) = \frac{\sum\limits_{j \text{ impair}}^{t-1} \binom{w}{j}\binom{n-w-1}{t-1-j}}{\binom{n-1}{t-1}} \tag{4.5}$$

Ces probabilités sont indépendantes de la position i. De cette façon, en omettant la dépendance en t, nous noterons tout simplement dans la suite q_1 et q_0. Introduisons désormais les biais ε_1 et ε_0 (dépendant évidemment de t) du décodage statistique,

$$q_0 = \frac{1}{2} + \varepsilon_0 \quad \text{et} \quad q_1 = \frac{1}{2} + \varepsilon_1 \tag{4.6}$$

Il s'avèrera, ce qui est essentiel, que $\varepsilon_1 \neq \varepsilon_0$. Nous pourrons alors utiliser cette différence "comme un distingueur". Il s'agit du coeur du décodage statistique qui n'est rien d'autre qu'un test entre les hypothèses,

$$\mathcal{H}_0 \ : \ e_i = 0 \quad \text{et} \quad \mathcal{H}_1 \ : \ e_1 = 1$$

reposant sur le calcul de la variable aléatoire V_m pour m tirages indépendants de vecteurs $\mathbf{h}^1, \cdots, \mathbf{h}^m \in \mathcal{H}_{t,i}(\mathcal{C})$ définie comme,

$$V_m \stackrel{\triangle}{=} \sum_{j=1}^{m} \text{signe}(\varepsilon_1 - \varepsilon_0) \cdot \langle \mathbf{y}, \mathbf{h}^j \rangle \in \mathbb{Z}.$$

Or d'après (4.5) et (4.6) nous avons :

$$\langle \mathbf{y}, \mathbf{h}^j \rangle \sim \text{Ber}(1/2 + \varepsilon_b) \quad \text{selon l'hypothèse } \mathcal{H}_b \text{ qui est faite.}$$

De cette façon, la valeur attendue,

$$E_b \stackrel{\triangle}{=} \mathbb{E}(V_m)$$

est donnée sous l'hypothèse $\mathcal{H}_b$ par,

$$E_b = m \times \text{signe}(\varepsilon_1 - \varepsilon_0)(1/2 + \varepsilon_b).$$

Remarquons que le terme $\text{signe}(\varepsilon_1 - \varepsilon_0)$ permet d'affirmer $E_1 > E_0$. Afin de pouvoir appliquer la borne de Chernoff nous ferons à partir de maintenant l'hypothèse suivante,

Hypothèse 2. *Les $\langle \mathbf{y}, \mathbf{h}^j \rangle$ sont des variables aléatoires indépendantes où $\mathbf{h}_i \hookleftarrow \mathcal{H}_{t,i}(\mathcal{C})$ pour $1 \leqslant j \leqslant m$.*

Proposition 4.2 (Borne de Chernoff). *Soient* $0 < p < 1$, $Y_1, \cdots, Y_m$ *indépendantes et distribuées comme* Ber(p). *Soit* $Z_m \stackrel{\triangle}{=} \sum_{k=1}^{m} Y_k$. *Alors,*

$$\forall \delta \geqslant 0, \quad \mathbb{P}\left(|Z_m - mp| \geqslant m\delta\right) \leqslant 2e^{-2m\delta^2}.$$

Conséquence : Sous l'hypothèse $\mathcal{H}_b$, nous avons :

$$\mathbb{P}\left(|V_m - m \times \text{signe}(\varepsilon_1 - \varepsilon_0) \cdot (1/2 + \varepsilon_b)| \geqslant m \cdot \frac{|\varepsilon_1 - \varepsilon_0|}{2}\right) \leqslant 2 \cdot 2^{-m \times \frac{(\varepsilon_1 - \varepsilon_0)^2}{2\ln(2)}}.$$

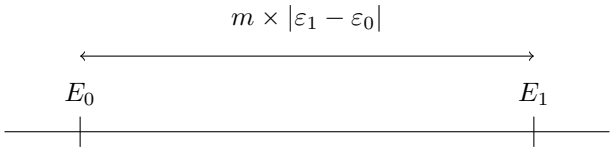
Pour prendre notre décision nous procédons de la façon suivante :

— Si $V_m < \frac{E_0 + E_1}{2}$ alors nous choisissons $\mathcal{H}_0$,

— Sinon nous choisissons $\mathcal{H}_1$.

où :

$$\frac{E_1 + E_0}{2} = \frac{m}{2}\text{signe}(\varepsilon_1 - \varepsilon_0)(1 + \varepsilon_1 + \varepsilon_0) \quad \text{et} \quad \frac{E_1 - E_0}{2} = m \times \frac{|\varepsilon_1 - \varepsilon_0|}{2}.$$



Nous en déduisons alors d'après la proposition 4.2 qu'avec m équations de parité de poids t,

$$\mathbb{P}\left(\text{nous choisissons } b'\right) \leqslant 2 \times 2^{-m \times \frac{(\varepsilon_1 - \varepsilon_0)^2}{2\ln(2)}} \quad \text{sous l'hypothèse } \mathcal{H}_b \text{ où } b \neq b'.$$

Il est donc essentiel de choisir m de l'ordre de :

$$m = \Omega\left(\frac{n}{(\varepsilon_1 - \varepsilon_0)^2}\right)$$

de façon à ce que nous fassions un mauvais choix avec une probabilité négligeable. Comme nous le verrons dans §4.2, $1/(\varepsilon_1 - \varepsilon_0)^2$ sera en n :

— exponentiel pour w/n et t/n constants,

— sous-exponentiel pour $w/n = o(1)$ et t/n constant.

Une décision optimale ? Tous les arguments que vous venons de donner pour trouver un distingueur sur une position erronée ou non peuvent sembler assez grossiers. Nous pouvons alors légitimement nous demander s'il existe un meilleur procédé. Autrement dit, peut-on diminuer la taille de l'échantillon de façon à ne se tromper qu'avec une probabilité négligeable ? Il s'avère dans notre cas où w et t sont linéaires en n que non, le terme $\tilde{O}\left(\frac{1}{(\varepsilon_0 - \varepsilon_1)^2}\right)$ est du bon ordre. Cette réponse est donnée par la théorie des types [CT91] dans le contexte des larges déviations et plus particulièrement par le théorème de Sanov [CT91] qui nous apprend que l'exposant de la borne de Chernoff est asymptotiquement optimale lorsque le biais, ici $\varepsilon_1 - \varepsilon_0$, est exponentiellement faible. Notre test statistique ne peut donc être amélioré.

Cependant, pour améliorer notre distingueur nous pourrions tout aussi bien utiliser une connaissance plus fine des hypothèses $\mathcal{H}_0$ et $\mathcal{H}_1$. En effet, nous savons que $\mathbb{P}(e_i = 1) = \frac{w}{n}$ et donc la probabilité sur l'erreur que $\mathcal{H}_0$ ou $\mathcal{H}_1$ soit vérifiée. Il s'avère encore une fois de plus que non, utiliser des tests d'hypothèses Bayesien se fondant sur les probabilités a priori ne change pas dans notre cas l'ordre de grandeur de tests $\tilde{O}\left(\frac{1}{(\varepsilon_1 - \varepsilon_0)^2}\right)$ quand t et w sont linéaires en n.

4.1.2 L'algorithme du décodage statistique

L'algorithme de décodage statistique consiste tout simplement à utiliser le distingueur que nous venons de décrire avec $\tilde{O}\left(\frac{1}{(\varepsilon_1-\varepsilon_0)^2}\right)$ équations de parité de poids t. Ce nombre dépend évidemment de n, w et t mais introduisons la notation suivante qui nous sera utile dans toute la suite.

Notation 6. $P_t \stackrel{\triangle}{=} \frac{1}{(\varepsilon_1-\varepsilon_0)^2}$.

Nous présentons le décodage statistique dans l'algorithme 2 où $\texttt{ParEq}_t(\mathbf{G}, i)$ désigne un algorithme calculant $\tilde{O}(P_t)$ éléments de $\mathcal{H}_{t,i}(\mathcal{C})$ avec $\mathcal{C}$ le code de matrice génératrice $\mathbf{G}$.

Algorithme 2 `DecoStat` : **Décodage Statistique**

Entrée : $\mathbf{G} \in \mathbb{F}_2^{k\times n}, \mathbf{y} \in \mathbb{F}_2^n$ et $t \in [\![0, n]\!]$
Sortie : $\mathbf{e}$
for $i = 1 \cdots n$ **do**
  $\mathcal{S}_i \leftarrow \texttt{ParEq}_t(\mathbf{G}, i)$
  $V_i \leftarrow 0$
  for all $h \in \mathcal{S}_i$ **do**
    $V_i \leftarrow V_i + \text{signe}(\varepsilon_1 - \varepsilon_0) \cdot \langle \mathbf{y}, \mathbf{h} \rangle$
  if $V_i < \text{signe}(\varepsilon_1 - \varepsilon_0) P_t \frac{1+\varepsilon_1+\varepsilon_0}{2}$ **then**
    $e_i \leftarrow 0$
  else
    $e_i \leftarrow 1$
return $\mathbf{e}$

Nous en déduisons alors facilement la proposition suivante.

Proposition 4.1. *Supposons que pour tout $i_0 \in [\![1, n]\!]$ nous disposons d'un algorithme calculant N_t équations de parité $\mathbf{h}$ de poids t avec $h_{i_0} = 1$ en temps T_t. Alors, sous les hypothèses 1 et 2, le problème du décodage binaire à distance $w \leqslant w_{\text{GV}}$ peut-être résolu en temps :*

$$\tilde{O}\left(P_t + T_t \cdot \max\left(1, \frac{P_t}{N_t}\right)\right).$$

où $P_t = 1/(\varepsilon_0 - \varepsilon_1)^2$.

Comme nous l'avons expliqué dans l'introduction, notre objectif est de calculer la complexité asymptotique du décodage statistique et donc en premier lieu de déterminer l'expression asymptotique de P_t. Fixons pour cela dans ce qui suit les trois notations qui suivent.

Notation 7.

$$\omega \stackrel{\triangle}{=} \frac{w}{n} \quad \textit{et} \quad \tau \stackrel{\triangle}{=} \frac{t}{n}.$$

$$\pi(\omega, \tau) \stackrel{\triangle}{=} \lim_{n\to+\infty} \frac{1}{n} \log_2(P_t).$$

Il peut sembler intrigant que nous nous intéressions ici à l'exposant asymptotique de P_t comme une borne inférieure. Cela est dû au fait que $\frac{1}{n}\log_2(P_t)$ ne converge pas pour certains poids w d'erreurs et t d'équations de parité. Cette quantité est, comme nous le verrons dans la prochaine section, proportionnelle aux valeurs prises par un polynôme de Krawtchouk et pour certains paramètres w et t ces dernières sont proches de ses zéros (second cas du théorème 4.1).

4.2 Équivalent asymptotique des polynômes de Krawtchouk

Notre objectif dans cette section est de démontrer le théorème suivant.

Théorème 4.1. *Nous avons :*

— *Si* $\omega \in \left]0, \frac{1}{2} - \sqrt{\tau(1-\tau)}\right[$ *:*

$$\pi(\omega,\tau) = 2\tau\log_2(r) - 2\omega\log_2(1-r) - 2(1-\omega)\log_2(1+r) + 2h_2(\tau)$$

où r *est la plus petite racine de* $(1-\tau)X^2 - (1-2\omega)X + \tau = 0$.

— *Si* $\omega \in \left]\frac{1}{2} - \sqrt{\tau(1-\tau)}, \frac{1}{2}\right[$ *:*

$$\pi(\omega,\tau) = h_2(\omega) + h_2(\tau) - 1.$$

La démonstration de ce théorème reposera de façon cruciale sur les polynômes de Krawtchouk dont nous rappelons la définition.

Définition 4.1 (Polynômes de Krawtchouk)**.** *Soient* $m \in \mathbb{N}$ *et* $v \in [\![0, m]\!]$. *Le polynôme de Krawtchouk d'ordre* m *et de degré* v *est défini comme :*

$$p_v^m(X) = \frac{(-1)^v}{2^v}\sum_{j=0}^{v}(-1)^j\binom{X}{j}\binom{m-X}{v-j}$$

où

$$\binom{X}{j} = \frac{1}{j!}X(X-1)\cdots(X-j+1).$$

Ces polynômes sont alors reliés à notre biais $\varepsilon_1 - \varepsilon_0$ comme montré dans le lemme qui suit.

Lemme 4.1. *Nous avons :*

$$\varepsilon_1 = -\frac{(-2)^{t-2}}{\binom{n-1}{t-1}}p_{t-1}^{n-1}(w-1) \quad \textit{et} \quad \varepsilon_0 = \frac{(-2)^{t-2}}{\binom{n-1}{t-1}}p_{t-1}^{n-1}(w).$$

Démonstration du lemme 4.1.

$$\begin{aligned}
-\frac{(-2)^{t-2}}{\binom{n-1}{t-1}}p_{t-1}^{n-1}(w-1) &= \frac{\sum_{j=0}^{t-1}(-1)^j\binom{w-1}{j}\binom{n-w}{t-1-j}}{2\binom{n-1}{t-1}} \\
&= \frac{\sum_{\substack{j=0\\ j \text{ pair}}}^{t-1}\binom{w-1}{j}\binom{n-w}{t-1-j} - \sum_{\substack{j=1\\ j \text{ impair}}}^{t-1}\binom{w-1}{j}\binom{n-w}{t-1-j}}{2\binom{n-1}{t-1}} \\
&= \frac{2\sum_{\substack{j=0\\ j \text{ pair}}}^{t-1}\binom{w-1}{j}\binom{n-w}{t-1-j} - \binom{n-1}{t-1}}{2\binom{n-1}{t-1}} \\
&= \varepsilon_1 \quad \text{(voir (4.5) et (4.6))} \qquad (4.7)
\end{aligned}$$

Un calcul similaire donne le résultat pour ε_0. □

Or d'après [IS98] nous connaissons le développement asymptotique des polynômes de Krawtchouk.

Notation 8. *Suivons les conventions de [IS98] concernant les nombres complexes :* $\mathbf{i}$ *désigne l'imaginaire pur vérifiant l'équation* $\mathbf{i}^2 = -1$*,* $\Re(z)$ *la partie réelle d'un complexe* z *et* $\Im(z)$ *sa partie imaginaire. Le logarithme complexe est quant à lui défini comme,*

$$\ln(z) \triangleq \ln|z| + \mathbf{i}\arg(z), \quad z \in \mathbb{C}\backslash]-\infty, 0]$$

où $\arg(z) \in [-\pi, \pi]$ *est l'argument d'un complexe.*

Théorème 4.2 ([IS98, théorème 3.1])**.** *Soient* m, v *et* s *trois entiers. Posons* $\nu \triangleq \frac{v}{m}, \alpha \triangleq \frac{1}{\nu}$ *et* $\sigma \triangleq \frac{s}{m}$*. Nous supposons que* $\alpha \geqslant 2$*. Soit :*

$$p(z) \triangleq \log_2 z - \frac{\sigma}{\nu}\log(1+z) - \left(\alpha - \frac{\sigma}{\nu}\right)\log_2(1-z).$$

L'équation $p'(z) = 0$ *a pour solutions* x_1 *et* x_2 *qui sont les racines de l'équation :*

$$(\alpha-1)X^2 + (\alpha - 2\frac{\sigma}{\nu})X + 1 = 0.$$

Soient $D \triangleq \left(\alpha - 2\frac{\sigma}{\nu}\right)^2 - 4(\alpha-1)$ *et* $\Delta \triangleq \alpha - \frac{2\sigma}{\nu}$*. Les deux racines* x_1 *et* x_2 *sont alors égales à* $\frac{-\Delta \pm \sqrt{D}}{2(\alpha-1)}$ *où* x_1 *est défini comme* $\frac{-\Delta+\sqrt{D}}{2(\alpha-1)}$*. Il y a alors deux cas à considérer :*

— *Si* $\frac{\sigma}{\nu} \in]0, \alpha/2 - \sqrt{\alpha-1}[$*, alors* D *est positif,* x_1 *est un nombre réel négatif et nous pouvons écrire :*

$$p_v^m(s) = Q_{\sigma,\nu}(v) 2^{-(p(x_1)+1)v} \tag{4.8}$$

où $Q_{\sigma,\nu}(v) \triangleq -\sqrt{\frac{1-r^2}{2\pi r D v}}(1 + O(v^{-1/2}))$ *avec* $r \triangleq -x_1$*.*

— *Si* $\frac{\sigma}{\nu} \in [\alpha/2 - \sqrt{\alpha-1}, \alpha/2]$*,* D *est négatif,* x_1 *est un nombre complexe et nous avons :*

$$p_v^m(s) = R_{\sigma,\nu}(v)\Im\left(\frac{2^{-(p(x_1)+1)v}}{x_1\sqrt{2p"(x_1)}}(1+\delta(v))\right) \tag{4.9}$$

où $\delta(v)$ *désigne une fonction égale à un* $o(1)$ *uniformément en* v*, et*

$$R_{\sigma,\nu}(v) \triangleq \frac{1 + O(v^{-1/2})}{\sqrt{\pi v}}.$$

Les formules données sont vraies par convergence uniforme sur tout compact inclue dans les intervalles correspondant.

Remarque 4.1. Notons que l'équation (4.8) est incorrectement formulée dans [IS98, théorème. 3.1]. Le problème étant que l'équation (3.20) de [IS98] est fausse. En effet, les quantités $p"(-r_1)$ et $p^{(3)}(-r_1)$ sont négatives et prendre leur racine carrée mène dans (3.20) à des complexes purs. La réparation est cependant aisée du fait que l'expression juste au dessus de (3.20) est correcte. Il suffit alors de choisir la partie imaginaire pour obtenir (4.8).

La preuve du théorème 4.1 est une conséquence de ce théorème. Cette dernière est technique et repose sur trois lemmes et un corollaire que nous décrivons maintenant jusqu'à la fin de cette section.

Fixons les notations suivantes qui nous seront d'une grande aide.

Notation 9.

$$\begin{aligned} m &\triangleq n-1 \\ v &\triangleq t-1 \\ \nu &\triangleq \frac{v}{m} \\ \alpha &\triangleq \frac{1}{\nu} \\ \sigma_0 &\triangleq \frac{w}{m} \\ \sigma_1 &\triangleq \frac{w-1}{m} \end{aligned}$$

et pour $i \in \{0,1\}$ nous introduisons les quantités,

$$\begin{aligned} p_i(z) &\triangleq \log_2 z - \frac{\sigma_i}{\nu}\log(1+z) - \left(\alpha - \frac{\sigma_i}{\nu}\right)\log_2(1-z) \\ \Delta_i &\triangleq \alpha - \frac{2\sigma_i}{\nu} \\ D_i &\triangleq \left(\alpha - 2\frac{\sigma_i}{\nu}\right)^2 - 4(\alpha-1) \\ z_i &\triangleq \frac{-\Delta_i + \sqrt{D_i}}{2(\alpha-1)} \end{aligned}$$

Nous commençons notre preuve par le lemme suivant.

Lemme 4.2. *Nous avons,*

$$\frac{\varepsilon_0}{\varepsilon_1} = -\frac{1+z_1}{1-z_1}\left(1 + O(t^{-1/2}\right).$$

Démonstration du lemme 4.2.

D'après le lemme 4.1,

$$\frac{\varepsilon_0}{\varepsilon_1} = -\frac{p_{t-1}^{n-1}(w)}{p_{t-1}^{n-1}(w-1)} \tag{4.10}$$

En injectant maintenant dans cette équation les expressions asymptotiques des polynômes de Krawtchouk données dans le théorème 4.2 nous obtenons :

$$\begin{aligned} \frac{\varepsilon_0}{\varepsilon_1} &= -\frac{Q_{\sigma_0,\nu}(v)2^{-p_0(z_0)v}}{Q_{\sigma_1,\nu}(v)2^{-p_1(z_1)v}} \\ &= -\frac{Q_{\sigma_0,\nu}(v)}{Q_{\sigma_1,\nu}(v)}2^{(p_1(z_1)-p_0(z_0))v}. \end{aligned} \tag{4.11}$$

Par définition $\sigma_1 = \sigma_0 - \frac{1}{m}$ et $z_1 = z_0 + O\left(\frac{1}{m}\right)$. De cette façon, dû à la forme particulière de $Q_{\sigma_i,\nu}(v)$ dans les deux cas nous en déduisons que,

$$\frac{Q_{\sigma_0,\nu}(v)}{Q_{\sigma_1,\nu}(v)} = 1 + O(v^{-1/2}). \tag{4.12}$$

Observons maintenant que,

$$\begin{aligned} \frac{\sigma_1}{\nu} - \frac{\sigma_0}{\nu} &= \frac{w-1}{v} - \frac{w}{v} \\ &= -\frac{1}{v}. \end{aligned}$$

De cette façon,

$$\begin{aligned}(p_1(z_1)-p_0(z_0))v &= \Big(\log_2(z_1)-\frac{\sigma_1}{\nu}\log_2(1+z_1)-(\alpha-\frac{\sigma_1}{\nu})\log_2(1-z_1)\\ &\quad -\log_2(z_0)+\frac{\sigma_0}{\nu}\log_2(1+z_0)+(\alpha-\frac{\sigma_0}{\nu})\log_2(1-z_0)\Big)v\\ &= \big(\log_2\frac{z_1}{z_0}-\frac{\sigma_0}{\nu}\log_2\frac{1+z_1}{1+z_0}-(\alpha-\frac{\sigma_0}{\nu})\log_2\frac{1-z_1}{1-z_0}\\ &\quad +\frac{1}{v}\log_2(1+z_1)-\frac{1}{v}\log_2(1-z_1)\big)v\\ &= \left(\log_2\frac{z_1}{z_0}-\frac{\sigma_0}{\nu}\log_2\frac{1+z_1}{1+z_0}-(\alpha-\frac{\sigma_0}{\nu})\log_2\frac{1-z_1}{1-z_0}\right)v\\ &\quad +\log_2\frac{1+z_1}{1-z_1} \qquad (4.13)\end{aligned}$$

Écrivons maintenant le terme : $\log_2\frac{z_1}{z_0}-\frac{\sigma_0}{\nu}\log_2\frac{1+z_1}{1+z_0}-(\alpha-\frac{\sigma_0}{\nu})\log_2\frac{1-z_1}{1-z_0}$ comme,

$$\log_2\frac{z_1}{z_0}-\frac{\sigma_0}{\nu}\log_2\frac{1+z_1}{1+z_0}-(\alpha-\frac{\sigma_0}{\nu})\log_2\frac{1-z_1}{1-z_0}=p_0(z_1)-p_0(z_0).$$

Maintenant, par définition $p_0'(z_0)=0$ et $z_1=z_0+\delta$ où $\delta=O(1/m)$. De cette manière,

$$p_0(z_1)-p_0(z_0)=p_0(z_0+\delta)-p_0(z_0)=O(\delta^2)=O(1/m^2).$$

Injecter cette équation (4.13) puis dans (4.11) termine la preuve du lemme. □

Nous en déduisons, entre autres du lemme qui précède, le résultat suivant.

Lemme 4.3. *Supposons $\alpha\geqslant 2$ et $\frac{\sigma_i}{\nu}\in(0,\alpha/2-\sqrt{\alpha-1})$ pour $i\in\{0,1\}$. Nous avons*

$$\varepsilon_0-\varepsilon_1=(-1)^v\sqrt{\frac{(1+z_1)(1-\nu)}{(z_1-z_1^2)D_1}}2^{-\left(p(z_1)+\frac{h_2(\nu)}{\nu}\right)v}(1+O(v^{-1/2})).$$

Démonstration du lemme 4.3.

Calculons,

$$\begin{aligned}\varepsilon_0-\varepsilon_1 &= -\varepsilon_1\frac{1+z_1}{1-z_1}(1+O(v^{-1/2}))-\varepsilon_1\\ &= -\varepsilon_1\left(\frac{1+z_1}{1-z_1}(1+O(v^{-1/2})+1\right)\\ &= -2\varepsilon_1\left(\frac{1}{1-z_1}+O\left(v^{-1/2}\frac{1+z_1}{1-z_1}\right)\right)\\ &= -\frac{2\varepsilon_1}{1-z_1}(1+O(v^{-1/2}) \quad \text{(come } -1\leqslant z_i\leqslant 0)\\ &= -\frac{(-2)^{v-1}\sqrt{2\pi v(1-\nu)}}{2^{\frac{vh_2(\nu)}{\nu}}}Q_{\sigma_1,\nu}(v)2^{-(p(z_1)+1)v}\frac{2}{1-z_1}(1+O(v^{-1/2})) \qquad (4.14)\\ &= \frac{(-2)^v\sqrt{2\pi v(1-\nu)}}{2^{\frac{vh_2(\nu)}{\nu}}}\sqrt{\frac{1-z_1^2}{-2\pi z_1D_1v}}2^{-(p(z_1)+1)v}\frac{2}{1-z_1}(1+O(v^{-1/2}))\\ &= (-1)^v\sqrt{\frac{(1+z_1)(1-\nu)}{(z_1-z_1^2)D_1}}2^{-\left(p(z_1)+\frac{h_2(\nu)}{\nu}\right)v}(1+O(v^{-1/2})) \qquad (4.15)\end{aligned}$$

où nous avons utilisé dans (4.14) :

$$\begin{aligned}\binom{m}{v} &= \frac{2^{mh_2(\nu)}}{\sqrt{2\pi m\nu(1-\nu)}}\\ &= \frac{2^{\frac{vh_2(\nu)}{\nu}}}{\sqrt{2\pi v(1-\nu)}}\end{aligned}$$

□

Le second cas du théorème 4.2, correspondant à $\frac{\sigma_i}{\omega} \in]\alpha/2 - \sqrt{\alpha-1}, \alpha/2[$, est capturé par le lemme qui suit (c'est justement le terme "sin" apparaissant dans ce lemme qui nous a amené à définir $\pi(\omega,\tau)$ comme une limite inférieure et non une limite).

Lemme 4.4. *Pour* $\frac{\sigma_i}{\omega} \in]\alpha/2 - \sqrt{\alpha-1}, \alpha/2[$ *et* $i \in \{0,1\}$ *nous avons,*

$$\varepsilon_1 - \varepsilon_0 = \frac{(-1)^v\sqrt{1-\nu}}{\left|(z_0 - z_0^2)\sqrt{{p_0}"(z_0)}\right|} 2^{-v\left(\Re(p_0(z_0)) + \frac{h_2(\nu)}{\nu}\right)} \sin\left(v\theta - \theta_0 + o(1)\right)\left(1+o(1)\right)$$

où $\theta \stackrel{\triangle}{=} \arg\left(2^{-p_0(z_0)}\right)$ *et* $\theta_0 \stackrel{\triangle}{=} \arg\left((z_0 - z_0^2)\sqrt{{p_0}"(z_0)}\right)$.

Démonstration du lemme 4.4.

La preuve de ce lemme est similaire à celle du lemme 4.2. D'après le lemme 4.1,

$$\varepsilon_1 - \varepsilon_0 = -\frac{(-2)^{t-2}}{\binom{n-1}{t-1}}\left(p_{t-1}^{n-1}(w) + p_{t-1}^{n-1}(w-1)\right) \tag{4.16}$$

En injectant l'expression asymptotique des polynômes de Krawtchouk donnée dans le théorème 4.2 dans l'expression (4.16) nous obtenons,

$$\begin{aligned}\varepsilon_1 - \varepsilon_0 = -\frac{(-2)^{t-2}}{\binom{n-1}{t-1}}\Big(&R_{\sigma_1,\nu}(v)\Im\left(\frac{2^{-(p_1(z_1)+1)v}}{z_1\sqrt{2{p_1}"(z_1)}}(1+\delta_1(v))\right)\\ &+ R_{\sigma_0,\nu}(v)\Im\left(\frac{2^{-(p_0(z_0)+1)v}}{z_0\sqrt{2{p_0}"(z_0)}}(1+\delta_0(v))\right)\Big)\end{aligned}$$

où les fonctions δ_i sont des $o(1)$ uniformément en v.
Nous avons désormais clairement $\sigma_1 = \sigma_0 - \frac{1}{m}$ and $z_1 = z_0 + O\left(\frac{1}{m}\right)$ et ainsi de la forme particulière $R_{\sigma_i,\nu}(v)$ nous en déduisons que,

$$\begin{aligned} R_{\sigma_1,\nu}(v) &= R_{\sigma_0,\nu}(v)\left(1+O(v^{-1/2})\right)\\ \frac{1}{z_1\sqrt{2{p_1}"(z_1)}} &= \frac{1}{z_0\sqrt{2{p_0}"(z_0)}}\left(1+O\left(\frac{1}{m}\right)\right)\end{aligned}$$

Ceci implique,

$$\begin{aligned}\varepsilon_1 - \varepsilon_0 &= \frac{(-1)^v}{2\binom{n-1}{t-1}}R_{\sigma_0,\nu}(v)\left(\Im\left(\frac{2^{-p_1(z_1)v}}{z_0\sqrt{2{p_0}"(z_0)}}(1+o(1))\right) + \Im\left(\frac{2^{-p_0(z_0)v}}{z_0\sqrt{2{p_0}"(z_0)}}(1+\delta_0(v))\right)\right)\\ &= \frac{(-1)^v}{2\binom{n-1}{t-1}}R_{\sigma_0,\nu}(v)\Im\left(\frac{2^{-p_0(z_0)v}}{z_0\sqrt{2{p_0}"(z_0)}}\left(1+\delta_0(v)+2^{(p_0(z_0)-p_1(z_1))v}(1+o(1))\right)\right)\end{aligned} \tag{4.17}$$

Observons maintenant que,

$$\begin{aligned} p_0(z_0) - p_1(z_1) &= \log_2(z_0) - \frac{\sigma_0}{\nu}\log_2(1+z_0) - (\alpha - \frac{\sigma_0}{\nu})\log_2(1-z_0) && (4.18)\\ &\quad - \log_2(z_1) + \frac{\sigma_1}{\nu}\log_2(1+z_1) + (\alpha - \frac{\sigma_1}{\nu})\log_2(1-z_1)\\ &= \log_2\frac{z_0}{z_1} - \frac{\sigma_0}{\nu}\log_2\frac{1+z_0}{1+z_1} - (\alpha - \frac{\sigma_0}{\nu})\log_2\frac{1-z_0}{1-z_1} && (4.19)\\ &\quad + \left(\frac{\sigma_1}{\nu} - \frac{\sigma_0}{\nu}\right)\log_2\frac{1+z_1}{1-z_1}\\ &= \log_2\frac{z_0}{z_1} - \frac{\sigma_0}{\nu}\log_2\frac{1+z_0}{1+z_1} - (\alpha - \frac{\sigma_0}{\nu})\log_2\frac{1-z_0}{1-z_1} - \frac{1}{v}\log_2\frac{1+z_1}{1-z_1} && (4.20)\end{aligned}$$

où (4.20) s'ensuit de l'observation :

$$\begin{aligned}\frac{\sigma_1}{\nu} - \frac{\sigma_0}{\nu} &= \frac{w-1}{v} - \frac{w}{v}\\ &= -\frac{1}{v}\end{aligned}$$

Rappelons maintenant que $z_1 = z_0 + \delta$ où $\delta = O(1/m)$ et :

$$\log_2 \frac{z_0}{z_1} - \frac{\sigma_0}{\nu} \log_2 \frac{1+z_0}{1+z_1} - (\alpha - \frac{\sigma_0}{\nu}) \log_2 \frac{1-z_0}{1-z_1} = p_0(z_0) - p_0(z_1)$$
$$= p_0(z_0) - p_0(z_0+\delta)$$

Or $p_0'(z_0) = 0$ et de cette façon,

$$p_0(z_0) - p_0(z_0+\delta) = O(\delta^2) = O(1/m^2).$$

Injectant cette équation dans (4.20) puis multiplier par v implique que,

$$(p_0(z_0) - p_1(z_1))v = -\log_2 \frac{1-z_1}{1+z_1} + O(1/v) = -\log_2 \frac{1-z_0}{1+z_0} + O(1/v) \tag{4.21}$$

Nous pouvons substituer cette expressions dans (4.17) pour obtenir,

$$\begin{aligned} \varepsilon_1 - \varepsilon_0 &= \frac{(-1)^v}{2\binom{n-1}{t-1}} R_{\sigma_0,\nu}(v) \Im \left(\frac{2^{-p_0(z_0)v}}{z_0\sqrt{2p_0"(z_0)}} \left(1 + \frac{1+z_0}{1-z_0} + o(1)\right)\right) \\ &= \frac{(-1)^v}{\binom{m}{v}} R_{\sigma_0,\nu}(v) \Im \left(\frac{2^{-p_0(z_0)v}}{(z_0-z_0^2)\sqrt{2p_0"(z_0)}} (1+o(1))\right) \end{aligned} \tag{4.22}$$

Rappelons maintenant que,

$$\begin{aligned} R_{\sigma,\nu}(v) &= \frac{1+o(1)}{\sqrt{\pi v}} \\ \binom{m}{v} &= \frac{2^{mh_2(\nu)}}{\sqrt{2\pi m\nu(1-\nu)}} \\ &= \frac{2^{\frac{vh_2(\nu)}{\nu}}}{\sqrt{2\pi v(1-\nu)}} \end{aligned}$$

Nous obtenons en injectant ceci dans (4.22) :

$$\varepsilon_1 - \varepsilon_0 = \frac{(-1)^v\sqrt{2\pi v(1-\nu)}}{\sqrt{\pi v}\left|(z_0-z_0^2)\sqrt{2p_0"(z_0)}\right|} 2^{-v\left(\Re(p_0(z_0)) + \frac{h_2(\nu)}{\nu}\right)} \sin(v\theta - \theta_0)(1+o(1)) \tag{4.23}$$

$$= \frac{(-1)^v\sqrt{1-\nu}}{\left|(z_0-z_0^2)\sqrt{p_0"(z_0)}\right|} 2^{-v\left(\Re(p_0(z_0)) + \frac{h_2(\nu)}{\nu}\right)} \sin(v\theta - \theta_0 + o(1))(1+o(1)) \tag{4.24}$$

ce qui conclut la preuve du lemme. □

Des lemmes 4.3 et 4.4 nous en déduisons immédiatement le corolaire qui suit.

Corollaire 4.1. *Posons* $\gamma \triangleq \frac{1}{\tau}$,

— *Si* $\frac{\omega}{\tau} \in]0, \gamma/2 - \sqrt{\gamma-1}[$ *:*

$$\pi(\omega,\tau) = 2\left(\omega\left(\log_2(r) - \frac{\omega}{\tau}\log_2(1-r) - (\gamma - \frac{\omega}{\tau})\log_2(1+r)\right) + h_2(\tau)\right)$$

où r *est la plus petite racine de* $(\gamma-1)X^2 - (\gamma - 2\frac{\omega}{\tau})X + 1$.

— *Si* $\frac{\omega}{\tau} \in]\gamma/2 - \sqrt{\gamma-1}, \gamma/2[$ *:*

$$\pi(\omega,\tau) = 2\left(\omega\Re\left(\log_2(z) - \frac{\omega}{\tau}\log_2(1+z) - (\gamma - \frac{\omega}{\tau})\log_2(1-z)\right) + h_2(\tau)\right)$$

où $z = re^{\mathbf{i}\varphi}$ *avec* $r = \frac{1}{\sqrt{\gamma-1}}$ *et* $\cos(\varphi) = \frac{2\frac{\omega}{\tau} - \gamma}{2\sqrt{\gamma-1}}$.

Remarque 4.2. Ces formules asymptotiques s'avèrent être particulièrement précises (et non une limite inférieure) dans des "régimes de paramètres cryptographiques" comme cela est montré dans la figure 4.1.

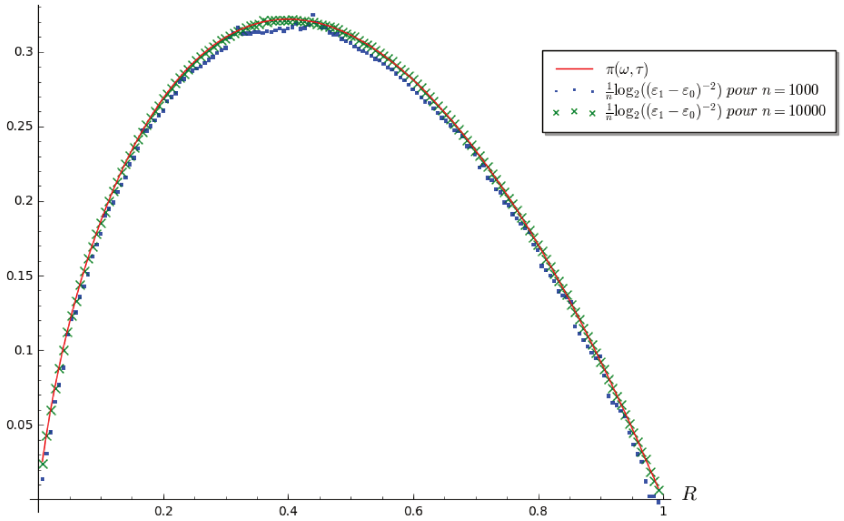


Figure 4.1 – Comparaison de $\pi(\omega,\tau)$ pour $\omega = h_2^{-1}(1-R)$, $\tau = R/2$ avec $\frac{1}{n}\log_2\left(1/(\varepsilon_1-\varepsilon_0)^2\right)$ (voir (4.6)) pour $w = \lfloor \omega n \rfloor$, $t = \lfloor \tau n \rfloor$ où $n = 1000$ et $n = 10000$ en fonction de R.

Nous pouvons maintenant prouver le théorème 4.1.

Démonstration du théorème 4.1.

Le premier cas est une simple réécriture du corolaire. Afin de prouver la formule correspondant au second cas rappelons que le z apparaissant dans le second cas du corolaire 4.1 satisfait $p'(z) = 0$ où

$$p(z) \stackrel{\triangle}{=} \omega \log_2 z - \tau \log_2(1+z) - (1-\tau)\log_2(1-z).$$

Considérons :

$$\begin{aligned} f(\omega,\tau) &\stackrel{\triangle}{=} 2\left(\tau\Re\left(\log_2(z) - \frac{\omega}{\tau}\log_2(1+z) - (\gamma - \frac{\omega}{\tau})\log_2(1-z)\right) + h_2(\tau)\right) \\ &= 2\Re(p(z)) + 2h_2(\tau). \end{aligned}$$

Dérivons cette expression en fonction de τ,

$$\begin{aligned} \frac{\partial f(\omega,\tau)}{\partial \tau} &= 2\Re(p'(z))\frac{\partial z}{\partial \tau} + 2\Re(\log_2(z)) + 2\log_2\frac{1-\tau}{\tau} \\ &= 2\Re(\log_2(z)) + 2\log_2\left(\frac{1-\tau}{\tau}\right) \end{aligned} \tag{4.25}$$

Or $z = re^{\mathrm{i}\varphi}$ avec $r = \frac{1}{\sqrt{\gamma-1}}$, nous en déduisons que,

$$2\Re(\log_2(z)) = 2\log_2 r = 2\log_2\left(\frac{1}{\sqrt{\gamma-1}}\right) = \log_2\left(\frac{1}{1/\tau-1}\right) = \log_2\left(\frac{\tau}{1-\tau}\right).$$

Substituons cette expression pour $2\Re(\log_2(z))$ dans l'équation (4.25) ce qui donne,

$$\frac{\partial f(\omega,\tau)}{\partial \tau} = \log_2\left(\frac{\tau}{1-\tau}\right) + 2\log_2\left(\frac{1-\tau}{\tau}\right) = \log_2\left(\frac{1-\tau}{\tau}\right) = h_2'(\tau). \tag{4.26}$$

Nous continuons la preuve en dérivant maintenant $f(\omega,\tau)$ en fonction de ω,

$$\begin{aligned}\frac{\partial f(\omega,\tau)}{\partial \omega} &= 2\Re(p'(z))\frac{\partial z}{\partial \omega} - 2\Re\left(\log_2(1+z) - \log_2(1-z)\right)\\ &= -2\Re\left(\log_2\left(\frac{1+z}{1-z}\right)\right)\end{aligned}$$

Rappelons que z est aussi défini comme une des racines de l'équation $(1-\tau)X^2 + (1-2\omega)X + \tau = 0$ (voir le théorème 4.2 pour savoir exactement quelle racine est choisie) et ainsi,

$$z = \frac{2\omega - 1 + \mathbf{i}\sqrt{4\tau(1-\tau) - (1-2\omega)^2}}{2(1-\tau)}.$$

De celà nous en déduisons que,

$$1+z = \frac{1-2\tau+2\omega+\mathbf{i}\sqrt{4\tau(1-\tau)-(1-2\omega)^2}}{2(1-\tau)}$$
$$1-z = \frac{3-2\tau-2\omega-\mathbf{i}\sqrt{4\tau(1-\tau)-(1-2\omega)^2}}{2(1-\tau)}$$

$$\begin{aligned}-2\Re\left(\log_2\left(\frac{1+z}{1-z}\right)\right) &= -2\Re\left(\log_2\left(\frac{1-2\tau+2\omega+\mathbf{i}\sqrt{4\tau(1-\tau)-(1-2\omega)^2}}{3-2\tau-2\omega-\mathbf{i}\sqrt{4\tau(1-\tau)-(1-2\omega)^2}}\right)\right)\\ &= -2\log_2\left|\frac{1-2\tau+2\omega+\mathbf{i}\sqrt{4\tau(1-\tau)-(1-2\omega)^2}}{3-2\tau-2\omega-\mathbf{i}\sqrt{4\tau(1-\tau)-(1-2\omega)^2}}\right|\\ &= -\log_2\frac{(1-2\tau+2\omega)^2+4\tau(1-\tau)-(1-2\omega)^2}{(3-2\tau-2\omega)^2+4\tau(1-\tau)-(1-2\omega)^2}\\ &= -\log_2\frac{1+4\tau^2+4\omega^2-4\tau+4\omega-8\tau\omega+4\tau-4\tau^2-1-4\omega^2+4\omega}{9+4\tau^2+4\omega^2-12\tau-12\omega+8\tau\omega+4\tau-4\tau^2-1-4\omega^2+4\omega}\\ &= -\log_2\frac{8\omega-8\tau\omega}{8-8\tau-8\omega+8\tau\omega}\\ &= -\log_2\frac{8\omega(1-\tau)}{8(1-\tau)(1-\omega)}\\ &= -\log_2\frac{\omega}{1-\omega}\\ &= h_2'(\omega).\end{aligned}$$

Ces deux résultats sur les dérivées impliquent que,

$$f(\omega,\tau) = h_2(\omega) + h_2(\tau) + C$$

pour une constante C dont on vérifie facilement qu'elle est égale à -1 en faisant $(\tau,\omega) \to (0,1/2)$ dans $f(\omega,\tau)$.

□

4.3 Deux modèles d'erreurs : canal binaire symétrique versus poids constant

[FKI07] a pour l'étude du décodage statistique introduit un autre modèle que celui où les équations de parité sont supposées tirées uniformément parmi les mots de poids t (on parlera de modèle à poids constant). Les auteurs de [FKI07] ont supposé que les équations de parité $\mathbf{h}$ sont distribuées comme des vecteurs de longueur n où les bits sont des variables aléatoires indépendantes et suivant une loi de Bernoulli de paramètre t/n. De cette façon, le poids attendu est bien t. En revanche, ce-dernier n'est évidemment plus fixé et peut varier. Désignons un tel tirage d'équations de parité par,

le modèle binomial de poids t et longueur n.

Ce modèle présente l'énorme avantage de simplifier grandement (en évitant toute considération sur des polynômes de Krawtchouk) l'étude du nombre d'équations de parité nécessaires au décodage statistique pour prendre une bonne décision. De plus, cette modélisation permit aux auteurs de [FKI07] d'aller encore plus loin en proposant une version itérative du décodage statistique dont ils ont pu donner une complexité asymptotique. En particulier, ces derniers ont démontré le résultat qui suit.

Proposition 4.3 ([FKI07, Proposition 2.1 p.405])**.** *Considérons la version itérative du décodage statistique* [FKI07] *proposé dans le modèle binomial de poids t et de longueur n. Cette version du décodage statistique demande pour corriger w erreurs avec une probabilité exponentiellement proche de 1 un nombre de $O(J_{\min})$ équations de parité où,*

$$J_{\min} = \left(\frac{n}{n-2t}\right)^{2(w-1)} = \left(1-\frac{2t}{n}\right)^{-2(w-1)}$$

avec la constante du "grand O" dépendant seulement du ratio t/n.

L'idée d'un décodage statistique itératif [FKI07] est que cela puisse apporter un certain gain par rapport à sa version naïve. Malheureusement, nous allons démontrer que dans le modèle binomial ces deux variantes de l'algorithme sont de même complexité à un facteur polynomial près.

Il est naturel dans l'étude du décodage statistique d'introduire les deux probabilités.

$$q_0^{\text{bin}} \triangleq \mathbb{P}^{\text{bin}}\left(\langle \mathbf{e}, \mathbf{h}\rangle = 1 \mid h_i = 1\right) \text{ quand } e_i = 0$$

$$q_1^{\text{bin}} \triangleq \mathbb{P}^{\text{bin}}\left(\langle \mathbf{e}, \mathbf{h}\rangle = 1 \mid h_i = 1\right) \text{ quand } e_i = 1$$

où la notation $\mathbb{P}^{\text{bin}}$ est là pour insister sur le fait que $\mathbf{h}$ est distribuée selon le modèle binomial. Ces deux quantités ne dépendent pas de i et comme précédemment nous définissons $\varepsilon_0^{\text{bin}}$ et $\varepsilon_0^{\text{bin}}$:

$$q_0^{\text{bin}} = \frac{1}{2} + \varepsilon_0^{\text{bin}} \quad \text{et} \quad q_1^{\text{bin}} = \frac{1}{2} + \varepsilon_1^{\text{bin}}.$$

Les calculs de [FKI07, §II. B] montrent alors que,

$$q_0^{\text{bin}} = \frac{1-\left(1-\frac{2t}{n}\right)^w}{2} \quad \text{et} \quad q_1^{\text{bin}} = \frac{1+\left(1-\frac{2t}{n}\right)^{w-1}}{2}$$

Ce qui implique,

$$\varepsilon_0^{\text{bin}} = -\frac{\left(1-\frac{2t}{n}\right)^w}{2} \quad \text{et} \quad \varepsilon_1^{\text{bin}} = \frac{\left(1-\frac{2t}{n}\right)^{w-1}}{2}.$$

Afin de comparer au mieux le modèle binomial et celui de poids constant, renommons les quantités q_0, q_1, ε_0 et ε_1 comme q_0^{con}, q_1^{con}, $\varepsilon_0^{\text{const}}$ et $\varepsilon_1^{\text{const}}$ respectivement. Dans le modèle binomial nous procédons comme précédemment pour décoder en calculant m équations de parité $\mathbf{h}^1, \ldots, \mathbf{h}^m$ dont le bit en position i est égal à un. Nous calculons ensuite le compteur :

$$V_m \triangleq \sum_{j=1}^{m} \text{signe}(\varepsilon_1^{\text{bin}} - \varepsilon_0^{\text{bin}})\langle \mathbf{y}, \mathbf{h}^k\rangle = \sum_{j=1}^{m} \langle \mathbf{y}, \mathbf{h}^k\rangle.$$

La moyenne de V_m est alors donnée par $E_b \triangleq m\left(\frac{1}{2} + \varepsilon_b^{\text{bin}}\right)$ où $e_i = b$. Nous décidons alors que $e_i = 0$ si $V_m < \frac{E_0+E_1}{2}$ et 1 sinon. En utilisant à nouveau la borne de Chernoff nous prendrons une mauvaise décision avec une probabilité négligeable dès lors que $m = \frac{n}{(\varepsilon_1^{\text{bin}} - \varepsilon_0^{\text{bin}})^2}$.

Observons maintenant que :

$$\begin{aligned}\frac{1}{(\varepsilon_1^{\text{bin}}-\varepsilon_0^{\text{bin}})^2} &= \frac{1}{(t/n)^2}\left(\frac{1}{(1-2t/n)^{w-1}}\right)\\ &= O\left((1-2t/n)^{-2(w-1)}\right)\\ &= O(J_{\text{min}})\end{aligned}$$

où le grand O dépend du ration t/n. Le décodage statistique naïf demande donc un nombre marginalement plus élevé (de l'ordre de n) d'équations de parité que la version itérative de [FKI07].

Résumons la situation du nombre d'équations de parité nécessaires au décodage statistique en fonction des modèles et de la version de l'algorithme considéré :

— avec la version itérative dans le modèle binomial :

$$O\left(\frac{1}{(\varepsilon_1^{\text{bin}}-\varepsilon_0^{\text{bin}})^2}\right),$$

— avec la version naïve dans le modèle binomial :

$$O\left(\frac{n}{(\varepsilon_1^{\text{bin}}-\varepsilon_0^{\text{bin}})^2}\right),$$

— avec le décodage statistique naïf dans le modèle à poids constant :

$$O\left(\frac{n}{(\varepsilon_1^{\text{const}}-\varepsilon_0^{\text{const}})^2}\right).$$

Il est alors naturel de se demander s'il existe une différence entre les modèles. Il peut sembler très tentant de conjecturer que ces derniers sont identiques, voir très proches étant donné que le poids *attendu* des équations de parité est dans les deux cas le même.

Il n'en n'est malheureusement rien, ces modèles sont bien différents. Nous sommes dans une situation de larges déviations où dans le modèle binomial certains poids d'équations de parité apparaissent avec une probabilité certes faible mais faisant diminuer le biais (lui-même faible) par rapport au modèle à poids constant. Pour illustrer ce point, choisissons $w = \omega n$ et $t = \tau n$ pour ω et τ constants. L'exposant du nombre d'équations de parité nécessaires au décodage statistique dans le modèle binomial est alors donné par,

$$\lim_{n\to+\infty}\frac{1}{n}\log_2\left(\frac{1}{(\varepsilon_1^{\text{bin}}-\varepsilon_0^{\text{bin}})^2}\right) = -2\omega\log_2(1-2\tau)$$

tandis que dans le modèle à poids constant $\lim_{n\to+\infty}\frac{1}{n}\log_2\left(\frac{1}{(\varepsilon_1^{\text{const}}-\varepsilon_0^{\text{const}})^2}\right)$ est donné par le théorème 4.1. Ces deux termes sont alors différents pour la plupart des paramètres. Un cas intéressant est celui où $\tau = R/2$ et $\omega = h_2^{-1}(1-R)$ avec $R \stackrel{\triangle}{=} k/n$ le rendement du code que l'on considère. Le poids relatif $\omega = h_2^{-1}(1-R)$ correspond au cas où l'on cherche à décoder à la distance de Gilbert-Varshamov (là où le décodage semble plus difficile) tandis que $\tau = R/2$ est le poids relatif d'équations de parité le plus facile à obtenir (en temps polynomial, ce qui n'est plus le cas pour $\tau < R/2$) comme nous le verrons dans §4.5. Nous comparons alors les exposants asymptotiques en fonction de R pour ces paramètres dans les figures 4.2, 4.3 et 4.4. Comme nous pouvons le constater les différences sont énormes.

Remarquons cependant que la différence entre les modèles semble s'amoindrir avec τ diminuant comme montré dans la figure 4.4 où $\tau = \frac{1}{2}h_2^{-1}(1-R)$. Cette intuition se confirme comme montré dans la proposition 4.4 : les modèles coïncident pour $\omega = o(1)$.

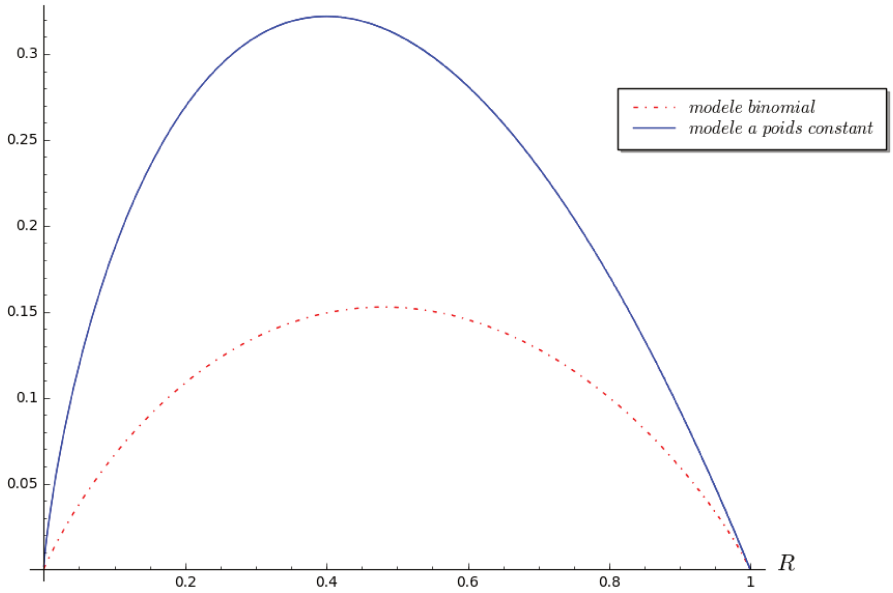


Figure 4.2 – Comparaison des exposants asymptotiques pour $\omega = h_2^{-1}(1-R)$ du nombre d'équations de parité nécessaires dans le modèle binomial et à poids constant en fonction de R.

Proposition 4.4. *Nous avons :*

$$\pi(\omega,\tau) \underset{\omega\to 0}{=} -2\omega\log_2(1-2\tau) + o(\omega)$$

Démonstration de la proposition 4.4.

Du fait que $\omega \to 0$, nous considérons la première formule du théorème 4.1 pour $\pi(\omega,\tau)$. Nous avons pour :

$$r = \frac{1-2\omega-\sqrt{(1-2\omega)^2-4\tau(1-\tau)}}{2(1-\tau)}$$

les calculs suivants,

$$\begin{aligned}\pi(\omega,\tau) &= 2\tau\log_2(r) - 2\omega\log_2(1-r) - 2(1-\omega)\log_2(1+r) + 2h_2(\tau)\\ &= 2\tau\log_2(r) - 2\log_2(1+r) - 2\omega\log_2\left(\frac{1-r}{1+r}\right) + 2h_2(\tau) \qquad (4.27)\end{aligned}$$

Nous allons maintenant calculer le développement asymptotique de r pour $\tau \to 0$. Commençons par,

$$\begin{aligned}r &= \frac{1-2\omega-\sqrt{1-4\tau(1-\tau)-4\omega+4\omega^2}}{2(1-\tau)}\\ &= \frac{1-2\omega-\sqrt{(1-2\tau)^2-4\omega+o(\omega)}}{2(1-\tau)} \qquad (4.28)\end{aligned}$$

Rappelons que :

$$(A^2-\varepsilon)^{1/2} \underset{\varepsilon\to 0}{=} A - \frac{\varepsilon}{2A} + o(\varepsilon)$$

dont nous déduisons :

$$\begin{aligned}r &= \frac{1-2\omega-(1-2\tau)+\frac{2\omega}{1-2\tau}+o(\omega)}{2(1-\tau)}\\ &= \frac{\tau}{1-\tau} + \frac{\omega\tau}{(1-\tau)(1-2\tau)} + o(\omega)\end{aligned}$$

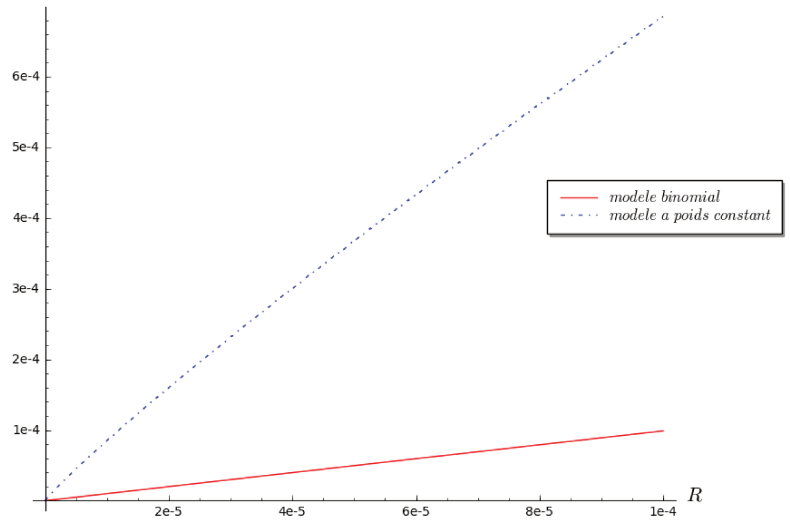


Figure 4.3 – Comparaison des exposants asymptotiques pour $\omega = h_2^{-1}(1-R)$ du nombre d'équations de parité nécessaires dans le modèle binomial et à poids constant pour des rendements proches de 0.

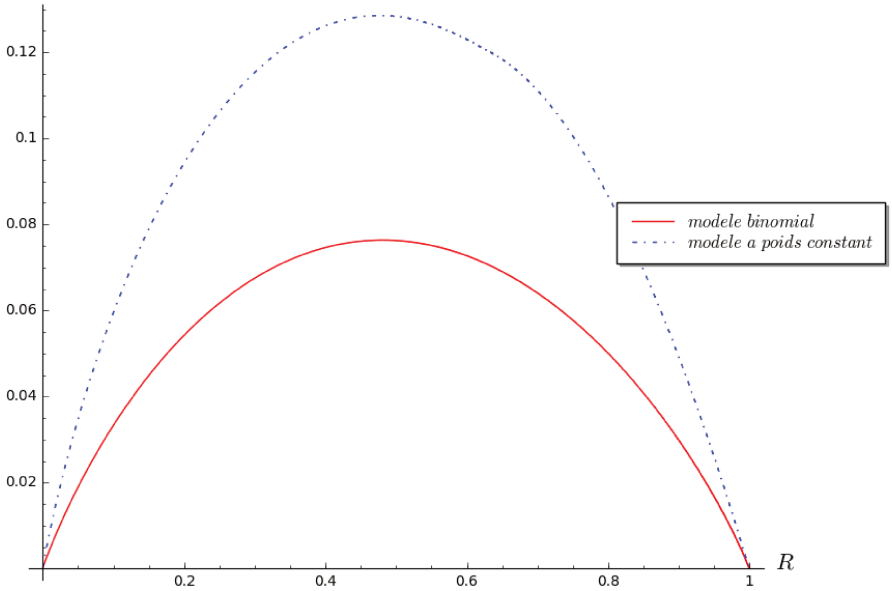


Figure 4.4 – Comparaison des exposants asymptotiques pour $\omega = \frac{1}{2}h_2^{-1}(1-R)$ du nombre d'équations de parité nécessaires dans le modèle binomial et à poids constant.

d'où :

$$1 - r = \frac{1-2\tau}{1-\tau} - \frac{\omega\tau}{(1-\tau)(1-2\tau)} + o(\omega)$$
$$1 + r = \frac{1}{1-\tau} + \frac{\omega\tau}{(1-\tau)(1-2\tau)} + o(\omega) \tag{4.29}$$

et donc :

$$-2\tau \log_2\left(\frac{1-r}{1+r}\right) \underset{\omega\to 0}{=} -2\omega \log_2(1-2\tau) + o(\omega). \tag{4.30}$$

Utilisons maintenant le fait que :

$$\log_2(A+\varepsilon) \underset{\varepsilon\to 0}{=} \log_2(A) + (1/\ln(2))\left(\frac{\varepsilon}{A} + o(\varepsilon)\right).$$

Donc en reprenant les équations (4.28) et (4.29) nous obtenons,

$$\log_2(r) = \log_2\left(\frac{\tau}{1-\tau}\right) + (1/\ln(2))\left(\frac{\omega}{1-2\tau} + o(\omega)\right),$$

$$\log_2(1+r) = \log_2\left(\frac{1}{1-\tau}\right) + (1/\ln(2))\left(\frac{\omega\tau}{1-2\tau} + o(\omega)\right).$$

D'où nous en déduisons,

$$\begin{aligned} 2\tau\log_2(r) - 2\log_2(1+r) &= 2\tau\log_2\left(\frac{\tau}{1-\tau}\right) - 2\log_2\left(\frac{1}{1-\tau}\right) + o(\omega) \\ &= 2\tau\log_2(\tau) + 2(1-\tau)\log_2(1-\tau) + o(\omega) \\ &= -2h_2(\tau) + o(\omega). \end{aligned}$$

En injectant cette équation avec (4.30) dans (4.27) nous terminons la preuve de la proposition.

□

S'intéresser au décodage à distance w sous-linéaire n'est pas dénué d'intérêt. En effet, comme nous l'avons vu dans la partie I de ce document, les codes de Goppa et MDPC utilisés lors de l'instanciation des crypto-systèmes de McEliece et Niederreiter viennent respectivement avec un algorithme de décodage à distance $O(n/\log_2(n))$ et $O(\sqrt{n})$. Pour ce régime de distance il a été démontré dans [CS16a] que tous les ISD sont de même complexité sous-exponentielle :

$$2^{-w\log_2(1-R)(1+o(1))}.$$

Or nous sommes en mesure de trouver en temps polynomial des équations de parité de poids $Rn/2$ comme nous le verrons dans 4.5. Le décodage statistique est alors pour $w = o(n)$ de complexité sous-exponentielle :

$$2^{-2w(\log_2(1-R)+o(1))}$$

ce qui donne un exposant double par rapport à celui des ISD. La question étant alors peut-on trouver des équations de parité de poids relatif $< R/2$ en temps polynomial ou un nombre sous-exponentiel en temps amorti polynomial ? Si oui, nous pourrions améliorer la complexité que nous venons d'énoncer. Malheureusement, au mieux de nos connaissances, il n'existe pas à ce jour de tels algorithmes.

4.4 Étudier le cas d'un poids unique est suffisant

Nous avons montré dans la section précédente que le nombre d'équations de parité nécessaires au décodage statistique dans le modèle binomial était bien inférieur au nombre requis dans le modèle à poids constant pour un décodage à distance linéaire. Le problème étant qu'il n'existe pas à ce jour de façon efficace de produire de telles équations dans le modèle binomial. Considérons le cas facile $w = 1 + Rn/2$ [1] où il est trivial de produire des équations de parité de ce poids en mettant la matrice de parité sous forme systématique. On peut alors vérifier que l'écart-type du poids produit de ces équations est différent de celui d'équations obtenues avec le modèle binomial de poids $1 + Rn/2$. Bien-sûr n'excluons pas pour autant l'existence d'un tel algorithme produisant ces équations dans ce modèle

1. voir la section 4.5 pour plus de détails

binomial. Cependant même si un tel algorithme venait à exister, il produirait naturellement des équations de parité de poids différents. Nous allons ici prouver sous certaines conditions, assez légères, qu'il suffirait de restreindre un tel algorithme aux équations de parité d'un certains poids parmi celle produites. Autrement dit, il est suffisant de nous intéresser au nombre d'équations de parité nécessaires au décodage statistique d'un poids fixé.

Commençons par fixer une position i de $\mathbf{y}$ dont nous voulons savoir si elle est erronée ou non. Considérons un algorithme produisant en temps T,

$$m = \sum_{j=1}^{n} m_j$$

équations de parité dont le bit en position i est égal à 1. Ici m_j désigne le nombre d'équations de parité de poids j produites par l'algorithme que nous notons,

$$\mathbf{h}_1^{(j)}, \cdots, \mathbf{h}_{m_j}^{(j)}.$$

Le décodage statistique utilise alors les valeurs de $\langle \mathbf{y}, \mathbf{h}_s^{(j)} \rangle$ pour prendre une décision. Introduisons les variables aléatoires :

$$X_s^j \stackrel{\triangle}{=} \langle \mathbf{y}, \mathbf{h}_s^{(j)} \rangle.$$

De façon similaire aux hypothèses 1 et 2, nous supposons que les variables aléatoires $\langle \mathbf{y}, \mathbf{h}_s^{(j)} \rangle$ sont indépendantes et que leur distribution est approchée par celle des $\langle \mathbf{y}, \mathbf{h}_s^{(j)} \rangle$ où $\mathbf{h}_s^{(j)}$ est tirée uniformément parmi les mots de poids j. De cette façon avec le biais $\varepsilon_b(j)$ défini dans §4.1.1 pour des équations de parité de poids j nous avons :

$$X_s^j \sim \text{Ber}(1/2 + \varepsilon_b(j)) \quad \text{selon que } e_i = b.$$

Notre objectif est alors de construire un distingueur des hypothèses :

$$\mathcal{H}_0 : e_i = 0 \quad \text{et} \quad \mathcal{H}_1 : e_i = 1.$$

De même que dans 4.1.1 nous allons utiliser un vote majoritaire. Introduisons pour cela la quantité q :

$$q \stackrel{\triangle}{=} \ln \left(\frac{\mathbb{P}_{\mathcal{H}_0} \left(X_1^1 = x_1^1, \cdots, X_{m_1}^1 = x_{m_1}^1, \cdots, X_1^n = x_1^n, \cdots, X_{m_n}^n = x_{m_n}^n \right)}{\mathbb{P}_{\mathcal{H}_1} \left(X_1^1 = x_1^1, \cdots, X_{m_1}^1 = x_{m_1}^1, \cdots, X_1^n = x_1^n, \cdots, X_{m_n}^n = x_{m_n}^n \right)} \right).$$

où les $x_i^j \in \{0, 1\}$ et $\mathbb{P}_{\mathcal{H}_b}$ signifie que nous faisons les calculs de probabilité (sur les équations de parité) en supposant l'hypothèse $\mathcal{H}_b$. Le test étant un vote majoritaire nous procédons comme : si $q > \Theta$, où Θ est un certain seuil, nous choisissons $\mathcal{H}_0$ ou $\mathcal{H}_1$ sinon.

Plus formellement, le test que nous proposons est celui de Neymann-Pearson [CT91]. Il est alors prouvé dans ce cas qu'aucune autre statistique ne permet de diminuer la probabilité de se tromper à taille d'échantillons (à un facteur polynomial près) fixée. Dans notre cas il est optimal de choisir,

$$\Theta = 0.$$

On peut dès lors vérifier que tout autre choix de seuil demandera un échantillon plus gros pour une même probabilité négligeable de prendre une mauvaise décision. Fixons maintenant :

$$p_l(j) \stackrel{\triangle}{=} 1/2 + \varepsilon_l(j)$$

$$I_0(j) \stackrel{\triangle}{=} \#\{0 \in \{x_1^j, \cdots, x_{m_j}^j\}\} \quad \text{et} \quad I_1(j) \stackrel{\triangle}{=} \#\{1 \in \{x_1^j, \cdots, x_{m_j}^j\}\}.$$

Nous avons,

$$\frac{\mathbb{P}_{\mathcal{H}_0}\left(X_1^1 = x_1^1, \cdots, X_{m_1}^1 = x_{m_1}^1, \cdots, X_1^n = x_1^n, \cdots, X_{m_n}^n = x_{m_n}^n\right)}{\mathbb{P}_{\mathcal{H}_1}\left(X_1^1 = x_1^1, \cdots, X_{m_1}^1 = x_{m_1}^1, \cdots, X_1^n = x_1^n, \cdots, X_{m_n}^n = x_{m_n}^n\right)} = \prod_{j=1}^{n} \frac{p_0(j)^{I_1(j)} \cdot (1-p_0(j))^{I_0(j)}}{p_1(j)^{I_1(j)} \cdot (1-p_1(j))^{I_0(j)}}$$

De cette façon, en prenant le logarithme de cette expression et en remarquant que $\sum_{k=1}^{m_j} X_k^j = I_1(j)$ et $I_1(j) + I_0(j) = m_j$, nous obtenons

$$\begin{aligned} q &= \sum_{j=1}^{n} I_0(j) \left[\ln(1-p_0(j)) - \ln(1-p_1(j))\right] + I_1(j) \left[\ln(p_0(j)) - \ln(p_1(j))\right] \\ &= \sum_{j=1}^{n} (m_j - I_1(j)) \left[\ln(1-p_0(j)) - \ln(1-p_1(j))\right] + \sum_{s=1}^{m_j} X_s^j \left[\ln(p_0(j)) - \ln(p_1(j))\right] \\ &= \sum_{j=1}^{n} \sum_{s=1}^{m_j} X_s \left[\ln(p_0(j)) - \ln(1-p_0(j)) + \ln(1-p_1(j)) - \ln(p_1(j))\right] \\ &\qquad + m_j \ln \frac{1-p_0(j)}{1-p_1(j)} \end{aligned}$$

Utilisons maintenant le développement de Taylor autour de 0 suivant,

$$\ln(1/2 + x) = -\ln(2) + 2x - \frac{4x^2}{2} + \frac{8x^3}{3} + o(x^3)$$

pour en déduire où $i \in \{0, 1\}$ que,

$$\begin{aligned} \ln(p_i(j)) &= \ln(1/2 + \varepsilon_i(j)) \\ &= -\ln(2) + 2\varepsilon_i(j) - 2\varepsilon_i(j)^2 + (8/3)\varepsilon_i(j)^3 + o(\varepsilon_i(j)^3) \end{aligned}$$

$$\begin{aligned} \ln(1-p_i(j)) &= \ln(1/2 - \varepsilon_i(j)) \\ &= -\ln(2) - 2\varepsilon_i(j) - 2\varepsilon_i(j)^2 - (8/3)\varepsilon_i(j)^3 + o(\varepsilon_i(j)^3) \end{aligned}$$

Nous obtenons alors,

$$\begin{aligned} q &= \sum_{j=1}^{n} \sum_{s=1}^{m_j} X_s \cdot \left((4\varepsilon_0(j) + (16/3)\varepsilon_0(j)^3 + o(\varepsilon_0(j)^3) - 4\varepsilon_1(j) - (16/3)\varepsilon_1(j)^3 + o(\varepsilon_1(j)^3)\right) \\ &\qquad - 2m_j \cdot (\varepsilon_0(j) - \varepsilon_1(j) + o(\varepsilon_0(j)) + o(\varepsilon_1(j))) \\ &= 4 \sum_{j=1}^{n} \sum_{s=1}^{m_j} X_s^j \left((\varepsilon_0(j) - \varepsilon_1(j) + O(\varepsilon_0(j)^3) + O(\varepsilon_1(j)^3)\right) + m_j \ln \frac{1-p_0(j)}{1-p_1(j)} \\ &\approx 4 \sum_{j=1}^{n} \sum_{s=1}^{m_j} Y_s^j + C \end{aligned}$$

où

$$Y_s^j \triangleq (\varepsilon_0(j) - \varepsilon_1(j)) X_s^j$$

et C est la constante définie comme,

$$C \triangleq m_j \ln \frac{1-p_0(j)}{1-p_1(j)}.$$

Ces calculs suggèrent alors d'utiliser les variables aléatoires Y_s^j pour fabriquer notre test par vote majoritaire. Par hypothèse sur les X_s^j, les Y_s^j sont indépendantes et sous l'hypothèse $\mathcal{H}_b$:

$$\mathbb{P}\left(Y_s^j = (\varepsilon_0(j) - \varepsilon_1(j))\right) = \frac{1}{2} - \varepsilon_b(j) \quad \text{et} \quad \mathbb{P}\left(Y_s^j = (\varepsilon_0(j) - \varepsilon_1(j))\right) = \frac{1}{2} + \varepsilon_b(j).$$

Nous en déduisons que valeur attendue de Y_s^j sous l'hypothèse $\mathcal{H}_b$ est donnée par,

$$\mathbb{E}\left(Y_s^j\right) = (\varepsilon_0(j) - \varepsilon_1(j)) \cdot \left(\frac{1}{2} + \varepsilon_b(j)\right).$$

Tout comme notre premier distingueur, nous définissons le compteur pour $m = \sum_{j=1}^n m_j$ tirages indépendants et uniformes de vecteurs $\mathbf{h}_s^{(j)} \in \mathcal{H}_{j,i}(\mathcal{C})$:

$$V_m \triangleq \sum_{j=1}^{n} \sum_{s=1}^{m_j} Y_s^j.$$

Notons sous l'hypothèse $\mathcal{H}_b$,

$$E_b \triangleq \mathbb{E}(V_m).$$

La différence $E_0 - E_1$ est alors donnée par,

$$\begin{aligned} E_0 - E_1 &= \sum_{j=1}^{n} \sum_{k=1}^{m_j} (\varepsilon_0(j) - \varepsilon_1(j)) \left(\frac{1}{2} + \varepsilon_0(j)\right) - (\varepsilon_0(j) - \varepsilon_1(j)) \left(\frac{1}{2} + \varepsilon_1(j)\right) \\ &= \sum_{j=1}^{n} \sum_{k=1}^{m_j} (\varepsilon_0(j) - \varepsilon_1(j))^2 \\ &= \sum_{j=1}^{n} m_j (\varepsilon_0(j) - \varepsilon_1(j))^2 \end{aligned} \tag{4.31}$$

Nous prenons alors notre décision comme précédemment en fonction de la variation de V_m autour de sa moyenne selon l'hypothèse $\mathcal{H}_b$. En revanche, les Y_s^j n'étant pas équi-distribuées nous ne pouvons pas appliquer la borne de Chernoff. Nous proposons alors d'utiliser la borne de Hoeffding.

Proposition 4.5 (Borne de Hoeffding)**.** *Soient $Y_1, \cdots, Y_m$ des variables aléatoires indépendantes, $a_1, \cdots, a_m$ et $b_1, \cdots, b_m$ avec $a_s < b_s$ tels que :*

$$\forall s, \quad \mathbb{P}\left(a_s \leqslant Y_s \leqslant b_s\right) = 1.$$

Fixons $Z_m \triangleq \sum_{s=1}^m Y_s$, alors

$$\forall \delta \geqslant 0, \quad \mathbb{P}\left(|Z_m - \mathbb{E}(Z_m)| \geqslant \delta\right) \leqslant 2\exp\left(-\frac{2\delta^2}{\sum_{s=1}^m (b_s - a_s)^2}\right).$$

Afin de distinguer les deux hypothèses, considérons $\delta \triangleq \frac{E_0 - E_1}{2}$. De cette façon, sous l'hypothèse $\mathcal{H}_b$ nous obtenons avec la borne de Hoeffding et (4.31),

$$\begin{aligned}\mathbb{P}\left(|V_m - E_b| \geqslant \frac{E_0 - E_1}{2}\right) &= \mathbb{P}\left(|V_m - E_b| \geqslant \sum_{j=1}^{n} \frac{m_j}{2}(\varepsilon_0(j) - \varepsilon_1(j))^2\right) \\ &\leqslant 2\exp\left(-\frac{2/4\left(\sum_{j=1}^{n} m_j(\varepsilon_0(j) - \varepsilon_1(j))^2\right)^2}{\sum_{j=1}^{n} m_j(\varepsilon_0(j) - \varepsilon_1(j))^2}\right) \\ &= 2\exp\left(-\frac{1}{2}\left(\sum_{j=1}^{n} m_j(\varepsilon_0(j) - \varepsilon_1(j))^2\right)\right).\end{aligned}$$

Pour prendre notre décision nous procédons comme,

— Si $V_m < \frac{E_0+E_1}{2}$ alors nous choisissons $\mathcal{H}_0$,

— Sinon nous choisissons $\mathcal{H}_1$.

Il est alors clair que la probabilité de prendre une mauvaise décision P_e est majorée par,

$$P_e \leqslant 2e^{-\frac{1}{2}\left(\sum_{j=1}^{n} m_j(\varepsilon_0(j)-\varepsilon_1(j))^2\right)}.$$

Supposons que l'on veuille $P_e \leqslant 2e^{-\eta}$ pour $\eta > 0$ quelconque. Il est alors suffisant que les quantités $m_1, \cdots, m_n$ vérifient,

$$\frac{1}{2}\sum_{j=1}^{n} m_j(\varepsilon_0(j) - \varepsilon_1(j))^2 \geqslant \eta \Rightarrow \sum_{j=1}^{n} m_j(\varepsilon_0(j) - \varepsilon_1(j))^2 \geqslant 2\eta \tag{4.32}$$

Notons que cette inéquation capture le bon ordre de grandeur des quantités $m_1, \cdots, m_j$ pour que P_e vérifie $P_e \leqslant 2e^{-\eta}$. En effet, dans notre cas les biais étant exponentiellement faible, la borne de Hoeffding est à un facteur polynomial près une égalité. De plus, encore une fois, chercher à utiliser les probabilités a priori sur les hypothèses $\mathcal{H}_0$ et $\mathcal{H}_1$ ne change pas l'ordre de grandeur du nombre d'équations de parité nécessaires au décodage.

Supposons maintenant que,

Hypothèse 3. *Si nous pouvons calculer m équations de parité de poids t en temps T, alors nous sommes capables d'en calculer $n \cdot m$ de ce poids en temps $O(nT)$.*

Cette hypothèse est vérifiée pour tout algorithme "raisonnable" produisant des équations de parité de poids t d'un code aléatoire aussi longtemps que $n \cdot m$ est une fraction constante et < 1 du nombre total d'équations de parité de poids t.

Soit j_0 tel que,

$$m_{j_0}(\varepsilon_0(j_0) - \varepsilon_1(j_0))^2 = \max_{1 \leqslant j \leqslant n}\{m_j(\varepsilon_0(j) - \varepsilon_1(j))^2\} \tag{4.33}$$

Il est alors clair que,

$$n \cdot m_{j_0}(\varepsilon_0(j_0) - \varepsilon_1(j_0))^2 \geqslant \sum_{j=1}^{n} m_j(\varepsilon_0(j) - \varepsilon_1(j))^2$$

et donc d'après (4.32),

$$\sum_{j=1}^{n} m_j(\varepsilon_0(j) - \varepsilon_1(j))^2 \geqslant 2\eta \Rightarrow 2e^{-\frac{1}{2}n \cdot m_{j_0}(\varepsilon_0(j)-\varepsilon_1(j))^2} \leqslant 2e^{-\eta} \tag{4.34}$$

De cette façon, supposons que notre algorithme calcule suffisamment d'équations de parité de poids $1, \cdots, n$ en temps T pour prendre une bonne décision avec une probabilité d'erreur $\leqslant 2e^{-\eta}$, c'est à dire,

$$\sum_{j=1}^{n} m_j(\varepsilon_0(j) - \varepsilon_1(j))^2 \geqslant 2\eta.$$

Alors d'après l'hypothèse 3 nous pouvons calculer $n \cdot m_{j_0}$ équations de parité de poids j_0 en temps $O(n \cdot T)$ où j_0 vérifie (4.33). Le distingueur de §4.1.1 fonctionne alors avec cette algorithme de calcul d'équations de parité de poids j_0 et sa probabilité d'erreur est $\leqslant 2e^{-\eta}$ d'après (4.34).

En résumé le décodage statistique est de même complexité (à un facteur polynomial près) avec un algorithme produisant m_j équations de parité de poids j pour $1 \leqslant j \leqslant n$ qu'avec un algorithme produisant $n \cdot m_{j_0}$ équations de parité d'un même poids j_0 vérifiant (4.33). L'étude du cas où les équations de parité sont de même poids est donc suffisant.

Cependant permettons nous d'insister sur le fait qu'il s'agit d'un résultat asymptotique. Par exemple, les algorithmes que nous allons présenter produisent de nombreuses équations de parité de poids proche de $1 + k/2$ et $p + (k - \ell)/2$. Il serait en pratique contre-productif de ne garder que celle de poids exactement $1 + k/2$ et $p + (k - \ell)/2$.

4.5 Améliorations et limites du décodage statistique

Maintenant que nous disposons de la formule donnant le nombre asymptotique d'équations de parité de poids t nécessaires au décodage statistique nous allons nous intéresser à comment les calculer. De cette façon, à l'aide de la proposition 4.1 nous en déduirons la complexité du décodage statistique.

Considérons un $[n, k]_2$-code $\mathcal{C}$ aléatoire. Par définition, une équation de parité désigne un élément de $\mathcal{C}^\perp$. Donc si $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ dénote une matrice génératrice de $\mathcal{C}$, alors $\mathbf{h}$ est une équation de parité si et seulement si :

$$\mathbf{h}\mathbf{G}^\intercal = \mathbf{0}. \tag{4.35}$$

Donc si nous cherchons une équation de parité de poids relatif $\tau \triangleq t/n$ nous faisons face à une instance du problème $\mathrm{SD}(2, 1 - R, \tau)$ où $R \triangleq k/n$. Nous proposons alors de reprendre les techniques algorithmes des ISD (décrites dans la partie I, chapitre 2) pour calculer les équations de parité nécessaires au décodage statistique.

4.5.1 Des équations de parité avec l'algorithme de Prange

La résolution de (4.35) avec l'algorithme de Prange consiste à choisir une permutation $\mathbf{P} \in \mathbb{F}_2^{n \times n}$ (tirage de l'ensemble d'information) puis calculer une matrice inversible $\mathbf{S} \in \mathbb{F}_2^{k \times k}$ telle que,

$$\mathbf{S}\mathbf{G}\mathbf{P} = \begin{pmatrix} \mathbf{1}_k & \mathbf{G}' \end{pmatrix} \quad \text{où} \quad \mathbf{G}' \in \mathbb{F}_2^{k \times (n-k)}.$$

Dans ce cas on peut vérifier que,

$$\mathbf{H} \triangleq \begin{pmatrix} \mathbf{G}'^\intercal & \mathbf{1}_{n-k} \end{pmatrix} \mathbf{P}^\intercal$$

est une matrice de parité du code $\mathcal{C}$ généré par la matrice $\mathbf{G}$ et ses lignes sont donc des équations de parité. Le code étant aléatoire, avec une probabilité polynomiale nous obtenons donc au moins une équation de poids,

$$1 + \frac{k}{2}. \tag{4.36}$$

Autrement dit avec ce simple algorithme nous pouvons trouver $\tilde{O}\left(P_{1+k/2}\right)$ équations de parité en temps $\tilde{O}\left(P_{1+k/2}\right)$. Nous en déduisons alors à l'aide de la proposition 4.1 le résultat suivant.

Proposition 4.6. *Sous les hypothèses 1 et 2, le problème du décodage d'un code aléatoire de rendement R à distance relative $\omega \triangleq w/n \leqslant \omega^-$ peut être résolu avec le décodage statistique en temps,*

$$\tilde{O}\left(2^{n\cdot\pi(\omega,R/2)}\right)$$

où $\pi(\omega, R/2)$ est donné dans le théorème 4.1.

Nous en déduisons alors avec le théorème 4.1 que lorsque nous cherchons à décoder à la distance relative de Gilbert-Varshamov, *ie :* $\omega = \omega^- = h_2^{-1}(1-R)$, la complexité du décodage statistique utilisant l'algorithme de Prange est donnée à un facteur polynomial près par :

$$2^{n\cdot(h_2(R/2)-R)}.$$

Nous comparons dans la figure 4.5 les exposants asymptotiques de l'algorithme de Prange et du décodage statistique utilisant des équations de parité de poids relatif $R/2$.

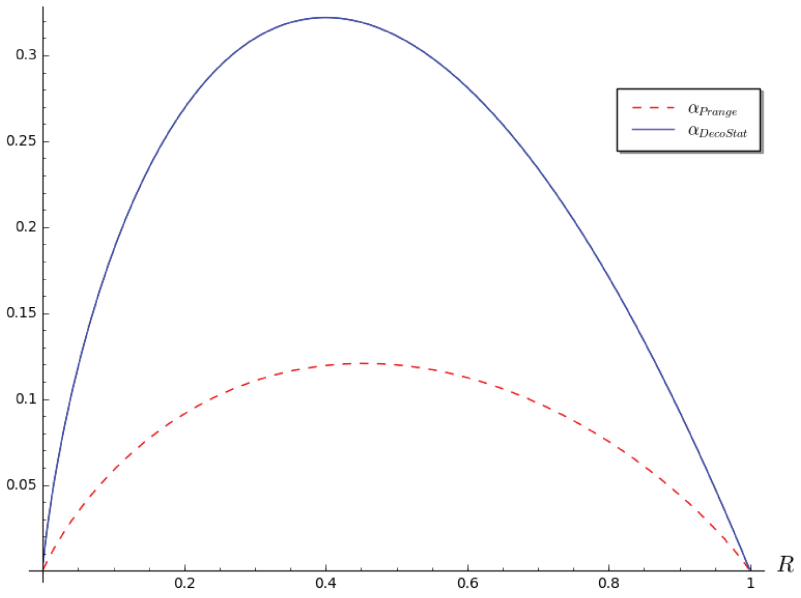


Figure 4.5 – Exposants asymptotiques α_{Prange} et $\alpha_{DecoStat}$ de la complexité de l'algorithme de Prange et du décodage statistique où est utilisé Prange pour $\omega = \omega^-$ en fonction de R.

Comme nous pouvons le constater le décodage statistique avec l'algorithme que nous venons de décrire est de bien moins bonne complexité que l'algorithme de Prange. L'idée pour l'améliorer repose alors sur la remarque suivante. Nous pouvons vérifier que le nombre d'équations nécessaires au décodage statistique de poids t, avec t/n fixé, vérifie $P_t < P_{k/2}$ pour $t/n < R/2$. Il est donc clair que pour améliorer le décodage statistique nous devons chercher un algorithme trouvant des équations de parité de poids relatif $< R/2$. Malheureusement de tels algorithmes sont dans l'état de l'art actuel de complexité exponentielle. La question de changer l'ordre de complexité semble d'autant plus difficile

que la résoudre reviendrait à trouver un algorithme résolvant SD$(2, 1-R, \tau)$ pour $\tau < R/2$ en temps polynomial. Or comme nous l'avons vu dans la partie I, chapitre 2, après 60 ans de recherche aucun algorithme depuis celui de Prange n'a changé l'ordre de grandeur (polynomial, exponentiel, sous-exponentiel) de la complexité de résolution de SD.

Néanmoins, bien qu'une équation de parité de poids $< R/2$ ait un coût exponentiel nous pouvons améliorer la complexité du décodage statistique. En effet, permettons de rappeler la proposition donnant cette dernière.

Proposition 4.1. *Supposons que pour tout $i_0 \in [\![1, n]\!]$ nous disposons d'un algorithme calculant N_t équations de parité $\mathbf{h}$ de poids t avec $h_{i_0} = 1$ en temps T_t. Alors, sous les hypothèses 1 et 2, le problème du décodage binaire à distance $w \leqslant w_{\text{GV}}$ peut-être résolu en temps :*

$$\tilde{O}\left(P_t + T_t \cdot \max\left(1, \frac{P_t}{N_t}\right)\right).$$

où $P_t = 1/(\varepsilon_0 - \varepsilon_1)^2$.

Il y a donc un véritable compromis entre le temps de calcul d'un ensemble d'équations de parité d'un certain poids et le temps amorti pour les trouver. L'algorithme de Dumer s'applique alors naturellement dans ce cadre. En effet, nous avons vu que ce dernier permet de trouver un nombre exponentiel de solutions au problème du décodage en temps amorti polynomial. Néanmoins avant de présenter cette amélioration, donnons une borne inférieure de la complexité du décodage statistique.

4.5.2 Une borne inférieure de complexité du décodage statistique

Le décodage statistique demande $\tilde{O}(P_t)$ équations de parité de poids t pour fonctionner. Nous avons donc clairement le fait suivant :

Fait 5. *Quelque soit la façon dont nous calculons des équations de parité de poids t, la complexité du décodage statistique utilisant ces équations sera toujours supérieure à P_t.*

Or rappelons maintenant que nous cherchons à décoder un $[n, k]_2$-code. Les équations de parité de poids t ne sont donc pas en quantité illimitée. Plus précisément nous nous attendons à ce qu'il en existe :

$$\frac{\binom{n}{t}}{2^k} = \tilde{O}\left(2^{n\cdot(h_2(\tau)-R)}\right).$$

Il est donc nécessaire pour que le décodage statistique puisse fonctionner en moyenne que nous ayons,

$$P_t \leqslant \frac{\binom{n}{t}}{2^k}. \tag{4.37}$$

En combinant cette équation avec le fait 5 il est donc clair que l'exposant asymptotique du décodage statistique pour décoder à la distance relative ω est minoré par,

$$\pi(\omega, \tau_0) \quad \text{où} \quad \tau_0 \triangleq \min\left\{\tau : \pi(\omega, \tau) \leqslant h_2(\tau) - R\right\}. \tag{4.38}$$

Une borne inférieure pour le décodage à la distance de Gilbert-Varshamov. Dans le cas où nous cherchons à décoder à la distance relative de Gilbert-Varshamov, nous pouvons vérifier avec le théorème 4.1 que :

$$\tau_0 = \frac{1}{2} - \sqrt{\omega^-(1-\omega^-)}.$$

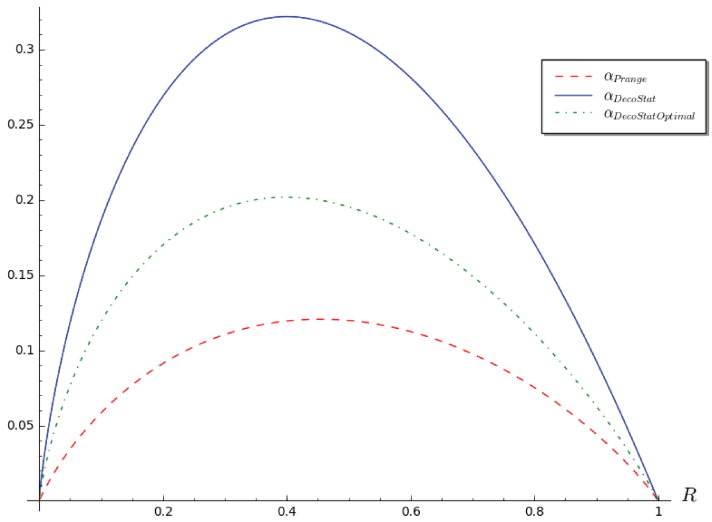


Figure 4.6 – Exposants asymptotiques de l'algorithme de Prange, du décodage statistique optimal et de celui utilisant l'algorithme de Prange pour $\omega = \omega^-$ en fonction de R.

Nous comparons alors dans la figure 4.6 la borne inférieure

$$\pi(\omega^-, \tau_0) = h_2(\tau_0) - R$$

de l'exposant asymptotique du décodage statistique avec l'algorithme de Prange en fonction de R. Comme nous pouvons le constater, des améliorations du décodage statistique utilisant des équations de parité de poids relatif $R/2$ sont bien possibles mais ces dernières resteront toujours moins efficaces que le plus simple des ISD, l'algorithme de Prange.

Une borne inférieure du décodage statistique pour le décodage à distance sous-linéaire. Concernant le décodage à distance sous-linéaire, *ie :* $\omega = o(1)$, la situation est en revanche différente. Rappelons que d'après la proposition 4.4,

$$\pi(\omega, \tau) \underset{\omega \to 0}{=} -2\omega \log_2(1 - 2\tau) + o(\omega)$$

alors que tous les ISD ont pour exposant asymptotique de complexité en base 2 [CS16a] :

$$-\omega \log_2(1 - R)(1 + o(1)).$$

Le nombre d'équations de parité de poids relatif τ sera donc asymptotiquement inférieur à la complexité des ISD dès lors que :

$$\tau < \frac{1 - \sqrt{1 - R}}{2}. \tag{4.39}$$

Il n'est donc pas impossible que le décodage statistique puisse faire mieux que les ISD pour le décodage à distance sous-linéaire. En revanche, d'après la proposition 4.1 cela demanderait un algorithme trouvant une ou un nombre sous-exponentiel d'équations de parité de poids τ vérifiant (4.39) en temps sous-exponentiel. Or dans l'état de l'art nous ne savons pas faire mieux qu'une complexité exponentielle dès lors que $\tau < R/2$.

4.5.3 Des équations de parité avec l'algorithme de Dumer

L'objectif de cette sous-section est de présenter une amélioration du calcul des équations de parité. Il est montré dans [Ove06, §4] comment calculer de telles équations avec

l'algorithme de Stern [Ste88]. Nous allons utiliser nous aussi ce cadre. Cependant, tandis que [Ove06] proposa de répéter [Ste88] de nombreuses fois afin de produire quelques équations de parité de poids plutôt faible, nous proposons dans ce qui suit une autre stratégie. Notre variation de l'algorithme de Dumer [Dum91] (extrêmement similaire à celui de Stern) produira en une exécution un grand nombre d'équations de parité de poids relatif $< R/2$. Notre analyse mènera finalement à un algorithme produisant un nombre exponentiel de solutions en temps amorti polynomial.

Rappelons que nous cherchons à résoudre en $\mathbf{h}$ l'équation,

$$\mathbf{h}\mathbf{G}^\intercal = \mathbf{0} \quad \text{pour} \quad \mathbf{G} \in \mathbb{F}_2^{k\times n}.$$

L'algorithme que nous proposons déroule alors dans le même cadre que les ISD et pour cela introduisons les deux paramètres,

$$\ell \in [\![0, k]\!] \quad \text{et} \quad p \in [\![0, n-k+\ell]\!].$$

Nous commençons par mettre $\mathbf{G}$ sous la forme suivante pour $\mathbf{S} \in \mathbb{F}_2^{k\times k}$ inversible et $\mathbf{P} \in \mathbb{F}_2^{n\times n}$ une permutation,

$$\mathbf{SGP} = \begin{pmatrix} \mathbf{1}_{k-\ell} & \mathbf{G}' \\ \mathbf{0} & \mathbf{G}'' \end{pmatrix} \quad \text{où} \quad \mathbf{G}' \in \mathbb{F}_2^{(k-\ell)\times(n-k+\ell)} \quad \text{et} \quad \mathbf{G}'' \in \mathbb{F}_2^{\ell\times(n-k+\ell)}.$$

Nous procédons ensuite en deux étapes.

1. Nous calculons un ensemble $\mathcal{S}$ de solutions $\mathbf{e}'' \in \mathbb{F}_2^{(n-k+\ell)}$ de poids p de :
$$\mathbf{e}''\mathbf{G}''^\intercal = \mathbf{0} \tag{4.40}$$
2. Pour tout $\mathbf{e}'' \in \mathcal{S}$ nous renvoyons :
$$\mathbf{h} \stackrel{\triangle}{=} (-\mathbf{e}''\mathbf{G}'^\intercal, \mathbf{e}'')\mathbf{P}^\intercal.$$

On vérifie facilement la correction de notre algorithme. De plus, la matrice $\mathbf{G}$ étant supposée aléatoire, les sorties $\mathbf{h}$ sont de poids moyen :

$$\frac{k-\ell}{2} + p. \tag{4.41}$$

Nous proposons alors pour résoudre le point 1. d'appliquer l'algorithme de Dumer que nous avons décrit dans §2.2.1, que l'on désignera ici par `DumerPar`. Nous en déduisons alors la proposition suivante :

Proposition 4.7. *Soit $n \in \mathbb{N}$ et les paramètres k, ℓ, p des fonction de n. L'algorithme* `DumerPar` *trouve en moyenne :*

$$\tilde{O}\left(\frac{\binom{n-k+\ell}{p}}{2^\ell}\right)$$

solutions $\mathbf{e}'' \in \mathbb{F}_2^{(n-k+\ell)}$ de poids p de (4.40) en temps :

$$\tilde{O}\left(\sqrt{\binom{n-k+\ell}{p}} + \frac{\binom{n-k+\ell}{p}}{2^\ell}\right).$$

Démonstration de la proposition 4.7.

Il suffit de décomposer en deux $\mathbf{G}'' = \begin{pmatrix}\mathbf{G}_1 & \mathbf{G}_2\end{pmatrix}$ où $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{F}_2^{\ell \times (n-k+\ell)}$ et de reprendre le raisonnement de §2.2.1 avec les deux listes :

$$\mathcal{L}_1 \triangleq \left\{ \mathbf{e}_1 \mathbf{G}_1^\intercal : \mathbf{e}_1 \in \mathbb{F}_2^{(n-k+\ell)/2}, |\mathbf{e}_1| = p/2 \right\},$$

$$\mathcal{L}_2 \triangleq \left\{ \mathbf{e}_2 \mathbf{G}_2^\intercal : \mathbf{e}_2 \in \mathbb{F}_2^{(n-k+\ell)/2}, |\mathbf{e}_2| = p/2 \right\}.$$

□

Notre objectif est désormais d'étudier asymptotiquement la complexité du décodage statistique avec le résultat de cette proposition. Pour cela introduisons les notations suivantes.

Notation 10.

$$N_{p,\ell} \triangleq \frac{\binom{n-k+\ell}{p}}{2^\ell}, \quad T_{p,\ell} \triangleq \sqrt{\binom{n-k+\ell}{p}} + \frac{\binom{n-k+\ell}{p}}{2^l},$$

$$\rho \triangleq \frac{p}{n} \quad \textit{et} \quad \lambda = \frac{\ell}{n}.$$

Observons que $N_{p,\ell}$ donne le nombre d'équations de parité que nous obtenons dans l'étape 2. de l'algorithme à partir des solutions renvoyées par `DumerPar` en une itération en temps $T_{p,\ell}$. Il y a alors de nombreuses façons de choisir les paramètres p et ℓ. En revanche, dans tous les cas, les équations de parité doivent être de poids supérieur au poids minimal pour lequel le décodage statistique peut fonctionner (voir §4.5.2). Nous en déduisons alors avec (4.38) et (4.41) que,

$$\lfloor n\tau_0(R,\omega) \rfloor \leqslant p + \frac{k-\ell}{2}$$

qui est équivalent à,

$$\tau_0(R,\omega) \leqslant \rho + \frac{R-\lambda}{2} \tag{4.42}$$

Le lemme qui suit donne alors le choix de (ρ, λ) pour obtenir des équations de parité en temps amorti $P(n)$ où $P \in \mathbb{R}[X]$.

Lemme 4.5. *Si :*

$$\rho = (1 - R + \lambda) \cdot h_2^{-1}\left(\frac{2\lambda}{1-R+\lambda}\right) \tag{4.43}$$

Alors avec l'algorithme `DumerPar` *nous obtenons des équations de parité de poids relatif* $\rho + \frac{R-\lambda}{2}$ *en temps amorti* $P(n)$ *où* $P \in \mathbb{R}[X]$*. De plus, avec cette contrainte nous avons :*

$$N_{p,\ell} = \tilde{O}\left(2^{\lambda \cdot n}\right).$$

Démonstration du lemme 4.5.

La contrainte (4.43) découle directement de $\frac{T_{p,\ell}}{N_{p,\ell}} = P(n)$ où P est un polynôme. □

Nous sommes maintenant en mesure de donner la complexité asymptotique du décodage statistique avec l'utilisation de l'algorithme `DumerPar`.

Proposition 4.8. *Avec les contraintes* (4.42)*,* (4.43) *et*

$$\lambda \leqslant \pi(\rho + \frac{R-\lambda}{2}, \omega) \tag{4.44}$$

pour les paramètres (ρ, λ)*, le décodage statistique est sous les hypothèses 1 et 2 de complexité asymptotique :*

$$\tilde{O}\left(2^{n \cdot (\pi(\rho + (R-\lambda)/2, \omega))}\right)$$

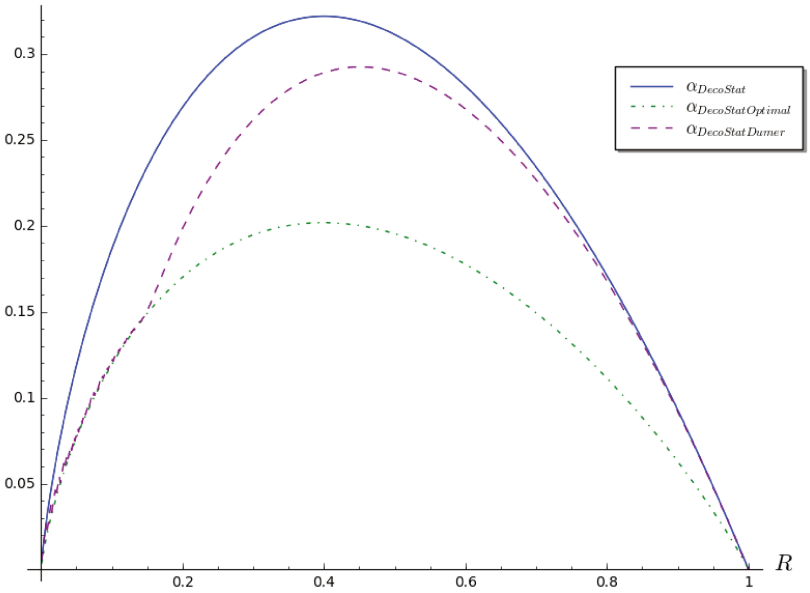


Figure 4.7 – Exposants asymptotiques du décodage statistique optimal, naïf et celui utilisant l'algorithme `DumerPar` pour $\omega = \omega^{-}$ en fonction de R.

Remarque 4.3. Nous résumons maintenant les différentes contraintes de l'algorithme avec leur signification :

— Avec (4.42) nous nous assurons qu'il y a assez d'équations de parité pour que le décodage fonctionne,

— La contrainte (4.43) permet à l'algorithme `DumerFusion` de renvoyer des solutions en temps amorti polynomial,

— Avec (4.44) l'algorithme `DumerFusion` ne fournit pas en moyenne plus de solutions que nécessaires.

Il ne nous reste donc maintenant plus qu'à minimiser en (ρ, λ) la quantité $\pi(\rho + (R - \lambda)/2, \omega)$ sous les contraintes de la proposition 4.8 pour obtenir la complexité optimale du décodage statistique avec l'algorithme `DumerPar`. Nous donnons alors dans la figure 4.7 l'exposant du décodage statistique avec cette stratégie pour le décodage à la distance relative de Gilbert-Varshamov ω^{-} en fonction de R.

Comme nous pouvons le constater, avec la stratégie de `DumerPar` le décodage statistique est de complexité optimale pour des rendements proches de 0. Nous pouvons encore augmenter la zone de rendements où le décodage est optimal en suivant les idées de [MMT11] et [Bec+12].

Troisième partie

Wave : un schéma de signature avec codes suivant la stratégie de GPV

Chapitre 5

Une nouvelle fonction GPV

Introduction

Codes et signatures. La conception d'une signature numérique de type hache et signe dont la sécurité repose sur le décodage générique est un problème ouvert depuis maintenant de nombreuses années. La première réponse à cette question fut donnée par le schéma [CFS01] que nous avons décrit dans §1.4.1. Malheureusement cette primitive est comme nous l'avons vu impraticable dès-lors que nous cherchons à l'instancier pour ne serait-ce que 128 bits de sécurité. D'autres propositions adaptant l'idée de CFS furent faites comme par exemple [Bal+13 ; Gli+14 ; Lee+17] mais toutes ont été cassées [PT16 ; MP16]. Il n'existe donc pas à ce jour de signature de type hache et signe en métrique de Hamming. La seule primitive de signature fondée sur les codes dans cette métrique est celle de Stern [Ste93b] utilisant un protocole d'identification à divulgation nulle de connaissance en signature avec la transformation de Fiat-Shamir [FS87], paradigme différent de celui du hache et signe.

Il y eut cependant de récents progrès pour l'autre métrique utilisée avec des codes : la métrique rang. La primitive `RankSign`, première (et unique) signature hache et signe fut proposée dans [Gab+14b]. Cette-dernière était particulièrement attrayante car jouissant de tailles de clef très petites. Malheureusement ce schéma ne peut aujourd'hui être considéré comme sûr à la lumière de l'attaque polynomiale que nous donnons dans la prochaine partie de ce document. La conception d'une signature de type hache et signe est donc aussi un problème ouvert en métrique rang. En revanche, il existe à ce jour deux signatures utilisant des codes en métrique rang. La première est une adaptation du protocole de Stern dans cette métrique [Bel+19]. La seconde, Durandal [Ara+19a], a récemment été proposée. Cette dernière réussit à adapter le paradigme de Schnorr-Lyubashevsky [Sch91 ; Lyu12] avec des codes correcteurs ce qui est toujours un problème ouvert en métrique de Hamming. Dans le cas de Durandal, du fait qu'il n'est pas prouvé que les signatures ne font pas fuiter d'information, la sécurité est réduite au problème PSSI+ [Ara+18, §4.1] capturant la possibilité de retrouver la clef secrète en utilisant des signatures.

La fonction syndrome. La candidate naturelle pour concevoir une signature hache et signe reposant sur les codes est la fonction à sens unique dite syndrome. Cette-dernière est définie sur les mots $\mathbf{e}$ de poids de Hamming w et pour une matrice de parité $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ comme,

$$f_{w,\mathbf{H}}(\mathbf{e}) \triangleq \mathbf{e}\mathbf{H}^\intercal.$$

Dans ce cas, signer un message $\mathbf{m}$ revient à être en mesure de pouvoir inverser avec une trappe un syndrome $\mathcal{H}(\mathbf{m})$ où $\mathcal{H}$ est une fonction de hachage cryptographique. Il est donc

nécessaire qu'au moins presque toute entrée admette un inverse selon $f_{w,\mathbf{H}}$. Il serait même idéal que cette fonction soit bijective. En effet, la fonction de hachage étant considérée comme aléatoire (modèle de l'oracle aléatoire), l'inverse de $\mathcal{H}(\mathbf{m})$ sera alors lui-même uniformément distribué dans son domaine. Cette propriété est particulièrement intéressante car elle signifie que les signatures ne révèlent aucune information. Malheureusement il y a ici une difficulté, afin d'espérer que $f_{w,\mathbf{H}}$ soit bijective il est nécessaire que la distance de décodage w soit égale à la borne de Gilbert-Varshamov w_{GV}. De cette façon, obtenir une primitive hache et signe avec $f_{w,\mathbf{H}}$ revient à être en mesure de décoder un code à la distance de Gilbert-Varshamov. Il s'agit d'un problème très difficile dans le paradigme des codes correcteurs utilisés aussi bien en cryptographie qu'en théorie de l'information. Les meilleurs algorithmes de décodage pouvant être utilisés en cryptographie le sont pour des w où $w/n \ll w_{\text{GV}}/n$, distance pour laquelle la probabilité qu'un syndrome admette un inverse selon $f_{w,\mathbf{H}}$ est exponentiellement faible. En revanche, la situation change drastiquement et ouvre la porte à de nouvelles possibilités si $w/n \geqslant (1+\varepsilon)w_{\text{GV}}/n$ ($\varepsilon > 0$) où dans ce cas chaque syndrome admet typiquement un nombre exponentiel de solutions. Malheureusement il y a désormais un problème de fuite d'information. L'algorithme inversant $f_{w,\mathbf{H}}(\mathcal{H}(\mathbf{m}))$ doit être précautionneux lorsqu'il "choisit" parmi un nombre exponentiel de solutions. Il se peut que ce choix soit biaisé et provoque une fuite d'information sur le secret utilisé lors de l'inversion.

Les fonctions GPV. C'est dans ce contexte et pour pallier ce problème de fuite d'information dans le cas non bijectif que Gentry, Peikert et Vaikuntanathan (GPV) [GPV08b] ont introduit un nouveau concept de fonctions à sens unique pour concevoir des signatures de type hache et signe. Ces derniers ont introduit les fonctions dites *one-way trapdoor preimage sampleable function* [GPV08b, §5.3] et que nous nommerons dans la suite *fonctions GPV*. Il s'agit essentiellement d'une famille de fonctions $(f_a)_a$ à sens unique et à trappe vérifiant avec une très bonne probabilité sur a les deux propriétés suivantes pour une certaine distribution $\mathcal{D}$. Les images $f_a(e)$ pour e tiré selon $\mathcal{D}$ sont statistiquement proches de l'uniforme (ou tout du moins indépendantes de la trappe). Il est de plus requis un algorithme inversant f_a avec sa trappe associée tel qu'*en toute entrée* y il est retourné un vecteur e distribué selon $\mathcal{D}$ conditionné à l'évènement $f_a(e) = y$. Autrement dit, avec cette propriété il n'y a aucune fuite d'information sur la trappe lors de l'inversion. Les auteurs de [GPV08b] ont alors réussi à instancier ce concept avec des réseaux euclidiens. Dans ce cas, la fonction candidate pour $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ (où $m \geqslant n$) est $f_{\mathbf{A}}$ définie sur des vecteurs courts (au sens de la métrique euclidienne) comme :

$$f_{\mathbf{A}}(\mathbf{e}) \stackrel{\triangle}{=} \mathbf{e}\mathbf{A}^{\mathsf{T}}.$$

Inverser cette fonction pour des entrées aléatoires revient alors exactement à résoudre le problème *Inhomogeneous Short Integer with Solution* (ISIS). La distribution proposée pour les vecteurs $\mathbf{e}$ est quant à elle une Gaussienne discrète d'une certaine variance σ^2. Il s'avère alors pour que les images $f_{\mathbf{A}}(\mathbf{e})$ soient uniformes qu'il est nécessaire de choisir la variance σ^2 suffisamment grande (plus précisément au-dessus du paramètre de lissage [MR09] du réseau considéré). Le nombre attendu de solutions est dans ce cas, comme lors de l'utilisation de codes correcteurs, *exponentiellement grand*. L'algorithme d'inversion avec la trappe est donc difficile à instancier étant donné que ce dernier peut facilement biaiser les solutions produites. Il est extrêmement difficile de fabriquer un algorithme d'inversion qui tire une solution indépendamment de la trappe. Les auteurs de [GPV08b] réussirent à fournir un tel algorithme ce qui leur permit de donner une signature de type hache et signe dont la sécurité est réduite de façon fine au problème *Short Integer with Solution* (SIS) mais aussi le premier IBE quantiquement sûr. Notons que l'algorithme de [GPV08b] s'inspire de celui de Klein [Kle00] étant lui-même une variante de l'algorithme de Babai [Bab86].

Notre contribution : une famille de fonctions GPVM et un schéma de signature. Notre principale contribution est de donner dans le paradigme des codes correcteurs une famille de fonctions $(f_{w,\mathbf{H}})_{\mathbf{H}}$ à sens unique et à trappe vérifiant les propriétés des fonctions GPV de façon légèrement relaxée. La distribution que nous avons choisi sur les vecteurs d'entrée $\mathbf{e}$ est l'uniforme sur les mots de poids de Hamming w. De plus, nous réclamons seulement qu'*en moyenne* sur les syndromes $\mathbf{s}$ (et non pour tout) la distribution de l'algorithme calculant un élément de $f^{-1}_{w,\mathbf{H}}(\mathbf{s})$ soit statistiquement proche d'un tirage uniforme parmi les mots de poids w. En revanche, concernant la propriété de distribution des images $f_{w,\mathbf{H}}(\mathbf{e})$ nous demandons comme pour les fonctions GPV qu'elle soit uniforme. Nous parlons dans la suite de fonctions *GPV en Moyenne* (GPVM). La propriété de l'inversion en moyenne sur les entrées ne nous empêchera pas pour autant d'instancier nos fonctions GPVM en un schéma de signature et d'en donner une réduction de sécurité fine dans le prochain chapitre. De plus, contrairement au cas des réseaux euclidiens où la taille de l'alphabet q grossit avec n, notre instanciation se fera quel que soit le niveau de sécurité pour $q = 3$.

Notre trappe : les codes $(U, U + V)$-généralisés. La trappe dans [GPV08b] consiste en une base courte d'un réseau euclidien. Le réseau généré à partir de cette trappe est alors indistinguable d'un réseau aléatoire. Notre trappe sera quant à elle d'une nature différente. Il s'agit des codes $(U, U + V)$-généralisés où U et V sont tous deux des codes aléatoires. Le nombre de codes $(U, U + V)$-généralisés de dimension $k = \Theta(n)$ est alors du même ordre $q^{\Theta(n^2)}$ que le nombre de codes de même paramètre. Un code $(U, U + V)$-généralisé $\mathcal{C}$ de longueur n sur l'alphabet $\mathbb{F}_q$ est fabriqué à partir de deux codes U et V de longueur $n/2$ et quatre vecteurs $\mathbf{a}, \mathbf{b}, \mathbf{c}$ et $\mathbf{d}$ de $\mathbb{F}_q^{n/2}$ comme le "mélange" suivant :

$$\mathcal{C} = \{(\mathbf{a} \odot \mathbf{u} + \mathbf{b} \odot \mathbf{v}, \mathbf{c} \odot \mathbf{u} + \mathbf{d} \odot \mathbf{v}) : \mathbf{u} \in U,\ \mathbf{v} \in V\}.$$

L'idée derrière cette construction est notre remarque que ces codes viennent avec un algorithme de décodage $\mathtt{D}_{UV}$ pouvant utiliser comme brique élémentaire tout algorithme de décodage générique $\mathtt{D}_{\mathsf{gen}}$. De plus, ce-dernier est plus efficace que si nous avions directement utilisé $\mathtt{D}_{\mathsf{gen}}$ sur le code. Dans notre cas nous proposerons d'utiliser l'algorithme de Prange pour $\mathtt{D}_{\mathsf{gen}}$. Rappelons que ce-dernier permet de décoder les syndromes $\mathbf{s} \in \mathbb{F}_q^{n-k}$ en fonction d'une matrice de parité quelconque $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ en temps polynomial à distance w dans l'intervalle $[\![\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k)]\!]$. L'algorithme $\mathtt{D}_{UV}$ utilise alors de façon maline $\mathtt{D}_{\mathsf{gen}}$ en décodant V puis U en fonction du résultat obtenu pour V. Le point crucial est que ce-dernier fait mieux que si nous avions directement utilisé $\mathtt{D}_{\mathsf{gen}}$ en produisant naturellement en temps polynomial des erreurs hors de l'intervalle $[\![\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k)]\!]$ là où tous les algorithmes de décodage générique sont de complexité exponentielle. Notre structure de codes $(U, U + V)$-généralisés nous offre donc un avantage et peut ainsi être utilisé comme trappe. En revanche, pour obtenir des fonctions GPVM il est nécessaire que notre décodage produise des erreurs uniformément distribuées parmi les mots de poids w. Nous proposerons pour ce faire d'utiliser pour la première fois en cryptographie utilisant des codes la méthode du rejet (*rejection sampling*). Cette-dernière sera comme nous le verrons particulièrement efficace. A l'aide de seulement un rejet environ toutes les 10 ou 12 signatures notre algorithme produit des erreurs statistiquement proches de l'uniforme sur S_w. De plus, tandis que notre trappe permet de décoder à distance w hors de l'intervalle $[\![\frac{q-1}{q}(n-k), k + \frac{q-1}{q}(n-k)]\!]$, notre inversion sans fuite d'information sera pour des $w > k + \frac{q-1}{q}(n-k)$. Nous proposons de cette façon le premier crypto-système fondé sur les codes correcteurs et reposant sur le décodage à grande distance, problème que nous avons étudié dans la partie II de ce document.

Le deuxième intérêt des codes $(U, U + V)$-généralisés pour construire une fonction GPVM provient du fait que le décodage fonctionne pour des U et V choisis *aléatoirement*.

Cette source d'aléa nous sera alors particulièrement utile pour prouver que les images de la fonction syndrome sont statistiquement proches de la distribution uniforme. La preuve de ce résultat reposera sur notre lemme 1.4 qui est lui-même une variation du *left-over hash lemma*.

5.1 Wave : des fonctions GPV en moyenne

5.1.1 Généralités : les fonctions GPV en moyenne fondées sur les codes

Dans ce travail nous serons dans le paradigme des signatures de type hache et signe (FDH) [BR96 ; Cor02] en utilisant comme fonction à sens unique :

$$f_{w,\mathbf{H}} : \begin{array}{ccc} S_{w,n} & \longrightarrow & \mathbb{F}_q^{n-k} \\ \mathbf{e} & \longmapsto & \mathbf{e}\mathbf{H}^\intercal \end{array} \tag{5.1}$$

La signature FDH associée utilisera une trappe (c'est à dire une structure particulière sur $\mathbf{H}$) pour trouver $\mathbf{e} \in f_{w,\mathbf{H}}^{-1}(\mathbf{h})$ où $\mathbf{h}$ est un certain "haché" du message à signer. Ici le domaine de la fonction est $S_{w,n}$ où le poids w sera tel que $f_{w,\mathbf{H}}$ est surjective et non bijective. Il est alors nécessaire d'être extrêmement précautionneux sur la façon d'inverser la fonction. En particulier "le choix" d'un inverse doit se faire sans révéler une quelconque information sur le secret comme nous l'avons évoqué lors de l'introduction. Nous donnons dans la prochaine définition la notion de *one-way trapdoor preimage sampleable function* [GPV08b, Definition 5.3.1] (fonctions GPV) en théorie des codes. En revanche, contrairement à [GPV08b] nous ne réclamons pas de $f_{w,\mathbf{H}}$ d'être une fonction résistante aux collisions. De plus, la propriété d'inversion sans fuite d'information est en moyenne sur les entrées et non pour toute entrée.

Définition 5.1 (Fonctions GPV en Moyenne (GPVM) fondée sur les codes)**.** *Il s'agit d'une paire d'algorithmes* $(\texttt{Trapdoor}, \texttt{InvertAlg})$ *et un triplet de fonctions* $(n(\lambda), k(\lambda), w(\lambda))$ *polynomiales en le paramètre de sécurité* λ *tels que*

- `Trapdoor`*,* la trappe*, est un algorithme probabiliste et polynomial d'entrée* 1^λ *renvoyant* $(\mathbf{H}, T)$ *où* $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ *de rang* $n-k$ *et* T *sa trappe associée,*
- `InvertAlg`*,* l'inversion*, est un algorithme probabiliste polynomial prenant en entrée* T*, un syndrome* $\mathbf{s} \in \mathbb{F}_q^{n-k}$ *et renvoyant* $\mathbf{e} \in S_{w,n}$ *tel que* $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$*.*

De plus, pour presque toute matrice $\mathbf{H}$ *renvoyée par* `Trapdoor` *la fonction est :*

1. Bien distribuée :

$$\rho(\mathbf{e}\mathbf{H}^\intercal, \mathbf{s}) \in \text{negl}(\lambda), \text{ où } \mathbf{e} \hookleftarrow S_{w,n} \text{ et } \mathbf{s} \hookleftarrow \mathbb{F}_q^{n-k}.$$

2. Sans fuite d'information en moyenne :

$$\rho\left(\texttt{InvertAlg}(\mathbf{s}, T), \mathbf{e}\right) \in \text{negl}(\lambda), \text{ où } \mathbf{e} \hookleftarrow S_{w,n} \text{ et } \mathbf{s} \hookleftarrow \mathbb{F}_q^{n-k}.$$

3. Sens unique sans la trappe : *pour tout algorithme probabiliste polynomial* $\mathcal{A}$

$$\mathbb{P}\left(\mathcal{A}(\mathbf{H}, \mathbf{s}) = \mathbf{e} \ : \ \mathbf{e}\mathbf{H}^\intercal = \mathbf{s}\right) \in \text{negl}(\lambda)$$

où la probabilité est calculée sur l'aléa interne de $\mathcal{A}$*, la matrice* $\mathbf{H}$ *et le syndrome* $\mathbf{s} \hookleftarrow \mathbb{F}_q^{n-k}$*.*

Remarque 5.1. Si nous avions traduit en théorie des codes la définition de [GPV08b] comme " les fonctions GPV" (sans notre adverbe en moyenne) nous aurions à la place du point 2 :

$$\forall \mathbf{s} \in \mathbb{F}_q^{n-k},\ \rho\left(\texttt{InvertAlg}(\mathbf{s}, T), \mathbf{e}_{\mathbf{s}}\right) \in \text{negl}(\lambda) \text{ où } \mathbf{e}_{\mathbf{s}} \hookleftarrow \left\{\mathbf{e} \in S_{w,n} : \mathbf{e}\mathbf{H}^{\intercal} = \mathbf{s}\right\}.$$

Le lien avec notre définition provient du calcul suivant comme cela fut remarqué dans [S19],

$$\begin{aligned}
&\rho(\ \texttt{InvertAlg}(\mathbf{s}, T), \mathbf{e}) \\
&\quad = \sum_{\mathbf{s}} \sum_{\mathbf{e} \in f_{\mathbf{H}}^{-1}(\mathbf{s})} \left| \frac{1}{|S_w|} - \frac{1}{q^{n-k}} \mathbb{P}(\texttt{InvertAlg}(\mathbf{s}, T) = \mathbf{e}) \right| \\
&\quad = \sum_{\mathbf{s}} \sum_{\mathbf{e} \in f_{\mathbf{H}}^{-1}(\mathbf{s})} \left| \frac{1}{|S_w|} - \frac{1}{q^{n-k}|f_{\mathbf{H}}^{-1}(\mathbf{s})|} + \frac{1}{q^{n-k}|f_{\mathbf{H}}^{-1}(\mathbf{s})|} - \frac{1}{q^{n-k}} \mathbb{P}(\texttt{InvertAlg}(\mathbf{s}, T) = \mathbf{e}) \right| \\
&\quad \geqslant \sum_{\mathbf{s}} \frac{1}{q^{n-k}} \sum_{\mathbf{e} \in f_{\mathbf{H}}^{-1}(\mathbf{s})} \left| \frac{1}{|f_{\mathbf{H}}^{-1}(\mathbf{s})|} - \mathbb{P}(\texttt{InvertAlg}(\mathbf{s}, T) = \mathbf{e}) \right| - \sum_{\mathbf{s}} \left| \frac{|f_{\mathbf{H}}^{-1}(\mathbf{s})|}{|S_w|} - \frac{1}{q^{n-k}} \right| \\
&\quad = \sum_{\mathbf{s} \in \mathbb{F}_q^{n-k}} \frac{1}{q^{n-k}} \rho\left(\texttt{InvertAlg}(\mathbf{s}, T), \mathbf{e}_s\right) - \rho(\mathbf{e}\mathbf{H}^{\intercal}, \mathbf{s}).
\end{aligned}$$

De cette façon, avec les propriétés 1 et 2 des fonctions GPVM nous en déduisons que la moyenne pour $\mathbf{s} \hookleftarrow \mathbb{F}_q^{n-k}$ des

$$\rho\left(\texttt{InvertAlg}(\mathbf{s}, T), \mathbf{e}_s\right)$$

est négligeable tandis que [GPV08b] requiert impérativement que toutes les distances statistiques $\rho\left(\texttt{InvertAlg}(\mathbf{s}, T), \mathbf{e}_s\right)$ le soient. Notre définition est donc plus faible que celle de [GPV08b]. En revanche, notons que notre propriété vraie en moyenne sur $\mathbf{s}$ l'implique pour presque toute entrée $\mathbf{s}$. En effet, si nous avons :

$$\frac{1}{q^{n-k}} \sum_{\mathbf{s} \in \mathbb{F}_q^{n-k}} \rho\left(\texttt{InvertAlg}(\mathbf{s}, T), \mathbf{e}_s\right) = \varepsilon,$$

alors

$$\frac{\#\left\{\mathbf{s} : \rho\left(\texttt{InvertAlg}(\mathbf{s}, T), \mathbf{e}_s\right) \geqslant \sqrt{\varepsilon}\right\}}{q^{n-k}} \leqslant \sqrt{\varepsilon}.$$

Il s'avèrera que cette définition "relaxée" de fonctions GPV est suffisante pour donner une réduction de sécurité EUF-CMA (voir §6) du schéma de signature que nous allons présenter dans ce qui suit. Cette définition a donc un véritable intérêt dans le sens où cette dernière montre que les conditions de [GPV08b] peuvent être relaxées.

Une fois que l'on dispose d'une famille de fonctions GPVM $(\texttt{Trapdoor}, \texttt{InvertAlg})$ nous obtenons facilement une signature de type hache et signe. La paire de clefs publique et privée est générée comme :

$$(\text{pk}, \text{sk}) \triangleq (\mathbf{H}, T) \hookleftarrow \texttt{Trapdoor}(\lambda).$$

Nous nous donnons ensuite une fonction de hachage cryptographique $\texttt{Hache} : \{0,1\}^* \to \mathbb{F}_q^{n-k}$ ainsi qu'un paramètre λ_0. Les algorithmes $\texttt{Sgn}^{\text{sk}}$ et $\texttt{Vrfy}^{\text{pk}}$ sont alors définis comme suit :

$\texttt{Sgn}^{\mathrm{sk}}(\mathbf{m})$:	$\texttt{Vrfy}^{\mathrm{pk}}(\mathbf{m},(\mathbf{e}',\mathbf{r}))$:
$\mathbf{r} \hookleftarrow \{0,1\}^{\lambda_0}$	$\mathbf{s} \leftarrow \texttt{Hache}(\mathbf{m},\mathbf{r})$
$\mathbf{s} \leftarrow \texttt{Hache}(\mathbf{m},\mathbf{r})$	$\texttt{si } \mathbf{e}'\mathbf{H}^\intercal = \mathbf{s} \texttt{ et } \lvert\mathbf{e}'\rvert = w \texttt{ renvoie } 1$
$\mathbf{e} \leftarrow \texttt{InvertAlg}(\mathbf{s},T)$	$\texttt{sinon renvoie } 0$
$\texttt{renvoie}(\mathbf{e},\mathbf{r})$	

Le point 2 de la définition 5.1 implique que l'algorithme $\texttt{Sgn}^{\mathrm{sk}}$ ne fait fuiter aucune information sur la clef secrète. Nous donnerons alors dans le prochain chapitre une réduction de sécurité fine de ce schéma dans le modèle de l'oracle aléatoire. La condition de bonne distribution peut sembler surprenante de prime abord. Cette dernière permet cependant dans la réduction de sécurité de remplacer les sorties de $\texttt{Hache}$ par $\mathbf{e}\mathbf{H}^\intercal$ avec $\mathbf{e} \hookleftarrow S_{w,n}$ connu du simulateur ce qui permet de signer "sans la clef" secrète. En revanche, nous ne suivrons pas la même stratégie de preuve que dans [GPV08b] où une réduction de sécurité elle aussi fine est donnée. Les auteurs de [GPV08b] réduisent la sécurité de leur schéma au problème de collision à distance euclidienne w. Ceci leur permit ensuite de réduire la sécurité de leur schéma au décodage en norme euclidienne (le problème ISIS) à distance essentiellement $\sqrt{2}w$. Dans notre cas w sera choisi élevé, plus particulièrement :

$$w \geqslant \frac{q-1}{q}(n-k). \tag{5.2}$$

Il s'avère alors que le problème de collision à distance w en métrique de Hamming est facile, *i.e* : pour $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ trouver $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_q^n$ telles que $|\mathbf{e}_1| = |\mathbf{e}_2| = w$ et $\mathbf{e}_1\mathbf{H}^\intercal = \mathbf{e}_2\mathbf{H}^\intercal$. L'idée pour cela est de construire $\mathbf{e}_1$ et $\mathbf{e}_2$ telles que leur différence est un mot du code de matrice de parité $\mathbf{H}$. En effet, commençons trouver $\mathbf{c} \in \mathbb{F}_q^n$ tel que :

$$\mathbf{c}\mathbf{H}^\intercal = \mathbf{0} \quad \text{et} \quad |\mathbf{c}| = \frac{q-1}{q}(n-k). \tag{5.3}$$

Ceci peut se faire en temps moyen polynomial à l'aide de l'algorithme de Prange. Maintenant du fait que $q \geqslant 3$, nous pouvons définir $\mathbf{e}_1, \mathbf{e}_2$ telles que pour les coordonnées i données par le support de $\mathbf{c}$, nous avons $\mathbf{e}_1(i), \mathbf{e}_2(i)$ non nulles et de différence égale à $\mathbf{c}(i)$. Les vecteurs $\mathbf{e}_1$ et $\mathbf{e}_2$ ont $(n-k)(q-1)/q \leqslant w$ (d'après (5.2)) coordonnées non nulles. On les complète alors *de la même façon* avec suffisamment de symboles non nuls pour qu'ils soient de poids w. Leur différence est alors bien égale à $\mathbf{c}$ ce qui donne bien une collision d'après (5.3).

Nous ne ramènerons donc évidemment pas la sécurité du schéma de signature au problème de collision. Afin d'éviter ce problème, tout en conservant une réduction de sécurité fine, nous montrerons dans §6 comment nous réduire à DOOM (voir le problème 1.12).

Il y aura de plus un autre point où notre preuve divergera de celle de [GPV08b] du fait que nous avons pour nos fonctions une absence de fuite d'information *en moyenne*. L'aléa interne $\mathbf{r}$, que l'on nommera sel, sera alors crucial.

5.1.2 La famille Wave de fonctions GPVM

5.1.2.1 Une première approche

Dans un premier temps nous avons proposé [DST17a] pour l'inversion de $f_{w,\mathbf{H}}$ l'utilisation de codes $(U, U+V)$. Rappelons qu'un code $(U, U+V)$ est défini à partir de deux codes U et V de même longueur, disons $n/2$, comme le code de longueur n et de dimension $\dim U + \dim V$ tel que :

$$(U, U+V) \triangleq \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) : \mathbf{u} \in U \text{ et } \mathbf{v} \in V\}.$$

Ces codes ont déjà été considérés en cryptographie pour instancier le schéma de McEliece. Nous pouvons par exemple citer [KKS05, p.225-228] ou plus récemment [SK14]. Cette dernière proposition a été cassée dans [Bar+16] mais elle utilise une sous-classe structurée des codes $(U, U+V)$: les codes polaires [Arı09]. Un code polaire étant défini de façon récursive comme un code $(U, U+V)$ où U et V sont eux-mêmes des codes $(U, U+V)$ jusqu'au moment où ces derniers sont de longueur un. Cette structure fait alors que le code a des mots de poids faible [Bar+16], défaut dont ne souffre pas un code $(U, U+V)$ où U et V sont aléatoires. L'intérêt de ces codes en cryptographie est alors double, ils ont (i) une faible structure et (ii) et un algorithme permettant d'inverser $f_{w,\mathbf{H}}$ comme nous l'avons décrit [DST17b] dans le cas du corps $\mathbb{F}_2$ pour des paramètres w telle que l'inversion sans la trappe est génériquement difficile. L'idée étant que nous pouvons tirer profit, sous la condition $\dim U > \dim V$, du fait que tout mot de U apparaît deux fois dans tout mot de code. Il y a cependant dans ce cas un problème d'un point de vue cryptographique. Commençons par rappeler la définition de *hull* [Sen97] d'un code $\mathcal{C}$.

Définition 5.2. *Le hull d'un code $\mathcal{C}$ est défini comme :*

$$\mathrm{hull}(\mathcal{C}) \triangleq \mathcal{C} \cap \mathcal{C}^{\perp}.$$

Il est clair que le hull d'un code linéaire est un espace-vectoriel. Sa dimension attendue est donnée dans la proposition qui suit.

Proposition 5.1 ([Sen97])**.** *La dimension attendue du hull d'un $[n,k]_2$-code aléatoire est un $O(1)$. De plus, cette dernière est $\leqslant t$ avec probabilité $\geqslant 1 - O\left(2^{-t}\right)$.*

Il s'avère alors malheureusement que le hull d'un code $(U, U+V)$ *binaire* où $\dim U > \dim V$ est anormal comme le montre la proposition qui suit.

Proposition 5.2. *Considérons un code $\mathcal{C}$ de longueur n étant un $(U, U+V)$ binaire permuté où $k_U \triangleq \dim U$ et $k_V \triangleq \dim V$ sont telles que $k_U > k_V$. Alors nous avons avec probabilité $1 - O(2^{k_V - k_U})$:*

$$\dim(\mathrm{hull}(\mathcal{C})) = k_U - k_V$$

où la probabilité est calculée les codes U et V uniformément distribués dans leur domaine.

Démonstration de la proposition 5.2.

La dimension d'un hull étant invariante par permutation, nous allons ici prouver qu'avec probabilité $1 - O(2^{k_V - k_U})$ nous avons $\dim(\mathrm{hull}(U, U+V)) = k_U - k_V$. Commençons par noter que (voir la proposition 5.3 un peu plus loin),

$$(U, U+V)^{\perp} = (V^{\perp} + U^{\perp}, V^{\perp}).$$

Cela implique,

$$\mathrm{hull}((U, U+V)) = (U, U+V) \cap (U^{\perp} + V^{\perp}, V^{\perp}).$$

De cette façon, pour tout tout vecteur $(\mathbf{u}, \mathbf{u}+\mathbf{v}) \in \mathrm{hull}((U, U+V))$ où $\mathbf{u} \in U$ et $\mathbf{v} \in V$ il existe $\mathbf{v}^{\perp} \in V^{\perp}$ et $\mathbf{u}^{\perp} \in U^{\perp}$ tels que

$$\begin{cases} \mathbf{u} = \mathbf{v}^{\perp} + \mathbf{u}^{\perp} \\ \mathbf{u} + \mathbf{v} = \mathbf{v}^{\perp} \end{cases}$$

$$\iff \begin{cases} \mathbf{v} = \mathbf{u}^{\perp} \\ \mathbf{u} + \mathbf{v} = \mathbf{v}^{\perp} \end{cases}$$

Donc $\mathbf{v} \in V \cap U^{\perp}$. Remarquons maintenant que,

$$\dim(V) + \dim(U^{\perp}) = k_V + n/2 - k_U = n/2 + k_V - k_U < n/2.$$

Nous en déduisons alors qu'avec probabilité $1 - O(2^{k_V - k_U})$ (sur le tirage des codes) nous avons $V \cap U^\perp = \{0\}$ et $\mathbf{v} = \mathbf{u}^\perp = \mathbf{0}$. Il s'ensuit que les vecteurs de $\mathrm{hull}(U, U+V)$ sont avec probabilité $1 - O(2^{k_V - k_U})$ de la forme $(\mathbf{x}, \mathbf{x})$ où $\mathbf{x} \in U \cap V^\perp$. Encore une fois, remarquons que,

$$\dim(U) + \dim(V^\perp) = k_U + n/2 - k_V = n/2 + k_U - k_V > n/2.$$

Nous en déduisons alors qu'avec probabilité $1 - O(2^{k_V - k_U})$ nous avons,

$$\dim(U \cap V^\perp) = k_U - k_V.$$

Ceci implique alors qu'avec probabilité $1 - O(2^{k_V - k_U})$,

$$\dim(\mathrm{hull}((U, U+V))) = \dim(U \cap V^\perp) = k_U - k_V$$

ce qui conclut la preuve de la proposition. □

Les codes $(U, U+V)$-binaires où $\dim U > \dim V$ ne peuvent donc pas être utilisés à des fins cryptographiques. En effet, nous pouvons facilement les distinguer de codes aléatoires, il suffit de calculer la dimension de leur hull qui est (même après avoir permuté le code) anormale si nous la comparons à la dimension attendue donnée dans la proposition 5.1. De plus, la preuve de la proposition 5.2 montre que le hull de ces codes $(U, U+V)$ révèle des mots permutés de la forme $(\mathbf{x}, \mathbf{x})$ où $\mathbf{x} \in U \cap V^\perp$ ce qui donne clairement de l'information sur la permutation utilisée.

Fort heureusement nous pouvons éviter cette propriété du hull en changeant la nature des codes considérés et tout en conservant l'avantage du décodage. C'est l'objet de notre généralisation.

5.1.2.2 Les codes $(U, U+V)$-généralisés

La première idée derrière notre généralisation fut d'augmenter la taille du corps sur lequel nous construisons nos codes $(U, U+V)$ Il s'avère dans ce cas qu'il existe désormais de nombreuses façons de mélanger deux codes U et V pour construire un code de dimension $\dim U + \dim V$ où U apparaît deux fois. Plus précisément, notre généralisation consiste en la définition suivante :

Définition 5.3. *Soient un entier n pair et $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$ des vecteurs de $\mathbb{F}_q^{n/2}$ vérifiant pour tout $i \in [\![1, n/2]\!]$:*

$$a_i c_i \neq 0 \tag{5.4}$$

$$a_i d_i - b_i c_i \neq 0 \tag{5.5}$$

Nous définissons pour le quadruplet de vecteurs $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ et les codes U, V sur $\mathbb{F}_q$ et de longueur $n/2$ le code $(U, U+V)$-généralisé comme :

$$\{(\mathbf{a} \odot \mathbf{u} + \mathbf{b} \odot \mathbf{v}, \mathbf{c} \odot \mathbf{u} + \mathbf{d} \odot \mathbf{v}) : \mathbf{u} \in U \text{ et } \mathbf{v} \in V\}.$$

Remarque 5.2. Nous fixerons dans toute la suite,

$$\forall i \in [\![1, n/2]\!], \quad a_i d_i - b_i c_i = 1.$$

Cela simplifiera dans ce qui suit certaines des expressions. De plus, nous ne perdons aucune généralité avec cette contrainte. On vérifie facilement que tout code $(U, U+V)$-généralisé peut être ramené à ce cas en choisissant bien U et V.

La seconde condition (5.5) est essentielle pour qu'un code $(U, U+V)$-généralisé soit de dimension $\dim U + \dim V$ comme nous le verrons dans la preuve de proposition qui suit.

Proposition 5.3. *Soient U (resp. V) un $[n/2, k_U]_q$-code (resp. $[n/2, k_V]_q$-code) et $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{F}_q^{n/2}$ vérifiant les équations (5.4) et (5.5). Notons $\mathcal{C}$ le code $(U, U+V)$-généralisé associé. Nous avons :*

$$\dim \mathcal{C} = k_U + k_V.$$

De plus, désignons par $\mathbf{G}_V \in \mathbb{F}_q^{k_V \times n/2}$ et $\mathbf{H}_V \in \mathbb{F}_q^{(n/2-k_V)\times n/2}$ (resp. $\mathbf{G}_U \in \mathbb{F}_q^{k_U \times n/2}$ et $\mathbf{H}_U \in \mathbb{F}_q^{(n/2-k_U)\times n/2}$) des matrices génératrice et de parité du code V (resp. U). Notons maintenant,

$$\mathbf{A} \triangleq \mathbf{Diag}(\mathbf{a}), \quad \mathbf{B} \triangleq \mathbf{Diag}(\mathbf{b}), \quad \mathbf{C} \triangleq \mathbf{Diag}(\mathbf{c}) \quad \textit{et} \quad \mathbf{D} \triangleq \mathbf{Diag}(\mathbf{d}).$$

La matrice,

$$\left(\begin{array}{c|c} \mathbf{G}_U\mathbf{A} & \mathbf{G}_U\mathbf{C} \\ \hline \mathbf{G}_V\mathbf{B} & \mathbf{G}_V\mathbf{D} \end{array}\right) \in \mathbb{F}_q^{(k_U+k_V)\times n}. \tag{5.6}$$

est alors une matrice génératrice de $\mathcal{C}$ tandis qu'une matrice de parité est donnée par :

$$\left(\begin{array}{c|c} \mathbf{H}_U\mathbf{D} & -\mathbf{H}_U\mathbf{B} \\ \hline -\mathbf{H}_V\mathbf{C} & \mathbf{H}_V\mathbf{A} \end{array}\right) \in \mathbb{F}_q^{(n-k_U-k_V)\times n}. \tag{5.7}$$

Démonstration de la proposition 5.3.

Il est clair que la matrice (5.6) génère le code $(U, U+V)$-généralisé considéré. Montrons que cette dernière est de rang plein et nous aurons les deux premières assertions de la proposition. Il suffit pour cela de commencer par remarquer que :

$$\left(\begin{array}{c|c} \mathbf{G}_U\mathbf{A} & \mathbf{G}_U\mathbf{C} \\ \hline \mathbf{G}_V\mathbf{B} & \mathbf{G}_V\mathbf{D} \end{array}\right) = \left(\begin{array}{c|c} \mathbf{G}_U & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{G}_V \end{array}\right) \left(\begin{array}{c|c} \mathbf{A} & \mathbf{C} \\ \hline \mathbf{B} & \mathbf{D} \end{array}\right)$$

La matrice $\left(\begin{array}{c|c} \mathbf{G}_U & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{G}_V \end{array}\right)$ est par définition des matrices génératrice $\mathbf{G}_U$ et $\mathbf{G}_V$ de rang $k_U + k_V$. Or maintenant, les matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ et $\mathbf{D}$ étant diagonales, le déterminant de la matrice carrés qu'elles forment dans l'égalité qui précède est donné par la produit des $(a_i d_i - b_i c_i)$ pour $1 \leqslant i \leqslant n/2$. Ce dernier étant non nul d'après (5.5), nous avons notre résultat.

Pour démontrer la fin de la proposition, remarquons tout d'abord que $\mathbf{G}\mathbf{H}^\intercal = \mathbf{0}$. Donc la matrice $\mathbf{H}$ génère un sous-code du dual de $\mathcal{C}$. Nous pouvons néanmoins vérifier avec le même raisonnement que précédemment que $\mathbf{H} \in \mathbb{F}_q^{(n-k_U-k_V)\times n}$ est de rang plein ce qui conclut la preuve.

□

Avec la définition des codes $(U, U+V)$-généralisés nous avons mélangé les codes U et V dans le sens où tout mot de code s'écrit aux coordonnées $(i, i+n/2)$ pour $1 \leqslant i \leqslant n/2$ comme :

$$(a_i u_i + b_i v_i, c_i u_i + d_i v_i) \quad \text{où} \quad \mathbf{u} \in U \quad \text{et} \quad \mathbf{v} \in V.$$

De cette façon, l'équation (5.4) nous assure que toutes les coordonnées de U apparaissent bien deux fois. Cette contrainte est alors essentielle pour disposer d'un algorithme de décodage tirant profit de la structure. Notre trappe utilisera alors cet avantage.

Remarquons maintenant qu'un code $(U, U+V)$ est bien un code $(U, U+V)$-généralisé. Il suffit de choisir $\mathbf{a}, \mathbf{b}, \mathbf{c}$ et $\mathbf{d}$ tels que :

$$\mathbf{a} = \mathbf{c} = \mathbf{d} = \mathbf{1} \quad \text{et} \quad \mathbf{b} = \mathbf{0}.$$

Dans le cas du corps $\mathbb{F}_2$, il s'agit même du seul code $(U, U+V)$-généralisé (à échange près des vecteurs $\mathbf{a}$ et $\mathbf{b}$). En revanche, ceci n'est plus vrai dès-lors que nous augmentons la taille du corps considéré. Par exemple, pour le corps $\mathbb{F}_3 = \{-1, 0, 1\}$, le code

$$(U+V, U-V) \triangleq \{(\mathbf{u}+\mathbf{v}, \mathbf{u}-\mathbf{v}) : \mathbf{u} \in U \text{ et } \mathbf{v} \in V\}$$

est un $(U, U+V)$-généralisé. Notre généralisation en est donc bien une. Finalement, cette-dernière peut être vue comme une extension "cryptographiquement intéressante" des codes $(U, U+V)$ classiquement considérés dans la littérature.

Introduisons maintenant quelques notations qui nous seront utiles tout au long de cette partie.

Notation 11. *Soient* $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ *et* $\mathbf{d}$ *des mots de* $\mathbb{F}_q^{n/2}$. *Nous notons*

$$\begin{array}{rccc} \varphi_{\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}} : \mathbb{F}_q^{n/2} \times \mathbb{F}_q^{n/2} & \to & \mathbb{F}_q^{n/2} \times \mathbb{F}_q^{n/2} \\ (\mathbf{x}, \mathbf{y}) & \longmapsto & (\mathbf{a} \odot \mathbf{x} + \mathbf{b} \odot \mathbf{y}, \mathbf{c} \odot \mathbf{x} + \mathbf{d} \odot \mathbf{y}) \end{array}$$

Nous dirons que $\varphi_{\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}}$ *est UV-*normalisée *si les vecteurs* $\mathbf{a}, \mathbf{b}, \mathbf{c}$ *et* $\mathbf{d}$ *vérifient les contraintes de la définition 5.3.*

Dans toute la suite, lorsqu'il n'y aura aucune ambigüité, une application UV-normalisée φ définira implicitement un quadruplet de vecteurs $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ tel que $\varphi = \varphi_{\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}}$. De cette façon, introduisons maintenant la notation suivante.

Notation 12. *Soient* φ *une application* UV*-normalisée. Nous noterons*

$$\mathcal{H}(\varphi, \mathbf{H}_U, \mathbf{H}_V) \triangleq \left(\begin{array}{c|c} \mathbf{H}_U\mathbf{D} & -\mathbf{H}_U\mathbf{B} \\ \hline -\mathbf{H}_V\mathbf{C} & \mathbf{H}_V\mathbf{A} \end{array}\right)$$

la matrice de parité d'un code $(U, U+V)$*-généralisé avec les notations de la définition 5.3.*

Définir `Trapdoor` **et** `InvertAlg`**.** A partir du niveau de sécurité λ nous montrerons dans §5.3.4 comment calculer les paramètres du système, à savoir n, k, w où $k \triangleq k_U + k_V$. L'algorithme $\texttt{Trapdoor}(\lambda)$ renvoie alors :

$$T \triangleq (\varphi, \mathbf{H}_U, \mathbf{H}_V, \mathbf{S}, \mathbf{P}) \quad \text{et} \quad \mathbf{H} \triangleq \mathbf{S}\mathcal{H}(\varphi, \mathbf{H}_U, \mathbf{H}_V)\mathbf{P}$$

où φ est une application UV-normalisée et les matrices $\mathbf{H}_U \in \mathbb{F}_q^{(n/2-k_U)\times n/2}$ de rang $n/2 - k_U$, $\mathbf{H}_V \in \mathbb{F}_q^{(n/2-k_V)\times n/2}$ de rang $n/2 - k_V$, $\mathbf{S} \in \mathbb{F}_q^{(n-k)\times(n-k)}$ inversible et $\mathbf{P} \in \mathbb{F}_q^{n\times n}$ une permutation. Chacune de ces matrices est *uniformément distribuée* dans son domaine.

Dans la suite nous donnerons un algorithme $D_{\varphi,\mathbf{H}_U,\mathbf{H}_V}$ permettant d'inverser $f_{w,\mathcal{H}(\varphi,\mathbf{H}_U,\mathbf{H}_V)}$. Nous prouverons que ce-dernier renvoie des erreurs $\mathbf{e}$ quasiment uniformément distribuées sur $S_{w,n}$ dès lors que ses entrées sont elles-même uniformément distribuée surs $\mathbb{F}_q^{n-k}$ (voir §5.3.1). Cela nous permettra alors d'instancier l'algorithme `InvertAlg` comme :

$$\begin{array}{l} \texttt{InvertAlg}(\mathrm{sk}, \mathbf{s}) \\ \quad \mathbf{e} \leftarrow D_{\varphi,\mathbf{H}_U,\mathbf{H}_V}(\mathbf{s}\big(\mathbf{S}^{-1}\big)^{\intercal}) \\ \quad \texttt{renvoie } \mathbf{eP} \end{array}$$

De cette façon nous aurons une famille de fonctions GPVM une fois que nous aurons prouvé dans §5.4 la condition 1 de la définition 5.1 (à l'aide de notre variation du *left-over-hash lemma, lemme 1.4*). Nous parlerons dans ce qui suit de la famille Wave-GPVM.

5.2 Inversion de la fonction syndrome avec la trappe des codes $(U, U+V)$-généralisés

Rappelons que la fonction syndrome $f_{w,\mathbf{H}}$ s'inverse génériquement avec l'algorithme de Prange (voir la partie II) pour $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ et w dans l'intervalle :

$$[\![w^-_{\text{easy}}, w^+_{\text{easy}}]\!] \quad \text{où} \quad w^-_{\text{easy}} \triangleq \frac{q-1}{q}(n-k) \quad \text{et} \quad w^+_{\text{easy}} \triangleq k + \frac{q-1}{q}(n-k).$$

De plus, rappelons qu'afin d'espérer l'existence en toute entrée $\mathbf{s} \in \mathbb{F}_q^{n-k}$ d'un inverse $f^{-1}_{\mathbf{H},w}(\mathbf{s})$ il est nécessaire que w soit dans l'intervalle

$$[\![w^-, w^+]\!] \quad \text{(voir §1.1.3)}.$$

Nous allons donner dans ce qui suit une procédure permettant d'inverser la fonction syndrome avec $\mathbf{H}$ une matrice de parité d'un code $(U, U+V)$-généralisé et $w \in [\![w^-_{\text{UV}}, w^+_{\text{UV}}]\!]$ vérifiant :

$$[\![w^-_{\text{easy}}, w^+_{\text{easy}}]\!] \subsetneq [\![w^-_{\text{UV}}, w^+_{\text{UV}}]\!] \subset [\![w^-, w^+]\!]$$

Nous résumons la situation dans la figure 5.1. La procédure que nous allons décrire est donnée pour expliquer comment tirer avantage de la trappe des codes $(U, U+V)$-généralisés dans le cas où

$$q \geqslant 3.$$

Une méthode avec rejet ainsi que plusieurs aléas internes seront nécessaires pour que l'inversion soit sans fuite d'information. Cela sera l'objet de la prochaine section.

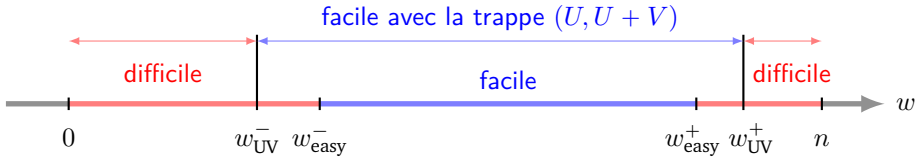


Figure 5.1 – Difficulté du décodage générique avec et sans trappe $(U, U+V)$.

Commençons par ce lemme :

Lemme 5.1. *Soit $\varphi \triangleq \varphi_{\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}}$ une application UV-*normalisée*. Cette dernière est bijective et d'inverse :*

$$\varphi^{-1}(\mathbf{x}, \mathbf{y}) = (\mathbf{d} \odot \mathbf{x} - \mathbf{b} \odot \mathbf{y}, -\mathbf{c} \odot \mathbf{x} + \mathbf{a} \odot \mathbf{y}).$$

Ce lemme nous permet d'introduire la notation suivante qui nous sera très utile pour montrer comment inverser la fonction syndrome des codes $(U, U+V)$-généralisés à distance w.

Notation 13. *Considérons $\mathbf{e} \in \mathbb{F}_q^n$. Nous noterons $\mathbf{e}_U$ et $\mathbf{e}_V$ les vecteurs de $\mathbb{F}_q^{n/2}$ tels que*

$$(\mathbf{e}_U, \mathbf{e}_V) = \varphi^{-1}(\mathbf{e}).$$

En effet, nous avons :

Proposition 5.4. *Inverser $f_{\mathcal{H}(\varphi,\mathbf{H}_U,\mathbf{H}_V),w}$ est équivalent à trouver $\mathbf{e} \in \mathbb{F}_q^n$ de poids w tel que*

$$\mathbf{e}_U \mathbf{H}_U^\intercal = \mathbf{s}^U \quad \textit{et} \quad \mathbf{e}_V \mathbf{H}_V^\intercal = \mathbf{s}^V \tag{5.8}$$

où $\mathbf{s} = (\mathbf{s}^U, \mathbf{s}^V)$ avec $\mathbf{s}^U \in \mathbb{F}_q^{n/2-k_U}$ et $\mathbf{s}^V \in \mathbb{F}_q^{n/2-k_V}$.

Notre algorithme de décodage consistera alors à décoder dans le code V pour obtenir $\mathbf{e}_V$ puis dans le code U afin d'avoir $\mathbf{e}_U$. De cette façon nous aurons une solution au problème du décodage d'un code $(U, U+V)$-généralisé.

Remarque 5.3. Nous avons mis des indices "supérieurs" en notant $\mathbf{s}^U$ et $\mathbf{s}^V$ afin d'éviter toute confusion avec la notation $\mathbf{e}_U$ et $\mathbf{e}_V$ que nous venons d'introduire.

Démonstration de la proposition 5.4.

Observons tout d'abord que,

$$\begin{aligned}\mathbf{e} &= \varphi(\mathbf{e}_U, \mathbf{e}_V)\\ &= (\mathbf{a}\odot\mathbf{e}_U + \mathbf{b}\odot\mathbf{e}_V, \mathbf{c}\odot\mathbf{e}_U + \mathbf{d}\odot\mathbf{e}_V)\\ &= (\mathbf{e}_U\mathbf{A} + \mathbf{e}_V\mathbf{B}, \mathbf{e}_U\mathbf{C} + \mathbf{e}_V\mathbf{D})\end{aligned}$$

où $\mathbf{A} = \mathbf{Diag}(\mathbf{a}), \mathbf{B} = \mathbf{Diag}(\mathbf{b}), \mathbf{C} = \mathbf{Diag}(\mathbf{c}), \mathbf{D} = \mathbf{Diag}(\mathbf{d})$. Donc d'après cette équation, $\mathbf{eH}^\intercal = \mathbf{s}$ se traduit en :

$$\begin{cases} \mathbf{e}_U\mathbf{A}\mathbf{D}^\intercal\mathbf{H}_U^\intercal + \mathbf{e}_V\mathbf{B}\mathbf{D}^\intercal\mathbf{H}_U^\intercal - \mathbf{e}_U\mathbf{C}\mathbf{B}^\intercal\mathbf{H}_U^\intercal - \mathbf{e}_V\mathbf{D}\mathbf{B}^\intercal\mathbf{H}_U^\intercal &= \mathbf{s}^U \\ -\mathbf{e}_U\mathbf{A}\mathbf{C}^\intercal\mathbf{H}_V^\intercal - \mathbf{e}_V\mathbf{B}\mathbf{C}^\intercal\mathbf{H}_V^\intercal + \mathbf{e}_U\mathbf{C}\mathbf{A}^\intercal\mathbf{H}_V^\intercal + \mathbf{e}_V\mathbf{D}\mathbf{A}^\intercal\mathbf{H}_V^\intercal &= \mathbf{s}^V \end{cases}$$

ce qui donne,

$$\mathbf{e}_U(\mathbf{AD} - \mathbf{BC})\mathbf{H}_U^\intercal = \mathbf{s}^U \quad \text{et} \quad \mathbf{e}_V(\mathbf{AD} - \mathbf{BC})\mathbf{H}_V^\intercal = \mathbf{s}^V.$$

Les matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ étant diagonales, donc symétriques, elles commutent. Nous terminons la preuve en notant que $\mathbf{AD} - \mathbf{BC} = \mathbf{1}_{n/2}$. □

Étant donné que nous n'avons fait aucune hypothèse sur les codes U et V (ils seront aléatoires dans notre instanciation), nous proposons d'utiliser pour les décoder l'algorithme de Prange [Pra62] que nous avons généralisé dans §3.1.1. Rappelons que ce dernier, lors du décodage d'un code de longueur n et dimension k, choisit la valeur de la solution sur k symboles puis complète sur $n-k$ symboles dits de redondance. Le problème étant maintenant que si l'on utilise cet algorithme indépendamment pour décoder U et V nous obtiendrons des erreurs dont l'espérance de poids est dans l'intervalle $[\![w^-_{\text{easy}}, w^+_{\text{easy}}]\!]$. En revanche, notons que toute erreur $\mathbf{e} = \varphi(\mathbf{e}_U, \mathbf{e}_V)$ obtenue après le décodage vérifie pour $i \in [\![1, n/2]\!]$:

$$\begin{cases} a_i\mathbf{e}_U(i) + b_i\mathbf{e}_V(i) = \mathbf{e}(i) \\ c_i\mathbf{e}_U(i) + d_i\mathbf{e}_V(i) = \mathbf{e}(i + n/2) \end{cases} \tag{5.9}$$

De cette façon, nous allons voir que nous serons en mesure d'obtenir en temps polynomial des solutions de poids hors de l'intervalle $[\![w^-_{\text{easy}}, w^+_{\text{easy}}]\!]$ si nous décodons :

1. $\mathbf{s}^V$ dans le code V avec l'algorithme de Prange pour obtenir $\mathbf{e}_V$,
2. puis $\mathbf{s}^U$ dans le code U avec l'algorithme de Prange *où nous choisissons avec précaution* $\mathbf{e}_U(i)$ *sur* $k_U = \dim(U)$ *positions en fonction de la valeur de* $\mathbf{e}_V(i)$.

Décrivons alors comment choisir "avec précaution" $\mathbf{e}_U(i)$ pour obtenir des solutions de grand poids.

Ajuster l'algorithme de Prange pour obtenir des solutions de gros poids avec des codes $(U, U+V)$-généralisés. Une fois le vecteur $\mathbf{e}_V$ obtenu, l'idée est de choisir $\mathbf{e}_U(i)$ sur k_U positions données par un ensemble d'information comme :

$$\begin{cases} a_i\mathbf{e}_U(i) + b_i\mathbf{e}_V(i) \neq 0 \\ c_i\mathbf{e}_U(i) + d_i\mathbf{e}_V(i) \neq 0 \end{cases}$$

Cela est alors toujours possible, *quelle que soit la valeur de* $\mathbf{e}_V(i)$, car

$$a_ic_i \neq 0 \quad \textit{et} \quad q \geqslant 3.$$

De cette façon, une fois l'algorithme de Prange terminé, nous obtenons avec la proposition 5.4 une erreur de longueur n telle que :

— $2k_U$ coordonnées sont $\neq 0$,

— les $n - 2k_U$ autres coordonnées sont uniformément distribuées sur $\mathbb{F}_q$.

La poids attendu de la solution est alors donné par :

$$\mathbb{E}(|\mathbf{e}|) = 2k_U + \frac{q-1}{q}(n-2k_U) = \frac{q-1}{q}n + \frac{2k_U}{q}$$

Le meilleur choix est de prendre $k_U = k$ jusqu'au moment où $\frac{q-1}{q}n + \frac{2k}{q} = n$, c'est à dire $k = n/2$. Si les valeurs de k sont choisies plus grandes, il suffit de prendre $k_U = n/2$ et $k_V = k - k_U$. Nous pouvons alors finalement obtenir des solutions en temps polynomial de poids :

$$w^+_{\mathsf{UV}} \triangleq \begin{cases} \frac{q-1}{q}n + \frac{2k}{q} & \text{si } k \leqslant n/2 \\ n & \text{sinon.} \end{cases}$$

On peut facilement vérifier que le poids des erreurs renvoyées par cette procédure est $\geqslant (1+\varepsilon)w^+_{\mathsf{easy}}$ pour tout $k \in]\!]0, n[\!]$ ce qui donne un avantage. Remarquons cependant que nous aurions pu tout aussi bien ajuster cette stratégie pour obtenir des erreurs de petit poids. En revanche, comme montré dans ce qui suit cette dernière est "moins efficace".

Une trappe moins efficace pour obtenir des erreurs de petits poids. De même que précédemment, une fois le vecteur $\mathbf{e}_V$ obtenu, si nous souhaitons obtenir une erreur de petit poids, la meilleure stratégie est la suivante. On choisit $\mathbf{e}_U(i)$ avec l'algorithme de Prange sur k_U positions telles que :

$$\begin{cases} a_i\mathbf{e}_U(i) + b_i\mathbf{e}_V(i) = 0 \\ c_i\mathbf{e}_U(i) + d_i\mathbf{e}_V(i) = 0 \end{cases}$$

Il n'existe alors une solution $\mathbf{e}_U(i)$ que si $\mathbf{e}_V(i) = 0$. En effet, dans le cas où $\mathbf{e}_V(i) \neq 0$, du fait que $a_id_i - b_ic_i \neq 0$, aucune solution $\mathbf{e}_U(i)$ n'est possible. Ainsi, contrairement au cas où nous souhaitions avoir des erreurs de gros poids, l'ensemble d'indices où l'algorithme permet de gagner deux fois est contraint à être de taille $n/2 - |\mathbf{e}_V|$. Autrement nous ne gagnons qu'une seule fois en choisissant bien les symboles lors du décodage de U. Le poids minimal que nous pouvons espérer en temps polynomial pour $\mathbf{e}_V$ avec l'algorithme de Prange est donné par $\frac{q-1}{q}(n/2 - k_V)$. De cette façon :

- si $k_U \leqslant n/2 - \frac{q-1}{q}(n/2 - k_V)$, nous pouvons obtenir une erreur dont :
  - $2k_U$ coordonnées sont égales à 0,
  - les $n - 2k_U$ autres coordonnées sont uniformément distribuées.
- si $k_U > n/2 - \frac{q-1}{q}(n/2 - k_V)$, nous pouvons obtenir une erreur dont :
  - $2\left(n/2 - \frac{q-1}{q}(n/2 - k_V)\right) = \frac{n}{q} + 2\frac{q-1}{q}k_V$ coordonnées sont égales à 0,
  - $k_U - \left(n/2 - \frac{q-1}{q}(n/2 - k_V)\right)$ autres coordonnées donnent des symboles nulles tandis qu'encore d'autres $k_U - \left(n/2 - \frac{q-1}{q}(n/2 - k_V)\right)$ symboles sont non nulles,
  - le reste coordonnées contient des symboles uniformément distribuées sur $\mathbb{F}_q$.

On peut dès-lors vérifier que cette stratégie permet d'obtenir en temps polynomial des erreurs de poids :

$$w^-_{\mathsf{UV}} \triangleq \begin{cases} \frac{q-1}{q}(n-2k) & \text{si } k \leqslant \frac{n}{2q} \\ \frac{2(q-1)^2}{(2q-1)q}(n-k) & \text{sinon.} \end{cases}$$

Cependant permettons-nous de donner dans le paragraphe qui suit une esquisse de la preuve donnant w^-_{UV}.

Calcul de w^-_{UV}.

Supposons pour simplifier que nous cherchons à décoder un code $(U, U+V)$ de dimension $k \triangleq k_U + k_V$. L'erreur recherchée est de la forme $\mathbf{e} \triangleq (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$. Nous commençons par décoder le $[n/2, k_V]_q$-code V ce qui donne $\mathbf{e}_V$. Découpons cette erreur en deux comme : $\mathbf{e}_V = (\mathbf{0}, \mathbf{e}''_V)$ où $\mathbf{0} \in \mathbb{F}_q^{n/2-|\mathbf{e}_V|}$ et $\mathbf{e}''_V(i) \neq 0$ pour tout i. Écrivons maintenant $\mathbf{e}_U$ selon ce découpage en deux : $\mathbf{e}_U = (\mathbf{e}'_U, \mathbf{e}''_U)$ avec $\mathbf{e}'_U \in \mathbb{F}_q^{n/2-|\mathbf{e}_V|}$. Supposons maintenant que

$$k_U \geqslant n/2 - |\mathbf{e}_V| \quad \text{(longueur de } \mathbf{e}'_U\text{)}. \tag{5.10}$$

Dans ce cas notre stratégie consiste avec l'algorithme de Prange à choisir $\mathbf{e}'_U = \mathbf{0}$ et $k_U - (n/2 - |\mathbf{e}_V|)$ symboles de $\mathbf{e}''_U$ comme 0. Nous obtenons alors typiquement :

$$|\mathbf{e}_U| = \frac{q-1}{q}(n/2 - k_U) \quad \text{et} \quad |\mathbf{e}_U + \mathbf{e}_V| = \frac{q-1}{q}(n/2 - k_U) + k_U - (n/2 - k_U) \tag{5.11}$$

La deuxième égalité provient du fait que $\mathbf{e}_U + \mathbf{e}_V = (\mathbf{0}, \mathbf{e}''_U + \mathbf{e}''_V)$. Or par construction, $k_U - (n/2 - |\mathbf{e}_V|)$ symboles de $\mathbf{e}''_U$ sont nulles et donc non nulles dans $\mathbf{e}''_U + \mathbf{e}''_V$. Le reste des $n/2 - k_U$ symboles de $\mathbf{e}''_U$ est typiquement uniformément distribué ce qui donne (5.11).
Finalement nous obtenons une erreur de poids typique (sous la condition (5.10)) :

$$2\frac{q-1}{q}(n/2 - k_U) + k_U - (n/2 - |\mathbf{e}_V|). \tag{5.12}$$

Dans notre algorithme $\mathbf{e}_V$ est obtenu avec l'algorithme de Prange simple (que des 0 dans la zone d'information) et son poids typique est alors :

$$|\mathbf{e}_V| = \frac{q-1}{q}(n/2 - k_V) = \frac{q-1}{q}(n/2 - k + k_U).$$

Le poids de l'erreur dans (5.12) sous la condition (5.10) est alors :

$$2\frac{q-1}{q}(n/2 - k_U) + k_U - (n/2 - |\mathbf{e}_V|) = n\left(\frac{2q-3}{2q}\right) + k_U\frac{1}{q} - k\frac{q-1}{q} \tag{5.13}$$

Donc pour minimiser ce poids sous la contrainte (5.10) on doit choisir k_U le plus petit possible, c'est à dire :

$$k_U = n/2 - |\mathbf{e}_V| = \frac{n}{2q} + \frac{q-1}{q}k - \frac{q-1}{q}k_U$$

ce qui donne,

$$\frac{2q-1}{q}k_U = \frac{n}{2q} + \frac{q-1}{q}k \iff k_U = \frac{n}{2(2q-1)} + \frac{q-1}{(2q-1)}k \tag{5.14}$$

Cependant par définition, $k_U \leqslant k$. Notre choix de k_U est donc admissible si et seulement si :

$$\frac{n}{2(2q-1)} + \frac{q-1}{(2q-1)}k \leqslant k \iff \frac{n}{2q} \leqslant k.$$

Nous retrouvons donc la condition de la définition de w^-_{UV}. Maintenant en injectant (5.14) dans (5.13) nous obtenons,

$$\begin{aligned}
&2\frac{q-1}{q}(n/2 - k_U) + k_U - (n/2 - |\mathbf{e}_V|) \\
&\qquad = n\left(\frac{2q-3}{2q}\right) + \frac{n}{2q(2q-1)} + \frac{q-1}{q(2q-1)}k - k\frac{q-1}{q} \\
&\qquad = n\left(\frac{(2q-3)(2q-1)+1}{2q(2q-1)}\right) + k\left(\frac{(q-1)-(q-1)(2q-1)}{q(2q-1)}\right) \\
&\qquad = n\left(\frac{4q^2-8q+4}{2q(2q-1)}\right) + k\left(\frac{-2q^2+4q-2}{q(2q-1)}\right) \\
&\qquad = n\left(\frac{(2q-2)^2}{2q(2q-1)}\right) - k\left(\frac{(2q-2)^2}{2q(2q-1)}\right) \\
&\qquad = \frac{2(q-1)^2}{q(2q-1)}(n-k)
\end{aligned}$$

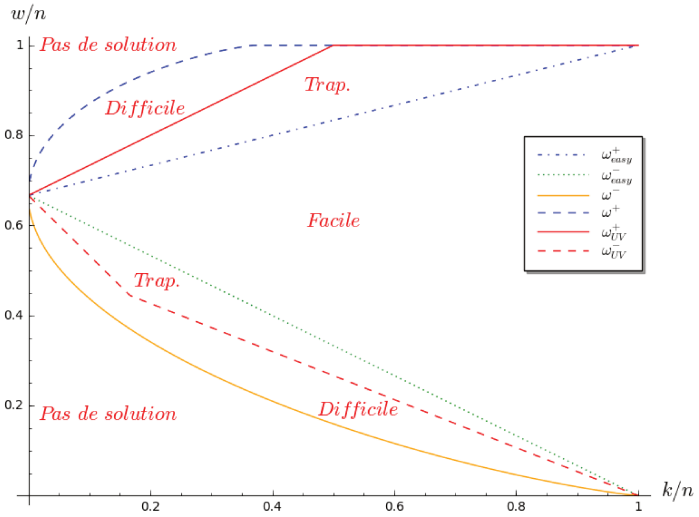


Figure 5.2 – Comparaison des différentes distances relatives w/n avec et sans la trappe des codes $(U, U+V)$-généralisés en fonction du rendement considéré k/n et $q = 3$.

Donc sous la condition (5.10), nous nous attendons typiquement avec notre stratégie à obtenir une erreur de poids $\frac{2(q-1)^2}{q(2q-1)}(n-k)$ si $k \geqslant \frac{n}{2q}$.
En revanche dans le cas où (5.10) n'est pas vérifiée, le même raisonnement montre cette fois-ci que sous la condition $k < \frac{n}{2q}$ nous pouvons obtenir une solution de poids typique $\frac{q-1}{q}(n-2k)$ en choisissant $k = k_U$. □

Nous comparons maintenant dans la figure 5.2 les différentes distances relatives en fonction du rendement k/n :

$$\omega^- \triangleq \frac{w^-}{n}, \quad \omega^+ \triangleq \frac{w^+}{n}, \quad \omega^-_{\text{easy}} \triangleq \frac{w^-_{\text{easy}}}{n}, \quad \omega^+_{\text{easy}} \triangleq \frac{w^+_{\text{easy}}}{n},$$

$$\omega^-_{\text{UV}} \triangleq \frac{w^-_{\text{UV}}}{n} \quad \text{et} \quad \omega^+_{\text{UV}} \triangleq \frac{w^+_{\text{UV}}}{n}.$$

La trappe des codes $(U, U+V)$-généralisés donne donc bien un avantage dans le sens où cette-dernière permet d'inverser la fonction syndrome dans des régimes de paramètres où le problème est génériquement difficile, c'est à dire de complexité exponentielle. En revanche, comme nous pouvons l'observer le "gain" est plus petit dans les zones de poids faible.

Une fuite d'information. Malheureusement la procédure que nous venons de décrire fait fuiter de l'information, *i.e* : les solutions $\mathbf{e} \in f^{-1}_{w,\mathbf{H}}(\mathbf{s})$ révèlent la structure du code $(U, U+V)$-généralisé utilisé. Le problème porte sur les paires de positions de la forme $(i, i+n/2)$. Supposons tout d'abord afin de simplifier la discussion que $q = 3$ et que le code utilisé est un $(U, U+V)$ strict,

$$\forall i \in [\![1, n/2]\!], \quad a_i = c_i = d_i = 1 \quad \text{et} \quad b_i = 0.$$

On peut dès-lors vérifier que pour toute erreur $\mathbf{e} \triangleq (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$ nous avons $|\mathbf{e}_V| = \#\left\{1 \leqslant i \leqslant n/2 : e_i \neq e_{i+n/2}\right\}$. Or $\mathbf{e}_V$ a été obtenu par l'algorithme de Prange et est donc de poids moyen $2/3(n/2 - k_V)$. De cette façon pour tout $i \in [\![1, n/2]\!]$:

$$\mathbb{P}\left(e_i \neq e_{i+n/2}\right) = \frac{1}{n/2}\frac{2}{3}(n/2 - k_V)\left(1 + o(1)\right) \tag{5.15}$$

où la probabilité est calculée sur les solutions fournies par notre procédure. En revanche, les autres paires (i,j) où $|i-j| \neq n/2$ ne sont pas corrélées à la structure $(U, U+V)$. Nous pouvons donc supposer dans ce cas que $\mathbb{P}(e_i \neq e_j)$ est bien approximée par le cas où $\mathbf{e}$ est uniformément distribuée parmi les mots de poids w, c'est à dire :

$$\begin{aligned}\mathbb{P}\left(e_i \neq e_j\right) &= 2\frac{\binom{n-2}{w-2}2^{w-2}}{2^w\binom{n}{w}} + 2^2\frac{\binom{n-2}{w-1}2^{w-1}}{2^w\binom{n}{w}} \\ &= \frac{\binom{n-2}{w-2}}{2\binom{n}{w}} + \frac{2\binom{n-2}{w-1}}{\binom{n}{w}} \\ &= \frac{w(w-1)}{n(n-1)} + 2\frac{w(n-w)}{n(n-1)} \\ &= \frac{4wn - 3w^2 - w}{n(n-1)}. \end{aligned} \tag{5.16}$$

Les deux probabilités (5.15) et (5.16) n'ont alors aucune raison d'être égales. Ceci provoque donc une fuite d'information. Un attaquant peut collectionner des signatures, c'est à dire des permutés de $\mathbf{e}$ puis calculer pour toute paire (i,j) le nombre de fois où $e_i \neq e_j$. Il retrouvera alors les paires de la forme $(i, i+n/2)$ permuté et donc la permutation utilisée, partie intégrante du secret. Il est donc essentiel de modifier notre procédure pour supprimer ce biais. C'est l'objet de ce qui suit.

5.3 Inversion sans fuite d'information : la méthode avec rejet

Nous restreignons à partir de maintenant notre étude au cas où,

$$\boxed{q = 3.}$$

Les résultats que nous allons énoncer se généralisent bien-sûr pour des q plus grands.

Afin d'être fonction une GPVM, nous devons donner un algorithme dont la distribution de sortie est (i) statistiquement proche de l'uniforme sur S_w et (ii) inversant notre fonction à sens unique. La procédure que nous avons décrite dans la précédente sous-section ne vérifie pas cette propriété. En revanche, si nous la modifions légèrement nous l'atteindrons. Ceci a cependant un coût : la zone de poids w que nous pouvons atteindre en temps polynomial est plus restreinte que précédemment. De façon surprenante, nous verrons que la stratégie pour obtenir des erreurs de petits poids permet l'inversion sans fuite d'information de la fonction syndrome avec au mieux des poids w^-_{easy}. La figure 5.3 résume la situation.

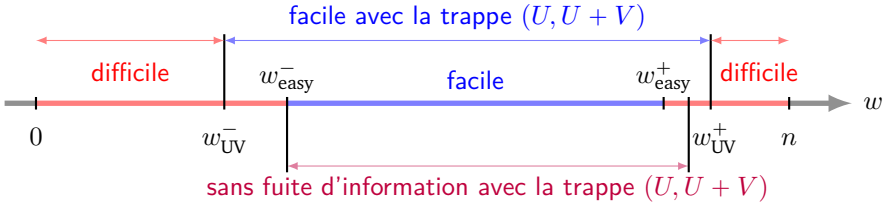


Figure 5.3 – Difficulté du décodage générique avec et sans la trappe $(U, U+V)$.

5.3.1 Méthode avec rejet pour atteindre une distribution uniforme

Notre modification du décodeur des $(U, U+V)$-généralisés va consister à utiliser une méthode de rejet sur le poids de $\mathbf{e}_V$ obtenu lors du premier décodage et une certaine quantité reliée à $\mathbf{e}_U$ qui est la sortie du deuxième décodage. Cependant, pour montrer que notre algorithme produit des erreurs uniformément distribuées sur S_w, il est nécessaire que les vecteurs $\mathbf{e}_V$ et $\mathbf{e}_U$ satisfassent les propriétés suivantes : la distribution de $\mathbf{e}_V$ n'est *que* fonction de son poids et celle de $\mathbf{e}_U$ d'une quantité reliée elle aussi à son poids. Nous verrons alors dans la prochaine section que l'algorithme de Prange est de façon naturelle statistiquement proche de cette propriété.

Le squelette de notre algorithme est donné dans l'algorithme 3. Nous supposons que DECODEV($\mathbf{H}_V, \mathbf{s}^V$) retourne un vecteur $\mathbf{e}_V$ satisfaisant $\mathbf{e}_V\mathbf{H}_V^\intercal = \mathbf{s}^V$ tandis que DECODEU($\mathbf{H}_U, \varphi, \mathbf{s}^U, \mathbf{e}_V$) renvoie un mot $\mathbf{e}_U$ vérifiant $\mathbf{e}_U\mathbf{H}_U^\intercal = \mathbf{s}^U$ *et* $|\varphi(\mathbf{e}_U, \mathbf{e}_V)| = w$. Nous avons ici décomposé le syndrome $\mathbf{s}$ comme $(\mathbf{s}^U, \mathbf{s}^V)$ où $\mathbf{s}^U \in \mathbb{F}_3^{n/2-k_U}$ et $\mathbf{s}^V \in \mathbb{F}_3^{n/2-k_V}$.

Algorithme 3 DECODEUV($\mathbf{H}_V, \mathbf{H}_U, \varphi, \mathbf{s}$)

1: **repeat**
2: $\quad \mathbf{e}_V \leftarrow$ DECODEV($\mathbf{H}_V, \mathbf{s}^V$)
3: **until** Condition 1 est vérifiée
4: **repeat**
5: $\quad \mathbf{e}_U \leftarrow$ DECODEU($\mathbf{H}_U, \varphi, \mathbf{s}^U, \mathbf{e}_V$) ▷ Nous supposons que $|\varphi(\mathbf{e}_U, \mathbf{e}_V)| = w$.
6: $\quad \mathbf{e} \leftarrow \varphi(\mathbf{e}_U, \mathbf{e}_V)$
7: **until** Condition 2 est vérifiée
8: **return** $\mathbf{e}$

Nous utiliserons dans toute la suite la notation suivante :

Notation 14. *Nous désignerons par* $\mathbf{e}^{\text{unif}}$ *la variable aléatoire distribuée comme :*

$$\mathbf{e}^{\text{unif}} \hookleftarrow S_w.$$

Notre objectif est que la distribution de $\mathbf{e}$ soit la même que celle de $\mathbf{e}^{\text{unif}}$. Pour cela nous allons nous assurer grâce aux conditions 1 et 2 que :

1. le vecteur $\mathbf{e}_V$ en entrée de DECODEU$(\cdot)$ à l'étape 5 a la même distribution que $\mathbf{e}_V^{\text{unif}}$,
2. le mot $\mathbf{e}_U$ passant la condition 2 à l'étape 7 conditionné à la valeur de $\mathbf{e}_V$ a la même distribution que $\mathbf{e}_U^{\text{unif}}$ conditionné au vecteur $\mathbf{e}_V^{\text{unif}}$.

La propriété qui suit, inspirée de l'algorithme de Prange, des décodeurs DECODEV$(\cdot)$ et DECODEU$(\cdot)$ nous sera extrêmement utile. Cette dernière suppose que la distribution des sorties des décodeurs ne dépend que du poids.

Définition 5.4 (Décodeurs uniforme en poids et m_1-uniforme)**.**

— DECODEV$(\cdot)$ *est dit* uniforme en poids *si ses sorties* $\mathbf{e}_V =$ DECODEV$(\mathbf{H}_V, \mathbf{s}^V)$ *sont telles que* $\mathbb{P}(\mathbf{e}_V)$ *est seulement une fonction de* $|\mathbf{e}_V|$ *quand* $\mathbf{s}^V \hookleftarrow \mathbb{F}_3^{n/2-k_V}$,

— DECODEU$(\cdot)$ *est dit* m_1*-uniforme si ses sorties* $\mathbf{e}_U =$ DECODEU$(\mathbf{H}_U, \varphi, \mathbf{s}^U, \mathbf{e}_V)$ *vérifient* $\mathbb{P}(\mathbf{e}_U|\mathbf{e}_V)$ *est juste fonction de la paire* $(|\mathbf{e}_V|, m_1(\varphi(\mathbf{e}_U, \mathbf{e}_V))$ *où :*

$$m_1(\mathbf{x}) \stackrel{\triangle}{=} \left|\left\{1 \leqslant i \leqslant n/2 : |(x_i, x_{i+n/2})| = 1\right\}\right|.$$

On vérifie alors facilement, du fait de l'uniformité de $\mathbf{e}^{\text{unif}}$, que pour tout $\mathbf{x} \in S_w$,

$$\mathbb{P}(\mathbf{e}_V^{\text{unif}} = \mathbf{x}_V) \quad \text{et} \quad \mathbb{P}(\mathbf{e}_U^{\text{unif}} = \mathbf{x}_U | \mathbf{e}_V^{\text{unif}} = \mathbf{x}_V)$$

sont seulement fonction de $|\mathbf{x}_V|$ et $(|\mathbf{x}_V|, m_1(\mathbf{x}))$. De cette façon, intuitivement si (i) $\mathbf{e}_V$ et $\mathbf{e}_V^{\text{unif}}$ suivent la même distribution de poids et si (ii) $m_1(\mathbf{e})$ conditionné à la valeur de $|\mathbf{e}_V|$ est distribué comme $m_1(\mathbf{e}^{\text{unif}})$ conditionné à $|\mathbf{e}_V^{\text{unif}}|$ alors nous obtiendrons $\mathbf{e} \sim \mathbf{e}^{\text{unif}}$. C'est l'objet du lemme qui suit.

Lemme 5.2. *Considérons* $\mathbf{e}$ *la sortie de l'algorithme 3 quand* $\mathbf{s}^V$ *et* $\mathbf{s}^U$ *sont uniformément distribués dans leurs domaines. De plus, supposons que* DECODEV$(\cdot)$ *est uniforme en poids et* DECODEU$(\cdot)$ *est* m_1*-uniforme. Supposons maintenant que pour tous* y *et* z *:*

$$|\mathbf{e}_V| \sim |\mathbf{e}_V^{\text{unif}}| \quad \textit{et} \quad \mathbb{P}(m_1(\mathbf{e}) = z \mid |\mathbf{e}_V| = y) = \mathbb{P}(m_1(\mathbf{e}^{\text{unif}}) = z \mid |\mathbf{e}_V^{\text{unif}}| = y).$$

Alors :

$$\mathbf{e} \sim \mathbf{e}^{\text{unif}}$$

où les probabilités sont calculées sur $\mathbf{s}^V$*,* $\mathbf{s}^U$ *et les aléas internes de* DECODEU$(\cdot)$ *et* DECODEV$(\cdot)$*.*

Démonstration du lemme 5.2.

Nous avons pour tout $\mathbf{x} \in S_w$:

$$\begin{aligned}
\mathbb{P}(\mathbf{e} = \mathbf{x}) &= \mathbb{P}(\mathbf{e}_U = \mathbf{x}_U \mid \mathbf{e}_V = \mathbf{x}_V)\mathbb{P}(\mathbf{e}_V = \mathbf{x}_V) \\
&= \mathbb{P}(\text{DECODEU}(\mathbf{H}_U, \varphi, \mathbf{s}^U, \mathbf{e}_V) = \mathbf{x}_U \mid \mathbf{e}_V = \mathbf{x}_V) \\
&\qquad \times \mathbb{P}(\text{DECODEV}(\mathbf{H}_V, \mathbf{s}^V) = \mathbf{x}_V) \\
&= \frac{\mathbb{P}(m_1(\mathbf{e}) = z \mid |\mathbf{e}_V| = y)}{n(y,z)} \frac{\mathbb{P}(|\mathbf{e}_V| = y)}{n(y)} \triangleq P
\end{aligned} \tag{5.17}$$

où

$$n(y) \triangleq |S_{y,n/2}|$$

et la quantité suivante ne dépendant que du poids y de $\mathbf{x}_V$:

$$n(y,z) \triangleq |\{\mathbf{e} \in \mathbb{F}_3^n : \mathbf{e}_V = \mathbf{x}_V \text{ et } m_1(\mathbf{e}) = z\}|.$$

L'équation (5.17) est d'un côté une conséquence de l'uniformité en poids DECODEV$(\cdot)$ et de l'autre de la m_1-uniformité de DECODEU$(\cdot)$. Nous concluons alors la preuve en remarquant que :

$$\begin{aligned}
P &= \frac{\mathbb{P}(m_1(\mathbf{e}^{\text{unif}}) = z \mid |\mathbf{e}_V^{\text{unif}}| = y)}{n(y,z)} \frac{\mathbb{P}(|\mathbf{e}_V^{\text{unif}}| = y)}{n(y)} \\
&= \mathbb{P}(\mathbf{e}_U^{\text{unif}} = \mathbf{x}_U \mid \mathbf{e}_V^{\text{unif}} = \mathbf{x}_V)\mathbb{P}(\mathbf{e}_V^{\text{unif}} = \mathbf{x}_V) \\
&= \mathbb{P}(\mathbf{e}^{\text{unif}} = \mathbf{x})
\end{aligned} \tag{5.18}$$

L'équation (5.18) découle des hypothèses sur les distributions de $|\mathbf{e}_V|$ et celle de $m_1(\mathbf{e})$ conditionnée au poids de $|\mathbf{e}_V|$. □

Ce lemme montre donc qu'il suffit pour affirmer que $\mathbf{e}$ est uniformément distribuée sur S_w de se débrouiller pour que $|\mathbf{e}_V|$ et $m_1(\mathbf{e})$ en fonction de $|\mathbf{e}_V|$ suivent les "bonnes distributions". En d'autres termes, nous allons désormais accepter les sorties $\mathbf{e}_V$ de DECODEV$(\cdot)$ en fonction du poids obtenue $|\mathbf{e}_V|$ avec une certaine probabilité $\mathbf{r}_V(|\mathbf{e}_V|)$ et les sorties $\mathbf{e}$ de DECODEU$(\cdot)$ avec une probabilité $\mathbf{r}_U(|\mathbf{e}_V|, m_1(\mathbf{e}))$. La procédure correspondante est spécifiée dans l'algorithme 4.

Nous pouvons alors démontrer la proposition suivante.

Proposition 5.5. *Considérons pour* $i, t \in [\![0, n/2]\!]$ *et* $s \in [\![0, t]\!]$

$$q_1(i) \triangleq \mathbb{P}\left(|\mathbf{e}_V| = i\right), \; q_1^{\text{unif}}(i) \triangleq \mathbb{P}\left(|\mathbf{e}_V^{\text{unif}}| = i\right) \tag{5.19}$$

$$q_2(s,t) \triangleq \mathbb{P}\left(m_1(\mathbf{e}) = s \mid |\mathbf{e}_V| = t\right), \; q_2^{\text{unif}}(s,t) \triangleq \mathbb{P}\left(m_1(\mathbf{e}^{\text{unif}}) = s \mid |\mathbf{e}_V^{\text{unif}}| = t\right) \tag{5.20}$$

Algorithme 4 DECODEUV($\mathbf{H}_V, \mathbf{H}_U, \varphi, \mathbf{s}$)

repeat
  $\mathbf{e}_V \leftarrow$ DECODEV($\mathbf{H}_V, \mathbf{s}^V$)
until rand($[0,1]$) $\leqslant \mathbf{r}_V(|\mathbf{e}_V|)$ ▷ rand($[0,1]$) renvoie un nombre aléatoire de $[0,1]$
repeat
  $\mathbf{e}_U \leftarrow$ DECODEU($\mathbf{H}_U, \varphi, \mathbf{s}^U, \mathbf{e}_V$)
  $\mathbf{e} \leftarrow \varphi(\mathbf{e}_U, \mathbf{e}_V)$
until rand($[0,1]$) $\leqslant \mathbf{r}_U(|\mathbf{e}_V|, m_1(\mathbf{e}))$
return $\mathbf{e}$

$$\mathbf{r}_V(i) \triangleq \frac{1}{M_V^{rs}} \frac{q_1^{\text{unif}}(i)}{q_1(i)} \quad \textit{et} \quad \mathbf{r}_U(s,t) \triangleq \frac{1}{M_U^{rs}(t)} \frac{q_2^{\text{unif}}(s,t)}{q_2(s,t)}$$

avec

$$M_V^{rs} \triangleq \max_{0 \leqslant i \leqslant n/2} \frac{q_1^{\text{unif}}(i)}{q_1(i)} \quad \textit{et} \quad M_U^{rs}(t) \triangleq \max_{0 \leqslant s \leqslant t} \frac{q_2^{\text{unif}}(s,t)}{q_2(s,t)}$$

Alors, si DECODEV($\cdot$) *est uniforme en poids et* DECODEU($\cdot$) *est m_1-uniforme, la sortie $\mathbf{e}$ de l'algorithme 4 vérifie*

$$\mathbf{e} \sim \mathbf{e}^{\text{unif}}.$$

Démonstration de la proposition 5.5.

Notons $\mathbf{e}_V^{\text{dec}}$ le vecteur obtenue à l'étape 4 de l'algorithme 3. Montrons que $|\mathbf{e}_V^{\text{dec}}|$ et $|\mathbf{e}_V^{\text{unif}}|$ sont équi-distribués. Notons tout d'abord que pour tout i :

$$0 \leqslant \mathbf{r}_V(i) = \frac{1}{M_V^{rs}} \frac{q_1^{\text{unif}}(i)}{q_1(i)} = \left(\inf_{0 \leqslant j \leqslant n/2} \frac{q_1(j)}{q_1^{\text{unif}}(j)} \right) \frac{q_1^{\text{unif}}(i)}{q_1(i)} \leqslant \frac{q_1(i)}{q_1^{\text{unif}}(i)} \frac{q_1^{\text{unif}}(i)}{q_1(i)} = 1$$

ce qui permet d'affirmer que toutes les coordonnées de $\mathbf{r}_V$ sont bien des probabilités. Introduisons maintenant pour tout $i \in [\![1, n]\!]$:

$$q_1^{\text{dec}}(i) \triangleq \mathbb{P}\left(|\mathbf{e}_V^{\text{dec}}| = i \right).$$

Observons alors que :

$$\begin{aligned} q_1^{\text{dec}}(i) &= \frac{\mathbf{r}_V(i) q_1(i)}{\sum_{0 \leqslant j \leqslant n/2} \mathbf{r}_V(j) q_1(j)} \\ &= \frac{q_1^{\text{unif}}(i)}{M_V^{rs} \sum\limits_{0 \leqslant j \leqslant n/2} \frac{1}{M_V^{rs}} q_1^{\text{unif}}(j)} \\ &= q_1^{\text{unif}}(i) \end{aligned}$$

où la première ligne découle de l'indépendance du tirage du réel rand($[0,1]$) et du poids de $|\mathbf{e}_V|$. De plus, lors de la dernière égalité nous avons utilisé le fait que $\sum\limits_{0 \leqslant j \leqslant n/2} q_1^{\text{unif}}(j) = 1$. On vérifie de même la deuxième partie de la proposition ce qui conclut la preuve. □

5.3.2 Application à l'algorithme de Prange

Afin d'instancier la méthode avec rejet, nous allons ici (i) donner les algorithmes DECODEV($\cdot$) et DECODEU($\cdot$) (utilisant des variations de l'algorithme de Prange) pour la stratégie permettant d'obtenir des erreurs de grand poids ainsi que montrer (ii) comment $q_1^{\text{unif}}, q_2^{\text{unif}}, q_1$ et q_2 sont calculées. Nous verrons alors dans la prochaine sous-section comment choisir les paramètres et distributions internes de DECODEV($\cdot$) et DECODEU($\cdot$)

pour que l'algorithme 3 produise des erreurs quasi uniformément distribuées sur S_w avec $w > w^+_{\text{easy}}$ en temps moyen polynomial. En revanche, nous montrerons dans §5.3.6 que si nous avions adopté la stratégie donnant des erreurs de petit poids pour DECODEV$(\cdot)$ et DECODEU$(\cdot)$, nous pourrions au mieux produire en temps polynomial des erreurs uniformément distribuées parmi les mots de poids w^-_{easy}.

Commençons par la proposition qui suit donnant les distributions q_1^{unif} et q_2^{unif}.

Proposition 5.6. *Soit n un entier pair, $w \leqslant n$, $i, t \leqslant n/2$ et $s \leqslant t$ des entiers. Nous avons,*

$$q_1^{\text{unif}}(i) = \frac{\binom{n/2}{i}}{\binom{n}{w}2^{w/2}} \sum_{\substack{p=0 \\ w+p\equiv 0 \mod 2}}^{i} \binom{i}{p}\binom{n/2-i}{(w+p)/2-i}2^{3p/2} \tag{5.21}$$

$$q_2^{\text{unif}}(s,t) = \begin{cases} \frac{\binom{t}{s}\binom{n/2-t}{\frac{w+s}{2}-t}2^{\frac{3s}{2}}}{\sum_p \binom{t}{p}\binom{n/2-t}{\frac{w+p}{2}-t}2^{\frac{3p}{2}}} & \textit{si } w+s \equiv 0 \mod 2 \\ 0 & \textit{sinon} \end{cases} \tag{5.22}$$

Démonstration de la proposition 5.6.

Calculons la distribution q_1^{unif}. Le lemme qui suit nous sera utile.

Lemme 5.3. *Décomposons $\mathbf{e}^{\text{unif}}$ comme $(\mathbf{e}_1, \mathbf{e}_2)$ où $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{F}_3^{n/2}$. Nous avons : $|\mathbf{e}_2 - \mathbf{e}_1| \sim |\mathbf{e}_V^{\text{unif}}|$.*

Démonstration du lemme 5.3.

Considérons,

$$\mathbf{e}'_1 \stackrel{\triangle}{=} \mathbf{c} \odot \mathbf{e}_1, \quad \mathbf{e}'_2 \stackrel{\triangle}{=} \mathbf{a} \odot \mathbf{e}_2 \quad \text{et} \quad \mathbf{e}' \stackrel{\triangle}{=} (\mathbf{e}'_1, \mathbf{e}'_2).$$

Il est clair que $\mathbf{e}' \hookleftarrow S_w$ du fait que les coordonnées de $\mathbf{a}$ et $\mathbf{c}$ sont toutes non nulles. Maintenant,

$$\mathbf{e}_V^{\text{unif}} = -\mathbf{c} \odot \mathbf{e}_1 + \mathbf{a} \odot \mathbf{e}_2 = \mathbf{e}'_2 - \mathbf{e}'_1.$$

Nous en déduisons alors que les erreurs $|\mathbf{e}_2 - \mathbf{e}_1|$ et $|\mathbf{e}_V^{\text{unif}}| = |\mathbf{e}'_2 - \mathbf{e}'_1|$ sont equi-distribuées. □

Ce lemme nous montre donc que pour obtenir la distribution q_1^{unif} il est suffisant de calculer pour tout $i \in [\![1, n/2]\!]$, $\mathbb{P}(|\mathbf{e}_2 - \mathbf{e}_1| = i)$ où $(\mathbf{e}_1, \mathbf{e}_2) \hookleftarrow S_w$. Définissons pour cela les quantités :

$$p \stackrel{\triangle}{=} |\{1 \leqslant i \leqslant n/2 : (\mathbf{e}_1(i), \mathbf{e}_2(i)) \in \{(1,0),(0,1),(-1,0),(0,-1)\}\}| \tag{5.23}$$

$$r \stackrel{\triangle}{=} |\{1 \leqslant i \leqslant n/2 : (\mathbf{e}_1(i), \mathbf{e}_2(i)) \in \{(1,-1),(-1,1)\}\}| \tag{5.24}$$

$$\ell \stackrel{\triangle}{=} |\{1 \leqslant i \leqslant n/2 : (\mathbf{e}_1(i), \mathbf{e}_2(i)) \in \{(1,1),(-1,-1)\}\}| \tag{5.25}$$

Nous avons,

$$w = |\mathbf{e}| = 2\ell + 2r + p \quad \text{et} \quad j = |\mathbf{e}_1 - \mathbf{e}_2| = p + r.$$

De cette façon,

$$p \equiv w \mod 2, \quad r = j - p \quad \text{et} \quad \ell = (w+p)/2 - j.$$

Il s'ensuit, après sommation sur tous les p possibles, que le nombre d'erreurs $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ de poids w telles que $|\mathbf{e}_1 - \mathbf{e}_2| = j$ est donné par

$$\sum_{\substack{p=0 \\ p\equiv w \mod 2}}^{j} \binom{n/2}{j}\binom{j}{p}4^p 2^{j-p}\binom{n/2-j}{\frac{w+p}{2}-j}2^{\frac{w+p}{2}-j} = \sum_{\substack{p=0 \\ p\equiv w \mod 2}}^{j} \binom{n/2}{j}\binom{j}{p}\binom{n/2-j}{\frac{w+p}{2}-j}2^{\frac{w+3p}{2}}$$

ce qui conclut la preuve concernant le calcul de q_1^{unif}. Déterminons maintenant la distribution q_2^{unif}.

Lemme 5.4. *Soit $n'(s,t)$ le nombre de mots $\mathbf{e}^{\text{unif}} = (\mathbf{e}_1, \mathbf{e}_2)$ de poids w vérifiant $|\mathbf{e}_2 - \mathbf{e}_1| = t$ et $m_1(\mathbf{e}^{\text{unif}}) = s$. Nous avons,*

$$n'(s,t) \quad = \quad \begin{cases} \binom{n/2}{t} 2^{w/2} \binom{t}{s} 2^{3s/2} \binom{n/2-t}{\frac{w+s}{2}-t} & \text{si } s \equiv w \bmod 2 \\ 0 \text{ sinon.} \end{cases}$$

Démonstration du lemme 5.4.

Nous utilisons les notations introduites par les équations (5.23), (5.24) et (5.25). Notons que $m_1(\mathbf{e}^{\text{unif}}) = p$. Nous avons pour les mots définissant $n'(s,t)$

$$p = s, \quad r = t - p = t - s \quad \text{et} \quad \ell = \frac{w+p}{2} - t = \frac{w+s}{2} - t.$$

La contrainte $p \equiv w \mod 2$ se traduit alors en $s \equiv w \mod 2$. □

Il suffit pour terminer la preuve de remarquer que :

$$\mathbb{P}(m_1(\mathbf{e}^{\text{unif}}) = s \mid |\mathbf{e}_V| = t) = \frac{n'(s,t)}{\sum_p n'(p,t)}.$$

□

Algorithme 5 DECODEV($\mathbf{H}_V, \mathbf{s}^V$) le décodeur renvoyant $\mathbf{e}_V$ tel que $\mathbf{e}_V \mathbf{H}_V^\intercal = \mathbf{s}^V$ où $\mathbf{H}_V \in \mathbb{F}_3^{(n/2-k_V)\times n/2}$.

Paramètres : $d \in [\![0, k_V]\!]$ et $\mathcal{D}_V$ une distribution sur $[\![0, k_V - d]\!]$

$\mathcal{J}, \mathcal{I} \leftarrow$ ENSEMBLELIBREP($\mathbf{H}_V$)
$\ell \hookleftarrow \mathcal{D}_V$
$\mathbf{x}_V \hookleftarrow \left\{ \mathbf{x} \in \mathbb{F}_3^{n/2} \mid |\mathbf{x}_\mathcal{J}| = \ell, \ \text{Supp}(\mathbf{x}) \subseteq \mathcal{I} \right\}$ ▷ $(\mathbf{x}_V)_{\mathcal{I}\setminus\mathcal{J}}$ est uniforme
$\mathbf{e}_V \leftarrow$ PRANGEALGÈBRELINÉAIRE($\mathbf{H}_V, \mathbf{s}^V, \mathcal{I}, \mathbf{x}_V$)
return $\mathbf{e}_V$

function ENSEMBLELIBREP($\mathbf{H}$)

Require: $\mathbf{H} \in \mathbb{F}_3^{(n-k)\times n}$
Ensure: $\mathcal{I}$ un ensemble d'information de $\langle \mathbf{H} \rangle^\perp$ et $\mathcal{J} \subseteq \mathcal{I}$ de taille $k - d$
repeat
  $\mathcal{J} \hookleftarrow [\![1, n]\!]$ de taille $k - d$
until $\operatorname{rang} \mathbf{H}_{\overline{\mathcal{J}}} = n - k$
repeat
  $\mathcal{J}' \hookleftarrow [\![1, n]\!] \setminus \mathcal{J}$ de taille d
  $\mathcal{I} \leftarrow \mathcal{J} \sqcup \mathcal{J}'$
until $\mathcal{I}$ est un ensemble d'information de $\langle \mathbf{H} \rangle^\perp$
return $\mathcal{J}, \mathcal{I}$

function PRANGEALGÈBRELINÉAIRE($\mathbf{H}, \mathbf{s}, \mathcal{I}, \mathbf{x}$)

Require: $\mathbf{H} \in \mathbb{F}_3^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_3^{n-k}$, $\mathcal{I}$ un ensemble d'information de $\langle \mathbf{H} \rangle^\perp$ et $\mathbf{x} \in \mathbb{F}_3^n$
Ensure: $\mathbf{e}\mathbf{H}^\intercal = \mathbf{s}$ et $\mathbf{e}_\mathcal{I} = \mathbf{x}_\mathcal{I}$
$\mathbf{P} \leftarrow$ une $n \times n$ permutation renvoyant $\mathcal{I}$ sur les k dernières coordonnées
$(\mathbf{A} \mid \mathbf{B}) \leftarrow \mathbf{H}\mathbf{P}$ ▷ $\mathbf{A} \in \mathbb{F}_3^{(n-k)\times(n-k)}$
$(\mathbf{0} \mid \mathbf{e}') \leftarrow \mathbf{x}$ ▷ $\mathbf{e}' \in \mathbb{F}_3^k$
$\mathbf{e} \leftarrow \left((\mathbf{s} - \mathbf{e}'\mathbf{B}^\intercal) \left(\mathbf{A}^{-1}\right)^\intercal, \mathbf{e}' \right) \mathbf{P}^\intercal$
return $\mathbf{e}$

Les décodeurs DECODEV$(\cdot)$, DECODEU$(\cdot)$ sont maintenant décrits dans les algorithmes 5 et 6. Ces derniers sont similaires à la généralisation de l'algorithme de Prange que

Algorithme 6 DECODEU($\mathbf{H}_U, \varphi, \mathbf{s}^U, \mathbf{e}_V$) le décodeur renvoyant $\mathbf{e}_U$ tel que $\mathbf{e}_U\mathbf{H}_U^\intercal = \mathbf{s}^U$ et $|\varphi(\mathbf{e}_U, \mathbf{e}_V)| = w$ où $\mathbf{H}_U \in \mathbb{F}_3^{(n/2-k_U)\times n/2}$.

Paramètres : $d \in [\![0, k_U]\!]$ et $(\mathcal{D}_U^t)_{0\leqslant t\leqslant n/2}$ une famille de distributions sur $[\![\min(0, t + k_U - n/2), t]\!]$.

$t \leftarrow |\mathbf{e}_V|$
$k_{\neq 0} \hookleftarrow \mathcal{D}_U^t$
$k_0 \leftarrow k'_U - k_{\neq 0}$ ▷ $k'_U \triangleq k_U - d$
repeat
  $\mathcal{J}, \mathcal{I} \leftarrow$ ENSEMBLELIBREPW($\mathbf{H}_U, \mathbf{e}_V, k_{\neq 0}$)
  $\mathbf{x}_U \hookleftarrow \{\mathbf{x} \in \mathbb{F}_3^{n/2} \mid \forall j \in \mathcal{J},\ \mathbf{x}(j) \notin \{-\frac{b_i}{a_i}\mathbf{e}_V(i), -\frac{d_i}{c_i}\mathbf{e}_V(i)\} \text{ et } \mathrm{Supp}(\mathbf{x}) \subseteq \mathcal{I}\}$
  $\mathbf{e}_U \leftarrow$ PRANGEALGÈBRELINÉAIRE($\mathbf{H}_U, \mathbf{s}^U, \mathcal{I}, \mathbf{x}_U$)
until $|\varphi(\mathbf{e}_U, \mathbf{e}_V)| = w$
return $\mathbf{e}_U$

function ENSEMBLELIBREPW($\mathbf{H}, \mathbf{x}, k_{\neq 0}$)

Require: $\mathbf{H} \in \mathbb{F}_3^{(n-k)\times n}$, $\mathbf{x} \in \mathbb{F}_3^n$ et $k_{\neq 0} \in [\![0, k]\!]$.
Ensure: $\mathcal{J}$ et $\mathcal{I}$ un ensemble d'information de $\langle\mathbf{H}\rangle^\perp$ tel que $|\{i \in \mathcal{J} : x_i \neq 0\}| = k_{\neq 0}$ et $\mathcal{J} \subset \mathcal{I}$ de taille $k - d$.
repeat
  $\mathcal{J}_1 \hookleftarrow \mathrm{Supp}(\mathbf{x})$ de taille $k_{\neq 0}$
  $\mathcal{J}_2 \hookleftarrow [\![1, n]\!] \backslash \mathrm{Supp}(\mathbf{x})$ de taille $k - d - k_{\neq 0}$.
  $\mathcal{J} \leftarrow \mathcal{J}_1 \sqcup \mathcal{J}_2$
until $\mathbf{rang}\, \mathbf{H}_{\overline{\mathcal{J}}} = n - k$
repeat
  $\mathcal{J}' \hookleftarrow [\![1, n]\!] \backslash \mathcal{J}$ de taille d
  $\mathcal{I} \leftarrow \mathcal{J} \sqcup \mathcal{J}'$
until $\mathcal{I}$ est un ensemble d'information de $\langle\mathbf{H}\rangle^\perp$
return $\mathcal{J}, \mathcal{I}$

nous avons décrite dans §3.1.1 avec comme différence le fait que nous avons introduit un paramètre d. Ce dernier sera essentiel pour prouver que les sorties produites pas l'algorithme 3 sont statistiquement indistinguables d'erreurs uniformément distribuées sur S_w. De plus, comme nous le verrons dans la sous-section 5.3.4, une bonne paramétrisation des distributions $\mathcal{D}_V$ et $\mathcal{D}_U^t$ nous permettra de minimiser le nombre de rejets. Nous avons alors :

Proposition 5.7. *Soit n un entier pair, $w \leqslant n$, $i, t, k_U, k_V \leqslant n/2$, $s \leqslant t$ et $d \leqslant \min(k_U, k_V)$ des entiers. De plus, considérons les entiers*

$$k'_V \triangleq k_V - d \quad \textit{et} \quad k'_U \triangleq k_U - d.$$

Soit X_V (respectivement. X_U^t) une variable aléatoire distribuée selon $\mathcal{D}_V$ (respectivement. $\mathcal{D}_U^t$). Nous avons,

$$q_1(i) = \sum_{t=0}^{i} \frac{\binom{n/2-k'_V}{i-t}2^{i-t}}{3^{n/2-k'_V}}\mathbb{P}(X_V = t) \tag{5.26}$$

$$q_2(s,t) = \begin{cases} \displaystyle\sum_{\substack{t+k'_U-n/2\leqslant k_{\neq 0}\leqslant t \\ k_0 \triangleq k'_U - k_{\neq 0}}} \frac{\binom{t-k_{\neq 0}}{s}\binom{n/2-t-k_0}{\frac{w+s}{2}-t-k_0}2^{\frac{3s}{2}}}{\sum_p \binom{t-k_{\neq 0}}{p}\binom{n/2-t-k_0}{\frac{w+p}{2}-t-k_0}2^{\frac{3p}{2}}}\mathbb{P}(X_U^t = k_{\neq 0}) & \textit{si } w \equiv s \bmod 2. \\ 0 & \textit{sinon} \end{cases} \tag{5.27}$$

Démonstration de la proposition 5.7.

Commençons par déterminer q_1. Par définition toute sortie $\mathbf{e}_V$ de DECODEV$(\cdot)$ est de la forme (à une permutation près) :

$$\left(\left(\mathbf{s}^V\mathbf{S}^\intercal - \mathbf{x}_V\mathbf{B}^\intercal\right)\mathbf{A}^{-1^\intercal}, \mathbf{x}_V\right)$$

Rappelons que le vecteur $\mathbf{s}^V$ est uniformément distribué tandis que $\mathbf{x}_V$ se décompose en deux selon $\mathcal{J}$ et $\mathcal{I}\backslash\mathcal{J}$ tel que :

$$(\mathbf{x}_V)_{\mathcal{J}} \hookleftarrow S_{\ell,k_V-d} \text{ où } \ell \hookleftarrow \mathcal{D}_V \quad \text{et} \quad (\mathbf{x}_V)_{\mathcal{I}\backslash\mathcal{J}} \hookleftarrow \mathbb{F}_3^d.$$

Nous pouvons donc écrire le poids de $\mathbf{e}_V$ comme $|\mathbf{e}_V| = S + T$ où S et T sont des variables aléatoires indépendantes telles que S désigne le poids d'un vecteur uniformément distribué sur $\mathbb{F}_3^{n/2-k_V+d}$ et T est distribuée selon $\mathcal{D}_V$. Nous en déduisons alors facilement q_1.

Calculons maintenant q_2 et introduisons pour cela $n(s,t,k_{\neq 0})$ le nombre de différents vecteurs $\mathbf{e}_U$ vérifiant $m_1(\mathbf{e}) = s$ et qui peuvent être renvoyés par DECODEU$(\cdot)$ pour :

— un mot fixé $\mathbf{e}_V$ de poids t,

— un ensemble $\mathcal{J}$ (inclu dans un ensemble d'information $\mathcal{I}$) tel que $\#\mathcal{J} \cap \mathrm{Supp}(\mathbf{e}) = k_{\neq 0}$,

Nous pouvons partitionner $[\![1, n/2]\!]$ comme :

$$[\![1,n/2]\!] = \mathcal{J} \cup \mathcal{I}_1 \cup \mathcal{I}_2 \quad \text{où} \quad \mathcal{I}_1 \triangleq \mathrm{Supp}(\mathbf{e})\backslash\mathcal{J} \quad \text{et} \quad \mathcal{I}_2 \triangleq [\![1,n/2]\!]\backslash\mathcal{I}_1 \cup \mathcal{J}.$$

Par hypothèse sur $\mathbf{e}_U$ nous savons que $|\mathcal{I}_1| = t - k_{\neq 0}$. De plus, $|\mathcal{J}| = k_U - d$ et $\mathcal{I}_2 = n/2 - |\mathcal{J}| - |\mathcal{I}_1| = n/2 - k_U + d - (t - k_{\neq 0}) = n/2 - t - k_0$ où $k_0 \triangleq k_U - d - k_{\neq 0}$. Posons maintenant pour $i \in \{0,1,2\}$:

$$\mathcal{J}_i \triangleq \left\{i \in [\![1,n/2]\!] : |(e_i, e_{i+n/2})| = i\right\} \quad \text{et} \quad j_i \triangleq |\mathcal{J}_i|.$$

Nous avons alors nécessairement :

$$j_1 = s \quad \text{et} \quad n - w = j_1 + 2j_0$$

d'où l'on en déduit que

$$j_0 = \frac{n-w-s}{2}.$$

Notons désormais que

$$\mathcal{J}_1 \subseteq \mathcal{I}_1 \quad \text{et} \quad \mathcal{J}_0 \subseteq \mathcal{I}_2.$$

Nous pouvons choisir comme nous le souhaitons $j_1 = s$ positions de $\mathcal{J}_1$ parmi les $t - k_{\neq 0}$ possibilités données par $\mathcal{I}_1$. De même, nous pouvons choisir $j_0 = \frac{n-w-s}{2}$ positions de $\mathcal{J}_0$ parmi les $n/2 - t - k_0$ de $\mathcal{I}_2$. Remarquons que les coordonnées du vecteur $\mathbf{e}_U$ données par $\mathcal{J}$ sont nécessairement fixées par choix avec l'algorithme de Prange. Il en est de même pour les positions données par $\mathcal{I}_1\backslash\mathcal{J}_1$ et $\mathcal{J}_0$. Concernant les positions i dans $\mathcal{J}_1 \cup (\mathcal{I}_2\backslash\mathcal{J}_0)$ il y a en revanche deux possibilités pour la valeur de $\mathbf{e}_U(i)$. Ceci implique que

$$\begin{aligned} n(s,t,k_{\neq 0}) &= \binom{t-k_{\neq 0}}{s}\binom{n/2-t-k_0}{\frac{n-w-s}{2}} 2^s 2^{n/2-t-k_0-\frac{n-w-s}{2}} \\ &= \binom{t-k_{\neq 0}}{s}\binom{n/2-t-k_0}{\frac{n-w-s}{2}} 2^{\frac{3s}{2}+\frac{w}{2}-t-k_0}. \end{aligned}$$

Nous en déduisons alors que,

$$\begin{aligned}\mathbb{P}(m_1(\mathbf{e}) = s \mid |\mathbf{e}_V| = t,\ \#\mathcal{J} \cap \mathrm{Supp}(\mathbf{e}_V) = k_{\neq 0}) &= \frac{n(s,t,k_{\neq 0})}{\sum_p n(s,t,p)} \\ &= \frac{\binom{t-k_{\neq 0}}{s}\binom{n/2-t-k_0}{\frac{n-w-s}{2}}2^{\frac{3s}{2}+\frac{w}{2}-t-k_0}}{\sum_p \binom{t-k_{\neq 0}}{p}\binom{n/2-t-k_0}{\frac{n-w-p}{2}}2^{\frac{3p}{2}+\frac{w}{2}-t-k_0}} \\ &= \frac{\binom{t-k_{\neq 0}}{s}\binom{n/2-t-k_0}{\frac{n-w-s}{2}}2^{\frac{3s}{2}}}{\sum_p \binom{t-k_{\neq 0}}{p}\binom{n/2-t-k_0}{\frac{n-w-p}{2}}2^{\frac{3p}{2}}}\end{aligned}$$

ce qui conclut la preuve en sommant sur toutes les valeurs $k_{\neq 0}$ possibles.

□

Maintenant que nous connaissons les distributions $q_1, q_1^{\text{unif}}, q_2$ et q_2^{unif} nous pourrions être tenté d'appliquer la proposition 5.5 afin d'affirmer que l'algorithme 3 utilisant les algorithmes 5 et 6 produit des erreurs uniformément distribuées parmi les mots de poids w. Ceci n'est malheureusement pas possible étant donné que les décodeurs DECODEV$(\cdot)$ et DECODEU$(\cdot)$ ne sont pas strictement uniformes en poids et m_1-uniforme. Nous pouvons néanmoins vérifier que cela aurait été le cas si pour les codes U et V tout ensemble de taille k_U et k_V était un ensemble d'information.

Le paramètre d dans les algorithmes DECODEV$(\cdot)$ et DECODEU$(\cdot)$ sert essentiellement à "palier" cela en ayant le rôle suivant. Étant donné $\mathbf{H} \in \mathbb{F}_3^{(n-k)\times n}$ de rang $n-k$, on tire un ensemble $\mathcal{J}$ de taille $k-d$ et on espère que $\mathbf{H}_{\overline{\mathcal{J}}} \in \mathbb{F}_3^{(n-k)\times(n-k+d)}$ est de rang $n-k$, *i.e* : $\mathcal{J}$ est inclu dans un ensemble d'information du $[n,k]_3$ code de matrice de parité $\mathbf{H}$. De cette façon, quand $3^d \approx 2^\lambda$, la probabilité que cela ne soit pas vérifiée sur les matrices $\mathbf{H}$ est de l'ordre de $\frac{1}{2^\lambda}$. Maintenant concernant l'algorithme 5 (resp. 6) nous tirons aléatoirement un ensemble de $k_V - d$ (resp. $k_U - d$) positions. Nous choisissons alors sur ces coordonnées un vecteur de poids donné par la distribution $\mathcal{D}_V$ (resp. $\mathcal{D}_U^t$). Ensuite, avec une probabilité de l'ordre de $1 - \frac{1}{2^\lambda}$, ces ensembles peuvent être complétés avec d positions en un ensemble d'information. Nous tirons alors aléatoirement un vecteur sur ces d positions. Nous terminons ensuite avec l'algorithme de Prange en calculant les $n/2 - k_V$ (resp. $n/2 - k_U$) symboles de redondance distribuées uniformément. De cette façon, nous avons simulé l'algorithme de Prange où les $k_V - d$ (resp. $k_U - d$) positions jouant le rôle d'ensemble d'information et les $n/2 - k_V + d$ (resp. $n/2 - k_U + d$) sont les symboles de redondance uniformément distribués. En revanche, cette fois-ci les $k_V - d$ (resp. $k_U - d$) positions jouant le rôle d'ensemble d'information sont presque uniformément distribuées. Nous sommes alors assez proches des conditions de la définition 5.4 pour prouver le théorème qui suit.

Théorème 5.1. *Soient* $\mathbf{e}$ *la sortie de l'algorithme 4 utilisant les algorithmes 5,6 et* $\mathbf{e}^{\text{unif}}$ *la variable aléatoire uniformément distribuée parmi les mots de poids* w*. Nous avons*

$$\mathbb{P}\left(\rho(\mathbf{e}, \mathbf{e}^{\text{unif}}) > 3^{-d/2}\right) \leqslant 3^{-d/2}$$

où la probabilité est calculée sur $\mathbf{H}_V$ *et* $\mathbf{H}_U$*.*

Notre objectif dans ce qui suit est de démontrer ce théorème à l'aide entre autres de l'inégalité de Markov. Cependant, dans la prochaine sous-section nous donnerons une preuve, pouvant être sautée en première lecture, plus fine et montrant que $\rho(\mathbf{e}, \mathbf{e}^{\text{unif}})$ est typiquement inférieure à $n^2 3^{-d}$. Cette dernière étend essentiellement à l'ordre 2 la preuve que nous allons donner.

Quoiqu'il en soit, la définition qui suit sera essentielle.

Définition 5.5 (Bon et mauvais sous-ensembles)**.** *Considérons les entiers $d \leqslant k \leqslant n$ et $\mathbf{H} \in \mathbb{F}_3^{(n-k)\times n}$. Un sous-ensemble $\mathcal{E} \subseteq [\![1, n]\!]$ de taille $k - d$ est dit* un bon ensemble *pour $\mathbf{H}$ si $\mathbf{H}_{\overline{\mathcal{E}}}$ est de rang plein. Autrement nous dirons que $\mathcal{E}$ est* un mauvais ensemble *de $\mathbf{H}$.*

La preuve du théorème 5.1 repose en partie sur l'introduction d'une variante des deux décodeurs DECODEV$(\cdot)$ et DECODEU$(\cdot)$, que nous noterons VARDECODEV$(\cdot)$ et VARDECODEU$(\cdot)$. Ces-derniers fonctionnent de la même façon que DECODEV$(\cdot)$ et DECODEU$(\cdot)$ quand $\mathcal{J}$ est bon ensemble pour $\mathbf{H}$. En revanche, quand ce n'est pas le cas, nos variations renvoient une erreur aléatoire qui n'aura aucune raison d'être solution du problème de décodage considéré. Cependant, nous choisirons toujours, comme peuvent le faire DECODEV$(\cdot)$ et DECODEU$(\cdot)$, les erreurs $\mathbf{e}_V$ et $\mathbf{e}_U$ selon les distributions $\mathcal{D}_V$ et $\mathcal{D}_U^t$ sur les sous-ensembles $\mathcal{J}$. Notons maintenant que pour les algorithmes VARDECODEV$(\cdot)$ et VARDECODEU$(\cdot)$ nous sommes sur par construction que l'ensemble $\mathcal{J}$ est uniformément distribué. De cette façon, nous pouvons vérifier que :

Fait 6. VARDECODEV$(\cdot)$ *est uniforme en poids tandis que* VARDECODEU$(\cdot)$ *est m_1-uniforme.*

Notre idée de considérer les variations VARDECODEV$(\cdot)$ et VARDECODEU$(\cdot)$ provient du fait que ces algorithmes sont de très bonnes approximations de DECODEV$(\cdot)$ et DECODEU$(\cdot)$ *mais aussi* qu'ils vérifient les propriétés d'uniformité en poids de la définition 5.2. De cette façon, d'après la proposition 5.5, les sorties de l'algorithme 3 utilisant VARDECODEV$(\cdot)$ et VARDECODEU$(\cdot)$ seront uniformes parmi les mots de poids w tout en étant proche du cas où DECODEV$(\cdot)$ et DECODEU$(\cdot)$ auraient été utilisés. Plus précisément, nous avons la proposition suivante.

Proposition 5.8. *La sortie de l'algorithme 3 utilisant les algorithmes* VARDECODEV$(\cdot)$ *et* VARDECODEU$(\cdot)$ *est distribuée comme $\mathbf{e}^{\text{unif}}$. Notons $\mathbf{e}$ la sortie de ce même algorithme mais cette fois-ci avec l'utilisation de* DECODEV$(\cdot)$ *et* DECODEU$(\cdot)$*. Introduisons les deux ensembles :*

— $\mathcal{E}_V \triangleq \{\mathcal{J} \subseteq [\![1, n/2]\!] : |\mathcal{J}| = k_V - d\}$,

— $\mathcal{E}_U(\mathbf{x}_V) \triangleq \{\mathcal{J} \subseteq [\![1, n/2]\!] : |\mathcal{J}| = k_U - d$ *et* $\#\mathcal{J} \cap \mathrm{Supp}(\mathbf{x}_V) = k_{\neq 0}\}$.

Définissons les variables aléatoires :

$$J^{\text{unif}} \hookleftarrow \mathcal{E}_V \quad \textit{et} \quad J^{\mathbf{H}_V} \hookleftarrow \mathcal{E}_V \cap \{\textit{ bons ensembles pour } \mathbf{H}_V\},$$

$$I^{\text{unif}}_{\mathbf{x}_V,\ell} \hookleftarrow \mathcal{E}_U(\mathbf{x}_V) \quad \textit{et} \quad I^{\mathbf{H}_U}_{\mathbf{x}_V,\ell} \hookleftarrow \mathcal{E}_U(\mathbf{x}_V) \cap \{\textit{ bons ensembles pour } \mathbf{H}_U\}.$$

Nous avons :

$$\rho\left(\mathbf{e}, \mathbf{e}^{\text{unif}}\right) \leqslant \rho\left(J^{\mathbf{H}_V}, J^{\text{unif}}\right) + \sum_{\mathbf{x}_V,\ell} \rho\left(I^{\mathbf{H}_U}_{\mathbf{x}_V,\ell}, I^{\text{unif}}_{\mathbf{x}_V,\ell}\right) \mathbb{P}\left(k_{\neq 0} = \ell \mid \mathbf{e}_V = \mathbf{x}_V\right) \mathbb{P}\left(\mathbf{e}_V^{\text{unif}} = \mathbf{x}_V\right)$$

Démonstration de la proposition 5.8.

La première affirmation de cette proposition est une conséquence directe du fait 6 ainsi que du lemme 5.2. Concernant le second point, nous allons utiliser le résultat [GM02, Proposition 8.10] qui suit.

Proposition 5.9. *Soient X et Y deux variables aléatoires à valeurs dans un même espace A. Considérons une troisième variable aléatoire Z à valeurs dans un ensemble B indépendante de X et Y. Alors pour toute fonction f définie sur $A \times B$ nous avons :*

$$\rho\left(f(X, Z), f(Y, Z)\right) \leqslant \rho\left(X, Y\right).$$

Définissons les distributions suivantes pour $\mathbf{x}_V \in \mathbb{F}_3^{n/2}$ et $\mathbf{x}_U \in \mathbb{F}_3^{n/2}$:

$$p(\mathbf{x}_V) \triangleq \mathbb{P}\left(\mathbf{e}_V = \mathbf{x}_V\right) \quad \text{et} \quad q(\mathbf{x}_V) \triangleq \mathbb{P}\left(\mathbf{e}_V^{\text{unif}} = \mathbf{x}_V\right),$$

$$p(\mathbf{x}_U|\mathbf{x}_V) \triangleq \mathbb{P}\left(\mathbf{e}_U = \mathbf{x}_U \mid \mathbf{e}_V = \mathbf{x}_V\right) \quad \text{et} \quad q(\mathbf{x}_V) \triangleq \mathbb{P}\left(\mathbf{e}_V^{\text{unif}} = \mathbf{x}_V\right).$$

Nous avons :

$$\begin{aligned}
\rho\left(\mathbf{e}, \mathbf{e}^{\text{unif}}\right) &= \rho\left(\mathbf{e}_U, \mathbf{e}_V, \mathbf{e}_U^{\text{unif}}, \mathbf{e}_V^{\text{unif}}\right) \\
&= \sum_{\mathbf{x}_V, \mathbf{x}_U} |p(\mathbf{x}_V)p(\mathbf{x}_U|\mathbf{x}_V) - q(\mathbf{x}_V)q(\mathbf{x}_U|\mathbf{x}_V)| \\
&= \sum_{\mathbf{x}_V, \mathbf{x}_U} |(p(\mathbf{x}_V) - q(\mathbf{x}_V))p(\mathbf{x}_U|\mathbf{x}_V) + (p(\mathbf{x}_U|\mathbf{x}_V) - q(\mathbf{x}_U|\mathbf{x}_V))q(\mathbf{x}_V)| \\
&\leqslant \sum_{\mathbf{x}_V, \mathbf{x}_U} |(p(\mathbf{x}_V) - q(\mathbf{x}_V))p(\mathbf{x}_U|\mathbf{x}_V)| + |(p(\mathbf{x}_U|\mathbf{x}_V) - q(\mathbf{x}_U|\mathbf{x}_V)q(\mathbf{x}_V)| \\
&= \sum_{\mathbf{x}_V} |(p(\mathbf{x}_V) - q(\mathbf{x}_V))| + \sum_{\mathbf{x}_V, \mathbf{x}_U} |p(\mathbf{x}_U|\mathbf{x}_V) - q(\mathbf{x}_U|\mathbf{x}_V)|\, q(\mathbf{x}_V) \qquad (5.28)
\end{aligned}$$

où nous avons utilisé dans la dernière ligne le fait que pour tout $\mathbf{x}_V$ nous avons $\sum_{\mathbf{x}_U} |p(\mathbf{x}_U|\mathbf{x}_V)| = 1$. Maintenant, d'après la proposition 5.9 :

$$\sum_{\mathbf{x}_V} |p(\mathbf{x}_V) - q(\mathbf{x}_V)| \leqslant \rho\left(J^{\mathbf{H}_V}, J^{\text{unif}}\right) \qquad (5.29)$$

étant donné que l'aléa interne $\mathcal{D}_V$ de DECODEV$(\cdot)$ est indépendant $J^{\mathbf{H}_V}$ et J^{unif}. Bornons désormais le second terme de l'inégalité (5.28). La distribution de $k_{\neq 0}$ n'est que fonction du poids de l'erreur donnée en entrée de DECODEU$(\cdot)$ ou VARDECODEU$(\cdot)$. De cette façon,

$$\mathbb{P}\left(k_{\neq 0} = \ell \mid \mathbf{e}_V = \mathbf{x}_V\right) = \mathbb{P}\left(k_{\neq 0} = \ell \mid \mathbf{e}_V^{\text{unif}} = \mathbf{x}_V\right). \qquad (5.30)$$

Définissons les distributions,

$$p(\mathbf{x}_U|\mathbf{x}_V, \ell) \triangleq \mathbb{P}(\mathbf{e}_U = \mathbf{x}_U \mid k_{\neq 0} = \ell, \mathbf{e}_V = \mathbf{x}_V),$$

$$q(\mathbf{x}_U|\mathbf{x}_V, \ell) \triangleq \mathbb{P}(\mathbf{e}_U^{\text{unif}} = \mathbf{x}_U \mid k_{\neq 0} = \ell, \mathbf{e}_V^{\text{unif}} = \mathbf{x}_V).$$

Nous déduisons de cette notation et de l'équation (5.30),

$$\begin{aligned}
p(\mathbf{x}_U|\mathbf{x}_V) - q(\mathbf{x}_U|\mathbf{x}_V) = \sum_{\ell} &(p(\mathbf{x}_U|\mathbf{x}_V, \ell) - q(\mathbf{x}_U|\mathbf{x}_V, \ell)) \\
&\times \mathbb{P}\left(k_{\neq 0} = \ell \mid \mathbf{e}_V = \mathbf{x}_V\right) \qquad (5.31)
\end{aligned}$$

L'aléa interne de DECODEU$(\cdot)$ et VARDECODEU$(\cdot)$ est indépendant de $I_{\mathbf{x}_V,\ell}^{\mathbf{H}_U}$ et $I_{\mathbf{x}_V,\ell}^{\text{unif}}$. Donc d'après la proposition 5.9 nous avons pour tout $\mathbf{x}_V$ et ℓ :

$$\sum_{x_U} |p(\mathbf{x}_U|\mathbf{x}_V, \ell) - q(\mathbf{x}_U|\mathbf{x}_V, \ell)| \leqslant \rho\left(I_{\mathbf{x}_V,\ell}^{\mathbf{H}_U}, I_{\mathbf{x}_V,\ell}^{\text{unif}}\right) \qquad (5.32)$$

Nous concluons alors la preuve de la proposition en combinant les équations (5.28), (5.29), (5.31) et (5.32). □

Les espérances de $\rho\left(J^{\mathbf{H}_V}, J^{\text{unif}}\right)$ et $\rho\left(I_{\mathbf{x}_V,\ell}^{\mathbf{H}_U}, I_{\mathbf{x}_V,\ell}^{\text{unif}}\right)$ sont maintenant majorées dans le lemme qui suit.

Lemme 5.5. *Nous avons :*

$$\rho\left(J^{\mathbf{H}_V}, J^{\text{unif}}\right) = \frac{\#\{\mathcal{J} \subseteq [\![1, n/2]\!] \text{ de taille } k_V - d \text{ et mauvais pour } \mathbf{H}_V\}}{\binom{n/2}{k_V - d}} \qquad (5.33)$$

$$\rho\left(I^{\mathbf{H}_U}_{\mathbf{x}_V,\ell}, I^{\text{unif}}_{\mathbf{x}_V,\ell}\right) = \frac{N_{\mathbf{x}_V,\ell}}{\binom{|\mathbf{x}_V|}{\ell}\binom{n/2-|\mathbf{x}_V|}{k_U-d-\ell}} \tag{5.34}$$

$$\mathbb{E}\left(\rho\left(J^{\mathbf{H}_V}, J^{\text{unif}}\right)\right) \leqslant \frac{3^{-d}}{2} \tag{5.35}$$

$$\mathbb{E}\left(\rho\left(I^{\mathbf{H}_U}_{\mathbf{x}_V,\ell}, I^{\text{unif}}_{\mathbf{x}_V,\ell}\right)\right) \leqslant \frac{3^{-d}}{2} \tag{5.36}$$

où $N_{\mathbf{x}_V,\ell}$ est le nombre de sous-ensembles $\subseteq [\![1, n/2]\!]$ de taille $k_U - d$ qui sont mauvais pour $\mathbf{H}_U$ et étant tels que leur intersection avec $\mathrm{Supp}(\mathbf{x}_V)$ est de taille ℓ.

Démonstration du lemme 5.5.

Les équations (5.33), (5.34) se déduisent du fait que la distance statistique entre les distributions uniformes sur $[\![1, s]\!]$ et $[\![1, t]\!]$ (avec $t \geqslant s$) est égale à $\frac{t-s}{t}$.

Indiçons de 1 à $\binom{n/2}{k-d}$ les sous-ensembles de $[\![1, n/2]\!]$ et de taille $k - d$. Notons X_i la variable aléatoire indicatrice de l'évènement "le sous-ensemble i est mauvais pour $\mathbf{H}_V$". Nous avons

$$N = \sum_{i=1}^{\binom{n/2}{k_V-d}} X_i. \tag{5.37}$$

Nous allons maintenant utiliser le lemme qui suit :

Lemme 5.6. *Soient d et m deux entiers où $d < m$ et $\mathbf{M} \hookleftarrow \mathbb{F}_3^{(m-d)\times m}$. Nous avons :*

$$\mathbb{P}\left(\mathbf{rang}(\mathbf{M}) < m - d\right) \leqslant \frac{1}{2 \cdot 3^d}.$$

Démonstration du lemme 5.6.

Soient $\mathbf{M}_1, \ldots, \mathbf{M}_{m-d}$ les lignes de la matrice $\mathbf{M}$. Considérons V_i l'espace vectoriel engendré par $\mathbf{M}_1, \ldots, \mathbf{M}_i$. Si $\mathbf{M}$ n'est pas de rang plein, alors nécessairement pour au moins un $i \in [\![1, m-d]\!]$ nous avons $\dim V_i = \dim V_{i-1} = i - 1$ où $V_{-1} \triangleq \{0\}$. La probabilité P que $\mathbf{M}$ ne soit pas de rang plein est donc majorée par :

$$\begin{aligned}
P &\leqslant \sum_{i=1}^{m-d} \mathbb{P}(\dim V_i = \dim V_{i-1} = i - 1) \\
&\leqslant \sum_{i=1}^{m-d} \mathbb{P}(\dim V_i = i - 1 | \dim V_{i-1} = i - 1) \\
&= \sum_{i=1}^{m-d} \frac{1}{3^{m+1-i}} \\
&\leqslant \frac{1}{2 \cdot 3^d}.
\end{aligned}$$

□

Ce lemme implique alors clairement que $\mathbb{P}(X_i = 1) \leqslant \frac{1}{2\cdot 3^d}$ et

$$\mathbb{E}\left(\rho\left(J^{\mathbf{H}_V}, J^{\text{unif}}\right)\right) = \mathbb{E}\left(\frac{N}{\binom{n/2}{k_V-d}}\right) = \sum_{i=1}^{\binom{n/2}{k_V-d}} \frac{\mathbb{P}(X_i = 1)}{\binom{n/2}{k_V-d}} \leqslant \frac{1}{2 \cdot 3^d}.$$

Cela prouve l'égalité (5.35). La preuve de (5.36) est quant à elle similaire, ce qui conclut notre démonstration du lemme. □

Nous sommes maintenant prêts à démontrer le théorème 5.1.

Démonstration du théorème 5.1.

L'inégalité de Markov implique directement que :

$$\begin{aligned}
&\mathbb{P}\big(\rho(\mathbf{e}, \mathbf{e}^{\text{unif}}) > 3^{-d/2}\big) \\
&\quad \leqslant 3^{d/2}\mathbb{E}\left(\rho(\mathbf{e}, \mathbf{e}^{\text{unif}})\right) \\
&\quad \leqslant 3^{d/2}\mathbb{E}\left(\rho\left(J^{\mathbf{H}_V}, J^{\text{unif}}\right) + \sum_{\mathbf{x}_V,\ell} \rho\left(I^{\mathbf{H}_U}_{\mathbf{x}_V,\ell}, I^{\text{unif}}_{\mathbf{x}_V,\ell}\right) \mathbb{P}\left(k_{\neq 0} = \ell \mid \mathbf{e}_V = \mathbf{x}_V\right) \mathbb{P}\left(\mathbf{e}^{\text{unif}}_V = \mathbf{x}_V\right)\right) \\
&\quad \text{(d'après la proposition 5.8)} \\
&\quad \leqslant 3^{d/2}\left(\frac{3^{-d}}{2} + \sum_{\mathbf{x}_V,\ell} \frac{3^{-d}}{2} \mathbb{P}\left(k_{\neq 0} = \ell \mid \mathbf{e}_V = \mathbf{x}_V\right) \mathbb{P}\left(\mathbf{e}^{\text{unif}}_V = \mathbf{x}_V\right)\right) \quad \text{(d'après le lemme 5.5)} \\
&\quad \leqslant 3^{-d/2}.
\end{aligned}$$

□

5.3.3 Un raffinement du théorème 5.1

Le théorème qui suit donne une bien meilleure borne que celle donnée dans le théorème 5.1. L'idée est de ne plus appliquer l'inégalité de Markov mais celle de Bienaymé-Tchebychev étant plus fine mais faisant intervenir la variance de la distribution.

Théorème 5.2. *Soit* $\mathbf{e}$ *la sortie de l'algorithme 3 avec l'utilisation de* DECODEV$(\cdot)$ *et* DECODEU$(\cdot)$. *Nous avons :*

$$\begin{aligned}
\mathbb{P}\left(\rho(\mathbf{e}, \mathbf{e}^{\text{unif}}) > \frac{(n/2+1)(n/2-k_U+d+1)+1}{3^d}\right) &\leqslant \frac{2}{\binom{n/2}{k-d}}\left(3^d + 2\cdot 3^{2d+\gamma n/2}\right) \\
&\quad + 3^d \sum_{t,\ell} \frac{2 + 4n3^{n\gamma_0/2}}{\binom{n/2}{t}\binom{t}{\ell}\binom{n/2-t}{k-d-\ell}(\alpha_{t,\ell}-1)^2}
\end{aligned}$$

où la probabilité est calculée sur le choix des matrices $\mathbf{H}_V$ *et* $\mathbf{H}_U$ *avec,*

$$\gamma \triangleq \min_{x>0}\left((1-R_V+\delta)\log_3\left(\frac{1+3x}{x}\right) + (R_V-\delta)\log_3(1+x)\right) - 1 + R_V$$

$$\gamma_1(\pi) \triangleq \inf_{x>0} \pi\log_3(1+3x) + (\tau-\pi)\log_3(1+x) - (\tau-\lambda)\log_3(x)$$

$$\begin{aligned}
\gamma_2(\pi) \triangleq \inf_{x>0} &(1-R+\delta-\pi)\log_3(1+3x) + (R-\delta+\pi-\tau)\log_3(1+x) \\
&- (1-R+\delta-\tau+\lambda)\log_3(x)
\end{aligned}$$

$$\begin{aligned}
\gamma_0 \triangleq R - 1 + \sup_{\pi}\big(&\gamma_1(\pi) + \gamma_2(\pi) + (1-R+\delta)h_3\left(\frac{\pi}{1-R+\delta}\right) \\
&+ (R-\delta)h_3\left(\frac{\tau-\pi}{R-\delta}\right)\big)
\end{aligned}$$

$$\alpha_{t,\ell} \triangleq \frac{2}{\mathbb{P}\left(k_{\neq 0} = \ell \mid |\mathbf{e}_V| = t\right)\mathbb{P}(|\mathbf{e}^{\text{unif}}_V| = t)}$$

et

$$\delta = \frac{d}{n/2}, \quad R_V \triangleq \frac{k_V}{n/2}, \quad R_U \triangleq \frac{k_U}{n/2}, \quad \tau \triangleq \frac{t}{n/2}, \quad \lambda \triangleq \frac{\ell}{n/2}.$$

Remarque 5.4. Nous aurons avec les paramètres choisis dans §5.3.4 (où $d = 81$),

$$\mathbb{P}\left(\rho(\mathbf{e}, \mathbf{e}^{\text{unif}}) > \frac{1}{2^{106}}\right) < 2^{-600}.$$

Les deux lemmes qui suivent nous seront utiles pour la démonstration de ce théorème.

Lemme 5.7. *Soient X et Y deux variables aléatoires de Bernoulli qui sont indépendantes conditionnées à un évènement E. Notons $\varepsilon \stackrel{\triangle}{=} \mathbb{P}(\overline{E})$. Alors,*

$$\mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y) \leqslant 2\varepsilon.$$

Démonstration du lemme 5.7.

Nous avons,

$$\begin{aligned}\mathbb{E}(XY) &= \mathbb{P}(X = 1, Y = 1|E)\mathbb{P}(E) + \mathbb{P}(X = 1, Y = 1|\overline{E})\mathbb{P}(\overline{E}) \\ &\leqslant \mathbb{P}(X = 1|E)\mathbb{P}(Y = 1|E)(1 - \varepsilon) + \epsilon.\end{aligned}$$

d'autre part,

$$\mathbb{E}(X)\mathbb{E}(Y) \geqslant \mathbb{P}(X = 1|E)\mathbb{P}(Y = 1|E)(1 - \varepsilon)^2.$$

Nous obtenons alors avec ces deux bornes,

$$\begin{aligned}\mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y) &\leqslant P(X = 1|E)\mathbb{P}(Y = 1|E)(1 - \varepsilon - (1 - \varepsilon)^2) + \varepsilon \\ &\leqslant 2\varepsilon.\end{aligned}$$

□

Lemme 5.8. *Considérons les trois entiers $s = \sigma n$, $t = \tau n$ et $w = \omega n$ tels que $w \leqslant \min(s, t)$. Nous avons,*

$$\sum_{i=0}^{w} \binom{s}{i}\binom{t}{w-i} 3^i \leqslant 3^{\gamma n}$$

où

$$\gamma \stackrel{\triangle}{=} \inf_{x>0} \sigma \log_3(1 + 3x) + \tau \log_3(1 + x) - \omega \log_3(x).$$

Démonstration du lemme 5.8.

Introduisons les trois fonctions polynomiales suivantes :

$$\begin{aligned}a(x) &\stackrel{\triangle}{=} \sum_{j=0}^{s} \binom{s}{j}(3x)^j = (1 + 3x)^s \\ b(x) &\stackrel{\triangle}{=} \sum_{j=0}^{t} \binom{t}{j} x^j = (1 + x)^t \\ c(x) &\stackrel{\triangle}{=} a(x)b(x)\end{aligned}$$

Nous définissons alors les coefficients c_k pour $c(x) = \sum_k c_k t^k$. De cette façon,

$$\begin{aligned}\sum_{i=0}^{w} \binom{s}{i}\binom{t}{w-i} 3^i &= c_w \\ &\leqslant \inf_{x>0} \frac{c(x)}{x^w} \\ &= \inf_{x>0} \frac{a(x)b(x)}{x^w} \\ &= \inf_{x>0} \frac{(1 + 3x)^s(1 + x)^t}{x^w} \\ &= \inf_{x>0} 3^{(\sigma \log_3(1+3x)+\tau \log_3(1+x)-\omega \log_3(x))n} \\ &= 3^{\gamma n}.\end{aligned}$$

□

Afin de prouver le théorème 5.2 nous allons, comme lors de la preuve du théorème 5.1, utiliser de façon cruciale la proposition 5.8. Cette-dernière nous apprend qu'il est suffisant de majorer

$$\rho\left(J^{\mathbf{H}_V}, J^{\text{unif}}\right) \quad \text{et} \quad \sum_{\mathbf{x}_V,\ell} \rho\left(I^{\mathbf{H}_U}_{\mathbf{x}_V,\ell}, I^{\text{unif}}_{\mathbf{x}_V,\ell}\right) \mathbb{P}\left(k_{\neq 0} = \ell \mid \mathbf{e}_V = \mathbf{x}_V\right) \mathbb{P}\left(\mathbf{e}^{\text{unif}}_V = \mathbf{x}_V\right)$$

pour en déduire une borne sur la distance statistique entre des signatures et des mots uniformément distribués sur S_w. Commençons par traiter $\rho\left(J^{\mathbf{H}_V}, J^{\text{unif}}\right)$.

Lemme 5.9. *Considérons* $\mathbf{H} \hookleftarrow \mathbb{F}_3^{(n/2-k)\times n/2}$ *et* $d \in [\![1,k]\!]$. *Nous définissons* $R \stackrel{\triangle}{=} k/(n/2)$ *et* $\delta = d/(n/2)$. *Soit*

$$\mathcal{E} \stackrel{\triangle}{=} \{\mathcal{J} \subseteq [\![1, n/2]\!] : |\mathcal{J}| = k - d\}.$$

Introduisons les deux variables aléatoires :

$$J^{\text{unif}} \hookleftarrow \mathcal{E} \quad \textit{et} \quad J^{\mathbf{H}} \hookleftarrow \mathcal{E} \cap \{\textit{bons ensemble pour } \mathbf{H}\}.$$

Nous avons,

$$\mathbb{P}\left(\rho(J^{\text{unif}}, J^{\mathbf{H}}) > \frac{1}{3^d}\right) \leqslant \frac{2}{\binom{n/2}{k-d}} \left(3^d + 2 \cdot 3^{2d+\gamma n/2}\right)$$

où

$$\gamma \stackrel{\triangle}{=} \min_{x>0} \left((1 - R + \delta) \log_3\left(\frac{1+3x}{x}\right) + (R - \delta)\log_3(1+x)\right) - 1 + R$$

Démonstration du lemme 5.9.

Soit N le nombre de sous-ensembles de $[\![1, n/2]\!]$ de taille $k-d$ qui sont mauvais pour $\mathbf{H}$. Rappelons que la distance statistique entre les distributions uniformes sur $[\![1,s]\!]$ et $[\![1,t]\!]$ (avec $t \geqslant s$) est égale à $\frac{t-s}{t}$. Nous en déduisons alors que

$$\rho(J^{\text{unif}}, J^{\mathbf{H}}) = \frac{N}{\binom{n/2}{k-d}}. \tag{5.38}$$

Indiçons de 1 à $\binom{n/2}{k-d}$ le nombre de sous-ensembles de $[\![1, n/2]\!]$ de taille $k-d$. Considérons la variable aléatoire X_i indicatrice de l'évènement "le sous-ensemble d'indice i est mauvais pour $\mathbf{H}$". Nous avons :

$$N = \sum_{i=1}^{\binom{n/2}{k-d}} X_i. \tag{5.39}$$

Donc en utilisant l'inégalité de Bienaymé-Tchebychev nous en déduisons que pour tout entier t :

$$\begin{aligned}
\mathbb{P}(N > \mathbb{E}(N) + t) \leqslant & \frac{\mathbf{Var}(N)}{t^2} \\
= & \frac{\sum_i \mathbf{Var}(X_i) + \sum_{i\neq j} \mathbb{E}(X_i X_j) - \mathbb{E}(X_i)\mathbb{E}(X_j)}{t^2} \\
\leqslant & \frac{\mathbb{E}(N)}{t^2} + \frac{1}{t^2}\left(\sum_{i\neq j} \mathbb{E}(X_i X_j) - \mathbb{E}(X_i)\mathbb{E}(X_j)\right)
\end{aligned} \tag{5.40}$$

où dans la dernière ligne nous avons utilisé le fait que $\mathbf{Var}(X_i) \leqslant \mathbb{E}(X_i^2)$ et $\mathbb{E}(X_i^2) = \mathbb{E}(X_i)$.

Déterminons maintenant une majoration du second terme de l'inégalité. Nous commençons par définir $\mathcal{E}_{i,j}$ comme étant l'intersection des complémentaires des ensembles indicés par i et j où $i \neq j$.

Par définition d'un mauvais sous-ensemble, il est clair que si $\mathcal{E}_{i,j} = \varnothing$ alors les évènements $X_i = 1$ et $X_j = 1$ sont indépendants et $\mathbb{E}(X_iX_j) = \mathbb{E}(X_i)\mathbb{E}(X_j)$. Autrement, posons

$$e_{i,j} \triangleq |\mathcal{E}_{i,j}| > 0.$$

Observons alors que les variables aléatoires X_i et X_j sont indépendantes conditionnées à l'évènement : $\mathbf{H}_{\mathcal{E}_{i,j}}$ est de rang plein. Nous pouvons alors appliquer le lemme 5.7, ce qui donne pour $e_{i,j} \geqslant 1$:

$$\mathbb{E}(X_iX_j) - \mathbb{E}(X_i)\mathbb{E}(X_j) \leqslant \frac{1}{3^{n/2-k-e_{i,j}}} \tag{5.41}$$

Calculons maintenant en utilisant l'équation (5.41) :

$$\begin{aligned}
&\sum_{i\neq j} \mathbb{E}(X_iX_j) - \mathbb{E}(X_i)\mathbb{E}(X_j) \\
&\quad = \sum_i \sum_{e=1}^{n/2-k+d} \sum_{j:e_{i,j}=e} \mathbb{E}(X_iX_j) - \mathbb{E}(X_i)\mathbb{E}(X_j) \\
&\quad \leqslant \sum_i \sum_{e=1}^{n/2-k+d} \sum_{j:e_{i,j}=e} \frac{1}{3^{n/2-k-e_{i,j}}} \\
&\quad \leqslant \frac{\binom{n/2}{k-d}}{3^{n/2-k}} \sum_{e=0}^{n/2-k+d} 3^e \binom{n/2-k+d}{e}\binom{k-d}{n/2-k+d-e}
\end{aligned} \tag{5.42}$$

Nous terminons alors la preuve en utilisant pour la somme qui précède le lemme 5.8 qui nous donne :

$$\begin{aligned}
\mathbb{P}(N > \mathbb{E}(N) + t) &\leqslant \frac{\mathbb{E}(N)}{t^2} + \frac{1}{t^2}\binom{n/2}{k-d}3^{\gamma n/2} \\
&\leqslant \frac{\binom{n/2}{k-d}}{2\cdot 3^d t^2} + \frac{1}{t^2}\binom{n/2}{k-d}3^{\gamma n/2}
\end{aligned}$$

où nous avons utilisé dans la dernière inégalité le fait que $\mathbb{E}(N) \leqslant \frac{\binom{n/2}{k-d}}{2\cdot 3^d}$ qui est une conséquence directe du lemme 5.6. De cette façon, en choisissant $t = \frac{\binom{n/2}{k-d}}{2\cdot 3^d}$,

$$\mathbb{P}\left(N > \mathbb{E}(N) + \frac{\binom{n/2}{k-d}}{2\cdot 3^d}\right) \leqslant \frac{1}{\binom{n/2}{k-d}}\left(2\cdot 3^d + 4\cdot 3^{2d+\gamma n/2}\right).$$

Mais maintenant comme $\mathbb{E}(N) \leqslant \frac{\binom{n/2}{k-d}}{2\cdot 3^d}$,

$$\mathbb{P}\left(N > \frac{\binom{n/2}{k-d}}{3^d}\right) \leqslant \frac{2}{\binom{n/2}{k-d}}\left(3^d + 2\cdot 3^{2d+\gamma n/2}\right)$$

ce qui conclut la preuve du lemme d'après l'équation (5.38).

□

Lemme 5.10. *Considérons* $\mathbf{H} \hookleftarrow \mathbb{F}_3^{(n/2-k)\times n/2}$. *Soient* $t \in [\![0, n/2]\!]$, $\ell \in [\![0, n/2]\!]$, $R \triangleq \frac{2k}{n}$, $\lambda \triangleq \frac{2\ell}{n}$ *et* $\tau \triangleq \frac{2t}{n}$. *Définissons*

$$\Delta \triangleq \frac{\binom{n/2}{t}\binom{t}{\ell}\binom{n/2-t}{k-d-\ell}}{2\cdot 3^d}(\alpha - 1) \tag{5.43}$$

$$\gamma_1(\pi) \triangleq \inf_{x>0} \pi\log_3(1+3x) + (\tau-\pi)\log_3(1+x) - (\tau-\lambda)\log_3(x)$$

$$\gamma_2(\pi) \stackrel{\triangle}{=} \inf_{x>0} (1-R+\delta-\pi)\log_3(1+3x) + (R-\delta+\pi-\tau)\log_3(1+x) \\ - (1-R+\delta-\tau+\lambda)\log_3(x)$$

$$\gamma_0 \stackrel{\triangle}{=} R - 1 + \sup_{\pi} \gamma_1(\pi) + \gamma_2(\pi) + (1-R+\delta)h_3\left(\frac{\pi}{1-R+\delta}\right) \\ + (R-\delta)h_3\left(\frac{\tau-\pi}{R-\delta}\right)$$

avec α une constante arbitraire vérifiant $\alpha > 1$. Nous avons,

$$\mathbb{P}\left(\frac{1}{\binom{n/2}{t}} \sum_{\mathbf{x}\in\{0,1\}^{n/2}:|\mathbf{x}|=t} \rho(I^{\text{unif}}_{\mathbf{x},\ell}, I^{\mathbf{H}}_{\mathbf{x},\ell}) > \frac{\alpha}{2\cdot 3^d}\right) \leqslant \frac{1}{(\alpha-1)\Delta} + \\ \frac{1}{\Delta^2} n \binom{n/2}{t}\binom{t}{\ell}\binom{n/2-t}{k-d-\ell} 3^{n\gamma_0/2}$$

où nous avons utilisé les notations de la proposition 5.8.

Démonstration du lemme 5.10.

Notons $N_{\mathbf{x},\ell}$ le nombre de sous-ensembles de $[\![1, n/2]\!]$ de taille $k-d$ tels que (i) leur intersection avec $\text{Supp}(\mathbf{x})$ est de taille ℓ et (ii) mauvais pour $\mathbf{H}$. Nous avons

$$\rho(I^{\text{unif}}_{\mathbf{x},\ell}, I^{\mathbf{H}}_{\mathbf{x},\ell}) = \frac{N_{\mathbf{x},\ell}}{\binom{|\mathbf{x}|}{\ell}\binom{n/2-|\mathbf{x}|}{k-d-\ell}} \tag{5.44}$$

Indiçons ces sous-ensembles de 1 à $\binom{|\mathbf{x}|}{\ell}\binom{n/2-|\mathbf{x}|}{k-d-\ell}$. Considérons la variable aléatoire $X_{\mathbf{x},\ell}(i)$ indicatrice de l'évènement "le sous-ensemble d'indice i est mauvais pour $\mathbf{H}$". Il est clair que nous avons

$$N_{\mathbf{x},\ell} = \sum_{i=1}^{\binom{|\mathbf{x}|}{\ell}\binom{n/2-|\mathbf{x}|}{k-d-\ell}} X_{\mathbf{x},\ell}(i) \tag{5.45}$$

Notons,

$$N \stackrel{\triangle}{=} \sum_{\mathbf{x}\in\{0,1\}^{n/2}:|\mathbf{x}|=t} N_{\mathbf{x},\ell} \tag{5.46}$$

Nous avons,

$$\sum_{\mathbf{x},\in\{0,1\}^{n/2}:|\mathbf{x}|=t} \rho(I^{\text{unif}}_{\mathbf{x},\ell}, I^{\mathbf{H}}_{\mathbf{x},\ell}) = N. \tag{5.47}$$

Donc en utilisant l'inégalité de Bienaymé-Tchebychev nous en déduisons que pour tout entier Δ :

$$\begin{aligned}
&\mathbb{P}(N > \mathbb{E}(N) + \Delta) \\
&\quad\leqslant \frac{\mathbf{Var}(N)}{\Delta^2} \\
&\quad= \frac{\sum_{\mathbf{x},i} \mathbf{Var}(X_{\mathbf{x},\ell}(i)) + \sum_{(\mathbf{x},i)\neq(\mathbf{y},j)} \left(\mathbb{E}\left(X_{\mathbf{x},\ell}(i)X_{\mathbf{y},\ell}(j)\right) - \mathbb{E}(X_{\mathbf{x},\ell}(i))\mathbb{E}(X_{\mathbf{y},\ell}(j))\right)}{\Delta^2} \\
&\quad\leqslant \frac{\mathbb{E}(N)}{\Delta^2} + \frac{1}{\Delta^2} \sum_{(\mathbf{x},i)\neq(\mathbf{y},j)} \left(\mathbb{E}\left(X_{\mathbf{x},\ell}(i)X_{\mathbf{y},m}(j)\right) - \mathbb{E}(X_{\mathbf{x},\ell}(i))\mathbb{E}(X_{\mathbf{y},m}(j))\right)
\end{aligned} \tag{5.48}$$

où nous avons utilisé dans la dernière inégalité le fait que $\mathbf{Var}(X_{\mathbf{x},\ell}(i)) \leqslant \mathbb{E}(X_{\mathbf{x},\ell}(i)^2)$ et $\mathbb{E}(X_{\mathbf{x},\ell}(i)^2) = \mathbb{E}(X_{\mathbf{x},\ell}(i))$. Majorons maintenant le second terme de l'inégalité. Commençons par définir $\mathcal{E}(\mathbf{x}, i, \mathbf{y}, j)$ comme l'intersection du complémentaire des ensembles indicés par i et j pour $(\mathbf{x}, i)$ et $(\mathbf{y}, j)$. Posons,

$$e(\mathbf{x}, i, \mathbf{y}, m) \stackrel{\triangle}{=} |\mathcal{E}(\mathbf{x}, i, \mathbf{y}, j)|$$

et supposons que $e(\mathbf{x}, i, \mathbf{y}, j) > 0$. Nous obtenons alors d'après le lemme 5.7 :

$$\mathbb{E}(X_{\mathbf{x},\ell}(i)X_{\mathbf{y},m}(j)) - \mathbb{E}(X_{\mathbf{x},\ell}(i))\mathbb{E}(X_{\mathbf{y},m}(j)) \leqslant \frac{1}{3^{|n/2-k-e(\mathbf{x},i,\mathbf{y},j)|}} \tag{5.49}$$

Lorsque $e(\mathbf{x}, i, \mathbf{y}, j) = 0$, les variables aléatoires $X_{\mathbf{x},\ell}(i)$ et $X_{\mathbf{y},m}(j)$ sont indépendantes et dans ce cas $\mathbb{E}(X_{\mathbf{x},\ell}(i)X_{\mathbf{y},m}(j)) - \mathbb{E}(X_{\mathbf{x},\ell}(i))\mathbb{E}(X_{\mathbf{y},m}(j)) = 0$. Ceci implique :

$$\begin{aligned}
&\sum_{(\mathbf{x},i)\neq(\mathbf{y},j)} (\mathbb{E}(X_{\mathbf{x},\ell}(i)X_{\mathbf{y},\ell}(j)) - \mathbb{E}(X_{\mathbf{x},\ell}(i))\mathbb{E}(X_{\mathbf{y},\ell}(j))) \\
&\leqslant \sum_{(\mathbf{x},i)} \sum_{e=1}^{n/2-k+d} \sum_{(\mathbf{y},j):e(\mathbf{x},i,\mathbf{y},j)=e} (\mathbb{E}(X_{\mathbf{x},\ell}(i)X_{\mathbf{y},m}(j)) - \mathbb{E}(X_{\mathbf{x},\ell}(i))\mathbb{E}(X_{\mathbf{y},m}(j))) \\
&\leqslant \sum_{(\mathbf{x},i)} \sum_{e=1}^{n/2-k+d} \sum_{(\mathbf{y},j):e(\mathbf{x},i,\mathbf{y},j)=e} \frac{1}{3^{|n/2-k-e|}} \quad \text{(d'après l'équation (5.49))}
\end{aligned}$$

Notre objectif est maintenant de calculer la quantité qui suit :

$$S(\mathbf{x}, i, \mathbf{y}) \stackrel{\triangle}{=} \sum_{e=1}^{n/2-k+d} \sum_{j:e(\mathbf{x},i,\mathbf{y},j)=e} \frac{1}{3^{|n/2-k-e|}} \tag{5.50}$$

Notons $\mathcal{E}_i$ (respectivement $\mathcal{F}_j$) le complémentaire de l'ensemble indicé par i (respectivement j) pour $\mathbf{x}$. Posons,

$$p \stackrel{\triangle}{=} |\mathrm{Supp}(\mathbf{y}) \cap \mathcal{E}_i| \,.$$

Il nous sera utile de partitionner $[\![1, n/2]\!]$ comme :

$$\begin{aligned}
[\![1, n/2]\!] = (\mathrm{Supp}(\mathbf{y}) \cap \mathcal{E}_i) \cup (\overline{\mathrm{Supp}(\mathbf{y})} \cap \mathcal{E}_i) \\
\cup (\mathrm{Supp}(\mathbf{y}) \cap \overline{\mathcal{E}_i}) \cup (\overline{\mathrm{Supp}(\mathbf{y})} \cap \overline{\mathcal{E}_i})
\end{aligned} \tag{5.51}$$

Ici,

$$|\mathcal{F}_j| = |\mathcal{E}_i| = n/2 - k + d \quad \text{et} \quad |\overline{\mathcal{E}}_i| = k - d.$$

Or par définition, nous avons $|\,\mathrm{Supp}(\mathbf{y})| = t$. De plus,

$$\begin{aligned}
\left|\overline{\mathrm{Supp}(\mathbf{y})} \cap \mathcal{E}_i\right| &= |\mathcal{E}_i| - |\mathrm{Supp}(\mathbf{y}) \cap \mathcal{E}_i| \\
&= n/2 - k + d - p && (5.52) \\
\left|\mathrm{Supp}(\mathbf{y}) \cap \overline{\mathcal{E}_i}\right| &= |\,\mathrm{Supp}(\mathbf{y})| - |\mathrm{Supp}(\mathbf{y}) \cap \mathcal{E}_i| \\
&= t - p && (5.53) \\
\left|\overline{\mathrm{Supp}(\mathbf{y})} \cap \overline{\mathcal{E}_i}\right| &= \left|\overline{\mathrm{Supp}(\mathbf{y})}\right| - \left|\overline{\mathrm{Supp}(\mathbf{y})} \cap \mathcal{E}_i\right| \\
&= n/2 - t - (n/2 - k + d - p) \\
&= k - d + p - t && (5.54)
\end{aligned}$$

Posons maintenant,

$$f \stackrel{\triangle}{=} |\mathrm{Supp}(\mathbf{y}) \cap \mathcal{E}_i \cap \mathcal{F}_j| \tag{5.55}$$

$$g \stackrel{\triangle}{=} \left|\overline{\mathrm{Supp}(\mathbf{y})} \cap \mathcal{E}_i \cap \mathcal{F}_j\right| \tag{5.56}$$

Observons alors que :

$$e = |\mathcal{E}_i \cap \mathcal{F}_j| = |\mathrm{Supp}(\mathbf{y}) \cap \mathcal{E}_i \cap \mathcal{F}_j| + \left|\overline{\mathrm{Supp}(\mathbf{y})} \cap \mathcal{E}_i \cap \mathcal{F}_j\right| = f + g \tag{5.57}$$

et le fait que

$$|\mathrm{Supp}(\mathbf{y}) \cap \mathcal{F}_j| = |\,\mathrm{Supp}(\mathbf{y})| - \left|\mathrm{Supp}(\mathbf{y}) \cap \overline{\mathcal{F}_j}\right| = t - \ell. \tag{5.58}$$

Calculons maintenant le cardinal de $\mathcal{F}_j$ intersecté avec les ensembles de la partition (5.51). Nous en connaissons déjà deux, déterminons les deux autres.

$$\begin{aligned}\left|\mathrm{Supp}(\mathbf{y}) \cap \overline{\mathcal{E}}_i \cap \mathcal{F}_j\right| &= |\mathrm{Supp}(\mathbf{y}) \cap \mathcal{F}_j| - |\mathrm{Supp}(\mathbf{y}) \cap \mathcal{E}_i \cap \mathcal{F}_j| \\ &= t - \ell - f \qquad (5.59)\\ \left|\overline{\mathrm{Supp}(\mathbf{y})} \cap \mathcal{F}_j \cap \overline{\mathcal{E}}_i\right| &= \left|\overline{\mathrm{Supp}(\mathbf{y})} \cap \mathcal{F}_j\right| - \left|\overline{\mathrm{Supp}(\mathbf{y})} \cap \mathcal{F}_j \cap \mathcal{E}_i\right| \\ &= |\mathcal{F}_j| - |\mathrm{Supp}(\mathbf{y}) \cap \mathcal{F}_j| - \left|\overline{\mathrm{Supp}(\mathbf{y})} \cap \mathcal{F}_j \cap \mathcal{E}_i\right| \\ &= n/2 - k + d - (t-\ell) - g \\ &= n/2 - k + d - t + \ell - g \qquad (5.60)\end{aligned}$$

De cette façon, la quantité $S(\mathbf{x}, i, \mathbf{y})$ de l'équation (5.50) est donnée en sommant sur tous les f et g comme :

$$\begin{aligned}S(\mathbf{x}, i, \mathbf{y}) &= \sum_{f,g} \binom{p}{f}\binom{n/2-k+d-p}{g}\binom{t-p}{t-\ell-f}\binom{k-d+p-t)}{n/2-k+d-t+\ell-g}\frac{1}{3^{|n/2-k-f-g|}} \\ &\leqslant \sum_{f,g} \binom{p}{f}\binom{n/2-k+d-p}{g}\binom{t-p}{t-\ell-f}\binom{k-d+p-t)}{n/2-k+d-t+\ell-g}\frac{1}{3^{n/2-k-f-g}} \\ &= \frac{1}{3^{n/2-k}}\sum_f \binom{p}{f}\binom{t-p}{t-\ell-f}3^f \sum_g \binom{n/2-k+d-p}{g}\binom{k-d+p-t)}{n/2-k+d-t+\ell-g}3^g\end{aligned} \qquad (5.61)$$

Nous utilisons le lemme 5.8 afin de borner (5.61) :

$$S(\mathbf{x}, i, \mathbf{y}) \leqslant 3^{(R-1)n/2}3^{\gamma_1(\pi)n/2}3^{\gamma_2(\pi)n/2} \quad \text{où} \quad \pi \triangleq \frac{2p}{n}.$$

Maintenant, le nombre de vecteurs $\mathbf{y}$ de poids t tels que $|\mathrm{Supp}(\mathbf{y}) \cap \mathcal{E}_i| = p$ est donné par,

$$\binom{n/2-k+d}{p}\binom{k-d}{t-p} \leqslant 3^{\left[(1-R+\delta)h_3\left(\frac{\pi}{1-R+\delta}\right)+(R-\delta)h_3\left(\frac{\tau-\pi}{R-\delta}\right)\right]n/2} \qquad (5.62)$$

Nous en déduisons alors que

$$\begin{aligned}\sum_{(\mathbf{x},i)\neq(\mathbf{y},j)} &\mathbb{E}(X_{\mathbf{x},\ell}(i)X_{\mathbf{y},\ell}(j)) - \mathbb{E}(X_{\mathbf{x},\ell}(i))\mathbb{E}(X_{\mathbf{y},\ell}(j)) \\ &\leqslant n\binom{n/2}{t}\binom{t}{\ell}\binom{n/2-t}{k-d-\ell}3^{n\gamma_0/2} \qquad (5.63)\end{aligned}$$

Nous obtenons en injectant cette équation dans (5.48),

$$\mathbb{P}(N > \mathbb{E}(N) + \Delta) \leqslant \frac{\mathbb{E}(N)}{\Delta^2} + \frac{1}{\Delta^2}n\binom{n/2}{t}\binom{t}{\ell}\binom{n/2-t}{k-d-\ell}3^{n\gamma_0/2}$$

Par définition de Δ (5.43), il est clair d'après le lemme 5.6 que $\mathbb{E}(N) \leqslant \frac{\Delta}{\alpha-1}$. Ainsi

$$\begin{aligned}\mathbb{P}\Big(\frac{1}{\binom{n/2}{t}} &\sum_{\mathbf{x}\in\{0,1\}^{n/2}:|\mathbf{x}|=t} \rho(I^{\mathrm{unif}}_{\mathbf{x},\ell}, I^{\mathbf{H}}_{\mathbf{x},\ell}) > \frac{1}{3^d}\Big) \\ &\leqslant \mathbb{P}(N > \mathbb{E}(N) + \Delta) \\ &\leqslant \frac{1}{(\alpha-1)\Delta} + \frac{1}{\Delta^2}n\binom{n/2}{t}\binom{t}{\ell}\binom{n/2-t}{k-d-\ell}3^{n\gamma_0/2}.\end{aligned}$$

□

Nous sommes maintenant prêts à démontrer le théorème 5.2.

Démonstration du théorème 5.2.

D'après la proposition 5.8,

$$\begin{aligned}\rho\left(\mathbf{e},\mathbf{e}^{\text{unif}}\right) &\leqslant \rho\left(J^{\mathbf{H}_V},J^{\text{unif}}\right)\\ &\quad+\sum_{\mathbf{x}_V\in\mathbb{F}_3^{n/2},\ell}\rho\left(I^{\mathbf{H}_U}_{\mathbf{x}_V,\ell},I^{\text{unif}}_{\mathbf{x}_V,\ell}\right)\mathbb{P}\left(k_{\neq 0}=\ell\mid \mathbf{e}_V=\mathbf{x}_V\right)\mathbb{P}\left(\mathbf{e}_V^{\text{unif}}=\mathbf{x}\right)\\ &=\rho\left(J^{\mathbf{H}_V},J^{\text{unif}}\right)+\sum_{t,\ell}\frac{1}{\binom{n/2}{t}}\sum_{\mathbf{x}\in\{0,1\}^{n/2}:|\mathbf{x}|=t}\rho\left(I^{\mathbf{H}_U}_{\mathbf{x},\ell},I^{\text{unif}}_{\mathbf{x},\ell}\right)\\ &\quad\mathbb{P}\left(k_{\neq 0}=\ell\mid|\mathbf{e}_V^{\text{unif}}|=t\right)\mathbb{P}\left(|\mathbf{e}_V^{\text{unif}}|=t\right)\end{aligned}$$

où nous avons utilisé le fait que $\rho\left(I^{\mathbf{H}_U}_{\mathbf{x},\ell},I^{\text{unif}}_{\mathbf{x},\ell}\right)$ est constant sur les $\mathbf{x}$ ayant le même support. Cela nous a permis de réduire la somme sur les $\mathbf{x}\in\mathbb{F}_3^{n/2}$ à $\{0,1\}^{n/2}$. Donc comme $\mathbb{P}(\cup_i A_i)\leqslant\sum_i\mathbb{P}(A_i)$, $t\in[\![0,n/2]\!]$ et $\ell\in[\![t+k_U-d-n/2,t]\!]$:

$$\begin{aligned}\mathbb{P}\left(\rho\left(\mathbf{e},\mathbf{e}^{\text{unif}}\right)>\frac{(n/2+1)(n/2-k_U+d+1)+1}{3^d}\right)&\\ \leqslant\mathbb{P}\left(\rho\left(J^{\mathbf{H}_V},J^{\text{unif}}\right)>\frac{1}{3^d}\right)&\\ +\sum_{t,\ell}\mathbb{P}\left(\frac{1}{\binom{n/2}{t}}\sum_{\mathbf{x}\{0,1\}^{n/2}:|\mathbf{x}|=t}\rho\left(I^{\mathbf{H}_U}_{\mathbf{x},\ell},I^{\text{unif}}_{\mathbf{x},\ell}\right)\mathbb{P}\left(k_{\neq 0}=\ell\mid|\mathbf{e}_V|=t\right)\mathbb{P}\left(|\mathbf{e}_V^{\text{unif}}|=t\right)>\frac{1}{3^d}\right).&\end{aligned}$$

Observons maintenant que

$$\begin{aligned}&\mathbb{P}\left(\frac{1}{\binom{n/2}{t}}\sum_{\mathbf{x}\{0,1\}^{n/2}:|\mathbf{x}|=t}\rho\left(I^{\mathbf{H}_U}_{\mathbf{x},\ell},I^{\text{unif}}_{\mathbf{x},\ell}\right)\mathbb{P}\left(k_{\neq 0}=\ell\mid|\mathbf{e}_V|=t\right)\mathbb{P}\left(|\mathbf{e}_V^{\text{unif}}|=t\right)>\frac{1}{3^d}\right)\\ &\leqslant\mathbb{P}\left(\frac{1}{\binom{n/2}{t}}\sum_{\mathbf{x}\{0,1\}^{n/2}:|\mathbf{x}|=t}\rho\left(I^{\mathbf{H}_U}_{\mathbf{x},\ell},I^{\text{unif}}_{\mathbf{x},\ell}\right)>\frac{1}{3^d\mathbb{P}\left(k_{\neq 0}=\ell\mid|\mathbf{e}_V|=t\right)\mathbb{P}\left(|\mathbf{e}_V^{\text{unif}}|=t\right)}\right)\end{aligned}\tag{5.64}$$

Posons,

$$\alpha\triangleq\frac{2}{\mathbb{P}\left(k_{\neq 0}=\ell\mid|\mathbf{e}_V|=t\right)\mathbb{P}\left(|\mathbf{e}_V|=t\right)}\geqslant 2\tag{5.65}$$

Il suffit alors pour conclure la preuve du théorème d'appliquer le lemme 5.10 avec α défini dans (5.65) pour tous les termes de (5.64) aussi longtemps que $\alpha\frac{1}{2\cdot 3^d}<1$ (autrement il suffit de borner la probabilité par 0). □

5.3.4 Instanciation des paramètres et distributions internes

Tout choix des distributions $\mathcal{D}_V$ et $\mathcal{D}_U^t$ dans les algorithmes 5 et 6 permet de produire des erreurs de poids w statistiquement proches de l'uniforme. Le problème étant ici le nombre de rejets. Nous pouvons prouver qu'il sera en moyenne donné par :

$$M_V^{rs}+\sum_t q_1(t)M_U^{rs}(t)$$

où

$$M_V^{\text{rs}}\triangleq\max_{0\leqslant i\leqslant n/2}\frac{q_1^{\text{unif}}(i)}{q_1(i)}\quad\text{et}\quad M_U^{\text{rs}}(t)\triangleq\max_{0\leqslant s\leqslant t}\frac{q_2^{\text{unif}}(s,t)}{q_2(s,t)}.\tag{5.66}$$

avec q_1,q_2 et $q_1^{\text{unif}},q_2^{\text{unif}}$ définis dans la proposition 5.5. Or rappelons que d'après la proposition 5.7 les distributions $\mathcal{D}_V$ et $\mathcal{D}_U^t$ interviennent directement dans les différents moments

des distributions q_1 et q_2. Nous allons montrer dans ce qui suit comment les choisir pour que les quantités M_V^{rs} et $M_U^{\text{rs}}(t)$ soient suffisamment proches de 1. Plus particulièrement, nous proposerons de choisir les distributions $\mathcal{D}_V$ et $\mathcal{D}_U^t$ comme des lois de Laplace tronquées.

Définition 5.6 (Loi de Laplace tronquée et discrète (LLTD))**.** *Soient μ, σ deux réels et a, b deux entiers. Une variable aléatoire est distribuée selon une loi de Laplace discrète et tronquée* (LLTD) *pour les paramètres μ, σ, a, b, ce que l'on notera $X \sim \text{Lap}(\mu, \sigma, a, b)$, si pour tout $i \in [\![a, b]\!]$,*

$$\mathbb{P}(X = i) = \frac{e^{-\frac{|i-\mu|}{\sigma}}}{N}$$

où N est un facteur de normalisation.

Remarque 5.5. Si $X \sim \text{Lap}(\mu, \sigma, a, b)$, alors

$$\mathbb{E}(X) \approx \mu \quad \text{et} \quad \text{Var}(X) \approx 2b^2.$$

Nous verrons alors tout au long de cette section que pour minimiser M_V^{rs} et $M_U^{rs}(t)$, les paramètres du systèmes w, k_U, k_V sont contraints et fonctions de $k = k_U + k_V$ et n. A la fin de cette sous-section nous proposerons alors des choix de n, k pour Wave et nous verrons qu'en choisissant correctement les distributions $\mathcal{D}_V$ et $\mathcal{D}_U(t)$ comme des LLTD nous aurons $M_V^{rs} \approx 1.0407$ et en moyenne sur t, $M_U^{\text{rs}}(t) \approx 1.0385$.

Commençons par montrer comment instancier $\mathcal{D}_V$.

Le cas du décodage de $\boldsymbol{V}$. Rappelons que toute erreur renvoyée par DECODEV$(\cdot)$ s'écrit à une permutation près comme :

$$\mathbf{e}_V \stackrel{\triangle}{=} (\mathbf{e}^{\text{info}}, \mathbf{e}^{\text{red}}) \in \mathbb{F}_3^{n/2}$$

où $|\mathbf{e}^{\text{info}}| \hookleftarrow \mathcal{D}_V$ et $\mathbf{e}^{\text{red}} \hookleftarrow \mathbb{F}_3^{n/2-k'_V}$ avec $k'_V = k_V - d$. Rappelons que $\mathbf{e}^{\text{unif}}$ est la variable aléatoire uniformément distribuée sur S_w. Il est clair que pour éviter un trop grand nombre de rejets sur $|\mathbf{e}_V|$, nous devons avoir :

$$\mathbb{E}(|\mathbf{e}_V|) = \mathbb{E}(|\mathbf{e}_V^{\text{unif}}|) \iff \mathbb{E}(\mathcal{D}_V) + \frac{2}{3}(n/2 - k'_V) = \mathbb{E}\left(|\mathbf{e}_V^{\text{unif}}|\right) \tag{5.67}$$

Introduisons alors pour toute la suite le paramètre $\alpha \in [0, 1]$ tel que

$$\mathbb{E}(\mathcal{D}_V) = (1 - \alpha)k'_V \tag{5.68}$$

Déterminons maintenant $\mathbb{E}(\mathbf{e}_V^{\text{unif}})$ pour en déduire d'après les deux équations qui précèdent une contrainte sur les paramètres k'_V, α et w. C'est l'objet du lemme qui suit.

Lemme 5.11. *Nous avons,*

$$\mathbb{E}\left(|\mathbf{e}_V^{\text{unif}}|\right) = \frac{n}{2}\left(1 - \left(1 - \frac{w}{n}\right)^2 - \frac{1}{2}\left(\frac{w}{n}\right)^2\right).$$

Démonstration du lemme 5.11.

Rappelons que d'après le lemme 5.3 :

$$|\mathbf{e}_V^{\text{unif}}| \sim |\mathbf{e}_2 - \mathbf{e}_1| \tag{5.69}$$

où nous avons décomposé en deux parties égales $\mathbf{e}^{\text{unif}} = (\mathbf{e}_1, \mathbf{e}_2)$. Notons maintenant pour tout $i \in [\![1, n]\!]$ la variable aléatoire e_i^{bin} distribuée sur $\mathbb{F}_3$ comme :

$$\mathbb{P}\left(e_i^{\text{bin}} = 1\right) = \mathbb{P}\left(e_i^{\text{bin}} = 2\right) = \frac{w}{2n} \quad \text{et} \quad \mathbb{P}\left(e_i^{\text{bin}} = 0\right) = 1 - \frac{w}{n}.$$

Il est alors clair que les symboles de $\mathbf{e}^{\text{unif}}(i)$ ont la même espérance que les e_i^{bin} pour $1 \leqslant i \leqslant n$. Donc par linéarité nous avons :

$$\begin{aligned}
\mathbb{E}\left(|\mathbf{e}_2 - \mathbf{e}_1|\right) &= \sum_{i=1}^{n/2} \mathbb{E}\left|e_i^{\text{bin}} - e_{i+n/2}^{\text{bin}}\right| \\
&= \sum_{i=1}^{n/2} \left(2\mathbb{P}\left(e_i^{\text{bin}} = 1\right)\mathbb{P}\left(e_i^{\text{bin}} = 2\right) + 4\mathbb{P}\left(e_i^{\text{bin}} = 1\right)\mathbb{P}\left(e_i^{\text{bin}} = 0\right)\right) \\
&= \frac{n}{2}\left(2\left(\frac{w}{2n}\right)^2 + 4\frac{w}{2n}\left(1 - \frac{w}{n}\right)\right) \\
&= \frac{n}{2}\left(\frac{1}{2}\left(\frac{w}{n}\right)^2 + 2\frac{w}{n}\left(1 - \frac{w}{n}\right)\right) \\
&= \frac{n}{2}\left(1 - \left(1 - \frac{w}{n}\right)^2 - \frac{1}{2}\left(\frac{w}{n}\right)^2\right)
\end{aligned}$$

ce qui conclut facilement la preuve avec l'équation (5.69) □

Nous déduisons donc de ce lemme et des équations (5.67), (5.68) l'égalité suivante :

$$(1-\alpha)k'_V + \frac{2}{3}(n/2 - k'_V) = \frac{n}{2}\left(1 - \left(1 - \frac{w}{n}\right)^2 - \frac{1}{2}\left(\frac{w}{n}\right)^2\right) \tag{5.70}$$

De cette façon nous nous assurons que les distributions de poids de l'algorithme DECODEV$(\cdot)$ et $|\mathbf{e}_V^{\text{unif}}|$ (q_1 et q_1^{unif}) ont les mêmes moyennes. Supposons maintenant que nous choisissions :

$$\alpha = \frac{1}{2} \quad \text{et} \quad \mathcal{D}_V \text{ la distribution constante égale à } \frac{1}{2}k'_V.$$

Nous traçons dans la figure 5.4 les distributions q_1 et q_1^{unif} pour les paramètres vérifiant (5.70) :

$$n = 2000, \quad k_V = 383, \quad d = 0 \quad \text{et} \quad w = 1746.$$

Nous constatons dans la première figure que les distributions q_1 et q_1^{unif} sont alignées ce qui est une conséquence de (5.70). En revanche, nous observons dans les queues de distributions que $q_1(i) \ll q_1^{\text{unif}}(i)$. Plus précisément, nous pouvons même montrer que la distribution $q_1(i)$ est dans les queues *exponentiellement faible* devant $q_1^{\text{unif}}(i)$. Nous pouvons donc en déduire avec (5.66) que le nombre de rejets sera dans ce cas exponentiellement grand. L'intuition de ce résultat est la suivante. Nous appliquons la méthode du rejet sur le poids de sortie de l'algorithme DECODEV$(\cdot)$ distribué comme q_1 avec pour objectif d'atteindre la distribution q_1^{unif}. Nous allons donc rejeter certains poids pour "coller" les deux distributions. Ici, du fait que les queues de distribution de q_1 sont exponentiellement faibles devant celles de la distribution cible q_1^{unif}, nous devons augmenter ces probabilités d'apparition. Pour cela il n'y a pas d'autres solutions que de rejeter un nombre exponentiel de fois les poids n'étant pas dans la queue de distribution.

La situation qui précède est problématique car elle mènera à un algorithme fonctionnant en temps exponentiel. Heureusement il existe dans notre cas une solution pour éviter ce écueil, nous pouvons inverser la tendance en faisant que $q_1(i) \gg q_1^{\text{unif}}(i)$ dans les queues de distribution. Il nous suffit pour cela d'augmenter la variance de q_1. L'astuce pour ce faire est de toujours choisir $\mathcal{D}_V$ de moyenne $(1-\alpha)k'_V$ mais cette fois-ci avec une variance élevée. Les distributions en jeu étant de nature exponentielle, nous proposons alors de choisir $\mathcal{D}_V$ comme

$$\text{Lap}((1-\alpha)k_V, \sigma_V, 0, k_V)$$

En effet ce type de loi est exponentielle et a l'avantage que sa variance et espérance sont des paramètres dissociés.

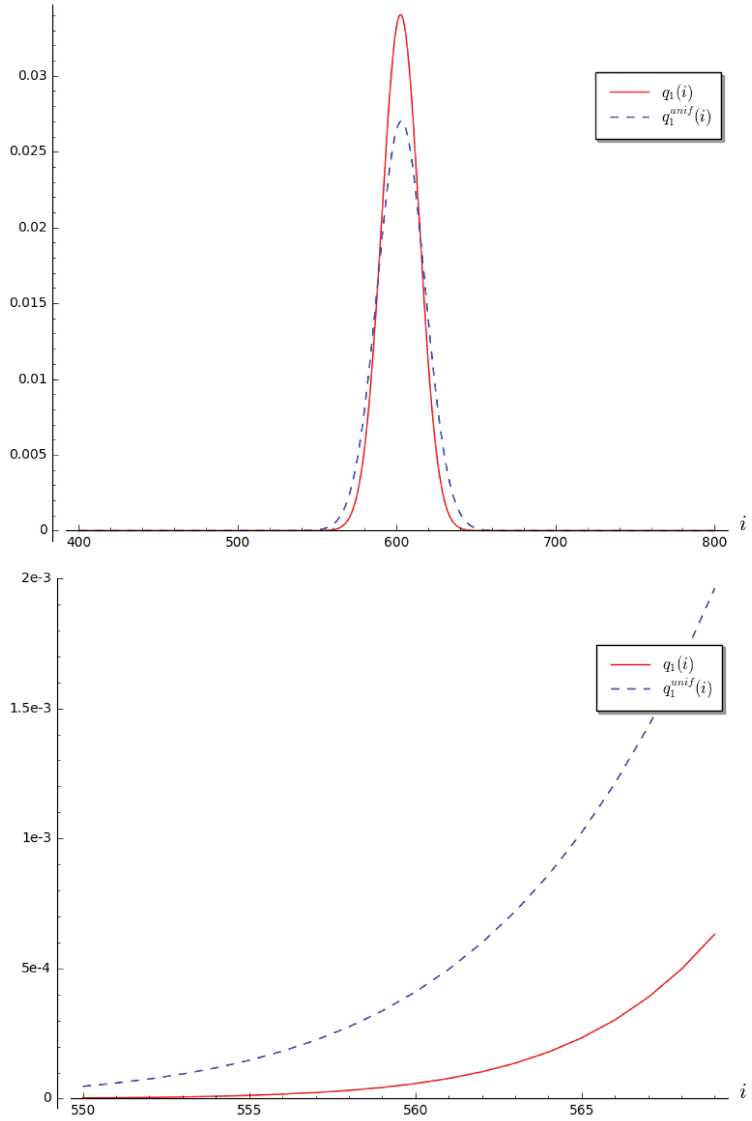


Figure 5.4 – Distributions q_1 et q_1^{unif} pour les paramètres $n = 2000, k_V = 383, d = 0, w = 1746, \alpha = 1/2$ et $\mathcal{D}_V$ la distribution constante égale à $k_V/2$.

Nous traçons dans la figure 5.5 les distributions q_1 et q_1^{unif} pour

$$n = 2000, \quad k_V = 383, \quad d = 0, \quad w = 1746, \quad \alpha = \frac{1}{2} \quad \text{et} \quad \sigma_V = 20.$$

Nous constatons donc bien que $q_1(i) \gg q_1^{\text{unif}}(i)$ dans les queues de distributions. De cette façon, le nombre de rejets donné par le ratio maximal $q_1^{\text{unif}}(i)/q_1(i)$ est atteint autour du poids typique 600 et sera de l'ordre de 2. Nous avons cependant été brutal en choisissant une variance très élevée pour $\mathcal{D}_V$. Pour les paramètres que nous proposons à la fin de cette section, nous avons choisis la variance pour "coller au mieux" les distributions q_1 et q_1^{unif}.

Deux questions peuvent alors être soulevées après cette discussion :

1. Peut-on choisir $\mathcal{D}_V$ de façon à avoir $M_V^{rs} = 1 - 2^{-\lambda}$ où λ est le paramètre de sécurité ?
2. Peut-on prouver qu'en choisissant correctement $\mathcal{D}_V$ le nombre de rejets est asymptotiquement ($n \to +\infty$) polynomial en n ?

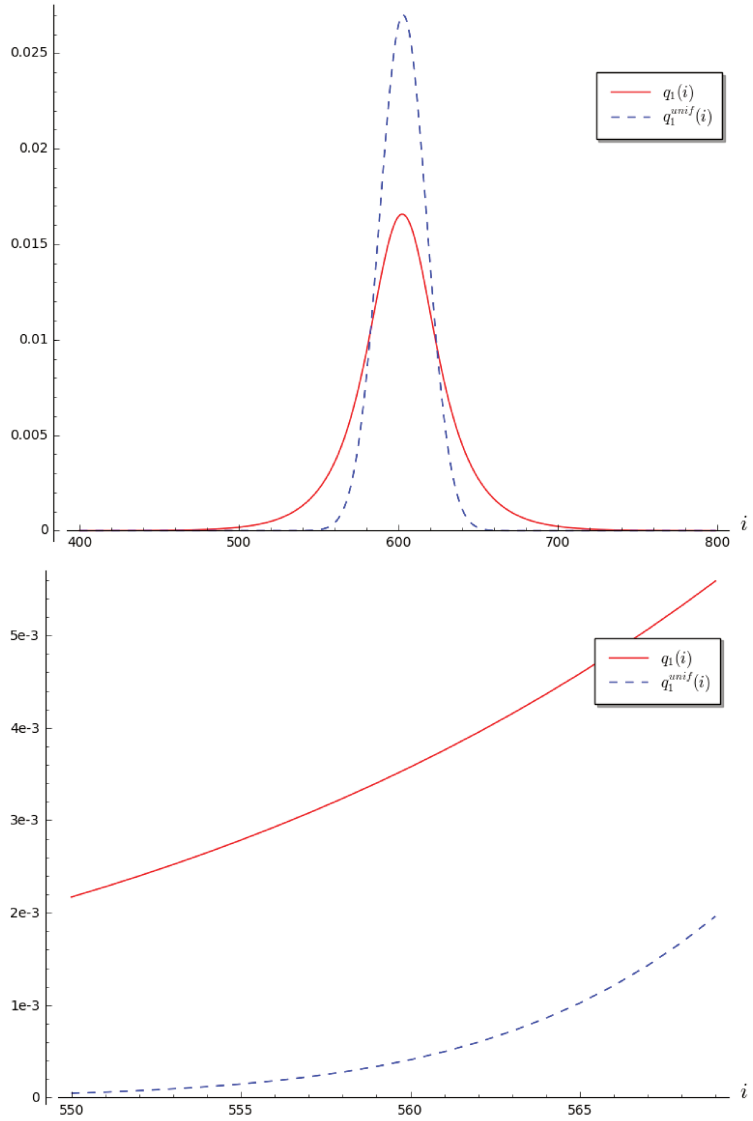


Figure 5.5 – Distributions q_1 et q_1^{unif} pour les paramètres $n = 2000, k_V = 383, d = 0, w = 1746, \alpha = 1/2$ et $\mathcal{D}_V$ distribuée comme $\text{Lap}(1/2k_V, 20, 0, k_V)$.

Ces questions sont ouvertes au moment de la rédaction de ce document. Concernant la première, il ne semble pas qu'un choix "simple" de distribution permette d'atteindre un nombre de rejets négligeable avec le paramètre de sécurité. En revanche, une piste intéressante pour répondre à la deuxième question est de choisir $\mathcal{D}_V$ distribuée sur $[\![0, k'_V]\!]$ comme $q_1^{\text{unif}}\left(i + \mathbb{E}(\mathbf{e}^{\text{unif}}) - (1-\alpha)k'_V\right))$ pour des i admissibles. Quoiqu'il en soit, ces deux questions sont hors du propos de cette thèse, attelons-nous plutôt à montrer comme les $\mathcal{D}_U^t$ sont instanciées.

Le cas du décodage de U. Nous allons comme précédemment pour minimiser le nombre de rejets commencer par choisir les paramètres et $\mathcal{D}_U^t$ de façon à aligner pour tout t les distributions $(q_2(s,t))_s$ et $(q_2^{\text{unif}}(s,t))_s$. Notre démarche sera en revanche plus heuristique que la précédente.

Le point s atteignant le maximum de $s \mapsto q_2^{\text{unif}}(s,t)$ est très bien approximé par la solution de

$$q_2^{\text{unif}}(s-1,t) = q_2^{\text{unif}}(s+1,t)$$

On peut alors vérifier avec l'expression de q_2^{unif} dans la proposition 5.6 et le lemme 1.2 donnant l'exposant asymptotique d'une binomiale qu'un tel s est solution de l'équation

$$\frac{8\,(t-s)\,(t-s+1)\,(n-w-s+1)}{(s+1)\,s\,(w+s+1-2\,t)} = 1.$$

Notons alors pour tout la suite $m_{\text{cible}}^{\max}(t)$ l'unique racine réelle de cette équation.

Reprenons les notation de l'algorithme 6. Nous allons commencer par montrer que le point où la fonction $s \mapsto q_2(s,t)$ atteint son maximum est donné par :

$$\frac{2}{3}\,(t - k_{\neq 0}) \tag{5.71}$$

pour $k_{\neq 0}$ fixé. Rappelons que

$$q_2(s,t) = \mathbb{P}\left(m_1(\mathbf{e}) = s \mid |\mathbf{e}_V| = t\right).$$

où

$$m_1(\mathbf{e}) = \#\left\{1 \leqslant i \leqslant n/2 : |(e_i, e_{i+n/2}) = 1\right\}.$$

Les seules positions i possibles où $|(e_i, e_{i+n/2})|$ peut valoir 1 sont d'après l'instruction 6 dans l'ensemble :

$$\mathcal{E} \stackrel{\triangle}{=} \text{Supp}(\mathbf{e}_V) \backslash \mathcal{J}.$$

Or pour ces $i \in \mathcal{E}$ le symbole e_i est uniformément distribué et distinct de $e_{i+n/2}$. De cette façon une position $i \in \mathcal{E}$ est comptée dans $m_1(\mathbf{e})$ avec une probabilité $2/3$. Maintenant, par définition, $|\mathcal{J}| = k_{\neq 0}$ et $t = |\mathbf{e}_V|$, donc :

$$|\text{Supp}(\mathbf{e}) \backslash \mathcal{J}| = t - k_{\neq 0}.$$

On en déduit alors que typiquement $m_1(\mathbf{e})$ est bien donné par l'équation (5.71). Ainsi, afin d'aligner les distributions $(q_2^{\text{unif}}(s,t))_s$ et $(q_2(s,t))_s$ il est nécessaire de choisir $k_{\neq 0}$ tel que

$$k_{\neq 0} = t - \frac{3}{2} m_{\text{cible}}^{\max}(t) \tag{5.72}$$

Or rappelons maintenant que $k_{\neq 0} \hookleftarrow \mathcal{D}_U^t$ (voir l'instruction 2). Commençons donc par choisir

$$\mathcal{D}_U^t \quad \text{la distribution constante et égale à} \quad t - \frac{3}{2} m_{\text{cible}}^{\max}(t).$$

Nous traçons avec ce choix de $\mathcal{D}_U^t$ dans la figure 5.6 les distributions $(q_2(s,t))_s$ et $(q_2^{\text{unif}}(s,t))_s$ pour

$$n = 2000, \quad w = 1746, \quad k_U = 619, \quad d = 0 \quad \text{et} \quad t = 602.$$

Avec ce jeu de paramètres nous avons $\lfloor m_{\text{cible}}^{\max}(t) \rfloor = 220$.

Nous observons alors le même phénomène que lors du décodage sur V où nous avions choisi $\mathcal{D}_V$ comme une distribution constante : dans les queues de distributions $q_2(s,t) \ll q_2^{\text{unif}}(s,t)$. De même que précédemment nous allons jouer sur $\mathcal{D}_U^t$ pour augmenter la variance de la distribution $q_2(s,t)$. De façon à laisser les courbes alignées, nous commençons par choisir $\mathcal{D}_U^t$ telle que :

$$\mathbb{E}(\mathcal{D}_U^t) = t - \frac{3}{2} m_{\text{cible}}^{\max}(t).$$

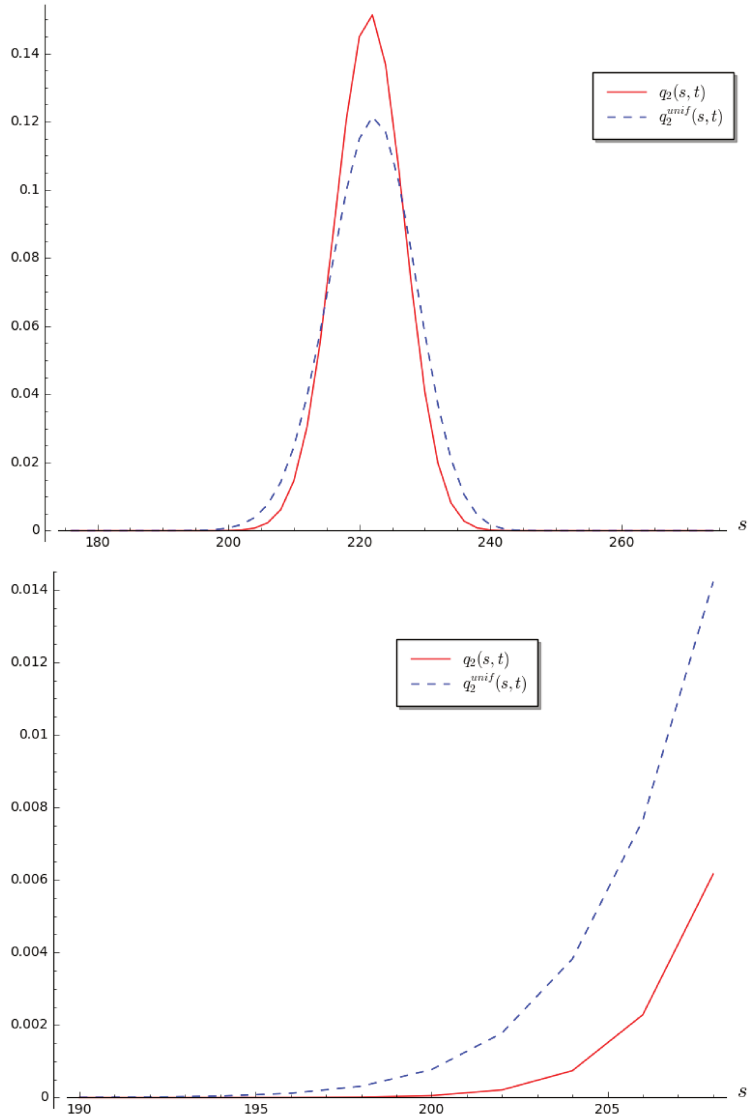


Figure 5.6 – Distributions $(q_2(s,t))_s$ et $(q_2^{\text{unif}}(s,t))_s$ pour les paramètres $n = 2000, w = 1746, k_U = 619, d = 0, t = 602$ et $\mathcal{D}_U$ la distribution constante égale à $t - \frac{3}{2}m^{\max}_{\text{cible}}(t)$.

Nous choisissons ensuite $\mathcal{D}_U^t$ avec une variance suffisamment élevée. Cela joue sur la distribution $q_2(s,t)$ car si $j \hookleftarrow \mathcal{D}_U^t$, alors le $m_1(\mathbf{e})$ typique obtenu avec l'algorithme est :

$$\frac{2}{3}(t-j)$$

d'après le raisonnement qui précède. Donc comme avant nous proposons de choisir $\mathcal{D}_U^t$ distribuée comme une LLTD. Plus précisément nous choisirons $\mathcal{D}_U^t$ distribuée comme :

$$\text{Lap}(t - \frac{3}{2}m^{\max}_{\text{cible}}(t), \sigma_U, \max(0, k_U - d + t - n/2), t).$$

La condition $\max(0, k_U + t - n/2)$ est ici car lorsque j est tiré selon $\mathcal{D}_U^t$, alors nous choisissons $k_U - j$ symboles d'information hors du support de $\mathbf{e}_V$ de taille t. Nous traçons

avec ce choix de $\mathcal{D}_U^t$ dans la figure 5.7 les distributions $(q_2(s,t))_s$ et $(q_1^{\text{unif}}(s,t))_s$ pour

$$n = 2000, \quad w = 1746, \quad k_U = 619, \quad d = 0, \quad t = 602 \quad \text{et} \quad \sigma_U = 6.$$

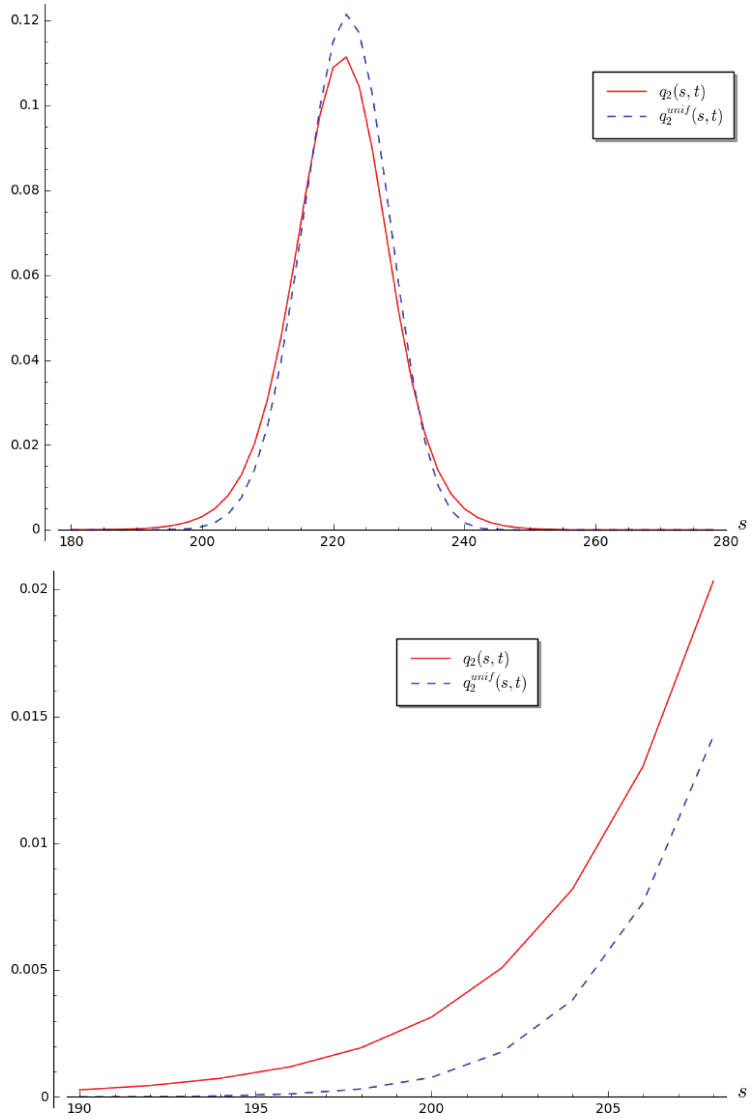


Figure 5.7 – Distributions $(q_2(s,t))_s$ et $(q_2^{\text{unif}}(s,t))_s$ avec les distributions internes $\mathcal{D}_U^t$ choisies comme $\text{Lap}(t - \frac{3}{2}m_{\text{cible}}^{\max}(t), 6, \max(0, k_U + t - n/2), t)$ pour les paramètres $n = 2000, w = 1746, k_U = 619, d = 0$ et $t = 602$.

Nous constatons donc bien que $q_2(s,t) \gg q_2^{\text{unif}}(s,t)$ dans les queues de distributions. Nous avons cependant encore une fois choisi la variance de $\mathcal{D}_U^t$ de façon grossière. Nous ferons alors mieux dans notre proposition de paramètres.

En revanche, après cette discussion nous pouvons comme précédemment nous poser les deux questions suivantes :

1. Peut-on choisir $\mathcal{D}_U^t$ de façon à avoir $M_U^{rs}(t) = 1 - 2^{-\lambda}$ où λ est le paramètre de sécurité ?

2. Peut-on prouver qu'en choisissant correctement $\mathcal{D}_U^t$ le nombre de rejets est asymptotiquement ($n \to +\infty$) polynomial en n ?

Encore une fois il ne semble pas qu'un choix "simple" de distribution permette d'atteindre un nombre de rejets négligeable avec le paramètre de sécurité. Concernant la seconde question, une piste intéressante est de choisir $\mathcal{D}_U^t$ distribuée sur $[\![\max(0, k'_U + t - n/2), t]\!]$ comme

$$q_2^{\text{unif}}\left(s + m_{\text{cible}}^{\max}(t) - t + \frac{3}{2}m_{\text{cible}}^{\max}(t)\right)$$

pour des s admissibles. Quoiqu'il en soit, il s'agit à nouveau de deux questions hors du propos de cette thèse.

5.3.5 Le choix des paramètres

Nous montrons maintenant comment choisir les paramètres de Wave. Commençons par rappeler que la stratégie décrite dans §5.2 (où l'on remplace k_U et k_V par $k'_U \triangleq k_U - d$ et $k'_V \triangleq k_V - d$) pour obtenir des erreurs de grand poids permet de produire en moyenne des solutions de poids :

$$\frac{2}{3}(n + k_U - d).$$

De cette façon pour que notre algorithme fonctionne en temps moyen polynomial nous devons nécessairement choisir w tel que

$$w = \frac{2}{3}(n + k_U - d) \tag{5.73}$$

On peut alors vérifier en combinant les équations (5.70) et (5.73) que pour les paramètres $\alpha \in [0, 1]$, d et la dimension du code

$$k \triangleq k_U + k_V$$

fixé nous avons :

$$w = \left\lfloor n\left(1 - \alpha + \frac{1}{3}\sqrt{(3\alpha - 1)\left(3\alpha + 4\frac{k - 2d}{n} - 1\right)}\right)\right\rfloor \tag{5.74}$$

$$k_V = d + \left\lfloor \frac{n}{2}\frac{3}{3\alpha - 1}\left(\left(1 - \frac{w}{n}\right)^2 + \frac{1}{2}\left(\frac{w}{n}\right)^2 - \frac{1}{3}\right)\right\rfloor \tag{5.75}$$

$$k_U = d + \left\lfloor \frac{n}{2}\left(-2 + 3\frac{w}{n}\right)\right\rfloor \tag{5.76}$$

Nous choisissons ensuite le paramètre d tel que :

$$3^d = 2^\lambda \iff d = \frac{\lambda}{\ln 3}. \tag{5.77}$$

Cela permet avec le théorème 5.2 d'affirmer que pour presque toute clef publique, la distance statistique entre les signatures produites et l'uniforme sur S_w est inférieur à $O\left(n^2 2^{-\lambda}\right)$.

De plus, nous devons choisir les distributions $\mathcal{D}_V$ et $\mathcal{D}_U^t$. Nous proposons alors :

$$\begin{cases} \mathcal{D}_V = & \text{Lap}(\mu_V, \sigma_V, 0, k'_V) \\ \mathcal{D}_U^t = & \text{Lap}(\mu_U(t), \sigma_U(t), t + k_U - d - n/2, t) \end{cases}$$

avec

$$\begin{cases} \mu_V = & (1 - \alpha)(k_V - d) \\ \mu_U(t) = & t - \frac{3}{2}m_{\text{cible}}^{\max}(t) + \varepsilon \end{cases}$$

où σ_V et σ_U sont à déterminer selon le jeu de paramètre choisi et $\varepsilon \geqslant 0$ une petite quantité permettant de minimiser encore plus le nombre de rejets. Nous avons alors constaté heuristiquement que typiquement $\varepsilon = 1$ ou 2 était un bon choix.

Instanciation proposée des paramètres. Nous proposons pour 128 bits de sécurité les paramètres suivants :

$$n = 8312, \quad k_U + k_V = 5062 \quad \text{où} \quad (k_U, k_V) = (3316, 1746)$$

$$w = 7698, \quad \alpha = 0.5854, \quad d = 81, \quad \sigma_V = 16.7, \quad \forall t,\ \sigma_U = 6.7 \text{ et } \varepsilon = 1.0.$$

Pour ce jeu de paramètres nous obtenons $M_V^{\text{rs}} \approx 1.0407$ et $M_U^{\text{rs}} \approx 1.0385$ en moyenne sur t et donc une probabilité de faire un rejet dans l'algorithme de l'ordre de 0.08.

Nous verrons dans le prochain chapitre que la sécurité de Wave se réduit au problème DOOM$(n, 3, R, \omega)$ où $R \triangleq \frac{k_U+k_V}{n}$, $\omega \triangleq \frac{w}{n}$ ainsi qu'au problème de distinguer un code aléatoire d'un code $(U, U+V)$-généralisé permuté, problème dont nous discuterons dans §7. Nous avons alors sélectionné les paramètres de Wave de façon à ce qu'ils respectent les équations (5.74) et que les meilleurs algorithmes résolvant les deux problèmes soient de complexité de l'ordre de 2^{128}. Plus précisément, nous avons procédé de la façon suivante. Tout d'abord nous avons,

— la complexité pour résoudre DOOM(n, q, R, ω) est de la forme $2^{c_M n(1+o(1))}$ avec l'étude que nous avons faite du décodage en grande distance de la partie II de ce document,

— la complexité du distingueur que nous présenterons dans §7.2 est de la forme $2^{c_K n(1+o(1))}$ où c_K est une fonction de k_U/n et k_V/n étudié dans §7.2.2.3, §7.2.2.4 et §7.2.2.5.

Avec les équations (5.74), (5.75) et (5.76) nous en déduisons que c_M et c_K sont tout deux fonctions fonction du rendement du code $R = k/n$ ainsi que du paramètre α. Nous avons alors cherché à minimiser la taille de clef publique $K \triangleq \log_2(3) n^2 R(1-R)$ (mise sous forme systématique) en bits avec la contrainte $c_M(R, \alpha) = c_K(R, \alpha)$. Nous avons obtenu,

$$R = 0.6089, \quad \alpha = 0.5854 \quad \text{et} \quad c_M \approx c_K \approx 0.0154.$$

De cette façon, pour λ bits de sécurité classique nous avons :

$$n = \frac{\lambda}{0.0154}, \quad w = 0.9261\, n, \quad k_U = 0.7978\, \frac{n}{2}, \quad k_V = 0.4201\, \frac{n}{2} \quad \text{et} \quad K = 0.3774\, n^2$$

5.3.6 Et les erreurs de petit poids ?

Il est maintenant légitime de se demander pourquoi nous avons choisi d'instancier Wave avec un décodage à grande distance. La raison que nous allons voir ici est la suivante. Au delà du fait que la trappe des codes $(U, U+V)$-généralisés permet de décoder à petite distance moins efficacement qu'en grandes distances dans $\mathbb{F}_3$, nous pouvons au mieux avec la stratégie de poids faible produire des erreurs uniformément distribuées parmi les mots de poids w^-_{easy} là où le problème est génériquement facile.

Nous pourrions reprendre tout le raisonnement de cette section pour démontrer ce résultat mais cela serait long et fastidieux. Nous pouvons être plus rapide en "approximant" le modèle d'erreur. Au lieu de chercher à produire des erreurs uniformément distribuées parmi les mots de poids w, supposons que la distributions des erreurs $\mathbf{e} \in \mathbb{F}_3^n$ soit telle que les symboles $\mathbf{e}(i)$ sont indépendants et équi-distribués comme

$$\mathbb{P}(\mathbf{e}(i) = 1) = \mathbb{P}(\mathbf{e}(i) = -1) = \omega/2 \quad \text{où} \quad \omega \triangleq \frac{w}{n}.$$

Ce modèle nous donnera les valeurs moyennes des distributions de poids considérées dans notre algorithme. Afin de simplifier les calculs qui vont suivre supposons, sans perte de généralité, que l'on cherche à décoder un code $(U, U+V)$. L'erreur que nous obtiendrons est donc de la forme $\mathbf{e} = (\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$ où :

$$\mathbb{P}(\mathbf{e}_U(i) = 1) = \mathbb{P}(\mathbf{e}_U(i) = -1) = \omega/2 \tag{5.78}$$

$$\mathbb{P}(\mathbf{e}_U(i) + \mathbf{e}_V(i) = 1) = \mathbb{P}(\mathbf{e}_U(i) + \mathbf{e}_V(i) = -1) = \omega/2 \tag{5.79}$$

De cette façon nous allons pouvoir prédire avec notre algorithme les paramètres que nous devons choisir pour espérer être en mesure de produire des erreurs uniformément distribuées sur S_w.

Notre algorithme de décodage consiste tout d'abord à décoder avec l'algorithme de Prange le $[n/2, k_V]_3$-code V pour obtenir $\mathbf{e}_V$. Pour cela nous mettons k_V zéro dans un ensemble d'information. Nous obtenons donc une erreur de poids moyen $\frac{2}{3}(n/2 - k_V)$. Notons alors $R_V \triangleq k_V/(n/2)$. Le poids relatif moyen est donc :

$$\frac{2}{3}(1 - R_V) \tag{5.80}$$

Déterminons maintenant la distribution de $|\mathbf{e}_V|$. Remarquons tout d'abord que,

$$\mathbf{e}_V = (\mathbf{e}_U + \mathbf{e}_V) - \mathbf{e}_U.$$

Nous en déduisons alors avec (5.78) et (5.79) que

$$\mathbb{P}(\mathbf{e}_V(i) = 1) = \mathbb{P}(\mathbf{e}_V(i) = -1) = \omega(1-\omega) + \frac{\omega^2}{4}. \tag{5.81}$$

Il est donc nécessaire d'après les équations (5.80) et (5.81) de choisir R_V comme :

$$\frac{2}{3}(1 - R_V) = 2\omega(\omega - 1) + \frac{\omega^2}{2} \iff R_V = 1 - \frac{3}{2}\left(2\omega(1-\omega) + \frac{\omega^2}{2}\right)$$

La deuxième étape de notre algorithme consiste à décoder avec l'algorithme de Prange le $[n/2, k_U]_3$-code U pour obtenir $\mathbf{e}_U$ mais en tenant compte de la valeur des $\mathbf{e}_V(i)$. Plus précisément, pour les k_U positions i d'un ensemble d'information,

— Si $\mathbf{e}_V(i) = 0$, nous choisissons $\mathbf{e}_U(i) = 0$,
— Si $\mathbf{e}_V(i) = b \in \{-1, 1\}$, nous choisissons $\mathbf{e}_U(i) = 0$ ou $-b$.

Ce choix est fait dans le but de minimiser le poids de l'erreur $(\mathbf{e}_U, \mathbf{e}_U + \mathbf{e}_V)$ obtenue à la fin du décodage. Notons alors dans ce qui suit k_{U_0} le nombre de symboles d'information choisis dans $\overline{\mathrm{Supp}}(\mathbf{e}_V) = \{i : \mathbf{e}_V(i) = 0\}$ et

$$R_{U_0} \triangleq \frac{k_{U_0}}{\#\overline{\mathrm{Supp}}(\mathbf{e}_V)}. \tag{5.82}$$

De cette façon le poids relatif moyen de $\mathbf{e}_U$ restreint à $\overline{\mathrm{Supp}}(\mathbf{e}_V)$ est :

$$\frac{2}{3}(1 - R_{U_0}). \tag{5.83}$$

Calculons donc les probabilités $\mathbb{P}(\mathbf{e}_U = b \mid \mathbf{e}_V(i) = 0)$ pour $b \in \{-1, 1\}$. Nous avons d'après (5.78), (5.79) et (5.81),

$$\mathbb{P}(\mathbf{e}_U(i) = 1 \mid \mathbf{e}_V(i) = 0) = \mathbb{P}(\mathbf{e}_U(i) = -1 \mid \mathbf{e}_V(i) = 0) = \frac{\omega^2/4}{1 - 2\omega(1-\omega) - \omega^2/2}.$$

Donc d'après l'équation (5.83), nous devons choisir R_{U_0} pour qu'en moyenne l'algorithme de Prange renvoie les bon poids tel que :

$$\frac{2}{3}(1-R_{U_0}) = 2\frac{\omega^2/4}{1-2\omega(1-\omega)-\omega^2/2} \iff R_{U_0} = 1-\frac{3}{2}\left(\frac{\omega^2/2}{1-2\omega(1-\omega)-\omega^2/2}\right) \tag{5.84}$$

Il nous reste maintenant à choisir $k_U - k_{U_0}$ symboles dans $\mathrm{Supp}(\mathbf{e}_V)$ tels que $\mathbf{e}_U(i) = 0$ ou $-\mathbf{e}_V(i)$. Notons :

$$R_{U_1} \triangleq \frac{k_U - k_{U_0}}{\#\,\mathrm{Supp}(\mathbf{e}_V)} \tag{5.85}$$

L'algorithme de Prange renverra $\mathbf{e}_U$ telle qu'en moyenne restreint à $\mathrm{Supp}(\mathbf{e}_V)$ il contient

$$\frac{1}{3}(1-R_{U_1}) \tag{5.86}$$

symboles égaux à $\mathbf{e}_V(i)$. Calculons $\mathbb{P}\left(\mathbf{e}_U(i) = b \mid \mathbf{e}_V(i) = b\right)$ pour $b \in \{-1,1\}$. Nous avons pour $b \in \{-1,1\}$,

$$\mathbb{P}\left(\mathbf{e}_U(i) = b \mid \mathbf{e}_V(i) = b\right) = \frac{w^2/4}{\omega(1-\omega)+\frac{\omega^2}{4}}.$$

Donc d'après l'équation (5.86), nous devons choisir R_{U_1} tel que :

$$\frac{1}{3}\left(1-R_{U_1}\right) = 2\frac{w^2/4}{\omega(1-\omega)+\frac{\omega^2}{4}} \iff R_{U_1} = 1-3\frac{w^2}{2\omega(1-\omega)+2\frac{\omega^2}{4}} \tag{5.87}$$

Nous pouvons alors en déduire d'après (5.82) et (5.85) que $R_U \triangleq \frac{k_U}{n/2}$ est égal à :

$$R_U = \frac{\#\overline{\mathrm{Supp}(\mathbf{e}_V)}}{n/2}R_{U_0} + \frac{\mathrm{Supp}(\mathbf{e}_V)}{n/2}R_{U_1}.$$

Nous faisons ici un raisonnement en moyenne, écrivons donc d'après (5.81) :

$$\frac{\#\overline{\mathrm{Supp}(\mathbf{e}_V)}}{n/2} = 1-2\omega(1-\omega)-\frac{\omega^2}{2} \quad \text{et} \quad \frac{\mathrm{Supp}(\mathbf{e}_V)}{n/2} = 2\omega(1-\omega)+\frac{\omega}{2}.$$

Nous en déduisons alors d'après les équations (5.84) et (5.87) le calcul suivant :

$$\begin{aligned} R_U &= 1-2\omega(1-\omega)-\frac{\omega^2}{2}-\frac{3}{2}\omega^2/2+2\omega(1-\omega)+\frac{\omega^2}{2}-3\omega^2 \\ &= 1-\frac{9}{2}\omega^2 \end{aligned}$$

Maintenant la dimension k du code vérifie $k = k_U + k_V$, d'où :

$$\begin{aligned} k &= \frac{n}{2}\left(R_U + R_V\right) \\ &= \frac{n}{2}\left(1-\frac{9}{2}\omega^2+1-\frac{3}{2}\left(2\omega(1-\omega)+\frac{\omega^2}{2}\right)\right) \\ &= \frac{n}{2}\left(2-\frac{9}{2}\omega^2-\frac{3}{2}\left(2\omega-\frac{3}{2}\omega^2\right)\right) \\ &= \frac{n}{2}\left(2-3\omega\right) \\ &= n-\frac{3}{2}w \quad (\omega = w/n). \end{aligned}$$

ce qui donne

$$w = \frac{2}{3}\left(n-k\right)$$

qui est bien le poids w^-_{easy}.

5.4 Des fonctions bien distribuées avec les codes $(U, U+V)$-généralisés

Dans toute cette section $\mathbf{H}_{\text{pk}}$ dénotera une clef publique $\mathbf{S}\mathcal{H}(\varphi, \mathbf{H}_U, \mathbf{H}_V)\mathbf{P}$ de la famille Wave comme décrit dans §5.1.2.

La structure aléatoire (sur $\mathbf{H}_U$ et $\mathbf{H}_V$) de $\mathbf{H}_{\text{pk}}$ nous permet de démontrer que les syndromes $\mathbf{e}\mathbf{H}_{\text{pk}}^{\intercal}$ sont statistiquement indistinguables de l'uniforme comme montré dans la proposition qui suit. De cette façon, la famille Wave est bien distribuée dans le sens du point 1 de la définition 5.1.

La définition qui suit nous sera utile.

Définition 5.7. (nombre de blocs V de type I). *Nous définissons le nombre de blocs V de type I pour un code $(U, U+V)$-généralisé de longueur n associé au quadruplet $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ comme :*

$$n_I \triangleq |\{1 \leqslant i \leqslant n/2 : b_i d_i = 0\}| .$$

Proposition 5.10. *Notons $\mathcal{D}_w^{\mathbf{H}}$ la distribution de $\mathbf{e}\mathbf{H}^{\intercal}$ quand $\mathbf{e} \hookleftarrow S_w$. De plus, considérons $\mathcal{U}$ la distribution uniforme sur $\mathbb{F}_3^{n-k}$. Nous avons :*

$$\mathbb{E}_{\mathbf{H}_{\text{pk}}}\left(\rho(\mathcal{D}_w^{\mathbf{H}_{\text{pk}}}, \mathcal{U})\right) \leqslant \frac{1}{2}\sqrt{\varepsilon}$$

où

$$\varepsilon = \frac{3^{n-k}}{2^w \binom{n}{w}} + \sum_{j=0}^{\frac{n}{2}} \frac{3^{\frac{n}{2}-k_V}\binom{\frac{n}{2}}{j}\left(\sum\limits_{p=0:p\equiv w \pmod 2}^{j} \binom{j}{p}\binom{\frac{n}{2}-j}{\frac{w+p}{2}-j} 2^{\frac{3p}{2}}\right)^2}{2^{w+j}\binom{n}{w}^2} + 3^{\frac{n}{2}-k_U}\left(\sum_{j=0}^{n_I} \frac{\binom{n_I}{j}\binom{n-n_I}{w-j}^2}{\binom{n}{w}^2 2^j}\right).$$

Cette borne décroît exponentiellement avec n pour certains régimes de paramètres comme montré par la proposition suivante.

Proposition 5.11. *Considérons les paramètres relatifs*

$$R_U \triangleq \frac{2k_U}{n}, \quad R_V \triangleq \frac{2k_V}{n}, \quad R \triangleq \frac{k}{n}, \quad \omega \triangleq \frac{w}{n} \quad \textit{et} \quad \nu \triangleq \frac{n_I}{n}$$

Alors avec les mêmes notations que la proposition 5.10, nous avons pour n tendant vers l'infini :

$$\mathbb{E}_{\mathbf{H}_{\text{pk}}}\left(\rho(\mathcal{D}_w^{\mathbf{H}_{\text{pk}}}, \mathcal{U})\right) \leqslant 2^{(\alpha+o(1))n}$$

où $\alpha \triangleq \frac{1}{2}\min\left((1-R)\log_2(3) - \omega - h_2(\omega), \alpha_1, \alpha_2\right)$ *et*

$$\alpha_1 \triangleq \min_{(x,y)\in\mathcal{R}} \frac{1}{2}(1-R_V)\log_2 3 - \omega - 2h_2(\omega) + \frac{h_2(x)}{2} + x\left(h_2(y) + \frac{3}{2}y - \frac{1}{2}\right) + (1-x)h_2\left(\frac{\omega - x(1-y)}{1-x}\right)$$

$$\mathcal{R} \triangleq \{(x,y) \in [0,1[\times[0,1] : 0 \leqslant \omega - x(1-y) \leqslant 1-x\}$$

$$\alpha_2 \triangleq \min_{\max(0,\omega+\nu-1)\leqslant x\leqslant \min(\nu,\omega)} \frac{1}{2}(1-R_U)\log_2 3 - 2h_2(\omega) + \nu h_2\left(\frac{x}{\nu}\right) + 2(1-\nu)h_2\left(\frac{\omega-x}{1-\nu}\right) - x.$$

Remarque 5.6. Pour les paramètres que nous avons donnés dans §5.3.5, nous avons $\varepsilon \approx 2^{-354}$ et $\alpha \approx -0.02135$. Pour obtenir ce résultat nous avons choisi $n_I = n/6$. Cela provient du fait le code $(U, U+V)$-généralisé est tiré aléatoirement et donc que les vecteurs $\mathbf{a}, \mathbf{b}, \mathbf{c}$ et $\mathbf{d}$ de $\mathbb{F}_3^{n/2}$ vérifiant $a_i c_i \neq 0$ et $a_i d_i - b_i c_i \neq 0$ donnent typiquement $n_I = n/6$.

Notons que la borne de la proposition 5.10 n'est en aucun cas fine. Cela provient du terme $3^{\frac{n}{2}-k_U}\left(\sum_{j=0}^{n_I} \frac{\binom{n_I}{j}\binom{n-n_I}{w-j}^2}{\binom{n}{w}^2 2^j}\right)$ qui est une borne extrêmement grossière comme nous le verrons à la fin de la démonstration de la proposition 5.10. Nous avons fait ce choix pour éviter de donner une formule encore plus complexe. Il est cependant facile d'améliorer le terme ε.

La preuve de la proposition 5.10 est une conséquence directe de la combinaison du lemme qui suit et du lemme 1.4 étant lui-même une variation du *left-over hash lemma*. Cette variation est ici nécessaire et essentielle car les matrices $\mathbf{H}_{\text{pk}}$ ne sont pas uniformément distribuées sur $\mathbb{F}_3^{(n-k)\times n}$.

Lemme 5.12. *Considérons $\mathbf{x}, \mathbf{y} \hookleftarrow S_w$. Nous avons :*

$$\mathbb{P}_{\mathbf{H}_{\text{pk}},\mathbf{x},\mathbf{y}}\left(\mathbf{x}\mathbf{H}_{\text{pk}}^\intercal = \mathbf{y}\mathbf{H}_{\text{pk}}^\intercal\right) \leqslant \frac{1}{3^{n-k}}(1+\varepsilon) \text{ avec } \varepsilon \text{ donnée dans la proposition 5.10.}$$

Démonstration du lemme 5.12.

La probabilité que nous cherchons à calculer est donnée d'après la notation 13 et la proposition 5.4 par :

$$\mathbb{P}\left((\mathbf{x}_U - \mathbf{y}_U)\mathbf{H}_U^\intercal = \mathbf{0} \quad \text{et} \quad (\mathbf{x}_V - \mathbf{y}_V)\mathbf{H}_V^\intercal = \mathbf{0}\right)$$

où cette dernière est calculée par rapport à $\mathbf{H}_U, \mathbf{H}_V, \mathbf{x}, \mathbf{y}$. Nous allons ici utiliser le lemme 1.3 montrant que :

$$\mathbb{P}_{\mathbf{H}}\left(\mathbf{y}\mathbf{H}^\intercal = \mathbf{s}\right) = \frac{1}{3^r} \tag{5.88}$$

où $\mathbf{H} \in \mathbb{F}_3^{r\times n}$ est uniformément distribuée, $\mathbf{y} \neq 0$ et $\mathbf{s} \in \mathbb{F}_3^r$ quelconques. Cela nous motive à introduire les quatre évènements disjoints :

— **Évènement** 1 **:**
$$\mathcal{E}_1 \triangleq \{\mathbf{x}_U = \mathbf{y}_U \quad \text{et} \quad \mathbf{x}_V \neq \mathbf{y}_V\},$$

— **Évènement** 2 **:**
$$\mathcal{E}_2 \triangleq \{\mathbf{x}_U \neq \mathbf{y}_U \quad \text{et} \quad \mathbf{x}_V = \mathbf{y}_V\},$$

— **Évènement** 3 **:**
$$\mathcal{E}_3 \triangleq \{\mathbf{x}_U \neq \mathbf{y}_U \quad \text{et} \quad \mathbf{x}_V \neq \mathbf{y}_V\},$$

— **Évènement** 4 **:**
$$\mathcal{E}_4 \triangleq \{\mathbf{x}_U = \mathbf{y}_U \quad \text{et} \quad \mathbf{x}_V = \mathbf{y}_V\}.$$

Selon ces évènements nous obtenons d'après (5.88) et $k = k_U + k_V$:

$$\begin{aligned}
&\mathbb{P}_{\mathbf{H}_{\text{sk}},\mathbf{x},\mathbf{y}}\left(\mathbf{x}\mathbf{H}_{\text{sk}}^\intercal = \mathbf{y}\mathbf{H}_{\text{sk}}^\intercal\right) \\
&= \sum_{i=1}^{4} \mathbb{P}_{\mathbf{H}_{\text{sk}}}\left(\mathbf{x}\mathbf{H}_{\text{sk}}^\intercal = \mathbf{y}\mathbf{H}_{\text{sk}}^\intercal | \mathcal{E}_i\right)\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_i\right) \\
&= \frac{\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_1\right)}{3^{n/2-k_V}} + \frac{\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_2\right)}{3^{n/2-k_U}} + \frac{\mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_3\right)}{3^{n-k}} + \mathbb{P}_{\mathbf{x},\mathbf{y}}\left(\mathcal{E}_4\right) \\
&\leqslant \frac{1}{3^{n-k}}\left(1 + 3^{n/2-k_U}\mathbb{P}\left(\mathcal{E}_1\right) + 3^{n/2-k_V}\mathbb{P}\left(\mathcal{E}_2\right) + 3^{n-k}\mathbb{P}(\mathcal{E}_4)\right)
\end{aligned} \tag{5.89}$$

où nous avons utilisé pour la dernière inégalité la borne triviale $\mathbb{P}(\mathcal{E}_3) \leqslant 1$. Calculons désormais une borne supérieure pour les probabilités des évènements $\mathcal{E}_1$, $\mathcal{E}_2$ et $\mathcal{E}_4$. Commençons par l'évènement $\mathcal{E}_4$. Rappelons que par définition des codes $(U, U+V)$-généralisés, nous avons :

$$\mathbb{P}_{\mathbf{x},\mathbf{y}}(\mathcal{E}_4) = \mathbb{P}(\mathbf{x} = \mathbf{y}) = \frac{1}{2^w \binom{n}{w}} \tag{5.90}$$

Estimons maintenant la probabilité de l'évènement $\mathcal{E}_2$ et pour cela commençons par remarquer que :

$$\mathbb{P}(\mathcal{E}_2) \leqslant \mathbb{P}(\mathbf{x}_V = \mathbf{y}_V).$$

Nous observons alors que pour toute erreur $\mathbf{e} \in \mathbb{F}_3^{n/2}$ de poids j :

$$\begin{aligned}\mathbb{P}(\mathbf{x}_V = \mathbf{e}) &= \mathbb{P}(\mathbf{x}_V = \mathbf{e} \mid |\mathbf{x}_V| = j)\,\mathbb{P}(|\mathbf{x}_V| = j) \\ &= \frac{1}{2^j \binom{n/2}{j}} q_1(j)\end{aligned}$$

où $q_1^{\text{unif}}(j)$ est défini comme $\mathbb{P}(|\mathbf{e}_V^{\text{unif}}| = j)$ et est donné dans la proposition 5.6. De cela nous en déduisons que :

$$\begin{aligned}\mathbb{P}(\mathbf{x}_V = \mathbf{y}_V) &= \sum_{j=0}^{n/2} \sum_{\mathbf{e} \in \mathbb{F}_3^{n/2} : |\mathbf{e}| = j} \mathbb{P}_{\mathbf{x}}(\mathbf{x}_V = \mathbf{e})^2 \\ &= \sum_{j=0}^{n/2} \frac{1}{2^j \binom{n/2}{j}} q_1^{\text{unif}}(j)^2\end{aligned}$$

ce qui donne :

$$\mathbb{P}(\mathcal{E}_2) \leqslant \sum_{j=0}^{n/2} \frac{q_1^{\text{unif}}(j)^2}{2^j \binom{n/2}{j}} \tag{5.91}$$

Attelons-nous maintenant à la probabilité de $\mathcal{E}_1$. Remarquons tout d'abord que :

$$\mathbb{P}(\mathcal{E}_1) \leqslant \mathbb{P}(\mathbf{x}_U = \mathbf{y}_U) \tag{5.92}$$

Par définition de $\mathbf{x}_U$ et $\mathbf{y}_U$, l'évènement $\mathbf{x}_U = \mathbf{y}_U$ est donné par :

$$\{\mathbf{d} \odot (\mathbf{x}_1 - \mathbf{y}_1) = \mathbf{b} \odot (\mathbf{x}_2 - \mathbf{y}_2)\}$$

qui est identique (à une permutation près des indices de $\mathbf{x}$ et $\mathbf{y}$ ainsi que la multiplication de certaines de leurs coordonnées par -1) au cas où :

$$b_1 = \cdots = b_{n_I} = 0 \quad \text{et} \qquad b_{n_I+1} = \cdots = b_{n/2} = d_1 = \cdots = d_{n/2} = 1$$

avec n_I le nombre de blocs de type I (voir la définition 5.7). Nous obtenons alors comme probabilité à majorer :

$$\mathbb{P}\left(\forall i \in [\![1, n_I]\!],\ (\mathbf{x}_1 - \mathbf{y}_1)(i) = 0, \forall i \in [\![n_I + 1, n/2]\!], (\mathbf{x}_1 - \mathbf{y}_1)(i) = (\mathbf{x}_2 - \mathbf{y}_2)(i)\right).$$

Nous avons alors clairement :

$$\begin{aligned}&\mathbb{P}(\forall i \in [\![1, n_I]\!],\ (\mathbf{x}_1 - \mathbf{y}_1)(i) = 0, \forall i \geqslant n_I + 1,\ (\mathbf{x}_1 - \mathbf{y}_1)(i) = (\mathbf{x}_2 - \mathbf{y}_2)(i)) \\ &\quad\leqslant \sum_{\mathbf{e} \in \mathbb{F}_3^{n_I}} \mathbb{P}(\forall i \in [\![1, n_I]\!], \mathbf{x}_1(i) = \mathbf{e}(i))^2 \qquad (5.93) \\ &\quad\leqslant \sum_{j=0}^{n_I} \sum_{\mathbf{e}' \in \mathbb{F}_3^{n_I} : |\mathbf{e}'| = j} \mathbb{P}\left(\forall i \in [\![1, n_I]\!], \mathbf{x}_1(i) = \mathbf{e}'(i)\right)^2 \\ &\quad= \sum_{j=0}^{n_I} \sum_{\mathbf{e}' \in \mathbb{F}_3^{n_I} : |\mathbf{e}'| = j} \left(\frac{\binom{n-n_I}{w-j} 2^{w-j}}{\binom{n}{w} 2^w}\right)^2 \\ &\quad= \sum_{j=0}^{n_I} \binom{n_I}{j} 2^j \left(\frac{\binom{n-n_I}{w-j}}{\binom{n}{w} 2^j}\right)^2\end{aligned}$$

ce qui donne avec (5.92) :

$$\mathbb{P}(\mathcal{E}_1) \leqslant \sum_{j=0}^{n_I} \binom{n_I}{j} 2^{-j} \left(\frac{\binom{n-n_I}{w-j}}{\binom{n}{w}} \right)^2$$

et donc conclut la preuve du lemme. Remarquons que notre borne est très large. Ceci est dû à la majoration (5.93).

□

Chapitre 6

Une réduction de sécurité des fonctions GPVM

Introduction

Le schéma de signature que permirent d'instancier dans le paradigme des réseaux euclidens les fonctions GPV $f_{\mathbf{A}}(\mathbf{e}) \stackrel{\triangle}{=} \mathbf{e}\mathbf{A}^\intercal$ pour $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ (où $m \geqslant n$) et $\mathbf{e} \in \mathbb{Z}_q^m$ vint avec une réduction de sécurité fine [GPV08b, §6.1 et §6.2] au problème de collision :

$$\text{trouver } \mathbf{e}, \mathbf{e}' \in \mathbb{Z}_q^m \quad \text{tels que} \quad |\mathbf{e}|_2, |\mathbf{e}'|_2 \leqslant w \quad \text{et} \quad \mathbf{e}\mathbf{A}^\intercal = \mathbf{e}'\mathbf{A}^\intercal.$$

avec $|\cdot|_2$ la norme euclidienne. Ce problème se réduit alors essentiellement au problème SIS (inverser $f_{\mathbf{A}}$) à distance $\sqrt{2}w$. Il est alors argüé que pour les paramètres proposés en pratique comme dans la soumission au NIST Falcon [Fou+17], les meilleures attaques contre ce problème sont essentiellement du même ordre de grandeur que celles contre SIS à distance w. La sécurité de la signature déduite des fonctions GPV en réseaux euclidiens est donc aujourd'hui réduite au problème SIS à distance w.

Nous pourrions donc être tenté d'utiliser cette réduction pour ramener la sécurité de Wave au problème du décodage générique SD. Malheureusement cela n'est pas possible pour trois raisons : (i) le problème de collision est comme nous l'avons vu dans §5.1.1 facile pour les paramètres que nous proposons, (ii) nous utilisons comme trappe la famille des codes $(U, U+V)$-généralisés permutés qui ne forme pas l'ensemble des codes et (iii) à la différence avec les fonctions GPV l'inversion de $f_{\mathbf{H}}(\mathbf{e}) \stackrel{\triangle}{=} \mathbf{e}\mathbf{H}^\intercal$ pour $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ et $\mathbf{e} \in S_{w,n}$ est prouvé indistinguable de la distribution uniforme sur S_w pour $\mathbf{s} \hookleftarrow \mathbb{F}_q^{n-k}$, c'est à dire en moyenne sur les entrées (ce qui l'implique dans notre cas pour presque toute entrée) d'où l'introduction du concept de fonction GPV en Moyenne (GPVM). Nous proposons alors dans ce chapitre une réduction fine palliant ces trois difficultés dans le modèle de sécurité EUF-CMA où tout adversaire a accès à des signatures de son choix comme nous le décrirons. Pour cela nous commençons par réduire la sécurité de notre schéma au problème DOOM à distance w, réduction d'ailleurs naturelle comme nous le verrons. Or ce problème est dans l'état de l'art algorithmique aussi difficile que le problème SD à distance w (voir le chapitre §3). Donc similairement aux réseaux euclidiens, nous nous réduisons au problème générique à distance w considéré en cryptographie avec des codes. Concernant le point (ii) nous nous réduirons de plus au problème de distinguer un code $(U, U+V)$-généralisé permuté d'un code aléatoire de façon similaire à la preuve de sécurité du chiffrement de McEliece où le problème de distinguer le code utilisé comme trappe d'un code quelconque apparaît naturellement (voir la proposition 1.11). Nous discuterons alors de la difficulté de ce problème dans le prochain chapitre dont nous prouverons d'ailleurs

qu'il est NP-complet. La complexité des meilleurs algorithmes résolvant ce problème semble aujourd'hui exponentielle en la longueur du code considéré. Le dernier point est quant à lui plus délicat. Il est clair que lorsque l'indistinguabilité est assurée pour *toute entrée*, l'inversion ne peut rien apprendre à tout adversaire, aussi puissant soit-il. Il s'avère comme nous le verrons que notre propriété en moyenne l'implique aussi. En revanche, il est important dans notre cas (ce qui est spécifié dans notre schéma) que lors de la signature d'un message $\mathbf{m}$ nous le hachions avec un sel $\mathbf{r}$, tiré aléatoirement dans l'algorithme, comme $\texttt{Hache}(\mathbf{m}, \mathbf{r})$. En effet dans le modèle de l'oracle aléatoire où $\texttt{Hache}(\cdot)$ est supposée être une fonction aléatoire, nous allons prouver que lorsque $\mathbf{r}$ est formé de suffisamment de bits, plus précisément un $O(\lambda)$, cela est suffisant.

6.1 Le modèle de sécurité et le paradigme des jeux

Nous reprenons en partie dans cette section la description [Sho04] des réductions de sécurité. Notre discussion sera informelle et nous renvoyons aux travaux de [Poi19] pour plus de détails.

L'objectif d'une réduction de sécurité d'un schéma cryptographique est de ramener la sécurité à un problème bien identifié, disons P. Autrement dit, la réduction doit montrer comment tout adversaire attaquant le schéma peut être utilisé pour casser le problème P. Dans ce contexte, la sécurité des schémas cryptographiques est souvent définie en terme de "jeux" entre un adversaire et un rival (*challenger*) lui posant un défi. Le niveau de sécurité est alors essentiellement mesuré comme le temps d'exécution de l'adversaire divisé par sa probabilité de répondre au défi posé, que l'on appellera probabilité de succès de l'adversaire dans le jeu G et que l'on notera $\mathbb{P}(G)$. Les deux entités en question, l'attaquant et le rival, sont des algorithmes probabilistes communiquant entre eux. De cette façon, un jeu peut être modélisé comme un espace probabiliste sur lequel sont définis les variables aléatoires des algorithmes. Considérons alors le jeu G_0 définissant la sécurité d'un schéma et un adversaire fixé $\mathcal{A}$. L'idée dans ce cadre pour faire une réduction de sécurité à un problème P est alors *pas à pas* de changer les distributions du rival dans le jeu G_0 (on dit qu'on simule le rival et on parle de simulateur) pour obtenir une suite de jeux $G_1, \cdots, G_N$ telle que la probabilité de succès de $\mathcal{A}$ dans le dernier jeu $\mathbb{P}(G_N)$ est (i) sa probabilité de casser le problème P et (ii) :

$$\forall i \in [\![0, N-1]\!], \quad |\mathbb{P}(G_i) - \mathbb{P}(G_{i+1})| \in \text{negl}(\lambda) \Rightarrow |\mathbb{P}(G_0) - \mathbb{P}(G_N)| \in \text{negl}(\lambda)$$

avec λ est le niveau de sécurité. Autrement dit, à chaque jeu nous avons fait des modifications qui n'ont pas changé (à un facteur négligeable près) la probabilité de succès de l'adversaire. Cependant il faut ici être précautionneux. Le problème porte principalement sur la signification du changement des distributions entre les jeux. Nous n'avons naturellement pas accès aux distributions internes de l'adversaire $\mathcal{A}$ ce-dernier étant quelconque. En revanche, nous pouvons simuler son rival lui donnant un défi. Nous pouvons alors utiliser trois outils à partir d'un jeu G_i pour définir le jeu G_{i+1},

1. Définir G_{i+1} comme étant identique à G_i à moins qu'un évènement F se produise où dans ce cas on dit que l'adversaire a échoué à résoudre le défi. On a alors (voir [Sho04, Lemme 1]),
$$|\mathbb{P}(G_i) - \mathbb{P}(G_{i+1})| \leqslant \mathbb{P}(F).$$
2. Définir G_{i+1} comme le jeu G_i où l'on remplace une distribution $\mathcal{D}$ utilisée par le rival par une distribution $\mathcal{D}'$. Nous avons,
$$|\mathbb{P}(G_i) - \mathbb{P}(G_{i+1})| \leqslant \rho\left(\mathcal{D}, \mathcal{D}'\right)$$
où $\rho(\cdot, \cdot)$ est la distance statistique.

3. Définir de même le jeu G_{i+1} comme le jeu G_i où l'on remplace une distribution $\mathcal{D}$ utilisée par le rival par une distribution $\mathcal{D}'$. Dans le cas où le simulateur peut fonctionner sans savoir si la distribution $\mathcal{D}$ ou $\mathcal{D}'$ est utilisée, nous avons

$$|\mathbb{P}(G_i) - \mathbb{P}(G_{i+1})| \leqslant \rho_c\left(\mathcal{D}, \mathcal{D}'\right)(t).$$

où $\rho_c(\cdot,\cdot)$ est la distance calculatoire et t le temps d'exécution du jeu.

Il faut cependant faire attention aux points 2 et 3. Le simulateur du rival doit toujours être en mesure de faire tourner le jeu après changement des distributions, c'est à dire de répondre aux requêtes de l'attaquant et de lui fournir son défi.

La preuve du point 2 repose essentiellement sur [GM02, Proposition 8.10]] que nous avons déjà utilisée dans le chapitre précédent où f joue ici le rôle de l'adversaire échangeant avec le simulateur :

Proposition 5.9. *Soient X et Y deux variables aléatoires à valeurs dans un même espace A. Considérons une troisième variable aléatoire Z à valeurs dans un ensemble B indépendante de X et Y. Alors pour toute fonction f définie sur $A \times B$ nous avons :*

$$\rho(f(X,Z), f(Y,Z)) \leqslant \rho(X,Y).$$

Concernant le point 3, l'argument est le suivant. Nous pouvons construire un distingueur entre les distributions $\mathcal{D}$ et $\mathcal{D}'$ à l'aide de l'adversaire et du simulateur. L'idée est tout simplement la suivante. Étant donné en entrée l'une de ces distributions $\mathcal{E}$, on fait fonctionner le jeu avec le simulateur et $\mathcal{E}$. Selon que $\mathcal{E} = \mathcal{D}$ ou $\mathcal{D}'$ nous serons alors dans le jeu i ou $i+1$. Notre distingueur renvoie alors 1 si le jeu est réussi et 0 sinon. On peut alors vérifier que la différence de probabilité de succès entre les deux jeux est exactement le succès de cet algorithme pour distinguer $\mathcal{D}$ et $\mathcal{D}'$. Nous obtenons donc bien l'inégalité avec la distance calculatoire.

Modèle de sécurité EUF-CMA des schémas de signature. Le modèle de sécurité classiquement utilisé pour un schéma de signature de type hache et signe est celui dit EUF-CMA. Dans ce modèle, l'adversaire peut avoir accès à N_{sign} signatures de son choix et calculer N_{hash} hachés de la fonction de hachage `Hache` qui est publique. Son objectif alors est de produire une signature valide d'un message dont il n'a jamais auparavant demandé une signature. Dans le contexte des réductions de sécurité utilisant les jeux, son rival dans ce modèle est défini comme suit.

Définition 6.1 (Modèle de sécurité EUF-CMA)**.** *Les procédures du défi dans le modèle de sécurité* EUF-CMA *du schéma de signature Wave, que l'on notera $\mathcal{S}_{\text{Wave}}$, sont définies comme :*

`proc Initialisation`	`proc Hache`$(\mathbf{m},\mathbf{r})$	`proc Signe`$(\mathbf{m})$
$(\mathrm{pk},\mathrm{sk}) \leftarrow$ `Gen`(1^λ)	`renvoie Hache`$(\mathbf{m},\mathbf{r})$	$\mathbf{r} \hookleftarrow \{0,1\}^{\lambda_0}$
$\mathbf{H}_{\text{pk}} \leftarrow \mathrm{pk}$		$\mathbf{s} \leftarrow$ `Hache`$(\mathbf{m},\mathbf{r})$
$(\varphi, \mathbf{H}_U, \mathbf{H}_V, \mathbf{S}, \mathbf{P}) \leftarrow \mathrm{sk}$		$\mathbf{e} \leftarrow D_{\varphi,\mathbf{H}_U,\mathbf{H}_V}(\mathbf{s}(\mathbf{S}^{-1})^\intercal)$
`renvoie` $\mathbf{H}_{\text{pk}}$		`renvoie` $(\mathbf{eP},\mathbf{r})$

`proc Fin`$(\mathbf{m},\mathbf{e},\mathbf{r})$
$\mathbf{s} \leftarrow$ `Hache`$(\mathbf{m},\mathbf{r})$
`renvoie` $\mathbf{e}\mathbf{H}_{\text{pk}}{}^\intercal = \mathbf{s} \wedge \lvert\mathbf{e}\rvert = w$

Le jeu EUF-CMA se déroule alors comme suit. L'adversaire fait appel à la procédure d'initialisation puis peut faire aux plus N_{sign} requêtes à `proc Signe` et N_{hash} appels à `proc Hache`. Le jeu est alors dit réussi si l'adversaire renvoie $(\mathbf{m},\mathbf{e},\mathbf{r})$ accepté par `proc Fin` et tel

que $\mathbf{m}$ n'a jamais été demandé à `proc Signe`. On définit alors le succès EUF-CMA contre le schéma Wave comme :

$$Succ_{Wave}^{\text{EUF-CMA}}(t, N_{\text{hash}}, N_{\text{sign}}) \stackrel{\triangle}{=} \max_{\mathcal{A}:|\mathcal{A}|\leqslant t} \left(\mathbb{P}\left(\mathcal{A} \text{ réussit le jeu EUF-CMA de Wave}\right)\right)$$

où la probabilité est calculée sur l'aléa interne de l'adversaire $\mathcal{A}$ ainsi que sur les procédure du défi. Notre objectif dans ce qui suit est alors de réduire la sécurité de Wave à des problèmes de codes que nous décrivons dans la prochaine section. Nous verrons alors que les propriétés des fonctions GPVM (voir la définition 5.1) sont essentielles.

6.2 Les problèmes de code considérés

Maintenant que nous avons vu le modèle de sécurité ainsi que le cadre dans lequel nous allons faire nos preuves, introduisons les problèmes calculatoires auxquels nous allons réduire de façon fine les fonctions GPVM Wave lors de leur instanciation en signature.

Le problème DOOM. Le premier problème calculatoire auquel nous nous réduisons est DOOM (voir le problème 1.12). Ce problème consiste pour les paramètres (n, q, R, ω, N) à se donner une matrice de parité $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ où $k \stackrel{\triangle}{=} \lfloor Rn \rfloor$ et N syndromes avec pour objectif de décoder uniquement *l'un d'entre eux* à la distance $w \stackrel{\triangle}{=} \lfloor \omega n \rfloor$. Ici w correspondra à la distance de signature, donc est plus grande que la borne de Gilbert-Varshamov. La distribution que nous considèrerons est alors donnée par :

$$\mathbf{H} \hookleftarrow \mathbb{F}_q^{(n-k)\times n} \quad \text{et} \quad \mathbf{s}_1, \cdots, \mathbf{s}_N \hookleftarrow \mathbb{F}_q^{n-k}$$

Nous définissons pour cette distribution le succès calculatoire de DOOM comme :

$$Succ^{\text{DOOM}(n,q,R,\omega,N)}(t) \stackrel{\triangle}{=} \max_{\mathcal{A}:|\mathcal{A}|\leqslant t} \Big(\mathbb{P}\big(\mathcal{A}(\mathbf{H}, \mathbf{s}_1, \cdots, \mathbf{s}_N) = \mathbf{e} \text{ tel que } \mathbf{e}\mathbf{H}^\intercal = \mathbf{s}_j \text{ pour un certain } j \in [\![1, N]\!]\big)\Big). \quad (6.1)$$

Le problème DOOM est naturel dans le modèle de sécurité EUF-CMA des signatures de type hache et signe. En effet, tout adversaire cherche à forger un message $\mathbf{m}$ de son choix, c'est à dire ici décoder le syndrome $\texttt{Hache}(\mathbf{m})$ où `Hache` est une fonction de hachage cryptographique. Dans ce contexte, si cela peut lui offrir un avantage, l'attaquant va naturellement hacher de nombreux messages (toujours de son choix) et tenter de décoder l'un d'entre eux, c'est à dire considérer le problème DOOM au lieu de SD. Nous avons étudié DOOM (voir §2.4) dans le cas binaire à la borne de Gilbert-Varshamov. Or comme nous l'avons vu il existe des algorithmes *exponentiellement* meilleurs que ceux résolvant SD pour ces paramètres. En revanche, comme nous l'avons constaté dans le cas ternaire §3.3.3 (nous observons le même phénomène en binaire [DST17c]) les meilleurs algorithmes résolvant DOOM gagnent de moins en moins par rapport à SD quand $\omega \stackrel{\triangle}{=} w/n$ s'éloigne de ω^- pour relativement rapidement ne plus donner aucun gain. Dans notre cas, la distance de signature est telle que DOOM ne donne plus aucun avantage algorithmique. Notre réduction de sécurité fine à DOOM est donc dans l'état de l'art algorithmique actuel une réduction au problème SD.

Distinguer un code $(U, U+V)$-généralisé d'un code aléatoire. Le second problème calculatoire auquel nous allons nous réduire est lié à la trappe utilisé dans Wave. En effet, l'avantage que nous utilisions pour signer est donné par un code $(U, U+V)$-généralisé. Ce code que l'on permute est rendu public. Nous allons réduire alors comme le schéma

de McEliece la sécurité de la clef au problème de distinguer un code public d'un code aléatoire. Pour cela considérons les paramètres (n, q, R_U, R_V). Soient les entiers $k_U \triangleq R_U \frac{n}{2}$, $k_V \triangleq R_V \frac{n}{2}$ et $k \triangleq k_U + k_V$. On définit les distributions,

— $\mathcal{D}_{\text{rand}}$ est la distribution uniforme sur les matrices de $\mathbb{F}_q^{(n-k)\times n}$,

— $\mathcal{D}_{\text{pub}}$ est la distribution uniforme sur les matrices de parité d'un code $(U, U+V)$-généralisé permuté aléatoire (voir §5.1.2) où U (resp. V) est un $[n/2, k_U]_3$-code (resp. $[n/2, k_V]_3$-code).

Le problème de distinguer $\mathcal{D}_{\text{rand}}$ et $\mathcal{D}_{\text{pub}}$ est dans l'état de l'art algorithmique, pour les paramètres que nous considérons dans Wave, exponentiel en n. Nous discuterons en détail de la difficulté de ce problème dans le prochain chapitre. Nous montrerons en particulier qu'il est NP-complet et nous proposerons un algorithme non trivial pour le résoudre.

6.3 La réduction

Nous présentons maintenant notre réduction de sécurité du schéma de signature Wave dans les modèles de sécurité EUF-CMA et de l'oracle aléatoire (la fonction de hachage est supposée être aléatoire). C'est l'objet du théorème qui suit.

Théorème 6.1 (Réduction de sécurité de Wave.)**.** *Soit N_{hash} (resp. N_{sign}) le nombre de requêtes faites à la fonction de hachage (resp. à l'oracle de signature). Supposons que la taille du sel $\mathbf{r} \in \mathbb{F}_3^{\lambda_0}$ est telle que $\lambda_0 = \lambda + 2\log_2(N_{\text{sign}})$ où λ est le paramètre de sécurité. Nous avons dans le modèle de l'oracle aléatoire pour tout t, $t_c = t + O\left(N_{\text{hash}} \cdot \lambda^2\right)$ et ε donné dans la proposition 5.10 :*

$$\begin{aligned} Succ^{\text{EUF-CMA}}_{Wave}(t, N_{\text{hash}}, N_{\text{sign}}) \leqslant\ & 2Succ^{\text{DOOM}(n,q,R,\omega,N)}(t_c) + \rho_c\left(\mathcal{D}_{\text{rand}}, \mathcal{D}_{\text{pub}}\right)(t_c) \\ & + N_{\text{sign}}\left(\mathbb{E}_{\mathbf{H}_{\text{pk}}}\left(\rho\left(\mathcal{D}_w^{\mathbf{H}_{\text{pk}}}, \mathcal{U}_w\right)\right) + \frac{\sqrt{\varepsilon}}{2} + \frac{N_{\text{hash}} + N_{\text{sign}}}{N_{\text{sign}}^2 \times 2^\lambda}\right) \\ & + \frac{1}{2}(N_{\text{hash}} + N_{\text{sign}})\sqrt{\varepsilon} + \frac{1}{2^\lambda} \end{aligned}$$

où $\mathcal{D}_w^{\mathbf{H}_{\text{pk}}}$ est la distribution suivante

— renvoie $(\mathbf{eP}, \mathbf{r})$ *distribué comme :* $\mathbf{s} \hookleftarrow \mathbb{F}_3^{n-k}$, $\mathbf{r} \hookleftarrow \{0,1\}^{\lambda_0}$, $\mathbf{e} \leftarrow D_{\varphi, \mathbf{H}_U, \mathbf{H}_V}(\mathbf{s}(\mathbf{S}^{-1})^\intercal)$.

avec $D_{\varphi, \mathbf{H}_U, \mathbf{H}_V}$ l'algorithme 4 utilisant les algorithmes 5 et 6. De plus, $\mathcal{U}_w$ est la distribution uniforme sur S_w.

Démonstration du théorème 6.1.

Soient $\mathcal{A}$ un $(t, N_{\text{sign}}, N_{\text{hash}}, \varepsilon)$-adversaire dans le modèle de sécurité EUF-CMA contre la signature $\mathcal{S}_{\text{Wave}}$. De plus, notons $R \triangleq k/n$ et $\omega \triangleq w/n$ le rendement et la distance relative de décodage considérés des $[n,k]_3$-codes dans $\mathcal{S}_{\text{Wave}}$. Considérons $(\mathbf{H}_0, \mathbf{s}_1, \cdots, \mathbf{s}_{N_{\text{hash}}})$ une instance de DOOM$(n, 3, R, \omega, N_{\text{hash}})$ uniformément distribuée. Permettons nous d'insister sur le fait que les syndromes $\mathbf{s}_j$ sont uniformément distribués sur $\mathbb{F}_3^{n-k}$.
Dans la démonstration qui suit nous allons présenter une série de jeux G_i. Nous désignerons alors par $\mathbb{P}(S_i)$ la probabilité de succès de l'algorithme $\mathcal{A}$ lors du jeu G_i.

Le jeu G_0 est celui considéré dans le modèle de sécurité EUF-CMA de $\mathcal{S}_{\text{Wave}}$.

Le jeu G_1 est identique à G_0 à moins que se produise l'évènement F défini comme : *un même sel $\mathbf{r}$ a été tiré lors de deux requêtes d'un message $\mathbf{m}$ à l'oracle de signature*. De cette façon, d'après [Sho04, lemme 1] nous obtenons,

$$\mathbb{P}(S_0) \leqslant \mathbb{P}(S_1) + \mathbb{P}(F).$$

Le lemme qui suit montre alors que pour $\lambda_0 = \lambda + 2\log_2(N_{\text{sign}})$, la probabilité de l'évènement F est négligeable en le paramètre de sécurité.

Lemme 6.1. *Nous avons pour $\lambda_0 = \lambda + 2\log_2(N_{\text{sign}})$,*

$$\mathbb{P}(F) \leqslant \frac{1}{2^\lambda}.$$

Démonstration du lemme 6.1.

Commençons par démontrer le lemme qui suit.

Lemme 6.2. *La probabilité de n'avoir aucune collision sur t tirages indépendants et uniforme dans un ensemble de taille n est majorée par t^2/n.*

Démonstration du lemme 6.2.

Notons pour $1 \leqslant i < j \leqslant t$ la variable aléatoire $X_{i,j}$ indicatrice de l'évènement "il y a collision lors du i-ème et j-ème tirage". La probabilité que nous cherchons à majorer est donc donnée par :

$$\mathbb{P}\left(\bigcup_{1\leqslant i<j\leqslant t} X_{i,j} = 1\right).$$

Or nous savons d'après l'inégalité de Bonferroni à l'ordre 1 que,

$$\mathbb{P}\left(\bigcup_{1\leqslant i<j\leqslant t} X_{i,j} = 1\right) \leqslant \sum_{1\leqslant i<j\leqslant t} \mathbb{P}(X_{i,j} = 1).$$

Les tirages se faisant maintenant dans un ensemble de taille n il est clair que

$$\mathbb{P}(X_{i,j} = 1) = \frac{1}{n}$$

d'où l'on en déduit le résultat. □

Dans notre cas, la probabilité de l'évènement F est majorée par l'inégalité qui précède pour $t = N_{\text{sign}}$ et $n = 2^{\lambda_0}$. De cette façon, avec $\lambda_0 = \lambda + 2\log_2 N_{\text{sign}}$, nous en déduisons que

$$\mathbb{P}(F) \leqslant \frac{N_{\text{sign}}^2}{2^{\lambda_0}} = \frac{1}{2^{\lambda_0 - 2\log_2(N_{\text{sign}})}} = \frac{1}{2^\lambda}$$

ce qui conclu la preuve. □

Le jeu G_2 consiste à modifier G_1 en remplaçant les procédures `Initialisation`, `Hache` et `Signe` comme suit (les modifications sont en rouge) :

`proc Initialisation`	`proc Hache`$(\mathbf{m}, \mathbf{r})$

$(\text{pk}, \text{sk}) \leftarrow$ `Gen`(1^λ)
$\mathbf{H}_{\text{pk}} \leftarrow \text{pk}$
$(\varphi, \mathbf{H}_U, \mathbf{H}_V, \mathbf{S}, \mathbf{P}) \leftarrow \text{sk}$
$j \leftarrow 0$
`renvoie` $\mathbf{H}_{\text{pk}}$
	`si` $L_{\mathbf{m}}$ `est indéfinie`
	  $L_{\mathbf{m}} \leftarrow N_{\text{sign}}$ `éléments aléatoires de` $\mathbb{F}_2^{\lambda_0}$
	`si` $\mathbf{r} \in L_{\mathbf{m}}$
	  $\mathbf{e}_{\mathbf{m},\mathbf{r}} \hookleftarrow S_w$
	  `renvoie` $\mathbf{e}_{\mathbf{m},\mathbf{r}}\mathbf{H}_{\text{pk}}^\intercal$
	`sinon`
	  $j \leftarrow j + 1$
	  `renvoie` $\mathbf{s}_j$

`proc Signe`$(\mathbf{m})$
`si` $L_{\mathbf{m}}$ `est indéfinie` $L_{\mathbf{m}} \leftarrow N_{\text{sign}}$ `éléments aléatoires de` $\mathbb{F}_2^{\lambda_0}$ $\mathbf{r} \leftarrow L_{\mathbf{m}}$`.suivant()` $\mathbf{s} \leftarrow$ `Hache`$(\mathbf{m}, \mathbf{r})$ $\mathbf{e} \leftarrow D_{\varphi,\mathbf{H}_U,\mathbf{H}_V}(\mathbf{s}(\mathbf{S}^{-1})^\intercal)$ `renvoie` $(\mathbf{eP}, \mathbf{r})$

Ici les appels à $L_{\mathbf{m}}.\texttt{next}()$ renvoient séquentiellement des éléments de $L_{\mathbf{m}}$. La liste est suffisamment grande pour répondre à toutes les requêtes. La procédure `Hache` quant à elle crée la liste $L_{\mathbf{m}}$ si besoin. De cette façon, si $\mathbf{r} \in L_{\mathbf{m}}$ la procédure renvoie $\mathbf{e}_{\mathbf{m},\mathbf{r}}\mathbf{H}_{\text{pk}}^{\intercal}$ où $\mathbf{e}_{\mathbf{m},\mathbf{r}}$ est uniformément distribuée sur S_w. Bien que nous ne l'utilisions pas encore, remarquons que $(\mathbf{e}_{\mathbf{m},\mathbf{r}}, \mathbf{r})$ est une signature valide du message $\mathbf{m}$. De plus, $\mathbf{e}_{\mathbf{m},\mathbf{r}}$ est ici stockée. Maintenant si $\mathbf{r} \notin L_{\mathbf{m}}$, la procédure `Hache` renvoie le syndrome $\mathbf{s}_j$ qui est lui-même uniformément distribué par hypothèse. Les sorties de la procédure `Signe` ne changent pas. En revanche, le sel $\mathbf{r}$ est désormais choisi dans $L_{\mathbf{m}}$.

La probabilité de succès de ce jeu peut être reliée à $\mathbb{P}(S_1)$ grâce au lemme qui suit.

Lemme 6.3.

$$\mathbb{P}(S_1) \leqslant \mathbb{P}(S_2) + \frac{N_{\text{hash}}}{2}\sqrt{\varepsilon} \quad \textit{où } \varepsilon \textit{ est donné dans la proposition 5.10.}$$

Avant de prouver ce lemme commençons par démontrer la proposition fondamental qui suit.

Proposition 6.1. *Soient les variables aléatoires discrètes X_i et Y_i où $i \in \{1, 2\}$ de même domaine $\mathcal{A}_i$. Notons pour $a_1 \in \mathcal{A}_1$, $p(.|a_1)$ la distribution de X_2 conditionnée à l'évènement $X_1 = a_1$ tandis que $q(.|a_1)$ est la distribution de Y_2 sachant que $Y_1 = a_1$. Nous avons,*

$$\rho\left(X_1X_2, Y_1Y_2\right) \leqslant \sup_{a_1 \in \mathcal{A}_1} \rho\left(p(.|a_1), q(.|a_1)\right) + \rho\left(X_1, Y_1\right).$$

Démonstration de la proposition 6.1.

Commençons par noter que,

$$\begin{aligned}\rho\left(X_1X_2, Y_1Y_2\right) &= \frac{1}{2}\sum_{a_1,a_2} |\mathbb{P}(X_1 = a_1, X_2 = a_2) - \mathbb{P}(Y_1 = a_1, Y_2 = a_2)| \\ &= \frac{1}{2}\sum_{a_1,a_2} |\mathbb{P}(X_2 = a_2|X_1 = a_1)\mathbb{P}(X_1 = a_1) \\ &\qquad - \mathbb{P}(Y_2 = a_2|Y_1 = a_1)\mathbb{P}(Y_1 = a_1)|\end{aligned}$$

Simplifions les notations et introduisons,

$$p(a_1) \triangleq \mathbb{P}(X_1 = a_1) \quad \text{et} \quad q(a_1) \triangleq \mathbb{P}(Y_1 = a).$$

Nous obtenons,

$$\begin{aligned}\rho\left(X_1X_2, Y_1Y_2\right) &= \frac{1}{2}\sum_{a_1,a_2} |p(a_2|a_1)p(a_1) - q(a_2|a_1)q(a_1)| \\ &= \frac{1}{2}\sum_{a_1,a_2} |p(a_2|a_1)p(a_1) - q(a_2|a_1)p(a_1) + q(a_2|a_1)p(a_1) - q(a_2|a_1)q(a_1)| \\ &\leqslant \frac{1}{2}\sum_{a_1,a_2} |p(a_2|a_1) - q(a_2|a_1)|\, p(a_1) + \frac{1}{2}\sum_{a_1,a_2} |p(a_1) - q(a_1)|\, q(a_2|a_1) \\ &\leqslant \sup_{a_1} \rho\left(p(.|a_1), q(.|a_1)\right) + \frac{1}{2}\sum_{a_1} |p(a_1) - q(a_1)| \underbrace{\sum_{a_2 \in \mathcal{A}_2} q(a_2|a_1)}_{=1} \\ &= \sup_{a_1 \in \mathcal{A}_1} \rho\left(p(.|a_1), q(.|a_1)\right) + \rho\left(X_1, Y_1\right).\end{aligned}$$

□

Démontrons maintenant le lemme 6.3.

Démonstration du lemme 6.3.

Les distributions des jeux 1 et 2 diffèrent à travers les sorties de l'oracle `Hache`. Dans le jeu 1, le syndrome renvoyé par `Hache` à la i-ème requête X_i est uniformément distribué (nous sommes dans le modèle de l'oracle aléatoire) dans $\mathbb{F}_3^{n-k}$. Dans le jeu 2, si un appel à `Hache` est fait avec la paire

$(\mathbf{m}, \mathbf{r})$ tel que $\mathbf{r} \in L_{\mathbf{m}}$, alors est renvoyé $Y_i = \mathbf{e}\mathbf{H}_{\text{pk}}^{\intercal}$ où $\mathbf{e}$ est uniformément distribué dans S_w tandis que si $\mathbf{r} \notin L_{\mathbf{m}}$, Y_i est uniformément distribuée. Nous avons,

$$\begin{aligned}\mathbb{P}(S_1) - \mathbb{P}(S_2) &= \sum_{\mathbf{H}} \mathbb{P}(\mathbf{H}_{\text{pk}} = \mathbf{H}) \left[\mathbb{P}(S_1 | \mathbf{H}_{\text{pk}} = \mathbf{H}) - \mathbb{P}(S_2 | \mathbf{H}_{\text{pk}} = \mathbf{H})\right] \\ &\leqslant \mathbb{E}_{\mathbf{H}_{\text{pk}}} \left(\rho\left(X_1 \cdots X_{N_{\text{hash}}}, Y_1 \cdots Y_{N_{\text{hash}}}\right)\right) \qquad (6.2)\end{aligned}$$

Notons qu'une application directe de la proposition 5.10 donne,

$$\mathbb{E}_{\mathbf{H}_{\text{pk}}} \left\{\rho\left(X_1, Y_1\right)\right\} \leqslant \frac{\sqrt{\varepsilon}}{2} \qquad (6.3)$$

Nous allons maintenant utiliser la proposition 6.1 pour borner $\rho\left(X_1 X_2, Y_1, Y_2\right)$. Ici $p(.|x_1)$ désigne la distribution conditionnelle de X_2 sachant que $X_1 = x_1$ tandis que $q(.|x_1)$ désigne la distribution conditionnelle de Y_2 étant donné l'évènement $Y_1 = x_1$. Nous avons :

$$\rho\left(X_1 X_2, Y_1 Y_2\right) = \sup_{x_1 \in \mathbb{F}_3^{n-k}} \rho\left(p(.|x_1), q(.|x_1)\right) + \rho\left(X_1, Y_1\right)$$

En utilisant maintenant la proposition 5.10 et l'équation (6.3) nous en déduisons que :

$$\begin{aligned}\mathbb{E}_{\mathbf{H}_{\text{pk}}} \left\{\rho\left(X_1 X_2, Y_1 Y_2\right)\right\} &\leqslant \frac{\sqrt{\varepsilon}}{2} + \frac{\sqrt{\varepsilon}}{2} \\ &= \sqrt{\varepsilon}.\end{aligned}$$

Un raisonnement par récurrence termine alors la preuve. □

Le jeu G_3 diffère du jeu 2 à travers les sorties des appels à `proc Signe` où au lieu d'être renvoyé "$(\mathbf{e}, \mathbf{r})$" avec "$\mathbf{e} \leftarrow D_{\varphi, \mathbf{H}_U, \mathbf{H}_V}(\mathbf{s}(\mathbf{S}^{-1})^{\intercal})$" nous renvoyons "$(\mathbf{e}\mathbf{P}, \mathbf{r})$" où "$\mathbf{e} \leftarrow \mathbf{e}_{\mathbf{m},\mathbf{r}}$". Les signatures $(\mathbf{e}, \mathbf{r})$ produites par `proc Signe` sont toujours valides. Nous allons prouver ici le lemme suivant.

Lemme 6.4.

$$\mathbb{P}\left(S_2\right) \leqslant \mathbb{P}\left(S_3\right) + N_{\text{sign}} \left(\mathbb{E}_{\mathbf{H}_{\text{pk}}} \left(\rho\left(\mathcal{U}_w, \mathcal{D}_w^{\mathbf{H}_{\text{pk}}}\right)\right) + \frac{\sqrt{\varepsilon}}{2} + \frac{N_{\text{hash}} + N_{\text{sign}}}{2^{\lambda_0}}\right)$$

où ε est donné dans la proposition 5.10.

Le lemme qui suit nous sera d'une grande aide et est *fondamental*. En effet, c'est ce dernier qui montre que la condition sans fuite d'information *en moyenne* sur les entrées des fonction GPVM est suffisante pour la réduction de sécurité. Ceci est essentiellement possible grâce à l'utilisation du sel $\mathbf{r}$.

Lemme 6.5. *Notons $X_i(\mathbf{m})$ la distribution de sortie du $i$ème appel à `proc Signe` dans le jeu 2 lorsque $\mathbf{m}$ est demandé tandis que $Y_i(\mathbf{m})$ dénote cette sortie dans le jeu 3.*
Alors, pour tout message $\mathbf{m}$ nous avons,

$$\rho\left(X_i(\mathbf{m}), Y_i(\mathbf{m})\right) \leqslant \mathbb{E}_{\mathbf{H}_{\text{pk}}} \left(\rho\left(\mathcal{U}_w, \mathcal{D}_w^{\mathbf{H}_{\text{pk}}}\right)\right) + \varepsilon + \frac{N_{\text{hash}} + N_{\text{sign}}}{2^{\lambda_0}}$$

où $\varepsilon_{\mathbf{H}_{\text{pk}}}$ est défini comme

$$\varepsilon_{\mathbf{H}_{\text{pk}}} \stackrel{\triangle}{=} \rho\left(\mathbf{e}\mathbf{H}_{\text{pk}}^{\intercal}, \mathbf{s}\right)$$

avec $\mathbf{s} \hookleftarrow \mathbb{F}_3^{n-k}$, $\mathbf{e} \hookleftarrow \mathcal{U}_w$ et $\mathcal{U}_w$ la distribution uniforme sur S_w.

Démonstration du lemme 6.5.

Notons respectivement $\mathcal{D}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ et $\mathcal{D}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ les distributions de $X_i(\mathbf{m})$ et $Y_i(\mathbf{m})$. Ces dernières sont obtenues comme suit,

— $(\mathbf{e}\mathbf{P}, \mathbf{r}) \hookleftarrow \mathcal{D}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ où $\mathbf{r} \leftarrow L_{\mathbf{m}}.\texttt{next}(), \mathbf{s} \leftarrow \text{Hache}(\mathbf{m}, \mathbf{r}), \mathbf{e} \leftarrow D_{\varphi, \mathbf{H}_U, \mathbf{H}_V}(\mathbf{s}(\mathbf{S}^{-1})^{\intercal})$.

— $(\mathbf{e},\mathbf{r}) \hookleftarrow \mathcal{D}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ où $\mathbf{r} \leftarrow L_{\mathbf{m}}.\texttt{next}(), \mathbf{e} \leftarrow \mathbf{e}_{\mathbf{m},\mathbf{r}}$

Soient $(\mathbf{m}_1,\mathbf{r}_1),\ldots,(\mathbf{m}_t,\mathbf{r}_t)$ les requêtes faites à `proc Hache`, incluant celles faites jusque-là par l'oracle `Signe` et définissons :

$$R \stackrel{\triangle}{=} \{\mathbf{r}_i : 1 \leqslant i \leqslant t\}.$$

Nous avons,

$$t \leqslant q \stackrel{\triangle}{=} N_{\text{hash}} + N_{\text{sign}}.$$

Introduisons maintenant les distributions $\mathcal{E}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ et $\mathcal{E}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ définies comme $\mathcal{D}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m}), \mathcal{D}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ mais conditionnées par l'évènement $\mathbf{r} \notin R$. Or comme $\mathbf{r} \notin R$ se produit avec probabilité $\geqslant 1 - \frac{q}{2^{\lambda_0}}$ (car $\mathbf{r}$ est uniformément tiré dans $\{0,1\}^{\lambda_0}$), nous avons

$$\rho(\mathcal{D}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m}), \mathcal{D}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})) \leqslant \rho(\mathcal{E}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m}), \mathcal{E}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})) + \frac{q}{2^{\lambda_0}} \tag{6.4}$$

Considérons maintenant la distribution intermédiaire $\mathcal{D}_{2.5}^{\mathbf{H}_{\text{pk}}}$ définie comme

— $(\mathbf{eP},\mathbf{r}) \hookleftarrow \mathcal{D}_{2.5}^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ où $\mathbf{r} \leftarrow L_{\mathbf{m}}.\texttt{next}(), \mathbf{s} \hookleftarrow \mathbb{F}_3^{n-k}, \mathbf{e} \leftarrow D_{\varphi,\mathbf{H}_U,\mathbf{H}_V}(\mathbf{s}(\mathbf{S}^{-1})^{\intercal})$.

et $\mathcal{E}_{2.5}^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ la distribution $\mathcal{D}_{2.5}^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ conditionnée par l'évènement $\mathbf{r} \notin R$. Maintenant comme $\mathbf{r} \notin R$ dans $\mathcal{E}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$, tout appel à Hache$(\mathbf{m},\mathbf{r})$ est nouveau et de cette façon renvoie un syndrome $\mathbf{s}$ qui est $\varepsilon_{\mathbf{H}_{\text{pk}}}$ proche statistiquement de l'uniforme. Donc nous avons,

$$\rho(\mathcal{E}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m}), \mathcal{E}_{2.5}^{\mathbf{H}_{\text{pk}}}(\mathbf{m})) \leqslant \varepsilon_{\mathbf{H}_{\text{pk}}} \tag{6.5}$$

Comparons désormais $\mathcal{E}_{2.5}^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ et $\mathcal{E}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$. La distribution $\mathcal{E}_{2.5}^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ renvoie un sel aléatoire $\mathbf{r} \notin R$ et $\mathbf{e}$ tirée selon $\mathcal{D}_w^{\mathbf{H}_{\text{pk}}}$ tandis que $\mathcal{E}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})$ renvoie elle aussi un sel aléatoire $\mathbf{r} \notin R$ mais une erreur $\mathbf{e}$ uniformément distribuée sur S_w. Ainsi,

$$\rho(\mathcal{E}_{2.5}^{\mathbf{H}_{\text{pk}}}(\mathbf{m}), \mathcal{E}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})) \leqslant \rho(\mathcal{D}_w^{\mathbf{H}_{\text{pk}}}, \mathcal{U}_w). \tag{6.6}$$

En combinant maintenant les équations (6.4),(6.5) et (6.6) nous obtenons,

$$\begin{aligned}\rho(\mathcal{D}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m}), \mathcal{D}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})) &\leqslant \rho(\mathcal{E}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m}), \mathcal{E}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})) + \frac{q}{2^{\lambda_0}} \\ &\leqslant \rho(\mathcal{E}_2^{\mathbf{H}_{\text{pk}}}(\mathbf{m}), \mathcal{E}_{2.5}^{\mathbf{H}_{\text{pk}}}(\mathbf{m})) + \rho(\mathcal{E}_{2.5}^{\mathbf{H}_{\text{pk}}}(\mathbf{m}), \mathcal{E}_3^{\mathbf{H}_{\text{pk}}}(\mathbf{m})) + \frac{q}{2^{\lambda_0}} \\ &\leqslant \varepsilon_{\mathbf{H}_{\text{pk}}} + \rho(\mathcal{D}_w^{\mathbf{H}_{\text{pk}}}, \mathcal{U}_w) + \frac{q}{2^{\lambda_0}}.\end{aligned}$$

ce qui conclut la preuve du lemme.

□

Démonstration du lemme 6.4.

Notons $X_1,\cdots,X_{N_{\text{sign}}}$ les distributions de sortie à chaque appel de `Signe` dans le jeu 2 tandis que nous notons $Y_1,\cdots,Y_{N_{\text{sign}}}$ celles pour le jeu 3. Nous avons,

$$\mathbb{P}(S_2) \leqslant \mathbb{P}(S_3) + \rho\left(X_1\cdots X_{N_{\text{sign}}}, Y_1\cdots Y_{N_{\text{sign}}}\right).$$

Nous bornons la distance statistique avec un raisonnement par récurrence à l'aide la proposition 6.1 et en bornant chaque terme avec le lemme 6.5 ce qui conclut la preuve du lemme.

□

Le jeu G_4 est celui où nous remplaçons la matrice $\mathbf{H}_{\text{pk}}$ par $\mathbf{H}_0$. Ceci est possible car l'oracle de signature n'utilise plus la trappe. De cette façon nous allons forcer l'adversaire à fabriquer une solution du problème DOOM considéré. Notons alors que s'il existe une différence de probabilité de succès de l'adversaire entre les deux jeux c'est que nous avons un distingueur entre les distributions $\mathcal{D}_{\text{pub}}$ et $\mathcal{D}_{\text{rand}}$. De cette façon,

$$\mathbb{P}(S_3) \leqslant \mathbb{P}(S_4) + \rho_c\left(\mathcal{D}_{\text{pub}}, \mathcal{D}_{\text{rand}}\right)(t_c).$$

Nous n'entrerons pas ici dans les détails mais il est possible avec une bonne structure de données "d'émuler" les listes $L_{\mathbf{m}}$ de façon à ce que chaque opération (particulièrement sa création) soit de complexité de l'ordre de $O(\lambda)$ comme nous l'avons démontré dans la version longue de notre papier sur Wave [DST19b]. Maintenant n étant linéaire en le paramètre de sécurité (les meilleures attaques contre les hypothèses calculatoires sont exponentielles en n), un appel à la procédure `proc Hache` est de l'ordre de $O(n^2)$ et le temps de calcul pour faire tourner les jeux est de la forme $t_c = t + O\left(N_{\text{hash}} \cdot \lambda^2\right)$.

Le jeu G_5 diffère maintenant dans la procédure de finalisation.

```
proc Fin(m, e, r)
s ← Hache(m, r)
b ← eH_pk^⊺ = s ∧ |e| = w
renvoie b ∧ r ∉ L_m
```

Supposons que l'adversaire renvoie une signature valide $(\mathbf{e}, \mathbf{r})$ pour le message $\mathbf{m}$. La probabilité de succès de l'adversaire dans G_5 est alors exactement la probabilité de l'évènement "$S_4 \wedge (\mathbf{r} \notin L_{\mathbf{m}})$".

Si la signature est *valide* c'est que le message $\mathbf{m}$ n'a jamais été demandé à l'oracle `Signe`. De cette façon l'adversaire n'a jamais eu accès aux éléments de la liste $L_{\mathbf{m}}$. Donc les deux évènements sont indépendants et nous obtenons,

$$\mathbb{P}(S_5) = (1 - 2^{-\lambda_0})^{N_{\text{sign}}} \mathbb{P}(S_4).$$

Or nous avons choisi $\lambda_0 = \lambda + 2\log_2(N_{\text{sign}}) \geqslant \log_2(N_{\text{sign}}^2)$, ce qui donne :

$$\left(1 - 2^{-\lambda_0}\right)^{N_{\text{sign}}} \geqslant \left(1 - \frac{1}{N_{\text{sign}}^2}\right)^{N_{\text{sign}}} \geqslant \frac{1}{2}.$$

D'où,

$$\mathbb{P}(S_5) \geqslant \frac{1}{2}\mathbb{P}(S_4). \tag{6.7}$$

La probabilité $\mathbb{P}(S_5)$ est alors exactement la probabilité pour $\mathcal{A}$ de renvoyer $\mathbf{e}_j \in S_w$ telle que $\mathbf{e}_j {\mathbf{H}_0}^{\intercal} = \mathbf{s}_j$ pour un certain indice j. Cela donne,

$$\mathbb{P}(S_5) \leqslant \ Succ_{\text{DOOM}}^{n,k,N_{\text{hash}},w}(t_c). \tag{6.8}$$

En combinant toutes les équations entre les jeux nous concluons facilement la preuve du théorème 6.1.

□

Chapitre 7

Distinguer un code $(U, U + V)$-généralisé permuté d'un code aléatoire

Introduction

La famille des codes $(U, U + V)$-généralisés permutés forme l'ensemble des trappes pouvant être utilisées dans l'instanciation de Wave en schéma de signature. Il est donc naturel que la sécurité du schéma de signature repose sur la difficulté de retrouver ne serait-ce qu'une information partielle sur la trappe utilisée étant donné que tout code n'est pas un code $(U, U+V)$-généralisé permuté. Dans notre réduction de sécurité apparaît alors clairement le problème de *distinguer* un code $(U, U + V)$-généralisé permuté d'un code aléatoire. Ce problème peut cependant sembler au premier regard ad-hoc. Néanmoins, même pour une sous-classe de ce dernier, décider si un code est un $(U, U + V)$-permuté ou non est NP-complet comme nous allons le prouver ici. Plus précisément, nous aurons ce résultat grâce à une réduction au problème NP-complet du mariage tri-dimensionnel (MTD) que nous avons introduit dans §1.1.1. En revanche, comme nous le remarquerons cette réduction se fait dans le cas où $\dim U < \dim V$. Or le décodeur que nous avons présenté dans §5.2 nécessite $\dim U > \dim V$ pour produire en temps polynomial des erreurs de poids relatif hors de l'intervalle $\left[\frac{q-1}{q}(1-R), R + \frac{q-1}{q}(1-R)\right]$ où R désigne le rendement du code $(U, U + V)$-généralisé utilisé. Cette difficulté n'est pas anecdotique. Dans le cas binaire où $\dim U > \dim V$ il est facile de distinguer un code $(U, U+V)$-permuté d'un code aléatoire à l'aide du hull comme montré dans les propositions 5.1 et 5.2. Ce problème explique pourquoi l'ancêtre de Wave, Surf [DST17c] ne peut pas être utilisé comme signature de type hache et signe.

Fort heureusement cette situation change drastiquement si nous généralisons la construction des codes $(U, U + V)$ tout en conservant l'avantage qu'offre leur décodeur. La première façon de faire, que nous n'évoquerons pas dans ce document, est de ne plus considérer deux codes U et V de même longueur mais trois U, V et W. Le rationnel est alors le même que pour les codes $(U, U + V)$. Nous pouvons mélanger ces trois codes de façon à ce qu'en décodant dans W puis V en utilisant la connaissance du résultat et finalement dans U nous obtenons un avantage. Pour cela le code U doit apparaître trois fois et les deux autres au moins une fois. De plus, la dimension totale du code doit être donnée par $\dim U + \dim V + \dim W$. Par exemple, les deux codes

$$(U, U + V, U + V + W) \triangleq \{(\mathbf{u}, \mathbf{u} + \mathbf{v}, \mathbf{u} + \mathbf{v} + \mathbf{w}) : \mathbf{u} \in U, \mathbf{v} \in V \text{ et } \mathbf{w} \in W\}$$

ou encore,

$$(U+V, U+W, U+V+W) \stackrel{\triangle}{=} \{(\mathbf{u}+\mathbf{v}, \mathbf{u}+\mathbf{w}, \mathbf{u}+\mathbf{v}+\mathbf{w}) : \mathbf{u} \in U, \mathbf{v} \in V \text{ et } \mathbf{w} \in W\}$$

offrent un avantage au décodage. Dans notre cas nous avons cependant considéré l'autre généralisation, celle des codes $(U, U+V)$-généralisés qui rappelons-le sont définis comme :

$$\{(\mathbf{a} \odot \mathbf{u} + \mathbf{b} \odot \mathbf{v}, \mathbf{c} \odot \mathbf{u} + \mathbf{d} \odot \mathbf{v}) : \mathbf{u} \in U \text{ et } \mathbf{v} \in V\}.$$

où pour que notre algorithme de décodage ait un avantage sur les décodages génériques il n'est pas n'est pas nécessaire d'avoir $\mathbf{a} = \mathbf{c} = \mathbf{d} = \mathbf{1}_{n/2}$ et $\mathbf{b} = \mathbf{0}_{n/2}$ (*i.e* : les codes $(U, U+V)$) mais seulement $\mathbf{a} \odot \mathbf{c}$ et $\mathbf{a} \odot \mathbf{d} - \mathbf{b} \odot \mathbf{c}$ sans coordonnées égales à 0. Ceci offre alors une grande liberté de choix des vecteurs $\mathbf{a}, \mathbf{b}, \mathbf{c}$ et $\mathbf{d}$ à coordonnées dans $\mathbb{F}_q$ dès-lors que $q \geqslant 3$. De plus, ces mots permettent de contrecarrer complètement toute attaque utilisant le hull. Ces-derniers changent la nature du problème considéré et nous pensons que notre preuve de son caractère NP-complet peut s'étendre aux paramètres $\dim U > \dim V$. De plus, il semble que la meilleure attaque structurelle se fonde sur l'observation suivante. Les codes $(U, U+V)$-généralisés que nous considèrerons ont des mots de poids légèrement inférieur à la distance minimale attendue d'un code aléatoire de mêmes paramètres comme nous le montrerons dans la deuxième section de ce chapitre. Il est alors très tentant de conjecturer que la seule façon pour distinguer un code aléatoire d'un code $(U, U+V)$-généralisé permuté est de détecter ces mots de poids légèrement anormal. En tout cas c'est l'approche que nous proposons de suivre dans ce chapitre en donnant un algorithme dans 7.2.2 accomplissant cette tache. Ce-dernier étant de complexité exponentielle, il est très facile de choisir les paramètres des codes $(U, U+V)$-généralisés considérés pour éviter notre attaque.

Notons pour conclure que notre algorithme repose sur les meilleurs algorithmes dont nous disposons pour décoder des mots bruités d'un code aléatoire. Il s'agit donc exactement des algorithmes utilisés pour résoudre le problème du décodage générique. Il semble donc tenant de conjecturer que l'on puisse réduire le problème de distinguer la structure $(U, U+V)$-généralisé au décodage générique.

7.1 Un problème NP-complet

La sécurité sur la clef du schéma de signature Wave repose sur le problème de savoir si un code linéaire est un code $(U, U+V)$-généralisé permuté ou non. Nous allons ici nous intéresser à une sous-classe de ce problème, à savoir

Problème 7.1 (Distinguer un code $(U, U+V)$ permuté d'un code quelconque)**.**

— Instance : *un code binaire $\mathcal{C}$ de longueur n et un entier k_U,*

— Décision : *il existe une permutation π de $\mathbb{F}_2^n$ tel que $\pi(\mathcal{C})$ est un code $(U, U+V)$ où $\dim(U) = k_U$ et $|\operatorname{Supp}(V)| = n/2$.*

Rappelons qu'un code $(U, U+V)$ est un code $(U, U+V)$-généralisé où :

$$\mathbf{a} = \mathbf{c} = \mathbf{d} = \mathbf{1} \quad \text{et} \quad \mathbf{b} = \mathbf{0}.$$

Dans tout ce qui suit nous utiliserons le point de vue matrice génératrice, c'est à dire que tout $[n,k]_2$-code sera spécifié par une matrice de rang plein $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ dite génératrice. L'objectif de cette section est de démontrer le théorème suivant.

Théorème 7.1. *Le problème 7.1 est* NP*-complet.*

La preuve de ce résultat se fera par une réduction au problème NP-complet du mariage tri-dimensionnel (MTD) que nous avons défini dans §1.1.1. Notons que cette réduction est la même que celle prouvant que le problème du décodage est NP-complet. Étant donné une instance du problème 7.1 : T et $U \subseteq T \times T \times T$ où $t \triangleq |T|$ et $s \triangleq |U|$, nous noterons $\mathbf{G}_{\text{MTD}} \in \mathbb{F}_2^{s \times 3t}$ la transposée de la matrice d'incidence associée (voir la discussion suivant le problème 1.6). Nous utiliserons alors de façon cruciale le lemme 1.1 montrant que :

$$\text{l'instance } T, U \text{ admet une solution } \iff \exists \mathbf{m} \in \mathbb{F}_2^s : |\mathbf{m}| = t \text{ et } \mathbf{m}\mathbf{G}_{\text{MTD}} = \mathbf{1}.$$

Remarque 7.1. Sans perte de généralité nous pouvons supposer que dans l'instance considérée de MTD nous avons $s \geqslant t + 1$ et qu'il n'existe pas de colonne nulle dans $\mathbf{G}_{\text{MTD}}$. Autrement, il serait facile de décider si le problème admet une solution.

Afin de réduire la problème 7.1 à MTD nous allons utiliser certaines des astuces introduites dans [BGK17 ; Wie06] (en ajoutant des matrices identités et invoquant des arguments de distance minimale).

La preuve du théorème 7.1 passera par une réduction intermédiaire au problème ad-hoc qui suit.

Problème 7.2. (Distinguer un code quelconque d'un code $(U, U + V)$ permuté à support contraint).

— Instance : *une matrice* $\mathbf{G} \in \mathbb{F}_2^{k \times n}$*, des entiers* k_U *et* M*,*

— Décision : *il existe une matrice de permutation* $\mathbf{P} \in \mathbb{F}_2^{n \times n}$ *tel que* $\mathbf{GP}$ *est une matrice génératrice d'un code* $(U, U + V)$ *vérifiant* $\dim(U) = k_U$*,* $|\operatorname{Supp}(U)| \geqslant M$ *et* $|\operatorname{Supp}(V)| = n/2$*.*

Ce problème est clairement dans NP. Nous allons démontrer la proposition qui suit montrant son caractère NP-difficile.

Proposition 7.1. *Le problème 7.2 est* NP*-complet.*

Commençons par rappeler que la distance minimale d'un code $\mathcal{C}$ est définie comme :

$$\min\{|\mathbf{c}| : \mathbf{c} \in \mathcal{C}\}.$$

Nous rappelons dans le lemme qui suit la distance minimale d'un code $(U, U + V)$ ainsi que la forme des mots l'atteignant. Ce lemme sera crucial pour la preuve de la proposition 7.1.

Lemme 7.1. *Soit* U (resp. V) *un code de distance minimale* d_U (resp. d_V)*. La distance minimale* d *du code* $(U, U + V)$ *est donnée par :*

$$d = \min(2d_U, d_V).$$

De plus, les mots de code atteignant ce minimum vérifient nécessairement l'un des points qui suit :

1. $(\mathbf{u}, \mathbf{u})$ *avec* $|\mathbf{u}| = d_U$*,*
2. $(\mathbf{0}, \mathbf{v})$ *avec* $|\mathbf{v}| = d_V$*,*
3. $(\mathbf{u}, \mathbf{0})$ *avec* $|\mathbf{u}| = d_V$*,*
4. $(\mathbf{u}, \mathbf{u} + \mathbf{v})$ *avec* $\mathbf{u} \neq \mathbf{0}$*,* $\operatorname{Supp}(\mathbf{u}) \subsetneq \operatorname{Supp}(\mathbf{v})$ *et* $|\mathbf{v}| = d_V$*.*

Démonstration du lemme 7.1.

Un code $(U, U + V)$ contient les mots $(\mathbf{u}, \mathbf{u})$ où $\mathbf{u} \in U$ et les mots $(\mathbf{0}, \mathbf{v})$ avec $\mathbf{v} \in V$. Il est donc clair que

$$d \leqslant \min(2d_U, d_V). \tag{7.1}$$

Considérons maintenant un mot de code $(\mathbf{u}, \mathbf{u}+\mathbf{v})$ non nul de $(U, U+V)$. Si $\mathbf{v} = \mathbf{0}$, alors nous avons $|(\mathbf{u}, \mathbf{u}+\mathbf{v})| = 2|\mathbf{u}| \geqslant 2d_U$. Maintenant si $\mathbf{v} \neq \mathbf{0}$,

$$\begin{aligned} |(\mathbf{u}, \mathbf{u}+\mathbf{v})| &= |\mathbf{u}| + |\mathbf{u}+\mathbf{v}| \\ &\geqslant |\mathbf{u}| + |\mathbf{v}| - |\mathbf{u}| \quad \text{(par inégalité triangulaire)} \\ &= |\mathbf{v}| \\ &\geqslant d_V. \end{aligned}$$

Nous en déduisons que $d \geqslant \min(2d_U, d_V)$ et de cette façon avec (7.1) nous obtenons le résultat concernant la distance minimale d'un code $(U, U+V)$.
Considérons maintenant $(\mathbf{u}, \mathbf{u}+\mathbf{v}) \in (U, U+V)$ tel que

$$\mathbf{u} \neq \mathbf{0}, \quad \mathbf{v} \neq \mathbf{0}, \quad \mathbf{u}+\mathbf{v} \neq \mathbf{0} \quad \text{et} \quad \operatorname{Supp}(\mathbf{u}) \nsubseteq \operatorname{Supp}(\mathbf{v}) \tag{7.2}$$

De l'équation $|\mathbf{u}+\mathbf{v}| = |\mathbf{u}| - 2|\operatorname{Supp}(\mathbf{u}) \cap \operatorname{Supp}(\mathbf{v})| + |\mathbf{v}|$ nous en déduisons que :

$$|(\mathbf{u}, \mathbf{u}+\mathbf{v})| = 2|\mathbf{u}| - 2|\operatorname{Supp}(\mathbf{u}) \cap \operatorname{Supp}(\mathbf{v})| + |\mathbf{v}| \tag{7.3}$$

Maintenant comme $\operatorname{Supp}(\mathbf{u}) \nsubseteq \operatorname{Supp}(\mathbf{v})$, nous avons $\operatorname{Supp}(\mathbf{u}) \cap \operatorname{Supp}(\mathbf{v}) \subsetneq \operatorname{Supp}(\mathbf{u})$. De cette façon, d'après l'équation (7.3) :

$$|(\mathbf{u}, \mathbf{u}+\mathbf{v})| > |\mathbf{v}| > 0$$

ce qui implique que $(\mathbf{u}, \mathbf{u}+\mathbf{v})$ vérifiant (7.2) ne peut pas atteindre la distance minimale du fait que $(\mathbf{0}, \mathbf{v}) \in (U, U+V)$ ce qui conclut la preuve. □

Nous sommes maintenant prêts à prouver la proposition 7.1.

Démonstration de la proposition 7.1.

Une réduction polynomiale du problème MTD au problème 7.2. Soit une matrice $\mathbf{G}_{\text{3DM}} \in \mathbb{F}_2^{s \times 3t}$ instance du problème MTD ne contenant aucune colonne nulle et telle que $s \geqslant t+1$. Définissons pour les entiers p, u :

$$\mathbf{1}_p(u) \triangleq \underbrace{(\mathbf{1}_p \cdots \mathbf{1}_p)}_{\text{u fois}} \in \mathbb{F}_2^{p \times up}$$

où $\mathbf{1}_p$ est la matrice identité de taille $p \times p$. De plus, nous désignerons par $\mathbf{0}_{p \times u}$ la matrice nulle de taille $p \times u$. Nous construisons maintenant en temps polynomial sur la taille de l'entrée :

$$\mathbf{G} = \left(\begin{array}{c|c} \mathbf{1}_t(7) \quad \mathbf{0}_{t \times 4(s-t)} & \mathbf{0}_{t \times (4s+3t)} \\ \hline \mathbf{0}_{s \times (4s+3t)} & \mathbf{1}_s(4) \quad \mathbf{G}_{\text{3DM}} \end{array} \right) \in \mathbb{F}_2^{(s+t) \times 2(4s+3t)} \tag{7.4}$$

Notre réduction va alors considérer l'instance du problème 7.2 :

$$(\mathbf{G}, t, 7t).$$

Instance avec solution de MTD $\implies$ Instance avec solution du problème 7.2. Supposons que l'instance $\mathbf{G}_{\text{3DM}}$ de MTD admette une solution, *i.e* : il existe t lignes de la matrice à supports disjoints. Cela donne l'existence d'un matrice de permutation $\mathbf{P}_1$ de taille $(4s+3t)$ telle que t lignes de la matrices $\begin{pmatrix} \mathbf{1}_s(4) & \mathbf{G}_{\text{3DM}} \end{pmatrix} \mathbf{P}_1$ forment $\begin{pmatrix} \mathbf{1}_t(7) & \mathbf{0}_{t \times 4(s-t)} \end{pmatrix}$. Donc $\mathbf{GP}$ où $\mathbf{P}$ est la matrice de permutation

$$\mathbf{P} \triangleq \begin{pmatrix} \mathbf{1}_{4s+3t} & \mathbf{0}_{4s+3t} \\ \mathbf{0}_{4s+3t} & \mathbf{P}_1 \end{pmatrix}$$

agissant seulement sur les $4s+3t$ colonnes, génère un code $(U, U+V)$ où U est de matrice génératrice $\begin{pmatrix} \mathbf{1}_t(7) & \mathbf{0}_{t \times 4(s-t)} \end{pmatrix}$. Ce dernier est donc de dimension t et de support de taille $7t$ tandis que le code V est quant à lui généré par $\begin{pmatrix} \mathbf{1}_s(4) & \mathbf{G}_{\text{3DM}} \end{pmatrix} \mathbf{P}_1$. Du fait qu'aucune colonne $\mathbf{G}_{\text{3DM}}$ n'est nulle, nous avons $|\operatorname{Supp}(V)| = 4s+3t$.

Instance avec solution de 7.2 $\implies$ Instance avec solution de MTD. Réciproquement, supposons qu'il existe une matrice de permutation binaire $\mathbf{P}$ de taille $2(4s+3t)\times 2(4s+3t)$ telle que $\mathbf{GP}$ génère un code $(U, U+V)$ où $\dim(U) = t$, $|\mathrm{Supp}(U)| \geqslant 7t$ et $|\mathrm{Supp}(V)| = 4s+3t$.

Commençons alors par le lemme qui suit donnant la distance minimale du code généré par $\mathbf{G}$.

Lemme 7.2. *La matrice $\mathbf{G}$ génère un code de distance minimale 7. De plus, les mots de code atteignant la distance minimale sont exactement les lignes de $\mathbf{G}$.*

Démonstration du lemme 7.2.

La somme de $r > 1$ lignes de la matrice $\begin{pmatrix}\mathbf{1}_s(4) & \mathbf{G}_{3\mathrm{DM}}\end{pmatrix}$ (*resp.* $\begin{pmatrix}\mathbf{1}_t(7) & \mathbf{0}_{t\times 4(s-t)}\end{pmatrix}$) donne un mot de poids au moins $4r > 7$ (*resp.* $7r > 7$). De plus, toutes les lignes de $\mathbf{G}$ sont de poids 7 ce qui conclut la preuve du lemme. □

Il s'ensuit directement que la distance minimale du code $(U, U+V)$ considéré est 7. Donc d'après le lemme 7.1, nous obtenons :

$$7 = \min(2d_U, d_V) \Rightarrow d_V = 7 \text{ et } d_U \geqslant 4$$

où d_U (*resp.* d_V) désigne la distance minimale du code U (*resp.* V). Cette propriété cruciale donne les lemmes qui suivent. Ces-derniers nous en apprenant plus sur la structure du code $(U, U+V)$ dont $\mathbf{GP}$ est la matrice génératrice.

Lemme 7.3. *Pour tout $\mathbf{u} \in U$ nous avons :*

$$(\mathbf{u}, \mathbf{0})\mathbf{P}^{-1} \textit{ est une ligne de } \mathbf{G} \iff (\mathbf{0}, \mathbf{u})\mathbf{P}^{-1} \textit{ est une ligne de } \mathbf{G}$$

Démonstration du lemme 7.3.

Nous savons que pour tout $\mathbf{u} \in U$, le mot $(\mathbf{u}, \mathbf{u}) \in (U, U+V)$. Il est donc clair que pour tout $\mathbf{u} \in U$:

$$(\mathbf{u}, \mathbf{0}) \in (U, U+V) \iff (\mathbf{0}, \mathbf{u}) \in (U, U+V).$$

De plus d'après le lemme 7.2, les lignes de $\mathbf{G}$ sont des mots de poids 7 et sont exactement les mots dont le poids est la de distance minimale. Donc $(\mathbf{u}, \mathbf{0})\mathbf{P}^{-1}$ est une ligne $\mathbf{G}$ si et seulement si $(\mathbf{0}, \mathbf{u})\mathbf{P}^{-1}$ est une ligne de $\mathbf{G}$ ce qui conclut la preuve du lemme. □

Lemme 7.4. *Il y a dans $\mathbf{G}$ exactement :*

- *t lignes de la forme $(\mathbf{u}, \mathbf{0})\mathbf{P}^{-1}$ où les $\mathbf{u}$ forment une base du code U et $|\mathbf{u}| = 7$,*
- *t lignes de la forme $(\mathbf{0}, \mathbf{u})\mathbf{P}^{-1}$ où $\mathbf{u} \in U$ et $|\mathbf{u}| = 7$,*
- *$s - t$ lignes de la forme $(\mathbf{0}, \mathbf{v})\mathbf{P}^{-1}$ où $\mathbf{v} \in V$ et $|\mathbf{v}| = 7$ mais $\mathbf{v} \notin U$.*

Démonstration du lemme 7.4.

Les lemmes 7.1 et 7.2 impliquent que toutes les lignes $\mathbf{G}$ sont nécessairement de la forme qui suit pour $\mathbf{u} \in U$ et $\mathbf{v} \in V$:

1. $(\mathbf{u}, \mathbf{u})\mathbf{P}^{-1}$ avec $2|\mathbf{u}| = 7$,
2. $(\mathbf{0}, \mathbf{v})\mathbf{P}^{-1}$ avec $|\mathbf{v}| = 7$,
3. $(\mathbf{u}, \mathbf{0})\mathbf{P}^{-1}$ avec $|\mathbf{u}| = 7$,
4. $(\mathbf{u}, \mathbf{u}+\mathbf{v})\mathbf{P}^{-1}$ avec $\mathrm{Supp}(\mathbf{u}) \subsetneq \mathrm{Supp}(\mathbf{v})$, $|\mathbf{v}| = 7$ et $1 \leqslant |\mathbf{u}| \leqslant 6$.

Le premier cas $(\mathbf{u}, \mathbf{u})$ est clairement impossible. Nous allons montrer qu'il en est de même pour le point 4. Dans ce qui suit l'ensemble $\{(\mathbf{u}_i, \mathbf{u}_i + \mathbf{v}_i)\mathbf{P}^{-1}\}_{1\leqslant i\leqslant \alpha}$ (*resp.* $\{(\mathbf{u}'_i, \mathbf{0})\mathbf{P}^{-1}\}_{1\leqslant i\leqslant \beta}$) désignera les lignes de $\mathbf{G}$ vérifiant le cas 4 (*resp.* 3) où $\alpha \in [\![0, (s+t)]\!]$ (*resp.* $\beta \in [\![0, (s+t)]\!]$). Montrons maintenant que :

$$\{\mathbf{u}_1, \cdots, \mathbf{u}_\alpha, \mathbf{u}'_1, \cdots, \mathbf{u}'_\beta\} \textit{ est une base de } U \tag{7.5}$$

Une famille génératrice. Tout les mots de codes $(\mathbf{u}, *)$ de $(U, U+V)$ pour un vecteur $\mathbf{u} \in U$ fixé peuvent être générés à l'aide de la matrice $\mathbf{G}$ par définition. Or tous les vecteurs pouvant générer un tel mot viennent d'être considérés. De cette façon, la famille de vecteurs de (7.5) est génératrice.

Une famille libre. Notons L_i et L'_j les lignes de $\mathbf{G}$ définies comme :

$$\forall i \in [\![1, \alpha]\!], \quad L_i \triangleq (\mathbf{u}_i, \mathbf{u}_i + \mathbf{v}_i)\mathbf{P}^{-1}$$

$$\forall j \in [\![1, \beta]\!], \quad L'_j \triangleq (\mathbf{u}'_j, \mathbf{0})\mathbf{P}^{-1}.$$

Remarquons maintenant que $|\mathbf{v}_i| = 7$, donc d'après le lemme 7.2, les mots de code $(\mathbf{0}, \mathbf{v}_i)\mathbf{P}^{-1}$ sont des lignes de $\mathbf{G}$. De plus, d'après le lemme 7.3, les mots de code $(\mathbf{0}, \mathbf{u}'_i)\mathbf{P}^{-1}$ sont aussi des lignes de $\mathbf{G}$. Donc pour tout i et j il existe $k_i \neq i$ et $\ell_j \neq j$ tels que

$$\forall i \in [\![1, \alpha]\!], \quad L_{k_i} = (\mathbf{0}, \mathbf{v}_i)\mathbf{P}^{-1} \neq L_i$$

$$\forall j \in [\![1, \beta]\!], \quad L_{\ell_j} = (\mathbf{0}, \mathbf{u}'_j)\mathbf{P}^{-1} \neq L_j$$

sont des lignes de $\mathbf{G}$. Notons alors $\tilde{\mathbf{G}}$ la matrice obtenue après avoir effectué les opérations linéaires qui suivent sur les lignes de $\mathbf{G}$:

$$\forall i \in [\![1, \alpha]\!],\ L_i \leftarrow L_i + L_{k_i} \quad ; \quad \forall j \in [\![1, \beta]\!],\ L_j \leftarrow L_j + L_{\ell_j}.$$

De cette façon, il y a $\alpha + \beta$ lignes de $\tilde{\mathbf{G}}$ de la forme :

$$\{(\mathbf{u}_1, \mathbf{u}_1)\mathbf{P}^{-1}, \cdots, (\mathbf{u}_\alpha, \mathbf{u}_\alpha)\mathbf{P}^{-1}, \\ (\mathbf{u}'_1, \mathbf{u}'_1)\mathbf{P}^{-1}, \cdots, (\mathbf{u}'_\beta, \mathbf{u}'_\beta)\mathbf{P}^{-1}\} \quad (7.6)$$

De plus, comme $L_{k_i} \neq L_i$ et $L_{\ell_j} \neq L_j$ pour tout i et j, nous avons

$$\mathbf{rang}(\mathbf{G}) = \mathbf{rang}(\tilde{\mathbf{G}}).$$

Or le rang $\mathbf{G}$ est donné par ses lignes $s + t > t$. Il s'ensuit que la famille de vecteurs donnée dans (7.6) est libre. Donc les mots de code $\big((\mathbf{u}_i), (\mathbf{u}'_j)\big)$ forment une famille libre ce qui prouve (7.5) avec en particulier le fait que $\alpha + \beta = \dim(U) = t$.

En utilisant la base de U donné dans (7.5) il est clair que :

$$\begin{aligned} |\operatorname{Supp}(U)| &\leqslant \sum_{i=1}^{\alpha} |\mathbf{u}_i| + \sum_{i=1}^{\beta} |\mathbf{u}'_i| \\ &\leqslant \sum_{i=1}^{\alpha} 6 + \sum_{i=1}^{\beta} 7 \\ &= 6\alpha + 7\beta \\ &= 6\alpha + 7(t - \alpha) = 7t - \alpha. \end{aligned}$$

Or comme $|\operatorname{Supp}(U)| \geqslant 7t$ nous en déduisons que $\alpha = 0$ ce qui implique qu'il n'existe pas de lignes de $\mathbf{G}$ vérifiant le point 4. Il y a donc t lignes de $\mathbf{G}$ de la forme $(\mathbf{u}, \mathbf{0})\mathbf{P}^{-1}$ où les mots de codes $\mathbf{u}$ forment une base de U. De plus, ces-dernières sont de poids de Hamming 7. Il y a donc aussi t lignes de la forme $(\mathbf{0}, \mathbf{u})\mathbf{P}^{-1}$ dans $\mathbf{G}$ d'après le lemme 7.3. Toutes les autres lignes de la matrice sont alors de la forme $(\mathbf{0}, \mathbf{v})\mathbf{P}^{-1}$ avec $|\mathbf{v}| = 7$. Le cas $\mathbf{v} \in U$ pour ces mots est impossible, autrement $(\mathbf{v}, \mathbf{0})\mathbf{P}^{-1}$ (voir le lemme 7.3) serait une ligne de $\mathbf{G}$ tandis que nous avons déjà considéré toutes les lignes de cette forme ce qui conclut la preuve. □

D'après les lemmes qui précèdent il y a t lignes $\{(\mathbf{u}_1,\mathbf{0})\mathbf{P}^{-1},\cdots,(\mathbf{u}_t,\mathbf{0})\mathbf{P}^{-1}\}$ dans $\mathbf{G}$ où les $\mathbf{u}_i$ sont de poids 7 et forment une base de U. De cette façon,

$$|\operatorname{Supp}(U)| \leqslant \sum_{i=1}^{t} |\mathbf{u}_i| = \sum_{i=1}^{t} 7 = 7t.$$

D'autre part nous avons $|\operatorname{Supp}(U)| \geqslant 7t$ ce qui implique que l'inégalité précédente est une égalité. De cette façon il est clair que les mots de code $\mathbf{u}_i$ sont de support disjoint et les $2t$ lignes de $\mathbf{G}$ (voir le lemme 7.3) :

$$\{(\mathbf{u}_i,\mathbf{0})\mathbf{P}^{-1},(\mathbf{0},\mathbf{u}_i)\mathbf{P}^{-1}\}_{1\leqslant i\leqslant t}$$

sont de support disjoint. Or rappelons que la matrice $\mathbf{G}$ est définie comme :

$$\mathbf{G} = \left(\begin{array}{c|c} \mathbf{1}_t(7)\ \ \mathbf{0}_{t\times 4(s-t)} & \mathbf{0}_{t\times(4s+3t)} \\ \hline \mathbf{0}_{s\times(4s+3t)} & \mathbf{1}_s(4)\ \ \mathbf{G}_{\text{3DM}} \end{array}\right) \in \mathbb{F}_2^{(s+t)\times 2(4s+3t)}.$$

Remarquons que la partie haute de cette matrice est formée de t lignes, il y a donc au moins t lignes de $\left(\mathbf{0}_{s\times(4s+3t)}|\ \ \mathbf{1}_s(4)\ \ \mathbf{G}_{\text{3DM}}\right)$ dont le support est disjoint. Cela donne l'existence d'une solution au problème MTD considéré dont $\mathbf{G}_{\text{3DM}}$ est la matrice d'incidence, ce qui conclut la preuve. □

Nous sommes maintenant prêt à prouver le théorème 7.1. Notre preuve utilise le caractère NP-complet du problème 7.1. Le lemme qui suit nous sera utile.

Lemme 7.5. *Soit $\mathbf{G}\in\mathbb{F}_2^{k\times n}$ et les entiers $k_U\leqslant k$, $M\leqslant n/2$. Nous avons,*

$$\begin{aligned}&\mathbf{G}\textit{ génère un code }(U,U+V)\textit{ permuté où }|\operatorname{Supp}(U)|\geqslant M\textit{ et }|\operatorname{Supp}(V)|=n/2\\ &\quad\iff \mathbf{G}\textit{ génère un code }(U,U+V)\textit{ permuté avec }|\operatorname{Supp}(V)|=n/2\\ &\qquad\qquad\textit{et le nombre de colonnes à }\mathbf{0}\textit{ dans }\mathbf{G}\textit{ est inférieur à }n/2-M.\end{aligned}$$

Démonstration du lemme 7.5.

Supposons que $\mathbf{G}\in\mathbb{F}_2^{k\times n}$ soit la matrice génératrice d'un code $(U,U+V)$ permuté. Il existe donc une matrice inversible $\mathbf{S}\in\mathbb{F}_2^{k\times k}$ et une permutation $\mathbf{P}\in\mathbb{F}_2^{n\times n}$ telles que :

$$\mathbf{SGP} = \begin{pmatrix} \mathbf{G}_U & \mathbf{G}_U \\ \mathbf{0}_{(k-k_U)\times n/2} & \mathbf{G}_V \end{pmatrix}$$

où $\mathbf{G}_U\in\mathbb{F}_2^{k_U\times n/2}$ (*resp.* $\mathbf{G}_V\in\mathbb{F}_2^{(k-k_U)\times n/2}$) est une matrice génératrice de U (*resp.* V). Supposons maintenant que $|\operatorname{Supp}(V)|=n/2$ ce qui signifie qu'il n'existe pas de colonne tout à zéro dans $\mathbf{G}_V$.

Remarquons maintenant que si $|\operatorname{Supp}(U)|<M$, alors il existe au moins $n/2-M$ colonnes qui sont égales à $\mathbf{0}$ dans la matrice $\mathbf{SGP}$. Réciproquement, s'il existe $n/2-M$ colonnes égales à $\mathbf{0}$, nous avons nécessairement $|\operatorname{Supp}(U)|<M$ du fait qu'aucune colonne de $\mathbf{G}_V$ n'est égale au vecteur tout à zéro.

La multiplication par $\mathbf{S}^{-1}$ et $\mathbf{P}^{-1}$ ne change pas le nombre de colonnes égale à $\mathbf{0}$. Nous en déduisons alors facilement la même équivalence sur $\mathbf{G}$ ce qui conclut la preuve. □

La preuve du théorème 7.1 s'ensuit facilement.

Démonstration du théorème 7.1.

Considérons une instance $(\mathbf{G},k_U,M)$ du problème 7.1. La réduction polynomiale au problème 7.1 consiste à vérifier s'il existe au plus $n/2-M$ colonnes de $\mathbf{G}$ qui sont égales à $\mathbf{0}$ puis de considérer le code de matrice génératrice $\mathbf{G}$ et l'entier k_U. Il suffit alors d'utiliser le lemme 7.5 pour conclure la preuve. □

Nous avons donc démontré que le problème de distinguer un code $(U,U+V)$ d'un code quelconque est NP-complet. Cependant, notre réduction a consisté à considérer :

$$\mathbf{G} = \left(\begin{array}{c|c} \mathbf{1}_t(7)\ \ \mathbf{0}_{t\times 4(s-t)} & \mathbf{0}_{t\times(4s+3t)} \\ \hline \mathbf{0}_{s\times(4s+3t)} & \mathbf{1}_s(4)\ \ \mathbf{G}_{\text{3DM}} \end{array}\right) \in \mathbb{F}_2^{(s+t)\times 2(4s+3t)}$$

comme candidate pour être une matrice génératrice d'un code $(U, U+V)$ où la partie gauche donne le code U. Or rappelons que

$$t \leqslant s + 1.$$

Notre réduction se fait donc pour des codes $(U, U+V)$ où $\dim(U) < \dim(V)$. Cette condition semble d'autant plus importante avec notre méthode de réduction que nous pouvons prouver une version "plus faible" de notre résultat. Considérons le problème où la question est de savoir si étant donné un code $\mathcal{C}$, ce dernier est un code $(U, U+V)$ permuté où la permutation n'agirait que sur la partie droite. La démonstration de ce résultat a été obtenue après réduction du problème de sous-code où U joue le rôle de sous-code de V. Commençons par rappeler ce problème de sous-code.

Problème 7.3 (Équivalence de sous-code)**.**

— Instance : *deux codes linéaires $\mathcal{C}$ et $\mathcal{D}$ de longueur n,*

— Décision : *il existe une permutation π sur les mots de longueur n telle que $\sigma(\mathcal{C}) \subseteq \mathcal{D}$.*

Ce problème a été prouvé NP-complet dans [BGK17]. Nous allons le réduire facilement au problème suivant.

Problème 7.4 (Une version faible du problème de distinguer un code $(U, U+V)$)**.**

— Instance : *Un code linéaire $\mathcal{C}$ de longueur paire n,*

— Décision : *Existe t-il deux codes U, V de longueur $n/2$ et une permutation π telles que :*

$$(U, U+V) = \{(\mathbf{x}, \pi(\mathbf{y})) \ : \ (\mathbf{x}, \mathbf{y}) \in \mathcal{C}, \ \mathbf{x} \in \mathbb{F}_2^{n/2}, \mathbf{y} \in \mathbb{F}_2^{n/2}\}.$$

Le théorème qui suit sera prouvée par une réduction au problème du sous-code.

Théorème 7.2. *Le problème 7.4 est* NP*-complet.*

Démonstration de la proposition 7.2.

Considérons une instance $(\mathcal{C}, \mathcal{D})$du problème d'équivalence de sous-codes et définissons le code :

$$\mathcal{C} \times \mathcal{D} \stackrel{\triangle}{=} \{(\mathbf{c}, \mathbf{d}) : \mathbf{c} \in \mathcal{C} \text{ et } \mathbf{d} \in \mathcal{D}\}.$$

Ce code forme alors une instance du problème 7.4. Ce dernier s'obtient en temps polynomial à partir d'une matrice génératrice de $\mathcal{C}$ et $\mathcal{D}$.

Supposons qu'il existe une permutation π telle que $\pi(\mathcal{C}) \subseteq \mathcal{D} \iff \mathcal{C} \subseteq \pi^{-1}(\mathcal{D})$. Alors le code,

$$\mathcal{C} \times \pi^{-1}(\mathcal{D}) \stackrel{\triangle}{=} \left\{(\mathbf{c}, \pi^{-1}(\mathbf{d})) : \mathbf{c} \in \mathcal{C} \text{ et } \mathbf{d} \in \mathcal{D}\right\} \tag{7.7}$$

est un code $(U, U+V)$ où $\mathcal{C}$ joue le rôle de U. En effet, $\mathcal{C} \subseteq \pi^{-1}(\mathcal{D})$ et donc le code $\mathcal{C} \times \pi^{-1}(\mathcal{D})$ contient tous les mots de la forme $(\mathbf{c}, \mathbf{c})$ pour $\mathbf{c} \in \mathcal{C}$. Ces mots avec ceux de la forme $(\mathbf{0}, \pi{-}1(\mathbf{d}))$ engendre le code $\mathcal{C} \times \pi^{-1}(\mathcal{D})$ ce qui donne bien le résultat.

Réciproquement, supposons que le code $\mathcal{C} \times \mathcal{D}$ est une instance admettant une solution du problème 7.4, *i.e* : il existe une permutation telle que,

$$\mathcal{C} \times \pi(\mathcal{D}) \stackrel{\triangle}{=} \{(\mathbf{c}, \pi(\mathbf{d})) : \mathbf{c} \in \mathcal{C} \text{ et } \mathbf{d} \in \mathcal{D}\}$$

est un code $(U, U+V)$. Le code $\mathcal{C}$ joue le rôle de U qui est un sous-code de $U+V = \pi(\mathcal{D})$, d'où $\pi^{-1}(\mathcal{C}) \subseteq \mathcal{D}$ ce qui conclut la preuve.

□

Dans le schéma de signature Wave il est nécessaire d'avoir $\dim U > \dim V$ pour avoir un avantage lors du décodage. Notre réduction n'est donc pas dans l'état actuel de bonne qualité. Nous avons cependant espoir de l'améliorer en tenant compte cette fois de la liberté supplémentaire qu'offre les codes $(U, U+V)$-généralisés avec le choix des vecteurs $\mathbf{a}, \mathbf{b}, \mathbf{c}$ et $\mathbf{d}$.

7.2 Une recherche de mots aux propriétés particulières.

Nous proposons dans cette section un algorithme pour distinguer un code $(U, U+V)$-généralisé permuté d'un code aléatoire. Ce dernier, que nous présenterons dans §7.2.2, repose sur le résultat suivant que nous démontrons dans la prochaine sous-section. Bien qu'un code $(U, U+V)$-généralisé semble proche d'un code aléatoire lorsque les codes U et V sont eux-mêmes aléatoires, il existe une différence dans la distribution de poids. Cet "écart" subsistant après permutation des coordonnées provient de l'existence de mots au poids anormalement élevé ou faible et étant de la forme $(\mathbf{a}\odot\mathbf{u}, \mathbf{c}\odot\mathbf{u})$ avec $\mathbf{u}\in U$ ou $(\mathbf{b}\odot\mathbf{v}, \mathbf{d}\odot\mathbf{v})$ tel que $\mathbf{v}\in V$.

7.2.1 Distribution de poids des codes $(U, U+V)$-généralisés

La proposition qui suit détermine la distribution de poids attendue d'un code $(U, U+V)$-généralisé aléatoire (voir la définition 5.7 du nombre de blocs de type I).

Proposition 7.2. *Supposons que l'on choisisse uniformément un code $(U, U+V)$-généralisé avec un nombre de blocs n_I de type I. Soient $a_{(\mathbf{u},\mathbf{v})}(z)$, $a_{(\mathbf{u},\mathbf{0})}(z)$ et $a_{(\mathbf{0},\mathbf{v})}(z)$ l'espérance du nombre de mots de poids z qui sont respectivement dans ce code de la forme $(\mathbf{a}\odot\mathbf{u}+\mathbf{b}\odot\mathbf{v}, \mathbf{c}\odot\mathbf{u}+\mathbf{d}\odot\mathbf{v})$, $(\mathbf{a}\odot\mathbf{u}, \mathbf{c}\odot\mathbf{u})$ et $(\mathbf{b}\odot\mathbf{v}, \mathbf{d}\odot\mathbf{v})$ où $\mathbf{u}\in U$ et $\mathbf{v}\in V$. Ces quantités sont alors données pour tout entier z pair de $[\![0,n]\!]$ par*

$$a_{(\mathbf{u},\mathbf{0})}(z) = \frac{\binom{n/2}{z/2}2^{z/2}}{3^{n/2-k_U}}, \quad a_{(\mathbf{0},\mathbf{v})}(z) = \frac{1}{3^{n/2-k_V}} \sum_{\substack{j=0\\ j \text{ pair}}}^{z} \binom{n_I}{j}\binom{n/2-n_I}{\frac{z-j}{2}}2^{(z+j)/2}$$

$$a_{(\mathbf{u},\mathbf{v})}(z) = a_{(\mathbf{u},\mathbf{0})}(z) + a_{(\mathbf{0},\mathbf{v})}(z) + \frac{1}{3^{n-k_U-k_V}}\left(\binom{n}{z}2^z - \binom{n/2}{z/2}2^{z/2} - \sum_{\substack{j=0\\ j \text{ pair}}}^{z}\binom{n_I}{j}\binom{n/2-n_I}{\frac{z-j}{2}}2^{(z+j)/2}\right)$$

et pour les entiers impairs $z\in[\![0,n]\!]$ par :

$$a_{(\mathbf{u},\mathbf{0})}(z) = 0, \quad a_{(\mathbf{0},\mathbf{v})}(z) = \frac{1}{3^{n/2-k_V}} \sum_{\substack{j=0\\ j \text{ impair}}}^{z} \binom{n_I}{j}\binom{n/2-n_I}{\frac{z-j}{2}}2^{(z+j)/2}$$

$$a_{(\mathbf{u},\mathbf{v})}(z) = a_{(\mathbf{0},\mathbf{v})}(z) + \frac{1}{3^{n-k_U-k_V}}\left(\binom{n}{z}2^z - \sum_{\substack{j=0\\ j \text{ impair}}}^{z}\binom{n_I}{j}\binom{n/2-n_I}{\frac{z-j}{2}}2^{(z+j)/2}\right).$$

D'autre part, lorsque nous choisissons aléatoirement un code de longueur n sur $\mathbb{F}_3$ et de dimension k_U+k_V, $a(z)$ le nombre attendu de mots de code de poids z est donné par :

$$a(z) = \frac{\binom{n}{z}2^z}{3^{n-k_U-k_V}}.$$

Démonstration de la proposition 7.2.

On déduit directement le résultat de la dernière partie de la proposition à l'aide du lemme 1.3.

Prouvons la première partie de la proposition 7.2 donnant la distribution de poids d'un code $(U, U+V)$-généralisé, à savoir un code de la forme :

$$(\mathbf{a}\odot U+\mathbf{b}\odot V, \mathbf{c}\odot U+\mathbf{d}\odot V) \stackrel{\triangle}{=} \{(\mathbf{a}\odot\mathbf{u}+\mathbf{b}\odot\mathbf{v}, \mathbf{c}\odot\mathbf{u}+\mathbf{d}\odot\mathbf{v})\}.$$

Dans la suite nous désignerons par $\mathcal{C}$ un tel code.

Distribution de poids de $(\mathbf{a} \odot \boldsymbol{U}, \mathbf{c} \odot \boldsymbol{U}) \stackrel{\triangle}{=} \{(\mathbf{a} \odot \mathbf{u}, \mathbf{c} \odot \mathbf{u}) : \mathbf{u} \in \boldsymbol{U}\}$ et $(\mathbf{b} \odot \boldsymbol{V}, \mathbf{d} \odot \boldsymbol{V}) \stackrel{\triangle}{=} \{(\mathbf{b} \odot \mathbf{v}, \mathbf{d} \odot \mathbf{v}) : \mathbf{v} \in \boldsymbol{V}\}$. Commençons tout d'abord par rappeler que par définition d'un code $(U, U+V)$-généralisé nous avons pour tout $i \in [\![1, n/2]\!]$,

$$a_i c_i \neq 0$$

Il s'ensuit alors directement d'après le lemme 1.3 que pour tout z

$$a_{(\mathbf{u},\mathbf{0})}(z) = \begin{cases} \frac{\binom{n/2}{z/2} 2^{z/2}}{3^{n/2-k_U}} & \text{si } z \text{ est pair} \\ 0 & \text{sinon.} \end{cases}$$

La distribution de poids des mots $(\mathbf{b} \odot \mathbf{v}, \mathbf{d} \odot \mathbf{v})$ pour $\mathbf{v} \in V$ est quant à elle légèrement plus sophistiquée. Cette-dernière dépend du nombre de blocs n_I de type I qui rappelons-le est défini comme :

$$n_I = |\{1 \leqslant i \leqslant n/2 : b_i d_i = 0\}| .$$

où par définition d'un code $(U, U+V)$-généralisé nous ne pouvons pas avoir simultanément $b_i = 0$ et $d_i = 0$ ($a_i d_i - b_i c_i \neq 0$). De cette façon, $a_{(\mathbf{0},\mathbf{v})}(z)$ est égal à la somme sur $j \in [\![1, n_I]\!]$ du nombre attendu de mots de poids $j + \frac{z-j}{2}$ d'un code aléatoire de longueur $n/2$ et de dimension k_V. Ici j compte le nombre de positions non-nulles parmi les n_I coordonnées apparaissant une fois dans le code $(U, U+V)$-généralisés considéré. Il est donc clair d'après le lemme 1.3 que pour les entiers z pairs :

$$a_{(\mathbf{0},\mathbf{v})}(z) = \frac{1}{3^{n/2-k_V}} \sum_{\substack{j=0 \\ j \text{ pair}}}^{z} \binom{n_I}{j} \binom{n/2 - n_I}{\frac{z-j}{2}} 2^{(z+j)/2}$$

alors que pour les z impairs la somme se fait sur les j impairs.

Distribution de poids de $\mathcal{C}$. Le code $(U, U+V)$-généralisé est uniformément distribué dans son domaine, c'est à dire que les matrices de parité $\mathbf{H}_U$ du code U et $\mathbf{H}_V$ du code V sont uniformément distribuées sur $\mathbb{F}_3^{(n/2-k_U)\times n/2}$ et $\mathbb{F}_3^{(n/2-k_V)\times n/2}$. Notons $Z \stackrel{\triangle}{=} \sum_{\mathbf{x} \in \mathbb{F}_3^n : |\mathbf{x}| = z} Z_{\mathbf{x}}$ où $Z_{\mathbf{x}}$ est la variable aléatoire indicatrice de l'évènement "$\mathbf{x} \in \mathcal{C}$". De cette façon,

$$\begin{aligned} a_{(\mathbf{u},\mathbf{v})}(z) &= \mathbb{E}(Z) \\ &= \sum_{\mathbf{x} \in \mathbb{F}_3^n : |\mathbf{x}| = z} \mathbb{P}(\mathbf{x} \in \mathcal{C}) \end{aligned} \tag{7.8}$$

Or d'après la proposition 5.4 nous avons,

$$\mathbf{x} \in \mathcal{C} \iff \mathbf{x}_U \mathbf{H}_U^\intercal = \mathbf{0} \quad \text{et} \quad \mathbf{x}_V \mathbf{H}_V^\intercal = \mathbf{0}.$$

Il y a donc trois cas à considérer (pour tout $1 \leqslant i \leqslant n/2$ nous avons $a_i d_i - b_i c_i \neq 0$) où à chaque fois nous appliquons le lemme 1.3,

Cas 1 : $\mathbf{x}_U = \mathbf{0}$ et $\mathbf{x}_V \neq \mathbf{0}$,

$$\mathbb{P}(\mathbf{x} \in \mathcal{C}) = \mathbb{P}(\mathbf{x}_V \mathbf{H}_V^\intercal = \mathbf{0}) = \frac{1}{3^{n/2-k_V}}$$

Cas 2 : $\mathbf{x}_U \neq \mathbf{0}$ et $\mathbf{x}_V = \mathbf{0}$,

$$\mathbb{P}(\mathbf{x} \in \mathcal{C}) = \mathbb{P}(\mathbf{x}_U \mathbf{H}_U^\intercal = \mathbf{0}) = \frac{1}{3^{n/2-k_U}}$$

Cas 3 : $\mathbf{x}_U \neq \mathbf{0}$ et $\mathbf{x}_V \neq \mathbf{0}$,

$$\mathbb{P}(\mathbf{x} \in \mathcal{C}) = \mathbb{P}(\mathbf{x}_V \mathbf{H}_V^\intercal = \mathbf{0}, \mathbf{x}_U \mathbf{H}_U^\intercal = \mathbf{0}) = \frac{1}{3^{n/2-k_U}} \frac{1}{3^{n/2-k_V}}$$

Nous pouvons alors conclure la preuve de la proposition. Il suffit de remplacer $\mathbb{P}(\mathbf{x} \in \mathcal{C})$ dans l'équation (7.8) et d'utiliser la définition du nombre de blocs de type I. □

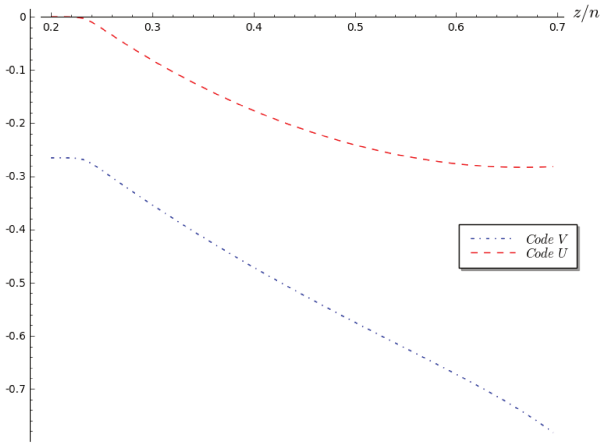


Figure 7.1 – $\alpha_{\mathbf{u}}(z/n)$ et $\alpha_{\mathbf{v}}(z/n)$ en fonction de $x \triangleq \frac{z}{n}$.

Cette proposition montre donc bien une différence entre la distribution de poids d'un code $(U, U+V)$-généralisé et d'un code aléatoire. Cela provient des mots où $\mathbf{u} = \mathbf{0}$ ou $\mathbf{v} = \mathbf{0}$. Définissons alors le logarithme normalisé des densités de mots de la forme $(\mathbf{a} \odot \mathbf{u}, \mathbf{c} \odot \mathbf{u})$ et $(\mathbf{b} \odot \mathbf{v}, \mathbf{d} \odot \mathbf{v})$ en fonction du poids relatif $x \triangleq \frac{z}{n}$ où z est *pair* :

$$\alpha_{\mathbf{u}}(z/n) \triangleq \frac{\log_2(a_{(\mathbf{u},\mathbf{0})}(z)/a_{(\mathbf{u},\mathbf{v})}(z))}{n} \quad \text{et} \quad \alpha_{\mathbf{v}}(z/n) \triangleq \frac{\log_2(a_{(\mathbf{0},\mathbf{v})}(z)/a_{(\mathbf{u},\mathbf{v})}(z))}{n}.$$

Nous avons tracé dans la figure 7.1 ces deux quantités en fonction du poids relatif x dans le cas où U est de rendement $\frac{k_U}{n/2} = 0.7$, V est de rendement $\frac{k_V}{n/2} = 0.3$ et $\frac{n_I}{n/2} = \frac{1}{2}$. Comme nous pouvons l'observer, pour des poids relatifs inférieurs approximativement à 0.26, presque tous les mots du code $(U, U+V)$-généralisé sont de la forme $(\mathbf{a} \odot \mathbf{u}, \mathbf{c} \odot \mathbf{u})$.

Ces considérations nous ont mené à proposer un algorithme pour distinguer un code $(U, U+V)$-généralisé permuté (une clef publique de la famille de fonctions Wave) d'un code aléatoire. Notre algorithme consiste essentiellement à utiliser des algorithmes permettant de trouver des mots de poids faible dans un code (typiquement les ISD). Le rationnel de cette approche est que la densité des vecteurs $(\mathbf{a} \odot \mathbf{u}, \mathbf{c} \odot \mathbf{u})$ ou $(\mathbf{b} \odot \mathbf{v}, \mathbf{d} \odot \mathbf{v})$ révélant la structure augmente lorsque le poids diminue. Nous faisons alors tourner ces algorithmes jusqu'à trouver un mot de la forme (permuté) $(\mathbf{a} \odot \mathbf{u}, \mathbf{c} \odot \mathbf{u})$ ou $(\mathbf{b} \odot \mathbf{v}, \mathbf{d} \odot \mathbf{v})$ avec $\mathbf{u} \in U$ et $\mathbf{v} \in V$. L'idée étant que si nous trouvons un de ces mots particuliers, alors nous pouvons en trouver d'autres beaucoup plus rapidement et donc en déduire qu'il s'agit de mots anormaux comme cela fut fait dans l'attaque contre la signature KKS [KKS97 ; KKS05] dans [OT11, §4.4]. L'algorithme que nous allons présenter est dans l'esprit similaire à cette dernière. D'une certaine façon, l'attaque de [OT11] permet de retrouver le support du code V. En revanche, dans le schéma KKS le suport de ce code est bien plus petit que dans notre cas. De plus, comme expliqué dans la conclusion de[OT11], l'attaque contre KKS est de nature exponentielle. Il s'avèrera comme nous allons le voir que les paramètres des codes $(U, U+V)$-généralisés que nous considèrerons font que l'exposant asymptotique de l'attaque contre KKS est assez élevé.

La notion de code poinçonné sera utile dans toute la suite. Rappelons que pour un code $\mathcal{C}$ de longueur n et $\mathcal{I} \subseteq [\![1, n/2]\!]$, le code $\mathcal{C}$ poinçonné sur les positions $\mathcal{I}$ est défini comme :

$$\underset{\mathcal{I}}{\mathrm{Poinc}}(\mathcal{C}) \triangleq \{\mathbf{c}_{\overline{\mathcal{I}}} = (c_j)_{j \in [\![1,n]\!] \setminus \mathcal{I}} : \mathbf{c} \in \mathcal{C}\} \tag{7.9}$$

7.2.2 Un algorithme de recherche des mots particuliers

7.2.2.1 Retrouver le code U à une permutation près

Nous considèrerons dans cette section le code permuté :

$$U' \triangleq (\mathbf{a} \odot U, \mathbf{c} \odot U)\mathbf{P} = \{(\mathbf{a} \odot \mathbf{u}, \mathbf{c} \odot \mathbf{u})\mathbf{P} : \mathbf{u} \in U\}.$$

Notre attaque va consister à retrouver une base de U' à partir du code $(U, U+V)$-généralisé permuté selon $\mathbf{P}$. Une fois que cela est fait nous retrouvons facilement les paires de positions $(i, i+n/2)$ permuté selon $\mathbf{P}$. En effet, il suffit de considérer toutes les paires de coordonnées et de ne que conserver que celles toujours égales ou distinctes.

La procédure que nous proposons pour retrouver U' est donné dans l'algorithme 7.

Algorithme 7 CALCULU$(p, \ell, \mathcal{C}_{\text{pk}}, N)$ où $\mathcal{C}_{\text{pk}}$ est le code publique utilisé pour la signature et N le nombre d'itérations de l'algorithme

Ensure: B contient des vecteurs de U'

for $i = 1, \ldots, N$ **do**
  $B \leftarrow \varnothing$
  On choisit uniformément $\mathcal{I} \subset [\![1, n]\!]$ de taille $n - k - \ell$
  $\mathcal{L} \leftarrow$ MOTSDECODE$(\text{Poinc}_{\mathcal{I}}(\mathcal{C}_{\text{pk}}), p)$
  for all $\mathbf{x} \in \mathcal{L}$ **do**
    $\mathbf{x} \leftarrow$ COMPLÈTE$(\mathbf{x}, \mathcal{I}, \mathcal{C}_{\text{pk}})$
    if VÉRIFIEU$(\mathbf{x})$ **then**
      on ajoute $\mathbf{x}$ à B
return B

Cet algorithme utilise les fonctions auxiliaires qui suivent.

— MOTSDECODE$(\text{Poinc}_{\mathcal{I}}(\mathcal{C}_{\text{pk}}), p)$ calcule tous les mots de code (ou une fraction constante) de poids p du code poinçonné $\text{Poinc}_{\mathcal{I}}(\mathcal{C}_{\text{pk}})$ déduit du code public. Tous les algorithmes ISD [Dum91 ; FS09 ; MMT11 ; Bec+12 ; MO15] utilisent en sous-routine un tel algorithme,

— COMPLÈTE$(\mathbf{x}, \mathcal{I}, \mathcal{C}_{\text{pk}})$ calcule le mot $\mathbf{c} \in \mathcal{C}_{\text{pk}}$ tel que $\mathbf{c}_{\overline{\mathcal{I}}} = \mathbf{x}$,

— VÉRIFIEU$(\mathbf{x})$ vérifie si $\mathbf{x}$ appartient à U'.

Choisir N de façon appropriée. Commençons notre étude par déterminer la valeur de N pour que CALCULU retourne $\Omega(1)$ éléments. Il s'agit essentiellement de l'analyse faite dans [OT11, §5.2].

Proposition 7.3. *La probabilité P_{succ} qu'une itération de la boucle (instruction 2) de* CALCULU *ajoute un élément à liste B est minorée par :*

$$P_{succ} \geqslant \sum_{z=0}^{n/2} \frac{\binom{n/2}{z}\binom{n/2-z}{k+\ell-2z} 2^{k+\ell-2z}}{\binom{n}{k+\ell}} f\left(\frac{\binom{k+\ell-2z}{p-2i}\binom{z}{i}2^{p-i}}{3^{\max(0,k+\ell-z-k_U)}}\right) \tag{7.10}$$

où f est la fonction définie comme

$$f(x) \triangleq \max\left(x(1-x/2), 1 - \frac{1}{x}\right).$$

L'algorithme 7 renvoie une liste non vide avec une probabilité $\Omega(1)$ dès-lors que N est choisi comme : $N = \Omega\left(\frac{1}{P_{succ}}\right)$.

Démonstration de la proposition 7.3.

Il sera ici utile de rappeler [OT11, Lemma 3].

Lemme 7.6. *Notons $\mathcal{C}_{rand}$ le code de matrice de parité $\mathbf{H}$ uniformément distribuée sur $\mathbb{F}_3^{r\times n}$. Fixons X un sous-ensemble de $\mathbb{F}_3^n$ de taille m. Nous avons*

$$\mathbb{P}(X \cap \mathcal{C}_{rand} \neq \varnothing) \geqslant f\left(\frac{m}{3^r}\right).$$

Nous dirons dans ce qui suit d'une paire de deux positions $i \neq j$ qu'elle est *assortie* (pour U') si et seulement si :

$$\exists \lambda \in \{-1, 1\} \quad : \quad \forall \mathbf{c} \in U', \quad c_i = \lambda c_j.$$

Le code considéré étant un $(U, U+V)$-généralisé, il existe par définition $n/2$ paires assorties ($a_i c_i \neq 0$ pour tout i). Notons dans ce qui suit la variable aléatoire :

$$Z \triangleq \#\left\{i, j \in [\![1, n]\!] \backslash \mathcal{I} : (i, j) \text{ paire assortie}\right\}$$

où $\mathcal{I}$ est l'ensemble aléatoire de taille $n-k-\ell$ tiré dans l'instruction 4 de l'algorithme 7.

Calculons la probabilité de succès P_{succ} en conditionnant sur les valeurs possibles de Z :

$$P_{\text{succ}} = \sum_{z=0}^{n/2} \mathbb{P}(Z = z)\mathbb{P}\left(\exists \mathbf{x} \in U' : |\mathbf{x}_{\bar{\mathcal{I}}}| = p \,\middle|\, Z = z\right) \tag{7.11}$$

Partitionnons maintenant $\bar{\mathcal{I}}$ comme

$$\bar{\mathcal{I}} = \mathcal{J}_1 \cup \mathcal{J}_2$$

où $\mathcal{J}_2$ est l'ensemble des positions donnant des paires assorties dans $\bar{\mathcal{I}}$. Notons que $|\mathcal{J}_2| = 2z$. Partitionnons désormais $\mathcal{J}_2$ comme :

$$\mathcal{J}_2 = \mathcal{J}_{21} \cup \mathcal{J}_{22}$$

où aucune paire d'éléments de $\mathcal{J}_{21}$ et $\mathcal{J}_{22}$ ne donne une paire assortie. Considérons maintenant les codes :

$$U'' \triangleq \underset{\mathcal{I}}{\text{Poinc}}(U') \quad \text{et} \quad U''' \triangleq \underset{\mathcal{I}\cup\mathcal{J}_{22}}{\text{Poinc}}(U').$$

Le code U''' est de longueur $n-(n-k-\ell+z) = k+\ell-z$ du fait que $|\mathcal{J}_{22}| = z$ et $|\mathcal{I}| = n-k-\ell$. Le rationnel derrière la définition du code U'' est que :

$$\mathbb{P}\left(\exists \mathbf{x} \in U' : |\mathbf{x}_{\bar{\mathcal{I}}}| = p \mid Z = z\right) = \mathbb{P}\left(\exists \mathbf{x} \in U'' : |\mathbf{x}| = p \mid Z = z\right).$$

Le problème étant maintenant que nous souhaiterions appliquer le lemme 7.6. Cela n'est malheureusement pas possible du fait que U'' contient des paires assorties (le code n'est pas aléatoire). C'est alors précisément la raison de l'introduction du code U'''. Ce dernier est en effet aléatoire de matrice de parité uniformément distribuée sur les matrices ternaires de taille $\max(0, k+\ell-z-k_U) \times (k+\ell-z)$. Nous pouvons donc lui appliquer le lemme 7.6. Cependant nous devons ici être précautionneux, les mots de poids p dans U'' n'ont pas le même probabilité d'apparition que dans U''' du fait qu'il existe des positions donnant des paires assorties. C'est la raison de notre introduction pour $i \in [\![0, \lfloor p/2 \rfloor]\!]$ des ensembles X_i définis comme :

$$X_i \triangleq \left\{\mathbf{x} = (x_i)_{i\in\bar{\mathcal{I}}\backslash\mathcal{J}_{22}} \in \mathbb{F}_3^{k+\ell-z} : |\mathbf{x}_{\mathcal{J}_1}| = p-2i,\ |\mathbf{x}_{\mathcal{J}_{21}}| = i\right\}.$$

Un mot de code de poids p dans U'' correspond alors après poinçonnage sur $\mathcal{J}_{22}$ à un élément de X_i. Nous en déduisons alors facilement la borne inférieure :

$$\mathbb{P}\left(\exists \mathbf{x} \in U' : |\mathbf{x}_{\bar{\mathcal{I}}}| = p \mid Z = z\right) \geqslant \max_{i=0}^{\lfloor p/2 \rfloor} \left(\mathbb{P}(X_i \cap U''' \neq \varnothing)\right). \tag{7.12}$$

Donc en appliquant le lemme 7.6 nous obtenons :

$$\mathbb{P}(X_i \cap U''' \neq \varnothing) \geqslant f\left(\frac{\binom{k+\ell-2z}{p-2i}\binom{z}{i}2^{p-i}}{3^{\max(0,k+\ell-z-k_U)}}\right). \tag{7.13}$$

D'autre part, notons que

$$\mathbb{P}(Z=z) = \frac{\binom{n/2}{z}\binom{n/2-z}{k+\ell-2z}2^{k+\ell-2z}}{\binom{n}{k+\ell}}$$

ce qui permet avec (7.11) de conclure la preuve. □

Complexité pour retrouver U à une permutation près. La complexité d'un appel à CALCULU peut-être estimée de la façon suivante. Notons $C_1(p,k,\ell)$ le temps de calcul de la liste des mots de poids p d'un code de longueur $k+\ell$ et de dimension k. Nous proposerons dans §7.2.2.4 d'utiliser l'algorithme de Dumer que nous avons décrit dans §2.2.1. Notre analyse se généralise avec des algorithmes plus performants comme [MMT11 ; Bec+12 ; MO15 ; BM17]. De cette façon, $C_1(p,k,\ell)$ donne la complexité de l'appel à MOTSDECODE$(\mathrm{Poinc}_{\mathcal{I}}(\mathcal{C}_{\mathrm{pk}}),p)$ lors de l'étape 4 de l'algorithme 7. L'appel à la fonction COMPLÈTE est un simple produit matrice-vecteur. Quelle que soit la façon dont la fonction VÉRIFIEU est instanciée, la complexité de CALCULU pour retrouver le code U permuté est clairement minorée par $\Omega\left(\frac{C_1(p,k,\ell)}{P_{\mathrm{succ}}}\right)$. Il s'avère même que nous pouvons montrer que cette-dernière est de l'ordre de $\Theta\left(\frac{C_1(p,k,\ell)}{P_{\mathrm{succ}}}\right)$. Cela repose sur une combinaison de deux techniques (voir [OT11]) que nous décrivons ici succinctement.

— Une fois qu'a été identifié un élément non-nul de U', il devient plus facile d'en trouver d'autres. Il suffit d'appliquer l'astuce décrite dans [OT11, §4.4] pour casser la signature KKS. L'idée est la suivante. Si nous faisons un nouvel appel à l'algorithme CALCULU mais cette fois-ci en choisissant l'ensemble $\mathcal{I}$ sur lequel nous poinçonnons le code tel qu'il ne contienne pas le support du mot de U' trouvé alors le nombre d'itérations N nécessaire pour trouver un nouvel élément de U' est négligeable en comparaison de la valeur de N d'origine.

— La fonction VÉRIFIEU peut être instanciée de façon à ce que sa complexité soit de l'ordre du nombre N d'itérations de l'algorithme. La stratégie pour cela diffère selon les valeurs de k et k_U. Pour certains paramètres il suffit tout simplement de vérifier si les mots trouvés sont de poids atypique. En revanche, dans d'autres cas nous pouvons procéder de la façon suivante. Nous fixons un seuil et si le poids de $\mathbf{x}$ est en dessous nous décidons qu'il s'agit d'un candidat potentiel. L'astuce est alors pour ces candidats d'appliquer le point qui précède. De cette façon si nous trouvons d'autres solutions plus rapidement c'est que le vecteur $\mathbf{x}$ était un élément de U'.

7.2.2.2 Retrouver le code V à une permutation près.

Nous considérons maintenant le code permuté,

$$V' \stackrel{\triangle}{=} (\mathbf{b}\odot V, \mathbf{d}\odot V)\mathbf{P} = \{(\mathbf{b}\odot \mathbf{v}, \mathbf{d}\odot \mathbf{v})\mathbf{P} \text{ où } \mathbf{v}\in V\}.$$

L'attaque dans ce cas consiste à retrouver une base de V'. Une fois que cela est fait, nous retrouvons facilement le support $\mathrm{Supp}(V')$ du code V' défini comme

$$\mathrm{Supp}(V') \stackrel{\triangle}{=} \left\{i\in[\![1,n]\!] : \exists \mathbf{v}'\in V',\ v'_i\neq 0\right\}.$$

Cela permet ainsi de retrouver le code V à une permutation près. L'algorithme pour retrouver V' est le même au nom près : nous remplaçons le terme CALCULU par CALCULV. Nous faisons ici cette simple modification cosmétique car selon que l'on cherche U' ou V' le nombre d'itérations N n'est pas le même.

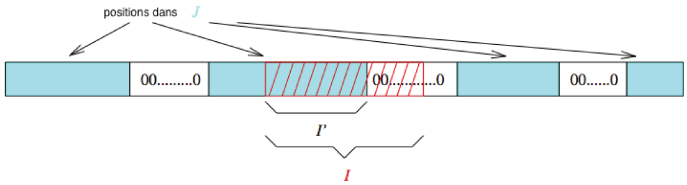


Figure 7.2 – Une figure représentant $\mathcal{J}$, $\mathcal{I}$ et $\mathcal{I}'$ en fonction de la forme d'un mot de code dans V'.

Choisir N de façon appropriée. Analysons comme dans la sous-section précédente comment N doit être choisi pour qu'un appel réussi à CALCULV renvoie $\Omega(1)$ éléments de V'. Nous avons ici le résultat suivant.

Proposition 7.4. *La probabilité P_{succ} qu'une itération de la boucle (instruction 2) de* CALCULV *ajoute un élément à la liste B est minorée par :*

$$P_{succ} \geqslant \sum_{z=0}^{\min(n-k-\ell,n-n_I)} \sum_{m=0}^{n/2-n_I} \frac{\binom{\frac{n}{2}-n_I}{m}\binom{n_I}{n-k-\ell-z}}{\binom{n}{n-k-\ell}} \max_{i=0}^{\lfloor p/2 \rfloor} f\left(\frac{\binom{n-n_I-z-2m}{p-2i}\binom{m}{i}2^{p-i}}{3^{\max(0,n-n_I-z-m-k_V)}}\right) \sum_{j=0}^{n/2-n_I-m} \binom{n/2-n_I-m}{j} 2^j \binom{n_I}{z-n+2n_I+2m+j}$$

où f est la fonction définie comme

$$f(x) \triangleq \max\left(x(1-x/2), 1-\frac{1}{x}\right).$$

L'algorithme CALCULV *renvoie une liste non vide avec une probabilité $\Omega(1)$ dès-lors que N est choisi comme : $N = \Omega\left(\frac{1}{P_{succ}}\right)$.*

Démonstration de la proposition 7.4.

Nous avons dans le code V', $\frac{n}{2} - n_I$ paires assorties (i,j) (il existe $\lambda \in \{-1,1\}$ tel que $c_i = \lambda c_j$ pour tout $\mathbf{c} \in V'$). Définissons l'ensemble $\mathcal{J}$ des images par $\mathbf{P}$ des positions $1 \leqslant i \leqslant n/2$ telles que $b_i \neq 0$ et des coordonnées $n/2 + j$ avec $0 \leqslant j \leqslant n/2$ telles que $d_j \neq 0$.

Remarque 7.2. D'après la définition 5.7 il s'ensuit que $|\mathcal{J}| = n - n_I$.

Introduisons maintenant les variables aléatoires,

$$\mathcal{I}' \triangleq \mathcal{I} \cap \mathcal{J}, \quad Z \triangleq |\mathcal{I}'| \quad \text{et} \quad M \triangleq \#\left\{i,j \in \mathcal{J}\backslash\mathcal{I}' : (i,j)\text{paire assortie}\right\}.$$

L'ensemble $\mathcal{J}\backslash\mathcal{I}'$ représente les positions qui ne sont pas nécessairement égales à 0 dans le code poinçonné $\mathrm{Poinc}_{\mathcal{I}}(V')$ (voir la figure 7.2). L'algorithme CALCULV renvoie par définition au moins un élément de V' s'il existe un vecteur de poids p dans $\mathrm{Poinc}_{\mathcal{I}'}(V')$. La probabilité de succès P_{succ} est donc donnée par :

$$P_{\text{succ}} = \sum_{z=0}^{\min(n-k-\ell,n-n_I)} \sum_{m=0}^{n/2-n_I} \mathbb{P}\left(\exists \mathbf{x} \in V' : |\mathbf{x}_{\mathcal{J}'}| = p \mid Z = z, M = m\right) \times \mathbb{P}(Z = z, M = m) \quad (7.14)$$

où

$$\mathcal{J}' \triangleq \mathcal{J}\backslash\mathcal{I}'.$$

Partitionnons maintenant $\mathcal{J}'$ comme :

$$\mathcal{J}' = \mathcal{J}_1 \cup \mathcal{J}_2$$

où $\mathcal{J}_2$ est l'ensemble des positions donnant des paires assorties dans $\mathcal{J}'$. Notons que $|\mathcal{J}_2| = 2m$. Partitionnons désormais $\mathcal{J}_2$ comme :

$$\mathcal{J}_2 = \mathcal{J}_{21} \cup \mathcal{J}_{22}$$

où aucune paire d'élément de $\mathcal{J}_{21}$ et $\mathcal{J}_{22}$ ne donne une paire assortie. Considérons maintenant les deux codes :

$$V'' \triangleq \underset{\mathcal{I}\cup\bar{\mathcal{J}}}{\text{Poinc}}(V') \quad \text{et} \quad V''' \triangleq \underset{\mathcal{I}\cup\bar{\mathcal{J}}\cup\mathcal{J}_{22}}{\text{Poinc}}(V').$$

Le code V'' est de longueur $n-n_I-z$ tandis que le second est de longueur $n-n_I-z-m$. Le rationnel derrière l'introduction du code V'' est le fait que :

$$\mathbb{P}\left(\exists \mathbf{x} \in V' : |\mathbf{x}_{\mathcal{J}'}| = p \mid Z = z\right) = \mathbb{P}\left(\exists \mathbf{x} \in V'' : |\mathbf{x}| = p \mid Z = z\right).$$

Le problème étant maintenant que nous souhaiterions appliquer le lemme 7.6 et que cela est malheureusement impossible du fait que V'' contient des paires assorties et n'est donc pas un code aléatoire. Il s'agit précisément de la raison pour laquelle nous avons introduit le code V'''. Ce-dernier peut en effet être considéré comme un code aléatoire dont la matrice de parité est tirée uniformément parmi les matrices ternaires de taille $\max(0, n-n_I-z-m-k_V) \times (n_V - z - m)$. Nous pouvons donc lui appliquer le lemme 7.6. Cependant nous devons ici faire attention, les mots de poids p dans V''' n'ont pas la même probabilité d'apparition que V'' du fait qu'il existe des paires de positions assorties. C'est la raison de notre introduction pour $i \in [\![0, \lfloor p/2 \rfloor]\!]$ des ensembles X_i définis comme :

$$X_i \triangleq \left\{ \mathbf{x} = (x_i)_{i \in \mathcal{J}' \setminus \mathcal{J}_{22}} \in \mathbb{F}_3^{n-n_I-z-m} : |\mathbf{x}_{\mathcal{J}_1}| = p - 2i,\ |\mathbf{x}_{\mathcal{J}_{21}}| = i \right\}.$$

Un mot de poids p dans V'' correspond alors après poinçonnage dans $\mathcal{J}_{22}$ à un élément de X_i. Nous en déduisons alors facilement la borne inférieure :

$$\mathbb{P}\left(\exists \mathbf{x} \in V' : |\mathbf{x}_{\bar{\mathcal{I}}}| = p \mid Z = z, M = m\right) \geqslant \max_{i=0}^{\lfloor p/2 \rfloor} \left(\mathbb{P}(X_i \cap V''' \neq \varnothing)\right) \tag{7.15}$$

Donc d'après le lemme 7.6 nous obtenons :

$$\mathbb{P}(X_i \cap V''' \neq \varnothing) \geqslant f\left(\frac{\binom{n-n_I-z-2m}{p-2i}\binom{m}{i}2^{p-i}}{3^{\max(0,n-n_I-z-m-k_V)}} \right) \tag{7.16}$$

D'autre part, nous avons :

$$\begin{aligned} \mathbb{P}(Z = z, M = m) = {} & \frac{\binom{\frac{n}{2}-n_I}{m}\binom{n_I}{n-k-\ell-z}}{\binom{n}{n-k-\ell}} \\ & \times \sum_{j=0}^{n/2-n_I-m} \binom{n/2-n_I-m}{j} 2^j \binom{n_I}{z-n+2n_I+2m+j} \end{aligned} \tag{7.17}$$

ce qui permet de conclure la preuve de la proposition. □

Complexité pour retrouver V à une permutation près. De façon similaire au cas du code U, le complexité pour retrouver un permuté de V est de l'ordre de $\Omega\left(\frac{C_1(p,k,\ell)}{P_{\text{succ}}}\right)$.

7.2.2.3 Distinguer un code $(U, U+V)$-généralisé d'un code aléatoire

Il nous semble dans le second cas que la seule connaissance de V' ou du code V permuté ne permet pas directement de retrouver les paires assorties du code $(U, U+V)$-généralisé permuté. Cependant, quoiqu'il en soit, un seul appel réussi aux algorithmes CALCULV et CALCULU permet d'affirmer que le code considéré est un $(U, U+V)$-généralisé permuté. En d'autres termes, notre algorithme est bien à minima un distingueur de complexité donnée par la proposition qui suit.

Proposition 7.5. *L'algorithme 7 permet de distinguer un code $(U, U+V)$-généralisé permuté d'un code aléatoire en temps :*

$$\min\left(O\left(\min_{p,\ell} C_U(p,\ell)\right), O\left(\min_{p,\ell} C_V(p,\ell)\right)\right)$$

$$C_U(p,\ell) \triangleq \frac{C_1(p,k,\ell)}{\sum_{z=0}^{n/2} \frac{\binom{n/2}{z}\binom{n/2-z}{k+\ell-2z}2^{k+\ell-2z}}{\binom{n}{k+\ell}} \max_{i=0}^{\lfloor p/2 \rfloor} f\left(\frac{\binom{k+\ell-2z}{p-2i}\binom{z}{i}2^{p-i}}{3^{\max(0,k+\ell-z-k_U)}}\right)} \tag{7.18}$$

$$C_V(p,\ell) \triangleq \frac{C_1(p,k,\ell)}{\sum_{\mathcal{I}} \frac{\binom{\frac{n}{2}-n_I}{m}\binom{n_I}{n-k-\ell-z}}{\binom{n}{n-k-\ell}} \max_{i=0}^{\lfloor p/2 \rfloor} f\left(\frac{\binom{n-n_I-z-2m}{p-2i}\binom{m}{i}2^{p-i}}{3^{\max(0,n-n_I-z-m-k_V)}}\right) \binom{n/2-n_I-m}{j} 2^j \binom{n_I}{z-n+2n_I+2m+j}} \tag{7.19}$$

où $C_1(p,k,\ell)$ est la complexité pour calculer une fraction constante du nombre de mots de poids p dans un code de longueur $k+\ell$ et de dimension k. Ici, f dénote la fonction

$$f(x) \triangleq \max\left(x(1-x/2), 1-\frac{1}{x}\right).$$

La somme du dénominateur (7.19) est quant à elle faite sur le domaine :

$$\mathcal{I} = \{(z,m,j) \mid 0 \leqslant z \leqslant \min(n-k-\ell, n-n_I), 0 \leqslant m \leqslant n/2-n_I, \\ 0 \leqslant j \leqslant n/2-n_I-m\}.$$

Cette proposition va nous permettre d'estimer notre attaque pour distinguer un code $(U, U+V)$-généralisé permuté d'un code aléatoire. Malheureusement, donner en l'état une estimation numérique des formules de $C_U(p,\ell)$ et $C_V(p,\ell)$ n'est pas chose aisée. De plus, ces dernières dépendent de l'algorithme utilisé pour calculer des mots de poids p dans un code poinçonné. Dans la sous-section qui suit nous proposons l'utilisation de l'algorithme de Dumer [Dum91] (voir §2.2.1 et tout particulièrement la proposition 2.4). Nous verrons alors dans ce cas comment donner une bonne estimation de $C_U(p,\ell)$.

7.2.2.4 Un calcul effectif de la complexité pour retrouver U

Étant donnés k, k_U, nous souhaitons estimer $\min_{p,\ell} C_U(p,\ell)$ où :

$$C_U(p,\ell) = C_{p,\ell}/P_{p,\ell} \tag{7.20}$$

avec $C_{p,\ell}$ désignant la complexité pour calculer avec l'algorithme de Dumer les mots de poids p dans un code de longueur $k+\ell$, c'est à dire (voir la proposition 2.4) :

$$C_{p,\ell} = C_1(p,k,\ell) = \max\left(L_{p,\ell}, L_{p,\ell}^2 3^{-\ell}\right) \text{ avec } L_{p,\ell} = \sqrt{\binom{k+\ell}{p}2^p} \tag{7.21}$$

et

$$P_{p,\ell} \triangleq \sum_{z=0}^{n/2} \left(\frac{\binom{n/2}{z}\binom{n/2-z}{k+\ell-2z}2^{k+\ell-2z}}{\binom{n}{k+\ell}} \max_{0\leqslant i\leqslant p/2} f\left(\frac{\binom{k+\ell-2z}{p-2i}\binom{z}{i}2^{p-i}}{3^{\max(0,k+\ell-z-k_U)}}\right)\right) \tag{7.22}$$

avec f définie dans la proposition 7.5. Commençons par simplifier cette-dernière. En effet, f est égale à un facteur constant (plus petit que 3) à $\min(1,x)$. Cela nous permettra

"d'accélérer" grandement les calculs. Supposons donc dans ce qui suit que $f(x) = \min(1, x)$. Écrivons maintenant :

$$P_{p,\ell} = \sum_{z=0}^{n/2} G_\ell(z) F_{p,\ell}(z)$$

avec

$$G_\ell(z) = \frac{\binom{n/2}{z}\binom{n/2-z}{k+\ell-2z}2^{k+\ell-2z}}{\binom{n}{k+\ell}}, \quad \phi_{p,\ell}(z,i) = \binom{k+\ell-2z}{p-2i}\binom{z}{i}2^{p-i}$$

$$F_{p,\ell}(z) = \max_{0\leqslant i\leqslant p/2} f\left(\frac{\binom{k+\ell-2z}{p-2i}\binom{z}{i}2^{p-i}}{3^{\max(0,k+\ell-z-k_U)}}\right) = \min\left(1, \frac{\max\limits_{0\leqslant i\leqslant p/2}\phi_{p,\ell}(z,i)}{3^{k+\ell-z-k_U}}\right),$$

Nous nous sommes intéressés au comportement asymptotique (pour $n \to +\infty$) de ces différentes quantités. Afin de simplifier les calculs qui suivent nous avons choisi de garder la notation des paramètres entiers k, k_U, p, ℓ, z, i *mais désormais*, $x \in \{k, k_U, p, \ell, z, i\}$ désignera, au lieu d'un entier, la valeur réelle x/n fixée.

Les fonctions $C_{p,\ell}, L_{p,\ell}, P_{p,\ell}, G_\ell, F_{p,\ell}, \phi_{p,\ell}$ seront quant à elles écrites comme $\lim_{n\to\infty} \frac{1}{n}\log_2 X$ où $X \in \{C_{p,\ell}, L_{p,\ell}, P_{p,\ell}, G_\ell, F_{p,\ell}, \phi_{p,\ell}\}$.

Avec cette convention [1],

$$\begin{aligned}
C_U(p,\ell) &= C_{p,\ell} - P_{p,\ell} \\
C_{p,\ell} &= \max\left(L_{p,\ell}, 2L_{p,\ell} - \ell\log_2 3\right) \text{ avec } L_{p,\ell} = \frac{k+\ell}{2} h_3\left(\frac{p}{k+\ell}\right) \\
G_\ell(z) &= \frac{1}{2}h_2(2z) + \left(\frac{1}{2} - z\right) h_3\left(\frac{k+\ell-2z}{\frac{1}{2}-z}\right) - h_2(k+\ell) \\
F_{p,\ell}(z) &= \min\left(0, \tilde{F}_{p,\ell}(z)\right) \\
\tilde{F}_{p,\ell}(z) &= \max_{0\leqslant i\leqslant p/2} \phi_{p,\ell}(z,i) - (k+\ell-z-k_U)\log_2 3 \\
\phi_{p,\ell}(z,i) &= (k+\ell-2z)h_3\left(\frac{p-2i}{k+\ell-2z}\right) + wh_3\left(\frac{i}{z}\right)
\end{aligned}$$

Nous remplaçons la somme du dénominateur de $P_{p,\ell}$ par le maximum sur z,

$$P_{p,\ell} = \max_{0\leqslant z\leqslant 1/2} (G_\ell(z) + F_{p,\ell}(z)) \tag{7.23}$$

ce qui a un sens asymptotiquement du fait que chaque terme de la somme est exponentiel. Afin de déterminer la valeur de z atteignant le maximum, il est nécessaire d'étudier les variations de $z \mapsto G_\ell(z)$ et $z \to F_{p,\ell}(z)$. Cependant, déterminons tout d'abord les variations de $i \mapsto \phi_{p,\ell}(z,i)$ dont ces quantités dépendent afin d'obtenir le terme dominant de $\max_{0\leqslant i\leqslant p/2}\phi_{p,\ell}(z,i)$.

— La dérivée partielle de $\phi_{p,\ell}(z,i)$ selon i est donnée par :

$$\frac{\partial\phi_{p,\ell}}{\partial i}(z,i) = \log_2 \frac{(p-2i)^2(z-i)}{2i(k+\ell-2z-p+2i)^2}.$$

Il s'ensuit que la valeur de i maximisant $\phi_{p,\ell}(z,i)$ est solution d'une équation polynomiale d'ordre 3 :

$$Q(i) = 2i(k+\ell-2z-p+2i)^2 - (p-2i)^2(z-i) \tag{7.24}$$

Nous en déduisons alors facilement que Q admet une racine réelle unique dans l'intervalle $[0, p/2]$. Notons cette-dernière $i_0(z)$. Nous avons alors :

$$\tilde{F}_{p,\ell}(z) = \phi_{p,\ell}(z, i_0(z)) - (k+\ell-z-k_U)\log_2 3$$

1. $h_q(x) = -x\log_q(x/(q-1)) - (1-x)\log_q(1-x)$

— Les variations de $z \mapsto \tilde{F}_{p,\ell}(z)$ sont données par le terme $z \log_2 3$. De cette façon, $\tilde{F}_{p,\ell}(z)$ est une fonction croissante de z. Notons z_1 la racine réelle (unique) de $\tilde{F}_{p,\ell}(z)$ dans l'intervalle $]k+\ell-1/2,(k+\ell)/2[$. La fonction $z \mapsto F_{p,\ell}(z)$ est croissante sur $]k+\ell-1/2,z_1]$ et est nulle pour tout $z \in [z_1,(k+\ell)/2[$.

— La dérivée de $z \to \tilde{F}_{p,\ell}(z)$ est égale à :

$$\begin{aligned}\frac{d\tilde{F}_{p,\ell}}{dz}(z) &= \frac{di_0}{dz}(z)\frac{\partial \phi_{p,\ell}}{\partial i}(z,i_0(z)) + \frac{\partial \phi_{p,\ell}}{\partial z}(z,i_0(z)) + \log_2(3)\\ &= \frac{\partial \phi_{p,\ell}}{\partial z}(z,i_0(z)) + \log_2(3)\\ &= \log_2 \frac{3z(k+\ell-2z-p+2i_0(z))^2}{(z-i_0(z))(k+\ell-2z)^2}.\end{aligned}$$

— La dérivée de $z \to G_\ell(z)$ est quant à elle égale à :

$$\frac{dG_\ell}{dz}(z) = \log_2 \frac{(k+\ell-2z)^2}{2z(1-2k-2\ell+2z)}$$

et est nulle pour $z_0 = (k+\ell)^2/2$. La fonction $z \mapsto G_\ell(z)$ est alors croissante sur $\in [k+\ell-1/2,z_0]$ et décroissante sur $[z_0,(k+\ell)/2]$. De plus, $G_\ell(z_0)=0$.

— La dérivée de $z \to G_\ell(z) + \tilde{F}_{p,\ell}(z)$ est égale à :

$$P'_{p,\ell}(z) = \frac{dG_\ell}{dz}(z) + \frac{d\tilde{F}_{p,\ell}}{dz}(z) = \log_2 \frac{3(k+\ell-2z-p+2i_0(z))^2}{2(z-i_0(z))(1-2k-2\ell+2z)}. \tag{7.25}$$

Il existe dans ce cas une unique valeur $z \in]k+\ell-1/2,(k+\ell)/2[$ qui annule le dénominateur de cette expression. Nous le noterons z_2.

Considérons une paire (p,ℓ),

— Calculons z_0, si $F_{p,\ell}(z_0)=0$ alors $P_{p,\ell}=0$ et $C_U(p,\ell)=C_{p,\ell}$.

— Calculons z_1, z_2 et $z=\min(z_1,z_2)$

$$C_U(p,\ell) = C_{p,\ell} - G_\ell(z) - F_{p,\ell}(z)$$

Proposition 7.6. *Pour tout (k,k_U,p,ℓ) considérons $z_0=(k+\ell)^2/2$ et z_1,z_2 dénotant les racines de $z \mapsto \tilde{F}_{p,\ell}(z)$ et $z \mapsto P'_{p,\ell}(z)$ pour $z \in]k+\ell-1/2,(k+\ell)/2[$. Nous avons*

$$W_{p,\ell} = C_{p,\ell} - G_\ell(z) - F_{p,\ell}(z), \text{ où } z = \max(z_0,\min(z_1,z_2)).$$

Encore d'autres simplifications.

— Une très bonne approximation de $i_0(z)$ est donnée par

$$i_0(z) \approx \frac{p}{2}\frac{pw}{pw+(k+\ell-2z)^2}.$$

Nous avons obtenu cette approximation en supposant que $Q(i)$, donné dans (7.24), est proche d'une fonction affine pour $i \in [0,p/2]$. Cela est corroboré par nos expérimentations.

— **Débarrassons-nous de p :** nous avons

$$C_{p,\ell} = \max\left(L_{p,\ell}, 2L_{p,\ell} - \ell \log_2 3\right)$$

Pour les valeurs optimales des paramètres p et ℓ, les deux termes sont égaux. Cela nous donne alors l'identité

$$h_3\left(\frac{p}{k+\ell}\right) = \frac{2\ell \log_2 3}{k+\ell}$$

ce qui permet de n'écrire la valeur optimale p qu'en fonction de ℓ.

Application aux codes de la famille Wave. Nous proposons actuellement pour l'instanciation de la famille de fonctions Wave, $k_U = 0.8451n/2$ et $k = 0.676n/2$ ce qui donne avec les notations relatives $k_U = 0.42255$ and $k = 0.676$. La valeur minimale de $W_{p,\ell}$ est alors atteinte pour $(p, \ell) = (0.0008048, 0.003088)$ et le terme dominant de l'équation (7.23) correspond à $z = 0.25135$. De cette façon,

$$\frac{1}{n} \log_2 \min_{p,\ell} C_U(p, \ell) = 0.01768.$$

Application aux duaux des codes de la famille Wave. L'analyse que nous venons de faire doit aussi être faite pour les duaux des codes $(U, U+V)$-généralisés considérés dans Wave. En effet nous pouvons vérifier que le dual d'un code $(U, U+V)$-généralisé est un code $(V^\perp + U^\perp, V^\perp)$-généralisé. Ainsi, nous devons faire notre analyse pour $n-k$ au lieu de k, $n/2 - k_V$ à la place de k_U et $n/2 - k_U$ remplaçant k_V. Autrement dit avec les paramètres que nous proposons, $k_U = 0.246545$ et $k = 0.324$. La valeur minimale de $C_U(p, \ell)$ est dans ce cas atteinte pour $(p, \ell) = (0.0004627, 0.001737)$ et le terme dominant dans l'équation (7.23) correspond à $z = 0.07598$. Nous obtenons

$$\frac{1}{n} \log_2 \min_{p,\ell} C_{V^\perp}(p, \ell) = 0.01811.$$

7.2.2.5 Exposant de sécurité pour retrouver V

Les paramètres que nous proposons pour Wave font que le coût $C_V(p, \ell)$ pour retrouver V est bien plus élevé que $C_U(p, \ell)$. Ceci s'explique car la densité des mots de la forme $(\mathbf{b} \odot \mathbf{v}, \mathbf{d} \odot \mathbf{v})$ est exponentiellement inférieure à celle des mots $(\mathbf{a} \odot \mathbf{u}, \mathbf{c} \odot \mathbf{u})$ pour tous les poids (exceptés ceux où les mots $(\mathbf{a} \odot \mathbf{u}, \mathbf{c} \odot \mathbf{u})$ n'existent pas). Nous avons la même propriété avec les duaux $U^\perp$ et $V^\perp$. De cette façon, pour Wave avec les paramètres que nous donne, $C_U(p, \ell)$ donne la meilleure complexité. L'étude qui précède est alors suffisante.

Quatrième partie

Attaques contre des schémas utilisant des codes en métrique rang

Chapitre 8

Attaque contre une signature en métrique rang : RankSign

Introduction

`RankSign` **et les signatures utilisant des codes.** Concevoir une signature efficace dont la sécurité repose entre autres sur le problème du décodage générique est un défi ouvert depuis maintenant de nombreuses années. L'histoire des codes en cryptographie n'a cependant pas manqué de propositions (et de cryptanalyses) comme nous avons pu le voir dans l'introduction de la partie I de ce document. Dans le cas particulier de la métrique rang, la signature `RankSign` [Gab+14b] qui a été décrite dans §1.4.2 ouvrit la voie à des solutions particulièrement intéressantes. A titre d'exemple nous comparons les tailles de clef publique et longueur de signature entre `RankSign` [Ara+17c] et Wave [DST19b] dans la table 8.1.

Table 8.1 – Comparaison des différentes tailles de clef et signature entre `RankSign` et Wave pour une sécurité de 128 bits.

	Taille de clef publique	Longueur de signature
`RankSign`	0.01MB	11kb
Wave	3MB	13kb

Les paramètres de `RankSign` sont extrêmement attrayants. Malheureusement nous trouvâmes [DT18c] une attaque contre ce schéma.

Origine de l'attaque. La sécurité de `RankSign` a été démontrée se réduire aux deux problèmes difficiles issus de la métrique rang : (i) le décodage générique RSD (voir le problème 1.9) et (ii) distinguer un code LRPC-augmenté (voir la définition 1.23) d'un code aléatoire. Nous allons cependant présenter dans ce qui suit une attaque algébrique polynomiale retrouvant la structure du LRPC-augmenté. La table 8.2 donne les résultats numériques de cette dernière appliquée aux paramètres proposés dans [Ara+17c].

Notre attaque repose sur les observations suivantes :

— pour être en mesure de signer efficacement les paramètres des codes LRPC-augmentés utilisés dans `RankSign` doivent être choisis très particulièrement,

— il s'avère de façon inattendue qu'en raison des contraintes sur les paramètres il existe des mots de poids deux dans les codes LRPC-augmentés de `RankSign`,

Table 8.2 – Attaque contre les paramètres de `RankSign` proposés au NIST.

Sécurité [Ara+17c]	Temps
128 bits	20.12 s
192 bits	125.64 s
256 bits	256.90 s

— ces mots de poids faible peuvent être retrouvés efficacement avec des techniques algébriques de type base de Gröbner. Ces derniers peuvent alors être utilisés pour forger des signatures à la place de l'utilisateur légitime. De plus, nous n'en discuterons pas ici mais lors de nos calculs par base de Gröbner nous avons remarqué expérimentalement que le degré de régularité du système sur $\mathbb{F}_q$ que nous cherchons à résoudre est bien plus petit que prévu par la théorie.

Cependant permettons nous d'insister sur le point suivant. A l'exception de `RankSign` notre attaque n'en est pas une contre toutes les utilisations de codes LRPC en cryptographie.

8.1 Le problème de `RankSign` : des mots de petit poids

Rappelons (voir §1.4.2) qu'une clef publique de `RankSign` est une matrice $\mathbf{H}_{\text{pk}}$ de la forme :

$$\mathbf{H}_{\text{pk}} = \mathbf{Q}\mathbf{H}' \in \mathbb{F}_{q^m}^{(n-k)\times(n+t-k)} \quad \text{avec} \quad \mathbf{H}' = \left[\mathbf{H}|\mathbf{R}\right]\mathbf{P},$$

$$\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times} \text{ homogène de poids } d \text{ (définition 1.17)}, \quad \mathbf{R} \in \mathbb{F}_{q^m}^{(n-k)\times t} \text{ quelconque},$$

$$\mathbf{P} \in \mathbb{F}_{q}^{(n+t)\times(n+t)} \text{ inversible} \quad \text{et} \quad \mathbf{Q} \in \mathbb{F}_{q^m}^{(n-k)\times(n-k)} \text{ inversible}.$$

De plus, afin d'être en mesure de signer à distance w les paramètres n, m, k, w, t et d sont contraints par les trois équations :

$$m = w(d+1), \quad w = t + \frac{n-k}{d} \quad \text{et} \quad n = (n-k)d.$$

Notons pour toute la suite $\mathcal{C}_{\text{pk}}$ le $[n+t, k]_{q^m}$ "code public" admettant $\mathbf{H}_{\text{pk}}$ comme matrice de parité, *i.e* :

$$\mathcal{C}_{\text{pk}} \triangleq \left\{\mathbf{c} \in \mathbb{F}_{q^m}^{n+t} : \mathbf{c}\mathbf{H}_{\text{pk}}^{\intercal} = \mathbf{0}\right\}. \tag{8.1}$$

Ce code est par définition un LRPC-augmenté de type (d, t). Dans ce cas, son dual $\mathcal{C}_{\text{pk}}^{\perp}$ a pour matrice génératrice $\mathbf{H}_{\text{pk}}$. La multiplication à gauche par une matrice inversible n'étant qu'un changement de base du code, il est clair que $\mathcal{C}_{\text{pk}}^{\perp}$ a des mots de poids $\leqslant t + d$ du fait que $\mathbf{P}$ est une isométrie pour la métrique rang. Une façon naturelle de retrouver la structure secrète de $\mathbf{H}_{\text{pk}}$ est alors de chercher ces mots dans $\mathcal{C}_{\text{pk}}^{\perp}$. Les auteurs de [Gab+14b] ont alors choisi les paramètres de `RankSign` pour qu'une telle recherche soit au-dessus du niveau de sécurité.

Malheureusement il s'avère dans le cas de `RankSign` qu'il existe une faiblesse provenant directement de $\mathcal{C}_{\text{pk}}$. Ce dernier a de nombreux mots de poids 2 dont on montrera dans la prochaine section qu'ils sont (i) facilement accessibles et (ii) permettent de signer sans le secret.

L'origine de ces mots de poids faible provient de façon essentielle des contraintes sur le choix des paramètres du système. Plus précisément, c'est entre autres grâce (ou à cause) de l'équation :

$$n = (n-k)d. \tag{8.2}$$

Cette-dernière intervient alors de façon essentielle dans la proposition qui suit montrant l'existence de mots de poids 2 dans $\mathcal{C}_{\text{pk}}$.

Proposition 8.1. *Considérons $\mathcal{C}_{pk}$ un $[n+t, k+t]_{q^m}$-code public de* RankSign*. De plus, supposons que ce dernier fut obtenu à partir d'un $[n,k]_{q^m}$-code* LRPC *de matrice de parité homogène dont les entrées sont dans un même espace F de dimension d où :*

$$n = (n-k)d.$$

Soit maintenant $F' \subseteq F$ un sous-espace de dimension 2 et :

$$\mathcal{C}'_{pk} \stackrel{\triangle}{=} \left\{ \mathbf{c} \in \mathcal{C}_{pk} : \forall i \in [\![1, n+t]\!],\ c_i \in F' \right\}.$$

Alors $\mathcal{C}'_{pk}$ est un sous $\mathbb{F}_q$-espace de $\mathbb{F}_{q^m}^{n+t}$ tel que :

$$\dim_{\mathbb{F}_q} \mathcal{C}'_{pk} \geqslant n/d.$$

La preuve de cette proposition repose de façon cruciale sur le lemme suivant.

Lemme 8.1. *Soit $\mathcal{C}_{\text{LRPC}}$ un $[n,k]_{q^m}$-code* LRPC *de matrice de parité $\mathbf{H}$ homogène ayant toutes ses entrées dans un sous-espace $F \subseteq \mathbb{F}_{q^m}$. Supposons qu'il existe un sous-espace $F' \subseteq \mathbb{F}_{q^m}$ tel que :*

$$(n-k)\dim(F \cdot F') < n \dim F'.$$

Alors, il existe des mots de code non nuls $\mathbf{c} \in \mathcal{C}_{\text{LRPC}}$ tels que :

$$\text{Supp}(\mathbf{c})_{\mathbb{F}_q} \subseteq F'.$$

De plus, l'ensemble :

$$\mathcal{C}' \stackrel{\triangle}{=} \left\{ \mathbf{c} \in \mathcal{C} : \forall i \in [\![1, n]\!],\ c_i \in F' \right\}$$

est un sous $\mathbb{F}_q$-espace vectoriel de $\mathbb{F}_{q^m}^n$ et :

$$\dim(\mathcal{C}') \geqslant n \dim F' - (n-k)\dim(F \cdot F').$$

Démonstration du lemme 8.1.

Notons $h_{i,j}$ la coordonnée de $\mathbf{H}$ à la ligne i et colonne j. Par définition, un mot de code $\mathbf{c} \in \mathcal{C}_{\text{LRPC}}$ vérifie :

$$\forall i \in [\![1, n-k]\!], \quad \sum_{j=1}^{n} h_{i,j} c_j = 0. \tag{8.3}$$

Supposons maintenant que $\mathbf{c}$ ait ses coordonnées dans F'. Écrivons donc ce-dernier dans une base $\{f'_1, \cdots, f'_d\}$ de F' comme :

$$\forall j \in [\![1, n]\!], \quad c_i = \sum_{\ell=1}^{d'} c_\ell^{(j)} f'_\ell$$

où les $c_\ell^{(i)} \in \mathbb{F}_q$. Le système d'équations (8.3) se met alors sous la forme :

$$\forall i \in [\![1, n-k]\!], \quad \sum_{j=1}^{n} \sum_{\ell=1}^{d'} c_\ell^{(j)} f'_\ell h_{i,j} = 0. \tag{8.4}$$

Les $h_{i,j}$ appartenant par définition à F, nous pouvons écrire ces $n-k$ équations sur $\mathbb{F}_{q^m}$ dans une base de $F \cdot F'$ (voir la définition 1.19 pour le produit d'espaces). Nous obtenons alors un système de :

$$(n-k)\dim(F \cdot F') \quad \text{équations sur } \mathbb{F}_q. \tag{8.5}$$

Or le vecteur $\mathbf{c}$ est déterminé par :

$$n \dim(F') \quad \text{scalaires sur } \mathbb{F}_q. \tag{8.6}$$

La combinaison de (8.5) et (8.6) permet alors de facilement conclure la preuve du lemme. □

Nous sommes maintenant en mesure de prouver la proposition 8.1.

Démonstration de la proposition 8.1

Soit $\mathbf{H}_{\text{pk}} = \mathbf{Q}\left[\mathbf{H}|\mathbf{R}\right]\mathbf{P} \in \mathbb{F}_{q^m}^{(n-k)\times(n+t)}$ la matrice publique de `RankSign`.
Choisissons $\{x_1, x_2, \ldots, x_d\}$ une base de F telle que $\{x_1, x_2\}$ est une base de F'. Nous observons alors que :

$$F \cdot F' = \text{Vect}_{\mathbb{F}_q}\left(x_i x_j : i \in [\![1,d]\!],\ j \in [\![1,2]\!]\right).$$

Le cardinal de l'ensemble $\{x_i x_j : i \in [\![1,d]\!],\ j \in [\![1,2]\!]\}$ est alors du fait que $x_1x_2 = x_2x_1$ de taille $2d-1$ (et non $2d$ comme nous pourrions l'attendre d'un produit d'espaces de dimensions 2 et d). D'où :

$$\dim(F \cdot F') \leqslant 2d-1$$

ce qui mène aux inégalités,

$$\begin{aligned} n\dim(F') - (n-k)\dim(F\cdot F') &\geqslant 2n - (n-k)(2d-1) \\ &= 2d(n-k) - (n-k)(2d-1) \text{ car } (n=(n-k)d) \\ &= n-k \\ &= \frac{n}{d} \quad \text{car } n=(n-k)d. \end{aligned}$$

Notons $\mathcal{C}_{\text{LRPC}}$ le code LRPC de poids d et de matrice de parité $\mathbf{H}$. Considérons le sous $\mathbb{F}_q$-espace vectoriel $\mathcal{C}'_{\text{LRPC}}$ défini comme :

$$\mathcal{C}'_{\text{LRPC}} \stackrel{\triangle}{=} \left\{\mathbf{c} \in \mathcal{C}_{\text{LRPC}} : \forall i \in [\![1,n]\!],\ c_i \in F'\right\}.$$

D'après le lemme 8.1 nous avons :

$$\dim_{\mathbb{F}_q} \mathcal{C}'_{\text{LRPC}} \geqslant \frac{n}{d}. \tag{8.7}$$

Considérons maintenant :

$$\mathcal{C}'_{\text{pk}} \stackrel{\triangle}{=} \{(\mathbf{c}_{\text{LRPC}}, \mathbf{0}_t)(\mathbf{P}^{-1})^{\intercal} : \mathbf{c}_{\text{LRPC}} \in \mathcal{C}'_{\text{LRPC}}\}$$

où $\mathbf{0}_t$ désigne le vecteur avec t zéros. Nous en déduisons avec l'équation (8.7) :

$$\dim_{\mathbb{F}_q} \mathcal{C}'_{\text{pk}} \geqslant \frac{n}{d}.$$

De plus, toutes les entrées de tout vecteur $\mathbf{c}' \in \mathcal{C}'_{\text{pk}}$ appartiennent au $\mathbb{F}_q$-espace F' du fait que les coefficients $\mathbf{P}$ sont dans $\mathbb{F}_q$.
Prouvons maintenant que $\mathcal{C}'_{\text{pk}} \subseteq \mathcal{C}_{\text{pk}}$. Considérons pour cela $\mathbf{c}' \in \mathcal{C}'_{\text{pk}}$. Nous pouvons alors écrire ce vecteur comme :

$$\mathbf{c}' = (\mathbf{c}_{\text{LRPC}}, \mathbf{0}_t)(\mathbf{P}^{-1})^{\intercal}.$$

Observons désormais que,

$$\begin{aligned} \mathbf{H}_{\text{pk}}\mathbf{c}'^{\intercal} &= \mathbf{H}_{\text{pk}}\mathbf{P}^{-1}(\mathbf{c}_{\text{LRPC}}, \mathbf{0}_t)^{\intercal} \\ &= \mathbf{Q}\left[\mathbf{H}|\mathbf{R}\right]\mathbf{P}\mathbf{P}^{-1}(\mathbf{c}_{\text{LRPC}}, \mathbf{0}_t)^{\intercal} \\ &= \mathbf{Q}\left[\mathbf{H}|\mathbf{R}\right](\mathbf{c}_{\text{LRPC}}, \mathbf{0}_t)^{\intercal} \\ &= \mathbf{Q}\mathbf{H}\mathbf{c}_{\text{LRPC}}{}^{\intercal} \quad (\ \mathbf{R} \in \mathbb{F}_{q^m}^{(n-k)\times t}\ ) \\ &= \mathbf{0} \quad (\mathbf{c}_{\text{LRPC}} \text{ appartient au code de matrice de parité } \mathbf{H}) \end{aligned}$$

ce qui prouve que $\mathcal{C}'_{\text{pk}} \subseteq \mathcal{C}_{\text{pk}}$ et conclut la preuve. □

La proposition 8.1 prouve sous certaines conditions *l'existence* de mots de petits poids rang 2 dans un code LRPC-augmenté utilisé dans `RankSign`. En revanche, ce résultat ne donne en aucun cas une façon efficace de les trouver. C'est l'objet de ce qui suit.

8.2 L'attaque

Nous décrivons maintenant notre attaque [DT18c] contre la signature `RankSign`. Le point de départ de cette-dernière est l'existence de mots de poids 1 dépendant du secret.

8.2.1 Des mots de poids 1 dans un code projeté

La proposition 8.1 montre l'existence de nombreux mots de poids 2 dans $\mathcal{C}_{\text{pk}}$ et de support n'importe quel sous-espace de dimension 2 de F. Ces mots impliquent alors la présence de vecteurs de poids 1 dans un code matriciel projeté. La remarque fondamentale est la suivante. Reprenons la matrice homogène $\mathbf{H}$ de poids d et de $\mathbb{F}_q$-espace associé F. Nous avons l'existence de $\alpha \in \mathbb{F}_{q^m}$ tel que :

$$1 \in \alpha F.$$

La matrice de parité $\alpha\mathbf{H}$ définit alors le même code que $\mathbf{H}$ tout comme $\alpha\mathbf{H}_{\text{pk}}$ définit encore $\mathcal{C}_{\text{pk}}$. Nous pouvons donc supposer dans la suite et sans perte de généralité que $1 \in F$. Considérons alors un supplémentaire V de $\text{Vect}_{\mathbb{F}_q}(1) = \mathbb{F}_q$ dans $\mathbb{F}_{q^m}$, c'est à dire le $\mathbb{F}_q$-espace de dimension $m-1$ tel que :

$$\mathbb{F}_{q^m} = V \oplus \mathbb{F}_q$$

et la projection sur V :

$$\pi : \mathbb{F}_{q^m} \to V.$$

Nous savons maintenant qu'il existe des mots de code $\mathbf{c} \in \mathcal{C}_{\text{pk}}$ de support $\text{Vect}_{\mathbb{F}_q}(1, x) \subseteq F$ ($1 \in F$) qui est de dimension 2. Donc le mot $\pi(\mathbf{c})$ est de support inclu dans $\text{Vect}_{\mathbb{F}_q}(\pi(x))$ qui est de dimension 1 sur $\mathbb{F}_{q^m}$.

Rappelons maintenant que $\mathcal{C}_{\text{pk}}$ peut s'écrire comme un code matriciel sur $\mathbb{F}_q$. Soit $\{\alpha_1, \cdots, \alpha_m\}$ une base de $\mathbb{F}_{q^m}$, vu comme un $\mathbb{F}_q$-espace vectoriel, telle que :

$$\alpha_m = 1.$$

Tout mot de $\mathbf{c} \in \mathcal{C}_{\text{pk}}$ peut s'écrire comme :

$$\mathbf{Mat}(\mathbf{c}) = (M_{i,j}) \in \mathbb{F}_q^{m \times n} \quad \text{où} \quad c_j = \sum_{i=1}^{m} M_{i,j}\alpha_j.$$

Nous pouvons alors projeter ce code matriciel sur V où $\dim_{\mathbb{F}_q}(V) = m-1$ à l'aide de π. De plus, du fait que $\alpha_m = 1$ l'écriture matricielle pour $\mathbf{c} \in \mathcal{C}_{\text{pk}}$ de $\pi(\mathbf{c}) \in V$ est donnée par :

$$\mathbf{Mat}^{\text{proj}}(\mathbf{c}) = (M_{i,j})_{\substack{1 \leqslant i \leqslant m-1 \\ 1 \leqslant j \leqslant n+t}} \in \mathbb{F}_q^{(m-1)\times(n+t)}.$$

Définissons alors $\mathcal{C}_{\text{pub}}^{\text{proj}}$ le code matriciel sur $\mathbb{F}_q^{(m-1)\times(n+t)}$ comme :

$$\mathcal{C}_{\text{pub}}^{\text{proj}} \triangleq \left\{ \mathbf{Mat}^{\text{proj}}(\mathbf{c}) : \mathbf{c} \in \mathcal{C}_{\text{pk}} \right\}. \tag{8.8}$$

Il est clair que typiquement :

Fait 7. *$\mathcal{C}_{\text{pub}}^{\text{proj}}$ contient des mots de poids 1.*

Le code $\mathcal{C}_{\text{pub}}^{\text{proj}}$ est un code matriciel sur $\mathbb{F}_q^{(m-1)\times(n+t)}$ typiquement de dimension $(k+t)m$ (ce que l'on suppose dans la suite), *i.e* : de dimension celle du code matriciel associé à $\mathcal{C}_{\text{pk}}$. De cette façon nous pouvons "relever" tout mot de poids 1 dans $\mathcal{C}_{\text{pub}}^{\text{proj}}$ en un mot de poids $\leqslant 2$ dans le code $\mathcal{C}_{\text{pk}}$. En effet, pour tout $\mathbf{c} \in \mathcal{C}_{\text{pk}}$, la dernière ligne de $\mathbf{Mat}(\mathbf{c})$ est déterminée de façon unique par les précédentes. Nous pouvons donc relever $\mathbf{Mat}^{\text{proj}}(\mathbf{c})$ en $\mathbf{Mat}(\mathbf{c})$ par une simple combinaison linéaire de ses lignes. Nous appellerons dans la suite cette opération déduisant $\mathbf{c}$ de $\mathbf{Mat}^{\text{proj}}(\mathbf{c})$ *le relèvement de $\mathcal{C}_{\text{pub}}^{\text{proj}}$ à $\mathcal{C}_{pk}$*.

8.2.2 Un aperçu de l'attaque

Trouver des mots de poids 1 dans $\mathcal{C}_{\text{pub}}^{\text{proj}}$ va révéler la structure du code LRPC secret et sous-jacent à la clef publique. Le relèvement de ces mots de $\mathcal{C}_{\text{pub}}^{\text{proj}}$ à $\mathcal{C}_{\text{pk}}$ va typiquement donner des mots de poids 2 qui n'ont aucune raison d'exister pour un code aléatoire de même taille que $\mathcal{C}_{\text{pk}}$. Ces mots révèlent un espace $F' \subseteq F$ et finalement tout l'espace F en recommençant un petit nombre de fois. Dans le cas des paramètres de `RankSign` [Gab+14b ; Ara+17c] où $d = 2$ nous pouvons à l'aide de F calculer une matrice de parité de $\mathcal{C}_{\text{pk}}$ permettant finalement de signer. En résumé, notre attaque se déroule en 4 étapes :

1. Nous trouvons $\mathbf{M} \in \mathcal{C}_{\text{pub}}^{\text{proj}}$ de rang 1 en résolvant un système bilinéaire à l'aide de technique de type base de Gröbner.
2. Nous relevons $\mathbf{M} \in \mathcal{C}_{\text{pub}}^{\text{proj}}$ à $\mathbf{c} \in \mathcal{C}_{\text{pk}}$ et nous calculons $F' \stackrel{\triangle}{=} \text{Supp}(\mathbf{c})_{\mathbb{F}_q}$.
3. De l'espace F' nous en déduisons le $\mathbb{F}_q$-espace
$$\mathcal{C}'_{\text{pk}} \stackrel{\triangle}{=} \left\{ \mathbf{c} \in \mathcal{C}_{\text{pk}} : \forall i \in [\![1, n+t]\!],\ c_i \in F' \right\}. \tag{8.9}$$
Pour $d = 2$ il s'agit typiquement d'un espace de dimension k sur $\mathbb{F}_q$.
4. Nous utilisons le sous-espace $\mathcal{C}'_{\text{pk}}$ de $\mathcal{C}_{\text{pk}}$ pour trouver une matrice de parité de $\mathcal{C}_{\text{pk}}$ permettant de signer à la place d'un utilisateur légitime.

Les étapes 2 et 3 sont évidentes. Nous donnons dans les deux prochaines sous-sections des détails concernant les points 1 et 4.

8.2.3 Trouver les mots de poids 1 par résolution d'un système bilinéaire

Le système bilinéaire de départ. Calculer des mots de poids 1 dans $\mathcal{C}_{\text{pub}}^{\text{proj}}$ peut être formulé comme un problème de décodage d'un code matriciel et donc comme une instance du problème MinRank [BFS99 ; Cou01]. Nous pourrions alors utiliser les techniques de résolution de [KS99 ; FLP08 ; FSEDS10 ; Spa12] mais il nous a semblé préférable de reprendre la modélisation algébrique suggérée dans [Ara+17c]. Cette dernière consiste tout d'abord d'écrire une matrice $\mathbf{M} \in \mathcal{C}_{\text{pub}}^{\text{proj}}$ de rang 1 comme :

$$\mathbf{M} = \begin{pmatrix} x_1y_1 & x_1y_2 & \cdots & x_1y_{n+t} \\ x_2y_1 & x_2y_2 & \cdots & x_2y_{n+t} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m-1}y_1 & x_{m-1}y_2 & \cdots & x_{m-1}y_{n+t} \end{pmatrix} \tag{8.10}$$

avec pour inconnues $\mathbf{y} = (y_1, \cdots, y_{n+t}) \in \mathbb{F}_q^{n+t}$ et $\mathbf{x} = (x_1, \cdots, x_{m-1}) \in \mathbb{F}_q^{m-1}$ jouant le rôle d'un élément de F' écrit sur V.

Rappelons que $\mathcal{C}_{\text{pub}}^{\text{proj}} \subseteq \mathbb{F}_q^{(m-1)\times(n+t)}$ (voir (8.8)) est un $\mathbb{F}_q$-espace de dimension $(k+t)m$. Écrivons alors ce dernier comme un sous-espace de $\mathbb{F}_q^{(m-1)(n+t)}$, *i.e* : toute matrice $\mathbf{M} = (M_{i,j}) \in \mathbb{F}_q^{(m-1)\times(n+t)}$ est vue comme un vecteur $\mathbf{m} = (m_\ell) \in \mathbb{F}_q^{(m-1)(n+t)}$ où :

$$m_{(i-1)(n+t)+j} \stackrel{\triangle}{=} M_{i,j}.$$

Nous pouvons alors calculer une matrice de parité de l'écriture vectoriel de $\mathcal{C}_{\text{pub}}^{\text{proj}}$:

$$\mathbf{H}_{\text{pub}}^{\text{proj}} = (h_{i,j}^{\text{proj}}) \in \mathbb{F}_q^{(N-K)\times N} \quad \text{où} \quad N = (m-1)(n+t) \quad \text{et} \quad K = (k+t)m.$$

De cette façon la matrice $\mathbf{M}$ définie dans (8.10) est un élément de $\mathcal{C}_{\text{pub}}^{\text{proj}}$ si les x_i et y_i vérifient,

$$\begin{cases} \sum_{j=1}^{n+t}\sum_{i=1}^{m-1} h^{\text{proj}}_{1,(i-1)(n+t)+j} x_i y_j = 0 \\ \quad\vdots \\ \sum_{j=1}^{n+t}\sum_{i=1}^{m-1} h^{\text{proj}}_{(n+t)(m-1)-(k+t)m,(i-1)(n+t)+j} x_i y_j = 0 \end{cases} \tag{8.11}$$

Restreindre le nombre de solutions. Nous proposons d'utiliser des techniques de type base de Gröbner implantées en Magma pour résoudre le système bilinéaire (8.11). Nous n'entrerons pas dans les détails techniques de résolution de systèmes à partir de bases de Gröbner dans ce document et nous utiliserons Magma en boîte noire. Cependant, permettons-nous de renvoyer le lecteur à la très bonne référence [Fau15].

La résolution par base de Gröbner d'un système avec de nombreuses solutions peut être franchement accélérée (tout particulièrement l'étape de réduction d'une base d'un certain ordre à l'ordre lexicographique utilisé pour calculer les solutions) si nous rajoutons des équations (ou éliminons des variables) qui feront qu'il ne restera qu'une solution. Dans le cas de notre système bilinéaire (8.11) les solutions sont en grand nombre et forment même un $\mathbb{F}_q$-espace de vecteurs. L'espace des solutions est donc particulièrement gros (q est de l'ordre de 2^{32}). L'objet de la discussion qui suit est de montrer comment se ramener à un système n'ayant plus qu'une solution.

D'après la bilinéarité du système (8.11) nous pouvons déjà fixer :

$$x_1 = 1 \tag{8.12}$$

dès lors qu'il existe une solution $\mathbf{x}$ tel que $x_1 \neq 0$ (ce qui est le cas typiquement).

Rappelons maintenant que $\mathcal{C}'_{\text{pk}}$ (voir (8.9)) est un $\mathbb{F}_q$-espace vectoriel de dimension n/d. Donc pour une solution $\mathbf{x}$ de (8.11) (donnant un vecteur de F' projeté sur V), l'ensemble associé des solutions $\mathbf{y}$ forme aussi un $\mathbb{F}_q$-espace vectoriel de dimension n/d. Nous pouvons alors rajouter les équations :

$$\forall i \in [\![1, \frac{n}{d} - 1]\!],\ y_i = 0 \quad \text{et} \quad y_{n/d} = 1 \tag{8.13}$$

tout en étant sûr avec une bonne probabilité (de l'ordre de $1 - 1/q$) que nous gardons une solution.

Il y a maintenant un deuxième degré de liberté sur $\mathbf{x}$ qui joue le rôle d'un vecteur de $F' \subseteq F$. En effet, même si $d = \dim(F) = 2$, il existe de nombreux espaces $\alpha \cdot F$ tels que $1 \in \alpha \cdot F$. Étudions ce cas en détail et pour cela commençons par écrire F comme :

$$F = \text{Vect}_{\mathbb{F}_q}(a, b) \quad \text{où} \quad a, b \in \mathbb{F}_{q^m}.$$

Nous souhaiterions savoir combien de valeurs $z \in \mathbb{F}_{q^m}$ sont possibles pour qu'il existe $c \neq 0$ tel que :

$$\text{Vect}_{\mathbb{F}_q}(a, b) = c\text{Vect}_{\mathbb{F}_q}(1, z).$$

Les solutions $\mathbf{x}$ seront alors données par les valeurs de z projetées sur le $\mathbb{F}_q$-espace $V = \text{Vect}_{\mathbb{F}_q}(\alpha_1, \cdots, \alpha_{m-1})$.

Les différentes valeurs possibles de z sont données par celles de c en fonction de a et b. Il y a deux cas à considérer :

— **Cas 1 :** $c = \frac{a+b\nu}{\mu}$ pour $\mu \in \mathbb{F}_q^\times$ and $\nu \in \mathbb{F}_q$. Alors,

$$z = \frac{\beta b}{a + b\nu} + \delta \quad \text{où} \quad \beta, \delta \in \mathbb{F}_q.$$

Table 8.3 – Attaque contre les paramètres de `RankSign` proposés au NIST.

Sécurité [Ara+17c]	(n, k, m, d, t, q)	Temps	Mémoire maximale
128 bits	$(20, 10, 21, 2, 2, 2^{32})$	20.12 s	49 MB
128 bits	$(24, 12, 24, 2, 2, 2^{24})$	31.75 s	65 MB
192 bits	$(24, 12, 27, 2, 3, 2^{32})$	125.64 s	97 MB
256 bits	$(28, 14, 30, 2, 3, 2^{32})$	256.90 s	137 MB

— **Cas 2 :** $c = \frac{b}{\mu}$ pour $\mu \in \mathbb{F}_q^\times$. Ici,

$$z = \alpha \frac{a}{b} + \delta \quad \text{où} \quad \alpha, \delta \in \mathbb{F}_q$$

Le terme δ s'efface après la projection de x sur $\mathrm{Vect}_{\mathbb{F}_q}(\alpha_1, \ldots, \alpha_{m-1})$. Ainsi nous avons essentiellement pour x deux degrés de liberté sur $\mathbb{F}_q$. Une de ces libertés a déjà été utilisée en choisissant $x_1 = 1$. Nous pouvons alors en ajouter une seconde en fixant $x_2 = \alpha$ pour un élément α quelconque de $\mathbb{F}_q$. Lors de nos expérimentations nous avons constaté qu'ajouter une équation de la forme :

$$(x_2 - \alpha)(x_2 - \beta) = 0 \tag{8.14}$$

pour $\alpha, \beta \hookleftarrow \mathbb{F}_q$ était optimal pour le calcul d'un ensemble de solutions.

Nous résumons maintenant ces "réglages" du système bilinéaire (8.11) dans la proposition qui suit.

Proposition 8.2. *Nous obtenons après les éliminations des variables* (8.12)*,* (8.13) *et* (8.14) *dans le système* (8.11) *:*

— $nm - k(m+1) - t + 2$ *équations,*

— $m - 1 + n + t$ *inconnues.*

Nous en déduisons alors que dans un "régime typique" de la métrique rang en cryptographie où $m \approx n$, $k \approx \frac{n}{2}$ et $t \ll n$ que le nombre d'équations est de l'ordre de n^2 tandis que le nombre d'inconnues est lui de l'ordre de n. De cette façon, nous sommes dans une zone de paramètres où nous nous attendons à ce que les calculs de base de Gröbner se fassent en temps polynomial.

8.2.4 Résultats numériques

Nous donnons dans la table 8.3 nos résultats numériques pour retrouver des mots de poids 2 dans un code public de `RankSign` pour les paramètres proposés au NIST [Ara+17c]. Ces résultats ont été obtenus avec Magma et l'utilisation d'un processeur Intel Core i5, cadencé à 1.6 GHz à un seul cœur et avec 8 Go de RAM.

8.2.5 Finir l'attaque

Nous présentons dans cette sous-section la fin de notre attaque en montrant comment être en mesure de signer avec seulement une clef publique. Notre présentation est faite pour les paramètres de `RankSign` proposés au NIST, en particulier $d = 2$. Nous avons alors d'après l'équation (8.2) :

$$n - k = n/2 = k.$$

D'après la sous-section qui précède nous avons en main le $\mathbb{F}_q$-code $\mathcal{C}'_{\mathrm{pk}}$ (voir (8.9)) de dimension $\geqslant n/d = n/2 = k$. Ce code est seulement $\mathbb{F}_q$-linéaire, étendons-le comme un

code $\mathbb{F}_{q^m}$-linéaire et notons cette extension $\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pk}}$. Plus précisément, si $\{\mathbf{c}'_1, \ldots, \mathbf{c}'_{k'}\}$ est une $\mathbb{F}_q$-base de $\mathcal{C}'_{\text{pk}}$, alors :

$$\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pk}} = \text{Vect}_{\mathbb{F}_{q^m}} \left(\mathbf{c}'_1, \ldots, \mathbf{c}'_{k'}\right).$$

Pour simplifier la discussion qui suit faisons l'hypothèse suivante (qui s'avère être vérifiée dans nos expériences numériques).

Hypothèse 4.

$$\dim_{\mathbb{F}_{q^m}} \mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}} = k.$$

Le rationnel de cette hypothèse est que (i) la dimension de $\mathcal{C}'_{\text{pk}}$ est typiquement $n/d = k$ et (ii) le fait qu'une $\mathbb{F}_q$-base de $\mathcal{C}'_{\text{pk}}$ est aussi très probablement une $\mathbb{F}_{q^m}$-base.

Lemme 8.2. *Sous l'hypothèse 4 le code* $\left(\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}}\right)^{\perp}$ *est un* $[n+t, n+t-k]_{q^m}$*-code* LRPC *dont une matrice de parité homogène associée a ses entrées dans le* $\mathbb{F}_q-$*espace* F *de dimension* 2 *tel que* $1 \in F$*. De plus, les ensembles :*

$$\mathcal{D} \stackrel{\triangle}{=} \{\mathbf{c} \in \left(\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}}\right)^{\perp} \ : \ \text{Supp}(\mathbf{c}) \subseteq \mathbb{F}_q\}$$

$$\mathcal{D}' \stackrel{\triangle}{=} \{\mathbf{c} \in \left(\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}}\right)^{\perp} \ : \ \text{Supp}(\mathbf{c}) \subseteq F\}$$

sont des $\mathbb{F}_q$*-espaces de dimensions respectives* $\geqslant t$ *et* $\geqslant n-k+2t$*.*

Démonstration du lemme 8.2.

D'après l'hypothèse 4, l'espace $\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pk}}$ est de dimension k sur $\mathbb{F}_{q^m}$ et alors,

$$\dim_{\mathbb{F}_{q^m}} \left(\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pk}}\right)^{\perp} = n+t-k.$$

Il existe par définition une matrice génératrice de $\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pk}}$ dont les lignes sont des éléments de $\mathcal{C}'_{\text{pk}}$. Cette dernière est donc homogène de poids 2. Notons alors l'espace associé F. Cette matrice génératrice est aussi une matrice de parité du code $\left(\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}}\right)^{\perp}$ qui est donc un code LRPC de poids 2. Vérifions maintenant les conditions du lemme 8.1 pour ce code LRPC avec $F' = \mathbb{F}_q$. Nous avons :

$$\begin{aligned}(n+t)\dim_{\mathbb{F}_q}(\mathbb{F}_q) - (n+t-(n+t-k))\dim_{\mathbb{F}_q}(F \cdot \mathbb{F}_q) &= n+t-2k \\ &= t \ (\text{car } 2k = n)\end{aligned}$$

Donc nous pouvons lui appliquer le lemme 8.1 ce qui donne le résultat voulu pour $\mathcal{D}$.

Nous appliquons encore une fois le lemme mais cette fois-ci pour $F' = F$. Disons que $F = \text{Vect}_{\mathbb{F}_q}(1, x_1)$. Nous obtenons alors comme borne inférieure pour la dimension de $\mathcal{D}'$,

$$\begin{aligned}(n+t)\dim_{\mathbb{F}_q}(F) - (n+t-(n+t-k))\dim_{\mathbb{F}_q}(F \cdot F) & \\ \geqslant 2(n+t) - 3k(\ F \cdot F = \text{Vect}_{\mathbb{F}_q}\left(1, x_1, x_1^2\right)) & \\ = n-k+2t \ (\text{car } 2k = n). &\end{aligned}$$

□

Pour finir notre attaque nous allons faire l'hypothèse suivante qui une fois de plus est corroborée par nos expérimentations.

Hypothèse 5. *Nous pouvons extraire des ensembles* $\mathcal{D}$ *et* $\mathcal{D}'$ *une base du code* $\left(\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}}\right)^{\perp}$ *sur* $\mathbb{F}_{q^m}$ *avec,*

1. t *mots de code de support* $\mathbb{F}_q$*,*

2. $n-k$ *mots de code de même support de dimension* 2 *(l'espace secret) et contenant* 1.

Lemme 8.3. *Sous les hypothèses 4 et 5, il existe une matrice de parité* $\mathbf{H}' \in \mathbb{F}_{q^m}^{(n+t-k)\times(n+t)}$ *de* $\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}}$*, deux matrices inversibles* $\mathbf{P} \in \mathbb{F}_q^{(n+t)\times(n+t)}$ *et* $\mathbf{S} \in \mathbb{F}_{q^m}(n+t-k) \times (n+t-k)$ *telles que :*

$$\mathbf{SH}'\mathbf{P} = \begin{pmatrix} \mathbf{1}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{pmatrix}$$

où $\mathbf{R} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ *est homogène de degré* 2*.*

Démonstration du lemme 8.3.

Sous les hypothèses 4 et 5 il existe une matrice génératrice de $\left(\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}}\right)^{\perp}$ et donc une matrice de parité de $\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}}$ homogène de degré 2. De plus, cette dernière a la particularité d'avoir ses t premières lignes de rang 1 (quitte à les réordonner). Notons-la $\mathbf{H}'$. Ainsi, par une élimination de Gauss sur ses lignes nous obtenons $\mathbf{S} \in \mathbb{F}_{q^m}^{(n+t-k)\times(n+t-k)}$ inversible telle que :

$$\mathbf{SH}' = \begin{pmatrix} \mathbf{1}_t & \mathbf{Q} \\ \mathbf{0} & \mathbf{R} \end{pmatrix}$$

où $\mathbf{Q} \in \mathbb{F}_q^{t\times(n+t)}$ et $\mathbf{R} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ homogène de degré 2. De cette façon il existe une matrice inversible $\mathbf{P} \in \mathbb{F}_q^{(n+t)\times(n+t)}$ telle que :

$$\mathbf{SH}'\mathbf{P} = \begin{pmatrix} \mathbf{1}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{pmatrix}$$

ce qui conclut la preuve. □

L'idée désormais pour signer à la place d'un utilisateur légitime est de se servir de la matrice homogène $\mathbf{R}$ et du décodeur décrit dans §1.4.2. Rappelons que pour la matrice $\mathbf{H}_{\text{pk}}$ (qui définit le code public $\mathcal{C}_{\text{pk}}$) et un message $\mathbf{m}$, une signature est une erreur $\mathbf{e}$ de poids $w = t + \frac{n-k}{d}$ telle que :

$$\mathbf{e}\mathbf{H}_{\text{pk}}^{\intercal} = \mathbf{s} \quad \text{où} \quad \mathbf{s} = \mathcal{H}(\mathbf{m})$$

avec $\mathcal{H}$ une fonction de hachage cryptographique. Notre algorithme fonctionne alors de la façon suivante :

1. Nous calculons $\mathbf{y} \in \mathbb{F}_{q^m}^{n+t}$ tel que $\mathbf{y}\mathbf{H}_{\text{pk}}^{\intercal} = \mathbf{s}$.
2. Soit $\mathbf{y}' = \mathbf{y}(\mathbf{P}^{-1})^{\intercal}$, nous calculons $\mathbf{s}' = (\mathbf{SH}'\mathbf{P})\mathbf{y}'^{\intercal}$.
3. Décomposons maintenant $\mathbf{s}'$ comme $(\mathbf{s}'_1, \mathbf{s}'_2)$ où $\mathbf{s}'_1 \in \mathbb{F}_{q^m}^t$ et $\mathbf{s}'_2 \in \mathbb{F}_{q^m}^{n-k}$. Nous appliquons alors le décodeur de §1.4.2 avec :
   - Le sous-espace $T \overset{\triangle}{=} \text{Supp}(\mathbf{s}'_1)$,
   - La matrice de parité $\mathbf{R}$ et le syndrome $\mathbf{s}'_2$.

   Il est retourné $\mathbf{e}'$ tel que $T \subseteq \text{Supp}(\mathbf{e}')$ et $\mathbf{e}'\mathbf{R}^{\intercal} = \mathbf{s}'_2$.
4. Nous renvoyons le vecteur $\mathbf{e} = (\mathbf{s}'_1, \mathbf{e}')\mathbf{P}^{\intercal}$.

Montrons pour conclure la correction de cet algorithme. En d'autres termes démontrons que $\mathbf{e}\mathbf{H}_{\text{pk}}^{\intercal} = \mathbf{s}$ avec $|\mathbf{e}| = w$ ce qui conclura notre attaque.
Démonstration de la correction.

Tout d'abord nous avons,

$$\begin{aligned}
\mathbf{H}'\mathbf{e}^\intercal &= \mathbf{H}'\mathbf{P}(\mathbf{s}'_1, \mathbf{e}')^\intercal \\
&= \mathbf{S}^{-1}(\mathbf{S}\mathbf{H}'\mathbf{P})(\mathbf{s}'_1, \mathbf{e}')^\intercal \\
&= \mathbf{S}^{-1}\begin{pmatrix} \mathbf{1}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{pmatrix}(\mathbf{s}'_1, \mathbf{e}')^\intercal \\
&= \mathbf{S}^{-1}\begin{pmatrix} \mathbf{s}_1'^\intercal \\ \mathbf{R}\mathbf{e}'^\intercal \end{pmatrix} \text{ (car } \mathbf{s}'_1 \text{ est de taille } t) \\
&= \mathbf{S}^{-1}\mathbf{s}'^\intercal \text{ (car } \mathbf{R}\mathbf{e}'^\intercal = \mathbf{s}_2'^\intercal) \\
&= \mathbf{S}^{-1}(\mathbf{S}\mathbf{H}'\mathbf{P})\mathbf{y}'^\intercal \\
&= \mathbf{H}'\mathbf{P}(\mathbf{P}^{-1}\mathbf{y}^\intercal) \\
&= \mathbf{H}'\mathbf{y}^\intercal.
\end{aligned}$$

Ceci implique alors que $\mathbf{H}'(\mathbf{e}-\mathbf{y})^\intercal = \mathbf{0}$ et $\mathbf{y}-\mathbf{e} \in \mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}}$. Rappelons maintenant que $\mathbb{F}_{q^m} \otimes \mathcal{C}'_{\text{pub}} \subseteq \mathcal{C}_{\text{pub}}$ et de cette façon $\mathbf{H}_{\text{pk}}(\mathbf{e}-\mathbf{y})^\intercal = \mathbf{0}$. Par linéarité nous en déduisons que $\mathbf{H}_{\text{pk}}\mathbf{e}^\intercal = \mathbf{H}_{\text{pk}}\mathbf{y}^\intercal = \mathbf{s}^\intercal$. □

Ainsi, sous la condition que le décodeur du point 3 fonctionne avec pour entrée la matrice $\mathbf{H}'$, le syndrome $\mathbf{s}$ et le sous-espace T, notre algorithme décode bien $\mathbf{s}$ relativement à $\mathbf{H}_{\text{pk}}$.

La matrice de parité $\mathbf{R} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ est homogène de degré 2. Nous pouvons alors appliquer le décodeur décrit pour `RankSign` dans §1.4.2 (les contraintes sur les paramètres sont par définition vérifiées ici). Ce dernier renverra une erreur $\mathbf{e}'$ contenant le support T de $\mathbf{s}'_1$ du bon poids w. Ainsi, l'erreur $\mathbf{e} = (\mathbf{s}'_1, \mathbf{e}')\mathbf{P}^\intercal$ est de même poids w étant donné que $\mathbf{P}$ est inversible et à coefficient dans le petit corps $\mathbb{F}_q$. □

Chapitre 9

Une attaque contre un IBE utilisant des codes

Introduction

L'IBE de [Gab+17b]. La conception d'un IBE est un problème cryptographique difficile. C'est dans ce contexte où les solutions ne sont pas légion que les auteurs de [Gab+17b] ont proposé le premier IBE dont la sécurité repose sur des problèmes issus des codes correcteurs comme nous l'avons décrit dans §1.4.3. Il est alors nécessaire d'avoir en main deux codes $\mathbb{F}_{q^m}$-linéaires, le premier :

— de longueur n_{signe}, de dimension k_{signe} que l'on sait décoder à distance w_{signe} et qui peut être instancié en signature de type hache et signe (un code LRPC-augmenté de `RankSign` dans [Gab+17b]),

ainsi qu'un second :

— de longueur n_{dec}, de dimension k_{dec} que l'on sait décoder à distance $w_{\text{signe}}w_{\text{dec}}$ qui peut être instancié en un chiffrement où le secret consiste essentiellement en des mots de poids w_{dec} de même support.

L'idée étant que la clef secrète associée à une identité id est la signature de id qui peut alors être utilisée avec le chiffrement `RankPKE` (voir §1.3.5). Les paramètres de cet IBE doivent cependant être choisis précautionneusement. En effet, w_{dec} et w_{signe} ne doivent pas être trop petits car autrement nous pourrions retrouver le secret du chiffrement ou signer à la place de l'utilisateur légitime. Ceci est alors contradictoire avec le fait que l'on doive disposer d'un code venant avec un algorithme de décodage à distance $w_{\text{dec}}w_{\text{signe}}$. La borne de Gilbert-Varshamov étant la plus élevée pour des rendements faibles il est alors nécessaire de choisir $k_{\text{dec}}/n_{\text{dec}}$ petit et donc n_{dec} grand (un k_{signe} trop faible implique aussi une faiblesse). Néanmoins le problème ne s'arrête pas ici. La sécurité du schéma `RankPKE` utilisé dans l'IBE se réduit au problème RSL (voir le problème 1.15) où sont donnés n_{dec} syndromes. La difficulté de ce problème diminuant avec n_{dec} augmentant il faut faire attention dans le choix des paramètres. Les auteurs de [Gab+17b] ont alors proposé des paramètres tenant compte de toutes ces contraintes, en particulier dans l'état de l'art des attaques contre RSL.

Notre contribution : une attaque efficace contre l'IBE de [Gab+17b]. Bien que l'IBE de [Gab+17b] ne puisse plus être instancié depuis notre attaque contre `RankSign` nous avons montré [DT18c] que le problème est plus profond que de trouver une nouvelle signature. Notre contribution [DT18c] fut de diminuer le nombre de syndromes nécessaires pour monter une attaque contre RSL. Nous avons alors montré que les paramètres de RSL

Table 9.1 – Attaque contre les paramètres du problème RSL utilisé dans l'IBE [Gab+17b]

Sécurité [Gab+17b]	Temps
128 bits	626s

proposés dans [Gab+17b] sont cassés par une attaque algébrique efficace dont la table 9.1 donne les résultats numériques.

Notre approche fut similaire à celle que nous avons eu pour casser `RankSign` :

— nous exhibons à partir des données publiques un code matriciel contenant de nombreux mots de poids faible révélant l'espace secret du problème RSL associé,

— nous trouvons ensuite ces mots de petit poids en résolvant un système d'équations bilinéaires très largement sur-déterminé à l'aide de bases de Gröbner où encore une fois le degré de régularité du système s'avéra bien plus petit que prévu.

Permettons nous d'insister sur le point suivant. Notre attaque a fonctionné contre les paramètres proposés dans [Gab+17b] car nous avons diminué le nombre de syndromes nécessaires pour monter une attaque contre RSL. Notre nouvelle borne contraint encore plus le choix des paramètres et rend l'instanciation de l'IBE encore plus difficile. En revanche, contrairement à `RankSign` nous montrerons que cette dernière n'est pas impossible. La zone de paramètres résistant à notre attaque est certes faible mais elle existe si une signature appropriée de type hache et signe venait à être trouvée.

En revanche, nous allons montrer que la situation est dramatique concernant l'adaptation de l'IBE de [Gab+17b] en métrique de Hamming sur le corps $\mathbb{F}_2$. Nous allons en effet voir qu'une utilisation de l'algorithme de Prange [Pra62] (le plus simple des algorithmes de décodage générique) permet de casser en temps polynomial l'IBE dans ce cas et quels que soient les paramètres choisis. Une adaptation directe de l'approche de GPV en code avec la métrique de Hamming dans $\mathbb{F}_2$ est donc un chemin difficile à prendre.

9.1 Attaque contre l'IBE en métrique rang

Rappelons que la sécurité de l'IBE [Gab+17b] se réduit entre autres au problème RSL suivant.

— Étant donné $\mathbf{G}_{\text{signe}} \in \mathbb{F}_{q^m}^{k_{\text{signe}} \times n_{\text{signe}}}$ et $\mathbf{G}_{\text{signe}}\mathbf{E}$ où $\mathbf{E} \in \mathbb{F}_{q^m}^{n_{\text{signe}} \times n_{\text{dec}}}$ est une matrice homogène d'espace associé F de dimension w_{dec}, retrouver F.

Nous allons montrer dans la prochaine sous-section que sous la condition $n_{\text{dec}} > w_{\text{dec}}(n_{\text{signe}} - k_{\text{signe}})$ (qui est vérifiée dans [Gab+17b] et que l'on suppose vraie dans toute la suite) que le code $\mathcal{C}$ défini par :

$$\mathcal{C} = \{\mathbf{e}(\mathbf{G}_{\text{sgn}}\mathbf{E})^{\mathsf{T}} : \mathbf{e} \in \mathbb{F}_q^{n_{\text{dec}}}\} \subseteq \mathbb{F}_{q^m}^{k_{\text{sgn}}} \tag{9.1}$$

est un $\mathbb{F}_q$-espace contenant des mots de poids $\leqslant w_{\text{dec}}$ qui révèlent F. Plus précisément, il s'avère que le sous-espace $\mathcal{C}' \triangleq \mathcal{C} \cap F^{k_{\text{signe}}}$ est de dimension $\geqslant n_{\text{dec}} - w_{\text{dec}}(n_{\text{signe}} - k_{\text{signe}})$. Nous montrerons alors dans §9.1.2 comment retrouver ce sous-espace (et donc F) à l'aide de techniques de type base de Gröbner. Nous conclurons la section en montrant, bien que les paramètres de l'IBE soient contraints, qu'il existe en principe une zone de paramètres évitant cette attaque.

9.1.1 Des mots de petits poids dans les instances de RSL

L'objet de cette sous-section est de démontrer la proposition qui suit.

Proposition 9.1. *Soit* $(\mathbf{A}, \mathbf{AE})$ *une instance de* RSL *pour les paramètres* n, k, N, w *avec* $\mathbf{A} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ *sous forme systématique et* $\mathbf{E} \in \mathbb{F}_{q^m}^{n\times N}$ *homogène dont tous les coefficients appartiennent à l'espace* F *de dimension* w*. De plus, supposons que,*

$$N > wk. \tag{9.2}$$

Considérons,

$$\begin{aligned} \mathcal{C} &\stackrel{\triangle}{=} \{\mathbf{e}(\mathbf{AE})^{\intercal} : \mathbf{e} \in \mathbb{F}_q^N\} \\ \mathcal{C}' &\stackrel{\triangle}{=} \mathcal{C} \cap F^{n-k}. \end{aligned}$$

Alors $\mathcal{C}'$ *est un sous* $\mathbb{F}_q$*-espace de* $\mathcal{C}$ *de dimension* $\geqslant N - wk$*.*

Démonstration de la proposition 9.1.

Décomposons tout $\mathbf{E}$ en deux parties, $\left[\frac{\mathbf{E}_1}{\mathbf{E}_2}\right]$ où $\mathbf{E}_1$ est la matrice formée des $n-k$ premières lignes de $\mathbf{E}$ et $\mathbf{E}_2$ les k dernières. La matrice $\mathbf{A}$ est sous forme systématique, *i.e* : $\begin{pmatrix}\mathbf{1}_{n-k} & \mathbf{A}'\end{pmatrix}$ où $\mathbf{A}' \in \mathbb{F}_{q^m}^{(n-k)\times k}$. Donc :

$$\mathbf{AE} = \mathbf{E}_1 + \mathbf{A}'\mathbf{E}_2$$

De cette façon, pour prouver notre proposition il suffit de démontrer que,

$$\mathcal{S} \stackrel{\triangle}{=} \{\mathbf{e} \in \mathbb{F}_q^N : \mathbf{E}_2\mathbf{e}^{\intercal} = \mathbf{0}\}$$

est un sous $\mathbb{F}_q$-espace de dimension $\geqslant N - wk$. En effet,

$$\forall \mathbf{e} \in \mathcal{S}, \quad \mathbf{e}(\mathbf{AE})^{\intercal} = \mathbf{e}\mathbf{E}_1^{\intercal} \in F^{n-k}$$

comme les coefficients de $\mathbf{E}_1$ appartiennent au $\mathbb{F}_q$-espace F et ceux de $\mathbf{e}$ sont dans $\mathbb{F}_q$. Notons $\mathbf{E}(i,j)$ le coefficient de $\mathbf{E}$ à la ligne i et la colonne j. Un mot de $\mathcal{S}$ satisfait par construction :

$$\forall i \in [\![1, k]\!], \quad \sum_{j=1}^{N} \mathbf{E}(i,j)e_j = 0.$$

Donc si nous écrivons ce système de k équations sur $\mathbb{F}_q$ dans une base de F sur $\mathbb{F}_q$ (car $\sum_{j=1}^{N} \mathbf{E}(i,j)e_j$ appartient par définition à $F \cdot \mathbb{F}_q = F$) nous obtenons $k \dim_{\mathbb{F}_q}(F) = kw$ équations linéaires sur $\mathbb{F}_q$. Le nombre d'inconnues sur $\mathbb{F}_q$ est quant à lui de N (les e_j car $\mathbf{e} \in \mathbb{F}_q^N$). L'espace de solutions de notre système est donc de dimension plus grande que $N - wk$ ce qui conclut la preuve de notre proposition. □

9.1.2 Comment trouver les mots de petits poids dans une instance du problème RSL

La proposition 9.1 montre que sous la condition $n_{\text{dec}} > w_{\text{dec}}(n_{\text{sgn}} - k_{\text{sgn}})$ il existe des mots de poids $\leqslant w_{\text{dec}}$ dans le code $\mathcal{C}$ (voir (9.1)) révélant l'espace F. Cette proposition ne dit en revanche rien sur la façon dont nous pouvons les retrouver. Nous proposons alors pour cela une modélisation algébrique en système bilinéaire du même type que celle que nous avons introduite lors de l'attaque contre `RankSign` dans §8.2.3.

Rappelons que pour $(\alpha_1, \cdots, \alpha_m)$ une base de $\mathbb{F}_{q^m}$ sur $\mathbb{F}_q$ nous pouvons écrire les éléments de $\mathbb{F}_{q^m}^{k_{\text{signe}}}$ comme des matrices de taille $m \times k_{\text{signe}}$:

$$\forall \mathbf{x} \in \mathbb{F}_{q^m}^{k_{\text{signe}}}, \quad \mathbf{Mat}(\mathbf{x}) = (X_{i,j}) \in \mathbb{F}_q^{m\times k_{\text{signe}}} \text{ où } x_j = \sum_{i=1}^{m} \alpha_i X_{i,j}.$$

Nous associons alors à $\mathcal{C}$ le code $\mathcal{C}^{\text{Mat}}$ défini par :

$$\mathcal{C}^{\text{Mat}} \stackrel{\triangle}{=} \{\mathbf{Mat}(\mathbf{c}) : \mathbf{c} \in \mathcal{C}\} \subseteq \mathbb{F}_q^{m\times k_{\text{signe}}}.$$

On vérifie alors que ce code matriciel est de dimension n_{dec}. De plus, en appliquant la proposition 9.1 nous avons :

Fait 8. *$\mathcal{C}^{\text{Mat}}$ contient des mots de poids $\leqslant \dim(F)$ qui forment un sous $\mathbb{F}_q$-espace de dimension $\geqslant n_{\text{dec}} - w_{\text{dec}}(n_{\text{signe}} - k_{\text{signe}})$.*

Il s'agit des mots de la forme $\mathbf{Mat}(\mathbf{c})$ avec $\text{Supp}(\mathbf{c})_{\mathbb{F}_q} \subseteq F$. Or pour un code de même taille que $\mathcal{C}^{\text{Mat}}$ avec les paramètres de [Gab+17b], la distance de Gilbert-Varshamov est bien supérieure à $\dim(F)$. De cette façon, si nous retrouvons des mots de $\mathcal{C}^{\text{Mat}}$ ces-derniers auront de grandes chances de révéler F ou à minima donneront un distingueur.

Le système bilinéaire considéré. Le problème de trouver des mots de poids $\dim_{\mathbb{F}_q}(F) = w_{\text{dec}}$ peut être écrit comme une instance de MinRank[BFS99 ; Cou01]. Nous allons comme dans le cas de RankSign utiliser la modélisation algébrique suggérée dans [Ara+17c]. Cette dernière consiste tout d'abord à écrire une matrice $\mathbf{M} \in \mathcal{C}^{\text{Mat}}$ de rang w_{dec} comme :

$$\mathbf{M} = \begin{pmatrix} \sum_{i=1}^{w_{\text{dec}}} x_1^i y_1^i & \sum_{i=1}^{w_{\text{dec}}} x_1^i y_2^i & \cdots & \sum_{i=1}^{w_{\text{dec}}} x_1^i y_{k_{\text{signe}}}^i \\ \sum_{i=1}^{w_{\text{dec}}} x_2^i y_1^i & \sum_{i=1}^{w_{\text{dec}}} x_2^i y_2^i & \cdots & \sum_{i=1}^{w_{\text{dec}}} x_2^i y_{k_{\text{signe}}}^i \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{w_{\text{dec}}} x_m^i y_1^i & \sum_{i=1}^{w_{\text{dec}}} x_m^i y_2^i & \cdots & \sum_{i=1}^{w_{\text{dec}}} x_m^i y_{k_{\text{signe}}}^i \end{pmatrix}. \tag{9.3}$$

avec pour inconnues $\mathbf{x}^i = (x_1^i, \cdots, x_m^i) \in \mathbb{F}_q^m$ (jouant le rôle d'une base de F écrite sur $\mathbb{F}_q$) et $\mathbf{y}_j^i \in \mathbb{F}_q^{k_{\text{signe}}}$ pour $1 \leqslant i \leqslant w_{\text{dec}}$ et $1 \leqslant j \leqslant k_{\text{signe}}$.

Rappelons maintenant que $\mathcal{C}^{\text{Mat}} \subseteq \mathbb{F}_q^{m \times k_{\text{signe}}}$ est un $\mathbb{F}_q$-espace vectoriel de dimension n_{dec}. Écrivons alors ce-dernier comme un sous-espace de $\mathbb{F}_q^{mk_{\text{signe}}}$, *i.e* : toute matrice $\mathbf{M} = (M_{ij})_{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant j \leqslant k_{\text{signe}}}}$ est vue comme un vecteur $\mathbf{m} = (m_\ell)_{1 \leqslant \ell \leqslant mk_{\text{signe}}}$ où :

$$m_{(i-1)k_{\text{signe}}+j} \triangleq M_{i,j}.$$

Nous pouvons alors calculer une matrice de parité de l'écriture vectoriel de $\mathcal{C}^{\text{Mat}}$,

$$\mathbf{H}^{\text{Mat}} = (h_{ij}^{\text{Mat}})_{\substack{1 \leqslant i \leqslant mk_{\text{signe}} - n_{\text{dec}} \\ 1 \leqslant j \leqslant mk_{\text{signe}}}}.$$

De cette façon la matrice $\mathbf{M}$ définie dans (9.3) est un élément de $\mathcal{C}^{\text{Mat}}$ si les x_i^ℓ et y_j^ℓ vérifient :

$$\begin{cases} \sum_{\ell=1}^{w_{\text{dec}}} \sum_{j=1}^{k_{\text{signe}}} \sum_{i=1}^{m} h_{1,(i-1)k_{\text{signe}}+j}^{\text{Mat}} x_i^\ell y_j^\ell = 0 \\ \quad \vdots \\ \sum_{\ell=1}^{w_{\text{dec}}} \sum_{j=1}^{k_{\text{signe}}} \sum_{i=1}^{m} h_{mk_{\text{signe}}-n_{\text{dec}},(i-1)k_{\text{signe}}+j}^{\text{Mat}} x_i^\ell y_j^\ell = 0 \end{cases} \tag{9.4}$$

Restreindre le nombre de solutions. Nous avons résolu le système (9.4) avec des techniques de type base de Gröbner implantées en Magma. Afin d'accélérer la résolution nous avons comme dans le cas de `RankSign` restreint le nombre de solutions en rajoutant des équations à (9.4) venant de la structure vectorielle de F et de l'ensemble de solutions.

Avec nos notations nous pouvons voir F comme un sous-espace $\mathbb{F}_q$-linéaire généré par la matrice :

$$\begin{pmatrix} x_1^1 & \cdots & x_m^1 \\ x_1^2 & \cdots & x_m^2 \\ \vdots & & \vdots \\ x_1^{w_{\text{dec}}} & \cdots & x_m^{w_{\text{dec}}} \end{pmatrix}$$

Table 9.2 – Attaque contre les paramètres de l'IBE de [Gab+17b]

Sécurité [Gab+17b]	$(n_{\text{signe}}, k_{\text{signe}}, m, w_{\text{dec}}, n_{\text{dec}}, k_{\text{dec}}, q)$	Temps	Mémoire maximale
128 bits	$(100, 80, 96, 4, 96, 9, 2^{192})$	626s	1.7 GB

De cette façon nous pouvons mettre cette matrice sous forme systématique (elle génèrera le même espace). Donc nous pouvons ajouter les équations

$$\forall (i,j) \in [\![1, w_{\text{dec}}]\!]^2,\ j \neq i, \quad x_i^j = 0 \quad \text{et} \quad x_i^i = 1 \tag{9.5}$$

sans modifier l'ensemble des solutions de poids w_{dec} avec une bonne probabilité. L'intérêt de cet ajout est que dans les solutions de notre système il n'y aura qu'une base de F et non toutes. De plus, rappelons que l'ensemble de solutions est un espace $\mathbb{F}_q$-linéaire de dimension $\geqslant n_{\text{dec}} - (n_{\text{signe}} - k_{\text{signe}})w_{\text{dec}}$. Donc nous pouvons supposer que pour $I \subseteq [\![1, k_{\text{signe}}]\!] \times [\![1, w_{\text{dec}}]\!]$ aléatoire et de taille $n_{\text{dec}} - (n_{\text{signe}} - k_{\text{signe}}) - 1$ il existe une solution telle que :

$$\forall (j,i) \in I,\ y_j^i = 0 \quad \text{et} \quad y_{j_0}^{i_0} = 1 \text{ pour } (i_0, j_0) \notin I. \tag{9.6}$$

Les équations (9.5) et (9.6) nous ont alors permis de réduire le nombre de variables (ou augmenter le nombre d'équations) du système bilinéaire que nous considérons. La proposition qui suit résume alors la situation :

Proposition 9.2. *Nous obtenons après les éliminations des variables* (9.5) *et* (9.6) *dans le système* (9.4) *:*

— $mk_{\text{signe}} + w_{\text{dec}}^2 + (n_{\text{signe}} - k_{\text{signe}})$ *équations,*

— $mw_{\text{dec}} + k_{\text{signe}}w_{\text{dec}}$ *inconnues.*

Nous en déduisons que dans le "régime typique" de [Gab+17b] où $m \approx n_{\text{signe}} \approx k_{\text{signe}}$ et $w_{\text{dec}} \approx n_{\text{signe}}^{\varepsilon}$ pour $\varepsilon \in]0,1[$, nous avons un nombre d'équations n_{signe}^2 et un nombre d'inconnues de l'ordre de $n_{\text{signe}}^{1+\varepsilon}$. Nous sommes donc typiquement dans un cas où les calculs de base de Gröbner se feront en temps sous-exponentiel.

9.1.3 Résultats numériques

Nous donnons dans la table 9.2 nos résultats numériques pour retrouver des mots de poids w_{dec} dans les instances du problème RSL avec les paramètres considérés dans [Gab+17b]. Ces résultats ont été obtenus avec Magma et l'utilisation d'un processeur Intel Core i5, cadencé à 1.6 GHz à un seul cœur et avec 8 Go de RAM. Dans notre implémentation nous vérifions que les mots de poids w_{dec} révèlent bien l'espace F généré comme étant le secret.

9.1.4 Éviter l'attaque

Bien que notre attaque casse les paramètres proposés dans [Gab+17b] nous allons ici montrer qu'il existe en principe un ensemble de paramètres l'évitant.

Commençons tout d'abord par rappeler qu'il existe génériquement trois contraintes sur les paramètres de l'IBE [Gab+17b].

Une contrainte sur la signature. La signature utilisée est de type hache et signe. Cette dernière consiste pour signer un message $\mathbf{m}$ à décoder son haché à distance w_{signe} d'un $[n_{\text{signe}}, k_{\text{signe}}]_{q^m}$-code. Le paramètre w_{signe} doit donc vérifier :

$$w_{\text{rGV}}(q, m, n_{\text{signe}}, k_{\text{signe}}) \leqslant w_{\text{signe}} \leqslant \frac{m(n_{\text{signe}} - k_{\text{signe}})}{\max(m, n_{\text{signe}})} \tag{9.7}$$

La borne inférieure (avec la distance de Gilbert-Varshamov) est une condition nécessaire pour qu'en presque tout mot il existe un mot de code à distance w_{signe}. La borne supérieure assure quant à elle que le problème de décodage considéré n'est pas génériquement facile (il s'agit du poids typique renvoyé par l'adaptation de l'algorithme de Prange [GRS16] en métrique rang).

Une contrainte sur le chiffrement. Le chiffrement utilisé dans l'IBE est `RankPKE` (voir §1.3.5) avec un code $\mathcal{C}_{\text{dec}}$ de longueur n_{dec} et de dimension k_{dec}. Rappelons que le déchiffrement consiste à décoder $\mathcal{C}_{\text{dec}}$ à distance $w_{\text{dec}} w_{\text{signe}}$. Or pour que le déchiffrement ait un sens il est nécessaire que la solution soit unique, *i.e* :

$$w_{\text{signe}} w_{\text{dec}} \leqslant w_{\text{rGV}}(q, m, n_{\text{dec}}, k_{\text{dec}}) \tag{9.8}$$

Une contrainte pour éviter notre attaque. Il suffit de choisir les paramètres tels que :

$$w_{\text{dec}}(n_{\text{signe}} - k_{\text{signe}}) \geqslant n_{\text{dec}} \tag{9.9}$$

L'ensemble des paramètres admissibles avec ces contraintes est alors non vide. Par exemple, si nous disposons d'une signature atteignant la borne (9.7), à savoir $w_{\text{signe}} = w_{\text{rGV}}(q, m, n_{\text{signe}}, k_{\text{signe}})$ nous pouvons choisir :

$$n_{\text{signe}} = 100, \quad k_{\text{signe}} = 75, \quad n_{\text{dec}} = 96, \quad k_{\text{dec}} = 4 \quad \text{et} \quad w_{\text{dec}} = 4.$$

Plus généralement, si nous souhaitons donner des paramètres à l'IBE de [Gab+17b] nous proposons de procéder de la façon suivante. Nous choisissons tout d'abord :

$$m = n_{\text{signe}}.$$

Nous prenons ensuite le paramètre k_{signe} tel que $\frac{w_{\text{rGV}}(q,m,n_{\text{signe}},k_{\text{signe}})}{n_{\text{signe}}-k_{\text{signe}}}$ est suffisamment petit (au pire de l'ordre de $1/2$). Nous supposons alors disposer d'un $[n_{\text{signe}}, k_{\text{signe}}]_{q^m}$-code muni d'un algorithme de signature à distance :

$$w_{\text{signe}} = (1 - \varepsilon)(n_{\text{signe}} - k_{\text{signe}}) \tag{9.10}$$

pour un certain ε. Nous choisissons ensuite un $[n_{\text{dec}}, k_{\text{dec}}]_{q^m}$-code de dimension suffisamment petite pour que :

$$w_{\text{rGV}}(q, m, n_{\text{dec}}, k_{\text{dec}}) \geqslant (1 - \varepsilon) n_{\text{dec}}$$

ce qui en principe est possible. Nous pouvons alors prendre w_{dec} vérifiant (9.8) tel que $w_{\text{signe}} w_{\text{dec}} \geqslant (1 - \varepsilon) n_{\text{dec}}$. De cette façon les deux contraintes (9.7) et (9.8) sont satisfaites. La contrainte (9.9) est quant à elle vérifiée car :

$$\begin{aligned} w_{\text{dec}}(n_{\text{signe}} - k_{\text{signe}}) &= \frac{w_{\text{signe}} w_{\text{dec}}}{1 - \varepsilon} && \text{(nous utilisons (9.10))} \\ &\geqslant \frac{n_{\text{dec}}(1 - \varepsilon)}{1 - \varepsilon} && \text{(d'après le choix particulier de } w_{\text{dec}}) \\ &= n_{\text{dec}} \end{aligned}$$

9.1.5 Comparaison avec les précédentes attaques contre RSL

Rappelons que le problème RSL pour les paramètres n, k, N, w peut s'exprimer de la façon suivante : nous avons accès à une matrice $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ de rang plein et N syndromes $\mathbf{eH}^\intercal$ pour $\mathbf{e} \hookleftarrow F^n$ où F est un sous-espace de $\mathbb{F}_{q^m}$ secret et de dimension w. Le problème consiste alors à retrouver F. Quand $N = 1$ il s'agit tout simplement de RSD (voir le problème 1.9) qui est le problème difficile sur lequel repose usuellement la cryptographie en métrique rang. En revanche, il est clair que la difficulté du problème diminue quand N augmente. La question naturelle est alors "à quel point N peut-il être supérieur à un tout en étant sûr que le problème reste difficile ?" Une première réponse à cette question fut donnée dans [Gab+17b, §4, p14] où il a été démontré que N doit vérifier :

$$N < wn \tag{9.11}$$

afin d'éviter une attaque polynomiale. Ici nous avons encore plus contraint N pour éviter une attaque,

$$N \leqslant wk \quad (k < n) \tag{9.12}$$

où k est la dimension du code de matrice de parité $\mathbf{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ utilisée dans une instance de RSL. Cette condition est alors plus forte que la précédente étant donné que nous avons toujours $k < n$. Cela eut un coût, N est certes plus contraint que précédemment mais notre attaque est sous-exponentielle et non polynomiale dès lors que (9.12) n'est pas vérifiée.

En revanche, dans le contexte de l'IBE [Gab+17b] notre nouvelle contrainte est significativement plus forte que la précédente. Cela provient du fait que sous l'hypothèse raisonnable 6 qui suit nous nous attendons à avoir $k \ll n$ qui se traduit dans l'IBE par $(n_{\text{signe}} - k_{\text{signe}}) \ll n_{\text{signe}}$. Or pour éviter notre attaque il doit être vérifié $n_{\text{dec}} \leqslant w_{\text{dec}}(n_{\text{signe}} - k_{\text{signe}})$.

Hypothèse 6. *Les paramètres m, k_{signe} qui sont des fonctions de n_{signe} vérifient :*

$$\frac{k_{\text{signe}}}{n_{\text{signe}}} = \Omega(1) \tag{9.13}$$

$$m = \Theta(n_{\text{signe}}) \tag{9.14}$$

$$w_{\text{dec}} \underset{n_{\text{signe}}\to+\infty}{\longrightarrow} +\infty \tag{9.15}$$

Cette hypothèse est minimale dans le cas de l'IBE de [Gab+17b] comme nous allons l'expliquer.

Les équations (9.13) et (9.14) nous assurent que $\frac{w_{\text{rGV}}(q,m,n_{\text{signe}},k_{\text{signe}})}{n_{\text{signe}}-k_{\text{signe}}}$ ne tend pas vers 1 ce qui est essentiel comme expliqué dans la sous-section précédente.

L'équation (9.15) permet d'éviter une attaque polynomial contre RSL. En effet, supposons que w_{dec} est bornée, *i.e* : $w_{\text{dec}} = O(1)$. Rappelons que dans l'IBE, les instances du problème RSL ont la forme suivante $(\mathbf{G}_{\text{signe}}, \mathbf{G}_{\text{signe}}\mathbf{E})$ où $\mathbf{G}_{\text{signe}} \in \mathbb{F}_{q^m}^{k_{\text{signe}}\times n_{\text{signe}}}$ et $\mathbf{E}$ est homogène de poids w_{dec} et d'espace associé F. Si nous résolvons le problème de décodage RSD avec comme syndrome la première colonne de $\mathbf{G}_{\text{signe}}\mathbf{E}$ et la matrice $\mathbf{G}_{\text{signe}}$ nous retrouverons avec une très bonne probabilité l'espace F du fait que w_{dec} est inférieur à la distance de Gilbert-Varshamov du code. Écrivons alors ce problème comme un système bilinéaire sur $\mathbb{F}_q$, nous obtiendrons alors :

1. $n_{\text{signe}} w_{\text{dec}}$ inconnues,
2. mk_{signe} équations.

Dans ce cas, sous les hypothèses (9.13), (9.14) et le fait que $k_{\text{signe}} \leqslant n_{\text{signe}}$, nous avons $mk_{\text{signe}} = \Theta(n_{\text{signe}}k_{\text{signe}}) = \Theta(n_{\text{signe}}^2)$ équations. Les inconnues sont quant à elles de l'ordre de $O(n_{\text{signe}})$ car $w_{\text{dec}} = O(1)$. Nous sommes donc dans un régime de paramètres où les

résolutions par base de Gröbner sont de complexité attendue polynomiale. Nous pouvons donc raisonnablement supposer que w_{dec} tend vers l'infini.

L'hypothèse 6 mène alors comme prévu à la proposition suivante.

Proposition 9.3. *Sous l'hypothèse 6, nous avons pour n_{signe} tendant vers l'infini :*

$$n_{\text{signe}} - k_{\text{signe}} = o(n_{\text{signe}}).$$

Démonstration de la proposition 9.3

D'après la proposition 1.17 nous obtenons,

$$w_{\text{signe}} w_{\text{dec}} \leqslant \min(n_{\text{dec}}, m)$$

et ainsi d'après l'hypothèse 6,

$$w_{\text{signe}} w_{\text{dec}} = O(n_{\text{signe}})$$

ce qui donne

$$\frac{w_{\text{signe}}}{n_{\text{signe}}} = O\left(\frac{1}{w_{\text{dec}}}\right). \tag{9.16}$$

Encore une fois d'après 6 nous savons que w_{dec} tend vers l'infini. De cette façon, nous avons :

$$w_{\text{signe}} = o(n_{\text{signe}}). \tag{9.17}$$

Maintenant sous la contrainte venant de la signature (voir (9.7)) nous avons que $w_{\text{signe}} \geqslant w_{\text{rGV}}(q, m, n_{\text{signe}}, k_{\text{signe}})$. On peut alors vérifier avec l'équation (1.23) et l'hypothèse 6, tout particulièrement (9.14), que la dernière inégalité et (9.17) impliquent $n_{\text{signe}} - k_{\text{signe}} = o(n_{\text{signe}})$ ce qui conclut la preuve de la proposition. □

9.2 Attaque contre l'IBE en métrique de Hamming

L'objet de cette section est de montrer qu'il existe une faiblesse fatale dans l'IBE de [Gab+17b] *en métrique de Hamming*. Nous allons en effet démontrer que les contraintes de la proposition 1.17 et une hypothèse raisonnable impliquent que l'IBE est attaqué en temps polynomial par l'algorithme de Prange [Pra62] que nous avons décrit dans §2.1.

Rappelons que pour l'IBE [Gab+17b] en métrique de Hamming (décrit dans §1.4.3) une donnée publique est $\mathbf{G}_{\text{signe}} \in \mathbb{F}_2^{k_{\text{signe}}}$ et $\mathbf{G}_{\text{signe}}\mathbf{E}$ où les colonnes de $\mathbf{E}$ sont secrètes et toutes de poids w_{dec}. Nous allons alors appliquer l'algorithme de Prange pour résoudre le problème de décodage :

— Étant donné $\mathbf{G}_{\text{signe}} \in \mathbb{F}_2^{k_{\text{signe}} \times n_{\text{signe}}}$ et $\mathbf{s} \in \mathbb{F}_2^{k_{\text{signe}}}$ tel qu'il existe $\mathbf{e}$ de poids w_{dec} pour lequel $\mathbf{s} = \mathbf{e}\mathbf{G}_{\text{signe}}^{\intercal}$, retrouver $\mathbf{e}$.

Nous retrouverons dans ce cas les colonnes de $\mathbf{E}$ et donc $\mathbf{E}$. Le schéma est alors cassé dès-lors que $\mathbf{E}$ est récupéré. En effet, tout attaquant connaît les messages chiffrés $\mathcal{H}(id)\mathbf{E} + \mathbf{m}\mathbf{G}_{\mathcal{C}_{\text{dec}}}$, la matrice $\mathbf{G}_{\text{dec}}$ et le haché de l'identité du destinataire. De cette façon ce-dernier retrouve facilement $\mathbf{m}\mathbf{G}_{\text{dec}}$ et donc $\mathbf{m}$.

La complexité du problème de décodage que nous venons d'énoncer avec l'algorithme de Prange est donnée à un facteur polynomial près par (voir 2.3) :

$$\frac{\binom{n_{\text{signe}}}{w_{\text{dec}}}}{\binom{k_{\text{signe}}}{w_{\text{dec}}}}. \tag{9.18}$$

Rappelons alors que cette-dernière est en n_{signe} si $k_{\text{signe}}/n_{\text{signe}}$ est fixé :

— exponentielle si $w_{\text{dec}} = \Theta(n_{\text{signe}})$,

— sous-exponentielle si $w_{\text{dec}} = o(n_{\text{signe}})$.

Or nous avons prouvé dans la proposition 1.17 que les paramètres de l'IBE en métrique de Hamming doivent nécessairement vérifier :

$$w_{\text{signe}} w_{\text{dec}} = O(n_{\text{signe}}).$$

Maintenant le paramètre w_{signe} ne peut pas être non plus trop petit car autrement les algorithmes de décodage génériques permettraient de forger une signature en temps polynomial, c'est à dire dans l'IBE retrouver $\mathbf{u}_{id}$ tel que :

$$|\mathbf{u}_{id}\mathbf{G}_{\mathcal{C}_{\text{signe}}} - \mathcal{H}(id)| = w_{\text{signe}}.$$

Il est donc nécessaire dans l'IBE que les paramètres w_{dec} et w_{signe} soient sous-linéaires en n_{signe}. Nous allons alors faire l'hypothèse que ces derniers le sont de la façon la plus faible qu'il soit.

Hypothèse 7.

$$w_{\text{dec}} = \Omega(n_{\text{signe}}^{\varepsilon}) \quad \textit{pour un certain} \quad \varepsilon > 0 \tag{9.19}$$

$$w_{\text{signe}} = \Omega(n_{\text{signe}}^{\varepsilon'}) \quad \textit{pour un certain} \quad \varepsilon' > 0 \tag{9.20}$$

Proposition 9.4. *Sous l'hypothèse 7, l'algorithme de Prange casse l'*IBE *[Gab+17b] en métrique de Hamming en temps polynomial en* n_{signe}.

Démonstration de la proposition 9.4.

Rappelons que les paramètres de l'IBE en métrique de Hamming sont $n_{\text{signe}}, k_{\text{signe}}, w_{\text{signe}}$. Par soucis de clarté, posons :

$$n \triangleq n_{\text{signe}}, \quad k \triangleq k_{\text{signe}} \quad \text{et} \quad w \triangleq w_{\text{signe}}.$$

En utilisant l'hypothèse 7 avec $ww_{\text{dec}} = O(n)$ nous obtenons :

$$\begin{aligned} w_{\text{dec}} &= O(n^{1-\varepsilon'}), \\ w_{\text{signe}} &= O(n^{1-\varepsilon}). \end{aligned}$$

Nous allons maintenant calculer l'exposant en base 2 de (9.18). D'après le lemme 1.2, ce-dernier est égal à :

$$\log_2 \binom{n}{w_{\text{dec}}} - \log_2 \binom{k}{w_{\text{dec}}} = nh\left(\frac{w_{\text{dec}}}{n}\right) - kh\left(\frac{w_{\text{dec}}}{k}\right) + O(\log_2(n))$$

où h est l'entropie binaire définie comme :

$$h(x) \triangleq -x\log_2 x - (1-x)\log_2(1-x).$$

De cette façon pour prouver que l'algorithme de Prange est polynomial en n il suffit de démontrer que :

$$nh\left(\frac{w_{\text{dec}}}{n}\right) - kh\left(\frac{w_{\text{dec}}}{k}\right) \tag{9.21}$$

est un $O(\log_2 n)$. Rappelons maintenant que pour une signature de type hache et signe utilisant des codes, le poids de décodage doit être plus grand que la borne de Gilbert-Varshamov :

$$w \geqslant nh^{-1}\left(1 - \frac{k}{n}\right) \iff \frac{k}{n} \geqslant 1 - h\left(\frac{w}{n}\right).$$

Du fait que $w = O\left(n^{1-\varepsilon}\right)$, $k \leqslant n$ et $h(x) \underset{x\to 0}{=} O\left(-x\log_2 x\right)$ nous obtenons,

$$\frac{k}{n} = 1 + O\left(-\frac{w}{n}\log_2\frac{w}{n}\right) \tag{9.22}$$

et $k \sim n$. Nous sommes maintenant prêts à montrer que (9.21) est un $O(\log_2 n)$.
Comme $k \sim n$, $w_{\text{dec}} = O(n^{1-\varepsilon'})$ et :

$$h(x) \underset{x\to 0}{=} -x\log_2 x + \frac{1}{\ln(2)}\left(x - \sum_{\ell=2}^{p-1}\frac{x^\ell}{\ell(\ell-1)} + O(x^p)\right)$$

pour un entier p plus grand que $\frac{1}{\varepsilon'}$ nous avons :

$$nh\left(\frac{w_{\text{dec}}}{n}\right) - kh\left(\frac{w_{\text{dec}}}{k}\right) = a(n) + b(n) + c(n) \tag{9.23}$$

où :

$$a(n) \triangleq k\frac{w_{\text{dec}}}{k}\log_2\frac{w_{\text{dec}}}{k} - n\frac{w_{\text{dec}}}{n}\log_2\frac{w_{\text{dec}}}{n}$$

$$b(n) \triangleq n\frac{w_{\text{dec}}}{\ln(2)n} - k\frac{w_{\text{dec}}}{\ln(2)k} + \frac{1}{\ln(2)}\sum_{\ell=2}^{p-1} k\frac{(w_{\text{dec}}/k)^\ell}{\ell(\ell-1)} - n\frac{(w_{\text{dec}}/n)^\ell}{\ell(\ell-1)} \tag{9.24}$$

et :

$$c(n) \triangleq nO\left(\frac{w_{\text{dec}}^p}{n^p}\right) + kO\left(\frac{w_{\text{dec}}^p}{k^p}\right).$$

Nous obtenons alors facilement :

$$\begin{aligned} c(n) &= O\left(\frac{w_{\text{dec}}^p}{n^{p-1}}\right) \\ &= O\left(\frac{1}{n^{p\varepsilon'-1}}\right) \quad (\text{car } w_{\text{dec}} = O(n^{1-\varepsilon'})) \\ &= o(1) \quad (\text{du fait que } p > 1/\varepsilon') \end{aligned}$$

Calculons maintenant $a(n)$.

$$\begin{aligned} a(n) &= k\frac{w_{\text{dec}}}{k}\log_2\frac{w_{\text{dec}}}{k} - n\frac{w_{\text{dec}}}{n}\log_2\frac{w_{\text{dec}}}{n} \\ &= w_{\text{dec}}\log_2\frac{w_{\text{dec}}}{k} - w_{\text{dec}}\log_2\frac{w_{\text{dec}}}{n} \\ &= -w_{\text{dec}}\log_2\frac{k}{n} \\ &= -w_{\text{dec}}\log_2\left(1 + O\left(\frac{w}{n}\log_2\frac{w}{n}\right)\right) \quad (\text{d'après (9.22)}) \end{aligned}$$

Rappelons maintenant que $w = O(n^{1-\varepsilon})$. Donc en utilisant $\log_2(1+x) \underset{x\to 0}{=} O(x)$ nous obtenons :

$$a(n) = -w_{\text{dec}}O\left(\frac{w}{n}\log_2\frac{w}{n}\right)$$

ce qui donne comme $ww_{\text{dec}} = O(n)$ et $w = O(n^{1-\varepsilon})$ et donc :

$$a(n) = O\left(\log_2\frac{w}{n}\right) = O\left(\log_2 n\right) \tag{9.25}$$

Calculons maintenant $b(n)$ défini dans (9.24).

$$\begin{aligned} b(n) &= n\frac{w_{\text{dec}}}{\ln(2)n} - k\frac{w_{\text{dec}}}{\ln(2)k} + \frac{1}{\ln(2)}\sum_{l=2}^{p-1} k\frac{(w_{\text{dec}}/k)^\ell}{\ell(\ell-1)} - n\frac{(w_{\text{dec}}/n)^\ell}{\ell(\ell-1)} \\ &= \frac{1}{\ln(2)}\sum_{l=2}^{p-1} w_{\text{dec}}^\ell\frac{1}{n^{\ell-1}}\frac{(k/n)^{1-\ell}-1}{\ell(\ell-1)} \end{aligned}$$

Or rappelons que,

$$\frac{k}{n} = 1 + O\left(-\frac{w}{n}\log_2\frac{w}{n}\right)$$

d'où,

$$\begin{aligned}\left(\frac{k}{n}\right)^{1-\ell} - 1 &= \left(1 + O\left(-\frac{w}{n}\log_2\frac{w}{n}\right)\right)^{1-\ell} - 1 \\ &= 1 + O\left(-\frac{w}{n}\log_2\frac{w}{n}\right) - 1 \\ &= O\left(-\frac{w}{n}\log_2\frac{w}{n}\right).\end{aligned}$$

Ainsi nous obtenons,

$$\begin{aligned}b(n) &= \frac{1}{\ln(2)}\sum_{\ell=2}^{p-1}\frac{w_{\text{dec}}^{\ell}}{n^{\ell-1}}\frac{1}{l(l-1)}O\left(-\frac{w}{n}\log_2\frac{w}{n}\right) \\ &= \frac{1}{\ln(2)}\sum_{\ell=2}^{p-1}\frac{w_{\text{dec}}^{\ell-1}}{\ell(\ell-1)n^{\ell-1}}O\left(-\frac{w_{\text{dec}}w}{n}\log_2\frac{w}{n}\right) \\ &= \frac{1}{\ln(2)}\sum_{\ell=2}^{p-1}o(1)O\left(-\log_2\frac{w}{n}\right) \\ &\quad \text{(car } w_{\text{dec}} = O(n^{1-\varepsilon'}) = o(n) \text{ comme } \varepsilon' > 0 \text{ et } w_{\text{dec}}w = O(n)\text{)}. \\ &= o\left(-\log_2\frac{w}{n}\right) \\ &= O\left(\log_2 n\right).\end{aligned}$$

De cette façon, en combinant ce résultat avec les équations (9.23) et (9.25) nous obtenons :

$$kh\left(\frac{w_{\text{dec}}}{k}\right) - nh\left(\frac{w_{\text{dec}}}{n}\right) = O\left(\log_2 n\right)$$

ce qui conclut la preuve. □

Conclusion et perspectives

Le problème du décodage générique est, comme nous l'avons vu tout au long de cette thèse, fondamental en cryptographie utilisant des codes correcteurs. Son étude à travers les meilleurs algorithmes le résolvant ou encore son nombre attendu de solutions en fonction des paramètres est donc essentielle. Nous nous sommes attelés dans la partie I de ce document à rappeler l'état de l'art concernant ces questions ainsi que présenter certains cryptosystèmes (en métrique de Hamming et rang). Nous avons présenté dans le chapitre 2 les algorithmes de décodage par ensemble d'information usuellement considérés pour le décodage générique en petite distance.

Ces rappels de la partie I ont été cruciaux pour notre étude du décodage statistique dans le chapitre 4. En effet, nous avons proposé des algorithmes inspirés de ceux par ensemble d'information pour calculer des équations de parité d'un code aléatoire. De plus, ils nous ont fourni une comparaison avec l'approche du décodage statistique dont nous avons donné la complexité asymptotique à l'aide d'une nouvelle étude asymptotique des polynômes de Krawtchouk dans le chapitre 4. Or comme nous l'avons vu, pour des paramètres cryptographiques, le décodage statistique ne peut faire mieux que l'algorithme de Prange.

Nos rappels sur les décodages par ensemble d'information nous ont aussi donné un repère pour le chapitre 3 où nous les généralisons au cas du décodage en grande distance. Nous avons alors montré que pour ces paramètres, le problème du décodage est au moins aussi difficile que son pendant en petite distance, ouvrant de nouvelles perspectives pour la cryptographie utilisant des codes correcteurs. Cette étude, pouvant sembler au premier abord non naturelle, est essentielle pour assurer la sécurité (ainsi que donner des paramètres concrets) du schéma de signature Wave que nous avons décrit tout au long de la partie III. Cette primitive cryptographique est la première à s'inscrire dans le cadre de l'approche de GPV [GPV08b] avec des codes correcteurs. Cette dernière est à ce jour la seule signature sûre avec Durandal [Ara+19a] et le protocole de Stern [Ste93b] utilisant des codes et étant efficace. Pour cela nous avons introduit une nouvelle trappe en cryptographie avec des codes : les codes $(U, U+V)$-généralisés permutés. Nous avons alors montré dans le chapitre 5 comment utiliser cette structure avec des codes U et V aléatoires pour résoudre le décodage en grande distance là où ce-dernier semble génériquement difficile. De plus, nous avons introduit une méthode de rejet extrêmement efficace pour que notre algorithme produise des signatures (prouvées) uniformément distribuées dans leur domaine. Nous avons ensuite montré dans le chapitre 6 que Wave est sûr dans le modèle de l'oracle aléatoire sous l'hypothèse que DOOM en grande distance est difficile ainsi que de distinguer un code $(U, U+V)$ généralisé permuté d'un code aléatoire. Les problèmes auxquels la sécurité de Wave se réduisent étant clairement établis, nous avons proposé un algorithme pour distinguer les codes utilisés dans la trappe de codes aléatoires. Le problème DOOM en grande distance est quant à lui étudié dans le chapitre 3. De plus, nous avons montré que le problème décisionnel sous-jacent au distingueur des codes $(U, U+V)$-généralisés permutés est NP-complet. De cette façon la sécurité de notre schéma à trappe utilisant des codes repose sur deux problèmes NP-complets, ce qui est au mieux

de nos connaissances une première.

Ce document se conclut par deux cryptanalyses en métrique rang contre `RankSign` [Gab+14b] et l'IBE [Gab+17a]. Ces deux attaques sont algébriques et utilisent entre autres des méthodes de résolution par base de Gröbner. Dans le cas de `RankSign` nous avons montré que le code LRPC sous-jacent à l'algorithme de signature contenait des mots de poids faible dû aux contraintes de la signature de type hache et signe et du décodeur par effacement utilisé. Dans le cas de l'IBE nous avons attaqué avec des techniques similaires les paramètres proposés lors de l'instanciation du problème RSL. Nous avons cependant montré qu'il était toujours possible de choisir les paramètres du système de façon à éviter notre attaque. En revanche ceci n'est pas possible pour notre adaptation de l'IBE en métrique de Hamming où nous montrons qu'une simple utilisation de l'algorithme de Prange casse le système. Ceci s'explique par les contraintes imposées génériquement par les algorithmes de chiffrement d'Alekhnovich et signature utilisés.

Nous souhaitons désormais conclure par une liste de quelques perspectives et problèmes ouverts soulevés lors du développement de cette thèse et qui nous semblent pertinents.

Perspectives et problèmes ouverts

i (Chapitre 1) : Nous avons donné une borne fonction de w des distances statistiques $\rho(\mathbf{e}\mathbf{H}^\intercal, \mathbf{s})$ pour $\mathbf{e} \hookleftarrow S_w$ et $\mathbf{s} \hookleftarrow \mathbb{F}_q^{n-k}$ en moyennant sur les matrices de parité $\mathbf{H}$ (et donc des codes). Or comme nous l'avons expliqué dans la remarque 1.3 il s'agit d'une version plus faible (avec des codes) du *smoothing parameter* des réseaux euclidiens. Nous souhaiterions donc un résultat similaire en donnant une borne sur la distance statistique $\rho(\mathbf{e}\mathbf{H}^\intercal, \mathbf{s})$ pour toute matrice fixée $\mathbf{H}$ et non en moyenne.

ii (Chapitre 3) : Le schéma de signature Wave est le premier, et actuellement seul, cryptosystème utilisant des codes avec un décodage en grande distance. Or comme nous l'avons vu ces paramètres font du décodage, dans l'état de l'art, un problème bien plus difficile que son pendant en petite distance. Une question naturelle et ouverte est donc de concevoir de nouveaux cryptosystèmes utilisant le décodage en grande distance comme par exemple des chiffrements *à la Alekhnovich* ou encore *à la McEliece* avec des codes à trappe. Ce dernier point peut sembler surprenant étant donné que le concept de décodage stricto-sensu n'est pas défini en grande distance. Ceci peut pourtant avoir du sens car pour tout poids relatif $\omega > w_{\text{GV+}}$ nous nous attendons typiquement à une unique solution par syndrome produit à partir d'une erreur de poids $w \triangleq \lfloor \omega n \rfloor$.

iii (Chapitre 5) : L'algorithme 3 de décodage de Wave utilise des distributions internes $\mathcal{D}_V$ et $\mathcal{D}_U^t$. Ces dernières permettent de minimiser le nombre de rejets. Nous n'avons cependant pas démontré qu'il existait un choix de ces distributions permettant asymptotiquement un nombre de rejets polynomial en n.

iv (Chapitre 5) : Nous avons dans §5.3.4 décrit comment choisir les distributions $\mathcal{D}_V$ et $\mathcal{D}_U^t$ de l'algorithme 3 pour obtenir un nombre de rejets proche de 0 pour un choix de paramètres fixé. Notre technique a consisté à choisir une distribution de Laplace tronquée avec espérance et variance bien choisies. Notre approche a été heuristique et nous avons démontré qu'elle s'avérait efficace, *i.e* : menant à un rejet toutes les 100 signatures. La question naturelle est alors peut-on choisir les distributions de façon à faire un rejet toutes les 2^λ signatures ? Ceci nous permettrait d'enlever la méthode du rejet dans une implémentation.

v (Chapitre 6) : Nous avons montré que le problème de décider si un code est un $(U, U+V)$-généralisé permuté ou non est NP-complet par une réduction polynomiale au problème du mariage Tri-dimensionnel. En revanche cette réduction s'est faite dans le cas binaire ($q = 2$) pour les codes uniquement $(U, U+V)$ et pour un régime

de paramètres $\dim(U) \triangleq k_U < \dim(V) \triangleq k_V$. Or dans le cas de Wave nous sommes dans un autre régime de paramètres, à savoir $q > 2$ et $k_U > k_V$. En revanche, du fait que $q > 2$, nous avons une plus grande liberté de choix de codes dans la trappe (d'où notre généralisation des codes $(U, U + V)$) ce qui offre sûrement une plus grande liberté pour une réduction. Ceci nous amène donc à penser qu'une réduction au cas où justement $k_U > k_V$ est possible.

vi (Chapitre 7) : Les meilleurs algorithmes que nous avons trouvé pour distinguer un code $(U, U+V)$-généralisé permuté d'un code aléatoire utilise une recherche de mots de poids plus faible que prévu. Ici les codes U et V étant aléatoires nos algorithmes utilisent comme brique élémentaire les algorithmes de décodage générique. La question étant maintenant peut-on réduire le problème de distinguer un code $(U, U + V)$-généralisé permuté d'un code aléatoire au problème du décodage générique ?

vii : Terminons désormais cette liste de problèmes ouverts par une perspective. Actuellement le schéma de signature le plus proposé avec des réseaux euclidiens au processus de standardisation du NIST est celui de Lyubashevsky [Lyu09b], lui-même inspiré de celui de Schnorr [Sch91]. Le cryptosystème de [Lyu09b] utilise le protocole de [Sch91] avec l'équivalent du décodage générique en réseaux euclidiens. Une méthode de rejet est proposée afin de produire des signatures ne faisant pas fuiter d'information. Nous pouvons alors trivialement adapter ce système avec des codes. Le problème alors majeur est justement celui de la fuite d'information où une méthode de rejet efficace est impossible à instancier directement. Peut-on donc trouver une façon d'adapter ce schéma avec par exemple une trappe supplémentaire (comme des codes structurés) ? Il est à noter que la signature Durandal [Ara+19a] a commencé à répondre à cette question en adaptant le schéma de [Lyu09b] en métrique rang. Une méthode est alors proposée afin d'éviter que les signatures ne fassent fuiter de l'information mais relativement à certaines attaques. La méthode employée ne permet malheureusement pas de prouver directement que les signatures ne font pas fuiter d'information.

Table des figures

Bibliographie

[AGS11] Carlos AGUILAR, Philippe GABORIT et Julien SCHREK. “A new zero-knowledge code based identification scheme with reduced communication”. In : *Proc. IEEE Inf. Theory Workshop- ITW 2011*. IEEE, oct. 2011, p. 648–652.

[Ale11] Michael ALEKHNOVICH. “More on Average Case vs Approximation Complexity”. In : *Computational Complexity* 20.4 (2011), p. 755–786.

[AM+18] Carlos AGUILAR MELCHOR, Olivier BLAZY, Jean-Christophe DENEUVILLE, Philippe GABORIT et Gilles ZÉMOR. “Efficient Encryption From Random Quasi-Cyclic Codes”. In : *IEEE Trans. Inform. Theory* 64.5 (2018), p. 3927–3943.

[Ara+17a] N. ARAGON, P. BARRETO, S. BETTAIEB, Loic BIDOUX, O. BLAZY, J.-C. DENEUVILLE, P. GABORIT, S. GUERON, T. GÜNEYSU, C. AGUILAR MELCHOR, R. MISOCZKI, E. PERSICHETTI, N. SENDRIER, J.-P. TILLICH et G. ZÉMOR. *BIKE*. NIST Round 1 submission for Post-Quantum Cryptography. Nov. 2017.

[Ara+17b] Nicolas ARAGON, Philippe GABORIT, Adrien HAUTEVILLE et Jean-Pierre TILLICH. “Improvement of Generic Attacks on the Rank Syndrome Decoding Problem”. working paper or preprint. Oct. 2017.

[Ara+17c] Nicolas ARAGON, Philippe GABORIT, Adrien HAUTEVILLE, Oliver RUATTA et Gilles ZÉMOR. *RankSign -a signature proposal for the NIST’s call-*. first round submission to the NIST post-quantum cryptography call. NIST Round 1 submission for Post-Quantum Cryptography. Nov. 2017.

[Ara+18] Nicolas ARAGON, Olivier BLAZY, Philippe GABORIT, Adrien HAUTEVILLE et Gilles ZÉMOR. “Durandal : a rank metric based signature scheme”. In : *IACR Cryptology ePrint Archive* (2018).

[Ara+19a] Nicolas ARAGON, Olivier BLAZY, Philippe GABORIT, Adrien HAUTEVILLE et Gilles ZÉMOR. “Durandal : a rank metric based signature scheme”. In : *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*. T. 11478. LNCS. Springer, 2019, p. 728–758.

[Ara+19b] Nicolas ARAGON, Olivier BLAZY, Jean-Christophe DENEUVILLE, Philippe GABORIT, Adrien HAUTEVILLE, Olivier RUATTA, Jean-Pierre TILLICH, Gilles ZÉMOR, Carlos AGUILAR MELCHOR, Slim BETTAIEB, Loïc BIDOUX, Bardet MAGALI et Ayoub OTMANI. *ROLLO (merger of Rank-Ouroboros, LAKE and LOCKER)*. Second round submission to the NIST post-quantum cryptography call. NIST Round 2 submission for Post-Quantum Cryptography. Mar. 2019.

[Aru+19] Frank ARUTE, Kunal ARYA, Ryan BABBUSH, Dave BACON, Joseph C BARDIN, Rami BARENDS, Rupak BISWAS, Sergio BOIXO, Fernando GSL BRANDAO, David A BUELL et al. “Quantum supremacy using a programmable superconducting processor”. In : *Nature* 574.7779 (2019), p. 505–510.

[Arı09] Erdal ARIKAN. “Channel polarization : a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels”. In : *IEEE Trans. Inform. Theory* 55.7 (2009), p. 3051–3073.

[Aug+08] Daniel AUGOT, Matthieu FINIASZ, Philippe GABORIT, Stéphane MANUEL et Nicolas SENDRIER. “SHA-3 proposal : FSB”. In : *Submission to the SHA3 NIST competition,* 2008.

[Bab86] László BABAI. “On Lovász' lattice reduction and the nearest lattice point problem”. In : *Combinatorica* 6.1 (1986), p. 1–13.

[Bal+13] Marco BALDI, Marco BIANCHI, Franco CHIARALUCE, Joachim ROSENTHAL et Davide SCHIPANI. “Using LDGM Codes and Sparse Syndromes to Achieve Digital Signatures”. In : *Post-Quantum Cryptography 2013*. T. 7932. LNCS. Springer, 2013, p. 1–15.

[Bar+11] Boaz BARAK, Yevgeniy DODIS, Hugo KRAWCZYK, Olivier PEREIRA, Krzysztof PIETRZAK, François-Xavier STANDAERT et Yu YU. “Leftover Hash Lemma, Revisited”. In : *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*. 2011, p. 1–20.

[Bar+16] Magali BARDET, Julia CHAULET, Vlad DRAGOI, Ayoub OTMANI et Jean-Pierre TILLICH. “Cryptanalysis of the McEliece Public Key Cryptosystem Based on Polar Codes”. In : *Post-Quantum Cryptography 2016*. LNCS. Fukuoka, Japan, fév. 2016, p. 118–143.

[BC07] Marco BALDI et Franco CHIARALUCE. “Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC Codes”. In : *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. Nice, France, juin 2007, p. 2591–2595.

[BCJ11] Anja BECKER, Jean-Sébastien CORON et Antoine JOUX. “Improved Generic Algorithms for Hard Knapsacks”. In : *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*. 2011, p. 364–385.

[Bec+12] Anja BECKER, Antoine JOUX, Alexander MAY et 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*. LNCS. Springer, 2012.

[Bel+19] Emanuele BELLINI, Florian CAULLERY, Philippe GABORIT, Marc MANZANO et Víctor MATEU. “Improved Veron Identification and Signature Schemes in the Rank Metric”. In : *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2019*. T. abs/1903.10212. Paris, France : IEEE, juil. 2019, p. 1872–1876.

[Ber+09] Thierry P. BERGER, Pierre-Louis CAYREL, Philippe GABORIT et Ayoub OTMANI. “Reducing Key Length of the McEliece Cryptosystem”. In : *Progress in Cryptology - AFRICACRYPT 2009*. Sous la dir. de Bart PRENEEL. T. 5580. LNCS. Gammarth, Tunisia, 2009, p. 77–97.

[Bet+19] Slim BETTAIEB, Loïc BIDOUX, Philippe GABORIT et Etienne MARCATEL. “Preventing Timing Attacks Against RQC Using Constant Time Decoding of Gabidulin Codes”. In : *Post-Quantum Cryptography 2019*. Sous la dir. de Jintai DING et Rainer STEINWANDT. T. 11505. LNCS. Chongquing, China : Springer, mai 2019, p. 371–386.

[BF01] Dan BONEH et Matthew K. FRANKLIN. “Identity-based encryption from the Weil pairing”. In : *Advances in Cryptology - CRYPTO 2001*. T. 2139. LNCS. Springer, août 2001, p. 213–229.

[BFS99] Jonathan F. BUSS, Gudmund S. FRANDSEN et Jeffrey O. SHALLIT. “The Computational Complexity of Some Problems of Linear Algebra”. In : *J. Comput. System Sci.* 58.3 (juin 1999), p. 572–596.

[BGK17] Thierry P. BERGER, Cheikh Thiécoumba GUEYE et Jean Belo KLAMTI. “A NP-Complete Problem in Coding Theory with Application to Code Based Cryptography”. In : *Codes, Cryptology and Information Security - Second International Conference, C2SI 2017, Rabat, Morocco, April 10-12, 2017, Proceedings - In Honor of Claude Carlet*. 2017, p. 230–237.

[BM17] Leif BOTH et Alexander MAY. “Optimizing BJMM with Nearest Neighbors : Full Decoding in $2^{2/21n}$ and McEliece Security”. In : *WCC Workshop on Coding and Cryptography*. Sept. 2017.

[BM18] Leif BOTH et Alexander MAY. “Decoding Linear Codes with High Error Rate and Its Impact for LPN Security”. In : *Post-Quantum Cryptography 2018*. Sous la dir. de Tanja LANGE et Rainer STEINWANDT. T. 10786. LNCS. Fort Lauderdale, FL, USA : Springer, avr. 2018, p. 25–46.

[BMS11] Paulo S.L.M BARRETO, Rafael MISOCZKI et Marcos A. Jr. SIMPLICIO. “One-time signature scheme from syndrome decoding over generic error-correcting codes”. In : *Journal of Systems and Software* 84.2 (2011), p. 198–204.

[BMT78] Elwyn BERLEKAMP, Robert MCELIECE et Henk van TILBORG. “On the inherent intractability of certain coding problems”. In : *IEEE Trans. Inform. Theory* 24.3 (mai 1978), p. 384–386.

[BR93] Mihir BELLARE et Phillip ROGAWAY. “Random Oracles are Practical : A Paradigm for Designing Efficient Protocols”. In : *CCS ’93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*. 1993, p. 62–73.

[BR96] Mihir BELLARE et Phillip ROGAWAY. “The Exact Security of Digital Signatures-How to Sign with RSA and Rabin”. In : *Advances in Cryptology - EUROCRYPT ’96*. T. 1070. LNCS. Springer, 1996, p. 399–416.

[Bri+19] Rémi BRICOUT, André CHAILLOUX, Thomas DEBRIS-ALAZARD et Matthieu LEQUESNE. *Ternary Syndrome Decoding with Large Weights*. preprint. arXiv :1903.07464, to appear in the proceedings of SAC 2019. Fév. 2019.

[CFG89] Mark CHAIMOVICH, Gregory FREIMAN et Zvi GALIL. “Solving dense subset-sum problems by using analytical number theory”. In : *J. Complexity* 5.3 (1989), p. 271–282.

[CFS01] Nicolas COURTOIS, Matthieu FINIASZ et Nicolas SENDRIER. “How to Achieve a McEliece-based Digital Signature Scheme”. In : *Advances in Cryptology - ASIACRYPT 2001*. T. 2248. LNCS. Gold Coast, Australia : Springer, 2001, p. 157–174.

[CG90] John T COFFEY et Rodney M GOODMAN. “The complexity of information set decoding”. In : *IEEE Transactions on Information Theory* 36.5 (1990), p. 1031–1037.

[Cha17] Julia CHAULET. “Étude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques”. Thèse de doct. University Pierre et Marie Curie, mar. 2017.

[CJ04] Jean-Sebastien CORON et Antoine JOUX. *Cryptanalysis of a provably secure cryptographic hash function*. IACR Cryptology ePrint Archive, Report 2004/013. http://eprint.iacr.org/. 2004.

[Coc01] Clifford COCKS. “An identity based encryption scheme based on quadratic residues”. In : *8th IMA International Conference on Cryptography and Coding*. LNCS. Springer, déc. 2001, p. 360–363.

[Cor02] Jean-Sébastien CORON. “Optimal Security Proofs for PSS and Other Signature Schemes”. In : *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*. 2002, p. 272–287.

[Cou+14] Alain COUVREUR, Philippe GABORIT, Valérie GAUTHIER-UMAÑA, Ayoub OTMANI et Jean-Pierre TILLICH. “Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes”. In : *Des. Codes Cryptogr.* 73.2 (2014), p. 641–666.

[Cou01] Nicolas COURTOIS. “Efficient zero-knowledge authentication based on a linear algebra problem MinRank”. In : *Advances in Cryptology - ASIACRYPT 2001*. T. 2248. LNCS. Gold Coast, Australia : Springer, 2001, p. 402–421.

[Cou19] Alain COUVREUR. *Introduction to Coding Theory*. http://www.lix.polytechnique.fr/~alain.couvreur/doc_ens/lecture_notes.pdf. Lecture note at MPRI. 2019.

[COV07] Pierre-Louis CAYREL, Ayoub OTMANI et Damien VERGNAUD. “On Kabatianskii-Krouk-Smeets Signatures”. In : *Arithmetic of Finite Fields - WAIFI 2007*. T. 4547. LNCS. Madrid, Spain, 2007, p. 237–251.

[CS16a] Rodolfo CANTO-TORRES et Nicolas SENDRIER. “Analysis of Information Set Decoding for a Sub-linear Error Weight”. In : *Post-Quantum Cryptography 2016*. LNCS. Fukuoka, Japan, fév. 2016, p. 144–161.

[CS16b] Julia CHAULET et Nicolas SENDRIER. “Worst case QC-MDPC decoder for McEliece cryptosystem”. In : *IEEE Conference, ISIT 2016*. IEEE Press, 2016, p. 1366–1370.

[CT19] Rodolfo CANTO-TORRES et Jean-Pierre TILLICH. “Speeding up decoding a code with a non-trivial automorphism group up to an exponential factor”. In : *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2019*. 2019, p. 1927–1931.

[CT91] Thomas M. COVER et Joy A. THOMAS. *Information Theory*. Wiley Series in Telecommunications. Wiley, 1991.

[Dal10] Léonard DALLOT. “Sécurité de protocoles cryptographiques fondés sur les codes correcteurs d’erreurs”. Thèse de doct. Université de Caen, 2010.

[DG17a] Nico DÖTTLING et Sanjam GARG. “From Selective IBE to Full IBE and Selective HIBE”. In : *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*. 2017, p. 372–408.

[DG17b] Nico DÖTTLING et Sanjam GARG. “Identity-Based Encryption from the Diffie-Hellman Assumption”. In : *Advances in Cryptology - CRYPTO 2017*. Sous la dir. de Jonathan KATZ et Hovav SHACHAM. T. 10401. LNCS. Santa Barbara, CA, USA : Springer, août 2017, p. 537–569.

[DH76] Whitfield DIFFIE et Martin HELLMAN. “New directions in cryptography”. In : *IEEE transactions on Information Theory* 22.6 (1976), p. 644–654.

[DST17a] T. DEBRIS-ALAZARD, N. SENDRIER et J.-P. TILLICH. *The problem with the SURF scheme*. preprint. arXiv :1706.08065. Nov. 2017.

[DST17b] Thomas DEBRIS-ALAZARD, Nicolas SENDRIER et Jean-Pierre TILLICH. *SURF : a new code-based signature scheme*. preprint. arXiv :1706.08065v3. Sept. 2017.

[DST17c] Thomas DEBRIS-ALAZARD, Nicolas SENDRIER et Jean-Pierre TILLICH. *The problem with the SURF scheme*. preprint. arXiv :1706.08065. Nov. 2017.

[DST19a] Thomas DEBRIS-ALAZARD, Nicolas SENDRIER et Jean-Pierre TILLICH. "Wave : A New Family of Trapdoor One-Way Preimage Sampleable Functions Based on Codes". In : *Advances in Cryptology - ASIACRYPT 2019*. LNCS. Kobe, Japan, déc. 2019.

[DST19b] Thomas DEBRIS-ALAZARD, Nicolas SENDRIER et Jean-Pierre TILLICH. *Wave : A New Family of Trapdoor One-Way Preimage Sampleable Functions Based on Codes*. Cryptology ePrint Archive, Report 2018/996. https://eprint.iacr.org/2018/996. Mar. 2019.

[DT17] Thomas DEBRIS-ALAZARD et Jean-Pierre TILLICH. "Statistical decoding". In : *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2017*. Aachen, Germany, juin 2017, p. 1798–1802.

[DT18a] T. DEBRIS-ALAZARD et J.-P. TILLICH. "Two attacks on rank metric code-based schemes : RankSign and an Identity-Based-Encryption scheme". In : *ASIACRYPT*. 2018.

[DT18b] Thomas DEBRIS-ALAZARD et Jean-Pierre TILLICH. *Two attacks on rank metric code-based schemes : RankSign and an Identity-Based-Encryption scheme*. preprint. IACR Cryptology ePrint Archive. Juin 2018.

[DT18c] Thomas DEBRIS-ALAZARD et Jean-Pierre TILLICH. "Two attacks on rank metric code-based schemes : RankSign and an Identity-Based-Encryption scheme". In : *Advances in Cryptology - ASIACRYPT 2018*. T. 11272. LNCS. Brisbane, Australia : Springer, déc. 2018, p. 62–92.

[Dum89] Il'ya DUMER. "Two decoding algorithms for linear codes". In : *Probl. Inf. Transm.* 25.1 (1989), p. 17–23.

[Dum91] Ilya DUMER. "On minimum distance decoding of linear codes". In : *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*. Moscow, 1991, p. 50–52.

[Döt+18] Nico DÖTTLING, Sanjam GARG, Mohammad HAJIABADI et Daniel MASNY. "New Constructions of Identity-Based and Key-Dependent Message Secure Encryption Schemes". In : *Public-Key Cryptography - PKC 2018*. Sous la dir. de Michel ABDALLA et Ricardo DAHAB. T. 10769. LNCS. Rio de Janeiro, Brazil : Springer, mar. 2018, p. 3–31.

[Döt14] Nico M. DÖTTLING. "Cryptography based on the Hardness of Decoding". Thèse de doct. Karlsruher Instituts für Technologie (KIT), mai 2014.

[Eli55] Peter ELIAS. "oding for Noisy Channels". In : *IRE conv. Rec.* 3 (1955), p. 37.

[Fab+17] Tomás FABSIC, Viliam HROMADA, Paul STANKOVSKI, Pavol ZAJAC, Qian GUO et Thomas JOHANSSON. "A Reaction Attack on the QC-LDPC McEliece Cryptosystem". In : *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*. T. 10346. LNCS. Springer, 2017, p. 51–68.

[Fau+10a] Jean-Charles FAUGÈRE, Valérie GAUTHIER, Ayoub OTMANI, Ludovic PERRET et Jean-Pierre TILLICH. *A Distinguisher for High Rate McEliece Cryptosystems*. IACR Cryptology ePrint Archive, Report2010/331. http ://eprint.iacr.org/. 2010.

[Fau+10b] Jean-Charles FAUGÈRE, Ayoub OTMANI, Ludovic PERRET et Jean-Pierre TILLICH. "Algebraic Cryptanalysis of McEliece Variants with Compact Keys". In : *Advances in Cryptology - EUROCRYPT 2010*. T. 6110. LNCS. 2010, p. 279–298.

[Fau+13] Je.-C. FAUGÈRE, V. GAUTHIER, A. OTMANI, L. PERRET et J.-P. TILLICH. “A Distinguisher for High Rate McEliece Cryptosystems”. In : *IEEE IT* (oct. 2013).

[Fau15] Jean-Charles FAUGÈRE. *Résolution des systèmes polynômiaux en utilisant les bases de Grobner*. http://www.lifl.fr/jncf2015/files/lecture-notes/faugere.pdf. Lecture note. 2015.

[Fin10] Matthieu FINIASZ. “Parallel-CFS - Strengthening the CFS McEliece-Based Signature Scheme”. In : *Selected Areas in Cryptography 17th International Workshop, 2010, Waterloo, Ontario, Canada, August 12-13, 2010, revised selected papers*. T. 6544. LNCS. Springer, 2010, p. 159–170.

[FKI07] Marc P. C. FOSSORIER, Kazukuni KOBARA et Hideki IMAI. “Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of McEliece cryptosystem”. In : *IEEE Trans. Inform. Theory* 53.1 (2007), p. 402–411.

[FLP08] Jean-Charles FAUGÈRE, Françoise LEVY-DIT-VEHEL et Ludovic PERRET. “Cryptanalysis of Minrank”. In : *Advances in Cryptology - CRYPTO 2008*. Sous la dir. de David WAGNER. T. 5157. LNCS. 2008, p. 280–296.

[Fou+17] Pierre-Alain FOUQUE, Jeffrey HOFFSTEIN, Paul KIRCHNER, Vadim LYUBASHEVSKY, Thomas PORNIN, Thomas PREST, Thomas RICOSSET, Gregor SEILER, William WHYTE et Zhenfei ZHANG. *Falcon : Fast-Fourier Lattice-based Compact Signatures over NTRU*. First round submission to the NIST post-quantum cryptography call. NIST Round 1 submission for Post-Quantum Cryptography. Nov. 2017.

[FP05] Abraham FLAXMAN et Bartosz PRZYDATEK. “Solving Medium-Density Subset Sum Problems in Expected Polynomial Time”. In : *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*. 2005, p. 305–314.

[FS09] Matthieu FINIASZ et Nicolas SENDRIER. “Security Bounds for the Design of Code-based Cryptosystems”. In : *Advances in Cryptology - ASIACRYPT 2009*. Sous la dir. de M. MATSUI. T. 5912. LNCS. Springer, 2009, p. 88–105.

[FS87] Amos FIAT et Adi SHAMIR. “How to Prove Yourself : Practical Solutions to Identification and Signature Problems”. In : *Advances in Cryptology - CRYPTO ’86*. Sous la dir. d’A.M. ODLYZKO. T. 263. LNCS. Springer, 1987, p. 186–194.

[FS96] Jean-Bernard FISCHER et Jacques STERN. “An efficient pseudo-random generator provably as secure as syndrome decoding”. In : *Advances in Cryptology - EUROCRYPT’96*. Sous la dir. d’Ueli MAURER. T. 1070. LNCS. Springer, 1996, p. 245–255. ISBN : ISBN 978-3-540-61186-8.

[FSEDS10] Jean-Charles FAUGÈRE, Mohab SAFEY EL DIN et Pierre-Jean SPAENLEHAUER. “Computing loci of rank defects of linear matrices using Gröbner bases and applications to cryptology”. In : *International Symposium on Symbolic and Algebraic Computation, ISSAC 2010, Munich, Germany, July 25-28, 2010*. 2010, p. 257–264.

[Fuk+17] Kazuhide FUKUSHIMA, Partha Sarathi ROY, Rui XU, Shinsaku KIYOMOTO, Kirill MOROZOV et Tsuyoshi TAKAGI. *RaCoSS (Random Code-based Signature Scheme)*. first round submission to the NIST post-quantum cryptography call. NIST Round 1 submission for Post-Quantum Cryptography. Nov. 2017.

[Gab+13] Philippe GABORIT, Gaétan MURAT, Olivier RUATTA et Gilles ZÉMOR. "Low Rank Parity Check codes and their application to cryptography". In : *Proceedings of the Workshop on Coding and Cryptography WCC'2013*. Bergen, Norway, 2013.

[Gab+14a] P. GABORIT, O. RUATTA, J. SCHREK et G. ZÉMOR. "RankSign : An Efficient Signature Algorithm Based on the Rank Metric". In : *Post-Quantum Cryptography*. 2014.

[Gab+14b] Philippe GABORIT, Olivier RUATTA, Julien SCHREK et Gilles ZÉMOR. "RankSign : An Efficient Signature Algorithm Based on the Rank Metric (extended version on arXiv)". In : *Post-Quantum Cryptography 2014*. T. 8772. LNCS. Springer, 2014, p. 88–107.

[Gab+16] Philippe GABORIT, Adrien HAUTEVILLE, Duong Hieu PHAN et Jean-Pierre TILLICH. *Identity-based Encryption from Rank Metric*. IACR Cryptology ePrint Archive, Report2017/623. http://eprint.iacr.org/. Mai 2016.

[Gab+17a] P. GABORIT, A. HAUTEVILLE, D. H. PHAN et J.-P. TILLICH. "Identity-based Encryption from Rank Metric". In : *Advances in Cryptology - CRYPTO*. 2017.

[Gab+17b] Philippe GABORIT, Adrien HAUTEVILLE, Duong Hieu PHAN et Jean-Pierre TILLICH. "Identity-based Encryption from Rank Metric". In : *Advances in Cryptology - CRYPTO2017*. T. 10403. LNCS. Santa Barbara, CA, USA : Springer, août 2017, p. 194–226.

[Gab05] Philippe GABORIT. "Shorter keys for code based cryptography". In : *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*. Bergen, Norway, mar. 2005, p. 81–91.

[Gab85] Ernest Mukhamedovich GABIDULIN. "Theory of codes with maximum rank distance". In : *Problemy Peredachi Informatsii* 21.1 (1985), p. 3–16.

[Gal63] Robert G. GALLAGER. *Low Density Parity Check Codes*. Cambridge, Massachusetts : M.I.T. Press, 1963.

[GHT16] Philippe GABORIT, Adrien HAUTEVILLE et Jean-Pierre TILLICH. "RankSynd a PRNG Based on Rank Metric". In : *Post-Quantum Cryptography 2016*. Fukuoka, Japan, fév. 2016, p. 18–28.

[GKH17] Cheikh Thiécoumba GUEYE, Jean Belo KLAMTI et Shoichi HIROSE. "Generalization of BJMM-ISD Using May-Ozerov Nearest Neighbor Algorithm over an Arbitrary Finite Field \mathbb F_q". In : *Codes, Cryptology and Information Security - Second International Conference, C2SI 2017, Rabat, Morocco, April 10-12, 2017, Proceedings - In Honor of Claude Carlet*. 2017, p. 96–109.

[Gli+14] Danilo GLIGOROSKI, Simona SAMARDJISKA, Håkon JACOBSEN et Sergey BEZZATEEV. *McEliece in the world of Escher*. IACR Cryptology ePrint Archive, Report2014/360. http://eprint.iacr.org/. 2014.

[GM02] Shafi GOLDWASSER et Daniele MICCIANCIO. *Complexity of Lattice Problems : A Cryptographic Perspective*. T. 671. Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, mar. 2002.

[GM91] Zvi GALIL et Oded MARGALIT. "An Almost Linear-Time Algorithm for the Dense Subset-Sum Problem". In : *SIAM J. Comput.* 20.6 (1991), p. 1157–1189.

[Gol01] Oded GOLDREICH. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001. ISBN : 0-521-79172-3.

[GPT91] Ernst M. GABIDULIN, A. V. PARAMONOV et O. V. TRETJAKOV. "Ideals over a non-commutative ring and their applications to cryptography". In : *Advances in Cryptology - EUROCRYPT'91*. LNCS 547. Brighton, avr. 1991, p. 482–489.

[GPV08a] C. GENTRY, C. PEIKERT et V. VAIKUNTANATHAN. “Trapdoors for hard lattices and new cryptographic constructions”. In : *STOC*. 2008.

[GPV08b] Craig GENTRY, Chris PEIKERT et Vinod VAIKUNTANATHAN. “Trapdoors for hard lattices and new cryptographic constructions”. In : *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM. 2008, p. 197–206.

[GRS00] Oded GOLDREICH, Ronitt RUBINFELD et Madhu SUDAN. “Learning Polynomials with Queries : The Highly Noisy Case”. In : *SIAM J. Discrete Math.* 13.4 (2000), p. 535–570.

[GRS13] Philippe GABORIT, Olivier RUATTA et Julien SCHREK. “On the complexity of the Rank Syndrome Decoding problem”. In : *CoRR* abs/1301.1026 (2013).

[GRS16] Philippe GABORIT, Olivier RUATTA et Julien SCHREK. “On the Complexity of the Rank Syndrome Decoding Problem”. In : *IEEE Trans. Information Theory* 62.2 (2016), p. 1006–1019.

[GS12] Philippe GABORIT et Julien SCHREK. “Efficient code-based one-time signature from automorphism groups with syndrome compatibility”. In : *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2012*. Cambridge, MA, USA, juil. 2012, p. 1982–1986.

[GZ16] Philippe GABORIT et Gilles ZÉMOR. “On the hardness of the decoding and the minimum distance problems for rank codes”. In : *IEEE Trans. Information Theory* 62(12) (2016), p. 7245–7252.

[Hir16] Shoichi HIROSE. “May-Ozerov Algorithm for Nearest-Neighbor Problem over $\mathbb{F}_q$ and Its Application to Information Set Decoding”. In : *Innovative Security Solutions for Information Technology and Communications - 9th International Conference, SECITC 2016, Bucharest, Romania, June 9-10, 2016, Revised Selected Papers*. 2016, p. 115–126.

[HJ10] Nicholas HOWGRAVE-GRAHAM et Antoine JOUX. “New generic algorithms for hard knapsacks”. In : *Advances in Cryptology - EUROCRYPT 2010*. Sous la dir. d’Henri GILBERT. T. 6110. LNCS. Sringer, 2010.

[HPS98] Jeffrey HOFFSTEIN, Jill PIPHER et Joseph H. SILVERMAN. “NTRU : A Ring-Based Public Key Cryptosystem”. In : *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*. Sous la dir. de Joe BUHLER. T. 1423. LNCS. Springer, 1998, p. 267–288.

[Hue+18] Andreas HUELSING, Daniel J. BERNSTEIN, Lorenz PANNY et Tanja LANGE. Official NIST comments made for RaCoSS. Official NIST comments made for RaCoSS. 2018.

[Int+18] Carmelo INTERLANDO, Karan KHATHURIA, Nicole ROHRER, Joachim ROSENTHAL et Violetta WEGER. “Generalization of the Ball-Collision Algorithm”. In : *arXiv preprint arXiv :1812.10955* (2018).

[IS98] Mourad E.H. ISMAIL et Plamen SIMEONOV. “Strong asymptotics for Krawtchouk polynomials”. In : *Journal of Computational and Applied Mathematics* (1998), p. 121–144.

[Jab01] Abdulrahman Al JABRI. “A statistical decoding algorithm for general linear block codes”. In : *Cryptography and coding. Proceedings of the 8th IMA International Conference*. Sous la dir. de Bahram HONARY. T. 2260. LNCS. Cirencester, UK : Springer, déc. 2001, p. 1–8.

[Jou00] Antoine JOUX. “A One Round Protocol for Tripartite Diffie-Hellman”. In : *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*. 2000, p. 385–394.

[Joz01] Richard JOZSA. “Quantum factoring, discrete logarithms, and the hidden subgroup problem”. In : *Computing in Science and Engineering* 3.2 (2001), p. 34–43.

[KKS05] Gregory KABATIANSKII, Evgenii KROUK et Sergei SEMENOV. *Error Correcting Coding and Security for Data Networks : Analysis of the Superchannel Concept*. John Wiley & Sons, 2005.

[KKS97] Gregory KABATIANSKII, Evgenii KROUK et Ben. J. M. SMEETS. “A Digital Signature Scheme Based on Random Error-Correcting Codes”. In : *IMA Int. Conf.* T. 1355. LNCS. Springer, 1997, p. 161–167.

[Kle00] Philip N. KLEIN. “Finding the closest lattice vector when it’s unusually close”. In : *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA*. 2000, p. 937–941.

[KS99] Aviad KIPNIS et Adi SHAMIR. “Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization”. In : *Advances in Cryptology - CRYPTO’99*. T. 1666. LNCS. Santa Barbara, California, USA : Springer, août 1999, p. 19–30.

[LB88] Pil J. LEE et Ernest F. BRICKELL. “An Observation on the Security of McEliece’s Public-Key Cryptosystem”. In : *Advances in Cryptology - EUROCRYPT’88*. T. 330. LNCS. Springer, 1988, p. 275–280.

[Lee+17] Wijik LEE, Young-Sik KIM, Yong-Woo LEE et Jong-Seon NO. *Post quantum signature scheme based on modified Reed-Muller code pqsigRM*. first round submission to the NIST post-quantum cryptography call. NIST Round 1 submission for Post-Quantum Cryptography. Nov. 2017.

[Leg11] Matthieu LEGEAY. “Permutation decoding : Towards an approach using algebraic properties of the σ-subcode”. In : *WCC 2011*. Sous la dir. de Daniel AUGOT et Anne CANTEAUT. 2011, p. 193–202.

[Leo88] Jeffrey LEON. “A probabilistic algorithm for computing minimum weights of large error-correcting codes”. In : *IEEE Trans. Inform. Theory* 34.5 (1988), p. 1354–1359.

[Loi06] Pierre LOIDREAU. *Properties of codes in rank metric*. 2006.

[Lyu05] Vadim LYUBASHEVSKY. “On Random High Density Subset Sums”. In : *Electronic Colloquium on Computational Complexity (ECCC)* 1.007 (2005).

[Lyu09a] V. LYUBASHEVSKY. “Fiat-Shamir with aborts : Applications to lattice and factoring-based signatures”. In : *ASIACRYPT*. 2009.

[Lyu09b] Vadim LYUBASHEVSKY. “Fiat-Shamir with aborts : Applications to lattice and factoring-based signatures”. In : *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2009, p. 598–616.

[Lyu12] Vadim LYUBASHEVSKY. “Lattice signatures without trapdoors”. In : *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2012, p. 738–755.

[MB09] Rafael MISOCZKI et Paulo BARRETO. “Compact McEliece Keys from Goppa Codes”. In : *Selected Areas in Cryptography*. Calgary, Canada, 2009.

[McE78] Robert J. MCELIECE. “A Public-Key System Based on Algebraic Coding Theory”. In : DSN Progress Report 44. Jet Propulsion Lab, 1978, p. 114–116.

[Meu17] Alexander MEURER. “A Coding-Theoretic Approach to Cryptanalysis”. Thèse de doct. Ruhr University Bochum, nov. 2017.

[Mis+12] Rafael MISOCZKI, Jean-Pierre TILLICH, Nicolas SENDRIER et Paulo S. L. M. BARRETO. *MDPC-McEliece : New McEliece Variants from Moderate Density Parity-Check Codes*. 2012.

[Mis+13] Rafael MISOCZKI, Jean-Pierre TILLICH, Nicolas SENDRIER et Paulo S. L. M. BARRETO. "MDPC-McEliece : New McEliece variants from Moderate Density Parity-Check codes". In : *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. 2013, p. 2069–2073.

[MMT11] Alexander MAY, Alexander MEURER et Enrico THOMAE. "Decoding random linear codes in $O(2^{0.054n})$". In : *Advances in Cryptology - ASIACRYPT 2011*. Sous la dir. de Dong Hoon LEE et Xiaoyun WANG. T. 7073. LNCS. Springer, 2011, p. 107–124.

[MO15] Alexander MAY et Ilya OZEROV. "On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes". In : *Advances in Cryptology - EUROCRYPT 2015*. Sous la dir. d'E. OSWALD et M. FISCHLIN. T. 9056. LNCS. Springer, 2015, p. 203–228.

[MP16] Dustin MOODY et Ray A. PERLNER. "Vulnerabilities of "McEliece in the World of Escher"". In : *Post-Quantum Cryptography 2016*. LNCS. Springer, 2016.

[MR07] Daniele MICCIANCIO et Oded REGEV. "Worst-Case to Average-Case Reductions Based on Gaussian Measures". In : *SIAM J. Comput.* 37.1 (2007), p. 267–302.

[MR09] Daniele MICCIANCIO et Oded REGEV. "Lattice-based cryptography". In : *Post-quantum cryptography*. Springer, 2009, p. 147–191.

[MS09] L. MINDER et A. SINCLAIR. "The extended *k*-tree algorithm". In : *Proceedings of SODA 2009*. Sous la dir. de C. MATHIEU. SIAM, 2009, p. 586–595.

[MS86] Florence J. MACWILLIAMS et Neil J. A. SLOANE. *The Theory of Error-Correcting Codes*. Fifth. Amsterdam : North–Holland, 1986.

[Nie85] Harald NIEDERREITER. "A Public-Key Cryptosystem based on Shift Register Sequences". In : *Advances in Cryptology - EUROCRYPT 1985*. T. 219. LNCS. 1985, p. 35–39.

[Nie86] Harald NIEDERREITER. "Knapsack-type cryptosystems and algebraic coding theory". In : *Problems of Control and Information Theory* 15.2 (1986), p. 159–166.

[OJ02] Alexei V. OURIVSKI et Thomas JOHANSSON. "New Technique for Decoding Codes in the Rank Metric and Its Cryptography Applications". English. In : *Problems of Information Transmission* 38.3 (2002), p. 237–246. ISSN : 0032-9460.

[OS09] Raphael OVERBECK et Nicolas SENDRIER. "Code-based cryptography". In : *Post-quantum cryptography*. Sous la dir. de Daniel J. BERNSTEIN, Johannes BUCHMANN et Erik DAHMEN. Springer, 2009, p. 95–145. ISBN : 978-3-540-88701-0.

[OT11] Ayoub OTMANI et Jean-Pierre TILLICH. "An Efficient Attack on All Concrete KKS Proposals". In : *Post-Quantum Cryptography 2011*. T. 7071. LNCS. 2011, p. 98–116.

[OT12] Ayoub OTMANI et Jean-Pierre TILLICH. "On the Design of Code-Based Signatures". In : *Code-based Cryptography Workshop (CBC 2012)*. Lyngby, Denmark, mai 2012.

[OTD10] Ayoub OTMANI, Jean-Pierre TILLICH et Léonard DALLOT. "Cryptanalysis of Two McEliece Cryptosystems Based on Quasi-Cyclic Codes". In : *Special Issues of Mathematics in Computer Science* 3.2 (jan. 2010), p. 129–140.

[Ove05] Raphael OVERBECK. “A New Structural Attack for GPT and Variants”. In : *Mycrypt*. T. 3715. LNCS. 2005, p. 50–63.

[Ove06] Raphael OVERBECK. “Statistical decoding revisited”. In : *Information security and privacy : 11th Australasian conference, ACISP 2006*. Sous la dir. de Reihaneh Safavi-Naini LYNN BATTEN. T. 4058. LNCS. Springer, 2006, p. 283–294.

[Pet10] Christiane PETERS. “Information-Set Decoding for Linear Codes over $\mathbf{F}_q$”. In : *Post-Quantum Cryptography 2010*. T. 6061. LNCS. Springer, 2010, p. 81–94.

[Poi19] David POINTCHEVAL. *Provable Security in the Computational Model*. https://www.di.ens.fr/david.pointcheval/enseignement/mpri2/cm2.pdf. Lecture note. 2019.

[Pra62] Eugene PRANGE. “The use of information sets in decoding cyclic codes”. In : *IRE Transactions on Information Theory* 8.5 (1962), p. 5–9.

[PRS17] Chris PEIKERT, Oded REGEV et Noah STEPHENS-DAVIDOWITZ. “Pseudorandomness of ring-LWE for any ring and modulus”. In : *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. 2017, p. 461–473.

[PT16] Aurélie PHESSO et Jean-Pierre TILLICH. “An Efficient Attack on a Code-Based Signature Scheme”. In : *Post-Quantum Cryptography 2016*. T. 9606. LNCS. Fukuoka, Japan : Springer, fév. 2016, p. 86–103.

[Reg05] Oded REGEV. “On lattices, learning with errors, random linear codes, and cryptography”. In : *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. 2005, p. 84–93.

[RSA78] Ronald L. RIVEST, Adi SHAMIR et Leonard M. ADLEMAN. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In : *Commun. ACM* 21.2 (1978), p. 120–126.

[S19] Communication personnelle avec Damien Stehlé.

[Sch91] Claus-Peter SCHNORR. “Efficient signature generation by smart cards”. In : *Journal of cryptology* 4.3 (1991), p. 161–174.

[Sen02] Nicolas SENDRIER. “Cryptosystèmes à clé publique basés sur les codes correcteurs d’erreurs”. In : *Mémoire d’habilitation à diriger des recherches, Université Paris 6*. 2002.

[Sen11a] Nicolas SENDRIER. “Decoding One Out of Many”. In : *Post-Quantum Cryptography 2011*. T. 7071. LNCS. 2011, p. 51–67.

[Sen11b] Nicolas SENDRIER. “The tightness of security reductions in code-based cryptography”. In : *Proc. IEEE Inf. Theory Workshop- ITW 2011*. IEEE, 2011, p. 415–419.

[Sen97] Nicolas SENDRIER. “On the dimension of the hull”. In : *SIAM J. Discrete Math.* T. 10. 2. 1997, p. 282–293.

[Sha84] Adi SHAMIR. “Identity-based cryptosystems and signature schemes”. In : *Advances in Cryptology - CRYPTO 84*. Sous la dir. de G. R. BLAKLEY et David CHAUM. T. 196. LNCS. Springer, 1984, p. 47–53.

[Sho04] Victor SHOUP. “Sequences of games : a tool for taming complexity in security proofs”. In : *IACR Cryptology ePrint Archive* 2004 (2004), p. 332.

[Sho94] Peter W. SHOR. “Algorithms for quantum computation : Discrete logarithms and factoring”. In : *FOCS*. Sous la dir. de S. GOLDWASSER. 1994, p. 124–134.

[SK14] Sujan Raj SHRESTHA et 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)*. IEEE. 2014, p. 368–372.

[SKK10] Danilo SILVA, Frank R. KSCHISCHANG et Ralf KÖTTER. “Communication over finite-field matrix channels”. In : *IEEE Trans. Information Theory* 56.3 (2010), p. 1296–1305.

[SOK00] Ryuichi SAKAI, Kiyoshi OHGISHI et Masao KASAHARA. “Cryptosystems based on pairing”. In : *SCIS 2000*. Okinawa, Japan, jan. 2000.

[Spa12] Pierre-Jean SPAENLENHAUER. “Résolution de systèmes multi-homogènes et determinantiels”. Thèse de doct. Univ. Pierre et Marie Curie- Paris 6, oct. 2012.

[SS92] Vladimir Michilovich SIDELNIKOV et S.O. SHESTAKOV. “On the insecurity of cryptosystems based on generalized Reed-Solomon codes”. In : *Discrete Math. Appl.* 1.4 (1992), p. 439–444.

[Ste88] Jacques STERN. “A method for finding codewords of small weight”. In : *Coding Theory and Applications*. Sous la dir. de G. D. COHEN et J. WOLFMANN. T. 388. LNCS. Springer, 1988, p. 106–113.

[Ste93a] J. STERN. “A New Identification Scheme Based on Syndrome Decoding”. In : *CRYPTO*. 1993.

[Ste93b] Jacques STERN. “A New Identification Scheme Based on Syndrome Decoding”. In : *Advances in Cryptology - CRYPTO’93*. Sous la dir. de D.R. STINSON. T. 773. LNCS. Springer, 1993, p. 13–21.

[SV18] Nicolas SENDRIER et Valentin VASSEUR. “On the Decoding Failure Rate of QC-MDPC Bit-Flipping Decoders”. In : *IACR Cryptology ePrint Archive* 2018 (2018). to appear in PQCrypto 2019, p. 1207.

[SV19] Nicolas SENDRIER et Valentin VASSEUR. “On the Decoding Failure Rate of QC-MDPC Bit-Flipping Decoders”. In : *Post-Quantum Cryptography 2019*. Sous la dir. de Jintai DING et Rainer STEINWANDT. T. 11505. LNCS. Chongquing, China : Springer, mai 2019, p. 404–416.

[Til18a] Jean-Pierre TILLICH. “The Decoding Failure Probability of MDPC Codes”. In : *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*. 2018, p. 941–945.

[Til18b] Jean-Pierre TILLICH. *The decoding failure probability of MDPC codes*. preprint. arXiv :1801.04668. Sept. 2018.

[Vér96] Pascal VÉRON. “Improved identification schemes based on error-correcting codes”. In : *Appl. Algebra Eng. Commun. Comput.* 8.1 (1996), p. 57–69.

[Wag02] David WAGNER. “A generalized birthday problem”. In : *Advances in Cryptology - CRYPTO 2002*. Sous la dir. de Moti YUNG. T. 2442. LNCS. Springer, 2002, p. 288–303. ISBN : 978-3-540-44050-5.

[Wie06] Christian WIESCHEBRINK. “Two NP-complete Problems in Coding Theory with an Application in Code Based Cryptography”. In : *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. 2006, p. 1733–1737.

[WMM10] Martin J. WAINWRIGHT, Elitza N. MANEVA et Emin MARTINIAN. “Lossy source compression using low-density generator matrix codes : analysis and algorithms”. In : *IEEE Trans. Information Theory* 56.3 (2010), p. 1351–1368.

[Agu+19] Carlos AGUILAR MELCHOR, Nicolas ARAGON, Slim BETTAIEB, Loïc BIDOUX, Olivier BLAZY, Jean-Christophe DENEUVILLE, Philippe GABORIT, Gilles ZÉMOR, Alain COUVREUR et Adrien HAUTEVILLE. *Rank Quasi Cyclic (RQC)*. Second round submission to the NIST post-quantum cryptography call. Avr. 2019.

[Can17] Rodolfo CANTO TORRES. "Asymptotic Analysis of ISD algorithms for the $q-$ary case". In : *Proceedings of the Tenth International Workshop on Coding and Cryptography WCC 2017*. Sept. 2017.

[ElG84] Taher ELGAMAL. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In : 1984.