THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Design and analysis of dual attacks in code- and lattice-based cryptography

Présentée par

Charles Meyer-Hilfiger

et dirigée par Nicolas Sendrier et Jean-Pierre Tillich

Préparée au sein de l'équipe-projet COSMIQ, Inria Paris

Soutenue publiquement le 30 septembre 2025 devant le jury composé de :

Nicolas Sendrier	Inria Paris, France	Directeur
Jean-Pierre Tillich	Inria Paris, France	Directeur
Thomas Johansson	Lund University, Suède	Rapporteur
Gilles Zémor	Université de Bordeaux, France	Rapporteur
Pierre-Alain Fouque	Université de Rennes, France	Examinateur
Elena Kirshanova	Technology Innovation Institute, Émirats	Examinatrice
Pierre Loidreau	Direction générale de l'Armement, France	Examinateur
Damien Stehlé	CryptoLab, France	Examinateur
Kévin Carrier	Institut Polytechnique de Paris, France	Invité
Thomas Debris-Alazard	Inria Paris-Saclay, France	Invité

Remerciements

Quel bonheur d'avoir pu passer ces quatre dernières années à COSMIQ, si je pouvais dire que c'est parce que je ne voulais pas vous quitter que j'ai mis du temps à rédiger !

Jean-Pierre, je ne pense pas avoir été ton étudiant le plus facile... malgré ça tu ne m'as jamais lâché. Merci pour tout. Pour tout ce que tu m'as appris, pour ces centaines d'heures que j'ai passées dans ton bureau, pour ton exigence, pour t'être tapé la relecture de cette thèse (et dire que tu aurais "voulu" la relire une deuxième fois). Merci de donner autant de toi, sans jamais compter. Quelle chance incroyable j'ai eue d'être ton étudiant !

Nicolas, merci de m'avoir donné ma chance il y a 4 ans, alors que je n'avais jamais fait de crypto, avec comme unique pédigrée de la motivation et beaucoup de mails. En fait, merci d'avoir été là pour moi dans ces moments cruciaux qui ont fait la différence. Merci aussi pour nos discussions scientifiques, d'avoir relu une partie de cette thèse, et de t'être occupé de l'administration. Merci pour ta bienveillance continuelle.

Kévin et Thomas, vous avez porté plusieurs casquettes, une de co-encadrement de cette thèse avec Jean-Pierre et Nicolas, puis une autre, celle de collègues que j'aime, avec qui on va faire la fête jusqu'au bout de la nuit. Je sais qu'il peut être difficile de mélanger les deux mais vous l'avez fait avec brio !

Thomas, au fond depuis ces premières journées C2 jusqu'à la soutenance de cette thèse, tu ne m'as jamais vraiment lâché. Tu donnes beaucoup. Merci pour tout ce que tu m'as appris scientifiquement. Merci pour ces soirées que tu as passées au bar à expliquer avec beaucoup de patience à "un étudiant rebelle" comment rédiger ou faire une présentation. Merci pour nos fous rires et les super moments passés, $1, 2, 3$, let's go !

Kévin, c'est vrai que tu forces l'admiration, je ne sais pas comment tu fais pour faire tout ce que tu fais... Merci pour ce que tu m'as appris, de tes techniques d'optimisation numérique de mage noir jusqu'à ton organisation militaire (ok ça j'ai pas encore réussi à prendre). Mais surtout, j'ai vraiment beaucoup aimé travailler avec toi, tu te mets au niveau des gens comme personne d'autre. Tu m'as fait aimer ma première comédie musicale, je dois te dire bravo, c'était pas gagné.

Un grand merci également à Pierre-Alain, qui pensait recruter un étudiant en post-doctorat et qui m'a permis malgré tout de terminer dans de très bonnes conditions la rédaction de mon manuscrit au sein de l'équipe Capsule de l'Irisa.

I would like to thank Thomas and Gilles for reviewing this thesis and Pierre-Alain, Elena, Pierre, Damien for being part of this jury.

La vie est belle à COSMIQ et surtout bien vivante, surtout quand on fait partie de la team des animaux à sang chaud composée de Valérian, Jean-Pierre, Dounia, Virgile, Agathe, Aurélien, Axel et Merlin qui s'écharpent régulièrement au baby-foot dans un lot de trash-talk et de techniques de déstabilisation variées. Plus généralement, les doctorants sont étonnament fun, au point où, avec Clara on s'était posé la question s'il n'y avait pas une compétition secrète

de la part des permanents pour avoir les étudiants les plus cool : combien de fois j'ai entendu Maria, des étoiles dans les yeux, et surtout avec fierté, me dire à quel point Dounia et Bastien étaient trop incroyables.

Merci à Christelle de faire tourner cette équipe aussi efficacement, merci pour ta patience, même face aux cas les plus forcenés...

Un grand merci à la team Augustin, Aurélie, Clara, Clémence, Jules et Nicolas qui ont été des potes fidèles et une source inépuisable de commérages autour de bières. Augustin, je suis un peu jaloux de toi et ton pull X n'y est pour rien... Merci pour toutes nos soirées, de la fête de la musique jusqu'à cette soirée casino où tu as (failli) tout gagner. Clara, t'es une vraie fusée de vérité, toujours prête à dégainer, j'ai adoré t'aider à partager tes TD de crypto, je ne sais pas si tu gardes un souvenir aussi éclatant, moi j'en garde un aussi éclatant de notre initiation au tir à l'arc. Clémence, tu es une vraie guerrière. C'est vrai que nos débuts ont été bordéliques, mais je suis heureux que nos relations se soient normalisées, j'ai tout de même gardé quelques insupportables pistaches en souvenir du bon vieux temps. Jules, mon compagnon de bureau pendant 3 ans. Je pense que tu as le don de faire partie de ces potes régulateurs d'humeur : tantôt tu trouves les mots quand la situation est dure, tantôt tu gardes secrètement une pique au cas où elle soit trop facile ! Valérian, t'es un vrai cool kid ! On est frères de sujet et de thèse et surtout frères de positions de travail ésotériques, toi-même tu connais le boost de productivité. Le seul truc qui ne va pas me manquer c'est l'odeur du grec à 16h.

Agathe, je t'avoue que je ne pensais pas que quelqu'un puisse tenir ton nouveau programme d'entraînement, mon rire moqueur s'est vite éteint... je suis admiratif. Merci pour ces pauses goûter où tu m'as allègrement nourri.

Axel, mon seul regret c'est qu'on n'ait jamais réussi à te traîner à l'espace bien-être. J'admire le fait que tu as réussi là où moi j'ai échoué à environ mille reprises, je te laisse deviner ce que ça peut être.

Dounia, je pourrais t'écrire des éloges sur plusieurs pages si tout ceci n'était pas terni par tes pissettes à répétition. Non en vrai, tu rayonnes.

Aurélien, je suis admiratif de ta capacité à "rien faire" et pourtant, je pensais être un expert dans le domaine (que ce que je dise soit clair, je parle bien d'un acte volontaire, difficile, et dans le cadre d'un week-end et pas d'un banal fainéantisme latent au travail).

Valentin, je sais que tu vas détester que je dise ça mais je ne pense pas avoir rencontré personne de plus humble et drôle que toi. Sur une formule que tu connais bien : je ne te remercie pas d'avoir gardé pour toi les 40 bières qu'on avait gagnées au concours Mario Kart du Bars & Beers. Notre duo sobrement dénommé "Les Cendars" (sache Nicolas que c'était l'idée de Valentin, et que, mes supplications pour qu'il mette fin à sa folie portant atteinte aux mœurs les plus fondamentales ont toutes été vaines) a fait honneur à notre héritage commun en oblitérant la concurrence. Merci aussi d'avoir répondu à mes mitraillettes de questions quand je suis arrivé en stage dans ton bureau !

Virgile, plus connu sous le nom de Coupling Constant, il peut être reconnu facilement par le fait qu'il se balade toujours avec sa clé Allen pour boulonner des objets "simples" du quotidien. C'est un self-made man, mais c'est avant tout mon frère d'armes, aussi littéralement que moralement. Bien que Jean-Pierre n'ait pas percé complètement le mystère de nos messes basses, si tu veux mon avis, je crois qu'il s'est fait son idée. Anthony, d'ailleurs je ne te remercie pas pour la maigre (inexistante) réprimande que tu as infligée à Virgile après son infâme chantage, je ne suis pas certain qu'il ait compris la leçon...

Antoine et Magalie, merci d'avoir partagé mon nouveau bureau. Antoine, j'adore ton

militantisme, peut-être un jour on aura la chance de se faire gazer à deux !

Bastien, sache que tu as ma sympathie compréhensive : tu me surpasses de loin dans cet art subtil de pouvoir sortir des dingueries.

Guilhem, la prochaine fois qu'on se croisera ça sera autour d'une gaufre Nutella Chantilly à la paillotte, j'ai dit.

Enfin, merci à Samo pour sa bonne humeur contagieuse, juste respecte-moi, mets ma tasse à la bonne hauteur dans ton bureau ; à Thomas V., à Johanna, à Merlin pour ce bout de nougat que je t'ai volé quand Virgile avait fermé sa boîte de Pandore à clé. Merci à Simona et Mathieu pour nos moments passés à Amsterdam, à Rocco, et à Pierre !

Enfin merci aux permanents de cette équipe : Anne parce que tu nous fais toujours rayonner et surtout pour m'avoir nourri pendant ma période de croissance, Maria pour ta joie de vivre communicative, André pour toujours voir le positif, Anthony parce que tu as pris Virgile en thèse, Léo parce que tu as pris Jules et Clémence en thèse (mais là je le dis pour de vrai), et enfin Pascale, mon exemple de solidité et de combativité !

Rachelle, merci pour ton message de soutien hier, j'espère "tout défoncer" aussi, bien que je ne sois pas certain que le jury partage cet avis, je tâcherai de ne pas détruire le pupitre si les questions sont trop difficiles. Merci pour ces pétanques dans la team sudiste et surtout pour nos soirées, toujours gaies, surtout cette dernière 3 mètres sous terre, littéralement et figurativement.

Enfin mes nouveaux collègues de Rennes, en particulier à Aurore et André pour leur bienveillance, à Yixin pour nos discussions, à Patrick parce que c'est un vrai amoureux de la fête, à Daniel pour tes conseils, à Clémence, parce que j'aurai peut-être le droit d'être un PNJ dans ton JDR, à Baptiste et Gaël pour les manifs qu'on a déjà faites.

Enfin aux jeunes de la communauté crypto : Maxime pour toutes nos bières et ton aide, Victor j'espère qu'on trailera ensemble un jour, Wessel I am sorry but I think what Virgile told you about that stupid joke I made was shamefully true.

"Qu'importe l'éternité de la damnation pour qui a vécu en un instant l'infini de la jouissance !"
~~Baudelaire~~. Améliorée par Nicolas.

"Are you on campus ???? Get back to me !!!!!"
Jean-Pierre (enfin pas vraiment...) après que Virgile ait pris un jour de télétravail

"S'il te plaît, ne le répète pas à Anthony"
Kévin, un sandwich chips/chorizo dans une main, un coca dans l'autre

"Direction le bar à lait"
Thomas D., à la recherche d'une alternative

"Si ça continue comme ça, je change de bureau"
Jules, ouvrant la fenêtre

"Je peux pas, j'ai yoga"
Anne, Maria, Bastien, 3 individus durablement sérénisés par le yoga.

"Halte là ! J'ai retrouvé ma chaussette !"
Charles, dans la cage d'escalier

"Tu n'aurais pas vu un Bulbizarre dans le couloir ?"
Bastien, sur un de ses jeux électroniques

"Demain, il faut que je crée de la valeur pour les actionnaires"
Valentin, un employé engagé.

Abstract

English

In this thesis, we design and analyze dual attacks for solving the decoding problem. In code-based cryptography dual attacks, a.k.a statistical decoding, were introduced by Al-Jabri in 2001. However, it turned out that his algorithm was not competitive against Information Set Decoders (ISD). Starting with the Prange algorithm in 1962, the ISD's have been the dominant family of decoders for the last 60 years and are used to parameterize code-based schemes.

Our main contribution is to dramatically improve dual attacks and show that they can beat the best ISD's for some parameter regimes. In particular, our best attack asymptotically significantly beats all previously known decoders for codes of constant rates smaller than 0.42 at the Gilbert-Varshamov distance. This result was obtained by revisiting Al-Jabri's statistical decoding algorithm and generalizing it using a splitting strategy to reduce decoding to a problem that is essentially the Learning Parity with Noise (LPN) problem. We then solve it using standard solvers. Part of our work also lies in the development of tools to analyze these attacks: we do not use the traditional independence assumptions that were used to analyze statistical decoding and which must be used with great care. Our tools are based on the Poisson summation formula and a model for the distribution of the weight enumerator of random linear codes. We base the analysis of our attacks on this model and verify it experimentally. We also devise a variant of our most advanced attack that has, up to polynomial factors, the same performance but which we can fully prove without using this model.

The second part of this work is dedicated to devising and analyzing dual attacks in lattice-based cryptography. Here, the problem that we target is the Learning With Errors (LWE) problem. In this case, dual attacks have been recently vastly improved, partly using a similar splitting strategy. In particular, two recent attacks, first by Guo & Johansson in 2021 and then by Matzov in 2022 claimed to reduce the security of the NIST standard Kyber (ML-KEM). However, Ducas & Pulles showed in 2023 that the key independence assumptions used in the analysis of those recent lattice-based attacks were flawed. This left open the question of how to analyze these attacks and whether they could really work as expected. We devise a slight variant of these recent dual attacks and provide new tools for analyzing them without these assumptions. In particular, we show that our attack comes dangerously close to Kyber's security claims and that we slightly beat those recent attacks. This settles the controversy over whether a dual-sieve attack can really work as expected.

Français

Dans cette thèse, nous concevons et analysons des attaques duales pour résoudre le problème du décodage.

En cryptographie basée sur les codes, les attaques duales ont été introduites par Al-Jabri en 2001 avec l'algorithme de décodage statistique. Cependant, il s'est avéré que son algorithme n'était pas compétitif face aux décodeurs par ensembles d'informations (ISD). En commençant par l'algorithme de Prange en 1962, les ISDs ont été la famille dominante de décodeurs ces 60 dernières années et sont utilisés pour paramétrer les schémas basés sur les codes. Notre contribution principale est d'améliorer drastiquement les attaques duales et de montrer qu'elles peuvent battre les meilleurs ISDs pour certains régimes de paramètres. En particulier, notre meilleure attaque bat asymptotiquement de manière significative tous les décodeurs connus auparavant pour des codes de taux constant inférieur à 0,42 à la distance de Gilbert-Varshamov. Ce résultat a été obtenu en revisitant l'algorithme de décodage statistique d'Al-Jabri et en le généralisant à l'aide d'une stratégie de découpage pour réduire le problème du décodage à un problème qui est essentiellement un problème d'apprentissage de parité avec du bruit (LPN). Nous le résolvons ensuite en utilisant des solveurs standards. Une partie de notre travail réside aussi dans le développement d'outils pour analyser ces attaques: nous n'utilisons pas les hypothèses d'indépendance qui étaient traditionnellement utilisées pour analyser le décodage statistique et qui doivent être utilisées avec beaucoup de précautions. Nos outils sont basés sur la formule de sommation de Poisson et un modèle pour la distribution de l'énumérateur de poids des codes linéaires aléatoires. Nous basons l'analyse de nos attaques sur ce modèle et le vérifions expérimentalement. Nous concevons aussi une variante de notre attaque la plus avancée qui a, à des facteurs polynomiaux près, la même performance que celle-ci mais que nous pouvons prouver entièrement sans utiliser ce modèle.

La deuxième partie de ce travail est dédiée à la conception et à l'analyse d'attaques duales en cryptographie basée sur les réseaux. Ici, le problème que nous ciblons est le problème d'apprentissage avec du bruit (LWE). Dans ce contexte, les attaques duales ont été récemment grandement améliorées, en partie en utilisant une stratégie de découpage similaire. En particulier, deux attaques récentes, d'abord par Guo & Johansson en 2021 puis par Matzov en 2022, ont affirmé réduire la sécurité du standard NIST Kyber (ML-KEM). Cependant, Ducas & Pulles ont montré en 2023 que les hypothèses d'indépendance clés utilisées dans l'analyse de ces récentes attaques à base de réseaux étaient erronées. Cela a laissé ouverte la question de savoir comment les analyser et si elles pouvaient réellement fonctionner comme prévu. Nous concevons une légère variante de ces attaques et fournissons de nouveaux outils pour les analyser sans ces hypothèses. En particulier, nous montrons que notre attaque s'approche dangereusement des revendications de sécurité de Kyber et que nous surpassons légèrement ces récentes attaques. Cela règle la controverse sur la question de savoir si une attaque duale (par crible) pouvait réellement fonctionner comme prévu.

Contents

Introduction

Post-Quantum Cryptography

The goal of public key cryptography is to enable confidential and authenticated communication over insecure channels without requiring a pre-shared secret. For this purpose, the security of almost every public-key scheme relies on the difficulty of solving some well-defined computational problems. For example the security of the first key-exchange scheme [DH76] relies on the difficulty of solving the discrete log problem and that of RSA [RSA78], the first ever public key encryption scheme relies partly on factoring large composite numbers. Both are at the moment widely used over the internet. Shortly after, McEliece [McE78] proposed an encryption scheme whose security relied partly on the hardness of generically decoding linear codes, however this scheme was not considered practical at that time.

Importantly, those underlying problems are conjectured to be hard. More precisely it is often conjectured that there exists a polynomial f in a security parameter $\lambda \in \mathbb{N}$, such that the complexity of the best algorithm solving this problem for an instance of size $f(\lambda)$ is 2^{λ}. This makes the schemes practical and scalable as it ensures that the communicating parties have an exponential advantage over an adversary. For the most widely used schemes those conjectures are put to the test by sometimes decades of active research and the complexity of the best algorithms are used to parametrize the scheme.

In a breakthrough paper, [Sho94] exhibited a quantum polynomial algorithm which allowed solving the problem underlying the security of [DH76] and [RSA78], making them insecure if a sufficiently powerful quantum computer came to exist. This is a major threat to the security of the world communication as both schemes are still overwhelmingly used over the internet. Together with the recent advances in quantum computing this triggered an international research effort to find schemes that are assumed to be quantum resistant, a.k.a Post-Quantum schemes. Quite amazingly the McEliece scheme, which is code-based and was one of the earliest ever PKE proposal belongs to this category. Since then there have been many new scheme proposals based on many different computational problems, some lattice-based, code-based, multivariate, isogney and hash-based schemes.

In 2017 the National Institute of Standards and Technology (NIST) launched a competition to standardize post-quantum schemes which partially ended in 2022 with the choice of 4 schemes, that are now standardized, an encryption scheme, Kyber (ML-KEM) and 3 signatures schemes Dilithium (ML-DSA), Falcon (FN-DSA) and Sphincs+ (SLH-DSA). Except for SPHINCS+, all of these schemes are based on the computational hardness of problems that involve structured lattices. To find alternatives which did not rely on these problems NIST continued the competition up until a few months ago when the only contender left where only code-based: the encryption schemes HQC [AAB$^+$22b], BIKE [AAB$^+$22a] and Classic McEliece [ABC$^+$22] which was a proposal based on the original scheme of McEliece. Finally,

very recently, HQC was selected as a winner to back up Kyber and will soon be standardized. Last, in 2022 the NIST launched another competition to standardize additional alternative signature schemes to those already standardized. To this day this competition is still ongoing and includes both lattices based and code-based schemes.

The decoding problem and code-based cryptography

At its core the problem is simply to solve an under-determined linear system but with an additional non-linear constraint that makes the problem hard. In its most simple form you are given say a binary matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$, a parity-check matrix, and a column vector $\mathbf{s} \in \mathbb{F}_2^{(n-k)\times 1}$, a syndrome, and the goal is to return if it exists, a row vector $\mathbf{e} \in \mathbb{F}_2^n$ of prescribed small Hamming weight $|\mathbf{e}| = t$ and such that $\mathbf{H}\mathbf{e}^\intercal = \mathbf{s}$. Without this constraint on the weight, the problem could easily be solved with linear algebra, but this weight constraint makes it strikingly harder. In fact, the associated decision problem is NP-complete and some natural average variant of this problem is conjectured to be hard as even after 60 years of active research and a very long line of work, for an appropriate choice of n, k, t, the average complexity of all known decoders is exponential in the error weight t.

An historic problem

In fact this decoding problem takes its root from coding theory, well before code-based cryptography. Coding theory was introduced 70 years ago by [Sha48] to transmit reliably information through noisy channels by adding redundancy. Say for example that the channel takes binary symbols and that each transmitted bit has a small probability of being flipped. One efficient way to add this redudancy is by encoding a message $\mathbf{m} \in \mathbb{F}_2^k$ into a longer binary word of length n. This is typically done using a binary linear code of length n and dimension k, namely a linear subspace of dimension k of the vector space $\mathbb{F}_2^n$, which is typically given under the form of a generator matrix $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ (i.e. a basis) of the code $\mathcal{C}$ that is such that $\mathcal{C} = \{\mathbf{m}\mathbf{G} \ : \ \mathbf{m} \in \mathbb{F}_2^k\}$ and transmitting the codeword $\mathbf{c} = \mathbf{m}\mathbf{G}$. After going through the channel, the receiver gets the noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c}$ is the transmitted codeword and $\mathbf{e}$ is an error vector coming from the noisy channel and which has presumably a small Hamming weight. The receiver's goal is then to recover $\mathbf{c}$ and hence $\mathbf{m}$ from this noisy codeword $\mathbf{y}$. This is basically the telecomunication variant of the decoding problem. One can readily see that this problem is in fact almost equivalent to the decoding problem defined above. Indeed, the receiver can compute with linear algebra a parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ of $\mathcal{C}$, that is a matrix such that $\mathcal{C} = \{\mathbf{H}\mathbf{c}^\intercal = \mathbf{0} \ : \ \mathbf{c} \in \mathcal{C}\}$ and compute the syndrome $\mathbf{H}\mathbf{y}^\intercal = \mathbf{H}(\mathbf{c}+\mathbf{e})^\intercal = \mathbf{H}\mathbf{e}^\intercal$. And conversly, given a syndrome $\mathbf{H}\mathbf{e}^\intercal$ and a matrix $\mathbf{H}$ one can compute easily with linear algebra a noisy codeword $\mathbf{y}$ which is such that $\mathbf{y} = \mathbf{c} + \mathbf{e}$. As such finding the closest codeword is really just finding the error $\mathbf{e}$ of smallest Hamming weight which has this specific syndrome. This left open for decades two major open questions: (1) what specific codes to choose to ensure that with good probability the closest codeword is actually the sent codeword $\mathbf{c}$, and (2) what procedure can we devise to actually find this closest codeword? Concerning the first point, there exist theoretical bounds that give the minimum amount of reduncancy one needs which is given in terms of the code rate $R \stackrel{\text{def}}{=} k/n$ in order to transmit reliably over a channel with a certain noise amount. Namely, Shannon's theorem states that if the coordinates of the error are chosen i.i.d according to a Bernoulli

distribution of parameter p, then essentially the rate of the code needs to be smaller than the channel capacity $R < 1 - h(p)$ where $h(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ is the binary entropy function. Conversely, it was shown that an overwhelming majority of linear codes achieve this bound, namely picking a linear code at random ensures with overwhelming probability that the picked code is optimal in that sense but as mentionned earlier but it was soon noticed that the decoding step was hard in general. This prompted a huge research effort to find specific families of linear codes which where good and for which we could find an efficient decoder, understand a decoder that could decode up to these optimal bounds and which runs in polynomial time in the length of the code. An early code proposal relevant in our context was made by Gallager [Gal63]. It relied on the creation of codes with extremely low weight (constant) dual codewords, namely LDPC. The dual $\mathcal{C}^\perp$ of a code linear code $\mathcal{C}$ is defined as

$$\mathcal{C}^\perp \stackrel{\text{def}}{=} \{\mathbf{h} \in \mathbb{F}_2^n \ : \ \langle \mathbf{c}, \mathbf{h} \rangle = 0, \quad \forall \mathbf{c} \in \mathcal{C}\}$$

and is itself a linear code whose codewords $\mathbf{h} \in \mathcal{C}^\perp$ are called dual codewords, or parity-checks of $\mathcal{C}$. Gallager key observation is that when $\mathbf{h}$ is a dual codeword of $\mathcal{C}$, namely $\mathbf{h} \in \mathcal{C}^\perp$, we have that

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{c} + \mathbf{e}, \mathbf{h} \rangle = \langle \mathbf{e}, \mathbf{h} \rangle = \sum_{i=1}^{n} e_i h_i = \sum_{i \in [\![1,\, n]\!] \,:\, h_i = 1} e_i$$

is essentially and crucially more biased toward 0 as the Hamming weight of $\mathbf{e}$ and $\mathbf{h}$ get smaller. This can be leveraged to decode, and, essentially regardless the technique used, the rational is that the amount of information learned, per dual vector $\mathbf{h}$, about the error $\mathbf{e}$ gets bigger as the weight of $\mathbf{h}$ decreases. One simple way of decoding could be for example, to recover the first coordinate e_1 of the error $\mathbf{e}$ one could consider a certain number of low weight dual vector involving this coordinate and decide that e_1 is 0 or 1 based on whether $\langle \mathbf{y}, \mathbf{h} \rangle = e_1 + \left\langle \mathbf{e}_{[\![2,\, n]\!]}, \mathbf{h}_{[\![2,\, n]\!]} \right\rangle$ is more tilted toward 0 or 1. Gallager's LDPC construction essentially consists in building a sparse dual basis: this is done by choosing some linearly independent vectors of very small, constant, Hamming weight. Concretely his decoder is more involved than the one presented above.

For code-based cryptograhy

Starting with McEliece a vast number of encryption schemes used codes that could be decoded to create trapdoor one-way functions allowing the design of Public-Key Encryption (PKE). The idea is basically to choose at random a code in a family of codes with an efficient t-error correcting procedure and publish an arbitrary basis of that linear code. Encryption can then be done by sending a noisy codeword (the secret is the error or the codeword) and decryption is done using the trapdoor decoder. Essentially the security of these schemes rely on the incapacity of the adversary to recover the efficient decoder from an arbitrary basis and on the difficulty of the decoding problem. The first instantiation of this was done by McEliece with Goppa codes, which is still unbroken. Since then proposals around Gallager's idea of using code with low weight vectors in the dual have emerged, the difficulty being that his codes cannot be used as is as the constant weight dual basis can be recovered in polynomial time from any basis of the code. Some proposal [BC07] was made using disguised QC-LDPC codes but was broken in [OTD10]. Then, there is the BIKE approach that increases the weight of the secret dual basis from constant to weight $\Theta(\sqrt{n})$. This basically allows to make it hard enough to recover the secret basis but at the cost of a smaller decoding distance

(which decreases the performance of the scheme) as, for each vector $\mathbf{h} \in \mathcal{C}^\perp$ of the secret dual basis $\langle \mathbf{y}, \mathbf{h} \rangle$ now contains way less information about the error compared to the LDPC context. Another approach was proposed by Alekchnovich [Ale03], his idea was to really take a random code but to plant in it a single secret moderate weight $\Theta(\sqrt{n})$ dual vector $\mathbf{h}$. The encryption of 0 is made by sending several $\mathbf{y}$ which are chosen uniformly at random in $\mathbb{F}_2^n$ and the encryption of 1 is made by sending noisy codewords, say $\mathbf{y} = \mathbf{c} + \mathbf{e}$, where $|\mathbf{e}| = \Theta(\sqrt{n})$. The receiver would then learn if 1 or 0 was sent by observing if $\langle \mathbf{y}, \mathbf{h} \rangle$ is tilted toward 0 or not. The advantage of this scheme is that it can be shown that its security solely relies on the decoding problem. Eventually this lead, with various improvement, to the HQC cryposystem which is now a NIST standard. Note that both HQC and BIKE add a quasi-cyclic structure to the underlying code for efficiency. Even if there are no known polynomial reduction from the average decoding to the average decoding of quasi-cyclic codes, the quasi-cyclic structure is not believed to weaken much the scheme as the existing techniques [Sen11] that leverage this quasi-cyclic structure to decode only gain a polynomial factor. This explains why it is highly relevant, even in this quasi cyclic context to study the difficulty of the standard average decoding problem. The security of both these scheme rely on the average (quasi-cyclic) decoding problem with a distance that is sublinear in the code length, essentially $\Theta(\sqrt{n})$ and with a code rate $R = k/n$ which is equal to $1/2$ for BIKE and around $1/3$ for HQC.

On the other spectrum of asymmetric cryptography, the difficulty of the decoding problem is also used to build code-based signature scheme. Many are constructed from building Zero-knowledge proof, starting with [Ste93], and applying the generic Fiat-Shamir transform [FS87]. Those schemes do not require a trapdoor decoder and thus the best performance is obtained with random codes and an error weight close the so called Gilbert-Varshamov bound, which is the weight for which the decoding problem is the hardest.

Generic decoders and hardness in the constant rate regime

The **main average problem** that we will target in this thesis is the following standard variant of the average decoding problem given as follows and that we state for convenience in its generator matrix form.

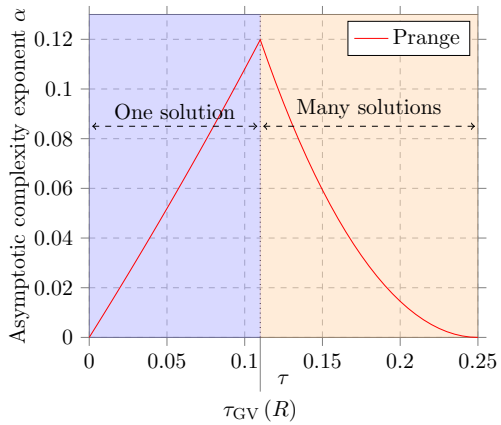
Definition 1 (The average decoding problem, generator matrix variant)**.** *Let $k, n, t \in \mathbb{N}$ be such that $k \leqslant n$ and $t \leqslant n$. Given $(\mathbf{G},\ \mathbf{y})$ and n, k, t where*

- $\mathbf{G}$ *is taken uniformly at random in $\mathbb{F}_2^{k \times n}$,*
- $\mathbf{y}$ *is taken as $\mathbf{y} = \mathbf{sG} + \mathbf{e}$ where the secret $\mathbf{s}$ is distributed uniformly at random in $\mathbb{F}_2^k$ and $\mathbf{e}$ is distributed uniformly at random among vectors of $\mathbb{F}_2^n$ of Hamming weight t,*

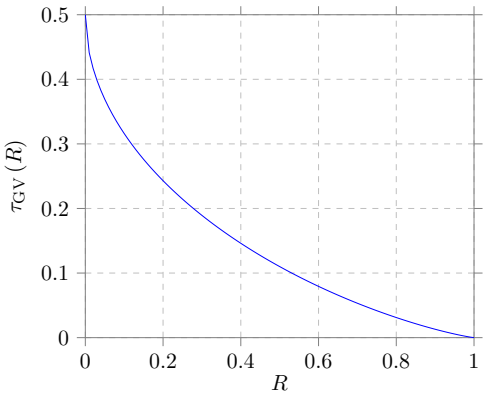
the goal is to find a vector $\mathbf{s} \in \mathbb{F}_2^k$ such that $|\mathbf{y} - \mathbf{sG}| = t$.

Crucially, the complexity of the best generic decoders, i.e. algorithms to solve this problem, are used to choose the parameters of the aforementioned code-based schemes. We will mostly study its hardness asymptotically in the traditional constant rate $R = k/n$ regime: the security of the aforementioned schemes rely on the difficulty of this problem in this constant rate regime.

Information set decoders The first non-trivial decoder (other than naively enumerating the secret or the error) was introduced by Prange [Pra62] in 1962. Basically he noticed that it was sufficient to know the value of the error $\mathbf{e}$ on k positions to recover the secret $\mathbf{s}$ with basic linear algebra. His idea was to select k positions at random and make the bet that they contain no error. This bet makes sense because the error is of low Hamming weight. Prange iterates with k fresh positions each time until eventually this bet is valid and the solution can be recovered. The complexity of this algorithm is essentially the cost of the linear algebra to solve the system, this is a polynomial in n, times the number of iterations, this is exponential in t and n for the right choices of parameters. In particular the problem is the hardest when t is the equal to the so called Gilbert-Varshamov distance, from this point the problem becomes easier because the number of solutions increases.



(a) Plot of α, the asymptotic time complexity exponent of Prange as a function of the relative weight τ at rate $R = 0.5$: the decoder solves the decoding problem given in Definition 1 with probability $1 - o(1)$ and in time, up to polynomial factors, $2^{\alpha n}$ when $t/n \to \tau$ and $k/n \to 0.5$.



(b) Relative Gilbert-Varshmov distance $\tau_{\mathrm{GV}}(R) \stackrel{\mathrm{def}}{=} h^{-1}(1-R)$ where $h^{-1}()$ is the inverse on $[0, 1/2]$ of the binary entropy function defined as $h(x) \stackrel{\mathrm{def}}{=} -x \log_2(\mathbf{x}) - (1-x)\log_2(1-x)$.

Since Prange there has been many improvements to ISD's, to name a few, [LB88, Ste88, Dum89, MMT11, BJMM12, MO15, BM17, BM18, Ess23, GJN24, DEEK24], the state of the art is given by [BM18]. For 60 years the ISD's have been the dominant family of generic decoders and are essentially the best decoders when decoding codes of constant rates. All those improved algorithm share with Prange the fact that they make a certain bet on k (or more) positions of the error, but the improvement comes by relaxing the bet and allowing a few number of positions of the error to be 1's. The goal now shifts to recovering which are those erroneous positions. In some cases finding those really is solving a smaller decoding problem in a related code of higher rate but at a smaller distance. Those ISD benefits massively from the fact that in those regimes there exist relatively efficient combinatorial decoders that can be used to decode. The first of which can be traced back to [Dum86]: it is a collision based approach gaining essentially a square root factor over the naive enumeration of the errors.

Very notably it turns out that when the decoding distance t is so below the Gilbert-Varshamov distance that it is sublinear in the codelength, that is $t = o(n)$, then the complexity of Prange and all subsequent ISD's improvements is some $2^{-t\log_2(1-R)(1+o(1))}$ [CS16] where the difference is only quantifiable in the second order term hidden in $o(1)$ and it remains

a major open question to find better decoder in this regime. This is the main reason why the security of the aforementioned encryption schemes (HQC, BIKE, Classic McEliece) is so stable.

Statistical decoding a.k.a the first dual attack In 2001, Al-Jabri's introduced statistical decoding [Jab01], a decoder not belonging to the ISD family. It can be considered as the first dual attack. His idea is to reuse Gallager's idea: concretely Al-Jabri, to recover the first coordinate of the error, computes many dual vectors $\mathbf{h}$ of low weight that involves this first coordinate (i.e. $h_1 = 1$) and decide that $e_1 = 0$ if

$$\langle \mathbf{y}, \mathbf{h} \rangle = e_1 + \left\langle \mathbf{e}_{[\![2,\, n]\!]}, \mathbf{h}_{[\![2,\, n]\!]} \right\rangle$$

is more tilted toward 0 than 1. The main difference is that because the code is random, for constant rates, we expect that even the smallest dual vector is of weight $\Omega(n)$ contrary to the weight in $\mathcal{O}(1)$ in the LDPC context, so each dual vector yield much less information about the error. Second, there is also the difficulty of computing those low weight dual vectors which is also a hard problem and is closely related to the decoding problem as most of the existing decoders can be used to solve the average decoding problem can be used to find low weight codewords in a random code. In this case one can show that the number of dual vectors to make a meaningful decision needs to be exponential in the error weight t and [DT17a] showed that statistical decoding did not compete with ISD.

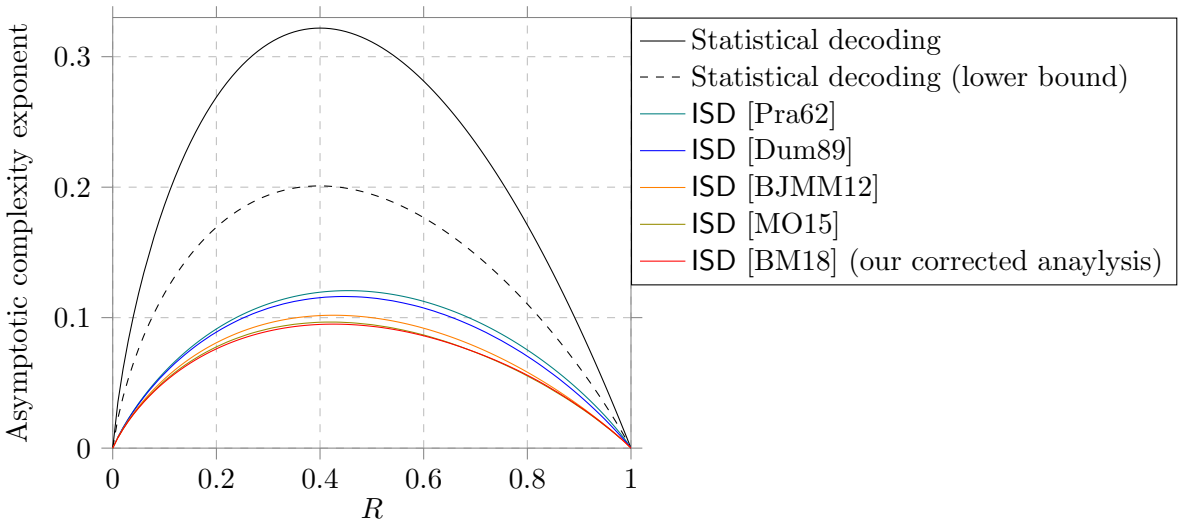


Figure 2: Asymptotic complexity exponent of some generic decoders when decoding random codes of rate $R = k/n$ at the Gilbert-Varshamov distance. The plain black statistical decoding curve is the asymptotic complexity of [Jab01] devised in [DT17a, Figure 7, Statistical Decoding]. The dashed black statistical decoding (lower bound) curve is the asymptotic complexity of Statistical decoding when the cost of computing the dual vectors is overlooked, this was given by [DT17a, Figure 7, Optimal Statistical Decoding].

Decoders for the very low rate regime, the LPN setting

Alternatively, there exist many code-based cryptographic constructions whose security rely on a slight variant of the aforementioned decoding problem called the Learning Parity with

Noise (LPN). In this variant one is given access to an oracle that when called outputs a noisy linear combination of a secret $(\mathbf{g}, \langle \mathbf{s}, \mathbf{g} \rangle + e) \in \mathbb{F}_2^k \times \mathbb{F}_2$ where $\mathbf{s} \in \mathbb{F}_2^k$ is some fixed unknown secret, $\mathbf{g}$ is uniformly random in $\mathbb{F}_2^k$ and e is a noise following a Bernoulli distribution of fixed known parameter. The goal is to recover the secret $\mathbf{s}$ with as many calls to the oracle as wanted. Putting the $\mathbf{g}^\intercal$ inside the columns of a generator matrix $\mathbf{G}$, this really is a decoding problem $\mathbf{y} = \mathbf{sG} + \mathbf{e}$ but now the length n can get arbitrarily large (and the error distribution is different from the average problem presented above). The fact that we have access to much more (i.e. we decode well below the Gilbert-Varshamov distance) samples than what is information-theoritically required to recover the secret makes the problem much easier. But, it is still conjectured to be hard and the first solver/decoder BKW [BKW03] showed that it could be solved in time and number of samples in $2^{\Theta(k/\log k)}$ for any Bernoulli with constant noise. Basically the idea of BKW and all subsequent solvers is to somehow reduce solving this LPN problem to another related LPN problem of smaller dimension, but with increased noise. This dimension reduction is done by finding some $\mathbf{g}_i$'s whose sum is zero on some fixed coordinates. Summing the associated samples $y_i = \langle \mathbf{s}, \mathbf{g}_i \rangle + e_i$ yields a new sample involving less coordinates of the secret but with a bigger noise. Finding such sum is done with a collision technique that successively cancels the vectors block by block. As an example BKW computes many such reduced samples by reducing to a dimension one problem and solve the resulting problem by majority voting. In fact one can see that this technique really is a kind of dual attack where one computes low-weight dual vectors $\mathbf{h}$ of the code generated by $\mathbf{G}$ where we deleted the first row, this yield the sample

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{sG} + \mathbf{e}, \mathbf{h} \rangle = \langle \mathbf{s}, \mathbf{hG}^\intercal \rangle + \langle \mathbf{e}, \mathbf{h} \rangle = \langle s_1, (\mathbf{hG}^\intercal)_1 \rangle + \sum_{i \in \mathrm{Supp}(\mathbf{h})} e_i.$$

The main difference with statistical decoding is that in this regime there exists dual vectors of extremely low weight ($\sqrt{k}$ essentially) which we can compute efficiently with the collision technique introduced in BKW. This is what makes this attack competitive here. BKW was later improved by [LF06] which relaxed the dimension loss by reducting to some higher (than one) dimensional problem and solving it efficiently using a Fast Fourier Transform (FFT) decoder whose use can be traced back to decoding Reed-Muller codes [Gre66]. This was improved by [GJL14] who introduced a novel secret compression technique to further reduce the dimension of the problem.

The LWE problem and lattice-based cryptography

In his seminal work, [Reg05] introduced a generalization of the decoding problem which is called Learning With Error (LWE). The difference being that the considered linear system lies in a much larger ring $\mathbb{Z}_q$ with say $q > 2$ instead of the binary field $\mathbb{F}_2$ and the metric changes from the Hamming weight to a much finer one: the coordinates of the error $\mathbf{e} \in \mathbb{Z}_q^n$ are chosen to be small compared to q. Here we qualify "small" by identifying $\mathbb{Z}_q$ with the set of integers (say q is odd) $\{ -(q-1)/2, \cdots 0, \cdots, (q-1)/2\}$.

Definition 2 (The LWE problem)**.** *Let $n, k, q \in \mathbb{N}$ with $k \leqslant n$ and let χ be a distribution with support in $\mathbb{Z}_q$. Given n, k, q, χ and $(\mathbf{G},\ \mathbf{y})$ where*

- $\mathbf{G}$ *is taken uniformly at random in* $(\mathbb{Z}_q)^{k \times n}$,

- $\mathbf{y}$ *is taken as* $\mathbf{y} = \mathbf{sG} + \mathbf{e}$ *where the secret* $\mathbf{s}$ *is taken uniformly at random in* $\mathbb{Z}_q^k$ *and where the coordinates of the error* $\mathbf{e}$ *are taken identically and independently at random according to* χ*,*

the goal is to recover $\mathbf{s}$*.*

In fact Regev introduced an encryption scheme whose security is based on the hardness of this problem and which can very essentially be seen as the analog of Alekhnovich scheme to this generalized setting. With various improvements this scheme eventually yielded Kyber which is now a NIST standard. One of which came from the introduction, for efficiency purposes, of an underlying polynomial ring structure, but it is not believed to degrade the hardness of the problem by more than polynomial factors in practice. This will be the reason why we stick, as it is traditionally done for generic attacks, to the study of the hardness of the problem defined over $\mathbb{Z}_q$. More generally, the hardness of LWE is now also at the center of a wide variety of cryptographic constructions, ranging from signatures schemes with the NIST standard Dilithium to Homomorphic Encryption schemes [BV11]. As such, finely studying the difficulty of this LWE problem is crucial as the complexity of the best solvers is used to select the parameters of those schemes.

In practice χ is often symmetric and concentrated around 0, and is a discrete Gaussian distribution in Regev's original work, or a closely related centered binomial distribution in Kyber. This naturally involves the Euclidean metric as in that case one can argue that the error $\mathbf{e}$ is roughly uniformly distributed in some Euclidean ball of small known radius and the problem really becomes to find the $\mathbf{s}$ such that the $\mathbf{y} - \mathbf{sG}$ is in that Euclidean ball. The two aforementioned schemes are deep down the injective regime where $\mathbf{s}$ is the unique solution with overwhelming probability. Contrary to the standard decoding problem we presented earlier, taking a higher modulus and using a finer norm (than the Hamming weight) at the same time somewhat changes the source of hardness of this problem in terms of both the underlying security reduction and the techniques used to solve it. Let us explain both these points.

Lattices

Regev [Reg05] gave arguments for the security of his scheme by showing that the average LWE problem was quantumly harder than some worst-case lattice problems that where difficult in practice. This is one of the reason why the schemes whose security are based on the hardness of the LWE problem are called lattice-based schemes. More precisely, a lattice Λ is a discrete subgroup of $\mathbb{R}^n$, that is it can be written as $\Lambda = \mathbf{B}\mathbb{Z}^n$ for some basis $\mathbf{B} \in \mathbb{R}^{n \times n}$. Regev showed that there was a polynomial quantum reduction from the problem of approximating (up to a polynomial factor $\sqrt{n}$) the length of the shortest non-zero vector in any given lattice to the LWE problem. And, while [AR05] showed that this last approximation problem was in $\mathrm{NP} \cap \mathrm{CoNP}$ it is also believed to be hard in practice as the best known algorithms solving this problem are sieving style algorithms [AKS01] that runs in exponential time $2^{\mathcal{O}(n)}$.

Solving LWE

Solving LWE in our case of interest is finding the closest (in terms of Euclidean norm) noisy codeword $\mathbf{y}$ in the code $\mathcal{C}$ generated by the given matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$. One standard way to

solve this problem is by periodizing the associated code $\mathcal{C}$ generated by $\mathbf{G}$ to make it a lattice by constructing:

$$\Lambda = \mathcal{C} + q\mathbb{Z}^n.$$

In that case solving LWE is finding the closest vector of the lattice $\Lambda \in \mathbb{R}^n$ to $\mathbf{y}$, seen as a vector in $\mathbb{R}^n$. This can then be turned into the problem of finding the shortest vector in a related lattice, which can be solved in time $2^{\mathcal{O}(n)}$ with sieving-style algorithm [AKS01], the best in terms of asymptotic time complexity is given by [BDGL16], whose technique allows one to heuristically find the shortest vector in time $2^{0.292n(1+o(1))}$. These are called primal attacks because they manipulate primal vectors, i.e. vectors in Λ, and were until recently the dominant family of solvers. One can note that this reduction does not work to decode a q-ary linear code with an error of low Hamming weight. In that case the best algorithm is essentially a generalization of the aforementioned Prange decoder, which becomes less and less interesting as the associated metric gets finer and is hence slightly less interesting for solving LWE.

Similar to coding theory and starting from [AR05], there exist dual attacks to solve LWE and were at first did not compete against primal attacks. The base idea is the same: by computing small dual vectors $\mathbf{h} \in \mathcal{C}^\perp$ we can leverage the fact that

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{c} + \mathbf{e}, \mathbf{h} \rangle = \langle \mathbf{e}, \mathbf{h} \rangle$$

is more biased toward the small values of $\mathbb{Z}_q$ as $\mathbf{e}$ and $\mathbf{h}$ gets smaller. This becomes lattice-related as, again, computing those small dual vectors can be done by periodizing $\mathcal{C}^\perp$ and using sieving algorithms as noted by [ADPS16b] for example. Those dual attacks were successively improved [Alb17, EJK20, GJ21, MAT22] up until those last two work that claimed to reduce the security of Kyber in particular. Recently [DP23b] showed that some independence assumptions that are at the center of the analysis of those recent attacks are flawed, and it is left as an open question how to actually analyze those attacks without these assumptions and if they really work as expected.

Contributions

Publications and preprints

Our main contributions lie in the design and analysis of new dual attacks in code-based and lattice based cryptography. This thesis main results are based on the following publications and preprints, we however rewrote them in a slightly different way, sometime including additional results.

- [CDMT22] Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, Jean-Pierre Tillich. "Statistical Decoding 2.0: Reducing Decoding to LPN". In : Advances in Cryptology – ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part IV

- [MT23] Charles Meyer-Hilfiger, Jean-Pierre Tillich. "Rigorous Foundations for Dual attacks in coding theory". In : Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV.

- [CDMT24] Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, Jean-Pierre Tillich. "Reduction from Sparse LPN to LPN, Dual Attack 3.0". In : Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part VII

  - A software and its documentation published at Eurocrypt 2024 Artifact to justify the claims made in the article. Available at `https://artifacts.iacr.org/eurocrypt/2024/a10/`

- [CMST25] Kévin Carrier, Charles Meyer-Hilfiger, Yixin Shen, Jean-Pierre Tillich. "Assessing the Impact of a Variant of MATZOV's Dual Attack on Kyber". To appear in Advances in Cryptology – CRYPTO 2025. ePrint available at `https://eprint.iacr.org/2022/1750`.

- Charles Meyer-Hilfiger. "Proving modern code-based dual attacks". To appear soon on ePrint.

Note that one of my contributing article [CMST25] is in fact an updated version of [CST22], a late 2022 preprint with the same authors except me. The main addition of this update is a new corrected analysis of the dual attack removing some commonly used independence assumptions that were shown to be flawed in [DP23b].

Summary of contributions

Code-based

The main contribution of this thesis is to significantly develop and improve dual attacks in code-based cryptography and to gain a fine understanding of their behavior.

In particular, our best attack asymptotically significantly beats all previously known decoders for codes of constant rates smaller than 0.42 at the Gilbert-Varshamov distance. It gives for the first time after 60 years, a decoding algorithm that asymptotically beats ISD's for a significant range and that makes a significant change in this lower rate regime. Our complexity results are given in Fig. 3. These results were obtained by revisiting Al-Jabri's statistical decoding algorithm and generalizing it using a splitting strategy to reduce decoding to a problem that is essentially the Learning Parity with Noise (LPN) problem. We then solve it using standard solvers.

As a side note, we found an error in the original analysis of the state of the art [BM18] that we show lead to a very significant underestimation of the asymptotic complexity of this ISD. We corrected its analysis and thus established a corrected reference for the state-of-the-art.

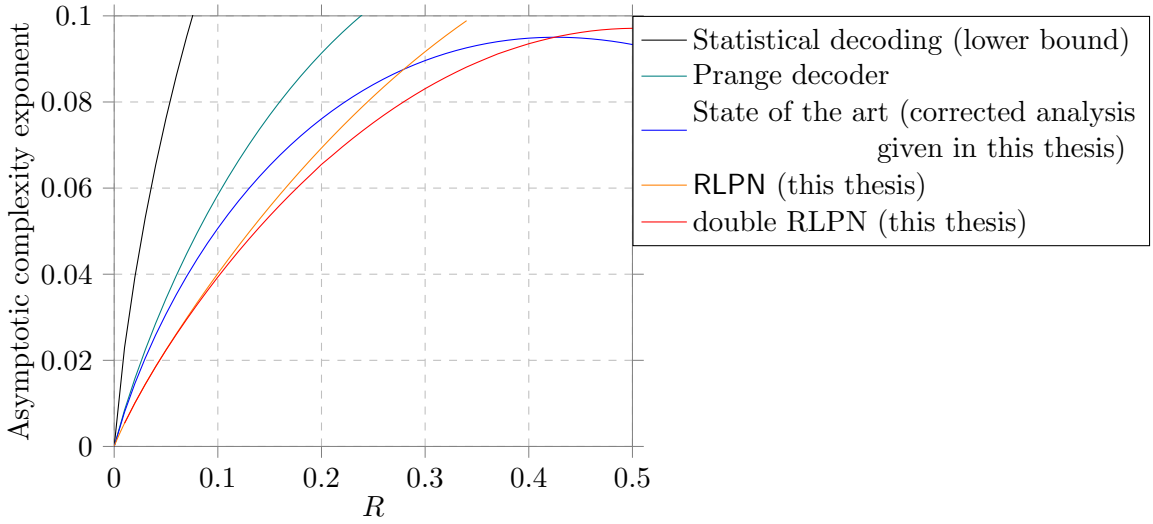


Figure 3: Asymptotic complexity exponent of some generic decoders when decoding random codes of rate $R = k/n$ at the Gilbert-Varshamov distance. The state of the art curve is given by the ISD given in [BM18] but with our corrected analysis.

The second main code-based contribution of this thesis lies in developing new tools to analyze dual attacks and giving algorithmic tweaks to make their analysis tractable without assumptions. In particular, we show experimentally that the standard independence assumptions used to analyze statistical decoding cannot be used to analyze our attacks in general. We develop an alternative path for analyzing our attacks without these independence assumptions. Our tools are based on the Poisson summation formula and a model for the distribution of the weight enumerator of random linear codes. We base the analysis of our attacks on this model and verify it experimentally. Finally, at the end of this thesis, we devise a variant of our most advanced attack that has the same performance up to polynomial factors but which we can fully prove without using any model.

Lattice-based

Our main lattice-based contribution lies in the design of a new lattice-based dual attack against LWE. Our attack is a variant of the recent lattice-based dual attack by [GJ21, MAT22] that claimed to diminish the security of Kyber but whose result were strongly questioned by [DP23b] who showed that the crucial independence assumptions underlying their analysis were flawed. These independence assumptions can be viewed as the lattice-based counterpart of the aforementioned code-based independence assumptions that where traditionally used for the analysis of dual attacks. In particular, we develop new tools to analyze those dual attacks without these assumptions and use them to show that our attack reduces the security of Kyber: our attack cost is 3.5/11.9/12.3 bits below the NIST requirements of 143/207/272 for Kyber-512/768/1024 where we used the same cost model as in [SAB+20, MAT22]. The cost of our attack matches and even slightly improves in some cases the complexities originally claimed by [MAT22]. This positively settles the recent controversy as to whether a lattice-based dual-sieve attacks can really work as expected.

Details and chapters outline

Code-based

Let us give more details about our contributions. The code-based dual attacks we devise can be seen as the analog of the recent lattice-based dual attacks: it benefits from a splitting strategy that was originally suggested but not exploited in [DT17b] and is also central to the competitiveness of recent lattice-based dual attacks [Alb17, EJK20, GJ21, MAT22]. We split the support $[\![1, n]\!]$ in two complementary subparts $\mathscr{P}$ and $\mathscr{N}$ and compute dual vectors $\mathbf{h}$ of that are of low-weight only on the part $\mathscr{N}$, each dual vector yields a noisy linear combination of the error $\mathbf{e}_{\mathscr{P}}$:

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{c} + \mathbf{e}, \mathbf{h} \rangle = \langle \mathbf{e}_{\mathscr{P}}, \mathbf{h}_{\mathscr{P}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle . \tag{1}$$

This is an LPN sample with secret $\mathbf{e}_{\mathscr{P}}$ and noise $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$. The rationale is to increase the size of $\mathscr{P}$ to naturally decrease the weight of $\mathbf{h}_{\mathscr{N}}$, thus increasing the bias ε of $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$ and hence decreasing the number of required dual vector to recover $\mathbf{e}_{\mathscr{P}}$. Simplifying a bit, one expects to require poly $(n)/\varepsilon^2$ such vectors where one can show that ε is exponentially small. This, of course, comes at an increased cost to actually recover $\mathbf{e}_{\mathscr{P}}$ but we can now balance it with the number of required dual vectors. We compute those sparse dual vectors using techniques coming from the ISD's and actually recover the secret $\mathbf{e}_{\mathscr{P}}$ with techniques coming from LPN solvers, that is with Fast Fourier Transform as used in [LF06] or a decoding technique [GJL14] to leverage the sparseness of the secret $\mathbf{e}_{\mathscr{P}}$ by compressing it with the help of a linear code. We call the former RLPN (Reducing Decoding to LPN) and the latter (which is a generalization of the former) double-RLPN. Note that the term LPN appearing in the name of our attack is motivated by the fact that we reduce decoding a code of constant rate to decoding some very long code of *reduced* dimension. But our samples are not distributed as true LPN samples, this is the reason why our dual attack suffers from technical complications in the algorithms and the analysis. In a nutshell, this all boils down to the fact that the considered dual vectors have a vast amount of intersection in their support making the $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$'s dependent of each others. This is a difficulty that also appeared in the analysis of statistical decoding, necessitating independence assumptions for its analysis [Ove06, DT17a] : the $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$'s are supposed to be mutually independent. We do not use those assumptions in our analysis. The outline of our code-based contributions chapters is given as follows.

- In the contribution Chapter 4 we provide the first assumption-free proof of a slight variant of statistical decoding. We do this with a second-order argument to give concentration bounds for estimating the bias of $\langle \mathbf{y}, \mathbf{h} \rangle$. This bias is the key quantity underlying the hardness and behavior of all dual attacks.

- In the contribution Chapter 5 we present RLPN. Notably, the introduction of this splitting strategy makes our analysis rely on some exponential strengthening of the second-order behavior of the aforementioned bias. We show experimentally that the standard independence assumptions cannot be used to that extend. To replace these assumptions, we state a new conjecture that we thoroughly validate experimentally with a new model. It relies on a dual expression for the bias coming from the Poisson summation formula and can be thought of as the code-based cousin of the duality formula used in the analysis of the first lattice-based dual attack [AR04].

- In the contribution Chapter 6 we present our best dual attack, double-RLPN. This chapter in essence contains a generalization of the two previous chapters. Reading this

chapter without reading the two previous ones is possible for a reader familiar with the topic.

- In the contribution Chapter 7 we present a slightly tweaked variant of double-RLPN which has essentially, up to polynomials factors, the same performance as the original double-RLPN algorithm but which we can fully prove, bypassing the need for any conjecture. Its analysis relies solely on the second-order concentration properties of the bias.

Those chapters are backed up by three state-of-the-art chapters starting from a short preliminary Chapter 1 providing the basics in coding theory. This is followed by Chapter 2 presenting the state of the art of generic decoders: we present there some ISD's and statistical decoding. This chapter contains as a contribution the correction of the analysis of [BM18]. Some of these ISD's will later be used inside our dual attack to produce the sparse dual vectors but reading this chapter is not necessary to understand our dual attack contribution chapters as we will use them almost as black boxes. The last state-of-the-art chapter is given by Chapter 3 where we present the LPN solvers that we will use to build our attack.

Lattice-based contributions

The new lattice-based dual attack we devise is in the same spirit as recent lattice-based dual attacks but we slightly improve it: our attack benefits from the same splitting strategy as given by Eq. (1) but in the LWE context, namely the vectors are in $\mathbb{Z}_q$ and are of small Euclidean norm. While recent attacks [GJ21, MAT22] reduce the cost of recovering the secret $\mathbf{e}_{\mathscr{P}}$ with different modulus switching techniques we use a generalization in the q-ary setting of the secret compressing technique that we used in double-RLPN to diminish the dimension and which was already used in [GJS16] in some related context. Here is the outline of our lattice-based contributions chapters.

- In contribution Chapter 8 we make a brief introduction to the LWE problem and lattice-based cryptography. As a contribution we develop the tools to analyze lattice-based dual attacks without using the flawed independence assumptions. Basically we do that by proposing a new model to predict the exponential behavior of key quantity underlying the behavior of these attacks in some simple case scenario. This key quantity is the "bias" (in some sense adapted to $\mathbb{Z}_q$) of $\langle \mathbf{y}, \mathbf{h} \rangle$. This is essentially the lattice-based analog of the conjecture we devised in the code-based Chapter 5 for RLPN.

- In contribution Chapter 9 we present our new dual attack against LWE and analyze it by generalizing the tools given in the previous chapter. We assess the complexity of our attack against Kyber.

Software

Finally, in Chapter 10 we give the documentation for an open source software we made to reproduce some claims made in Chapter 8 and Chapter 6. Its main purpose is to verify our model about the exponential behavior of the bias of $\langle \mathbf{y}, \mathbf{h} \rangle$ in codes and lattices. This also contains a mean to reproduce our asymptotic complexity results given in Fig. 3. As our algorithms have a lot of parameters, those asymptotic complexity exponents were obtained through extensive numerical optimization.

Notation

General notation

Sets

$\mathbb{Z}$ The ring of integers
$\mathbb{N}$ The set of non-negative integers
$\mathbb{R}$ The field of real numbers
$\mathbb{C}$ The field of complex numbers
$\mathbb{F}_q$ The finite field with q elements where q is a prime power.
$\mathbb{Z}_q$ The quotient ring $\mathbb{Z}/q\mathbb{Z}$ of integers modulo q
$[\![a,\, b]\!] = \{i \in \mathbb{Z} \,:\, a \leqslant x \leqslant b\,\}$ The set of integers between a and b, both included.
$[a,\, b] = \{x \in \mathbb{R} \,:\, a \leqslant x \leqslant b\,\}$ The set of real numbers between x and y, both included.
$]a,\, b[= \{x \in \mathbb{R} \,:\, a < x < b\,\}$ The set of real numbers between x and y, both excluded. $|\mathbf{x}|$ is the absolute value of $\mathbf{x} \in \mathbb{R}$.

Operations on sets

$|X|$ The cardinality or number of elements in the set X.
$A \setminus B = \{\, a \in A \,:\, a \neq B\}$
$A + B = \{\, a + b \,:\, a \in A,\, b \in B\}$
$A \times B = \{(a, b) \;\; : \, a \in A,\, b \in B\}$
$\overline{X}$ The complement of X: when X is a subset of Y, $\overline{X} = \{\, \mathbf{y} \in Y \,:\, \mathbf{y} \neq X\}$ is the complement of X in Y.
X^+ The subset of X composed of the non-negative elements of X

Operations on vectors and matrices

All the vectors are in row form. A matrix with 1 column and n rows is called a syndrome.
x_i the i'th coordinate of the vector $\mathbf{x}$
$\mathbf{x}_{\mathscr{I}}$ the projection of the vector $\mathbf{x}$ on the coordinates given by $\mathscr{I} \subset [\![1,\, n]\!]$.
$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i$ the dot-product of two length n vectors $\mathbf{x}$ and $\mathbf{y}$.
$\mathrm{Supp}\,(\mathbf{x}) = \{i \in [\![1,\, n]\!] \,:\, x_i \neq 0\,\}$ the support of a length n vector $\mathbf{x}$.
$\mathbf{A}^\intercal$ Transpose of the matrix $\mathbf{A}$
$\mathbf{x}^\intercal$ Transpose of a length n vector is a matrix with n rows and 1 column (a syndrome).
$\mathbf{A}_{\mathscr{I}}$ The matrix $\mathbf{A}$ where we keep only the columns given by $\mathscr{I}$
$\mathbf{A}_{\mathscr{I} \times \mathscr{J}}$ The matrix $\mathbf{A}$ where we keep only the columns given by $\mathscr{I}$ and the rows given by $\mathscr{J}$.
$\mathbf{I}_n$ is the identity matrix with n rows and columns.

$\mathbf{0}_n$ is the null vector of length n.
$\mathrm{rank}\,(\mathbf{A})$ is the dimension of the row space of $\mathbf{A}$.
$\dim(\mathcal{C})$ is the dimension of the linear space $\mathcal{C}$.

Norm and ball

$|\mathbf{x}| = |\{\ i \in [\![1,\ n]\!]\ :\ x_i \neq 0\}|$ The Hamming weight of the length n vector $\mathbf{x}$.
$||\mathbf{x}|| = \sqrt{\sum_{i=1}^{n} x_i^2}$ The Euclidean norm of the length n vector $\mathbf{x}$. When $\mathbf{x} \in \mathbb{Z}_q^n$ the coordinates x_i of $\mathbf{x}$ are seen as integers such that $|x_i| \leqslant q/2$
$||\mathbf{x}||_\infty = \max_{i \in [\![1,\ n]\!]} x_i$ The infinity norm of the length n vector $\mathbf{x}$
$\mathcal{S}_t^n = \{\ \mathbf{x} \in \mathbb{F}_2^n\ :\ |\mathbf{x}| = t\}$ The Hamming sphere of radius t.
$\mathsf{Ball}_t^n = \{\ \mathbf{x} \in \mathbb{R}^n\ :\ ||\mathbf{x}|| \leqslant t\}$ The Euclidean ball of radius t.

Standard functions

$\log_2$ is the logarithm in base 2. exp is the exponential function. e is Euler's number. $\lfloor x \rceil$ is the integer that is the nearest to x. $\lfloor x \rfloor$ is greatest integer that does not exceed x. $\lceil x \rceil$ is the smallest integer that is not less than x. $a\%q$ is a modulo q, sometimes written as $a\,(mod\,q)$.
$n! = n(n-1)\cdots 1$ is the factorial of n.
$\mathbf{1}_A$ is the indicator function of the set A, namely $\mathbf{1}_A(x) = 1$ if $x \in A$ otherwise 0.
$\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is the binomial coefficient. In particular $\binom{n}{k} = |\mathcal{S}_k^n|$.
$h(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ binary entropy function. $h^{-1}()$ is the inverse of $h()$ on $[0,\ 1/2]$.

Asymptotic notation

Definition 3. *$X \subset \mathbb{N}^n$ and let $f : X \to \mathbb{R}^+$ and $g : X \to \mathbb{R}^+$ be two functions. We write that*

- *$f \in \mathcal{O}(g)$ if there exists a constant $C > 0$ such that for any $\mathbf{x} \in X$ we have that $f(\mathbf{x}) \leqslant C\, g(\mathbf{x})$*
- *$f \in \Omega(g)$ is there exists a constant $C > 0$ such that for any $\mathbf{x} \in X$ we have that $f(\mathbf{x}) \geqslant C\, g(\mathbf{x})$*
- *$f \in \Theta(g)$ if there exists a constant $C > 0$ such that for any $\mathbf{x} \in X$ we have that $g(\mathbf{x})/C \leqslant f(\mathbf{x}) \leqslant C\, g(\mathbf{x})$*
- *$f \in o(g)$ if for every $\varepsilon > 0$ there exists $N > 0$ such that for any $\mathbf{x} \in X$ such that $||\ \mathbf{x}||_\infty > N$ then $f(\mathbf{x}) \leqslant \varepsilon g(\mathbf{x})$.*
- *$f \in \omega(g)$ if for every $W > 0$ there exists $N > 0$ such that for any $\mathbf{x} \in X$ such that $||\ \mathbf{x}||_\infty > N$ then $f(\mathbf{x}) \geqslant W\, g(\mathbf{x})$.*
- *$f \in \widetilde{\mathcal{O}}(g)$ if there exists two constants $C, \alpha > 0$ such that for any $\mathbf{x} \in X$ we have that $f(\mathbf{x}) \leqslant C\ ||\ \mathbf{x}||_\infty^\alpha\, g(\mathbf{x})$.*

- $f \in \tilde{\Theta}(g)$ *if there exists two constants* $C, \alpha > 0$ *such that for any* $\mathbf{x} \in X$ *we have that* $g(\mathbf{x}) / (C \mid\mid \mathbf{x}||_\infty^\alpha) \leqslant f(\mathbf{x}) \leqslant C \mid\mid \mathbf{x}||_\infty^\alpha g(\mathbf{x})$
- $f \in \tilde{\Omega}(g)$ *if there exists two constants* $C,\ \alpha > 0$ *such that for any* $\mathbf{x} \in X$ *we have that* $f(\mathbf{x}) \geqslant C \mid\mid \mathbf{x}||_\infty^\alpha g(\mathbf{x})$
- $f \in \text{poly}(g)$ *if there exists two constants* $C,\ \alpha > 0$ *such that for any* $\mathbf{x} \in X$ *we have that* $f(\mathbf{x}) \leqslant C\, g(\mathbf{x})^\alpha$

We sometimes replace the $\in$ *by an equality* $=$ *symbol.*

To stay simple and concise, we often use in the intermediate lemmas and for independent results, the multivariate variant of these landau notations while making the function implicit. For example the following will be used.

Proposition 1. *Let* $n \in \mathbb{N}$ *and let* $k \in \mathbb{N}$ *be such that* $k \leqslant n$*. We have that*

$$\binom{n}{k} = \tilde{\Theta}\left(2^{nh(k/n)}\right)$$

Remark 1. *Formally, the previous proposition claims that, if* X *is the subset of couples* (n, k) *of* $\mathbb{N}^2$ *such that* $k \leqslant n$ *and* $f : X \to \mathbb{R}$ *is the function defined as* $f(n,k) = \binom{n}{k}$ *and* $g : X \to \mathbb{R}$ *is the function defined as* $g(n,k) = 2^{nh(k/n)}$ *then we have that* $f = \tilde{\Theta}(g)$*.*

We will also often use the univariate counterpart of these landau notations, in particular when giving our main theorems. We sometimes emphasize that this unique variable grows to infinity even though it is already redundant from the definition.

Proposition 2. *Let* $n \in \mathbb{N}$ *be a variable growing to infinity, let* τ *be an implicit function of* n *with value in* $[0,\ 1]$ *and let* k *be the implicit function of* n *defined as* $k = \lfloor \tau n \rfloor$*. We have that*

$$\binom{n}{k} = \tilde{\Theta}\left(2^{nh(\tau)}\right).$$

We add that the $\tilde{\Theta}()$ *does not depend on* τ*.*

Remark 2. *In this case we claim that for any function* $\tau : \mathbb{N} \to [0, 1]$ *we have that* $f = \mathcal{O}(g)$ *where* $f : \mathbb{N} \to \mathbb{R}$ *is the univariate function defined as* $f(n) = \binom{n}{\lfloor \tau(n) n \rfloor}$ *and* $g : \mathbb{N} \to \mathbb{R}$ *is the univariate function defined as* $g(n) = 2^{nh(\tau(n))}$*. The last sentence of the proposition means that in fact there exists a constant* $C > 0$ *such that for any* $\tau : \mathbb{N} \to [0, 1]$ *we have that* $\forall n \in \mathbb{N},\ \ f(n) \leqslant C\, g(n)$*.*

Probabilities

We write that $X \sim \mathcal{D}$ to denote that the random variable X is distributed/sampled according to the distribution $\mathcal{D}$. We sometime say that X follows $\mathcal{D}$. We use i.i.d as an abbreviation for "identically and independently distributed". We write that $\mathbf{x} \sim \mathcal{D}^n$ to specify that $\mathbf{x}$ is a random vector of length n and where each coordinate is drawn i.i.d according to $\mathcal{D}$.

We write $\mathbb{P}_{X \sim \mathcal{D}}(X = x)$ to denote the probability that some random variable X distributed according to $\mathcal{D}$ is equal to x. When X is a random variable defined in the context

we simply write $\mathbb{P}(X = x)$ or sometime $\mathbb{P}_X(X = x)$. If X and Y are two random variable we denote by $\mathbb{P}(X = x \mid Y = y)$ the probability that $X = x$ conditioned on the event that $Y = y$. We denote by $\mathbb{E}(X)$ and $\mathbf{Var}(X)$ the expectation and variance respectively of X. The survival function of X is defined as $f(x) = \mathbb{P}(X \geqslant x)$. We denote by $\mathcal{U}(\mathcal{S})$ the uniform distribution over a set $\mathcal{S}$. The bias $\text{bias}(X)$ of a random variable X taking values in $\{0, 1\}$ is defined as $\text{bias}(X) \stackrel{\text{def}}{=} \mathbb{P}(X = 0) - \mathbb{P}(X = 1)$.

Proposition 3 (Markov inequality)**.** *For any α and any nondecreasing nonnegative function f such that $f(\alpha) > 0$ and any random variable X we have that*

$$\mathbb{P}(X > \alpha) \leqslant \frac{\mathbb{E}(f(X))}{f(\alpha)}.$$

Corollary 1 (Bienaymé–Chebyshev inequality)**.** *For any random variable X and any $\alpha > 0$ we have*

$$\mathbb{P}(|X - \mathbb{E}(X)| > \alpha) \leqslant \frac{\mathbf{Var}(X)}{\alpha^2}.$$

Probability distributions

$\text{Ber}(p)$ is the Bernouilli distribution of parameter p. By definition if $X \sim \text{Ber}(p)$ then $\mathbb{P}(X = 1) = p$ and $\mathbb{P}(X = 0) = 1 - p$.
$\text{Binomial}(n,\ p)$ is the Binomial distribution with n trials with success probability p. By definition if $X \sim \text{Binomial}(n,\ p)$ then $\mathbb{P}(X = i) = \binom{n}{i} p^i (1 - p)^{n-i}$.
$\text{Poisson}(\lambda)$ is the Poisson distribution of parameter λ. By definition if $X \sim \text{Poisson}(\lambda)$ then $\mathbb{P}(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$. The mean and variance of X are λ.
$\text{Exponential}(\lambda)$ is the Exponential distribution of parameter λ. By definition if $X \sim \text{Exponential}(\lambda)$ then $\mathbb{P}(X \geqslant x) = e^{-\lambda x}$. The mean of X is $1/\lambda$.
$\mathcal{N}(\mu,\ \sigma^2)$ is the normal distribution with mean μ and standard deviation σ. By definition if $X \sim \mathcal{N}(\mu,\ \sigma^2)$ the probability density function of X is $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$.

Part I

State of the art for decoding random binary linear codes

Chapter 1

Preliminaries in coding theory

Summary

In this short chapter we give a minimal set of definitions and essential properties related to codes and random codes that we will use profusely throughout this thesis. A more in depth coverage on error correcting codes can be found in [HP03] and a more targeted presentation regarding random codes and the decoding problem can be found in [Bar97] or the lecture notes [Deb23]. Our presentation is mostly technical, we first give in Section 1.1 the basic definitions related to codes, in Section 1.2 we give the basic operation one can make on them, in Section 1.3 we give a brief overview on random codes and their properties and finally in Section 1.4 we present quickly the decoding problem.

Contents

1.1 Essential definitions

The main object of study in this thesis will be binary linear codes.

Definition 4 (Binary linear code)**.** *We say that $\mathcal{C}$ is a binary linear code of length n and dimension k if it is a linear subspace of dimension k of $\mathbb{F}_2^n$. We say in short that $\mathcal{C}$ is an $[n,k]$-linear code and denote by $\mathfrak{C}[n,k]$ the set all of $[n,k]$-linear codes. We call $R \stackrel{def}{=} k/n$ the rate of the code. The elements of $\mathcal{C}$ are called codewords.*

In majority, the codes we handle in this thesis are binary linear codes. This is the case basically for all our code-based related chapters and only our last lattice related chapter will handle q-ary linear codes, that is, q is a prime power and $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^n$. To minimize the number of notations here we only describe here the binary case but all definitions and proposition trivially adapt to the q-ary casator or by a parity-che by replacing 2 by q in the following definitions and propositions.

Notation 1 (Notation valid in all this thesis)**.** *In this thesis, all codes handled, unless explicitly specified otherwise are binary linear codes. So when we say that "Let $\mathcal{C}$ be a code" we mean that $\mathcal{C}$ is a binary linear code.*

We give here the definition of the dual of a code. It will be central in this thesis.

Definition 5 (Dual of a linear code)**.** *The dual of binary linear code $\mathcal{C}$ is defined as*

$$\mathcal{C}^{\perp} = \{\mathbf{h} \in \mathcal{C} \ : \ \langle \mathbf{h}, \mathbf{c} \rangle = 0, \ \forall \mathbf{c} \in \mathcal{C}\}$$

where $\langle , \rangle$ is the standard dot product in $\mathbb{F}_2$, namely given two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{F}_2^n$ it is given by $\langle \mathbf{x}, \mathbf{y} \rangle \stackrel{def}{=} \sum_{i=1}^{n} x_i y_i \in \mathbb{F}_2$.

Clearly, we have the following relations between the code and its dual.

Fact 1 (Relations of a code with its dual)**.** *Let $\mathcal{C}$ be a linear code of dimension k and length n. We have that*

- *$\mathcal{C}^{\perp}$ is a linear code of dimension $n-k$ and length n.*
- *$\mathcal{C} = \left(\mathcal{C}^{\perp}\right)^{\perp}$.*

This motivates the following definition.

Definition 6 (Codimension of a code)**.** *The codimension of a code is the dimension of the dual of the code.*

Now, every code, seen as a subspace benefits from two efficient description, either by a generator or by a parity-check matrix.

Definition 7 (Generator matrix and parity-check matrix)**.** *Let $n, k \in \mathbb{N}$ with $k \leqslant n$ and let $\mathcal{C}$ be a binary linear code of length n. We say that $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ is a generator matrix of the code $\mathcal{C}$ if*

$$\mathcal{C} = \{\mathbf{mG} \ : \ \mathbf{m} \in \mathbb{F}_2^k\}.$$

We say that $\mathbf{H} \in \mathbb{F}_2^{(n-k) \times n}$ is a parity-check matrix of the code $\mathcal{C}$ if

$$\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_2^n \ : \ \mathbf{H}\mathbf{c}^{\intercal} = \mathbf{0}\}.$$

It is easy to go from one to the other: with basic linear algebra we can compute in time poly (n) a parity-check matrix from a generator matrix, and conversely and each is in fact linked in the following way.

Fact 2. *If* $\mathbf{G}$ *is a generator matrix of a linear code* $\mathcal{C}$ *then it is a parity-check matrix of the code* $\mathcal{C}^\perp$*. Conversely, if* $\mathbf{H}$ *is a parity-check matrix of a* $\mathcal{C}$ *then it is a generator matrix of* $\mathcal{C}^\perp$*.*

Definition 8 (Information set)**.** *Let* $\mathcal{C}$ *be a linear code of length* n *and dimension* k*. Let* $\mathscr{I} \subset [\![1,\, n]\!]$*. We say that* $\mathscr{I}$ *contains an information set of* $\mathcal{C}$ *if*

$$\dim\left(\{\mathbf{c}_{\mathscr{I}} \,:\, \mathbf{c} \in \mathcal{C}\}\right) = k.$$

If $|\mathscr{I}| = k$ *we say more precisely that* $\mathscr{I}$ *is an information set of* $\mathcal{C}$*.*

One can easily check that $\mathscr{I}$ is an information set of $\mathcal{C}$ by checking that $(\mathbf{G})_{\mathscr{I}}$ is of rank k or alternatively that $(\mathbf{H})_{\overline{\mathscr{I}}}$ is of rank $n-k$.

A code $\mathcal{C}$ naturally partition the space $\mathbb{F}_2^n$ into 2^{n-k} cosests.

Definition 9 (Coset)**.** *Let* $\mathcal{C}$ *be code of length* n *and* $\mathbf{y}$ *be a vector of* $\mathbb{F}_q^n$*. We say that*

$$\mathcal{C} + \mathbf{y} \stackrel{def}{=} \{\mathbf{c} + \mathbf{y} \,:\, \mathbf{c} \in \mathcal{C}\}$$

is a coset code of $\mathcal{C}$*.*

Each coset can be uniquely identified with its syndrome, as defined bellow.

Definition 10 (Syndrome)**.** *Fixing* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ *a parity-check matrix of a binary linear code* $\mathcal{C}$*, we say that a vector* $\mathbf{y} \in \mathbb{F}_2^n$ *has syndrome* $\mathbf{s} \in \mathbb{F}_2^{(n-k)\times 1}$ *if and only if*

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

Observe that by definition the codewords are exactly the vectors with null syndrome.

Last, we will often need in this thesis to count the number of codewords in a code or a coset code which are of a certain given Hamming weight, hence the following.

Definition 11 (Weight enumerator)**.** *Let* $\mathcal{S} \subset \mathbb{F}_2^n$*. We call the weight enumerator of* $\mathcal{S}$ *the function give associates for each* $t \in \mathbb{N}$ *the number of elements of* $\mathcal{S}$ *of Hamming weight* t*. Namely, we define*

$$N_t(\mathcal{S}) \stackrel{def}{=} \left|\mathcal{S} \bigcap \mathcal{S}_t^n\right|.$$

1.2 Algebraic operations on linear codes

We present here how the basic operations and construction one can make on linear codes. As a vast majority of the algorithms we devise in this thesis rely on these operations we also quickly devise how this is done algorithmically so that our algorithms are non-ambiguous.

1.2.1 Puncturing

We will often consider here the punctured of a code, namely the code where we only keep some positions.

Definition 12 (Puncturing a code)**.** *Let $\mathcal{C}$ be a code of length n and let $\mathscr{I} \subset [\![1, n]\!]$. We define $\mathcal{C}_{\mathscr{I}}$, the code $\mathcal{C}$ punctured on $\mathscr{I}$ as*

$$\mathcal{C}_{\mathscr{I}} \stackrel{def}{=} \{\mathbf{c}_{\mathscr{I}} \ : \ \mathbf{c} \in \mathcal{C}\}.$$

As long as we do not puncture too much, this operation leaves the dimension untouched.

Fact 3. *Let $\mathcal{C}$ be a linear code of length n and dimension k. $\mathscr{I}$ contains an information set of $\mathcal{C}$ if and only if $\mathcal{C}_{\mathscr{I}}$ is a linear code of length $|\mathscr{I}|$ and dimension k.*

1.2.1.1 Algorithm

In this thesis, puncturing $\mathcal{C}$ on $\mathscr{I}$ will only be performed when the code $\mathcal{C}$ is given as a parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ and when furthermore $\mathscr{I}$ contains an information set of $\mathcal{C}$. This is done in polynomial time by computing a partial Gaussian pivot on the columns of $\mathbf{H}$ given by $\overline{\mathscr{I}}$. Namely, we compute an invertible pivot matrix $\mathbf{J} \in \mathbb{F}_2^{(n-k)\times(n-k)}$ that is such that $(\mathbf{JH})_{\overline{\mathscr{I}}} = \begin{pmatrix} \mathbf{I}_{|\overline{\mathscr{I}}|} \\ \mathbf{0} \end{pmatrix}$ and output a data structure containing the parity-check matrix of the code $\mathcal{C}_{\mathscr{I}}$ composed of the last $n - k - \left|\overline{\mathscr{I}}\right|$ lines of $(\mathbf{JH})_{\mathscr{I}}$.

1.2.2 Shortening

One can also build another code by considering the code which contains all the codewords that are zero on some coordinates (which we then forget).

Definition 13 (Shortening a code)**.** *Let $\mathcal{C}$ be a code of length n. We define $\mathcal{C}^{\mathscr{I}}$, the code $\mathcal{C}$ shortened on $\mathscr{N}$ as*

$$\mathcal{C}^{\mathscr{N}} \stackrel{def}{=} \{\mathbf{c}_{\mathscr{N}} \ : \ \mathbf{c} \in \mathcal{C} \text{ and } \mathbf{c}_{\overline{\mathscr{N}}} = \mathbf{0}\}.$$

In fact, shortening a code can be seen as puncturing the dual of the code.

Lemma 1. *(Relation between shortening and puncturing [HP03, Theorem 1.5.7]) Let $\mathcal{C}$ be a linear code of length n and let $\mathscr{N} \subset [\![1, n]\!]$ be a set. We have that*

$$\mathcal{C}^{\mathscr{N}} = \left(\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\right)^{\perp}$$

Using this relation along with Fact 3 it is clear that essentially the dimension of the shortened code decreases by the number of deleted coordinates.

Fact 4. *Let $\mathcal{C}$ be a linear code of length n and dimension k and $\mathscr{N} \subset [\![1, n]\!]$ be a set. Then $\mathcal{C}^{\mathscr{N}}$ is a linear code of length $|\mathscr{I}|$ and dimension $k - \left|\overline{\mathscr{I}}\right|$ if and only if $\mathscr{N}$ contains an information set of $\mathcal{C}^{\perp}$.*

1.2.2.1 Algorithm

Algorithmically in this thesis, shortening $\mathcal{C}$ on $\mathscr{N}$ will only ever be performed when the code $\mathcal{C}$ by a generator matrix $\mathbf{G}$ and when $\mathscr{N}$ contains an information set of $\mathcal{C}^\perp$. This is done in polynomial time by computing a partial Gaussian pivot on the columns of $\mathbf{G}$ given by $\overline{\mathscr{N}}$. Namely, we compute an invertible pivot matrix $\mathbf{J} \in \mathbb{F}_2^{k\times k}$ that is such that $(\mathbf{JG})_{\overline{\mathscr{N}}} = \begin{pmatrix} \mathbf{I}_{|\overline{\mathscr{N}}|} \\ \mathbf{0} \end{pmatrix}$ and output a data structure containing the generator matrix of the code $\mathcal{C}^{\mathscr{N}}$ composed of the last $k - \left|\overline{\mathscr{N}}\right|$ lines of $(\mathbf{JG})_{\mathscr{N}}$.

1.2.3 Lifting onto a code

We will profusely use the fact that knowing the value of a codeword on an information set of the code, by definition, uniquely identifies that codeword. More, we can go from one to the other with a lifting operation that is a linear and that we name in the next definition.

Definition 14 (Lifting an element)**.** *Let $\mathcal{C}$ be a linear code of length n and let $\mathscr{I} \subset [\![1, n]\!]$ be a set containing an information set of $\mathcal{C}$. We define* $\mathrm{Lift}\,(\mathcal{C},\ \mathscr{I},\ \mathbf{v})$ *the linear function that for each $\mathbf{v} \in \mathcal{C}_{\mathscr{I}}$ associates*

$$\mathrm{Lift}\,(\mathcal{C},\ \mathscr{I},\ \mathbf{v}) \stackrel{def}{=} \mathbf{c} \tag{1.1}$$

where $\mathbf{c}$ is the unique codeword of $\mathcal{C}$ which is such that $\mathbf{c}_{\mathscr{I}} = \mathbf{v}$. We call $\mathbf{c}$ the lift of $\mathbf{v}$. Last, we define

$\mathrm{Lift}\,(\mathcal{C},\ \mathscr{I}) \stackrel{def}{=} \mathbf{R}$ *where $\mathbf{R} \in \mathbb{F}_2^{(n-|\mathscr{I}|)\times|\mathscr{I}|}$ is the unique matrix such that $\forall \mathbf{c} \in \mathcal{C},\ \mathbf{c}_{\overline{\mathscr{I}}} = \mathbf{c}_{\mathscr{I}}\mathbf{R}^\intercal$.*

1.2.3.1 Algorithm

Algorithmically in this thesis, this lifting operation will only ever be performed on codes $\mathcal{C}$ which are given as a $\mathbb{F}_2^{(n-k)\times n}$ parity-check matrix $\mathbf{H}$. This is done in polynomial time by computing a partial Gaussian pivot on the columns of $\mathbf{H}$ given by $\overline{\mathscr{I}}$. That is we compute an invertible matrix $\mathbf{J} \in \mathbb{F}_2^{k\times k}$ whose entries only depends on $\mathbf{H}_{\overline{\mathscr{I}}}$ and which is such that $(\mathbf{JH})_{\overline{\mathscr{I}}} = \begin{pmatrix} \mathbf{I}_{|\overline{\mathscr{I}}|} \\ \mathbf{0} \end{pmatrix}$. In that case the matrix $\mathrm{Lift}\,(\mathcal{C},\ \mathscr{I})$ is given by the first $\left|\overline{\mathscr{I}}\right|$ lines of $(\mathbf{JH})_{\mathscr{I}}$.

1.3 Random binary linear codes

We give here some basic properties on random binary linear codes.

1.3.1 Randomness model

Let us give precise meaning to what we will mean by random codes in this thesis, we consider the 3 following standard definitions.

Definition 15. *We define the following distributions:*

1. *We name* $\mathcal{U}_{\mathbf{H}}(n,\ k)$ *the distribution on the linear codes of length* n *obtained by choosing its parity-check matrix* $\mathbf{H}$ *uniformly at random in* $\mathbb{F}_2^{(n-k)\times n}$*. When* $\mathcal{C} \sim \mathcal{U}_{\mathbf{H}}(n,\ k)$ *we denote by* $\mathbf{H}(\mathcal{C})$ *the underlying parity-check matrix* $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$*.*
2. *We name* $\mathcal{U}_{\mathbf{G}}(n,\ k)$ *the distribution on the linear codes of length* n *obtained by choosing its generator matrix* $\mathbf{G}$ *uniformly at random in* $\mathbb{F}_2^{k\times n}$*. When* $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\ k)$ *we denote by* $\mathbf{G}(\mathcal{C})$ *the underlying generator matrix* $\mathbf{G} \in \mathbb{F}_2^{k\times n}$*.*
3. *We name* $\mathcal{U}(n,\ k)$ *the uniform distribution on the linear codes of length* n *and dimension* k*, that is* $\mathcal{U}(\mathfrak{C}[n,k])$*. It is obtained by choosing its parity-check matrix* $\mathbf{H}$ *uniformly at random among the matrices of* $\mathbb{F}_2^{(n-k)\times n}$ *of rank* $n-k$*.*

Because a generator matrix of the code is the parity-check matrix of the dual one can interchange these distributions in the following way.

Fact 5. *Let* $n, k \in \mathbb{N}$ *with* $k \leqslant n$*. We have that*

$$\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\ k) \Leftrightarrow \mathcal{C}^{\perp} \sim \mathcal{U}_{\mathbf{H}}(n,\ n-k), \qquad \text{and,} \qquad \mathcal{C} \sim \mathcal{U}(n,\ k) \Leftrightarrow \mathcal{C}^{\perp} \sim \mathcal{U}(n,\ k).$$

It is easy to see that those distributions are closely related, up to rank default the distributions are the same. Because a $k \times n$ random binary matrix has rank default with probability $\mathcal{O}\left(2^{-(n-k)}\right)$ we directly have that the statistical distance between those distributions is close.

Proposition 4 (Statistical distance between the distributions)**.** *The statistical distance between* $\mathcal{U}_{\mathbf{H}}(n,\ k)$ *and* $\mathcal{U}(n,\ k)$ *is*

$$\Delta\left(\mathcal{U}_{\mathbf{H}}(n,\ k),\ \mathcal{U}(n,\ k)\right) = \mathcal{O}\left(2^{-k}\right)$$

Using the same argument, this allows to see basically that any size bigger than k is an information set of the code with overwhelming probability.

Fact 6 (Probability of being an information set)**.** *Let* $\mathscr{I}$ *be a fixed subset of* $[\![1,\ n]\!]$*. For any of the random models considered in Definition 15, if* $|\mathscr{I}| = k + \omega(1)$ *then* $\mathscr{I}$ *is an information set of* $\mathcal{C}$ *with probability* $1 - o(1)$*, if* $|\mathscr{I}| = k$ *then* $\mathscr{I}$ *is an information set with probability* $\Omega(1)$*.*

One easily see, for example by looking at the algorithm given in Section 1.2, how those distributions interacts with lifting, puncturing and shortening.

Fact 7 (Distributions related to lifting, puncturing and shortening.)**.** *If* $\mathcal{C} \sim \mathcal{U}_{\mathbf{H}}(n,\ k)$ *and* $\mathscr{I}$ *is a fixed subset of* $[\![1,\ n]\!]$ *then conditioned on the event that* $\mathscr{I}$ *contains an information set of* $\mathcal{C}$ *we have that*

- $\mathcal{C}_{\mathscr{I}} \sim \mathcal{U}_{\mathbf{H}}(|\mathscr{I}|,\ k)$*.*
- $\mathrm{Lift}(\mathcal{C},\ \mathscr{I}) \sim \mathcal{U}\left(\mathbb{F}_2^{(|\overline{\mathscr{I}}|)\times|\mathscr{I}|}\right)$

If $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\ k)$ *and* $\mathscr{N}$ *is a fixed subset of* $[\![1,\ n]\!]$ *then conditionned on the event that* $\mathscr{N}$ *contains an information set of* $\mathcal{C}^{\perp}$ *we have that*

- $\mathcal{C}^{\mathscr{N}} \sim \mathcal{U}_{\mathbf{G}}\left(|\mathscr{N}|,\ k - \overline{|\mathscr{N}|}\right)$*.*
- $\mathrm{Lift}\left(\mathcal{C}^{\perp},\ \mathscr{N}\right) \sim \mathcal{U}\left(\mathbb{F}_2^{(|\overline{\mathscr{N}}|)\times|\mathscr{N}|}\right)$

1.3.2 First and second order statistics

We will very often use the following standard probabilities that some fixed element belongs to a random code.

Proposition 5. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n \setminus \{0\}$ be fixed vectors such that $\mathbf{x} \neq \mathbf{y}$. Let $\mathcal{C} \sim \mathcal{U}_{\mathbf{H}}(n,\ k)$ we have that*

$$\mathbb{P}(\mathbf{x} \in \mathcal{C}) = \frac{1}{2^{n-k}}, \qquad \mathbb{P}(\mathbf{x} \in \mathcal{C},\ \mathbf{y} \in \mathcal{C}) = \left(\frac{1}{2^{n-k}}\right)^2, \tag{1.2}$$

Proof. Let us denote by $\mathbf{H} = \mathbf{H}(\mathcal{C})$ the parity-check matrix used to construct the code $\mathcal{C}$. We have that $\mathbf{x} \in \mathcal{C}$ if and only if $\mathbf{H}\mathbf{x}^\intercal = \mathbf{0}$, thus

$$\begin{aligned}\mathbb{P}(\mathbf{x} \in \mathcal{C}) &= \mathbb{P}(\mathbf{H}\mathbf{x}^\intercal = \mathbf{0}) \\ &= \frac{1}{2^{n-k}}\end{aligned}$$

where in the last line we used the fact that $\mathbf{H} \sim \mathcal{U}\left(\mathbb{F}_2^{(n-k)\times n}\right)$ and that $\mathbf{x}^\intercal$ is a fixed non-null vector of $\mathbb{F}_2^n$. Now the second equality comes from the fact that

$$\begin{aligned}\mathbb{P}(\mathbf{x} \in \mathcal{C},\ \mathbf{y} \in \mathcal{C}) &= \mathbb{P}(\mathbf{y} \in \mathcal{C} \mid \mathbf{x} \in \mathcal{C})\, \mathbb{P}(\mathbf{x} \in \mathcal{C}) \\ &= \mathbb{P}(\mathbf{H}\mathbf{y}^\intercal = \mathbf{0} \mid \mathbf{H}\mathbf{x}^\intercal = \mathbf{0})\, \frac{1}{2^{n-k}} \\ &= \left(\frac{1}{2^{n-k}}\right)^2\end{aligned}$$

where in the last line we used the fact that $\mathbf{H}\mathbf{x}^\intercal$ and $\mathbf{H}\mathbf{y}^\intercal$ are independent variables. Indeed as $\mathbf{x}$ and $\mathbf{x}$ are fixed vectors such that $\mathbf{y} \neq \mathbf{x}$ and $\mathbf{y} \neq \mathbf{0}$, there exists, possibly interchanging the role of $\mathbf{x}$ and $\mathbf{y}$, at least one position i such that $y_i = 1$ and $x_i = 0$ which gives that

$$\begin{aligned}\mathbf{H}\mathbf{y}^\intercal &= \mathbf{H}_i + \mathbf{H}_{\setminus i}\left(\mathbf{y}_{\setminus i}\right)^\intercal, \\ \mathbf{H}\mathbf{x}^\intercal &= \mathbf{H}_{\setminus i}\left(\mathbf{x}_{\setminus i}\right)^\intercal.\end{aligned}$$

Each are independent since all the columns of $\mathbf{H}$ are taken independently. □

1.3.3 Weight enumerator and its second-order concentration

The previous proposition directly yields that the covariance of $\mathbf{1}_{\mathbf{x}\in\mathcal{C}}$ and $\mathbf{1}_{\mathbf{y}\in\mathcal{C}}$, this allows to easily compute the first two moments of say the weight enumerator of the code for example.

Lemma 2 (First two moment of the weight enumerator)**.** *Let n, k, t with $t \neq 0$ and let $\mathbf{y} \in \mathbb{F}_2^n$ and let $\mathcal{C} \sim \mathcal{U}_{\mathbf{H}}(n,\ k)$. We have that*

$$\begin{aligned}\mathbb{E}\left(N_t(\mathcal{C} + \mathbf{y})\right) &= \frac{\binom{n}{t}}{2^{n-k}} \\ \mathbf{Var}\left(N_t(\mathcal{C} + \mathbf{y})\right) &\leqslant \frac{\binom{n}{t}}{2^{n-k}}\end{aligned}$$

Proof. This is a direct consequence of Proposition 5. Starting from the fact that

$$N_t\left(\mathcal{C}+\mathbf{y}\right)=\sum_{\mathbf{x}\in\mathcal{S}_t^n}\mathbf{1}_{\mathbf{x}\in\mathcal{C}+\mathbf{y}}$$

and using the linearity of the expectation we have that

$$\begin{aligned}\mathbb{E}\left(N_t\left(\mathcal{C}+\mathbf{y}\right)\right)&=\sum_{\mathbf{x}\in\mathcal{S}_t^n}\mathbb{P}\left(\mathbf{x}\in\mathcal{C}+\mathbf{y}\right)\\&=\frac{\binom{n}{t}}{2^{n-k}}\qquad(\textit{Proposition 5}).\end{aligned}$$

The variance is obtained by

$$\begin{aligned}\mathbf{Var}\left(N_t\left(\mathcal{C}+\mathbf{y}\right)\right)&=\sum_{\mathbf{x}\in\mathcal{S}_t^n}\mathbf{Var}\left(\mathbf{1}_{\mathbf{x}\in\mathcal{C}+\mathbf{y}}\right)+\sum_{\mathbf{x},\,\mathbf{z}\in\mathcal{S}_t^n\,:\,\mathbf{x}\neq\mathbf{z}}\mathbf{Cov}\left(\mathbf{1}_{\mathbf{x}\in\mathcal{C}+\mathbf{y}},\,\mathbf{1}_{\mathbf{z}\in\mathcal{C}+\mathbf{y}}\right)\\&\leqslant\sum_{\mathbf{x}\in\mathcal{S}_t^n}\mathbb{P}\left(\mathbf{x}\in\mathcal{C}+\mathbf{y}\right)+\sum_{\mathbf{x},\,\mathbf{z}\in\mathcal{S}_t^n\,:\,\mathbf{x}\neq\mathbf{z}}\mathbb{P}\left(\mathbf{x}\in\mathcal{C}+\mathbf{y},\,\mathbf{z}\in\mathcal{C}+\mathbf{y}\right)\\&=\frac{\binom{n}{t}}{2^{n-k}}+0\qquad(\textit{Proposition 5}).\end{aligned}$$

□

In turn, using Byenemé-Chebychev inequality on these moments allows devising second-order bounds on the weight enumerator.

Proposition 6 (Concentration of the weight enumerator)**.** *Let $n,k,t\in\mathbb{N}$ and let f be a positive function. Let $\mathbf{y}\in\mathbb{F}_2^n$ and let $\mathcal{C}\sim\mathcal{U}_{\mathbf{H}}\left(n,\,k\right)$. We have that,*

$$\mathbb{P}\left(\left|N_t\left(\mathcal{C}+\mathbf{y}\right)-\frac{\binom{n}{t}}{2^{n-k}}\right|\geqslant f(n)\sqrt{\frac{\binom{n}{t}}{2^{n-k}}}\right)\leqslant\frac{1}{f(n)}.$$

1.3.4 Gilbert-Varshamov distance

The so called Gilbert-Varshamov is the expected minimum Hamming distance of the code, or, in the case of linear codes, the expected Hamming weight of the minimum non-zero weight codeword. It is readily seen, from Lemma 2 that the following definition captures this interpretation.

Definition 16 (Gilbert-Varshamov distance)**.** *Let $n,k\in\mathbb{N}$ be such that $k\leqslant n$. We define the Gilbert-Varshamov distance $d_{\mathrm{GV}}\left(n,\,k\right)$ as the largest integer such that*

$$\sum_{i=0}^{d_{\mathrm{GV}}(n,\,k)}\binom{n}{i}\leqslant 2^{n-k}.$$

One can capture the asymptotic behavior of this quantity using the standard binomial expansion given as follows.

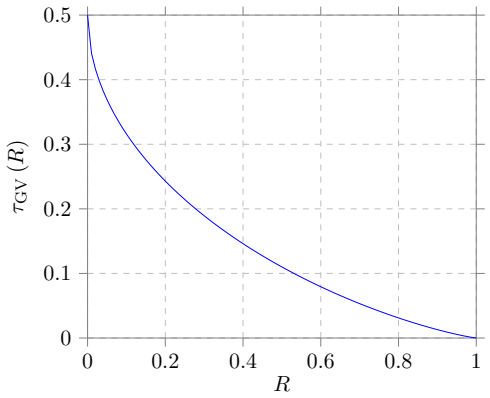


Figure 1.1: Relative Gilbert-Varshmov distance $\tau_{\mathrm{GV}}(R) \stackrel{\mathrm{def}}{=} h^{-1}(1-R)$

Proposition 7 (Binomial expansion)**.** *Let $n, k \in \mathbb{N}$ with $0 < k < n$. We have that*

$$\binom{n}{k} = \tilde{\Theta}\left(2^{n\, h(k/n)}\right). \tag{1.3}$$

We denote, with a slight abuse in the definition, the relative Gilbert-Varshmov distance as follows.

Definition 17 (Relative Gilbert-Varshmov distance)**.** *Let $R \in [0,\ 1]$. We define the relative Gilbert-Varshamov distance at rate R, namely $\tau_{\mathrm{GV}}(R)$ as*

$$\tau_{\mathrm{GV}}(R) \stackrel{def}{=} h^{-1}(1-R)$$

where $h^{-1}()$ is the inverse on $[0,\ 1/2]$ of the binary entropy function $h(x) \stackrel{def}{=} -x\log_2(x) - (1-x)\log_2(1-x)$.

This function is plotted in Fig. 1.1. In particular, by using Proposition 7 and forgetting about the polynomial factors, we can see that $d_{\mathrm{GV}}(n,\ k)$ and $\tau_{\mathrm{GV}}(k/n)\, n$ can be interchanged at will.

Lemma 3. *Let $R \in]0,\ 1[$ be an implicit function of $n \in \mathbb{N}$ and define $k \stackrel{def}{=} \lfloor Rn \rfloor$ and $t \stackrel{def}{=} \lfloor \tau_{\mathrm{GV}}(R)\, n \rfloor$. We have that*

$$\binom{n}{t} = \tilde{\Theta}\left(\binom{n}{d_{\mathrm{GV}}(n,\ k)}\right) = \tilde{\Theta}\left(2^{n-k}\right)$$

and that

$$\frac{d_{\mathrm{GV}}(n,\ k)}{n} = \tau_{\mathrm{GV}}(R) + o(1).$$

1.4 The decoding problem

One of the main part of this thesis will be dedicated finding new algorithm to decode linear code, let us give one of the possible definition of an instance of this problem.

Definition 18 (Decoding problem)**.** *We define the decoding problem with parameter (n, k, t) as the search problem given as follows.*

- *Input:* $(\mathcal{C},\ \mathbf{y})$ *where* $\mathcal{C}$ *is a code given as a generator matrix* $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ *of rank* k *and* $\mathbf{y} \in \mathbb{F}_2^n$*.*
- *Output:* $\mathbf{e} \in \mathcal{S}_t^n$ *such that* $\mathbf{y} - \mathbf{e} \in \mathcal{C}$

The associated decision problem is NP-Complete [BMvT78]. Clearly, and to make a parallel with our introduction, solving this problem is really just finding an error $\mathbf{e}$ of Hamming weight t whose syndrome match that of $\mathbf{y}$. Indeed, we can compute in polynomial time a parity-check matrix $\mathbf{H}$ from the generator matrix $\mathbf{G}$ and use the following fact.

Fact 8. *Let* $\mathcal{C}$ *be a linear code and* $\mathbf{H}$ *be a parity-check matrix of* $\mathcal{C}$*, and* $\mathbf{y}$, $\mathbf{e}$ *two vectors of* $\mathbb{F}_2^n$*. We have that*

$$\mathbf{y} - \mathbf{e} \in \mathcal{C} \Leftrightarrow \mathbf{H}\mathbf{e}^\intercal = \mathbf{H}\mathbf{y}^\intercal.$$

To say it differently, this is really solving an underdetermined linear system with a constraint on the Hamming weight. This constraint makes the problem hard.

Note that, contrary to the telecommunication variant of this problem where we want to recover the closest codeword to a vector, in code-based cryptography, usually breaking a scheme only requires to produce any solution that is close enough, hence the definition. In fact, the security of the schemes never rely on the hardness of the worst-case instances but rather rely on the difficulty of average instances. Many distributions can be considered but in this thesis we study the most standard and historic average variant of the problem given as follows.

Definition 19 (Average decoding problem $\mathrm{DP}_{\mathbf{H}}(n, k, t)$ and $\mathrm{DP}_{\mathbf{G}}(n, k, t)$.)**.** *For any* $n, k, t \in \mathbb{N}$ *such that* $k,\ t \leqslant n$ *the average decoding problem at distance* t*,* $\mathrm{DP}_{\mathbf{H}}(n, k, t)$ *is defined as: we are given* $(\mathcal{C},\ \mathbf{y})$ *which are chosen as follows:*

- *The code* $\mathcal{C}$ *is given as its parity-check matrix* $\mathbf{H}$ *that was chosen uniformly at random in* $\mathbb{F}_2^{(n-k)\times n}$*, namely* $\mathcal{C} \sim \mathcal{U}_{\mathbf{H}}(n,\ k)$*.*
- *The noisy codeword* $\mathbf{y} = \mathbf{c} + \mathbf{e}$ *is taken by choosing* $\mathbf{c} \sim \mathcal{U}(\mathcal{C})$ *and* $\mathbf{e} \sim \mathcal{U}(\mathcal{S}_t^n)$*.*

We must return a vector $\mathbf{e} \in \mathcal{S}_t^n$ *such that* $\mathbf{y} - \mathbf{e} \in \mathcal{C}$*. We also define the average decoding problem* $\mathrm{DP}_{\mathbf{G}}(n, k, t)$ *as above but in this case* $\mathcal{C}$ *is taken by choosing its generator matrix* $\mathbf{G}$ *uniformly at random in* $\mathbb{F}_2^{k\times n}$*, namely* $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\ k)$*.*

Remark 3. *Note that both problems* $\mathrm{DP}_{\mathbf{G}}(n, k, t)$ *and* $\mathrm{DP}_{\mathbf{H}}(n, k, t)$ *are closely related as, using Proposition 4 we can conclude that any algorithm solving one problem can be turned into an algorithm solving the other with only a polynomial overhead to construct a generator matrix from a parity-check matrix (and vice-versa) and a negligible loss in probability of success due to the negligible statistical distance between the two input distributions.*

For the right choices of parameter this problem is conjectured to be hard on average, that means that for some right choices for n, k, t, all the existing algorithm solving this problem with non-negligible probability run in time $2^{\Omega(t)}$. The complexity of the best decoders, that is, algorithms solving this problem with non-negligible probability, are used to choose the parameters of code-based encryption scheme such as HQC [AAB+22b], which is now becoming

a NIST standard, or BIKE [AAB+22a] or the signature scheme SDitH [AMBB+24] that is currently in the second round of the NIST additional signature scheme competition. Notably the security of this last scheme rely on the difficulty on the decoding problem at the aforementioned Gilbert-Varshamov distance. It can readily be seen from the definition that this is the distance where we expect essentially (up to polynomial factors) that there exists one non-planted solution to the decoding problem.

Fact 9. *Let n, k and $t \stackrel{def}{=} d_{\mathrm{GV}}(n,\ k)$. Let $\mathcal{C} \sim \mathcal{U}_{\mathbf{H}}(n,\ k)$ and let $\mathbf{y} \in \mathbb{F}_2^n$. We have that*

$$\mathbb{E}\left(|\{\mathbf{e} \in \mathcal{S}_t^n \ : \ \mathbf{y} - \mathbf{e} \in \mathcal{C}\}|\right) = \tilde{\Theta}(1).$$

As we will see later, this distance is where the decoding problem is the hardest. The rough intuition for that is that because we are looking for any solution and not one in particular, when the distance grows further than the Gilbert-Varshamov distance the number of solutions explodes, making the problem easier. It is traditional to compare the algorithm at this hardest distance as a first step to compare the algorithms between each others, this will mostly be our regime of interest in this thesis. However, we will also give some more details about the hardness of the problem in other parameter regime when presenting decoders in the next chapter.

Chapter 2

State of the art of generic decoders

Summary

We present here the state of the art of generic decoders. We focus our presentation on the algorithms tailored to classically decode random binary linear codes of constant rates and at a distance that grows linearly with the codelength but which is always less than or equal to the Gilbert-Varshamov distance. We also focus our presentation on the best time complexity exponent rather than looking at time-memory tradeoffs. This chapter is composed of two sections. The first is dedicated to exposing some Information Set Decoders or in short ISD. These have been for 60 years the dominant and most efficient family of decoders in our regime of interest. We will use some of them later as a subroutine for the dual attacks we will devise in our contribution chapters. As a contribution, we found an error in the original analysis of the state of the art of these ISD, namely [BM18]. We show that this leads to a very significant underestimation of the time complexity of the algorithm. We make a corrected analysis and give as such the new baseline for the complexity of the state of the art of ISD's. The second section of this chapter is dedicated to exposing the first dual attack in coding theory: statistical decoding, it was shown to be completely uncompetitive compared to the ISD's. Overall during these 60 years of research these decoders have improved in the high to moderate rate regime but no real progress has been made in the low-rate regime.

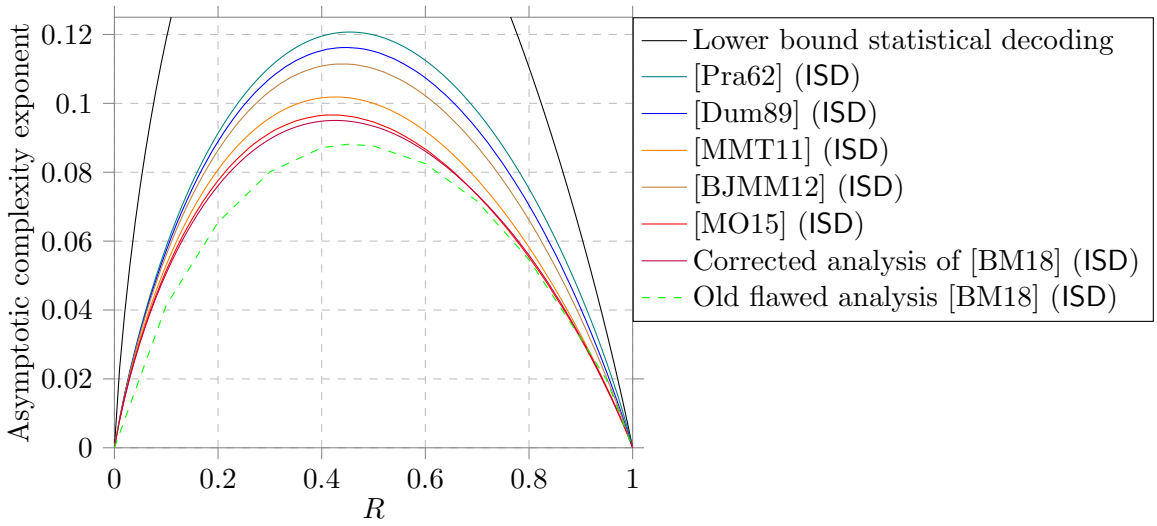


Figure 2.1: The asymptotic complexity exponent relative to the codelength n (growing to infinity) of some decoders to decode random linear codes of rate $R = k/n$ at the Gilbert-Varshamov distance. More precisely we plotted the α's such that the time complexity of each of those algorithms is some $2^{\alpha(1+o(1))n}$ to solve the average decoding problem $\mathrm{DP}_{\mathbf{H}}(n, k, t)$ (defined in Definition 19) when $k \stackrel{\text{def}}{=} \lfloor Rn \rfloor$ and $t \stackrel{\text{def}}{=} \lfloor \tau_{\mathrm{GV}}(R)\, n \rfloor$. The complexity claim regarding [BM18] was obtained with Theorem 2. The black curve is the lower bound given by [DT17a] of statistical decoding of [Jab01].

Contents

2.1 Information Set Decoders

Starting from Prange [Pra62], information set decoders have been for 60 years the leading family of decoders to decode codes of constant rate. The name comes from the fact that all these decoders leverage that knowing the value of the secret codeword on an information set of the code allows recovering the whole codeword, and hence decode. Say we are given a linear code $\mathcal{C}$ of dimension k and length n and some noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$, where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e}$ is an error vector of low Hamming weight, say t, and that we want to decode this noisy codeword. In a nutshell Prange idea is to choose k coordinates of the support $[\![1,\, n]\!]$, this indeed forms an information set of $\mathcal{C}$ with good probability, and bet that, on those k coordinates the error vector $\mathbf{e}$ is zero. This is a meaningful bet because we know that the error is of low Hamming weight. If the bet was good, it can decode with basic linear algebra. Basically Prange iterates this, choosing each time k random positions, until eventually this bet is valid and the solution can be recovered. Still, in our regime of interest, this requires an exponential number of iteration, this represents the source of the hardness of this algorithm. This algorithm was widely improved over the years, to name a few [LB88, Ste88, Dum89, MMT11, BJMM12, MO15, BM17, BM18, Ess23, GJN24, DEEK24], the state of the art is given by [BM18]. All those improved algorithms share with the Prange algorithm the fact that they make a bet on the value of the error on a few number of positions, k or more, but the improvement comes by relaxing the bet and allowing a few number of position of the error to be non-zero on the selected positions. The goal is then to try to determine those erroneous positions in the most efficiently way. This is done typically with some collision or near-collision based techniques.

Note, as we will see later, that all these information set decoders can be turned into a procedure to compute low weight vectors of the code, or more tailored to our purpose in this thesis, low weight dual vectors. In fact, all dual attacks we devise will need procedures which compute dual vectors somewhat efficiently. We chose to analyze our attacks when the building block (which we will later call subroutines) of [Dum89, BJMM12] are used to compute the low weight dual vectors. This choice is slightly arbitrary because we could also have used more efficient routines such as [MO15, BM17, BM18, DEEK24] but, as we will see the subroutines of [Dum89, BJMM12] very naturally can be turned to compute all the dual vectors of a certain weight, this simplifies the analysis of our dual attacks.

We present in detail these two algorithms. We also present the state of the art [BM18] because we found a flaw in its analysis which leads, as we show, to a significant underestimation of the complexity of the algorithm. We correct it and compute its new asymptotic complexity exponent, yielding the baseline for the complexity of the state of the art of the ISD's.

In the body of the text of the coming subsections we use the following notation.

Notation 2. *We consider $\mathcal{C}$ a binary linear code of length n and $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ a parity-check matrix of $\mathcal{C}$. We want to decode at distance t the noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e} \in \mathcal{S}_t^n$ is an error vector of low weight $t \leqslant n/2$. When probabilities are involved, one can see $\mathcal{C}$, $\mathbf{y}$ as an instance of $\mathrm{DP}_{\mathbf{H}}(n, k, t)$, that is $\mathcal{C} \sim \mathcal{U}_{\mathbf{H}}(n,\, k)$ is chosen by taking $\mathbf{H}$ uniformly at random in $\mathbb{F}_2^{(n-k)\times n}$ and $\mathbf{e} \sim \mathcal{U}(\mathcal{S}_t^n)$. When asymptotic notations are used, k, t are implicit function of n where n is a parameter growing to infinity.*

2.1.1 Prange decoder

The idea behind the Prange decoder [Pra62] which is the first information set decoder is that having access to a set subset of the support $\mathscr{N} \subset [\![1,\, n]\!]$ which does not contain any errors, that is which is such $\mathbf{e}_{\mathscr{N}} = \mathbf{0}$, and contains an information set of the code allows to easily solving the decoding problem. Indeed, because $\mathscr{N}$ is error free we have that $\mathbf{y}_{\mathscr{N}} = \mathbf{c}_{\mathscr{N}}$ and because $\mathscr{N}$ contains an information set we can simply lift $\mathbf{y}_{\mathscr{N}}$ onto $\mathcal{C}$ to recover the whole codeword $\mathbf{c}$. The idea is then simply to iterate, choosing each time a different subset $\mathscr{N}$ until one verifies this. It is easy to see that, up to polynomial factors, it is optimal to choose $\mathscr{N}$ of size exactly k, the dimension of the code $\mathcal{C}$.

Algorithm 1 Prange decoder

Name: ISD-PRANGE($\mathcal{C}$, $\mathbf{y}$, t)
Input: $\mathcal{C}$, $\mathbf{y}$, t

1: **for** $i = 1 \cdots N_{\text{iter}}$ **do** ▷ N_{iter} *is exponential*
2: $\quad \mathscr{N} \xleftarrow{\$} \{\mathscr{I} \subset [\![1,\, n]\!] \,:\, |\mathscr{I}| = k\}$
3: $\quad \widetilde{\mathbf{c}} \leftarrow \text{Lift}\left(\mathcal{C},\, \mathscr{N},\, \mathbf{y}_{\mathscr{N}}\right)$ ▷ *If $\mathscr{N}$ is not an information set of $\mathcal{C}$ then this procedure fails making this iteration end earlier*
4: $\quad \widetilde{\mathbf{e}} \leftarrow \mathbf{y} - \widetilde{\mathbf{c}}$
5: $\quad$ **if** $|\widetilde{\mathbf{e}}| = t$ **then**
6: $\quad\quad$ **return** $\widetilde{\mathbf{e}}$

The time complexity is, up to polynomial factors, the number of iterations we need to make

Proposition 8. *Let $n, k, t \in \mathbb{N}$. There exists N_{iter} such that Algorithm 1 finds the original error $\mathbf{e}^*$ when given an instance of $\text{DP}_{\mathbf{H}}(n, k, t)$ with probability $\Omega(1)$ in time and memory*

$$\textbf{\textit{Time}} = \text{poly}(n)\, \frac{\binom{n}{t}}{\binom{n-k}{t}}, \qquad \textbf{\textit{Memory}} = \text{poly}(n).$$

Corollary 2. *Let n be growing to infinity and let R, τ be implicit functions of n. Let $k = \lfloor Rn \rfloor$ and $t = \lfloor \tau n \rfloor$. There exists N_{iter} such that Algorithm 1 finds the original error $\mathbf{e}^*$ when given an instance of $\text{DP}_{\mathbf{H}}(n, k, t)$ with probability $\Omega(1)$ in time and memory*

$$\textbf{\textit{Time}} = \widetilde{\mathcal{O}}\left(2^{\alpha_{\text{Prange}}(R,\, \tau)\, n}\right), \qquad \textbf{\textit{Memory}} = \widetilde{\mathcal{O}}(1)$$

where

$$\alpha_{\text{Prange}}(R,\, \tau) \overset{def}{=} h(\tau) - (1 - R)\, h\left(\tau / (1 - R)\right).$$

Note that we could also show more generally that the Prange decoder solves $\text{DP}_{\mathbf{G}}(n, k, t)$ (i.e. returns a solution to the decoding problem and not specifically $\mathbf{e}^*$) with probability $\Omega(1)$ and in time

$$\text{poly}(n)\, \frac{\binom{n}{t}}{\binom{n-k}{t}} \frac{1}{\max\left(1,\, \binom{n}{t}/2^{n-k}\right)} \tag{2.1}$$

which is essentially the inverse of the probability of success that we find a given solution to the decoding problem multiplied by the average number of solutions $\binom{t}{n}/2^{n-k}$.

2.1.1.1 Hardest decoding distance

This last equation allows to plot the following Fig. 2.2 which gives the complexity of Prange to find any solution to a decoding problem as a function of the distance. Notably one can see

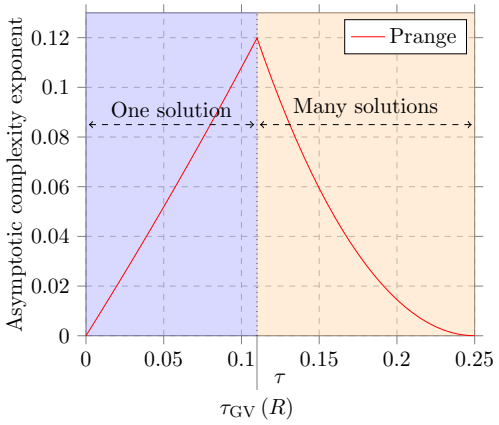


Figure 2.2: Asymptotic complexity exponent $\alpha(R,\ \tau)$, relative to the codelength n of the Prange algorithm as a function of the relative weight τ when decoding codes of rate $R = 0.5$. More precisely $\alpha(R,\ \tau) \stackrel{\text{def}}{=} h(\tau) - (1-R)h(\tau/(1-R)) - \max(0,\ h(\tau) - (1-R))$

two distinct regime in this figure, one in which the asymptotic exponent is increasing and the other decreasing at a peak around $\tau \approx 0.11$. The first represents the injective regime, that is when we expect that the only solution is the planted one, the complexity increases with τ as the Prange bet gets less likely to be valid. This is true up until a certain point where the complexity peaks. This peak coincides with the relative Gilbert-Varshamov distance $\tau_{\text{GV}}(R)$ which is the distance where we expect that there exists exactly one non-planted solution. When τ grows even more, the decoding problem is expected to admit even more solutions, an exponential number of them, this actually makes the problem easier as we are looking for any solution and not one in particular.

Note also that the asymptotic complexity of all subsequent ISD we present have the same shape: they peak at the Gilbert-Varshamov distance. Because it is usually at this distance that the difference of complexity is the biggest it is traditional to compare the complexity of the new decoders at this point.

2.1.1.2 Polynomial regime

Notably it is easy to see that when $t = (n-k)/2$ the complexity of the Prange decoder becomes polynomial. It can be shown directly with the previous equality, but it is insightful here to rather look at the typical shape of the error produced by an iteration of the Prange decoder. Forgetting about the setting in which a solution is planted suppose we are given an uniformly random vector $\mathbf{y}$ of $\mathbb{F}_2^n$ and $\mathcal{C} \sim \mathcal{U}_{\mathbf{H}}(n,\ k)$ and say a fixed subset $\mathscr{I}$. Then, conditioned on the event that $\mathbf{y}_{\mathscr{I}} \neq \mathbf{0}$ and that $\mathscr{I}$ is an information set of $\mathcal{C}$ we have, using Fact 7, that $\widetilde{\mathbf{c}}_{\overline{\mathscr{I}}}$ is distributed as $\mathcal{U}\left(\mathbb{F}_2^{n-k}\right)$ and is independent of $\mathbf{y}_{\overline{\mathscr{I}}}$. By definition of $\widetilde{\mathbf{e}} \stackrel{\text{def}}{=} \mathbf{y} - \widetilde{\mathbf{c}}$ this means that $\widetilde{\mathbf{e}}_{\overline{\mathscr{I}}} \sim \mathcal{U}\left(\mathbb{F}_2^{n-k}\right)$. Moreover, by construction of $\widetilde{\mathbf{c}}$ we have naturally that $\widetilde{\mathbf{e}}_{\mathscr{I}} = \mathbf{0}$, thus, $\widetilde{\mathbf{e}}$ has a typical weight of $(n-k)/2$. So we really only have to make a polynomial number of iterations

to make sure that this weight is obtained. Notably it is also possible to slightly tweak Prange in order to make its complexity polynomial for any $t \in [\![(n-k)/2,\ n/2]\!]$: instead of lifting $\mathbf{y}_{\mathscr{N}}$ we can lift $\mathbf{y}_{\mathscr{I}} + \mathbf{r}$ where $\mathbf{r}$ is some artificial additional error of weight $t - (n-k)/2$ to make sure that the final error is still of typical weight t.

2.1.1.3 When the error weight is sublinear in the codelength

Importantly, the complexity of the Prange decoder when the error weight is sublinear in n is as follows.

Proposition 9 (Complexity of Prange in the sublinear regime [CS15])**.** *Let n growing to infinity. There exists N_{iter} an implicit function of n such that for any k, t implicit functions of n such that $t = o(n)$ and such that $R \overset{def}{=} k/n$ is bounded away from 0 and 1, then Algorithm 1 solves $\mathrm{DP}_{\mathbf{H}}(n, k, t)$ with probability $1 - o(1)$ in time*

$$\widetilde{\mathcal{O}}\Big(\, 2^{-t \log_2(1-R)}\Big)$$

Quite amazingly even after almost 60 years of research no algorithm is known to strictly beat the first order exponent (here they are the terms in the exponent which are linear in t) of the Prange decoder. More precisely, even the most advanced ISD's have are only marginally better than Prange in the sublinear regime as they have a time complexity of

$$\mathrm{poly}\,(n)\ 2^{-t \log_2(1-R) - o(t)}$$

where the $o(t)$ is some positive term [CS15]. It remains a major open question of to devise a better decoding algorithm in this regime.

2.1.1.4 Behavior at extreme rates

We can show that when decoding codes of extreme rates at Gilbert-Varshamov distance the Prange decoder really has the complexity of naive enumeration algorithms. In the low rate regime its complexity is that of the complexity of enumerating all the codewords, namely 2^k and in the high rate regime its complexity is that enumerating the errors, namely $\binom{n}{t}$ which is of the order of 2^{n-k} when t is at the Gilbert-Varshamov distance.

Proposition 10 (Performance of Prange at the extreme rates)**.** *Let R and τ be implicit functions of $n \in \mathbb{N}$ and define $k \overset{def}{=} \lfloor Rn \rfloor$ and $t \overset{def}{=} \lfloor \tau_{\mathrm{GV}}(R)\, n \rfloor$. There exists N_{iter} an implicit function of n such that the time complexity of Algorithm 1 to solve $\mathrm{DP}_{\mathbf{H}}(n, k, t)$ with probability $\Omega(1)$ as n grows to infinity is*

$$T \overset{def}{=} \widetilde{\mathcal{O}}\Big(2^{R(1+o(1))n}\Big) \qquad \text{if } R = o(1)\,, \tag{2.2}$$

$$T \overset{def}{=} \widetilde{\mathcal{O}}\Big(2^{(1-R)(1+o(1))n}\Big) \qquad \text{if } R = 1 - o(1)\,. \tag{2.3}$$

We prove it here quickly here as we have not been able to find a proof of this statement in the literature (for example Eq. (2.3) can be found without a proof in [Deb23, Proposition 3.0.1]). Let us give the outline of the proof of Eq. (2.3), when $R = 1 - o(1)$. Recall that the complexity of Prange, as given in Corollary 2 is

$$\widetilde{\mathcal{O}}\Big(2^{\alpha_{\mathrm{Prange}}(R,\, \tau_{\mathrm{GV}}(R))(1+o(1))n}\Big)$$

where

$$\alpha_{\text{Prange}}(R,\ \tau) \stackrel{\text{def}}{=} h(\tau) - (1-R)\, h(\tau/(1-R)).$$

As we decode at the relative Gilbert-Varshamov distance we have that $\tau = \tau_{\text{GV}}(R) = h^{-1}(1-R)$ or alternatively that $1 - R = h(\tau)$. By replacing we have that

$$\alpha_{\text{Prange}}(R,\ \tau_{\text{GV}}(R)) = (1-R)\left[1 - h\left(\tau_{\text{GV}}(R)/h(\tau_{\text{GV}}(R))\right)\right].$$

This means that to prove Eq. (2.3) we are only left with proving that $h\left(\tau_{\text{GV}}(R)/h(\tau_{\text{GV}}(R))\right) = o(1)$ when $R = 1 - o(1)$. To prove it we use several times the following facts

$$\tau_{\text{GV}}(R) = o(1) \quad \text{when} \quad R = 1 - o(1) \tag{2.4}$$

$$h(x) = -x\log_2(x)\,(1+o(1)) \quad \text{when} \quad x = o(1). \tag{2.5}$$

Let us suppose that $R = 1 - o(1)$. We have that

$$\frac{\tau_{\text{GV}}(R)}{h(\tau_{\text{GV}}(R))} = \frac{\tau_{\text{GV}}(R)}{-\tau_{\text{GV}}(R)\log_2(\tau_{\text{GV}}(R))\,(1+o(1))} \qquad \text{(Using Eqs. (2.4) and (2.5))} \tag{2.6}$$

$$= \frac{1}{-\log_2(\tau_{\text{GV}}(R))\,(1+o(1))} \tag{2.7}$$

$$= o(1) \qquad \text{(Using Eq. (2.4))} \tag{2.8}$$

As such, plugging this last equation into Eq. (2.5) we get that

$$\begin{aligned} h\left(\frac{\tau_{\text{GV}}(R)}{h(\tau_{\text{GV}}(R))}\right) &= -\frac{\tau_{\text{GV}}(R)}{h(\tau_{\text{GV}}(R))}\log_2\left(\frac{\tau_{\text{GV}}(R)}{h(\tau_{\text{GV}}(R))}\right)(1+o(1)) \\ &= \frac{\log_2\left(-1/\log_2(\tau_{\text{GV}}(R))\,(1+o(1))\right)}{\log_2(\tau_{\text{GV}}(R))\,(1+o(1))}(1+o(1)) \qquad \text{(Using twice Eq. (2.7))} \\ &= \frac{-\log_2\left(-\log_2(\tau_{\text{GV}}(R))\,(1+o(1))\right)}{\log_2(\tau_{\text{GV}}(R))\,(1+o(1))}(1+o(1)) \\ &= o(1) \end{aligned}$$

which shows our result. Note that due to this log log term the convergence is extremely slow, this is why one cannot really glimpse in Fig. 2.1 that the Prange algorithm really is the complexity of the naive error enumeration.

2.1.2 Reducing decoding to decoding a code of higher rate

Idea. As we will see in the next section, we explicit some collision based decoders which are vastly inefficient compared to Prange decoder for moderate code rates but benefit from an exponential gain for codes of high rates. Most notably in certain regime those collision decoders can recover all the solution to a decoding problem with an optimal complexity: it can be equal to the number of solutions. The first of these was a collision based decoder presented by Dumer in [Dum86], gaining essentially a square root factor over naive error enumeration, and thus, in light of Proposition 10, gaining a square root over the Pange decoder in the high rate regime when decoding at the Gilbert-Varsmahov distance. Later, in [Dum89] (and to some extent [Ste88, LB88]) used the fact that decoding $\mathcal{C}$ can be done by decoding $\mathcal{C}$ punctured in some positions, the point being that for codes of high rates we can

use the aforementioned collision decoder. The idea is simply to choose a subset of positions $\mathscr{I} \subset [\![1,\, n]\!]$ and puncturing the code on those positions and basically only looking at the problem of decoding

$$\mathbf{y}_{\mathscr{I}} = \mathbf{c}_{\mathscr{I}} + \mathbf{e}_{\mathscr{I}}$$

in the higher rate code $\mathcal{C}_{\mathscr{I}}$. The idea here is to relax the Prange bet by allowing a few numbers of errors this allows to gain in the probability of success but now we have to solve an additional decoding problem. Note that as long as $\mathscr{I}$ contains an information set of $\mathcal{C}$ then recovering $\mathbf{c}$ from $\mathbf{e}_{\mathscr{I}}$ can be done by lifting $\mathbf{y}_{\mathscr{I}} - \mathbf{e}_{\mathscr{I}}$ so basically we need to find all the solutions to this smaller decoding problem, namely compute (or a subset)

$$\{\ \widetilde{\mathbf{e}_{\mathscr{I}}} \in \mathcal{S}_p^n\ :\ \mathbf{y}_{\mathscr{I}} - \widetilde{\mathbf{e}_{\mathscr{I}}} \in \mathcal{C}_{\mathscr{I}}\}$$

and test each $\widetilde{\mathbf{e}_{\mathscr{I}}}$ for $\mathbf{e}_{\mathscr{I}}$. Importantly so that the bet is valid with higher probability we want $\mathscr{I}$ to be the smallest possible but here contrary to Prange where $|\mathscr{I}| = k$, we have to take $\mathscr{I}$ bigger in order to mitigate the number of solutions to this smaller decoding problem which can get prohibitively high. Indeed, provided that $\mathscr{I}$ contains an information set, this really is decoding a code of length $|\mathscr{I}|$ and dimension k. This introduces an extra parameter ℓ which defines the extra number of chosen positions, namely $|\mathscr{I}| = k + \ell$.

Important point: parameter scaling. Note that in the regime of interest in this thesis, that is when the rate R of the code is constant and the error weight $|\mathbf{e}| = t$ grows linearly in the codelength n, namely $t = \tau n$ for some τ then the introduced extra parameters ℓ and p also grow linearly in n. This really allows us the have an exponential impact on the complexity of the algorithm. Indeed, betting that $|\mathbf{e}_{\mathscr{I}}| = p$ where p grows linearly in n allows to increase by an exponential factor the proabability that the bet is valid compared to the full zero Prange bet. This also means that ℓ must also grow linearly in t (hence n) so that this gain in probability is not lost in the number of solutions of the smaller decoding problem. Still in the regime of interest this number of solutions will be exponentially big but the collision decoders which are used can in certain setting produce many solutions in constant amortized time. Of course the specific value of the scaling constant ρ, λ such that $\ell = \lambda n$ and $p = \rho n$ are then determined by minimizing the time complexity of the algorithm which will be a function of the rate of the code R and the relative error weight τ.

One must keep in mind that in all this thesis that whenever new parameters are introduced, they scale linearly in the code length in our regime of interest.

Algorithm. This framework is written explicitly in Algorithm 2.

Algorithm 2 Dumer ISD framework as in [Dum89]

Name: DUMER-ISD-FRAMEWORK($\mathcal{C}$, $\mathbf{y}$, t)
Parameter: N_{iter}, ℓ

for $i = 1 \cdots N_{\text{iter}}$ **do** ▷ *N_{iter} will be exponential in t*
  $\mathscr{I} \xleftarrow{\$} \{\mathscr{I} \subset [\![1, n]\!] : |\mathscr{I}| = k + \ell\}$ ▷ *Hope that the error verify the bet, for example say that $|\mathbf{e}_{\mathscr{I}}| = p$ for a certain p*
  $\mathcal{S} \leftarrow$ SUB-DECODER($\mathcal{C}_{\mathscr{I}}$, $\mathbf{y}_{\mathscr{I}}$) ▷ *Return a set of errors $\widetilde{\mathbf{e}_{\mathscr{I}}}$ of low weight such that $\mathbf{y}_{\mathscr{I}} - \widetilde{\mathbf{e}_{\mathscr{I}}} \in \mathcal{C}_{\mathscr{I}}$*
  for $\widetilde{\mathbf{e}_{\mathscr{I}}} \in \mathcal{S}$ **do**
    $\widetilde{\mathbf{c}} \leftarrow \text{Lift}\left(\mathcal{C}, \mathscr{I}, \mathbf{y}_{\mathscr{I}} + \widetilde{\mathbf{e}_{\mathscr{I}}}\right)$
    $\widetilde{\mathbf{e}} \leftarrow \mathbf{y} - \widetilde{\mathbf{c}}$
    if $|\widetilde{\mathbf{e}}| = t$ **then**
      return $\widetilde{\mathbf{e}}$

As a remark note that taking $p = t$, $\ell = n - k$ and $N_{\text{iter}} = 1$ are valid choice of parameters for Algorithm 2, thus this generic framework is necessarily at least as good as the DECODER subroutine it uses.

In this setting, we clearly have the following giving the complexity of Algorithm 2.

Proposition 11. *Let $n \in \mathbb{N}$ be growing to infinity. For any k, t, p, $\ell \in \mathbb{N}$ functions of n and any procedure* SUB-DECODER *which is such that*

$$\forall \mathbf{e}' \in \mathcal{S}_p^{n-s}, \quad \mathbb{P}_{(\mathcal{D},\mathbf{z})\sim \mathrm{DP}_{\mathbf{H}}(k+\ell,k,p)}\left(\mathbf{e}' \in \text{SUB-DECODER}(\mathcal{D}, \mathbf{z}) \,\middle|\, \mathbf{z} + \mathbf{e}' \in \mathcal{D}\right) = 1 - o(1)$$

then there exists N_{iter} an implicit function of n such that Algorithm 2 solves $\mathrm{DP}_{\mathbf{H}}(n, k, t)$ with probability $1 - o(1)$ in time and memory

$$\textbf{Time} = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{\binom{n-k-\ell}{t-p}\binom{k+\ell}{p}} T_{\text{Sub-Decoder}}\right), \qquad \textbf{Memory} = \widetilde{\mathcal{O}}(M_{\text{Sub-Decoder}})$$

where $T_{\text{Sub-Decoder}}$ and $M_{\text{Sub-Decoder}}$ are respectively the time and memory complexity of SUB-DECODER.

We present next two of those high rate efficient decoders [Dum89, BJMM12] which are used inside this framework. To distinguish between the high rate efficient decoder from the ISD's that uses it we call the former the subroutine (i.e. [Dum89] subroutine and [BJMM12] subroutine) and the latter without specification (or sometimes with an ISD in front). We will see that each can naturally be used to produce essentially all the solutions to the decoding problem.

2.1.2.1 Collision based decoders for the high rate regime

In this section we devise combinatoric decoders to decode a noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$ onto a linear code $\mathcal{C}$. These decoders will in practice always be used as a subroutine inside Algorithm 2.

Opposite to the Prange decoder where basically we enumerate tuples of positions that we hope are error free, here we rather enumerate erroneous positions and the goal here is to exhaust the whole search space of the error, namely $\mathcal{S}_t^n$, without naively enumerating all the $\binom{n}{t}$ possibilities. These combinatoric decoders leverage the fact that if we know that a

solution $\mathbf{e}$ to the decoding problem, namely such that $\mathbf{y}+\mathbf{e} \in \mathcal{C}$, can be expressed as a sum $\mathbf{e} = \mathbf{e}^{(1)} + \mathbf{e}^{(2)}$ of elements belonging to smaller sets $\mathbf{e}^{(1)} \in \mathcal{S}_1$ and $\mathbf{e}^{(2)} \in \mathcal{S}_1$ then it is sufficient to compute $\mathcal{S}_1$ and $\mathcal{S}_2$ and merging them by computing $\mathcal{S}_1 \bowtie_{\mathcal{C}}^{(\mathbf{y})} \mathcal{S}_2$ as defined next.

Definition 20 (Merging sets)**.** *Let $\mathbf{y} \in \mathbb{F}_2^n$ and let $\mathcal{C}$ be a linear code of length n. Let $\mathcal{S}_1, \mathcal{S}_2$ be two sets composed of vectors of $\mathbb{F}_2^n$. We define*

$$\mathcal{S}_1 \bowtie_{\mathcal{C}}^{(\mathbf{y})} \mathcal{S}_2 \stackrel{def}{=} \{\mathbf{e} \,:\, \exists \left(\mathbf{e}^{(1)},\, \mathbf{e}^{(2)}\right) \in \mathcal{S}_1 \times \mathcal{S}_1 \,:\, \mathbf{e} = \mathbf{e}^{(1)} + \mathbf{e}^{(2)} \text{ and } \mathbf{y}+\mathbf{e} \in \mathcal{C}\}.$$

In practice, this is done by taking $\mathbf{H} = \mathbf{H}(\mathcal{C})$ a parity-check matrix of $\mathcal{C}$, storing $\mathbf{H}\mathbf{e}^{(1)}$ for all $\mathbf{e}^{(1)} \in \mathcal{S}_1$ in a hash-table and checking for each $\mathbf{e}^{(2)} \in \mathcal{S}_2$ if $\mathbf{H}\left(\mathbf{e}^{(2)} + \mathbf{y}\right)$ is in the hash-table. Hence, the cost is, up to polynomial factors, the size of $\mathcal{S}_1$ and $\mathcal{S}_2$ and the size of $\mathcal{S}_1 \bowtie_{\mathcal{C}}^{(\mathbf{y})} \mathcal{S}_2$ (i.e. the number of found solutions to the decoding problem).

Lemma 4. *Given $\mathcal{S}_1, \mathcal{S}_2, \mathcal{C}$, $\mathbf{y}$, the time complexity of computing $\mathcal{S}_1 \bowtie_{\mathcal{C}}^{(\mathbf{y})} \mathcal{S}_2$ is*

$$\widetilde{\mathcal{O}}\left(\left(|\mathcal{S}_1| + |\mathcal{S}_2| + \left|\mathcal{S}_1 \bowtie_{\mathcal{C}}^{(\mathbf{y})} \mathcal{S}_2\right|\right)\right).$$

2.1.2.2 Dumer's collision decoder a.k.a Dumer subroutine

The question really is now how to choose the subsets $\mathcal{S}_i$ well enough to minimize their size while maximizing the intersection of their sum with $\mathcal{S}_t^n$. Because of the lack of additive structure of the sphere it is handy, to minimize the search space, to consider two disjoint subsets I_1 and I_2 and compute

$$\mathcal{S}_1 \stackrel{\text{def}}{=} \{\mathbf{e} \in \mathbb{F}_2^n \,:\, \mathbf{e}_{I_1} \in \mathcal{S}_{\lfloor t/2 \rfloor}^{\lfloor n/2 \rfloor} \text{ and } \mathbf{e}_{I_2} = \mathbf{0}\}.$$
$$\mathcal{S}_2 \stackrel{\text{def}}{=} \{\mathbf{e} \in \mathbb{F}_2^n \,:\, \mathbf{e}_{I_2} \in \mathcal{S}_{\lceil t/2 \rceil}^{\lceil n/2 \rceil} \text{ and } \mathbf{e}_{I_1} = \mathbf{0}\}.$$

This is a good choice as:

Lemma 5. *There exists a poly-bounded function such that for any $n, t \in \mathbb{N}$ such that $t \leqslant n$ and any vector $\mathbf{e} \in \mathcal{S}_t^n$ we have that*

$$\mathbb{P}_{I_1}\left(\mathbf{e} \in \mathcal{S}_1 + \mathcal{S}_2\right) \geqslant \frac{1}{f(n)}$$

Consequently it is readily seen that randomizing the choice of I_1 and iterating a polynomial number of times allows guaranteeing that we exhaust the whole search space with good probability.

Algorithm 3 Dumer first decoder [Dum86] (Or [Dum89] subroutine)

Name: DUMER-SUBROUTINE($\mathcal{C}$, $\mathbf{y}$, t)
Parameter: N_{iter}, STOP ▷ *N_{iter} is a polynomial in n*

1: ▷ *Returns FAIL if the algorithm takes more than* STOP *step to finish. Used to upper bound the complexity of the algorithm without damaging the probability of success*
2: $\mathcal{S} \leftarrow \emptyset$
3: **for** $i = 1 \cdots N_{\text{iter}}$ **do**
4: $\quad \mathscr{I}_1 \xleftarrow{\$} \{\mathscr{I} \subset [\![1, n]\!] \ : \ |\mathscr{I}| = \lfloor n/2 \rfloor\}$
5: $\quad \mathscr{I}_2 \leftarrow [\![1, n]\!] \setminus \mathscr{I}_1$
6: $\quad \mathcal{S}_1 \leftarrow \{\mathbf{e} \in \mathbb{F}_2^n \ : \ |\mathbf{e}_{\mathscr{I}_1}| = \lfloor t/2 \rfloor \text{ and } \mathbf{e}_{\mathscr{I}_2} = \mathbf{0}\}$
7: $\quad \mathcal{S}_2 \leftarrow \{\mathbf{e} \in \mathbb{F}_2^n \ : \ |\mathbf{e}_{\mathscr{I}_2}| = \lceil t/2 \rceil \text{ and } \mathbf{e}_{\mathscr{I}_1} = \mathbf{0}\}$
8: $\quad \mathcal{S} \leftarrow \mathcal{S} \bigcup \left[\mathcal{S}_2 \bowtie_{\mathcal{C}}^{(\mathbf{y})} \mathcal{S}_1\right]$
9: **return** $\mathcal{S}$

All in all, it is easy to show that in fact Dumer's decoder returns all the solution to the decoding problem with good probability.

Proposition 12. *There exist N_{iter}, STOP some implicit function of n such that for any $k, t \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$, Algorithm 3, when given an instance $(\mathcal{C}, \mathbf{y})$ of $\mathrm{DP}_{\mathbf{H}}(n, k, t)$, outputs with probability $1 - o(1)$ the set $\mathcal{C} + \mathbf{y} \bigcap \mathcal{S}_t^n$ in*

$$\textbf{\textit{Time}} = \widetilde{\mathcal{O}}\left(\left(\sqrt{\binom{n}{t}} + \frac{\binom{n}{t}}{2^{n-k}}\right)\right), \qquad \textbf{\textit{Memory}} = \widetilde{\mathcal{O}}\left(\sqrt{\binom{n}{t}}\right).$$

Proof. As the chosen subsets are independent in each iteration, is is easy to see that it is sufficient that N_{iter} is a big enough polynomial to ensure that with probability $1 - o(1)$, for every $\mathbf{e} \in \mathcal{S}_t^n$ there exists an iteration such that $\mathbf{e}$ can be expressed as a sum of an element of $\mathcal{S}_1$ and $\mathcal{S}_2$. Concerning the time complexity of the algorithm it is readily seen that

$$|S_i| \leqslant \binom{\lceil n/2 \rceil}{\lceil t/2 \rceil} = \widetilde{\mathcal{O}}\left(\sqrt{\binom{n}{t}}\right),$$

$$\mathbb{E}\left(\mathcal{S}_2 \bowtie_{\mathcal{C}}^{(\mathbf{y})} \mathcal{S}_1\right) \leqslant \mathbb{E}\left(\mathcal{C} + \mathbf{y} \bigcap \mathcal{S}_t^n\right) \leqslant \frac{\binom{n}{t}}{2^{n-k}} + 1.$$

Consequently, the expected cost C of an iteration is

$$C = \widetilde{\mathcal{O}}\left(\left(\sqrt{\binom{n}{t}} + \frac{\binom{n}{t}}{2^{n-k}}\right)\right).$$

Finally, choosing STOP to be $n\, N_{\text{iter}}\, C$ allows concluding, using Markov inequality, that with probability $1 - o(1)$ the algorithm will stop by itself before less than STOP operations. □

Remark 4. *Note that we made the previous proposition the most rigorous possible to give the general idea of the proof strategy behind these propositions. But to keep our statement and algorithm simple, from now on, we make the* STOP *parameter implicit, but keep in mind that it will always secretly be there to upper bound the complexity of the algorithms without damaging the probability of success.*

Remark 5. *Importantly one can note that in fact this very simple decoder can, in the regime* $\binom{n}{t}/2^{n-k} > \sqrt{\binom{n}{t}}$ *compute solutions in amortized time* $\mathcal{O}(1)$*. This means that the cost of the algorithm is essentially the number of solutions.*

Last, clearly, when decoding at Gilbert-Varshamov distance, in the regime where the rate $R = k/n$ the complexity of Dumer's decoder is

$$\widetilde{\mathcal{O}}\left(\sqrt{2^{n-k}}\right),$$

this is the square root of the Prange decoder in the high rate regime.

Computing all the low-weight dual vectors with Dumer subroutine. Related to our purpose, this routine can also trivially be used to compute all the low-weight dual codewords of $\mathcal{C}$: we only have to give as input the code $\mathcal{C}^{\perp}$ and the vector $\mathbf{y} = \mathbf{0}$. We will, in particular, give the asymptotic complexity exponent of the dual attack we devise in this thesis when this subroutine is used to produce dual vectors. As such, for completeness we give its asymptotic complexity exponent.

Proposition 13. *Let* $R, \tau \in \mathbb{N}$ *be implicit functions of* $n \in \mathbb{N}$ *and let* $k \stackrel{def}{=} \lfloor R\, n \rfloor$ *and* $t \stackrel{def}{=} \lfloor \tau\, n \rfloor$*. There exists an algorithm that when given an instance* $(\mathcal{C},\ \mathbf{y})$ *of* $\mathrm{DP}_{\mathbf{G}}(n, k, t)$*, outputs with probability* $1-o(1)$ *the set* $\mathcal{C}^{\perp} \bigcap \mathcal{S}_t^n$ *in time and memory respectively* $\widetilde{\mathcal{O}}\left(2^{\alpha_{dual-Dumer-routine}(R,\, \tau)n}\right)$ *and* $\widetilde{\mathcal{O}}\left(2^{\beta_{dual-Dumer-routine}(R,\, \tau)n}\right)$ *where:*

$$\alpha_{dual-Dumer-routine}\,(R,\ \tau) \stackrel{def}{=} \max\left(\frac{h\,(\tau)}{2},\ h\,(\tau) - R\right)$$

$$\beta_{dual-Dumer-routine}\,(R,\ \tau) \stackrel{def}{=} \frac{h\,(\tau)}{2}.$$

2.1.2.3 Using representation to decode, MMT11 and BJMM12

Dumer's collision decoder was then improved by using a so called representation technique, which was originally introduced and improved in [HJ10, BCJ11] to solve hard Knapsack instances and which was then adapted to solve the decoding problem in [MMT11, BJMM12]. It allows gaining in some regime more than a square root over the naive enumeration and provides an asymptotic improvement over Dumer's decoder. We present here the general idea behind the technique and then present [BJMM12].

General idea. They note that the error vector $\mathbf{e} \in \mathcal{S}_t^n$ has an exponential number of low weights representations, that is couple $\left(\mathbf{e}^{(1)}, \mathbf{e}^{(2)}\right)$ of vectors of low weights, say each vector is in $\mathcal{S}_{t'}^n$ where $t' \geqslant t/2$, and which are such that $\mathbf{e} = \mathbf{e}^{(1)} + \mathbf{e}^{(2)}$. Their idea is, rather than to search for the error vector directly, to search for one of its representations, the key gain will come from the fact that they are able to decimate the space of representations efficiently, namely construct two lists, $\mathcal{L}_1$ and $\mathcal{L}_2$, of vectors of $\mathcal{S}_{t'}^n$ with the following properties.

1. (Decimation) Each list is of size essentially $\binom{n}{t'}/\textsc{Repr}(n, t, t')$ where $\textsc{Repr}(n, t, t')$ is the number of representations of $\mathbf{e}$ as given in Lemma 6 and depends only on $t = |\mathbf{e}|$, n the ambient space dimension and t' the weight of the representations. Roughly we can think of this as an oblivious operation: for each vector $\mathbf{e} \in \mathcal{S}_t^n$ the first list say contains essentially only one $\mathbf{e}^{(1)}$ such that there exists $\mathbf{e}^{(2)}$ such that $(\mathbf{e}^{(1)}, \mathbf{e}^{(2)})$ is a representation of $\mathbf{e}$.

2. (Correctness) Crucially, when $\mathbf{e}$ is a solution to the decoding problem, the second list contains the associated $\mathbf{e}^{(2)}$.

Lemma 6. *Given $\mathbf{e} \in \mathbb{F}_2^n$ we define the set of $t'-$representation of $\mathbf{e}$ as*

$$\textsc{Repr}(\mathbf{e}, t') = \{\left(\mathbf{e}^{(1)}, \mathbf{e}^{(2)}\right) \in \mathcal{S}_{t'}^n \times \mathcal{S}_{t'}^n \ : \ \mathbf{e} = \mathbf{e}^{(1)} + \mathbf{e}^{(2)}\}.$$

For $\mathbf{e} \in \mathcal{S}_t^n$ the number of $t'-$representations is given by

$$\textsc{Repr}(n, t, t') \stackrel{def}{=} \binom{t}{t/2}\binom{n-t}{t'-t/2}$$

when t is even, else it is 0.

The fact that we are able to verify both properties is what makes this technique viable. When we want to solve a decoding problem this is possible by exploiting the linearity of the problem: choosing some random linear code $\mathcal{C}'$ of length n and with right dimension k', say

$$n - k' \approx \textsc{Repr}(n, t, t')$$

to get filter out the representations and compute

$$\mathcal{L}_1 = \{ \ \mathbf{e}^{(1)} \in \mathcal{S}_{t'}^n \ : \ \mathbf{e}^{(1)} \in \mathcal{C}'\}.$$

The correctness is obtained by noting that if $\mathcal{C}'$ is in fact a supercode of $\mathcal{C}$ we have:

Fact 10. *Suppose $\mathbf{y} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e} = \mathbf{e}^{(1)} + \mathbf{e}^{(2)}$ and that $\mathcal{C} \subset \mathcal{C}'$*

$$\mathbf{e}^{(1)} \in \mathcal{C}' \Leftrightarrow \mathbf{y} + \mathbf{e}^{(2)} \in \mathcal{C}'.$$

Thus, taking

$$\mathcal{L}_2 = \{ \ \mathbf{e}^{(2)} \in \mathcal{S}_{t'}^n \ : \ \mathbf{y} + \mathbf{e}^{(2)} \in \mathcal{C}'\}$$

is the right corresponding choice. One can note that computing both lists is in fact solving yet another decoding problem (or finding low weight codeword), but now we are decoding an $[n, k']$-linear code of higher dimension but at a smaller distance t'. At last, we can recover the error $\mathbf{e}$ by computing $\mathcal{L}_1 \bowtie_{\mathbf{y}}^{(\mathcal{C})} \mathcal{L}_2$ and then by filtering the resulting list to keep only the solution of good weight t. In practice for the optimal parameters, this last filtering step will in fact delete an exponential number of ill-weighted elements, this essentially comes from the discussion below.

Because $t' < t$ this new decoding problem is easier than the original one in the sense that, say using Dumer's decoder we now have to construct lists of smaller size $\sqrt{\binom{n}{t'}}$ instead of the $\sqrt{\binom{n}{t}}$ at first. The whole tradeoff comes from the fact that we want that $t' \geqslant t/2$ is the smallest possible so that this new decoding problem gets easier but the smallest t' the biggest the number of solutions of this new decoding problem gets, namely the size $\binom{n}{t'}/\textsc{Repr}(n, t, t')$ of the lists $\mathcal{L}_1, \mathcal{L}_2$ increases, and can at some point dominate the cost.

In practice, to get in a zone where Dumer's decoder is efficient, this is iterated a few times, namely we use this procedure recursively to compute $\mathcal{L}_1$ and $\mathcal{L}_2$, and finally we use Dumer's decoder at the end.

The algorithm. As we said, the strategy described above is applied recursively m times, in the original proposal [BJMM12]. Say at level 0 you want to solve the original decoding problem, i.e. compute

$$\mathcal{L}_0 = \{ \mathbf{e} \in \mathcal{S}_t^n \ : \ \mathbf{y} + \mathbf{e} \in \mathcal{C}\}, \qquad t_0 \stackrel{\text{def}}{=} t.$$

This is done by choosing a super code $\mathcal{C}'$ of $\mathcal{C}$ constructed by choosing the last r_1 rows of the parity-check matrix of $\mathcal{C}$ and searching for weight t_1 representations of the error that are in the sets

$$\mathcal{L}_{1,1} = \{ \mathbf{e}^{(1)} \in \mathcal{S}_{t_1}^n \ : \ \mathbf{y} + \mathbf{e}^{(1)} \in \mathcal{C}'\},$$
$$\mathcal{L}_{1,2} = \{ \mathbf{e}^{(2)} \in \mathcal{S}_{t_1}^n \ : \ \mathbf{e}^{(2)} \in \mathcal{C}'\}.$$

Carefully choosing t_1 and r_1 together allows to ensure that one representation survives with good probability, see Lemma 7. Now, computing the lists $\mathcal{L}_{1,1}$ and $\mathcal{L}_{1,2}$ is really just solving another decoding problem and this is done by applying the procedure recursively. At level j, to solve the decoding problem at distance t_j and codimension r_j we construct a new super code by taking the last r_{j+1} rows of the parity-check matrix and look for t_{j+1} representations of the solution (of the decoding problem considered at level j). At last, at the last level $j = m$ the problem is solved with the Dumer subroutine. In the original proposal of [BJMM12] the algorithm presented was taken with 3 levels of recursions, namely $m = 3$. Using more levels do not appear to make significant changes in the complexity. The 3-depth algorithm is written in Algorithm 4. Note that we also included, as it was the case in the original article, the fact that this whole procedure is iterated a polynomial number of times and randomized to find at the end the solution with probability $1 - o(1)$, this is possible of course provided that the parameters verify the constraint of Lemma 7. This randomization is done by taking a random shift $\mathbf{r}$ each time and search for representations that are such that $\mathbf{y} + \mathbf{e}^{(1)} + \mathbf{r} \in \mathcal{C}'$ and $\mathbf{e}^{(2)} + \mathbf{r} \in \mathcal{C}'$.

Lemma 7. *There exists a positive poly-bounded function f such that for any $t, k, t_1, r_1 \in \mathbb{N}$ implicit functions of n such that*

$$\textsc{Repr}(n, t, t_1) \geqslant 2^{r_1}$$

then

$$\mathbb{P}\left(\exists \left(\mathbf{e}^{(1)}, \ \mathbf{e}^{(2)}\right) \in \textsc{Repr}(\mathbf{e}, t_1) \ : \ \mathbf{y} + \mathbf{e}^{(1)} \in \mathcal{C}' \ \middle| \ \mathbf{y} + \mathbf{e} \in \mathcal{C}\right) \geqslant \frac{1}{f(n)}$$

where $\mathcal{C}$, $\mathbf{y}$ is an instance of $\mathrm{DP}_{\mathbf{H}}(n, k, t)$ and where $\mathcal{C}'$ is the code whose parity-check matrix is composed of the first r_1 rows of the parity-check matrix $\mathbf{H}(\mathcal{C})$ of $\mathcal{C}$ and $\mathbf{e}$ is any fixed vector of $\mathcal{S}_t^n$.

Algorithm 4 BJMM subroutine

Name: BJMM-Subroutine($\mathcal{C}$, $\mathbf{y}$, t)

Parameter: $\mathbf{r}$, $\mathbf{t}$, N_{iter} ▷ $\mathbf{r}$, $\mathbf{t}$ *are vectors containing the codimension of thesupercodes and the weight of the representations at each level. We always have that* $t_0 = t$*, the distance of the original decoding problem*

$\mathcal{S} \leftarrow \emptyset$
while $i = 1 \cdots N_{\text{iter}}$ **do**
  $\mathcal{S} \leftarrow \mathcal{S} \bigcup \text{BJMM-Rec}(\mathcal{C}, \mathbf{y}, 0)$
procedure BJMM-Rec($\mathcal{C}$, $\mathbf{y}$, j) ▷ *j is level of recursion*
  $\mathcal{L} \leftarrow \emptyset$ ▷ *Contains the found solutions* $\mathbf{e}$ *of weight* t *such that* $\mathbf{y} + \mathbf{e} \in \mathcal{C}$
  $\mathbf{r} \xleftarrow{\$} \mathbb{F}_2^n$ ▷ *Allows to randomize each iteration*
  $\mathcal{C}' \leftarrow$ The code whose parity check-matrix is obtained by keeping only the last r_j rows of $\mathbf{H}(\mathcal{C})$
  if $j < 2$ **then**
    $\mathcal{L}_1 \leftarrow \text{BJMM-Rec}(\mathcal{C}', \mathbf{y} + \mathbf{r}, j + 1)$
    $\mathcal{L}_2 \leftarrow \text{BJMM-Rec}(\mathcal{C}', \mathbf{r}, j + 1)$
  else
    $\mathcal{L}_1 \leftarrow \text{Dumer-Subroutine}(\mathcal{C}', \mathbf{y}, t_j; N_{\text{iter}} = 1)$
    $\mathcal{L}_2 \leftarrow \text{Dumer-Subroutine}(\mathcal{C}', \mathbf{0}, t_j; N_{\text{iter}} = 1)$
  $\mathcal{L}' \leftarrow \mathcal{L}_1 \bowtie_{\mathcal{C}'}^{(\mathbf{y})} \mathcal{L}_2$
  $\mathcal{L} \leftarrow \mathcal{L} \bigcup \{\mathbf{e} \in \mathcal{L} : |\mathbf{e}| = t_j\}$ ▷ $t_0 \leftarrow t$
  return $\mathcal{L}$

The performance of this subroutine is given by the following statement.

Corollary 3 (Corollary of [BJMM12, Theorem 1]. The complexity of the BJMM subroutine Algorithm 4)**.** *Let* $n \in \mathbb{N}$ *be growing to infinity. For any* $k, t \in \mathbb{N}$ *and* $\mathbf{r} \in \mathbb{N}^2$, $\mathbf{t} \in \mathbb{N}^2$ *implicit functions of* n *such that*

$$\text{Repr}(n, t, t_1) \in \tilde{\Theta}\left(2^{r_1}\right) \qquad \text{and} \qquad \text{Repr}(n, t_1, t_2) \in \tilde{\Theta}\left(2^{r_2}\right) \tag{2.9}$$

There exists a poly-bounded function N_{iter} *such that Algorithm 4 solves* $\text{DP}_{\mathbf{H}}(n, k, t)$ *with probability* $1 - o(1)$ *in*

$$\textbf{\textit{Time}} = \widetilde{\mathcal{O}}(\max(T_1, T_2, T_3)), \qquad \textbf{\textit{Memory}} = \widetilde{\mathcal{O}}(\max(S_1, S_2, S_3)).$$

where

$$S_1 = \frac{\binom{n}{t_1}}{2^{r_1}}, \qquad S_2 = \frac{\binom{n}{t_2}}{2^{r_2}}, \qquad S_3 = \sqrt{\binom{n}{t_2}}$$

and

$$T_1 = \max\left(S_1, \frac{S_1^2}{2^{(n-k)-r_1}}\right), \quad T_2 = \max\left(S_2, \frac{S_2^2}{2^{r_1-r_2}}\right), \quad T_3 = \max\left(S_3, \frac{S_3^2}{2^{r_2}}\right).$$

Proof. This comes from [BJMM12, Theorem 1] by taking $\gamma = \mathcal{O}(\log_2 n)$ in the theorem. □

Computing short dual vectors. Again, we will also analyze our dual attack when this procedure is used to compute low weight dual vectors (by giving it as input $\mathcal{C}^\perp$ and $\mathbf{0}$). As such, for completness, because the time complexity exponent of our dual attack will depend on the time complexity exponent of this procedure we provide it next.

Proposition 14. *Asymptotic time complexity exponent of the BJMM subroutine to compute a proportion $1-o(1)$ of all the dual vectors of a certain weight. Let $n \in \mathbb{N}$ be growing to infinity. For any $R,\ \tau,\ \rho_1, \rho_2, \tau_1, \tau_2 \in]0,\ 1[$ implicit function of n such that*

$$\rho_2 \leqslant \rho_1 \leqslant (1-R), \quad 2\,\tau_2 \geqslant \tau_1, \quad 2\,\tau_1 \geqslant \tau$$

and

$$\rho_1 = \nu_{\text{repr}}\left(\tau,\ \tau_1\right) \qquad \text{and} \qquad \rho_2 = \nu_{\text{repr}}\left(\tau_1,\ \tau_2\right) \qquad \text{(enough representations)}$$

there exists an algorithm that is such that when given $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}\left(n,\ \lfloor \tau n \rfloor\right)$ this algorithm outputs with probability $1-o(1)$ a set $\mathcal{S} \subset \mathcal{C}^{\perp} \bigcap \mathcal{S}^n_{\lfloor \tau n \rfloor}$ which is such that $|\mathcal{S}| \,/\, \left|\mathcal{C}^{\perp} \bigcap \mathcal{S}^n_{\lfloor \tau n \rfloor}\right| = 1-o(1)$ in time and memory respectively $\widetilde{\mathcal{O}}\left(2^{\alpha_{\text{dual-BJMM-routine}}(R,\omega\,;\tau_1,\tau_2,\rho_1,\rho_2)n}\right)$ and $\widetilde{\mathcal{O}}\left(2^{\beta_{\text{dual-BJMM-routine}}(R,\omega\,;\tau_1,\tau_2,\rho_1,\rho_2)n}\right)$ where

$$\begin{aligned}
\alpha_{\text{dual-BJMM-routine}}(R,\omega\,;\tau_1,\tau_2,\rho_1,\rho_2) &\stackrel{def}{=} \max(\gamma_1,\gamma_2,\gamma_3),\\
\beta_{\text{dual-BJMM-routine}}(R,\omega\,;\tau_1,\tau_2,\rho_1,\rho_2) &\stackrel{def}{=} \max\left(\mu_1,\ \mu_2,\ \mu_3\right)
\end{aligned}$$

and where

$$\mu_1 \stackrel{def}{=} h(\tau_1) - \rho_1, \quad \mu_2 \stackrel{def}{=} h(\tau_2) - \rho_2, \quad \mu_3 \stackrel{def}{=} h(\tau_2)/2 \qquad \text{(list sizes)}$$

and

$$\gamma_1 \stackrel{def}{=} \max\left(\mu_1,\ 2\,\mu_1 - (1-R-\rho_1)\right), \quad \gamma_2 \stackrel{def}{=} \max\left(\mu_2,\ 2\,\mu_2 - (\rho_1-\rho_2)\right), \quad \gamma_3 \stackrel{def}{=} \max\left(\mu_3,\ 2\,\mu_3 - \rho_2\right).$$

Definition 21. *Given $R,\ \tau$ we denote more simply*

$$\begin{aligned}
\alpha_{\text{dual-BJMM-routine}}(R,\omega\,) &\stackrel{def}{=} \alpha_{\text{dual-BJMM-routine}}(R,\omega\,;\pi_1^*,\pi_2^*,\lambda_1^*,\lambda_2^*)\\
\beta_{\text{dual-BJMM-routine}}(R,\omega\,) &\stackrel{def}{=} \beta_{\text{dual-BJMM-routine}}(R,\omega\,;\pi_1^*,\pi_2^*,\lambda_1^*,\lambda_2^*)
\end{aligned}$$

and where the parameters $(\pi_1^*,\pi_2^*,\lambda_1^*,\lambda_2^*)$ minimize the time complexity exponent, $\alpha_{\text{dual-BJMM-routine}}(R,\omega\,;\tau_1^*,\tau_2^*,\rho_1^*,\rho_2^*)$ under the previous constraints.

2.1.3 Reducing decoding to a noisy syndrome decoding problem

In this section we give the ideas behind some of the most recent solvers [MO15, BM17, BM18, Ess23]. The idea of these algorithms is simply to notice that, starting again from Prange, when we select a set $\mathscr{I}$ of k positions we can relax the bet by allowing that $\mathbf{e}_{\mathscr{I}}$ contains a few erroneous positions and try to recover $\mathbf{e}_{\mathscr{I}}$ by making the following observation.

Fact 11. *Let $\mathcal{C}$ be an $[n,k]$-linear code. Let $\mathbf{y} \in \mathbb{F}_2^n$ be a vector and let $\mathscr{I} \subset [\![1,\ n]\!]$ be a information set of $\mathcal{C}$ of size k. Let*

$$\mathbf{R} \stackrel{def}{=} \text{Lift}\left(\mathcal{C},\ \mathscr{I}\right), \qquad\qquad \mathbf{s}' \stackrel{def}{=} \mathbf{y}_{\overline{\mathscr{I}}} + \mathbf{R}\mathbf{y}_{\mathscr{I}}^{\intercal}.$$

Let $\mathbf{e} \in \mathbb{F}_2^n$ be a vector, we have that

$$\mathbf{y} + \mathbf{e} \in \mathcal{C} \qquad \Leftrightarrow \qquad \left|\mathbf{R}\mathbf{e}_{\mathscr{I}}^{\intercal} - \mathbf{s}'\right| = t - \left|\mathbf{e}_{\mathscr{I}}\right|. \tag{2.10}$$

Proof. Suppose $\mathbf{y} + \mathbf{e} \in \mathcal{C}$. Thus $\mathbf{y} = \mathbf{c} + \mathbf{e}$ for some $\mathbf{c} \in \mathcal{C}$. By definition of the lift $\mathbf{R}$ we have that

$$\mathbf{R}\mathbf{e}^\intercal_{\mathscr{I}} = \mathbf{R}\mathbf{c}^\intercal_{\mathscr{I}} + \mathbf{R}\mathbf{y}^\intercal_{\mathscr{I}} = \mathbf{c}_{\overline{\mathscr{I}}} + \mathbf{R}\mathbf{y}^\intercal_{\mathscr{I}} = \mathbf{y}_{\overline{\mathscr{I}}} + \mathbf{e}_{\overline{\mathscr{I}}} + \mathbf{R}\mathbf{y}^\intercal_{\mathscr{I}} = \mathbf{s}' + \mathbf{e}_{\overline{\mathscr{I}}}.$$

The converse is the same. □

So really, here provided that $|\mathbf{e}_{\mathscr{I}}|$ is of betted weight, say w, then we recovering $\mathbf{e}_{\mathscr{I}}$ is really just finding a weight w vector $\mathbf{z}$ which is such that

$$|\mathbf{R}\mathbf{z}^\intercal + \mathbf{s}| = t - w.$$

We call this problem the noisy syndrome decoding problem and define its average variant as follows.

Definition 22 (Noisy syndrome decoding $\mathrm{NSDP}_{\mathbf{R}}(r, k, w, p)$)**.** *Let r, k, t, w the average noisy syndrome decoding problem $\mathrm{NSDP}_{\mathbf{R}}(r, k, t, w)$ is defined as follow: given $\mathbf{R} \sim \mathcal{U}\left(\mathbb{F}_2^{r \times k}\right)$ and $\mathbf{s} = \mathbf{R}\mathbf{z}^\intercal + \mathbf{e}'$ where $\mathbf{z} \sim \mathcal{U}\left(\mathcal{S}_w^k\right)$ and $\mathbf{e}' \sim \mathcal{U}\left(\mathcal{S}_p^r\right)$, the goal is to output an $\mathbf{z} \in \mathcal{S}_w^k$ such that $|\mathbf{R}\mathbf{z}^\intercal + \mathbf{s}| = p$.*

This problem is clearly hard and can be seen as some kind of generalization of the decoding problem. One important note is that what makes this approach viable is that there exists some solver for this problem which vastly outperforms the naive search. Of course this lead to a new tradeoff as this new problem gets harder and harder as w increases while the bet that $|\mathbf{e}_{\mathscr{I}}|$ gets verified becomes likelier. As always with these algorithms in the regime that are of interest to us (namely the rate is constant and the error weight $|\mathbf{e}| = t$ is linear in the codelength n), the bet on $|\mathbf{e}_{\mathscr{I}}|$ will essentially be linear in the error weight $|\mathbf{e}| = t$ so that it can really have an exponential impact on the algorithm.

Note that, from a high level point of view this is somewhat profitable compared to the Dumer ISD framework as we reduced the number of selected positions this allows to reduce the cost of the bet and to have a smaller search space for the error. On the contrary it is not straightforward to compare apriori the hardness of this new problem to the hardness of the higher rate decoding problem appearing in the Dumer ISD framework.

The algorithm is presented in Algorithm 5 and relies on a certain procedure solving the noisy syndrome decoding problem.

Algorithm 5 Reducing decoding to noisy-syndrome decoding [MO15]

Name: ReducingDecodingToNoisySyndrome($\mathcal{C}$, $\mathbf{y}$, t)

1: **for** $i = 1 \cdots N_{\text{iter}}$ **do** ▷ *N_{iter} will be exponential so that a bet on the error weight distribution is valid*
2: $\quad\mathscr{I} \leftarrow \{\mathscr{I} \subset [\![1, n]\!] : |\mathscr{I}| = k\}$
3: $\quad\mathbf{R} \leftarrow \mathrm{Lift}(\mathcal{C}, \mathscr{I})$
4: $\quad\mathbf{s} \leftarrow \mathbf{R}\mathbf{y}_{\mathscr{I}} + \mathbf{y}_{\overline{\mathscr{I}}}$ ▷ *The noisy syndrome*
5: $\quad\mathcal{S}' \leftarrow$ NoisySyndrome-Decoder($\mathbf{R}$, $\mathbf{s}$, w, $t - w$) ▷ *Return some $\mathbf{e}_{\mathscr{I}}$ such that $|\mathbf{R}\mathbf{e}_{\mathscr{I}} + \mathbf{s}'| = t - |\mathbf{e}_{\mathscr{I}}|$*
6: $\quad\mathcal{S} \leftarrow \{\, \mathbf{e} \in \mathcal{S}_t^n : \mathbf{e}_{\mathscr{I}} \in \mathcal{S}' \text{ and } \mathbf{e}_{\overline{\mathscr{I}}} = \mathbf{R}\mathbf{e}_{\mathscr{I}}\}$ ▷ *Each $\mathbf{e}_{\mathscr{I}}$ is lifted to a solution to the original decoding problem $\mathbf{y} + \mathbf{e} \in \mathcal{C}$*
7: $\quad$**return** $\mathcal{S}$

The complexity of the algorithm is given as follows.

Proposition 15. *Let $n \in \mathbb{N}$ be growing to infinity. Let k, w, p be functions of n. There exists N_{iter} a function of n such that provided that the procedure* NoisySyndrome-Decoder*() is such that it solves an instance of* $\text{NSDP}_{\mathbf{R}}(n-k, k, w, t-w)$ *with probability* $1-o(1)$ *then Algorithm 5 solves* $\text{DP}_{\mathbf{H}}(n, k, t)$ *with probability* $1-o(1)$ *in*

$$\textbf{\textit{Time}} = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{\binom{n-k}{t-w}\binom{k}{w}} T_{\text{NSD}}\right), \qquad \textbf{\textit{Memory}} = \widetilde{\mathcal{O}}(M_{\text{NSD}})$$

where T_{NSD} and M_{NSD} are respectively the time and memory complexity of NoisySyndrome-Decoder*().*

Last, note that there are in fact two variants of the previously introduced algorithm which are presented in [MO15], the other one is a mixed approach between the ISD Dumer framework and this approach which we do not present here as, as of today the most efficient approach comes from [BM18] which is in this framework.

2.1.3.1 Near-collisions techniques

Here we want to solve the $\text{NSDP}_{\mathbf{R}}(r, k, w, p)$ problem, namely we are given a matrix $\mathbf{R} \in \mathbb{F}_2^{r \times k}$ and a noisy syndrome $\mathbf{s} \in \mathbb{F}_2^{r \times 1}$ and the goal is to recover a solution $\mathbf{z} \in \mathcal{S}_w^k$ such that $|\mathbf{Rz} + \mathbf{s}| = p$. In some of the algorithm we present here we will sometime need to compute essentially all the solutions to this problem, in that case we can show easily that, without counting the planted solution, the expected number of solutions is of the order

$$\binom{k}{w}\frac{\binom{n}{p}}{2^{n-k}}.$$

Of course the goal here is to do better than enumerating naively $\mathcal{S}_w^k$. The idea here is, similar to the Dumer collision decoder to somehow diminish the search space. Namely, to compute two smaller subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ which are such that we can guarantee that the solution $\mathbf{z}$ to the noisy syndrome problem can be decomposed as

$$\mathbf{z} = \mathbf{z_1} + \mathbf{z_2}, \qquad \text{where} \qquad \mathbf{z_1} \in \mathcal{S}_1 \text{ and } \mathbf{z_2} \in \mathcal{S}_2.$$

But contrary to the Dumer collision decoder, from there, recovering $\mathbf{z}$ reduces to solving a near-collision problem at distance p. Namely, as described in Algorithm 6, we just have to compute

$$\begin{aligned}\mathcal{L}_1 &= \{\ \mathbf{Rz_1}^\intercal\ :\ \mathbf{z_1} \in \mathcal{S}_1\}\\ \mathcal{L}_2 &= \{\ \mathbf{Rz_1}^\intercal + \mathbf{s}\ :\ \mathbf{z_2} \in \mathcal{S}_2\}\end{aligned}$$

and then find near-collisions, that is couple $(\mathbf{a},\ \mathbf{b}) \in \mathcal{L}_1 \times \mathcal{L}_2$ that are such that there sum is of low weight:

$$|\mathbf{a} + \mathbf{b}| = p.$$

Algorithm 6

Name: NEAR-COLLIDE($\mathcal{S}_1,\ \mathcal{S}_2,\ \mathbf{R},\ \mathbf{s},\ p$) ▷ *Returns some elements* $\mathbf{z}$ *such that* $\mathbf{z} = \mathbf{z_1} + \mathbf{z_2}$ *where* $\mathbf{z_1},\ \mathbf{z_2} \in \mathcal{S}_1 \times \mathcal{S}_2$ *and* $|\mathbf{R}\mathbf{z_1}^\intercal + \mathbf{z_2}^\intercal\ + \mathbf{s}| = p$

1: $\mathcal{L}_1 \leftarrow \{\ \mathbf{R}\mathbf{z_1}^\intercal\ :\ \mathbf{z_1} \in \mathcal{S}_1\}$
2: $\mathcal{L}_2 \leftarrow \{\ \mathbf{R}\mathbf{z_2}^\intercal + \mathbf{s}\ :\ \mathbf{z_2} \in \mathcal{S}_2\}$
3: $\mathcal{L} \leftarrow$ NEAR-COLLISION($\mathcal{L}_1,\ \mathcal{L}_2,\ p$) ▷ *Return a subset of* $\{(\mathbf{a},\ \mathbf{b})\ \in \mathcal{L}_1 \times \mathcal{L}_2\ :\ |\mathbf{a} + \mathbf{b}| = p\}$
4: $\mathcal{S} \leftarrow \{\ \mathbf{z_1} + \mathbf{z_2}\ :\ (\mathbf{z_1},\ \mathbf{z_2}) \in \mathcal{S}_1 \times \mathcal{S}_2$ and $(\mathbf{R}\mathbf{z_1}^\intercal,\ \mathbf{R}\mathbf{z_2}^\intercal + \mathbf{s}) \in \mathcal{L}\}$ ▷ *is done in time* $\widetilde{\mathcal{O}}(|\mathcal{L}|)$ *with a hash table*
5: **return** $\mathcal{S}$

2.1.3.1.1 Complexity of near-collisions techniques As explained before, because all the parameters grows linearly in the codelength in our regime of interest, these list will be of exponential size thus it is important to devise better algorithm than the naive quadratic $\widetilde{\mathcal{O}}(|\mathcal{L}_1|\,|\mathcal{L}_2|)$ near collision algorithm. Say here, as it will be the case in practice that they are, up to polynimal factors of equal size. [MO15] devised a Locallity-Sensitive-Hashing based procedure to find near-collision which was tailored for these regimes, later [Car20] noticed that the result from [MO15] could be generalized by using some older Locality-Sensitive-based techniques which where never analyzed in this setting. Then [EKZ21] extended [MO15] work and obtained essentially (up to different superpolynomial but subexponential factors) the same results as [Car20]. We state the result here which we will use for now one as the complexity of the NEAR-COLLISION procedure and give just after an overview, in Section 2.1.3.1.2 of the algorithm underlying the following theorem.

Theorem 1 (Crollary of [Car20, Theorem 9.1.5] or [EKZ21, Theorem 1, p. 8] and the discussion around.)**.** *Let* n *be growing to infinity and let* $\tau \in]0, 1/2[$ *and* $\lambda \in]0, 1[$ *be constants. There exists an algorithm which is such that when given two random lists* $\mathcal{L}_1,\ \mathcal{L}_2$ *which are such that* $|\mathcal{L}_1| \in \widetilde{\mathcal{O}}\left(2^{\lambda n}\right),\ |\mathcal{L}_2| \in \widetilde{\mathcal{O}}\left(2^{\lambda n}\right)$ *and which are composed of random vectors of* $\mathbb{F}_2^n$ *whose distribution satisfies the following properties:*

- *For each index* i*, the* i*'th element of each list is uniformly distributed in* $\mathbb{F}_2^n$*.*
- *There exists at most a number* $\widetilde{\mathcal{O}}(1)$ *of couples of index* (i^*, j^*) *such that the* i^**'th element of* $\mathcal{L}_1$ *and the* j^**'the element of* $\mathcal{L}_2$ *are dependent.*

then this algorithm outputs a set $\mathcal{S} \subset \{\ (\mathbf{x},\ \mathbf{y}) \in \mathcal{L}_1 \times \mathcal{L}_2\ :\ |\mathbf{x} + \mathbf{x}| = \lfloor \tau n \rfloor\}$ *which is such that any element of* $\{\ (\mathbf{x},\ \mathbf{y}) \in \mathcal{L}_1 \times \mathcal{L}_2\ :\ |\mathbf{x} + \mathbf{x}| = \lfloor \tau n \rfloor\}$ *is in* $\mathcal{S}$ *with probability* $1 - o(1)$*. This is done in time and memory upper bounded by*

$$\left(2^{\lambda n}\right)^{\alpha_{\mathrm{NNS}}(\lambda,\omega)(1+o(1))}$$

where

$$\alpha_{\mathrm{NNS}}(\lambda, \omega) \stackrel{def}{=} \begin{cases} (1-\tau)\left(1 - h\left(\frac{\tau^* - \tau/2}{1-\tau}\right)\right) & \text{if } \tau \leqslant 2\,\tau^*\,(1-\tau^*) \\ 2\lambda + h(\tau) - 1 & \text{otherwise} \end{cases} \tag{2.11}$$

and $\tau^* \stackrel{def}{=} h^{-1}(1-\lambda)$*.*

Remark 6. *Note that in both [Car20] and [MO15, EKZ21] there is a hidden superpolynomial factor coming from the* $2^{o(1)n}$ *term in the complexity* $2^{\alpha_{\text{NNS}}(\lambda,\omega)(1+o(1))n}$*. In [Car20] this* $o(1)$ *term is in* $\Theta\left(\sqrt{n\log(n)}\right)$ *but he also gives a way to make it polynomial without a proof but with experimental evidences [Car20, Figure 9.3] supporting his claim.*

Other near-collision techniques. There exists some other techniques that can be used and which are not based on this Locality-Sensitive-Hashing framework but rather on some fast matrix multiplication techniques [Val15, KKK18, KKKOC16, Alm19, AZ23] and which can, in certain regime strictly beat these Locality-Sensitive-Hashing based techniques. But the typical setting in which these last algorithms are tailored is to find a unique planted pair of close vector in two lists in a regime where we do not expect any other non-planted solutions. More precisely the complexity statement of [KKK18, Alm19] is stated as follows: for a given constant $\rho > 0$ there exists a $\lambda > 0$ such that as long as $|\mathcal{L}_1| = \mathcal{O}\left(2^{\lambda n}\right)$ then finding a planted pair of vector at distance ρn can be done in time $\mathcal{O}\left(|\mathcal{L}_1|^{1.582}\right)$. This means that this theorem still applies when the list are of exponential size (but not too big) and the distance is linear in n. The latest work by [AZ23] proposed for the first time a mix between the LSH approach and this matrix multiplication framework (this approach is not better than the LSH framework in all the regimes though) and we don't know what this last approach would yield in performance in our setting.

2.1.3.1.2 Using codes to find near-collisions We give here some detail and history about the algorithm underlying Theorem 1. The algorithm which are traditionally used to solve this near-collision problem or other related problems are Locality-Sensitive-Hashing (LSH) based [DHL$^+$94, BE98, IM98, GMO10, Dub10, MO15, Car20, EKZ21].In a nutshell a LSH here is a somewhat balanced many-to-one function which verifies the key property that it maps two random elements of the space, say $\mathbb{F}_2^n$, to the same value with increasing probability as these two elements gets closer. Having access to such an LSH function gives an advantage to find near-collision: by hashing each element of each list we can naively compare only those elements of the lists whose hash is the same as this allows to compare only those pairs which already have an increased probability of being close. In practice, we really need a family of LSH so this can be randomized and done sufficiently many times so that the advantage becomes non negligible. One can see that there is a competition between the output size of the LSH that we would like to be as big as possible to diminish the number of comparison and the locality property we require on it, which naturally requires that this output is not too big. Solving this near-collision problem using this framework was initiated by [DHL$^+$94] and it was proposed there to use codes to build LSH: from a code $\mathcal{C} \subset \mathbb{F}_2^n$ locally hashing an element of $\mathbb{F}_2^n$ can be done by decoding it to a close codeword. We give a description of this framework in Algorithm 7.

Algorithm 7 LSH Framework

Name: NEAR-COLLISION($\mathcal{L}_1$, $\mathcal{L}_2$)
Input: $\mathcal{L}_1$, $\mathcal{L}_2$
Parameter: n, w, $\mathcal{F}$ ▷ *The ambient space is $\mathbb{F}_2^n$, we are looking for vectors at distance w using a family $\mathcal{F}$ of code of length n*

1: **for** $i = 1 \cdots N_{\text{iter}}$ **do** ▷ *Can be polynomial or exponential depending on the list size, the length of the vector and the target weight.*
2: $\quad \mathcal{C} \xleftarrow{\$} \mathcal{F}$
3: $\quad \mathcal{H} \leftarrow \emptyset$ ▷ *Hash table*
4: $\quad \mathcal{S} \leftarrow \emptyset$ ▷ *Set containing the near collisions*
5: $\quad$ **for** $\mathbf{x} \in \mathcal{L}_1$ **do**
6: $\quad\quad \mathbf{c} \leftarrow \text{DECODE}(\mathcal{C}, \mathbf{x})$ ▷ *We must have access for each code $\mathcal{C}$ in the family $\mathcal{F}$ to a procedure $\text{DECODE}(\mathcal{C}, \mathbf{x})$ which outputs efficiently a codeword of $\mathcal{C}$ close to $\mathbf{x}$, the closer the better*
7: $\quad\quad \mathcal{H}[\mathbf{c}]$ .APPEND($\mathbf{x}$) ▷ *Add $\mathbf{x}$ in the entry $\mathbf{c}$ of the hash-table*
8: $\quad$ **for** $\mathbf{y} \in \mathcal{L}_2$ **do**
9: $\quad\quad \mathbf{c} \leftarrow \text{DECODE}(\mathcal{C}, \mathbf{y})$
10: $\quad\quad$ **for** $\mathbf{x} \in \mathcal{H}[\mathbf{c}]$ **do**
11: $\quad\quad\quad$ If $|\mathbf{x} + \mathbf{y}| = w$ then $\mathcal{S} \leftarrow \mathcal{S} \bigcup \{(\mathbf{x}, \mathbf{y})\}$
12: $\quad$ **return** $\mathcal{S}$

When the vectors of the lists are distributed uniformly at random in the ambiant space $\mathbb{F}_2^n$. When the vectors in each list $\mathcal{L}_1$ and $\mathcal{L}_2$ are distributed uniformly at random in $\mathbb{F}_2^n$ one can convince himself that taking codes with good covering capability and decoding at the nearest codeword seems like a good choice to ensure that the space $\mathbb{F}_2^n$ is split in roughly equal sized buckets while maximizing the probability that two close vectors are decoded onto the same codeword. Clearly, each phenomenon compete depending on the number of codewords, say 2^k, in the code $\mathcal{C}$.

This motivates the choice for analyzing this LSH framework with random codes (say $\mathscr{F}$ is the family of all $[n, k]$-linear code for example, but the linearity does not change anything to the analysis) and neglect the cost of decoding to obtain an idea of what is achievable. A first analysis was done in [GMO10] but in a setting that is inapplicable here and then [Car20] made the first analysis in the setting we are interested in and this ultimately lead to Theorem 1, note that the analysis is technical N_{iter} and k can be optimized analytically.

Note this is not completely sufficient to state Theorem 1 because we still need a family of code which would be as good as random code but which we know how to decode. To this end [Car20] proposed to use, inspired by [BDGL16], a cartesian product $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_b$ of $b = \mathcal{O}\left(\sqrt{n/log(n)}\right)$ small random linear codes where each $\mathcal{C}_i$ is of length $\mathcal{O}\left(\sqrt{n \log(n)}\right)$. In this case, decoding a word can be done sub exponential but super polynomial time $2^{\mathcal{O}\left(\sqrt{n \log(n)}\right)} = 2^{o(n)}$ by essentially enumerating the codewords of each code, in an independent manner. Last it is also shown there that the use of a product of small random code instead of a random code incurs a loss of order $2^{o(n)}$ in the probability that two close elements decode onto the same codeword (this translates into an additional factor $2^{o(n)}$ in the number of iterations N_{iter} we have to make). Both phenomena are the reason behind the super polynomial term in Theorem 1.

Lastly, [Car20] proposed to use Polar codes [Arı09] and their list decoder [TV15] instead of product of small codes. It was conjectured that this allowed to yield only a polynomial overhead, this was supported with some experimental evidences [Car20, Figure 9.3]. The rough intuition behind this choice and behind this conjecture is that Polar codes and their decoder are good for compression, namely they achieve the Shannon rate-distortion bounds over a Binary Symmetric Channel.

2.1.3.2 A simple near-collision based noisy syndrome solver

For example, a simple algorithm Algorithm 8 using this Near-Collision routine was devised in [MO15] and consists essentially in splitting the support in two equal part an enumerating weight $w/2$ vectors on each part in a Dumer like fashion.

Algorithm 8 Simplest variant of [MO15] to solve the noisy syndrome problem

Name: NSD-MO15($\mathbf{R}$, $\mathbf{s}$, w, p)

1: $\mathscr{I}_1 \xleftarrow{\$} \{\mathscr{I} \subset [\![1, n]\!] \ : \ |\mathscr{I}| = \lfloor n/2 \rfloor\}$
2: $\mathscr{I}_2 \leftarrow [\![1, n]\!] \setminus \mathscr{I}_1$
3: $\mathcal{S}_1 \leftarrow \{\mathbf{e} \in \mathbb{F}_2^n \ : \ |\mathbf{e}_{\mathscr{I}_1}| = \lfloor w/2 \rfloor \text{ and } \mathbf{e}_{\mathscr{I}_2} = \mathbf{0}\}$
4: $\mathcal{S}_2 \leftarrow \{\mathbf{e} \in \mathbb{F}_2^n \ : \ |\mathbf{e}_{\mathscr{I}_2}| = \lceil w/2 \rceil \text{ and } \mathbf{e}_{\mathscr{I}_1} = \mathbf{0}\}$
5: $\mathcal{L} \leftarrow \mathrm{NNS}(\mathcal{S}_1, \mathcal{S}_2, \mathbf{R}, \mathbf{s}, p)$
6: **return** $\mathcal{L}$

2.1.3.3 State of the art: representations and the BM18 algorithm

This last algorithm to solve the noisy syndrome decoding problem was then improved in [BM18] by adapting the representation technique (see Section 2.1.2.3) to this particular setting. To this date, this algorithm, when used inside the "Reducing decoding to the noisy syndrome decoding" framework given in Algorithm 5 is the best algorithm to decode random linear codes of constant rates at Gilbert-Varshamov distance. This algorithm, similar to [BJMM12] is a multi-depth algorithm. We start by presenting the depth-2 variant to give the idea and then give the depth-4 variant that is currently used to set the baseline of the complexity of the best decoders (the authors of [BM18, Page 21] note that they were unable to improve the complexity due to the space of parameters that started to get too large for their optimizer).

Finally, we show that there was an error in the original analysis of this algorithm which lead to a significative underestimation of the complexity of the algorithm. We correct the analysis and give new complexity estimates.

2.1.3.3.1 The algorithm General idea. Here again, instead of searching for the error vector $\mathbf{z} \in \mathcal{S}_w^k$ which is solution to our noisy syndrome decoding problem

$$|\mathbf{Rz} + \mathbf{s}| = p.$$

we search for say a weight t_1 representant $(\mathbf{z_1}, \mathbf{z_2})$ of $\mathbf{z}$, namely

$$\mathbf{z} = \mathbf{z_1} + \mathbf{z_2}, \qquad \text{where} \qquad |\mathbf{z_1}| = |\mathbf{z_2}| = t_1.$$

The goal here is, as before, to compute two lists of $\mathcal{L}_1,\ \mathcal{L}_2$ such that if $\mathbf{z}$ is a solution to our noisy syndrome then there exists a representation $(\mathbf{z_1},\ \mathbf{z_2})$ of $\mathbf{z}$ which belongs to $\mathcal{L}_1 \times \mathcal{L}_2$. This solution can then be recovered with a Near-Collision search. Here, the key equality coming into play to ensure that we can control the decimation, namely ensuing that we can filter out $\mathbf{z_1}$ and $\mathbf{z_2}$ somewhat independently is the fact that we have that

$$|\mathbf{R}\mathbf{z_2} + \mathbf{R}\mathbf{z_1} + \mathbf{s}| = p.$$

Say for example and to simplify that we consider the unique weight p_1 which is such that the sum of two uniformly random vectors of weight p and p_1 respectively is of typical weight p_1, then we can ensure that if $|\mathbf{R}\mathbf{z_1}| = p_1$ then with probability $\tilde{\Omega}(1)$ we have that $|\mathbf{R}\mathbf{z_2} + \mathbf{s}| = p_1$. Of course, to ensure that there can actually exist such a representant (here the condition is way too strong), we filter out the representants which verify a similar condition but on a restricted subset of the rows of $\mathbf{R}$, and choose the number of rows such that only one representant survives the condition. One can really see that finding the representant verifying this condition is really just solving yet another easier related noisy syndrome decoding problem and that this technique can be applied recursively. We add also that in general the typical weight condition on p_1 is relaxed to reduce to an easier sub-problem but at the cost of losing the one to one correspondence $\mathbf{z_1} \in \mathcal{L}_1 \Leftrightarrow \mathbf{z_2} \in \mathcal{L}_2$.

Describing more precisely the depth 2 algorithm. Let us be more precise, denoting by $\mathbf{R}^{(1)} \in \mathbb{F}_2^{r_1 \times k}$ and $\mathbf{s}^{(1)} \in \mathbb{F}_2^{r_1 \times 1}$ the last r_1 rows of $\mathbf{R}$ and $\mathbf{s}$ respectively, say

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}^{(0)} \\ \mathbf{R}^{(1)} \end{pmatrix}, \qquad \mathbf{s} = \begin{pmatrix} \mathbf{s}^{(0)} \\ \mathbf{s}^{(1)} \end{pmatrix},\ \mathbf{R}^{(1)} \in \mathbb{F}_2^{r_1 \times k},\ \mathbf{s}^{(1)} \in \mathbb{F}_2^{r_1 \times 1}.$$

and denote by $p^{(0)}$ and $p^{(1)}$ the weight on each part, namely

$$\left|\mathbf{R}^{(0)}\mathbf{z} + \mathbf{s}^{(0)}\right| = p^{(0)},$$
$$\left|\mathbf{R}^{(1)}\mathbf{z} + \mathbf{s}^{(1)}\right| = p^{(1)}$$

with

$$p^{(0)} + p^{(1)} = p.$$

The algorithm searchs for weight w_1 representations of $\mathbf{z}$ which are of weight p_1 on the first part, namely the algorithm computes

$$\mathcal{L}_1 = \{\ \mathbf{z_1} \in \mathcal{S}_{w_1}^k\ :\ \left|\mathbf{R}^{(1)}\mathbf{z_1}^\intercal\right| = p_1\},$$
$$\mathcal{L}_2 = \{\ \mathbf{z_2} \in \mathcal{S}_{w_1}^k\ :\ \left|\mathbf{R}^{(1)}\mathbf{z_2}^\intercal + \mathbf{s}^{(1)}\right| = p_1\}.$$

In this depth-2 variant these lists are computed with the previously introduced near-collision decoder Algorithm 8. Provided that all the quantities are well-chosen, namely in our case here that

$$\textsc{Repr}(k, w,\ w_1) \geqslant \frac{2^{r_1}}{\textsc{Repr}(r_1, p^{(1)}, p_1)}.$$

we can ensure that with good probability [BM18, Lemma 1], there exists some representations $(\mathbf{z_1}\mathbf{z_2})$ of $\mathbf{z}$ which are in $\mathcal{L}_1 \times \mathcal{L}_2$.

Then the algorithm recovers $\mathbf{z}$ by finding the pairs $(\mathbf{z}_1,\ \mathbf{z}_2) \in \mathcal{L}_1 \times \mathcal{L}_2$ which yield a solution to the noisy syndrome decoding problem, that is such that $|\mathbf{z_1} + \mathbf{z_2}| = w$ and $|\mathbf{Rz_1} + \mathbf{Rz_2} + \mathbf{s}| = p$. This is done in two steps. Fist by doing an near-collision search on the first part of the support to find the couples that are such that $\left|\mathbf{R}^{(0)}\mathbf{z_1} + \mathbf{R}^{(0)}\mathbf{z_2} + \mathbf{s}^{(0)}\right| = p^{(0)}$, that is we compute

$$\mathrm{NNS}(\mathcal{L}_1,\ \mathcal{L}_2,\ \mathbf{R}^{(0)},\ \mathbf{s}^{(0)}).$$

Second, it naively filters the resulting list to keep only the resulting couples $(\mathbf{z_1},\ \mathbf{z_2})$ which are such that $\mathbf{z_1} + \mathbf{z_2} = w$ and such that $\left|\mathbf{R}^{(1)}\mathbf{z_1} + \mathbf{R}^{(1)}\mathbf{z_2} + \mathbf{s}^{(2)}\right| = p^{(1)}$. Notably, this last filtering step can filter out an exponential number of candidates depending on the parameters.

Importantly, see that, apriori we might have to make a bet that $p^{(1)}$ and $p^{(2)}$ are not the typical weight in order to balance the costs of the different step of the algorithm. So really the algorithm bets that the weight of $|\mathbf{Rz} + \mathbf{s}| = p$ decomposes well and iterates until this bet is valid, each iteration is randomized by permuting the rows of $\mathbf{H}$ and $\mathbf{s}$.

The depth 4 algorithm. The full scale algorithm is in fact a depth m procedure where the above design rational is applied somewhat (but not completely) recursively: the algorithm keeps a very fine control on the shape of the representations. We present the depth 4-variant which was used in the original article to make the complexity claims.

Recall that we are looking for a solution $\mathbf{z}$ of the noisy syndrome decoding problem

$$|\mathbf{Rz}^\intercal + \mathbf{s}| = p, \qquad \text{where} \qquad |\mathbf{z}| = w, \qquad \mathbf{R} \in \mathbb{F}_2^{r \times k}.$$

The algorithm starts by making a bet on the weight of the syndrome on each subpart, namely we carefully choose some parameters r_i and $P_0^{(i)}$ and bet that

$$\forall i \in [\![0,\ 3]\!]\ \left|\mathbf{R}^{(i)}\mathbf{z}^\intercal + \mathbf{s}^{(i)}\right| = P_0^{(i)}$$

where

$$\sum_{i=1}^{3} P_0^{(i)} = p,\ \ \sum_{i=0}^{3} r_i = r,\ \ \mathbf{R} = \begin{pmatrix} \mathbf{R}^{(0)} \\ \vdots \\ \mathbf{R}^{(3)} \end{pmatrix},\ \ \mathbf{s} = \begin{pmatrix} \mathbf{s}^{(0)} \\ \vdots \\ \mathbf{s}^{(3)} \end{pmatrix},\ \ \forall i \in [\![1,\ 3]\!]\ \mathbf{R}^{(i)} \in \mathbb{F}_2^{r_i \times k},\ \mathbf{s}^{(i)} \in \mathbb{F}_2^{r_i \times 1}. \tag{2.12}$$

The overall goal is to compute the list

$$\mathcal{L}_0 = \{\ \mathbf{z} \in \mathcal{S}_w^k\ :\ \forall i \in [\![0,\ 3]\!]\ \left|\mathbf{R}^{(i)}\mathbf{z}^\intercal + \mathbf{s}^{(i)}\right| = \mathbf{P}_0^{(i)}\}$$

where, provided the bet is valid, our solution lies. Computing this list is done by searching for weight w_1 representations $(\mathbf{z_1},\ \mathbf{z_2})$ of $\mathbf{z}$ that are in

$$\mathcal{L}_{1,1} = \{\ \mathbf{z_1} \in \mathcal{S}_{w_1}^k\ :\ \forall i \in [\![1,\ 3]\!]\ \left|\mathbf{R}^{(i)}\mathbf{z_1}^\intercal\right| = \mathbf{P}_1^{(i)}\}$$
$$\mathcal{L}_{1,2} = \{\ \mathbf{z_2} \in \mathcal{S}_{w_1}^k\ :\ \forall i \in [\![1,\ 3]\!]\ \left|\mathbf{R}^{(i)}\mathbf{z_2}^\intercal + \mathbf{s}^{(i)}\right| = \mathbf{P}_1^{(i)}\}$$

where w_1 and the $\mathbf{P}_1^{(i)}$'s are carefully chosen parameters to ensure that essentially one representation $(\mathbf{z_1},\ \mathbf{z_2})$ belongs to $\mathcal{L}_{1,1} \times \mathcal{L}_{1,2}$. [BM18, Lemma 2] shows in that regard that is sufficient to take these parameters such that

$$\textsc{Repr}(k, w, w_1) \geqslant \prod_{h=1}^{3} \frac{2^{r_h}}{\textsc{Repr}(r_h, P_0^{(h)}, P_1^{(h)})} \tag{2.13}$$

to ensure that the aforementioned condition is met with sufficiently good probability. Now, each of these lists $\mathcal{L}_{1,1},\ \mathcal{L}_{1,2}$ are computed similarly up until the last level which is solved with the near-collision solver Algorithm 8. All in all it is readily seen, using again [BM18, Lemma 2] that it is sufficient to have, along with Eq. (2.13), that

$$\forall i \in [\![1,\ 2]\!] \qquad \textsc{Repr}(k, w_i, w_{i+1}) \geqslant \prod_{h=i+1}^{3} \frac{2^{r_h}}{\textsc{Repr}(r_h, P_i^{(h)}, P_{i+1}^{(h)})} \tag{2.14}$$

to ensure that at the end the solution to the original noisy syndrome decoding problem is in the final list with good probability. We give the algorithm in Algorithm 9.

Algorithm 9 [BM18] subroutine

Name: NSD-BM18($\mathbf{R},\ \mathbf{s},\ w,\ p$)
Parameter: $\mathbf{r},\ \mathbf{w},\ \mathbf{P}$
Parameter: N_{iter} ▷ *We always define w_0 (the first coordinate of $\mathbf{w}$) as equal to w the weight of the solution.*

while $i = 1 \cdots N_{\text{iter}}$ **do** ▷ *Iterate an exponential number of times until the bet is valid*
  $\mathbf{R}, \mathbf{s} \leftarrow$ Randomly permute the rows of $\mathbf{R}$ use the same permutation to permute the rows of $\mathbf{s}$
  Compute $\mathbf{R}^{(i)},\ \mathbf{s}^{(i)}$ for all $i \in [\![1,\ 4]\!]$ as in Eq. (2.12). ▷ *We have that $\mathbf{R}^{(i)} \in \mathbb{F}_2^{r_i \times k}$ and $\mathbf{s}^{(i)} \in \mathbb{F}_2^{r_i \times 1}$*
  $\mathcal{L}_{3,1} \leftarrow$ MO15-NS($\mathbf{R}^{(3)},\ \mathbf{s}^{(3)}\ ;\ w_3,\ P_3^{(3)}$)
  $\mathcal{L}_{3,2} \leftarrow$ MO15-NS($\mathbf{R}^{(3)},\ \mathbf{0}_{r_3}\ ;\ w_3,\ P_3^{(3)}$)
  while $j = 2, 1, 0$ **do**
    $\mathcal{L}'_{j,1} \leftarrow \text{NNS}(\mathcal{L}_{j+1,1},\ \mathcal{L}_{j+1,1},\ \mathbf{R}^{(j)},\ \mathbf{0}_{r_j},\ P_j^{(j)})$
    $\mathcal{L}_{j,1} \leftarrow \{\mathbf{z} \in \mathcal{L}'_{j,1}\ :\ |\mathbf{z}| = w_j \text{ and } \forall i \in [\![j,\ 3]\!]\ \left|\mathbf{R}^{(i)}\mathbf{z}^\intercal\right| = P_j^{(i)}\ \}$ ▷ *Filtering to keep the good weights*
    $\mathcal{L}'_{j,2} \leftarrow \text{NNS}(\mathcal{L}_{j+1,1},\ \mathcal{L}_{j+1,2},\ \mathbf{R}^{(j)},\ \mathbf{s}^{(j)},\ P_j^{(j)})$
    $\mathcal{L}_{j,2} \leftarrow \{\mathbf{z} \in \mathcal{L}'_{j,2}\ :\ |\mathbf{z}| = w_j \text{ and } \forall i \in [\![j,\ 3]\!]\ \left|\mathbf{R}^{(i)}zv^\intercal + \mathbf{s}^{(i)}\right| = P_j^{(i)}\ \}$
  return $\mathcal{L}_{0,2}$

2.1.3.4 Contribution: new complexity estimates for the state of the art

We found an error in the original analysis of the algorithm which lead to a big underestimation of the original complexity of the algorithm. The complexity of this algorithm crucially depends on the size of the lists $\mathcal{L}_{j,1},\ \mathcal{L}_{j,2}$ in Algorithm 9. At some point, the authors claim that for $j \in [\![0,\ 1]\!]$ we have that

$$\text{([BM18, Page 16])}\ \mathbb{E}\left(|\mathcal{L}_{i,1}|\right) \leqslant \binom{k}{w_i}$$
$$\mathbb{P}_{\mathbf{a} \sim \mathcal{U}\left(\mathbb{F}_2^{(r_j)}\right)}\left(|\mathbf{a}| = P_j^{(j)}\right) \prod_{h=j+1}^{3} \mathbb{P}_{\mathbf{a},\ \mathbf{b} \sim \mathcal{U}\left(\mathbb{F}_2^{(r_h)}\right)}\left(|\mathbf{a}+\mathbf{b}| = P_j^{(h)} \,\middle|\, |\mathbf{a}| = |\mathbf{b}| = P_{j+1}^{(h)}\right) \tag{2.15}$$

Those last probabilities represent the probability that a couple of $\mathcal{L}_{i+1,1} \times \mathcal{L}_{i+1,1}$ produces an element fit to be in $\mathcal{L}_{i,1}$. But clearly, the term in $\binom{k}{w_i}$ is not the right term that should be in front of those probabilities.

In fact the estimates given by Eq. (2.15) are largely too optimistic. For example, taking the original optimal parameters devised in [BM18] and re-computing the complexity of the algorithm with the news corrected list size estimates that we give next yields much worse complexity exponents. **A small fix.** We fix the analysis by noting that, by construction, for $j \in [\![0,\, 3]\!]$ we have

$$\mathcal{L}_{j,1} \subset \{\mathbf{z} \in \mathcal{S}_{w_j}^k \,:\, \forall i \in [\![j,\, 3]\!] \, \left|\mathbf{R}^{(i)}\mathbf{z}^\intercal\right| = P_j^{(i)} \,\}.$$

Thus, counting the potential planted solution we have that:

$$\mathbb{E}\left(|\mathcal{L}_{j,1}|\right) \leqslant \frac{\prod_{i=j}^{3} \binom{r_i}{P_j^{(i)}}}{2^{\sum_{i=j}^{3} r_i}} + 1. \tag{2.16}$$

In fact this bound is tight up to polynomial factors as long as the parameter verify the constraint that we have enough representations Eq. (2.13) and Eq. (2.14). The same holds for the other list $|\mathcal{L}_{j,2}|$.

Contribution : new complexity estimates for [BM18]. Let us now come back to the decoding problem $\mathrm{DP}_{\mathbf{H}}(n, k, t)$ and analyze the full [BM18] algorithm: that is when we use this last subroutine Algorithm 9 onto the noisy-syndrome framework Algorithm 5. The analysis of [BM18] relies on 3 ingredients.

- 1) Choosing the right number of iterations (in Algorithm 5 and Algorithm 9) such that the bet of the weight of the error on each subpart is valid.
- 2) Choosing the parameters correctly to ensure that a solution have a representation in each level, this is already given by Eq. (2.13) and Eq. (2.14).
- 3) Estimating the size of the lists. This is given by Eq. (2.16)

Concerning the first point, it is readily seen that in the main framework Algorithm 5 we want that there exists an iteration which is such that $|\mathbf{e}_{\mathscr{I}}| = w$. Now, to solve the noisy syndrome decoding problem given by $\left|\mathbf{R}\mathbf{e}_{\mathscr{I}}^\intercal + \mathbf{s}\right| = t - w$ where we recall that $\mathbf{R} = \mathrm{Lift}(\mathcal{C},\, \mathscr{I})$ and $\mathbf{s} = \mathbf{y}_{\overline{\mathscr{I}}} + \mathbf{R}\mathbf{y}_{\mathscr{I}}^\intercal$ as defined in Fact 11, Algorithm 9 iterates so that the there exists an iteration such that for all $i \in [\![0,\, 3]\!]$ we have that $\left|\mathbf{R}^{(i)}\mathbf{e}_{\mathscr{I}}^\intercal + \mathbf{s}^{(i)}\right| = P_0^{(i)}$. Clearly we need a total of

$$\widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{\binom{k}{w}\prod_{i=0}^{3}\binom{r_i}{P_0^{(i)}}}\right)$$

iterations to ensure that all these conditions are valid at least once.

All in all we can give the final theorem giving the performance of [BM18]. The following theorem is only a slightly rewritten variant of the discussion of [BM18, §4] but where we replaced the expected size of the list by our corrected value. Note that we also used the near-collision search procedure of Theorem 1. It is more general than near-collision technique from [MO15] that is used in [BM18]. The complexity results we gave in Fig. 2.1 concerning [BM18] were obtained by minimizing α_{BM18} under the constraints given in the theorem. We show in particular that the complexity of the algorithm was largely underestimated. However, Fig. 2.1 shows that this algorithm is still the state of the art, its time complexity exponent slightly beats the previous best ISD [BM17]. Those results were latter confirmed by Esser [Ess23]. Notably, he noticed furthermore that in fact [BM18] with the corrected analysis had a significantly better memory complexity (we did not look at the memory complexity in our original in our work published in [CDMT22]) than say the previous best decoders [BM17].

Theorem 2 (Corrected complexity exponent of [BM18])**.** *Let $n \in \mathbb{N}$ be growing to infinity and let $R, \tau \in]0,\ 1[$ be positive constants. If there exists some positive constants $\rho,\ \omega_0,\ \omega_1,\ \omega_2,\ \omega_3,\ \pi,\ \pi_3^{(3)},\ \pi_2^{(2)},\ \pi_2^{(3)},\ \pi_1^{(1)},\ \dots,\ \pi_1^{(3)},\ \pi_0^{(0)},\ \cdots,\ \pi_0^{(3)} \in]0,\ 1[$ such that*

$$\rho = 1 - R, \quad \pi = \tau - \omega, \quad \omega_0 = \omega, \quad 0 < \omega < \min(\tau,\ R/2), \quad \sum_{i=0}^{3} \pi_0^{(i)} = \pi, \quad \sum_{i=0}^{3} \rho_i = \rho$$

$$\forall i \in [\![0,\ 2]\!] 2\,\omega_{i+1} \geqslant \omega_i, \qquad \forall j \in [\![0,\ 3]\!]\ \ \forall i \in [\![j,\ 3]\!] 2\,\pi_{i+1}^{(j)} \geqslant \pi_i^{(j)}$$

and such that

$$\forall i \in [\![0,\ 2]\!], \quad R\,\nu_{\mathrm{Repr}}(\omega_i/R,\ \omega_{i+1}/R) = \sum_{h=i+1}^{3} \rho_h - \rho_h \nu_{\mathrm{Repr}}\left(\pi_i^{(h)}/\rho_h,\ \pi_{i+1}^{(h)}/\rho_h\right) \quad \textit{(Enough Representations)} \tag{2.17}$$

where the asymptotic number of representations ν_{repr} is defined as

$$\nu_{\mathrm{repr}}(\eta,\ \eta') \stackrel{def}{=} \eta + (1-\eta)\, h\left(\left(\eta' - \eta/2\right)/(1-\eta)\right)$$

then, there exists an algorithm that solves $\mathrm{DP}_{\mathbf{H}}(n, \lfloor Rn \rfloor, \lfloor \tau n \rfloor)$ with probability $1 - o(1)$ in time $2^{\alpha_{BM18}(1+o(1))n}$ and memory $2^{\beta_{BM18}(1+o(1))n}$ where

$$\alpha_{BM18} \stackrel{def}{=} \gamma_{\mathrm{Bet}} + \max(\gamma_0,\ \gamma_1,\ \gamma_2,\ \gamma_3)$$
$$\beta_{BM18} \stackrel{def}{=} \max(\lambda_1,\ \lambda_2,\ \lambda_3,\ \lambda_4)$$

and where

$$\gamma_i = \lambda_{i+1}\,\rho_i\,\alpha_{\mathrm{NNS}}\left(\lambda_{i+1}/\rho_i,\ \pi_i^{(i)}/\rho_i\right), \qquad \textit{for } i = 0 \cdots 3 \qquad \textit{(Cost of NNS)}$$

$$\lambda_4 \stackrel{def}{=} \frac{R}{2} h(\omega_3/R), \qquad \textit{(list size)}$$

$$\lambda_i \stackrel{def}{=} R\,h(\omega_i/R) - \sum_{h=i}^{3} \rho_h, \qquad \textit{for } i = 1 \cdots 3, \qquad \textit{(List size)}$$

$$\gamma_{\mathrm{Bet}} \stackrel{def}{=} h(\tau) - \left[Rh(\omega/R) + \sum_{i=0}^{3} \rho_i h\left(\pi_i^{(i)}/\rho_i\right) \right], \qquad \textit{(number of iteration)}$$

2.1.3.5 Further work

It was also proposed in [Ess23] a generalization of [BM18] by replacing the naive filtering steps that is done at each level (see Line 8 of Algorithm 9) and make it at the same time as the Near-Collision step, all at once. The difference is that now the parts of the syndrome appearing in the near-collision search are not uniformly distributed but rather are uniform on some subpart and lie on spheres on some other subparts. It was proposed there a technical way to achieve this but in the regime studied in the paper (decoding codes of constant rates at the Gilbert-Varshamov distance), this however did not yield any improvement on the time complexity over the original [BM18] algorithm. In fact Kévin Carrier, Jean-Pierre Tillich and

I tried independently of [Ess23] to achieve the same goal (embedding the naive filtering onto the NNS step) but tried it differently. Basically we started back with the LSH framework and built a locality-sensitive-hashing function which was tailored to this setting. Basically we concatenated some smaller LSH (one on each subpart) where each was tailored for the specific distribution of the corresponding subpart. For the subpart where the syndromes where distributed spherically we reused the LSH built in [Car20, §8.2.3] which essentially hash by decoding on some non-linear spherical code. Still, it seems that this does not give a better asymptotic time complexity exponent either.

More recently [FA25] noticed that [BM18] could be improved by reusing many times the lower level lists: conditioned on the fact that the bet on the first part of the syndrome $\mathbf{R}^{(1)}\mathbf{z} + \mathbf{s}^{(1)}$ one compute once the lists of the first level $\mathcal{L}_{1,1}$ and $\mathcal{L}_{1,2}$ and use them multiple time by randomizing the order of the rest of the rows at each iteration. This however, did not yield better time complexity exponent either but it improved time-memory tradoffs.

2.2 Dual attacks in coding theory: statistical decoding

Idea of dual decoders Say we are given a noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{e}$ is an error vector of known small weight t. Dual decoders, to recover the error vector $\mathbf{e}$, use the fact that for any dual vector $\mathbf{h} \in \mathcal{C}^\perp$ we know $\langle \mathbf{e}, \mathbf{h} \rangle$ as we have that

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{e}, \mathbf{h} \rangle . \tag{2.18}$$

Of course, recovering the error directly by linear algebra is not possible directly here as $\dim\left(\mathcal{C}^\perp\right) = n - k < n$. The important point is to notice that if $\mathbf{h}$ is sparse, then $\langle \mathbf{e}, \mathbf{h} \rangle$ is biased toward 0 as it is the inner product of two sparse vectors. The key observation being that it is essentially more biased toward 0 as the weight of $\mathbf{e}$ and $\mathbf{h}$ decreases. This can then be leveraged to decode.

The origin: Gallager LDPC decoder. Leveraging such ideas can be traced back, not for attacks, but in the search of a family of efficiently decodable codes by Gallager in 1963 [Gal63]. His idea was to construct a code containing extremely sparse, of Hamming weight $\mathcal{O}(1)$ dual vectors, namely a Low-Density Parity-Check (LDPC) which he would decode iteratively by flipping those positions which appear in sufficiently many unsatisfied low-weight parity-check equations. Forgetting about the iterative part of his algorithm one could restate his idea as follows. To recover say the first position of the error, e_1, it leverages the fact that a sparse dual vector $\mathbf{h}$ which is such that $h_1 = 1$ gives rise to a noisy version of e_1, namely

$$\langle \mathbf{y}, \mathbf{h} \rangle = e_1 + \left\langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \right\rangle \tag{2.19}$$

where it is readily seen that the noise, $\left\langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \right\rangle$ is biased toward 0 as $\mathbf{e}$ and $\mathbf{h}$ are sparse. It then recovers e_1 by majority voting by taking all the $\mathbf{h}$'s of weight $\mathcal{O}(1)$, and essentially choosing for e_1 whichever value of $\langle \mathbf{y}, \mathbf{h} \rangle$ appears the most. Note that one could think intuitively that the aforementioned lack of dimensionality somewhat does not a apply here as we are only trying to recover 1 position at a time.

2.2.1 The first dual attack: Statistical Decoding.

Later, in 2001 Al-Jabri [Jab01], used exactly this idea but to make an algorithm to decode generically linear codes, he called it statistical decoding. The difference here being that the

dual vectors of low weight must be computed and that they are a priori not as sparse as before, namely for a random code of constant rate we expect that even the shortest dual vector is of weight $\Omega(n)$. This automatically makes $\langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \rangle$ way less tilted toward 0, or to say it differently, its bias, $\varepsilon = \mathbb{P}\left(\langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \rangle = 0\right) - \mathbb{P}\left(\langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \rangle = 1\right)$, is smaller than in the LDPC decoding context. This gives the key insight on where the hardness of this attack comes from: the smaller this bias is, the noisier e_1 is, and more precisely, as we will see, the attack will require essentially a number $N = 1/\varepsilon^2$ of sparse dual vectors in order to recover e_1 with good probability. As we will see later, when decoding random linear code, this bias is exponentially small in the weight of the error, and, as such the decoder will require an exponential number of sparse dual vectors to decode. This algorithm was slightly improved (by polynomial factors) and analyzed non asymptotically by [Ove06]. Then, [FKI07] proposed an iterative version of this algorithm and analyzed it under a certain simple model. Then [DT17a] made a systematic and asymptotic analysis of Statistical decoding and showed that the simpler model made in [FKI07] could not hold.

Very interestingly statistical decoding completely departed from the traditional attacks, in the sense that it did not make any bet on the error like for ISD's. Also, the combinatorial techniques (collision, representation, ...) used by most ISD's are very efficient to decode/compute low weight codewords in the high regime rates, i.e. gaining a square root over Prange from the introduction of collisions by Dumer, but have little to no gain in the low regime rate, i.e. all the ISD's have the complexity of the Prange decoder. In reverse, it is readily seen that these techniques could be used to efficiently compute low weight dual vectors in the low regime rate, for example to produce an exponential number of them in amortized time $\text{poly}(n)$ [DT17a, §7.3]. As such, one could expect that one of the advantages of such dual attacks would be to gain in the low regime rate.

2.2.1.1 Poor performances of statistical decoding.

When the weight of the error is linear in n. In fact, it turns out that the above intuition is completely insufficient here, indeed as [DT17a, Figure 6] shows, even by considering a genie-aided variant of statistical decoding, that is if we neglect the cost of computing the dual vectors (the complexity of this Genie-aided variant is then simply $\widetilde{\mathcal{O}}(N)$, the number of dual vectors considered) is significantly outperformed asymptotically by even the Prange decoder, when decoding at the Gilbert-Varshamov distance, see Fig. 2.3.

When the weight of the error is sublinear in n. Suppose that we are decoding a code of rate R in the sublinear error weight regime, that is $t = o(n)$. [DT17a, Proposition 7] also analyzed statistical decoding, when a slight variation of the Prange decoder was used to compute the short dual vectors: it produces vectors of relative weight $R/2 + o(1)$ in time $\text{poly}(n)$. In that case, the complexity of the attack was also much worse than even the Prange ISD decoder, namely they got its square:

$$2^{-2\,t\,\log_2(1-R)(1+o(1))}. \tag{2.20}$$

Using anything else than this variation of Prange's decoder to compute lower weight dual vectors in this sublinear regime seems unfeasible since all the other methods which strictly beat this relative $R/2$ weight are all exponential in n. All in all [DT17a] concludes that it looks hard to find any relevant parameters regime where this algorithm would even beat the Prange decoder.

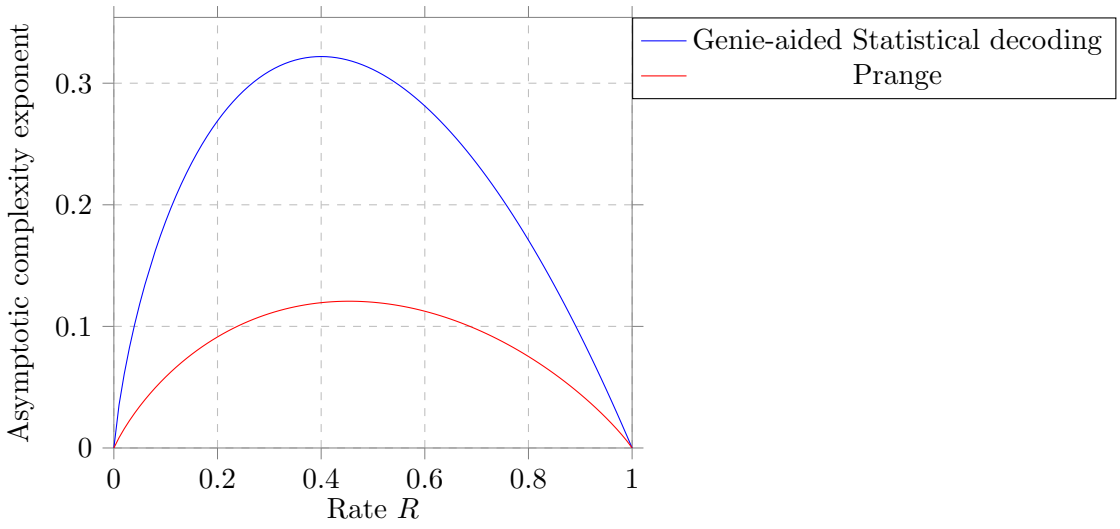


Figure 2.3: Asymptotic complexity exponent of the Prange decoder and the asymptotic complexity exponent (as it is defined in [DT17a, Definition 1] and given in [DT17a, Theorem 2]) of Genie-aided Statistical decoding when decoding a random code of rate R at the Gilbert-Varshamov distance.

A path to improvement as an open question. In [DT17a, Conclusion], it is suggested that instead of recovering only one position of the error at a time, one could consider multiple positions of the error at a time. This is precisely the key idea we will leverage in Chapter 5 to make a dramatic gain in the asymptotic exponent of the algorithm.

Before giving the key steps of the analysis of [DT17a] which yielded those results let us be more precise about what the algorithm does.

2.2.2 The statistical decoding algorithm

Repeat n times the following procedure to recover each position at a time. To recover the i'th position e_i compute a set $\mathscr{H}$ of N dual vectors $\mathbf{h}$ of low weight and such that $h_i = 1$. Then compute a score function F which encodes the number of times $\langle \mathbf{y}, \mathbf{h} \rangle$ is 0 or 1, namely

$$F \stackrel{\text{def}}{=} \sum_{\mathbf{h} \in \mathscr{H}} \langle \mathbf{y}, \mathbf{h} \rangle \tag{2.21}$$

where $\langle \mathbf{y}, \mathbf{h} \rangle$ is computed over $\mathbb{F}_2$ but the sum is computed over $\mathbb{Z}$. It then decides the value of e_i depending on if F is greater than a well chosen threshold T. For most parameters the decision will be that $e_i = 0$ if $F \geqslant T$, but it can be reversed in some degenerate parameters where $\langle \mathbf{e}_{\setminus i}, \mathbf{h}_{\setminus i} \rangle$ would in fact be biased toward 1 and not 0. The complexity of this algorithm is, up to a polynomial factor, the complexity of computing the N sparse dual vectors.

2.2.3 Analysis

Let us recap here the main steps of the asymptotic analysis of statistical decoding made by [DT17a], the non asymptotic part of this analysis can also be found in [Ove06]. Let us

focus here on recovering the first position e_1. The question really is how big must N be in order for the distinguisher to be able to make the right decision with good probability. The rough intuition coming from information theory is that the number of dual vector considered must be greater than $1/\varepsilon^2$ where ε is the bias of the noise $\langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \rangle$ (whatever this means distribution wise). Before telling precisely how this was achieved let us make two remarks coming from [DT17a].

Single dual weight w. It is sufficient, as long as we are only interested in the asymptotic exponent and not in the polynomial term in front of the complexity, to restrict ourselves to a single dual vector weight w. This is precisely what we will do in this thesis. The intuition behind this is that the bias of the noise will exponentially depend on $w \stackrel{\text{def}}{=} |\mathbf{h}|$ and thus a single weight will dominate the whole complexity.

Having sufficiently many dual vectors. Note that there is the natural constraint on N, the number of computed dual vectors, that it must be smaller than the number of available dual vectors of weight w,

$$N \leqslant \frac{\binom{n}{w}}{2^k} \tag{2.22}$$

which is the expected number for random codes. As such, two terms are competing: w must be sufficiently big such that the number of available dual vectors is superior to the number of needed dual vectors to distinguish, but this also increases with w. This is illustrated later in Fig. 2.4.

2.2.3.1 Distribution of the score function

The analysis is done in the natural framework where the procedure computing the dual vectors outputs a list $\mathscr{H}$ of N vectors $\mathbf{h}$ which are each taken uniformly and independently at random in $\mathcal{C}^\perp$ and such that $|\mathbf{h}|$ is weight w and $h_1 = 1$. It boils down to understanding the distribution of the score function which we recall is

$$\begin{aligned} F &\stackrel{\text{def}}{=} \sum_{\mathbf{h} \in \mathscr{H}} \langle \mathbf{y}, \mathbf{h} \rangle \\ &= \sum_{\mathbf{h} \in \mathscr{H}} \langle \mathbf{e}, \mathbf{h} \rangle \\ &= e_1 + \sum_{\mathbf{h} \in \mathscr{H}} \langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \rangle . \end{aligned}$$

Notice that a priori the distribution of each term in the sum deeply depends on the structure of the underlying code $\mathcal{C}$. As such, this leads [Ove06, DT17a] to make the following assumption to make their analysis tractable.

Assumption 1 (Distribution of the noise [DT17a, Asssumption 1].)**.** *The distribution of $\langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \rangle$ is approximated by the distribution of $\langle \mathbf{e}_{\setminus 1}, \mathbf{h}' \rangle$ when $\mathbf{h}'$ is taken uniformly at random in $\mathcal{S}_{w-1}^{n-1}$.*

This allows to make intervene the fundamental quantity $K_w^{(n)}(t)$, namely the Krawtchouk polynomial of order n and degree w which is related to the bias of $\langle \mathbf{e}, \mathbf{h} \rangle$ in the following way.

Proposition 16. *Let* $\mathbf{e} \in \mathcal{S}_t^n$. *Let* $\mathbf{h} \sim \mathcal{U}\left(\mathcal{S}_w^n\right)$. *We have that*

$$\mathbb{P}\left(\langle \mathbf{e}, \mathbf{h} \rangle = 1\right) = \frac{1 - \delta_w^{(n)}(t)}{2}, \qquad \text{where} \qquad \delta_w^{(n)}(t) \stackrel{def}{=} \frac{K_w^{(n)}(t)}{\binom{n}{w}}.$$

It can be written alternatively as:

$$\mathbb{E}\left((-1)^{\langle \mathbf{e}, \mathbf{h} \rangle}\right) = \delta_w^{(n)}(t).$$

Remark 7. *We make a quick survey regarding these polynomials in Section 2.2.4 which will serve a reference for the rest of this thesis.*

That is, under the previous assumption, and using the previous proposition with the fact that $\langle \mathbf{y}, \mathbf{h} \rangle = e_1 + \left\langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \right\rangle$ and that $\left|\mathbf{h}_{\setminus 1}\right| = t - e_1$ we get that

$$\mathbb{P}_{\mathbf{h}}\left(\langle \mathbf{y}, \mathbf{h} \rangle = 1\right) = \frac{1 + \delta_{w-1}^{(n-1)}(t-1)}{2}, \qquad \text{if } e_1 = 1,$$
$$\mathbb{P}_{\mathbf{h}}\left(\langle \mathbf{y}, \mathbf{h} \rangle = 1\right) = \frac{1 - \delta_{w-1}^{(n-1)}(t)}{2}, \qquad \text{if } e_1 = 0.$$

Still, this assumption is insufficient to say anything useful about the distribution of F as this deeply depends on the dependencies between each sample, as such they make this additionnal independence assumption.

Assumption 2 (Independence [DT17a, Asssumption 2])**.** $\left(\left\langle \mathbf{e}_{\setminus 1}, \mathbf{h}_{\setminus 1} \right\rangle\right)_{\mathbf{h} \in \mathscr{H}}$ *are mutually independent.*

All in all, this allows to model the score function as a Binomial distribution.

Model 1 (Model on the score function [Ove06, DT17a])**.** *The score function is modeled as follows:*

$$F|e_1 = 1 \sim \text{Binomial}\left(N,\ \frac{1 + \delta_{w-1}^{(n-1)}(t-1)}{2}\right)$$
$$F|e_1 = 0 \sim \text{Binomial}\left(N,\ \frac{1 - \delta_{w-1}^{(n-1)}(t)}{2}\right).$$

2.2.3.2 Concentration of the score function

Applying Chernoff's bound allows to show that as long as their respective expectations are some $\Omega\left(\sqrt{f(n)}\right)\sqrt{N}$ apart then we can distinguish both cases with probability $1 - e^{-\Omega(f(n))}$. Using the union bound, we can show that taking

$$N = \mathcal{O}\left(\frac{\log n}{\left(\delta_{w-1}^{(n-1)}(t) + \delta_{w-1}^{(n-1)}(t-1)\right)^2}\right) \tag{2.23}$$

allows recovering $\mathbf{e}$ with constant probability.

2.2.3.3 Asymptotic analysis

The goal here is to devise the asymptotic expansion of $\left(\delta_{w-1}^{(n-1)}(t)+\delta_{w-1}^{(n-1)}(t-1)\right)^2$. Recalling that $\delta_w^{(n)}(t) \stackrel{\text{def}}{=} \frac{K_w^{(n)}(t)}{\binom{n}{w}}$ and as we already know that $\binom{n}{w}=\widetilde{\mathcal{O}}\left(2^{h(w/n)n}\right)$, the asymptotic analysis of $\left(\delta_{w-1}^{(n-1)}(t)+\delta_{w-1}^{(n-1)}(t-1)\right)^2$ only boils down to devising the asymptotic expansion of $\left(K_{w-1}^{(n-1)}(t+1)+K_{w-1}^{(n-1)}(t+1)\right)^2$. The asymptotic expansion of $K_{w-1}^{(n-1)}(x)$ is known from [IS98, Theorem 3.1]: it can essentially be expressed as some $f(x)2^{g(x)}$ where f is poly bounded and can be positive, negative or null, and g is essentially positive and decreasing in $x \leqslant n/2$. Starting from this, [DT17a] estimates the asymptotic expansion of $\delta_{w-1}^{(n-1)}(t)+\delta_{w-1}^{(n-1)}(t-1)$ with the help of 3 quite technical statements [DT17a, Lemma 2,3,4]. They do it by adding the expansion of $\delta_{w-1}^{(n-1)}(t)$ and $\delta_{w-1}^{(n-1)}(t-1)$ and computing the result. The difficulty being that in certain regimes (when $w \geqslant n/2-\sqrt{t(n-t)}$) Krawtchouk polynomials oscillate and change sign leading to the possibility that $\delta_{w-1}^{(n-1)}(t-1)$ and $\delta_{w-1}^{(n-1)}(t)$ do not add up but cancel out (a sin appears in [DT17a, Lemma 4]). All in all, this leads them, to tackle this cancelling issue, to express N asymptotically as a limit inferior, essentially yielding that

$$\liminf N = -2\kappa(\tau,\ \omega)$$

where $\kappa(,\ )$ is defined later, in Eq. (2.34) and which we will see is in fact essentially and up to polynomial factors the asymptotic expansion of $\delta_w^{(n)}(t)$.

A surprising fact. Notably, when decoding a code of constant rate R at relative distance $\tau_{\mathrm{GV}}(\tau)$ the first relative weight ω where there are enough dual vectors to decode is at

$$\omega=\left[\frac{1}{2}-\sqrt{\tau_{\mathrm{GV}}(\tau)\left(1-\tau_{\mathrm{GV}}(\tau)\right)}\right](1+o(1))$$

which is around the first relative (to n) zero of $K_w^{(n)}$. The following figure Fig. 2.4 illustrates this behavior. Note that, as the figure shows, the fact that we observe that when ω increases further beyond the first intersection point (the first relative zero of $K_w^{(n)}$) the number of available dual vectors stays equal to the number of required dual vectors (i.e. we have no choices but to compute all the available dual vectors of relative weight ω) is predictable. The contrary (i.e. we have more dual vectors available than needed) would have meant that we could have decoded with good probability above the Gilbert-Varshamov distance: as we will see $\kappa(,\ )$ is continuous in both variables, as such any gap between the two curves would have meant that we could have increased the relative decoding distance τ above $\tau_{\mathrm{GV}}(R)$.

2.2.4 A short survey on Krawtchouk polynomials

A lot of the following propositions and definitions can be found in [KS21].

2.2.4.1 On the hypercube

Krawtchouck polynomials can simply be defined using Fourier theory.

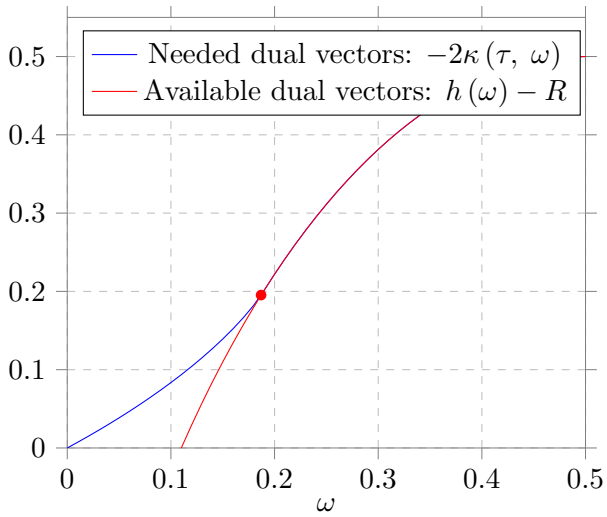


Figure 2.4: Asymptotic exponent, relative to the code length, of the number of needed dual vectors of relative weight ω against the number of available dual vectors when decoding a code of rate $R = 0.5$ at relative distance $\tau = \tau_{\mathrm{GV}}(R)$. The red dot is at $\omega = 1/2 - \sqrt{\tau(1-\tau)}$.

Definition 23 (Fourier transform)**.** *Let $f : \mathbb{F}_2^n \to \mathbb{R}$ be a real valued function. The Fourier transform $\widehat{f}$ of f is defined as the function*

$$\begin{array}{rccc} \widehat{f} & : & \mathbb{F}_2^n & \to & \mathbb{R} \\ & & \mathbf{y} & \mapsto & \sum_{\mathbf{x}\in\mathbb{F}_2^n} f(\mathbf{x})(-1)^{\langle \mathbf{y},\mathbf{x}\rangle} \end{array}$$

A Krawtchouk polynomial is exactly the Fourier transform of a Hamming sphere.

Definition 24 (Krawtchouk polynomial. First definition on $\mathbb{F}_2^n$)**.** *Let $n \in \mathbb{N}$ and let $t, w \in [\![0, n]\!]$. Let $\mathbf{x}$ be any vector of $\mathcal{S}_t^n$, we define the Krawtchouk polynomial of order n and degree w as*

$$K_w^{(n)}(t) \stackrel{def}{=} \widehat{\mathbf{1}_{\mathcal{S}_w^n}}(\mathbf{x}). \tag{2.24}$$

As this definition is independent of $\mathbf{x} \in \mathcal{S}_t^n$ we will also abusively define and use the notation

$$K_w^{(n)}(\mathbf{x}) \stackrel{def}{=} K_w^{(n)}(|\mathbf{x}|).$$

Notice that this is, up to a factor $\binom{n}{w}$, by definition, the bias of $\langle \mathbf{h}, \mathbf{x}\rangle$ when $\mathbf{x}$ is of weight t and $\mathbf{h}$ is taken uniformly at random in $\mathcal{S}_t^n$.

Definition 25. *Let $n, w, t \in \mathbb{N}$. We define*

$$\delta_w^{(n)}(t) \stackrel{def}{=} \frac{K_w^{(n)}(t)}{\binom{n}{w}}$$

Fact 12. *Let $\mathbf{e} \in \mathcal{S}_t^n$ be a fixed vector and $\mathbf{h} \sim \mathcal{U}(\mathcal{S}_w^n)$. We have that*

$$\mathbb{P}(\langle \mathbf{e}, \mathbf{h}\rangle = 1) = \frac{1 - \delta_w^{(n)}(t)}{2} \qquad \text{and} \qquad \mathbb{E}\left((-1)^{\langle \mathbf{e},\mathbf{h}\rangle}\right) = \delta_w^{(n)}(t).$$

Notably, Krawtchouck often appear in coding theory to relate the weight enumerator of a linear code with the weight enumerator of its dual with the celebrated MacWilliams identity.

Theorem 3 (MacWilliams identity [MS86, Theorem 1])**.** *Let $\mathcal{C}$ be an $[n,k]$-linear code. For any $w \in [\![0,\, n]\!]$ we have that*

$$N_w\left(\mathcal{C}^\perp\right) = \frac{1}{2^k}\sum_{i=0}^{n} N_i\left(\mathcal{C}\right) K_w^{(n)}\left(i\right)$$

where $N_i\left(\mathcal{C}\right)$ is the number of codeword of $\mathcal{C}$ of weight i.

This theorem is in fact a direct consequence of the Poisson summation formula and the definition of Kratwchouk polynomials.

Proposition 17 (Poisson summation formula [MS86, Lemma 11])**.** *Let $\mathcal{C}$ be an $[n,k]$-linear code and let $f : \mathbb{F}_2^n \to \mathbb{R}$ be a function. We have that*

$$\sum_{\mathbf{h}\in\mathcal{C}^\perp} f(\mathbf{h}) = \frac{1}{2^k}\sum_{\mathbf{c}\in\mathcal{C}} \widehat{f}(\mathbf{c}).$$

2.2.4.1.1 Basic properties The following basic properties can easily be derived from the definition and tells us that, when devising a property on Krawtchouk polynomials one can interchange the role of t and w at will and that it is enough to study them up until $t \leqslant \frac{n}{2}$.

Proposition 18 (Basic properties)**.** *We have that*

$$K_w^{(n)}\left(0\right) = \binom{n}{w}.$$

Symmetry

$$K_w^{(n)}\left(t\right) = (-1)^w K_w^{(n)}\left(n-t\right).$$

Reciprocity

$$\binom{n}{t} K_w^{(n)}\left(t\right) = \binom{n}{w} K_t^{(n)}\left(w\right).$$

This directly yield that the bias is symmetric.

Fact 13.

$$\delta_w^{(n)}\left(t\right) = \delta_t^{(n)}\left(w\right)$$

Importantly the family of polynomials $\left(K_0^{(n)},\, K_1^{(n)},\, \ldots,\, K_n^{(n)}\right)$ is orthogonal relative to the natural counting measure. This simply comes from Parseval identity along with the fact that $K_w^{(n)} = \widehat{\mathbf{1}_{\mathcal{S}_w^n}}$ to argue that the associated inner product is $\left\langle K_w^{(n)}, K_t^{(n)} \right\rangle = 2^n \left\langle \mathbf{1}_{\mathcal{S}_w^n}, \mathbf{1}_{\mathcal{S}_t^n} \right\rangle$.

Proposition 19 (Orthogonality of Krawtchouk polynomials)**.** *Let us define the finite measure $\mu\left(i\right)$ as*

$$\mu\left(i\right) \stackrel{def}{=} \binom{n}{i} \tag{2.25}$$

and the associated inner product $\langle , \rangle$ *for* f *and* g *two functions taking values from* $[\![0,\, n]\!]$*:*

$$\langle f, g \rangle \stackrel{def}{=} \sum_{i=0}^{n} \mu(i)\, f(i)\, g(i) \tag{2.26}$$

We have that

$$\left\langle K_w^{(n)}, K_t^{(n)} \right\rangle = \begin{cases} 0 & \text{if } w \neq t \\ 2^n \binom{n}{w} & \text{if } w = t. \end{cases} \tag{2.27}$$

Proof.

$$\begin{aligned} \left\langle K_w^{(n)}, K_t^{(n)} \right\rangle &= \sum_{\mathbf{x} \in \mathbb{F}_2^n} K_w^{(n)}(\mathbf{x})\, K_t^{(n)}(\mathbf{x}) \\ &= \sum_{\mathbf{x} \in \mathbb{F}_2^n} \sum_{\mathbf{h} \in \mathcal{S}_w^n} (-1)^{\langle \mathbf{h}, \mathbf{x} \rangle} \sum_{\mathbf{g} \in \mathcal{S}_t^n} (-1)^{\langle \mathbf{g}, \mathbf{x} \rangle} \\ &= \sum_{\mathbf{h} \in \mathcal{S}_w^n} \sum_{\mathbf{g} \in \mathcal{S}_t^n} \sum_{\mathbf{x} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{h}+\mathbf{g}, \mathbf{x} \rangle} \\ &= \sum_{\mathbf{h} \in \mathcal{S}_w^n} \sum_{\mathbf{g} \in \mathcal{S}_t^n} 2^n \mathbf{1}_{\mathbf{h}=\mathbf{g}} \\ &= \binom{w}{n} 2^n \mathbf{1}_{w=t}. \end{aligned}$$

□

Last, Krawtchouk polynomials have a lot of recurrence relations. We will use the two following in this thesis to easily compute the difference of biases.

Proposition 20. *Let* $n, w, t \in \mathbb{N}$ *such that* $1 \leqslant w, t \leqslant n$*. We have that*

$$\begin{aligned} K_w^{(n)}(t-1) &= K_w^{(n-1)}(t-1) + K_{w-1}^{(n-1)}(t-1), \\ K_w^{(n)}(t) &= K_w^{(n-1)}(t-1) - K_{w-1}^{(n-1)}(t-1). \end{aligned}$$

Proof. By Definition 24 of Krawtchouk polynomials for any $\mathbf{x} \in \mathcal{S}_v^n$ we have

$$\begin{aligned} K_w^{(n)}(v) &= K_w^{(n)}(\mathbf{x}) \\ &= \sum_{\mathbf{h} \in \mathcal{S}_w^n} (-1)^{\langle \mathbf{h}, \mathbf{x} \rangle}. \end{aligned}$$

By taking any $\mathbf{x}' \in \mathcal{S}_{t-1}^{n-1}$ and constructing $\mathbf{x} = (0 \,||\, \mathbf{x}') \in \mathcal{S}_{t-1}^n$ we get, by decomposing the previous sum on the values of $\mathbf{h}$ on the first position, that

$$K_w^{(n)}(t-1) = K_w^{(n-1)}(t-1) + K_{w-1}^{(n-1)}(t-1).$$

In the same manner, taking this time $\mathbf{x} = (1 \,||\, \mathbf{x}') \in \mathcal{S}_t^n$ we get

$$K_w^{(n)}(t) = K_w^{(n-1)}(t-1) - K_{w-1}^{(n-1)}(t-1).$$

□

2.2.4.2 As a real polynomial

Let us now give a more "computationally effective", alternate expression for $K_w^{(n)}$:

Proposition 21 (Krawtchouk polynomial. Second definition.)**.** *Let $w, n \in \mathbb{N}$ and let $x \in \mathbb{R}$. We define the Krawtchouk polynomial of degree w and order n as:*

$$\begin{array}{rccc} K_w^{(n)} & : & \mathbb{R} & \to & \mathbb{R} \\ & & x & \mapsto & \sum_{j=0}^{w} (-1)^j \binom{x}{j} \binom{n-j}{w-j}. \end{array} \tag{2.28}$$

This definition is coherent with Definition 24 on $[\![0,\, n]\!]$.

It is readily seen that

Fact 14. *$K_w^{(n)}$ is a real polynomial of degree w.*

In general, orthogonal polynomials of degree w have w distinct real roots, more precisely we have the following.

Proposition 22 (Roots [KS21, §2.2.5])**.** *Let w be a function of n. $K_w^{(n)}$ has w real valued roots $r_1 < \cdots < r_w$, all lying in the root region $[n/2 - \sqrt{w(n-w)},\ n/2 + \sqrt{w(n-w)}]$. Moreover,*

$$r_{i+1} - r_i = o(n)\,.$$

Definition 26. *We define the root region limit as*

$$\mathrm{Root}\left(K_w^{(n)}\right) \stackrel{def}{=} n/2 - \sqrt{w(n-w)}.$$

This root region profoundly delimits the behavior of the Krawtchouck polynomials, namely outside this root region it is monotonic, and more precisely decreasing in $[\![0, n/2 - \sqrt{w(1-w)}]\!]$, but inside the root region it oscillates while being unimodal between each root, for example see Fig. 2.5.

2.2.4.3 Asymptotic behavior

Lastly, let us state the asymptotic behavior of these polynomials. This has a central role when analyzing asymptotically dual attacks. First in the root region, it turns out that $K_w^{(n)}$ achieves its norm between each root, namely:

Proposition 23 (Behavior inside the root region [KS21, §2.2.6])**.** *For any $n,\ w \in \mathbb{N}$ such that $w \in [\![0,\, n]\!]$ there exist $w-1$ integers $t_i \in \mathbb{N}$, each in between the roots: $r_i < t_i < r_{i+1}$, where $K_w^{(n)}$ achieves point-wise its L^2 norm, more precisely:*

$$\binom{n}{t_i} K_w^{(n)}(t_i)^2 = \tilde{\Omega}_n(1)\ 2^n \binom{n}{w}. \tag{2.29}$$

Second, outside the root region we have

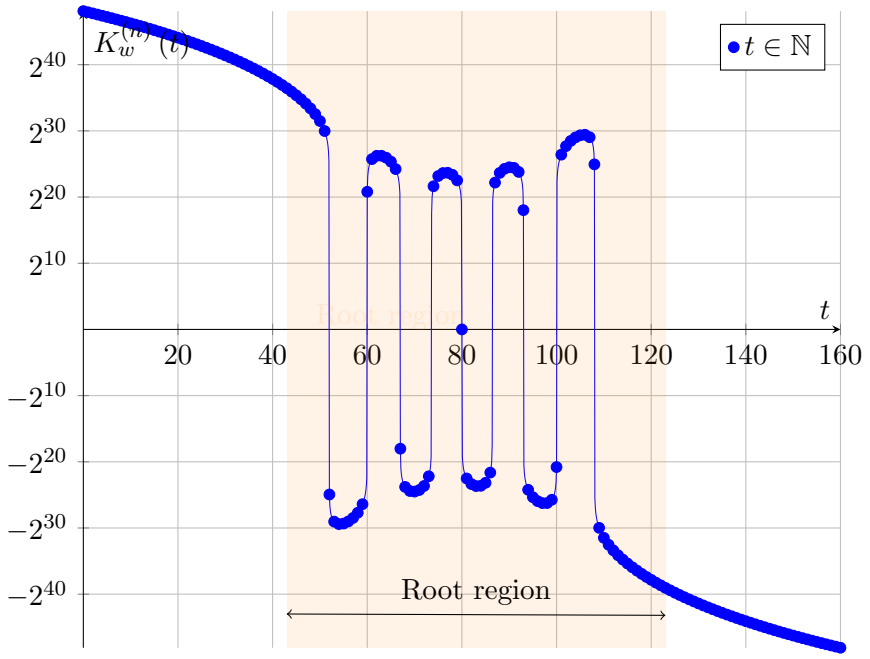


Figure 2.5: Symmetric logarithmic plot of $K_w^{(n)}$ with $w = 9$ and $n = 160$.

Proposition 24 (Behavior outside the root region [KS21, §2.2.7])**.** *For any* $t \leqslant \frac{n}{2} - \sqrt{w\,(n-w)}$ *we have that:*

$$K_w^{(n)}(t) = \tilde{\Omega}\left(2^{\gamma(\tau,\,\omega)\,n}\right) \tag{2.30}$$

where $\gamma(\tau, \omega)$ *is defined as follows for any* $\tau, \omega \in [0,\, \frac{1}{2}]$ *such that* $\tau \leqslant \omega^{\perp}$ *where* $\omega^{\perp} \stackrel{def}{=} \frac{1}{2} - \sqrt{\omega(1-\omega)}$.
Let $z(\omega, \tau) \stackrel{def}{=} \frac{1-2\tau-\sqrt{D(\omega,\tau)}}{2(1-\omega)}$ *where* $D(\omega, \tau) \stackrel{def}{=} (1-2\tau)^2 - 4\omega(1-\omega)$ *we define*

$$\gamma(\omega, \tau) \stackrel{def}{=} \tau \log_2(1 - z(\omega, \tau)) + (1-\tau)\log_2(1 + z(\omega, \tau)) - \omega \log_2(z(\omega, \tau)).$$

A simple proof of a weaker version (where it is supposed that w and t grow linearly with n) of both previous propositions can be found [KL95, §IV]. Let us define the function $\widetilde{\kappa}(,)$ that captures the asymptotic behavior of $\frac{\log_2\left|K_w^{(n)}(t)\right|}{n}$, or, to say it more precisely of the points t which verifies either Eq. (2.29) or Eq. (2.30), namely

Definition 27. *Let* $\omega, \tau \in [0,\, 1]$*. We define* $\widetilde{\kappa}(\omega,\, \tau)$ *as:*

$$\widetilde{\kappa}(\omega,\, \tau) = \begin{cases} \gamma(\omega,\, \tau) & \text{if } \tau < \omega^{\perp} \\ \frac{1+h(\omega)-h(\tau)}{2} & \text{else} \end{cases} \tag{2.31}$$

As shown by [KS21, §2.1.2] this function continuous in both variables and, fixing ω, is decreasing and concave for $\tau < \omega^{\perp}$ and decreasing and convex for $\tau \in [\omega^{\perp},\, 1/2]$, we illustrate it in Fig. 2.6.

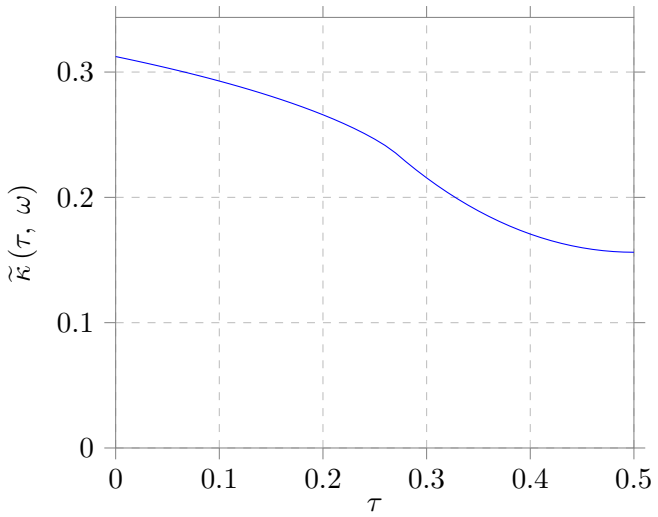


Figure 2.6: Plot of $\widetilde{\kappa}(\omega,\ \tau)$ as a function of τ with $\omega = 0.05625$.

Asymptotic properties of Krawtchouk polynomials

Proposition 25 ([KS21, §2.1.2])**.** *$\widetilde{\kappa}(,\ )$ is continuous and differentiable with*

$$\frac{\partial \widetilde{\kappa}(\omega,\ \tau)}{\partial \tau} = \begin{cases} \log_2(r(\omega,\tau)) & \text{if } \tau < \omega^{\perp} \\ \frac{1}{2}\log_2\left(\frac{\tau}{1-\tau}\right) & \text{if } \omega^{\perp} < \tau < \frac{1}{2} \end{cases}$$

where

$$r(\omega,\tau) = \frac{1-2\omega+\sqrt{(1-2\omega)^2 - 4\tau(1-`\tau)}}{2(1-\tau)}.$$

Moreover, fixing ω, it is decreasing in $0 \leqslant \tau \leqslant 1/2$.

Proposition 26 ([KS21, Lemma 2.3])**.** *For $\omega, \tau \in [0,\ 1/2]$ we have that*

$$h(\tau) + \widetilde{\kappa}(\omega,\ \tau) = h(\omega) + \widetilde{\kappa}(\tau,\ \omega)$$

Corollary 4. *We have that*

$$\frac{\partial \widetilde{\kappa}(\omega,\ \tau)}{\partial \omega} = \begin{cases} -\log_2\left(\frac{\omega}{1-\omega}\right) + \log_2(r(\tau,\omega)) & \text{if } \tau < \omega^{\perp} \\ -\frac{1}{2}\log_2\left(\frac{\omega}{1-\omega}\right) & \text{if } \omega^{\perp} < \tau < \frac{1}{2} \end{cases}$$

where $r(\omega,\tau)$ is defined in Proposition 25.

2.2.4.4 Asymptotic expansion of the key bias

Finally, we are now able to give the asymptotic expansion of the bias $\delta_w^{(n)}(t) \stackrel{\text{def}}{=} \frac{K_w^{(n)}(t)}{\binom{w}{n}}$ using the two propositions giving the asymptotic behavior of Krawtchouk polynomial inside and outside the root region. This will be used in all this thesis.

Corollary 5 (Asymptotic expansion of the bias)**.** *Let $n \in \mathbb{N}$ be growing to infinity and let $\tau,\ \omega \in]0,\ 1/2[$ be implicit functions of n and define $t \stackrel{def}{=} \lfloor \tau\, n \rfloor$. We have that*

1. *If $\tau < \omega^{\perp}$ then by defining $w \stackrel{def}{=} \lfloor \omega n \rfloor$ we have that*

$$\left|\delta_w^{(n)}(t)\right| = \tilde{\Omega}\left(2^{\kappa(\omega,\,\tau)\,n}\right) \tag{2.32}$$

   where we add furthermore that the $\tilde{\Omega}()$ does not depend on τ and ω.

2. *If $\tau \geqslant \omega^{\perp}$ there exists $w = \omega\, n + o(n)$ such that*

$$\left|\delta_w^{(n)}(t)\right| = 2^{(\kappa(\omega,\,\tau)+o(1))\,n}, \tag{2.33}$$

where

$$\kappa(\omega,\ \tau) \stackrel{def}{=} \widetilde{\kappa}(\omega,\ \tau) - h(\omega) \tag{2.34}$$

and where $\widetilde{\kappa}(,\ )$ is defined in Definition 27.

Remark 8. *More formally the first point really say that there exists a positive function f such that for any two functions $\tau,\ \omega : \mathbb{N} \to [0, 1/2]$ we have that for any $n \in \mathbb{N}$ that*

$$\left|\delta_{\lfloor \omega n \rfloor}^{(n)}(\lfloor \tau n \rfloor)\right| \geqslant \frac{1}{f(n)} 2^{\kappa(\omega,\,\tau)\,n}.$$

Proof. The first point is straightforward from Proposition 24. The second point comes by using the Krawtchouk reciprocity rule Proposition 18 along with the fact that two consecutive roots are $o(n)$ distanced (Proposition 22) along with Proposition 23 telling us that Krawtchouk polynomials achieve their norms between consecutive roots. □

Now, using the expression of the derivative of Krawtchouk polynomials given above we can devise easily the shape of the bias $\delta_w^{(n)}(t)$ when t is sublinear in n.

Proposition 27 (Asymptotic expansion of the bias in the sublinear regime)**.** *Let $\tau = o(1)$ and let $\omega \in [0,\ 1/2[$ be a constant. We have that*

$$\kappa(\omega,\ \tau) = \tau \log_2(1 - 2\omega) + o(\tau). \tag{2.35}$$

Proof. From Proposition 25 we have that

$$\frac{\partial \kappa(\omega,\ \tau)}{\partial \tau} = \log_2\left(\frac{1 - 2\omega + \sqrt{(1-2\omega)^2 - 4\tau(1-\tau)}}{2(1-\tau)}\right)$$

Using Taylor's theorem in τ we get that:

$$\begin{aligned}\kappa(\omega,\ \tau) &= \kappa(\omega,\ 0) + \tau\ \log_2\left(\frac{1 - 2\omega + \sqrt{(1-2\omega)^2}}{2}\right) + o(\tau)\\ &= \tau\ \log_2(1-2\omega) + o(\tau)\end{aligned}$$

□

Chapter 3

The LPN problem

Summary

In this very short chapter, we present the LPN problem and, in a non-exhaustive manner, introduce two of the algorithms used to solve it. We will later use these ideas to build our dual attack.

Contents

3.1 Definition

The LPN problem is a learning problem whose hardness underlies many post-quantum cryptographic constructions. To name a few, these include encryption schemes [Ale03, DP12, DMN12, DV13], the HB authentication protocol family, and its extensions for constructing symmetric-key message authentication codes (MACs) [HB01, JW05, GRS08, KPC+11, HKL+12]. It is a special case of the LWE problem in lattice-based cryptography [Reg05]. The complexity of the best-known LPN solvers are used to determine the parameters of the schemes built on it. Essentially, in this problem, given access to an oracle that outputs, upon each call, a noisy linear combination of a secret, the goal is to recover the secret using as many queries as needed.

Definition 28 (LPN oracle $\mathcal{O}(\mathbf{s},\ \varepsilon)$ and LPN problem $\mathrm{LPN}(k,\ \varepsilon)$)**.** *For any* $\mathbf{s} \in \mathbb{F}_2^k$ *and* $\varepsilon \in [0,\ 1]$ *we define* $\mathcal{O}(\mathbf{s},\ \varepsilon)$ *the oracle that when called outputs*

$$(\mathbf{g},\ \langle \mathbf{s}, \mathbf{g} \rangle + e) \qquad \textit{where} \begin{cases} \mathbf{g} \sim \mathcal{U}\left(\mathbb{F}_2^k\right) \\ e \sim \text{Bernoulli}\left(\frac{1-\varepsilon}{2}\right) \\ \mathbf{g} \textit{ and } e \textit{ are independent} \end{cases}.$$

We sometimes refer to $\tau \stackrel{def}{=} \frac{1-\varepsilon}{2}$ *as the noise level of the oracle and* ε *as its bias. We define the LPN problem* $LPN(k,\ \varepsilon)$ *of dimension* k *and bias* ε *as the problem where a secret* $\mathbf{s}$ *is chosen uniformly at random in* $\mathbb{F}_2^k$ *and you are given access to* $\mathcal{O}(\mathbf{s}, \varepsilon)$ *with the goal of recovering* $\mathbf{s}$ *from this oracle.*

Clearly, after having generated N samples $(\mathbf{g}^{(i)}, b^{(i)})$, solving the problem really is solving a decoding problem of length N and dimension k given by the following generator matrix and noisy codeword:

$$\mathbf{G} \stackrel{\text{def}}{=} \left(\left(\mathbf{g}^{(1)}\right)^\intercal \cdots \left(\mathbf{g}^{(N)}\right)^\intercal\right) \quad \text{and} \quad \mathbf{y} \stackrel{\text{def}}{=} \left(b^{(1)} \cdots b^{(N)}\right) \tag{3.1}$$

The difference with $\mathrm{DP}_{\mathbf{G}}(N, k, t)$ is that the noise is not of fixed weight t. More precisely we are interested in the following decoding problem.

Definition 29. *Decoding problem* $\mathrm{DP}_{\mathbf{G}}^{\mathrm{Ber}}(N, k, \varepsilon)$*. Given* $\mathcal{C}$ *and* $\mathbf{y}$ *where* $\mathcal{C}$ *is chosen by taking its generator matrix* $\mathbf{G} \sim \mathcal{U}\left(\mathbb{F}_2^{k \times N}\right)$ *and* $\mathbf{y} = \mathbf{c} + \mathbf{e}$ *where* $\mathbf{c} = \mathbf{s}\mathbf{G}$ *and* $\mathbf{s} \sim \mathcal{U}\left(\mathbb{F}_2^k\right)$ *and the coordinates of* $\mathbf{e}$ *are distributed i.i.d according to* $\text{Bernoulli}\left(\frac{1-\varepsilon}{2}\right)$*. The goal is to recover* $\mathbf{s}$*.*

From Shannon's second theorem for linear codes, see [RU08, Th. 4.68, P. 203], it is possible to have a simple condition for being able to recover the secret.

Fact 15. *As long as* $N = \Omega\left(s/\varepsilon^2\right)$ *then it is possible to solve* $\mathrm{DP}_{\mathbf{G}}^{\mathrm{Ber}}(N, k, \varepsilon)$ *with probability* $1 - o(1)$*. More precisely we have that* $\mathbf{s} = \arg\min_{\mathbf{m} \in \mathbb{F}_2^k} |\mathbf{y} - \mathbf{m}\mathbf{G}|$ *with probability* $1 - o(1)$*.*

Basically the theorem states that as long as the rate of the code s/N is below the channel capacity C then we can recover the secret with good probability. Here the channel considered is a binary symmetric channel with crossover probability $\frac{1-\varepsilon}{2}$ whose capacity C is given by $1 - h(1-\varepsilon) = \Omega\left(\varepsilon^2\right)$. This gives our fact.

Coming back to the LPN problem, because we have access to an unlimited amount of samples, solving it really is solving $\mathrm{DP}_{\mathbf{G}}^{\mathrm{Ber}}(N, k, \varepsilon)$ but with N arbitrarily large. Basically this

means that we can have access to many more samples than what is information-theoretically needed to recover the secret with good probability. This makes the problem much easier than the standard decoding problem $\mathrm{DP}_{\mathbf{H}}(n,k,t)$ at constant rate and constant relative error weight. For example, when the bias ε is constant, all the existing algorithms for solving the LPN problem, starting from BKW [BKW03], run in slightly subexponential time

$$2^{\Theta(k/\log(k))}.$$

This is also the number of required queries to the oracle for the BKW algorithm, or to say differently the length of the final code. The security of all the LPN based cryptographic constructions that we are aware of are based on a regime where the noise $\tau = \left(\frac{1-\varepsilon}{2}\right)$ is relatively low, either constant, or growing to zero: the problem is still conjectured to be viable for cryptography for noises as little as $\tau = 1/n^c$ for any constant $0 < c < 1$ [YS16]. In that case however, the best attacks are based on a Prange bet [EKM17] coming from ISD's. This essentially allows beating BKW in the low noise regime $\tau = o(1/\log_2(k))$. We do not focus on those last solvers in this exposition.

3.2 Algorithms for solving LPN

In the LPN world in the mild noise regime, say constant, the dominant family solvers are given by improvements over BKW [BKW03, LF06, GJL20, BTV15, KF15, BV16, ZJW16]. All these techniques share the characteristic that they reduce to another LPN problem of smaller dimension but increased noise. Note that all these algorithms require a certain different number N of LPN samples which is determined when analyzing the algorithm so without loss of generality we can suppose that it is fixed.

We take some liberty in our exposition, we refer the reader to the original algorithm for precision, our goal here is to explain the following three points.

- Present quickly BKW and highlight its differences and similarities with statistical decoding.
- Present the idea behind the FFT solver used in [LF06].
- Present the idea behind the so-called covering code technique of [GJL20].

3.2.1 The BKW dimenion reduction procedure

Basically the idea of the BKW reduction is to reduce solving LPN $(k,\ \varepsilon)$ to solving LPN $(k-k',\ \varepsilon')$ where the dimension $k-k'$ is reduced but the bias $\varepsilon' < \varepsilon$ has also decreased, i.e. the noise has increased. This dimension reduction is done by generating a batch of samples $(\mathbf{g}^{(i)},\ \langle \mathbf{s}, \mathbf{g}^{(i)} \rangle + e_i)$ and then by searching for $\mathbf{g}^{(i)}$'s whose sum is zero on some fixed coordinates, say $\sum_{i\in\mathcal{I}} \mathbf{g}^{(i)}$ is zero on the last k' coordinates. Summing the corresponding sample yields a new sample

$$\left(\sum_{i\in\mathcal{I}} \mathbf{g}^{(i)},\ \left\langle \mathbf{s}, \sum_{i\in\mathcal{I}} \mathbf{g}^{(i)} \right\rangle + \sum_{i\in\mathcal{I}} e_i\right)$$

involving only the $k-k'$ first coordinates of the secret but with larger noise. This is exactly distributed as the output of the oracle $\mathcal{O}(\mathbf{s}_{[\![1,\,k-k']\!]},\ \varepsilon')$ where the bias ε' of the new noise

$\sum_{i\in\mathcal{I}} e_i$ is readily given by

$$\varepsilon' = \varepsilon^{|\mathscr{I}|}.$$

The above equation is a direct consequence of the Piling-up lemma.

Lemma 8 (Piling-up lemma)**.** *Let* $e_1 \sim \text{Bernouilli}\left(\frac{1-\varepsilon_1}{2}\right)$ *and* $e_1 \sim \text{Bernouilli}\left(\frac{1-\varepsilon_2}{2}\right)$ *be two independent Bernoulli variables of bias* ε_1 *and* ε_2 *respectively. Then, the sum of the two variables has bias* $\varepsilon_1\varepsilon_2$*, namely*

$$e_1 + e_2 \sim \text{Bernouilli}\left(\frac{1-\varepsilon_1\varepsilon_2}{2}\right).$$

These sums are obtained with a collision technique that cancels the vectors block by block in a progressive manner. Basically, we start by writing $k' = a \times b$ for two integers a and b and split the last k' coordinates of the vectors into a blocks where each block is composed of b coordinates. In the first step, we make 2^b groups, each group contains the $\mathbf{g}^{(i)}$ that are equal the first block. In each group we choose arbitrarily a vector and sum it to all the other vectors in the group and then discard this vector. This allows cancelling the first block. We repeat this process to progressively cancelling the other blocks.

If we started with M samples we essentially end up with $M - a2^b$ new samples (because of at each step we lost 2^b samples) of dimension $k - k'$, but now, because those samples are the sum of (at most) 2^{2^a} original samples the new bias is (at minimum) ε^{2^a}. Note that these new samples are not completely independent.

3.2.2 The BKW Algorithm

In the original BKW algorithm the problem is reduced to a dimension 1 LPN problem which is then solved by majority voting. What we present here is not exactly what is done in the original article. Say we are in the usual constant noise regime, namely $\varepsilon = \Theta(1)$. Taking $k' = k - 1$ and $a = \frac{1}{2}\log_2 k'$ and $b = 2k'/\log_2 k'$ and $M = a2^b + 1 = 2^{\mathcal{O}(k/\log_2 k)}$ allows us to produce essentially 1 LPN samples of dimension 1 with secret s_1 and where the bias of the noise is (at minimum) $\varepsilon^{\sqrt{k}} = 2^{\Theta(\sqrt{k})}$ in time $2^{\mathcal{O}(k/\log_2 k)}$ by using $2^{\mathcal{O}(k/\log_2 k)}$ calls to the original LPN oracle. We can simply iterate this procedure $2^{\Theta(\sqrt{k})}$ times each time with fresh new samples to produce enough independent reduces samples to be able to recover s_1 with good probability. The overall complexity of this attack is $2^{\mathcal{O}(k/\log_2 k)}$ in time and $N = 2^{\mathcal{O}(k/\log_2 k)}$ in query complexity to the oracle.

3.2.2.1 Link and difference with statistical decoding

Slightly abstracting the BKW algorithm, one can notice that this really is a dual attack. In essence BKW computes low-weight dual vectors $\mathbf{h}$ of the code generated by $\mathbf{G} \in \mathbb{F}_2^{k\times N}$ but with the first row removed. Each dual vector gives the following reduced LPN sample:

$$\begin{aligned}\langle \mathbf{y}, \mathbf{h}\rangle &= \langle \mathbf{c} + \mathbf{e}, \mathbf{h}\rangle \\ &= \langle \mathbf{sG}, \mathbf{h}\rangle + \langle \mathbf{e}, \mathbf{h}\rangle \\ &= \langle \mathbf{s}, \mathbf{hG}^\intercal\rangle + \langle \mathbf{e}, \mathbf{h}\rangle \\ &= \langle s_1, (\mathbf{hG}^\intercal)_1\rangle + \langle \mathbf{e}, \mathbf{h}\rangle .\end{aligned}$$

Here the support of $\mathbf{h}$ is basically the set $\mathscr{I}$ presented in the previous section and represent which samples we select. The secret is given by s_1 and the noise $\langle \mathbf{e}, \mathbf{h} \rangle$ increases with the noise rate of the error $\mathbf{e}$ and the weight of the dual vector. The secret s_1 is recovered by majority voting, and, as long as we have computed sufficiently many sparse dual vectors $\mathbf{h}$ we can make the right decision with good probability.

Here, unlike statistical decoding that was studied in the constant rate regime, for LPN, the length of the code can get arbitrarily long. As such we have access to much lower weight dual vectors that we can compute efficiently. This makes $\langle \mathbf{e}, \mathbf{h} \rangle$ significantly less noisy than what we had in statistical decoding. BKW benefits from this. Additionally, in the original BKW proposal (and the illustrative solver we gave), all those created dual vectors have disjoint support. This makes the analysis tractable as in that case the variables $\langle \mathbf{e}, \mathbf{h} \rangle$ are mutually independent (hence, we do not need here Assumption 2 as needed in the analysis of statistical decoding). This is loosened in LF1 and LF2 [LF06] to save some samples but it makes the analysis heuristic.

3.2.3 Reducing to a higher dimension for free with an FFT

BKW was later improved in [LF06] by using an FFT technique that can be traced back to [Gre66] for decoding Reed-Muller codes.

Proposition 28. *Let $N, k \in \mathbb{N}$. There exists a procedure that for any $[N, k]$-linear code $\mathcal{C}$ with generator matrix $\mathbf{G} \in \mathbb{F}_2^{k \times N}$ and any $\mathbf{y} \in \mathbb{F}_2^N$ outputs for all $\mathbf{m} \in \mathbb{F}_2^k$ the value of $|\mathbf{mG} + \mathbf{y}|$ in time and memory*

$$\mathcal{O}\Big(\max\Big(k2^k,\ N\Big)\Big).$$

In particular this allows us to find the codeword of $\mathcal{C}$ which is the closest to $\mathbf{y}$.

We give more details about this procedure and the proof of this statement in Section 3.2.3.1. The idea is to notice that reducing to an LPN problem of dimension one as is the case in BKW is suboptimal. Indeed, clearly, the number N' of reduced LPN samples $\langle s_1, (\mathbf{hG}^\intercal)_1 \rangle + \langle \mathbf{e}, \mathbf{h} \rangle$ that we need (say $N' = 2^{\Theta(k)}$ from the previous section) to recover s_1 is well above $k'2^{k'}$ where $k' = 1$ is the dimension of this reduced problem. So basically we are deep into a regime where the FFT decoder cost is dominated just by the length N' of the code. Consequently, to balance the costs, we have an interest in reducing to a higher dimension LPN problem by computing dual vectors in a lower dimensional subcode, this increases k' but decreases N', the number of required samples to solve the reduced problem, as now we are able to compute dual vectors of lower weight.

3.2.3.1 An FFT based decoder for codes of very small rate

This decoder basically computes the distance between $\mathbf{y}$ and all the codewords of the code $\mathbf{c} \in \mathcal{C}$ using a Fourier transform as described in the next standard proposition.

Proposition 29 (Key proposition)**.** *Let $\mathbf{G} \in \mathbb{F}_2^{k \times N}$ and $\mathbf{y} \in \mathbb{F}_2^N$. Define the function f that, for each $\mathbf{a} \in \mathbb{F}_2^k$ associates*

$$f(\mathbf{a}) \stackrel{def}{=} \sum_{i \in [\![1,\, N]\!] \,:\, \mathbf{G}_i = \mathbf{a}} (-1)^{y_i}.$$

We have that the Fourier transform $\widehat{f}$ of f verifies:

$$\forall \mathbf{m} \in \mathbb{F}_2^k \qquad \widehat{f}(\mathbf{m}) = n - 2\,|\mathbf{y} - \mathbf{mG}|\,.$$

Proof.

$$\begin{aligned}\widehat{f}(\mathbf{x}) &\stackrel{\text{def}}{=} \sum_{\mathbf{r}\in\mathbb{F}_2^s} f(\mathbf{r})(-1)^{\langle \mathbf{r},\mathbf{x}\rangle} \\ &= \sum_{\mathbf{a}\in\mathbb{F}_2^k} \sum_{i\in[\![1,N]\!]\,:\,\mathbf{G}_i=\mathbf{a}} (-1)^{y_i-\langle \mathbf{a},\mathbf{x}\rangle} \\ &= \sum_{i=1}^{N} (-1)^{y_i-\langle \mathbf{G}_i,\mathbf{x}\rangle} \\ &= n - 2\left|\mathbf{y}-\mathbf{mG}\right|.\end{aligned}$$

□

Finally, using a standard Fast Fourier Transform (FFT) these distances can be computed in time $\mathcal{O}\left(k2^k\right)$.

Proposition 30 (Fast Fourier Transform [CT65])**.** *There exists a procedure a Fast Fourier Transform* FFT *which takes as input a function $f:\mathbb{F}_2^k\to\mathbb{R}$ and outputs its Fourier transform $\widehat{f}$ in time and memory*

$$\mathcal{O}\left(k2^k\right)$$

Proof. We only give the outline of this standard proof. For $b\in\mathbb{F}_2$, let us define the function $f_b:\mathbb{F}_2^{n-1}\to\mathbb{R}$ as $f_b(\mathbf{r})=f((b\,||\,\mathbf{r}))$. The FFT$(f)$ procedure uses the following recursion formula

$$\begin{aligned}\widehat{f}(\mathbf{x}) &\stackrel{\text{def}}{=} \sum_{\mathbf{r}\in\mathbb{F}_2^n} f(\mathbf{r})(-1)^{\langle \mathbf{r},\mathbf{x}\rangle} \\ &= \sum_{\mathbf{r}\in\mathbb{F}_2^{n-1}} f((0\,||\,\mathbf{r}))(-1)^{\langle \mathbf{r},\mathbf{x}_{\setminus 1}\rangle} + (-1)^{x_1}\sum_{\mathbf{r}\in\mathbb{F}_2^{n-1}} f((1\,||\,\mathbf{r}))(-1)^{\langle \mathbf{r},\mathbf{x}_{\setminus 1}\rangle} \\ &= \widehat{f}_0\left(\mathbf{x}_{\setminus 1}\right) + (-1)^{x_1}\widehat{f}_1\left(\mathbf{x}_{\setminus 1}\right)\end{aligned}$$

to compute $\widehat{f}(\mathbf{x})$ in batch. □

Algorithm 10 An LPN solver using the FFT trick of [LF06]

Name: FFT-LPN-Solver
Input: $\mathcal{L}$ ▷ *$\mathcal{L}$ is a list of samples of the form $(\mathbf{a},b)$ where $\mathbf{a}\in\mathbb{F}_2^k$ and $b\in\mathbb{F}_2$*
1: Compute for each $\mathbf{a}\in\mathbb{F}_2^k$
$$f^{\mathcal{L}}(\mathbf{a}) = \sum_{\substack{(\mathbf{g},b)\in\mathcal{L}\\ \mathbf{g}=\mathbf{a}}} (-1)^b$$
2: $F^{\mathcal{L}}\leftarrow \text{FFT}\left(f^{\mathcal{L}}\right)$
3: **return** $F^{\mathcal{L}}$

3.2.4 The covering code technique

The main technical novelty in [GJL14] lies in a method, that does not involve combining samples, to reduce the dimension of an LPN problem whose secret $\mathbf{s}$ is of low Hamming weight at a cost of an increase in the noise (growing with the dimension loss and the secret weight). We present this reduction in Section 3.2.4.1. The crux is that there exists a standard reduction to make the coordinates of the secret distributed as the error, this reduction does not increase the noise but simply requires that we accept loosing a few samples, say k if we want to completely replace the distribution of the secret. In a setting where the noise is constant this makes the secret sparse. We recall the reduction here.

Fact 16 (Corollary of Fact 11)**.** *Let $\mathcal{C}$ be an $[N, k]$-linear code of generator matrix $\mathbf{G}$ and let $\mathbf{y} = \mathbf{sG} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{C}$. Provided that $[\![k+1,\ N]\!]$ is an information set of $\mathcal{C}^\perp$ we have that*

$$\mathbf{y}_{[\![k+1,\,N]\!]} + \mathbf{y}_{[\![1,\,k]\!]}\mathbf{R} = \left(\mathbf{e}_{[\![1,\,k]\!]}\right)\mathbf{R} + \mathbf{e}_{[\![k+1,\,N]\!]}$$

where $\mathbf{R} \stackrel{def}{=} \text{Lift}\left(\mathcal{C}^\perp,\ [\![k+1,\ N]\!]\right)$. We recall that $\mathbf{R} \in \mathbb{F}_2^{k\times(N-k)}$ is the unique matrix such that $\mathbf{h}_{[\![1,\,k]\!]} = \mathbf{R}\left(\mathbf{h}_{[\![k+1,\,N]\!]}\right)$.

Remark 9. *The new secret is given by $\left(\mathbf{e}_{[\![1,\,k]\!]}\right)$.*

Solving this new problem trivially allows us to solve the original one. In practice however [GJL14] does not need to completely replace the secret distribution but rather say only make the first s positions of the secret sparse (this can be done at the cost of s samples: the first s positions of the secret becomes $\mathbf{e}_{[\![1,\,s]\!]}$, the rest is unchanged). So, from these mixed shaped samples they start to apply some BKW reduction steps to reduce to an LPN problem of dimension s and whose secret is $\mathbf{e}_{[\![1,\,s]\!]}$ and then apply the following reduction using the sparseness of the secret to reduce the dimension even more.

3.2.4.1 Reducing the dimension of a sparse LPN problem

Say we have access to an oracle giving access to the following sample

$$(\mathbf{g},\ \langle \mathbf{s}, \mathbf{g}\rangle + e)$$

where the secret $\mathbf{s} \in \mathbb{F}_2^k$ is of low Hamming weight and the error e is taken according to a certain Bernouilli distribution. Note that this reduction is oblivious to the fact that the LPN sample can come from an original LPN problem or come from say an LPN obtained with a BKW type reduction. The idea is to approximate $\mathbf{g}$ into a lower dimension subspace. Say we have an auxiliary $[k, k_{\text{aux}}]$ linear code $\mathcal{C}_{\text{aux}}$ and we decode $\mathbf{g}$ onto $\mathcal{C}_{\text{aux}}$ to obtain

$$\mathbf{g} = \mathbf{c}_{\text{aux}} + \mathbf{e}_{\text{aux}}, \qquad \text{where } \mathbf{c}_{\text{aux}} \in \mathcal{C}_{\text{aux}} \text{ and } |\mathbf{e}_{\text{aux}}| \text{ is low}$$

Now clearly this allows to rewrite our sample as

$$\begin{aligned} \langle \mathbf{s}, \mathbf{g}\rangle + e &= \langle \mathbf{s}, \mathbf{c}_{\text{aux}} + \mathbf{e}_{\text{aux}}\rangle + e && (3.2)\\ &= \langle \mathbf{s}, \mathbf{c}_{\text{aux}}\rangle + \langle \mathbf{s}, \mathbf{e}_{\text{aux}}\rangle + e && (3.3)\end{aligned}$$

Importantly, notice that we have now a linear combination of the secret $\mathbf{s}$ with some codeword $\mathbf{c}_{\text{aux}}$ lying in a lower dimension subspace, this can be leveraged by simply taking a generator marix $\mathbf{G}_{\text{aux}}$ of $\mathcal{C}_{\text{aux}}$ and writing that $\mathbf{c}_{\text{aux}} = \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}}$ to obtain that

$$\langle \mathbf{s}, \mathbf{c}_{\text{aux}}\rangle + \langle \mathbf{s}, \mathbf{e}_{\text{aux}}\rangle + e = \langle \mathbf{s}\mathbf{G}_{\text{aux}}^\intercal, \mathbf{m}_{\text{aux}}\rangle + \langle \mathbf{s}, \mathbf{e}_{\text{aux}}\rangle + e.$$

Hence we in fact have access to the following sample

$$(\mathbf{m}_{\text{aux}},\ \langle \mathbf{s}, \mathbf{g}\rangle + e) \qquad = \qquad \left(\underbrace{\mathbf{m}_{\text{aux}}}_{\in \mathbb{F}_2^{k_{\text{aux}}}},\ \left\langle \mathbf{m}_{\text{aux}}, \underbrace{\mathbf{s}\mathbf{G}_{\text{aux}}^{\mathsf{T}}}_{\text{secret}} \right\rangle + \underbrace{\langle \mathbf{e}_{\text{aux}}, \mathbf{s}\rangle + e}_{\text{noise}} \right).$$

Clearly the dimension of the LPN sample has decreased from k to k_{aux} but now we have an additional noise term $\langle \mathbf{e}_{\text{aux}}, \mathbf{s}\rangle$ which clearly increases with the dimension loss: at best if we decoded $\mathbf{g}$ onto the closest codeword $\mathbf{c}_{\text{aux}}$ of $\mathbf{c}_{\text{aux}}$ we can expect that $\mathbf{e}_{\text{aux}}$ is of the order $d_{\text{GV}}(k,\ k_{\text{aux}})$.

3.2.4.2 Rest of the algorithm

In practice after the reduction an FFT is performed to solve this last LPN problem and if the number of samples is right they expect to be able to distinguish the compressed secret $\mathbf{s}\mathbf{G}_{\text{aux}}^{\mathsf{T}}$ and argue that they can easily recover the secret from there. An additional guessing phase is present in the algorithm that we did not present here. Overall this gave a profitable tradeoff between the BKW reduction step and this new code based reduction step.

It was proposed there to choose good codes as auxiliary codes $\mathcal{C}_{\text{aux}}$, say perfect codes and precompute once and for all a syndrome table that is used to decode. More precisely, considering $\mathbf{H}$ a parity-check matrix of $\mathcal{C}_{\text{aux}}$, store in a table $(\mathbf{H}\mathbf{e}_{\text{aux}},\ \mathbf{e}_{\text{aux}})$ for all the errors $\mathbf{e}_{\text{aux}} \in \mathbb{F}_2^k$. Each sample $\mathbf{g}$ is then decoded by computing its syndrome $\mathbf{H}\mathbf{g}$ and returning the associated $\mathbf{e}_{\text{aux}}$ of lowest weight. For efficiency and as a practical construction, it was proposed there to construct $\mathcal{C}_{\text{aux}}$ as a juxtaposition (i.e. the Cartesian product) of smaller good codes. The rationale is that the syndrome table of each code can be enumerated independently and that provided that we took enough codes, we can make this enumeration step negligible in front of the other costs of the algorithm.

Part II

Contributions

Chapter 4

Proof of a variant of statistical decoding

Summary

The contribution of this short chapter is to prove, without using any assumptions (see the traditional Assumption 1 and Assumption 2), a variant of Statistical decoding against the problem of distinguishing between a noisy codeword and a uniformly random word in the space. We do this by devising a second-order concentration bound on the score function that encodes the bias of the main quantity underlying dual attacks. We obtain, up to polynomial factors, the same results as if we had used the traditional assumptions. We discuss the procedures that can be used to compute the low-weight dual vectors and their output distribution. We turn this distinguisher into a decoder with a standard reduction, we will use this provable dual attack in the next chapter. Finally, we propose a simplification of the asymptotic analysis of statistical decoding given by [DT17a].

Contents

4.1 A simple dual distinguisher

4.1.1 The distinguishing problem

In this chapter we devise a simple dual attack against the problem of distinguishing a noisy codeword from a uniformly distributed word of the space.

Definition 30 (Distinguishing problem $\mathrm{DiP}_{\mathbf{G}}(n,k,t)$)**.** *Given a couple $(\mathcal{C},\mathbf{y})$ where $\mathcal{C}\sim\mathcal{U}_{\mathbf{G}}(n,\ k)$, namely $\mathcal{C}$ is given as a generator matrix $\mathbf{G}$ that was taken uniformly at random in $\mathbb{F}_2^{k\times n}$ and $\mathbf{y}$ is chosen according to the uniform word distribution $\mathcal{D}_0$ with probability $1/2$ or according to the noisy codeword distribution $\mathcal{D}_1$ with probability $1/2$, the goal is to output $i\in\{0,1\}$ and the decision is good if $\mathbf{y}$ was chosen according to $\mathcal{D}_i$.*

- $\mathcal{D}_0=\mathcal{U}(\mathbb{F}_2^n)$.
- $\mathcal{D}_1$ *outputs* $\mathbf{y}=\mathbf{c}+\mathbf{e}$ *where* $\mathbf{c}\sim\mathcal{U}(\mathcal{C})$ *and* $\mathbf{e}\sim\mathcal{U}(\mathcal{S}_t^n)$.

Remark 10. *Note that we consider the case when $\mathcal{C}$ is taken by choosing its generator matrix uniformly at random rather than its parity-check matrix as it makes the analysis of our dual attacks slightly easier. But this is only a very minor tweak compared to the previous setting, see Remark 3 for more details.*

This distinguishing problem is of importance as there are many standard polynomial reductions from decoding to distinguishing, we give such a reduction in Section 4.4.

4.1.2 Dual distinguisher

We recall that the base observation behind dual attacks in general is that when $\mathbf{y}=\mathbf{c}+\mathbf{e}$ is a noisy codeword, a sparse (low Hamming weight) dual vector $\mathbf{h}\in\mathcal{C}^\perp$ yields a linear combination of the error $\mathbf{e}$, namely

$$\langle\mathbf{y},\mathbf{h}\rangle=\langle\mathbf{e},\mathbf{h}\rangle$$

that is essentially more biased toward 0 (equal to 0 more often than it is equal to 1) as the weight of $\mathbf{e}$ and $\mathbf{h}$ decreases. This can be leveraged to decode $\mathbf{y}$ or in our case distinguish whether $\mathbf{y}$ is a noisy codeword or uniformly random. Indeed, in the latter case, one can notice that, because the positions of $\mathbf{y}$ are taken at random in $\mathbb{F}_2^n$ the random variable $\langle\mathbf{y},\mathbf{h}\rangle$ is not biased, i.e. is uniform in $\mathbb{F}_2$. Now we can increase this small advantage by computing many dual vectors and make a decision based on the number of times $\langle\mathbf{y},\mathbf{h}\rangle$ is 0 or 1. This is done by encoding this number in a score function F (that we define for convenience slightly differently but equivalently from the definition given in Statistical decoding, see Eq. (2.21)).

Definition 31 (Score function)**.** *Let $\mathbf{y}\in\mathbb{F}_2^n$ and let $\mathscr{H}\subset\mathbb{F}_2^n$. Define*

$$F_{\mathscr{H}}(\mathbf{y})\stackrel{def}{=}\sum_{\mathbf{h}\in\mathscr{H}}(-1)^{\langle\mathbf{y},\mathbf{h}\rangle}$$

or simply F when the context is clear.

Fact 17. *Let $\mathcal{C}$ be a linear code of length n and let $\mathscr{H}\subset\mathcal{C}^\perp\bigcap\mathcal{S}_w^n$ and let $\mathbf{y}=\mathbf{c}+\mathbf{e}$ with $\mathbf{c}\in\mathcal{C}$. We have that*

$$F_{\mathscr{H}}(\mathbf{y})=F_{\mathscr{H}}(\mathbf{e})=|\mathscr{H}|\ \mathrm{bias}_{\mathbf{h}\sim\mathcal{U}(\mathscr{H})}(\langle\mathbf{e},\mathbf{h}\rangle).$$

Clearly, we expect that $F_{\mathscr{H}}(\mathbf{y})$ is 0 when $\mathbf{y}$ is taken uniformly at random in $\mathbb{F}_2^n$ while we expect that $F_{\mathscr{H}}(\mathbf{y})$ is greater when $\mathbf{y}$ is a noisy codeword. More particularly $F_{\mathscr{H}}$ essentially grows as the weight of $\mathbf{e}$ and $\mathbf{h}$ decreases and as the size of $\mathscr{H}$ increases. Basically our distinguisher starts by computing a subset $\mathscr{H}$ of dual vectors of small weight and makes a decision upon the value of $F_{\mathscr{H}}(\mathbf{y})$, say by deciding that we are in the noisy codeword case if the score function is above a well-chosen threshold T. The general algorithm is given in Algorithm 11.

Algorithm 11 Dual-Distinguisher

Name: DUAL-DISTINGUISHER
Input: $\mathcal{C}$, $\mathbf{y} \in \mathbb{F}_2^n$
Parameter: $T \in \mathbb{N}$ (A threshold)
Require: Procedure COMPUTE-DUAL-VECTORS($\mathcal{C}$) outputting a set of small vectors of $\mathcal{C}^\perp$.
Output: 1 if we decide that $\mathbf{y}$ is a noisy codeword and 0 otherwise

```
1: H ← COMPUTE-DUAL-VECTORS(C)
2: if |F_H(y)| > T then
3:     return 1                          ▷ Decide y is a noisy codeword
4: else
5:     return 0                          ▷ Decide y is uniform
```

Now, the analysis of this distinguisher relies on the estimation of the value of the score function in each case. This of course depends on the output distribution of the procedure computing the low-weight dual vectors. The main tool here are concentration bounds on the score function around its expectation in each case. The natural choice for the threshold T is then the middle point between the expectations in each case.

4.2 Analysis when all the dual vectors of a certain weight are computed

We make the analysis when $\mathscr{H}$ is the set of all dual vectors of weight w. We recall in Section 4.3 some procedures that naturally have this output distribution but this is interesting independently to study the genie-aided performance of the distinguisher (i.e. the complexity but overlooking the cost of computing the dual vectors).

In that case we expect that the number N of dual vectors available is as follows.

Lemma 9 (Expected number of available dual vectors.)**.** *Let $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n, k)$ and $\mathscr{H} \stackrel{def}{=} \mathcal{C}^\perp \bigcap \mathcal{S}_w^n$, we have that*

$$\mathbb{E}(|\mathscr{H}|) = \frac{\binom{n}{w}}{2^k}.$$

Proof. This is a direct corollary of the standard Lemma 2 and Fact 5. □

4.2.1 Second-order concentration bounds and result

We recall, in this setting, the bias $\delta_w^{(n)}(t)$ naturally intervenes, as, forgetting about the code structure, we have the following.

Fact 18. *Let* $\mathbf{e} \in \mathcal{S}_t^n$ *and let* $\mathbf{h} \sim \mathcal{U}(\mathcal{S}_w^n)$*. The bias of* $\langle \mathbf{e}, \mathbf{h} \rangle$ *is*

$$\mathbb{E}\left((-1)^{\langle \mathbf{e}, \mathbf{h} \rangle}\right) = \delta_w^{(n)}(t)$$

where $\delta_w^{(n)}(t)$ *is defined in Definition 25.*

Our main tool for the analysis of the dual distinguisher is the following second-order concentration bound on the score function that involves the two previous quantities. We obtain it by computing the expected value and the variance of the score function and by applying Bienaymé–Chebyshev inequality.

Proposition 31 (Second-order concentration bounds on the score function)**.** *Let* $n, k, t, w \in \mathbb{N}$ *and let* f *be a positive function. Let* $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\ k)$ *and* $\mathscr{H} \stackrel{def}{=} \mathcal{C}^\perp \bigcap \mathcal{S}_w^n$ *and define* $N \stackrel{def}{=} \binom{n}{w}/2^k$*. We have that*

$$\mathbb{P}\left(\left|F_{\mathscr{H}}(\mathbf{e}) - N\delta_w^{(n)}(t)\right| \geqslant f(n)\sqrt{N}\right) \leqslant \frac{1}{f(n)} \qquad \text{if } \mathbf{e} \in \mathcal{S}_t^n.$$

$$\mathbb{P}\left(|F_{\mathscr{H}}(\mathbf{y})| \geqslant f(n)\sqrt{N}\right) \leqslant \frac{1}{f(n)} \qquad \text{if } \mathbf{y} \sim \mathcal{U}(\mathbb{F}_2^n).$$

Moreover $N\delta_w^{(n)}(t)$ *and* 0 *are the respective expected values of* $F_{\mathscr{H}}(\mathbf{e})$ *and* $F_{\mathscr{H}}(\mathbf{y})$ *and finally* N *is an upper bound on the variance of* $F_{\mathscr{H}}$ *in both cases.*

This allows us to show that as long as

$$N \geqslant f(n)^2/\delta_w^{(n)}(t)^2$$

then we can distinguish the uniform case from the noisy codeword case with probability $1 - f(n)/2$. In particular, taking $f(n) = \sqrt{\log_2(n)}$ we can state the following theorem giving the performance of our dual distinguisher.

Theorem 4 (Correctness of Algorithm 11 when we compute all the dual vectors of a certain weight)**.** *For all* $k, t, w \in \mathbb{N}$ *implicit functions of* $n \in \mathbb{N}$ *and any procedure* Compute-Dual-Vectors *that are such that*

$$\mathbb{P}_{\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\ k)}\left(\textsc{Compute-Dual-Vectors}(\mathcal{C}) = \mathcal{C}^\perp \bigcap \mathcal{S}_w^n\right) = 1 - o(1)$$

and

$$\frac{\binom{n}{w}}{2^k} \geqslant \frac{\log_2(n)}{\delta_w^{(n)}(t)^2}$$

then, taking $T \stackrel{def}{=} \frac{1}{2}\frac{\binom{n}{w}}{2^k}\delta_w^{(n)}(t)$*, Algorithm 11 solves* $\mathrm{DiP}_{\mathbf{G}}(n, k, t)$ *with probability* $1 - o(1)$ *in time and memory that are, up to polynomial factors, the time and memory complexities of the* Compute-Dual-Vectors *procedure.*

Remark 11. *Note that we state the previous theorem when the procedure computing the dual vectors is only asked to produce the whole set only with probability* $1 - o(1)$*. We do this so that there actually exists an efficient procedure that matches this requirement, see for example the Dumer's collision procedure Proposition 13.*

Proof. This is a corollary of the second-order concentration bounds given in Proposition 31. □

4.2.2 Discussion

Note that here we reasoned on the score function, but alternatively we could have reasoned on the bias of $\langle \mathbf{y}, \mathbf{h} \rangle$. Notably, one could reformulate those second-order concentration bounds to show that as long as

$$\frac{\binom{n}{w}}{2^k} = \frac{\omega(1)}{\delta_w^{(n)}(t)^2}$$

then, in the noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$ setting, we have, with probability $1 - o(1)$ over the choice of $\mathcal{C}$, that

$$\mathrm{bias}_{\mathbf{h} \sim \mathcal{U}\left(\mathcal{C}^\perp \bigcap \mathcal{S}_w^n\right)}\left(\langle \mathbf{h}, \mathbf{e} \rangle\right) = \delta_w^{(n)}(t)\left(1 + o(1)\right).$$

In a sense our result matches the traditional learning result that if we have access to N independent realizations of either the uniform distribution over $\mathbb{F}_2$ or a Bernoulli distribution with bias $\delta_w^{(n)}(t)$ then we need around $N = \omega(1)/\delta_w^{(n)}(t)^2$ samples to distinguish each case with probability $1 - o(1)$.

Clearly, if we had used the independence assumptions, see Assumption 2, used in [DT17a] and that assume that the terms in the sum appearing in the score function are independent, we would have been able to show with the Chernoff bound that as long as $N \geqslant f(n)^2/\delta_w^{(n)}(t)^2$ then we can distinguish with probability $1 - e^{-\Omega(f(n))}$ whereas without the assumption we can only show with our second order concentration bound that we succeed distinguishing with probability $1 - \mathcal{O}(1/f(n))$. This means that we only lose a polynomial factor as long as we want to distinguish with a poly-bounded advantage, this is sufficient for our use cases here but will be insufficient in the next chapter where this distinguisher will be invoked an exponential number of times, this will require, as we show a careful conjecture on the exponential tail behavior of the score function that we will devise in the next chapter, in Section 5.4.

4.2.3 Asymptotic analysis

We give here the asymptotic counterpart of Theorem 4. The goal here is to prove that given asymptotic parameters R, τ, ω verifying that $h(\omega) - R \geqslant -2\,\kappa(\omega,\ \tau)$ there exists an infinite sequence of closely related non-asymptotic parameters k, t, w that verify the non-asymptotic constraints of Theorem 4, namely that $\binom{n}{w}/2^k = (\log_2 n)/\delta_w^{(n)}(t)^2$. Indeed, we recall that $\kappa(\omega,\ \tau)$ is the asymptotic expansion of the bias $\delta_w^{(n)}(t)$ as defined in Corollary 5. The slight technicality comes from the fact that the asymptotic constraints do not account for polynomial factors. The whole point being that w should be close to ωn as it directly appears in the complexity of the procedure computing the dual vectors (which we did not state here). To simplify the statement we restrict the statement to the case where t is outside the root region of $K_w^{(n)}$, namely when $\omega < \tau^\perp \stackrel{\text{def}}{=} \frac{1}{2} - \sqrt{\tau(1-\tau)}$, this allows for the bias to achieve its expansion up to polynomial factors at any point, see Corollary 5. This constraint is very naturally verified by any sound parameters, see Fig. 2.4.

Theorem 5 (Asymptotic counterpart of Theorem 4)**.** *There exists g, a function of $n \in \mathbb{N}$, such that $g(n) \in \mathcal{O}((\log_2 n)/n)$ and such that for any implicit functions of n, $R \in]0, 1[$, $\tau,\ \omega \in]0,\ 1/2[$ that are such that*

$$h(\omega) - R \geqslant -2\,\kappa(\omega,\ \tau) \ \text{ and } \omega < \tau^\perp - g$$

then by defining $k \stackrel{def}{=} \lfloor Rn \rfloor$ *and* $t \stackrel{def}{=} \lfloor \tau n \rfloor$ *there exists* w *an implicit function of* n *that is such that* $w - \omega n \in \mathcal{O}(\log_2 n)$ *and such that provided that*

$$\mathbb{P}_{\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\, k)} \left(\textsc{Compute-Dual-Vectors}(\mathcal{C}) = \mathcal{C}^{\perp} \bigcap \mathcal{S}_w^n \right) = 1 - o(1)$$

and taking $T \stackrel{def}{=} (1/2)(\binom{n}{w}/2^k)\delta_w^{(n)}(t)$*, Algorithm 11 solves* $\mathrm{DP}_{\mathbf{G}}(n, k, t)$ *with probability* $1 - o(1)$*. We recall that* $\kappa(,\,)$ *is the asymptotic expansion of the bias given in Corollary 5.*

Remark 12. *We could state a similar theorem without the constraint that* $\omega < \tau^{\perp} - g$ *but then we would only be able to show that* w *is in* $w - \omega n \in o(n)$ *by using the fact that a Krawtchouk polynomial achieves its asymptotic expansion in the root region, up to a correction factor of* $o(n)$*, as given by Corollary 5.*

Proof of Theorem 5. We only have to show that there exists w and implicit function of n such that $\frac{\binom{n}{w}}{2^k} = \Omega\left(\frac{\log_2 n}{\delta_w^{(n)}(t)^2}\right)$ and such that $w - \omega n \in \mathcal{O}(\log_2 n)$. For any w we have that

$$\begin{aligned}\frac{\binom{n}{w}}{2^k}\delta_w^{(n)}(t)^2 &= \frac{K_w^{(n)}(t)^2}{\binom{n}{w}2^k} \\ &= \frac{\binom{n}{w}K_t^{(n)}(w)^2}{\binom{n}{t}^2 2^k}\end{aligned}$$

It is readily seen that using Taylor's theorem along with the following Lemma 10 allows us to prove our result. □

Lemma 10. *Let us fix* $\tau \in [0,\, 1/2]$*. Consider the function which associates* ω *to*

$$h(\omega) + 2\,\widetilde{\kappa}(\tau,\, \omega)\,.$$

For any $\omega \in]0,\, 1/2]$ *such that* $\tau < \omega^{\perp}$ *this function is differentiable and has a positive derivative. We recall that* $\widetilde{\kappa}(,\,)$ *is the asymptotic expansion of Krawtchouk polynomials given in Definition 27.*

Proof. Let us compute the derivative of this function of ω which we call ϕ. We only have to show that, for any values of $\tau \in [0,\, 1/2]$, we have that $\frac{\partial \phi(\omega)}{\partial \omega} > 0$. From Proposition 25 we have that

$$\frac{\partial \widetilde{\kappa}(\tau,\, \omega)}{\partial \omega} = \log_2\left(\frac{1 - 2\tau + \sqrt{(1-2\tau)^2 - 4\omega(1 - \text{`}\omega)}}{2\,(1-\omega)}\right).$$

And we can show that

$$\frac{\partial h(\omega)}{\partial \omega} = -\log_2\left(\frac{\omega}{1-\omega}\right).$$

As such

$$\frac{\partial \phi(\omega)}{\partial \omega} = \log_2\left(\frac{1 - 2\tau + \sqrt{(1-2\tau)^2 - 4\omega(1 - \text{`}\omega)}}{2\omega}\right). \tag{4.1}$$

Now, by hypothesis we have that $\tau < \omega^{\perp}$, namely

$$\tau < \frac{1}{2} - \sqrt{\omega\left(1-\omega\right)}.$$

Thus

$$1 - 2\tau > 2\sqrt{\omega\left(1-\omega\right)}.$$

Plugging this in Eq. (4.1) we directly get that

$$\begin{aligned}\frac{\partial\phi\left(\omega\right)}{\partial\omega} &> \log_2\left(\frac{2\sqrt{\omega\left(1-\omega\right)}}{2\omega}\right)\\ &> \log_2\left(\sqrt{\frac{1-\omega}{\omega}}\right).\end{aligned}$$

This concludes the proof as $\omega \leqslant 1/2$ we have that $\frac{1-\omega}{\omega} \geqslant 1$. □

4.2.4 Proof of the concentration bound

Here we prove our second order-concentration bound on the score function given in Proposition 31. We recall the proposition here.

Proposition 31 (Second-order concentration bounds on the score function)**.** *Let $n, k, t, w \in \mathbb{N}$ and let f be a positive function. Let $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}\left(n,\ k\right)$ and $\mathscr{H} \stackrel{def}{=} \mathcal{C}^{\perp} \bigcap \mathcal{S}_w^n$ and define $N \stackrel{def}{=} \binom{n}{w}/2^k$. We have that*

$$\begin{aligned}\mathbb{P}\left(\left|F_{\mathscr{H}}\left(\mathbf{e}\right) - N\delta_w^{(n)}\left(t\right)\right| \geqslant f(n)\sqrt{N}\right) &\leqslant \frac{1}{f(n)} && \text{if } \mathbf{e} \in \mathcal{S}_t^n.\\ \mathbb{P}\left(\left|F_{\mathscr{H}}\left(\mathbf{y}\right)\right| \geqslant f(n)\sqrt{N}\right) &\leqslant \frac{1}{f(n)} && \text{if } \mathbf{y} \sim \mathcal{U}\left(\mathbb{F}_2^n\right).\end{aligned}$$

Moreover $N\delta_w^{(n)}\left(t\right)$ and 0 are the respective expected values of $F_{\mathscr{H}}\left(\mathbf{e}\right)$ and $F_{\mathscr{H}}\left(\mathbf{y}\right)$ and finally N is an upper bound on the variance of $F_{\mathscr{H}}$ in both cases.

This is done by computing the expected value and the variance of the score function and applying Bienaymé-Chebyshev inequality.

Lemma 11. *Using the notations of Proposition 31 we have that*

$$\begin{aligned}\mathbb{E}\left(F_{\mathscr{H}}\left(\mathbf{y}\right)\right) &= 0, && \text{if } \mathbf{y} \sim \mathcal{U}\left(\mathbb{F}_2^n\right)\\ \mathbb{E}\left(F_{\mathscr{H}}\left(\mathbf{e}\right)\right) &= N\,\delta_w^{(n)}\left(t\right), && \text{if } \mathbf{e} \in \mathcal{S}_t^n.\end{aligned}$$

Proof of Lemma 11. We only compute the value of $\mathbb{E}\left(F_{\mathscr{H}}\left(\mathbf{e}\right)\right)$ here. The two expectations

are computed in the same way. Using the linearity of expectation we have that

$$\begin{aligned}\mathbb{E}\left(F_{\mathscr{H}}\left(\mathbf{e}\right)\right) &= \sum_{\mathbf{h}\in\mathcal{S}_w^n}\left(-1\right)^{\langle\mathbf{e},\mathbf{h}\rangle}\mathbb{E}\left(\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right)\\ &= \sum_{\mathbf{h}\in\mathcal{S}_w^n}\left(-1\right)^{\langle\mathbf{e},\mathbf{h}\rangle}\mathbb{P}\left(\mathbf{1}_{\mathbf{h}\in\mathcal{C}^{\perp}\bigcap\mathcal{S}_w^n}\right)\\ &= \sum_{\mathbf{h}\in\mathcal{S}_w^n}\left(-1\right)^{\langle\mathbf{e},\mathbf{h}\rangle}\frac{1}{2^k}\\ &= \frac{1}{2^k}K_w^{(n)}\left(t\right)\\ &= N\delta_w^{(n)}\left(t\right).\end{aligned}$$

□

Lemma 12. *Using the notations of Proposition 31, we have that:*

$$\begin{aligned}&\mathbf{Var}\left(F_{\mathscr{H}}\left(\mathbf{e}\right)\right)\leqslant N && \text{if } \mathbf{e}\in\mathcal{S}_w^n\\ &\mathbf{Var}\left(F_{\mathscr{H}}\left(\mathbf{y}\right)\right)\leqslant N && \text{if } \mathbf{y}\sim\mathcal{U}\left(\mathbb{F}_2^n\right).\end{aligned}$$

Proof. Let us start with the computation of $\mathbf{Var}\left(F_{\mathscr{H}}\left(\mathbf{e}\right)\right)$. We have that

$$\begin{aligned}\mathbf{Var}\left(F\left(\mathbf{e}\right)\right) &= \mathbf{Var}\left(\sum_{\mathbf{h}\in\mathcal{S}_w^n}\left(-1\right)^{\langle\mathbf{e},\mathbf{h}\rangle}\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right)\\ &= \sum_{\mathbf{h}\in\mathcal{S}_w^n}\mathbf{Var}\left(\left(-1\right)^{\langle\mathbf{e},\mathbf{h}\rangle}\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right)+2\sum_{\substack{\mathbf{h},\mathbf{g}\in\mathcal{S}_w^n\\ \mathbf{h}\neq\mathbf{g}}}\mathbf{Cov}\left(\left(-1\right)^{\langle\mathbf{e},\mathbf{h}\rangle}\mathbf{1}_{\mathbf{h}\in\mathscr{H}},\ \left(-1\right)^{\langle\mathbf{e},\mathbf{g}\rangle}\mathbf{1}_{\mathbf{g}\in\mathscr{H}},\ \right)\end{aligned} \tag{4.2}$$

Let us compute both sums. For the first sum we have that

$$\begin{aligned}\sum_{\mathbf{h}\in\mathcal{S}_w^n}\mathbf{Var}\left(\left(-1\right)^{\langle\mathbf{e},\mathbf{h}\rangle}\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right) &= \sum_{\mathbf{h}\in\mathcal{S}_w^n}\mathbf{Var}\left(\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right) && (4.3)\\ &\leqslant \sum_{\mathbf{h}\in\mathcal{S}_w^n}\mathbb{E}\left(\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right) && (4.4)\\ &= \mathbb{E}\left(\left|\mathscr{H}\right|\right) && (4.5)\\ &= N && (4.6)\end{aligned}$$

where the last equality comes from Lemma 9. Let us now compute the second sum. We have that

$$\sum_{\substack{\mathbf{h},\mathbf{g}\in\mathcal{S}_w^n\\ \mathbf{h}\neq\mathbf{g}}}\mathbf{Cov}\left(\left(-1\right)^{\langle\mathbf{e},\mathbf{h}\rangle}\mathbf{1}_{\mathbf{h}\in\mathscr{H}},\ \left(-1\right)^{\langle\mathbf{e},\mathbf{g}\rangle}\mathbf{1}_{\mathbf{g}\in\mathscr{H}},\ \right)=\sum_{\mathbf{h}\in\mathcal{S}_w^n}\left(-1\right)^{\langle\mathbf{e},\mathbf{h}\rangle}\sum_{\mathbf{g}\in\mathcal{S}_w^n\,:\,\mathbf{g}\neq\mathbf{h}}\left(-1\right)^{\langle\mathbf{e},\mathbf{g}\rangle}\mathbf{Cov}\left(\mathbf{1}_{\mathbf{h}\in\mathscr{H}},\ \mathbf{1}_{\mathbf{g}\in\mathscr{H}},\ \right)$$

From Proposition 5 we get that

$$\begin{aligned}\mathbf{Cov}\left(\mathbf{1}_{\mathbf{h}\in\mathscr{H}},\ \mathbf{1}_{\mathbf{g}\in\mathscr{H}},\ \right) &= \mathbb{P}\left(\mathbf{h}\in\mathscr{H},\ \mathbf{g}\in\mathscr{H}\right)-\mathbb{P}\left(\mathbf{h}\in\mathscr{H}\right)\mathbb{P}\left(\mathbf{g}\in\mathscr{H}\right)\\ &= 0\end{aligned}$$

Thus this second sum is 0 and we have our result for $\mathbf{Var}\left(F\left(\mathbf{e}\right)\right)$.

Let us now compute the case for $\mathbf{Var}\left(F_{\mathscr{H}}\left(\mathbf{y}\right)\right)$. We have in the same manner that

$$\mathbf{Var}\left(F\left(\mathbf{y}\right)\right) = \sum_{\mathbf{h}\in\mathcal{S}_w^n} \mathbf{Var}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\,\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right) + 2 \sum_{\substack{\mathbf{h},\mathbf{g}\in\mathcal{S}_w^n \\ \mathbf{h}\neq\mathbf{g}}} \mathbf{Cov}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\,\mathbf{1}_{\mathbf{h}\in\mathscr{H}},\ (-1)^{\langle \mathbf{y},\mathbf{g}\rangle}\,\mathbf{1}_{\mathbf{g}\in\mathscr{H}},\ \right) \tag{4.7}$$

The first sum is equal to

$$\begin{aligned}
\sum_{\mathbf{h}\in\mathcal{S}_w^n} \mathbf{Var}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\,\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right) &= \sum_{\mathbf{h}\in\mathcal{S}_w^n} \mathbf{Var}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\right)\mathbf{Var}\left(\,\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right) && (4.8)\\
&\leqslant \sum_{\mathbf{h}\in\mathcal{S}_w^n} \mathbf{Var}\left(\,\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right) && (4.9)\\
&\leqslant N. && (4.10)
\end{aligned}$$

And the second sum is

$$\begin{aligned}
&\sum_{\substack{\mathbf{h},\mathbf{g}\in\mathcal{S}_w^n \\ \mathbf{h}\neq\mathbf{g}}} \mathbf{Cov}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\,\mathbf{1}_{\mathbf{h}\in\mathscr{H}},\ (-1)^{\langle \mathbf{y},\mathbf{g}\rangle}\,\mathbf{1}_{\mathbf{g}\in\mathscr{H}},\ \right)\\
&= \sum_{\mathbf{h}\in\mathcal{S}_w^n}\sum_{\mathbf{g}\in\mathcal{S}_w^n\,:\,\mathbf{g}\neq\mathbf{h}} \mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{h}+\mathbf{g}\rangle}\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\mathbf{1}_{\mathbf{g}\in\mathscr{H}}\right) - \mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right)\mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{g}\rangle}\mathbf{1}_{\mathbf{g}\in\mathscr{H}}\right)\\
&\quad \sum_{\mathbf{h}\in\mathcal{S}_w^n}\sum_{\mathbf{g}\in\mathcal{S}_w^n\,:\,\mathbf{g}\neq\mathbf{h}} \mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{h}+\mathbf{g}\rangle}\right)\mathbb{E}\left(\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\mathbf{1}_{\mathbf{g}\in\mathscr{H}}\right) - \mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\right)\mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{g}\rangle}\right)\mathbb{E}\left(\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right)\mathbb{E}\left(\mathbf{1}_{\mathbf{g}\in\mathscr{H}}\right)\\
&\quad \sum_{\mathbf{h}\in\mathcal{S}_w^n}\sum_{\mathbf{g}\in\mathcal{S}_w^n\,:\,\mathbf{g}\neq\mathbf{h}} \mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{h}+\mathbf{g}\rangle}\right)\mathbb{E}\left(\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\mathbf{1}_{\mathbf{g}\in\mathscr{H}}\right) - \mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\right)\mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{g}\rangle}\right)\mathbb{E}\left(\mathbf{1}_{\mathbf{h}\in\mathscr{H}}\right)\mathbb{E}\left(\mathbf{1}_{\mathbf{g}\in\mathscr{H}}\right)\\
&= 0
\end{aligned} \tag{4.11}$$

where the last equality comes from the fact that $\mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{h}+\mathbf{g}\rangle}\right) = 0$ and $\mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\right) = 0$. This concludes the proof. □

4.3 About the procedure used to compute the sparse dual vectors

Let us now make a slightly more general discussion on the possible output distributions of Compute-Dual-Vectors. First, notice that all the current ISD decoders, or some slight variation of them, can readily be used to produce dual vectors of small weight w. If the decoder takes a triplet $\mathcal{C}, \mathbf{y}, t$ to decode $\mathbf{y}$ onto $\mathcal{C}$ at distance t we can give it instead $\mathcal{C}^\perp, \mathbf{0}, w$. Let us first break their output distribution into a few categories that we list below.

4.3.1 Computing the whole set of dual vectors of a certain weight

In this thesis, the analysis of our dual attacks and our complexity claims will be made in most cases when the procedure computing the dual vectors outputs the set of all dual vectors of a certain weight w with good probability. This is an arbitrary choice. This is essentially the case of the simple Dumer subroutine, see Proposition 13 and the more advanced BJMMM subroutine, see Proposition 14.

4.3.2 Computing a uniform looking subset of the dual vectors of a certain weight and the sublinear regime

A subset of size N of $\mathcal{C}^\perp \bigcap \mathcal{S}_w^n$ which is somewhat nicely distributed i.e. we could roughly model the distribution of $\mathscr{H}$ as if it was taken uniformly at random from all subsets of size N of $\mathcal{C}^\perp \bigcap \mathcal{S}_w^n$. Some recent sieving techniques [GJN24, DEEK24] can be used to compute some uniform looking smaller proportion of the dual vectors of weight w. Also, a slight adaptation of the Prange ISD decoder could be used for this case, it is particularly useful when the error is sublinear in the error weight as this is the only technique known right now which can compute dual vectors of weight $< n/2$ in sub exponential time could be used.

Algorithm 1 (Adapted Prange Decoder)**.** *We call the Adapted Prange Decoder the algorithm which, given a code $\mathcal{C}$ of length n, a dimension k, a weight w and a target N runs the following procedure a number*

$$n^2 \, N \frac{\binom{k}{w}}{2^k}$$

of time: choose at random a subset $\mathscr{I}$ of $[\![1,\, n]\!]$ of size $n-k$ and a random element i of $\mathscr{I}$ and compute, if it exists, the unique $\mathbf{h} \in \mathcal{C}^\perp$ such that $\mathbf{h}_{\mathscr{I}\setminus\{i\}} = \mathbf{0}$ and $h_i = 1$. Store $\mathbf{h}$ in a set if $|\mathbf{h}| = w$. Stop if we have found a total of N distinct dual vectors of weight w.

Model 2. *We make the model that given as input $\mathcal{C}, N, w$ Algorithm 1 returns a subset of size N (it is exists) that was chosen uniformly at random among subset of size N of $\mathcal{C}^\perp \bigcap \mathcal{S}_w^n$.*

In particular, we can show the following theorem giving the performance of the dual distinguisher to distinguish when the error weight is sublinear in the code length and when this Prange routine is used to compute the dual vectors.

Theorem 6 (Complexity of the distinguisher in the sublinear regime)**.** *Let $R \in]0,\, 1[$ be a constant and let t be an implicit function of $n \in \mathbb{N}$ such that $t = o(n)$. There exists N and w and two implicit functions of n that are such that*

$$N \in \widetilde{\mathcal{O}}\Big(2^{-t \log_2(1-R)(1+o(1))}\Big) \qquad \text{and} \qquad w \in nR/2 + \mathcal{O}(\log_2 n)$$

and such we have the following. Using Algorithm 1 to compute dual vectors in Algorithm 11 and supposing that Model 2 is valid, Algorithm 11 with $T \stackrel{def}{=} (1/2) N \delta_w^{(n)}(t)$ solves $\mathrm{DP}_{\mathbf{G}}(n, \lfloor Rn \rfloor, t)$ with probability $1 - o(1)$ in time

$$\widetilde{\mathcal{O}}\Big(2^{-2\,t\,\log_2(1-R)(1+o(1))}\Big).$$

We do not show this theorem completely but give the main ingredients for the proof. Basically we can deduce from our second-order concentration bounds that as long as $N \geqslant \frac{n}{\delta_w^{(n)}(t)^2}$ and $N < \frac{1}{2}\frac{\binom{n}{w}}{2^k}$ then under Model 2 we can distinguish with probability $1 - o(1)$. The added condition that $N < \frac{1}{2}\frac{\binom{n}{w}}{2^k}$ only comes from the fact that we need that there exists at least N dual vectors of weight w with overwhelming probability. This last condition is readily verified by the N in the theorem because R is constant and $t = o(n)$. We conclude the proof with Proposition 27 which gives that $\delta_w^{(n)}(t) = 2^{t \log_2(1-2\omega)(1+o(1))n} = 2^{t \log_2(1-R+\mathcal{O}(\log_2 n/n))(1+o(1))n} = 2^{t \log_2(1-R)(1+o(1))n}$ where the last equality comes from Taylor's theorem.

4.3.3 More involved distributions

Here we discuss technique that can be used to essentially produce the set of all vectors of $\mathcal{C}^\perp \bigcap \mathcal{V}$ where $\mathcal{V}$ is some fixed and known subset of $\mathcal{S}_w^n$. Essentially this can be done with an iteration of a full-fledged ISD, namely an iteration of the Dumer ISD framework of Algorithm 2 or an iteration of Algorithm 5.

For example one iteration of [MO15, BM17, BM18] naturally produces dual vectors of weight w but they will be of distinct and fixed known weights $w^{(i)}$ on some subpart $\mathscr{P}^{(i)}$ of the support. In that case, and with $\mathbf{y} = \mathbf{c} + \mathbf{e}$ we would have that:

$$\mathbb{E}\left(F_{\mathscr{H}}\left(\mathbf{y}\right)\right) = N \prod_i \delta_{w^{(i)}}^{(|\mathscr{P}^{(i)}|)}\left(|e_{\mathscr{P}^{(i)}}|\right).$$

Note that we could even use a slightly modified iteration of the ISD [Dum91, MMT11, BJMM12]. An iteration of those ISD's produces dual vectors which are of small Hamming weight on some part $\mathscr{I}$ of the support and which are uniformly distributed on some other part $\mathscr{R}$ of the support (namely on the extended information part and the redundant part respectively). As such, considering $\mathbf{h}$ such a dual vector, we have, as soon as the error is non-null on the $\mathscr{R}$ part that $\langle \mathbf{e}, \mathbf{h} \rangle$ is not biased, which is destructive for our algorithm. We can remedy to this problem by filtering the dual vectors in a way that we keep only those which are of fixed weight, say p, on the part $\mathscr{R}$. But, we can make $\langle \mathbf{e}, \mathbf{h} \rangle$ biased toward 0 even with a polynomial filtering, that is, choosing wisely $p = |\mathscr{R}|/2 - o(n)$ close to the typical relative weight $1/2$ is sufficient to obtain a usable bias, even if the weight of the error on $\mathscr{R}$ is say linear in $|\mathscr{R}|$. It is clear that in these regimes, $|\mathbf{e}_{\mathscr{R}}|$ lies inside the root region of $K_w^{(n)}$ and from Corollary 5 we get that there exists some point $o(n)$ close to $|\mathscr{R}|/2$ such that the bias related to the part $\mathscr{R}$ achieves its asymptotic expansion, namely in that case we have

$$\delta_p^{(|\mathscr{R}|)}\left(|\mathbf{e}_{\mathscr{R}}|\right) = \frac{2^{-o(n)}}{\sqrt{\binom{|\mathscr{R}|}{|\mathbf{e}_{\mathscr{R}}|}}}.$$

4.4 Turning this distinguisher into a decoder

We build here a proven dual decoder using standard reductions from decoding to distinguishing (see for example [Deb23]). For example, we can show that

Proposition 32 (Reducing decoding ($\mathrm{DP}_{\mathbf{G}}(n,k,t)$) to distinguishing ($\mathrm{DiP}_{\mathbf{G}}(n,k,t)$))**.** *Let $n, k, t \in \mathbb{N}$ and let f be a function of n. Given an algorithm which solves $\mathrm{DiP}_{\mathbf{G}}(n,k,t)$ with an advantage of $1 - 1/(n\,f(n))$ then there exists an algorithm which solves $\mathrm{DP}_{\mathbf{G}}(n,k,t)$ with probability $1 - \mathcal{O}(1/f(n))$.*

With this reduction we can build the following decoder:

Algorithm 2. *We call* Variant-Statistical-Decoding$(\mathcal{C}, \mathbf{y}, t)$ *the procedure which given a code $\mathcal{C}$, a noisy codeword $\mathbf{y}$ and a decoding distance t along with some parameter w and applies the reduction of Proposition 32 using Algorithm 11 for the distinguisher.*

whose performance directly follows from this last proposition Proposition 32 along with the correctness of our dual distinguisher given in Theorem 4. Let us now explain how the reduction in Proposition 32 is obtained. Say we are given a code $\mathcal{C}$ and some noisy codeword

$\mathbf{y}$ obtained by $\mathrm{DP}_{\mathbf{G}}(n,k,t)$. In this problem, the code is in fact given through its generator matrix $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ which is taken uniformly at random in $\mathbb{F}_2^{k\times n}$ and $\mathbf{y}$ is some $\mathbf{c}+\mathbf{e}$ where $\mathbf{c} = \mathbf{mG}$ for some $\mathbf{m} \in \mathbb{F}_2^k$ and $\mathbf{e} \in \mathcal{S}_t^n$. Our goal will be to recover each position of the message $\mathbf{m}$. To recover the i'th position of $\mathbf{m}$ we change the generator matrix $\mathbf{G}$ into say $\mathbf{G}'$ by adding a vector $\mathbf{x}$ which was uniformly and independently taken from $\mathbb{F}_2^n$ to its i'th line and notice that $\mathbf{G}'$ is uniformly distributed in $\mathbb{F}_2^{k\times n}$, that it is independent of $\mathbf{y}$ and that as such

- If $m_i = 0$ then $(\mathbf{G}',\ \mathbf{y})$ is distributed according to the distribution $\mathcal{D}_0$ in $\mathrm{DiP}_{\mathbf{G}}(n,k,t)$. This is due to the fact that $\mathbf{c} = \mathbf{mG} = \mathbf{mG}'$ thus $\mathbf{y}$ is a noisy codeword of the code generated by $\mathbf{G}'$

- If $m_i = 1$ then $(\mathbf{G}',\ \mathbf{y})$ is distributed according to the distribution $\mathcal{D}_1$ in $\mathrm{DiP}_{\mathbf{G}}(n,k,t)$. This is due to the fact that $\mathbf{c} = \mathbf{mG} = \mathbf{mG}' + \mathbf{x}$ where we conclude with the fact that $\mathbf{G}'$ and $\mathbf{x}$ are independent.

This decoder guesses $\mathbf{m}$ and returns the error $\mathbf{mG}+\mathbf{y}$ if it is of weight t, else it returns fails. The proof of Proposition 32 follows from the union bound as we can upper bound the probability of failing to recover m_i correctly by some $\mathcal{O}(1/f(n))$.

4.5 A simplification of the analysis of [DT17a]

As a side note, we devise a simplification of [DT17a] analysis. As explained in Section 2.2.3.3 at one point in their analysis [DT17a, Lemma 2,3,4] they estimate the asymptotic expansion of some sum of bias, namely $\left(\delta_{w-1}^{(n-1)}(t) + \delta_{w-1}^{(n-1)}(t-1)\right)^2$ by adding the asymptotic expansion of $K_{w-1}^{(n-1)}$ known from [IS98, Theorem 3.1] and compute the result. Here we note that this sum of bias can be expressed directly as another bias allowing us to use directly the known expansion of the bias (i.e. Corollary 5).

Proposition 33 (Differences of expected value)**.**

$$\delta_{w-1}^{(n-1)}(t) + \delta_{w-1}^{(n-1)}(t-1) = \Theta(1)\,\delta_{w-1}^{(n-2)}(t-1)$$

Proof. Recall that

$$\delta_w^{(n)}(t) = \frac{K_w^{(n)}(t)}{\binom{n}{w}}.$$

By using the recurrence relations that we gave in Proposition 20 we have that

$$K_{w-1}^{(n-1)}(t-1) = K_{w-1}^{(n-2)}(t-1) + K_{w-2}^{(n-2)}(t-1)\,.K_{w-1}^{(n-1)}(t) \quad = K_{w-1}^{(n-2)}(t-1) - K_{w-2}^{(n-2)}(t-1)\,.$$

Notice that

$$\binom{n-1}{w-1} = \frac{n-1}{n-w}\binom{n-2}{w-1}.$$

As such

$$\begin{aligned}\delta_{w-1}^{(n-1)}(t)+\delta_{w-1}^{(n-1)}(t-1) &= \frac{2\,K_{w-1}^{(n-2)}(t-1)}{\binom{n-1}{w-1}}\\ &= 2\,\frac{n-w}{n-1}\,\frac{K_{w-1}^{(n-2)}(t-1)}{\binom{n-2}{w-1}}\\ &= \Theta(1)\,\delta_{w-1}^{(n-2)}(t-1)\,.\end{aligned}$$

□

Chapter 5

Dual Attack 2.0 : Reducing Decoding to LPN

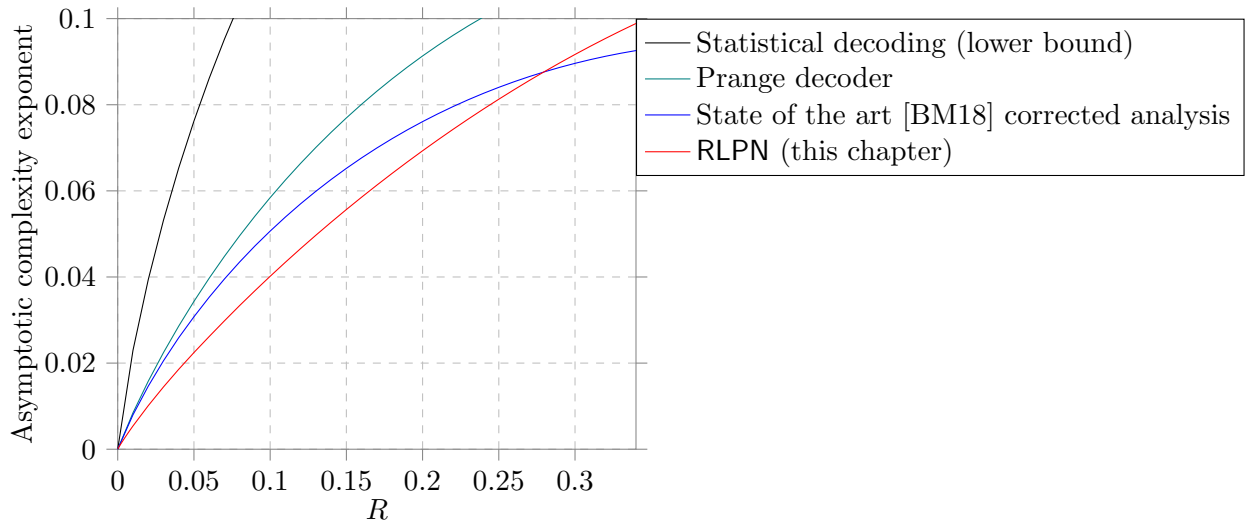


Figure 5.1: Asymptotic complexity exponent, relative to the code length n of some generic decoders when decoding a random code of rate R at the Gilbert-Varshamov distance. Our corrected analysis of [BM18] is given in Section 2.1.3.4. The statistical decoding curve is the lower bound on [Jab01] given by [DT17a, Figure 7, Optimal Statistical decoding] by supposing that each dual vector can be computed in polynomial time.

Summary

In this chapter we devise and analyze a new dual attack that we call RLPN (Reducing Decoding to LPN). Our algorithm revisits statistical decoding and makes dramatic improvements to the original algorithm. We show that it significantly asymptotically outperforms all known generic decoders when decoding codes of small constant rates smaller than 0.3 at the Gilbert-Varshamov distance. Our algorithm introduces a splitting strategy that was originally suggested but not exploited in [DT17a]: we use the fact that we can slightly modify statistical

decoding so that each dual vector now yields an LPN sample where the secret is some part of the error vector and where the noise of the sample involves only some part of the error and the dual vector. Our algorithm computes many such dual vectors and solves the resulting LPN problem with an FFT as used in [LF06]. We show experimentally that, to analyze our algorithm, we cannot make the simplistic model that our LPN samples are distributed as true LPN samples. Our analysis relies on the second-order concentration bound we devised in the previous chapter and on some exponential strengthening of these bounds. We are unable to prove the latter but provide a conjecture that we verify experimentally. This allows us to conclude that the performance of our algorithm is not affected by this distribution discrepancy. This chapter contains a rewritten version of [CDMT22] and [MT23].

Contents

5.1 Introduction

5.1.1 Reducing Decoding to LPN

Say we are given a decoding problem $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e} \in \mathcal{S}_t^n$. We notice that if we split the support $[\![1,\, n]\!]$ in two fixed parts $\mathscr{P}$ and $\mathscr{N}$, then a dual vector $\mathbf{h} \in \mathcal{C}^\perp$ yields

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{e}_{\mathscr{P}}, \mathbf{h}_{\mathscr{P}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle \,,$$

and, we notice that if $\mathbf{h}$ is of low weight on $\mathscr{N}$ this $\mathbf{h}$ directly gives us access to the following LPN sample:

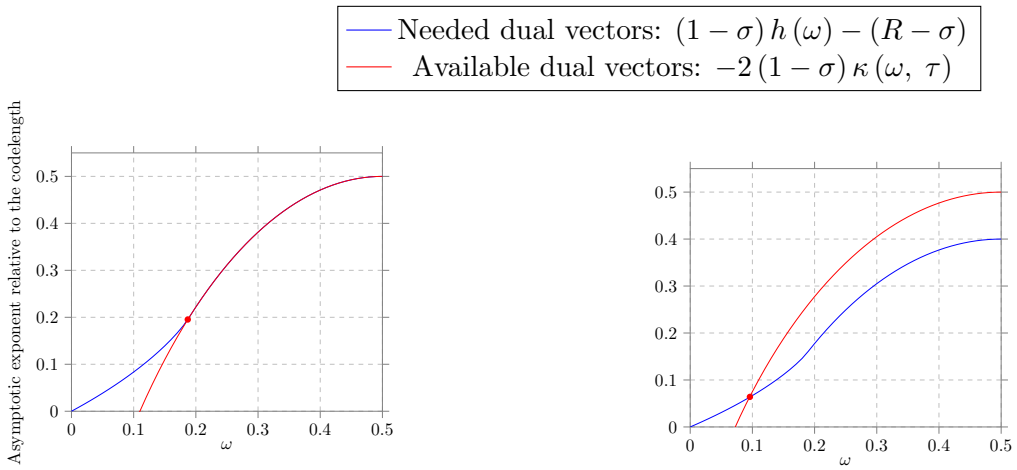
$$(\mathbf{a},\ \langle \mathbf{a}, \mathbf{s} \rangle + e) \quad \text{where} \quad \begin{cases} \mathbf{a} &= \mathbf{h}_{\mathscr{P}} \\ \mathbf{s} &= \mathbf{e}_{\mathscr{P}} \\ e &= \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle \end{cases} \tag{5.1}$$

where the secret $\mathbf{s}$ is given by $\mathbf{e}_{\mathscr{P}}$ and the noise e is given by $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$ more biased toward 0 as $\mathbf{e}_{\mathscr{N}}$ and $\mathbf{h}_{\mathscr{N}}$ are of lower weights. As such, given many $\mathbf{h}$ of low weight say w on $\mathscr{N}$ we get many such LPN samples. We call our new algorithm leveraging this reduction RLPN (Reducing Decoding to LPN).

5.1.2 Rationale

In essence, statistical decoding is this strategy but with $|\mathscr{P}| = 1$, our approach is to take $|\mathscr{P}|$ bigger. Intuitively this allows us to naturally increase the bias of the noise $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$ and hence decrease the number of needed dual vectors to recover $\mathbf{e}_{\mathscr{P}}$. Let us be more precise, define the bias of the noise $\varepsilon = \mathrm{bias}_{\mathbf{h}}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle\right)$, making some simplifications, one could expect that $N \approx s/\varepsilon^2$ samples will be needed in order to recover the secret $\mathbf{e}_{\mathscr{P}}$ (this would be true by using Shannon's second theorem [RU08, Theorem 4.68, p. 203] and supposing that our LPN samples are true LPN samples, see Fact 15). We illustrate the gain of taking $\mathscr{P}$ bigger in Fig. 5.2b. For illustration purpose, we suppose that the dual vectors of $\mathcal{C}$ are taken uniformly of weight w on $\mathscr{N}$ and we make the approximation that we can forget about the code structure to get that

$$\varepsilon \approx \mathrm{bias}_{\mathbf{h}'_{\mathscr{N}} \sim \mathcal{U}\left(\mathcal{S}_w^{|\mathscr{N}|}\right)}\left(\left\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}} \right\rangle\right) = \delta_w^{(|\mathscr{N}|)}\left(|\mathbf{e}_{\mathscr{N}}|\right).$$



(a) In statistical decoding, $\sigma = 0$

(b) In RLPN, when say $\sigma = 0.2$

Figure 5.2: Asymptotic exponent, relative to the code length n, representing the number of available dual vectors against the number of needed dual vectors to recover the error as a function of the relative weight $\omega \stackrel{\text{def}}{=} |\mathbf{h}_{\mathscr{N}}| / |\mathscr{N}|$ of the dual vectors on $\mathscr{N}$. Here we are in the setting where we decode a code of rate $R = 1/2$ at the Gilbert-Varshamov distance and where we suppose that $|\mathbf{e}_{\mathscr{N}}|$ is of typical weight, namely $\tau \stackrel{\text{def}}{=} |\mathbf{e}_{\mathscr{N}}| / |\mathscr{N}|$ is equal to $\tau_{\text{GV}}(R)$. We denote by $\sigma \stackrel{\text{def}}{=} |\mathscr{P}| / n$ as such $|\mathscr{N}| / n = 1 - \sigma$.

Remark 13. *Notice that increasing $|\mathscr{P}|$ gives two benefits: at a fixed relative weight it increases the bias and the number of dual vectors.*

Of course, those gains in the bias come with the negative side effect that we now have to solve a problem of dimension $|\mathscr{P}|$. This is more complicated than in statistical decoding where $|\mathscr{P}| = 1$ and where the problem was solved with majority voting. However, this splitting strategy is beneficial because we can now balance the cost of recovering $\mathbf{e}_{\mathscr{P}}$ with the number N of needed dual vectors. For example the FFT technique we use recovers the secret run in time $\widetilde{\mathcal{O}}\left(2^{|\mathscr{P}|} + N\right)$. In statistical decoding, $|\mathscr{P}| = 1$ and N is exponential as mentioned above, so we are deep in a regime where we can afford a larger $\mathscr{P}$, hence diminishing the number of needed dual vectors N and hence diminishing the overall costs.

5.1.3 The RLPN algorithm

Basically the main step of our algorithm consists in choosing at random two complementary subsets $\mathscr{P}$ and $\mathscr{N}$ of $[\![1, n]\!]$ of size say s and $n - s$ respectively and computing many (an exponential number) dual vectors of low weight w on $\mathscr{N}$, each giving the LPN samples $(\mathbf{h}_{\mathscr{P}}, \langle \mathbf{y}, \mathbf{h} \rangle)$. To solve our related LPN problem, we compute a score function $F^{\mathcal{L}}$ which encodes for each $\mathbf{x}$, how biased is $\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle$ as a sum $\sum_{\mathbf{h} \in \mathscr{H}} = (-1)^{\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle}$, namely we have

Definition 32 (Score function of an LPN problem)**.** *Let $\mathcal{L}$ be a list of elements of the form $(\mathbf{a}, b)$ with $\mathbf{a} \in \mathbb{F}_2^s$ and $b \in \{0, 1\}$. We define for $\mathbf{x} \in \mathbb{F}_2^s$ the score function related to the* LPN *problem given by $\mathcal{L}$ as*

$$F^{\mathcal{L}}(\mathbf{x}) = \sum_{(\mathbf{a}, b) \in \mathcal{L}} (-1)^{b - \langle \mathbf{a}, \mathbf{x} \rangle}.$$

This score function is encodes the distance between the code $\mathcal{D} \stackrel{\text{def}}{=} \{\mathbf{c}_{\mathbf{x}} \stackrel{\text{def}}{=} (\langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle)_{\mathbf{h} \in \mathscr{H}} : \mathbf{x} \in \mathbb{F}_2^s \}$ generated by the LPN samples and the received noisy codeword $\mathbf{b} \stackrel{\text{def}}{=} (\langle \mathbf{y}, \mathbf{h} \rangle)_{\mathbf{h} \in \mathscr{H}} = \mathbf{c}_{\mathbf{e}_{\mathscr{P}}} + (\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle)_{\mathbf{h} \in \mathscr{H}}$. More precisely we have that

$$F^{\mathcal{L}}(\mathbf{x}) = |\mathscr{H}| - |\mathbf{b} - \mathbf{c}_{\mathbf{x}}|.$$

In other words maximizing the LPN score function is really finding the closest codeword of $\mathcal{D}$ to $(\langle \mathbf{y}, \mathbf{h} \rangle)_{\mathbf{h} \in \mathscr{H}}$. But, as we will see, even if the number of LPN samples is of the order $N \approx 1/\varepsilon^2$, because of some linear structure in our LPN samples we are not able to directly conclude that the secret $\mathbf{e}_{\mathscr{P}}$ is the maximum of the LPN score function. We will have to account for a few number of false candidates $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ whose score can be as big as the score associated to the secret $\mathbf{e}_{\mathscr{P}}$. Essentially, we will keep only those candidates $\mathbf{x}$ whose associated score is sufficiently big. Testing a candidate for $\mathbf{e}_{\mathscr{P}}$ and recovering the rest of the error $\mathbf{e}_{\mathscr{N}}$ is done by solving yet another smaller decoding problem.

Because the bias ε increases as the weight of $\mathbf{e}_{\mathscr{N}}$ decreases we will also add a slight bet that

$$|\mathbf{e}_{\mathscr{N}}| = u$$

for a certain parameter u and iterate our algorithm until this bet is valid, the idea is that the loss in the number of iteration should be outweighed by the gain in the bias.

5.1.4 Analysis

We analyze of our algorithm in the case where all the dual vectors of weight w on $\mathscr{N}$ are produced: in that case we can show that the bias of the noise of the LPN samples can be tightly estimated by $\varepsilon \approx \delta_w^{(n-s)}(u)$ which is essentially the bias of $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$ when forgetting about the code structure (and supposing that the bet is valid).

The technical difficulty of the analysis lies in the fact that, as we will show, our obtained LPN samples do not behave like true LPN samples hence we cannot use the usual information theoretic tools to show for example that we could recover the secret $\mathbf{e}_{\mathscr{P}}$ with overwhelming probability provided that the number of LPN sample is of order $N \approx 1/\varepsilon^2$. In fact, we show that, contrary to the true LPN setting there can exist in our algorithm some candidates $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$, which we call false candidates, and which have an unusually high score and are thus mistaken for the secret. As the complexity of testing a candidate directly appears in the complexity we must estimate their number precisely. The conclusion of our analysis is that taking $N \approx 1/\varepsilon^2$ is basically completely sufficient so that false candidates are not a problem: their number can be poly-bounded, which means that their potential presence does not change the overall complexity by more than a polynomial factor.

We give here the main steps behind our analysis. We notice that in fact analyzing our algorithm essentially boils down to analyzing some (stronger) variant of our previously introduced dual distinguisher. Indeed we can use some linearity relation between the coordinates of the dual vectors $\mathbf{h}$'s to rewrite the LPN samples as

$$\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle = \langle f(\mathbf{x}), \mathbf{h}_{\mathscr{N}} \rangle$$

for a certain fixed (but unknown) affine function f where we prove that $f(\mathbf{x})$ is uniformly random when $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ and $f(\mathbf{e}_{\mathscr{P}}) = \mathbf{e}_{\mathscr{N}}$. Clearly $\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}$ where it is readily seen that $\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}$ is generally an $[n-s, n-k]$ linear code, thus $\mathbf{h}_{\mathscr{N}}$ is in fact a dual vector of low weight

w of an $[n-s, k-s]$ linear code $\left(\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}\right)^{\perp} = \mathcal{C}^{\mathscr{N}}$ which is simply the code $\mathcal{C}$ shortened in $|\mathscr{N}|$ positions. Clearly, using the invariance $\left\langle f(\mathbf{x}) + \mathbf{c}^{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \right\rangle$ for $\mathbf{c}^{\mathscr{N}} \in \mathcal{C}^{\mathscr{N}}$ we see that this term is biased toward zero as soon as $f(\mathbf{x})$ is sufficiently close to $\mathcal{C}^{\mathscr{N}}$. This rewritting also show that our LPN score function can rewritten as a score function as defined in the previous chapter, namely

$$F^{\mathcal{L}}(\mathbf{x}) = \sum_{\mathbf{h}_{\mathscr{N}} \in \mathcal{S}_w^{n-s} \bigcap \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}} (-1)^{\langle f(\mathbf{x}), \mathbf{h}_{\mathscr{N}} \rangle}.$$

Contrary to our previously introduced dual distinguisher, we must now distinguish the secret $\mathbf{e}_{\mathscr{P}}$ (associated to $f(\mathbf{e}_{\mathscr{P}}) = \mathbf{e}_{\mathscr{N}}$ of low weight) among an **exponential** number of candidates $\mathbf{x}$ (associated to $f(\mathbf{x})$ uniformly random). Interestingly, the second-order concentration bounds devised in the previous chapter allows us to prove that as long as the number of dual vectors of weight w on $\mathscr{N}$ is of the order

$$N = \mathrm{poly}(n) / \delta_w^{(n-s)}(u)^2$$

then we can, using the value of $F^{\mathcal{L}}(\mathbf{x})$, distinguish an $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ from the secret $\mathbf{e}_{\mathscr{P}}$ with polynomial probability (our intuition is by the way that taking N smaller would lead to our reduction being completely useless: we would have no usable advantage at all).

However, as we have an exponential number of $\mathbf{x}$ to compare against $\mathbf{e}_{\mathscr{P}}$, these bounds will be completely insufficient to say anything useful and some exponential strengthening will be needed. We will not be able to prove directly the needed exponential strengthening but we make the natural conjecture that the only case where some $\mathbf{x}$ is mistaken for $\mathbf{e}_{\mathscr{P}}$ is when $f(\mathbf{x})$ is unusually close to $\mathcal{C}^{\mathscr{N}}$, namely when $f(\mathbf{x})$ is at distance less than $f(\mathbf{e}_{\mathscr{P}}) = \mathbf{e}_{\mathscr{N}}$ from $\mathcal{C}^{\mathscr{N}}$. We give extremely strong experimental evidences that this conjecture is true by devising a key duality formula, see Proposition 34, for the score function: it shoes that $F^{\mathcal{L}}(\mathbf{x})$ deeply relates to the weight enumerator of some coset code of $\mathcal{C}^{\mathscr{N}}$, namely $\mathcal{C}^{\mathscr{N}} + f(\mathbf{x})$. By making a simple model on the distribution of the weight enumerator we are able to prove our conjecture while showing experimentally that making this model does not change the distribution of the score function.

Proposition 34 (Duality formula for the score function)**.** *Let n, k, w and let $\mathcal{C} \in \mathfrak{C}[n,k]$ and let $\mathbf{y} \in \mathbb{F}_2^n$. We have that*

$$\sum_{\mathbf{h} \in \mathcal{C}^{\perp} \bigcap \mathcal{S}_w^n} (-1)^{\langle \mathbf{y}, \mathbf{h} \rangle} = \frac{1}{2^k} \sum_{i=0}^{n} N_i(\mathcal{C} + \mathbf{y}) K_w^{(n)}(i) \tag{5.2}$$

where we recall that the weight enumerator is given by $N_i(\mathcal{C} + \mathbf{y}) \stackrel{def}{=} |\mathcal{C} \bigcap \mathcal{S}_i^n|$.

Related work

Our dual attack can be seen as the coding theoretic analogue of the dual attacks in lattice-based cryptography. Namely, the first dual attack in lattice [AR04] could be seen as the lattice equivalent to statistical decoding and was also completely uncompetitive against the lattices based primal attacks. It has then gained a series of crucial improvements [Alb17, EJK20, GJ21, MAT22, CST22], one which coming from a similar splitting strategy. All the analysis of these lattice-based dual attacks rely on some independence assumptions.

Initially, we proposed and published in [CDMT22] a first version of our RLPN decoder were we assumed in the analysis some variant of these independence assumptions [CDMT22, Assumption 3.6], namely that the obtained LPN samples behaved like true LPN samples. At that time we noticed by running some experimentation that this model was not always accurate [CDMT22, Figure 3.1], but we conjectured there that the discrepancy between this ideal model and experiments does not impact our asymptotic analysis of the algorithm by more than a polynomial factor.

Then, in lattice-based cryptography, the dual attacks were strongly questioned by [DP23b] who showed that these independence assumptions were in contradiction with some theorems in certain regimes or with well-tested heuristics in some other regimes. The paper however did not give a way to analyze them, and it was left as an open question if these lattice-based dual attacks really could work as expected.

In turn, we proved in [MT23] that the LPN model we made in [CDMT22] to analyze our decoder could not always hold and, it was shown that the number of false candidates in our RLPN decoder given in [CDMT22] was in fact exponentially large for some parameters, whereas there should be none if the assumptions held. However, we gave at the same time an approach for analyzing our attack with the help of an accurate model. In particular, we also showed that we had to slightly modify the original RLPN decoder to be able to achieve to originally claimed the complexity exponent given in [CDMT22]. The scientific part of our work in [MT23] was made independently of [DP23b].

5.2 The RLPN algorithm

Recall that we want to decode at distance t a given noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{e} \in \mathcal{S}_t^n$ is an error vector of weight t and $\mathbf{c}$ is a codeword of a given linear code $\mathcal{C}$ of length n.

5.2.1 Outline of the algorithm

The main step of our algorithm consists by first choosing at random two complementary subsets $\mathscr{P}$ and $\mathscr{N}$ of $[\![1,\, n]\!]$ of a certain size say s and $n-s$ and then proceed by calling the three following main procedures:

1. Compute-LPN-Sample. It computes many dual vectors $\mathbf{h} \in \mathcal{C}^\perp$ of low weight w on $\mathscr{N}$ and store and returns the list, say $\mathcal{L}$, composed of the LPN samples $(\mathbf{a}, b) = (\mathbf{h}_{\mathscr{P}}, \langle \mathbf{y}, \mathbf{h} \rangle)$ with secret $\mathbf{e}_{\mathscr{P}}$, as described in Eq. (5.1).

2. LPN-Solver. It takes as input the list of LPN samples and outputs a (small) set of candidates $\mathbf{x}$ for the secret of the LPN problem $\mathbf{e}_{\mathscr{P}}$.

3. Recover-Full-error. It takes as input the set of candidates, if $\mathbf{e}_{\mathscr{P}}$ is in the set of candidate we expect that this procedure returns the full error $\mathbf{e}$ or it fails.

Moreover as we explained in the introduction, we in fact make some additional bet that the weight of the error on the part $\mathscr{N}$, namely $|\mathbf{e}_{\mathscr{N}}|$ is of unusually low weight, say u: this allows to drastically reduce the number of needed LPN samples in order to solve our LPN problem (i.e. recover $\mathbf{e}_{\mathscr{P}}$ with good probability). The rationale is that when this bet is invalid simply that $\mathbf{e}_{\mathscr{P}}$ will not be found in the set of candidate but as soon as it is $\mathbf{e}_{\mathscr{P}}$ will appear in it.

As such we iterate this main step a certain number N_{iter} of time in order for our bet to be verified at least once. Overall this main structure is described in the following Algorithm 12.

Algorithm 12 The Reducing Decoding to LPN (RLPN) algorithm

Name: RLPN
Input: $\mathcal{C} \in \mathfrak{C}[n,k]$, $\mathbf{y} \in \mathbb{F}_2^n$, t
Parameter: s, w, u and T, N_{iter}

1: **while** $i = 1 \dots N_{\text{iter}}$ **do**
2: $\quad \mathscr{P} \xleftarrow{\$} \{\mathscr{P} \subset [\![1, n]\!] : |\mathscr{P}| = s\}$ ▷ *Hope that* $\mathbf{e}_{\mathscr{N}} = u$
3: $\quad \mathscr{N} \leftarrow [\![1, n]\!] \setminus \mathscr{P}$
4: $\quad \mathcal{L} \leftarrow$ Create-LPN-Samples$(\mathcal{C}, \mathscr{N}; w)$ ▷ *$\mathcal{L}$ contains an expected number of N LPN samples of the form $(\mathbf{a}, b)$ where $\mathbf{a} \in \mathbb{F}_2^s$ and $b \in \{0,1\}$*
5: $\quad \mathcal{S} \leftarrow$ RLPN-LPN-Solver$(\mathcal{L}; t-u, T)$ ▷ *Returns a set of candidates for the secret of the LPN problem given by $\mathcal{L}$, $t-u$ is the bet of weight of the secret and T is a treshold*
6: $\quad \mathbf{e} \leftarrow$ RLPN-Recover-full-error$(\mathcal{S}, \mathscr{P}, \mathscr{N}, \mathbf{y}, u)$ ▷ *Return either $\perp$ if $\mathbf{e}_{\mathscr{P}} \notin \mathcal{S}$ but returns $\mathbf{e}$ if $\mathbf{e}_{\mathscr{P}} \in \mathcal{S}$ and $|\mathbf{e}_{\mathscr{N}}| = u$*
7: $\quad$ **if** $\mathbf{e} \neq \perp$ **then**
8: $\quad\quad$ **return** $\mathbf{e}$

Let us now proceed describing in more details each of the individual 3 procedures.

5.2.2 Creating the LPN samples

Our main goal here is to compute a set $\mathscr{H}$ of dual vectors of $\mathcal{C}$ which are of small Hamming weight w on a subpart $\mathscr{N}$: each of these dual vectors $\mathbf{h} \in \mathscr{H}$ yields an LPN sample $(\mathbf{h}_{\mathscr{P}}, \langle \mathbf{y}, \mathbf{h} \rangle)$ which we store and return as a list $\mathcal{L}$. We note that computing the dual vectors of this shape reduces to finding dual vectors of weight w in a code of smaller rate and length, namely $\mathcal{C}^{\mathscr{N}}$, which is, extremely beneficial as it makes computing dual vectors much easier compared to statistical decoding say. Indeed, note first that as we are only interested on the weight of $\mathbf{h}_{\mathscr{N}}$, provided that $\mathscr{N}$ is an information set of the dual (which will be the case with overwhelming probability), we can only focus on this part. Namely, we have the following.

Lemma 13. *Let $\mathcal{C}$ be a linear code of length n and let $\mathscr{P}$ and $\mathscr{N}$ be two complementary subsets of $[\![1, n]\!]$ of size s and $n-s$ respectively. Provided that $\mathscr{N}$ is an (extended) information set of $\mathcal{C}^\perp$, the matrix $\mathbf{R} = \text{Lift}\left(\mathcal{C}^\perp, \mathscr{N}\right)$ in $\mathbb{F}_2^{s \times (n-s)}$ is such that*

$$\mathbf{h}_{\mathscr{P}} = \mathbf{h}_{\mathscr{N}} \mathbf{R}^\intercal, \qquad \forall \mathbf{h} \in \mathcal{C}^\perp.$$

Conversely, every $\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^\perp\right)_{\mathscr{N}}$ can be uniquely lifted to a dual vector $\mathbf{h} \in \mathcal{C}^\perp$ by defining $\mathbf{h}_{\mathscr{P}} = \mathbf{h}_{\mathscr{N}} \mathbf{R}^\intercal$.

Now, notice now that the space in which $\mathbf{h}_{\mathscr{N}}$ lives can be rewritten as the dual of the code $\mathcal{C}$ shortened on $\mathscr{N}$, namely we recall.

Lemma 1. *(Relation between shortening and puncturing [HP03, Theorem 1.5.7]) Let $\mathcal{C}$ be a linear code of length n and let $\mathscr{N} \subset [\![1, n]\!]$ be a set. We have that*

$$\mathcal{C}^{\mathscr{N}} = \left(\left(\mathcal{C}^\perp\right)_{\mathscr{N}}\right)^\perp$$

To compare with the ISD's, remember that they essentially leverage technique which are efficient to decode in the high rate regime by reducing decoding to decoding in a code of higher rate by the way of a bet on the error weight on some information set. We leverage exactly the dual of this phenomenon by using the fact that these techniques are efficient to compute low weight dual vectors in the low rate regime.

5.2.2.1 The procedure

Using this remark, we can build our procedure Create-LPN-samples from any procedure computing sparse dual vectors, say Compute-Short-Dual-Vectors. We start by producing a subset $\mathscr{H}_{\mathscr{N}}$ of dual vectors of $\mathcal{C}^{\mathscr{N}}$ by calling Compute-Dual-Vectors$(\mathcal{C}^{\mathscr{N}}; w)$ which outputs a subset of $\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp} \bigcap \mathcal{S}_w^{n-s}$. We then uniquely lift each of these vectors $\mathbf{h}_{\mathscr{N}} \in \mathscr{H}_{\mathscr{N}}$ into dual vectors $\mathbf{h} \in \mathcal{C}^{\perp}$ of weight w on $\mathscr{N}$ with the Lift $\left(\mathcal{C}^{\perp},\ \mathscr{N},\ \mathbf{h}_{\mathscr{N}}\right)$ procedure described in Definition 14. We recall that this lifting procedure can be obtained by making a partial Gaussian elimination on the position given by $\mathscr{P}$ of the given generator matrix $\mathbf{G}$ of the code $\mathcal{C}$. Say for the sake of the discussion that $\mathscr{P} = [\![1,\ s]\!]$ and $\mathscr{N} = [\![s+1,\ n]\!]$, we would thus put $\mathbf{G}$ in the form

$$\mathbf{G} = \begin{pmatrix} \mathbf{I}_s & \mathbf{R} \\ \mathbf{0}_{k-s\times s} & \mathbf{G}' \end{pmatrix} \tag{5.3}$$

where $\mathbf{R} \in \mathbb{F}_2^{s\times(n-s)}$ and $\mathbf{G}' \in \mathbb{F}_2^{(k-s)\times(n-s)}$ is a generator matrix of $\mathcal{C}^{\mathscr{N}}$. Then, lifting to $\mathbf{h} \in \mathcal{C}^{\perp}$ from $\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}$ is made by computing $\mathbf{h}_{\mathscr{P}} \stackrel{\text{def}}{=} \mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}$ as we must have that $\mathbf{G}\mathbf{h}^{\intercal} = \mathbf{0}$. This leads to Algorithm 13 whose complexity is, up to a polynomial factor the complexity of the call to Compute-Short-Dual-Vectors.

Algorithm 13 Computing the LPN samples

Name: Create-LPN-Samples
Input: $\mathcal{C} \in \mathfrak{C}[n,k]$, $\mathbf{y}$, $\mathscr{N}$
Require: A procedure Compute-Dual-Vectors$(\mathcal{D}; w)$ which given a linear code $\mathcal{D}$ outputs a subset of the dual vectors of $\mathcal{D}$ of weight w.
Parameter: w

1: **If** $\dim(\mathcal{C}_{\mathscr{P}}) = s$ **continue else** $i \leftarrow i+1$ and **go to** Line 2 of Algorithm 12 ▷ *Check that $\mathscr{N}$ is an information set of $\mathcal{C}^{\perp}$. Continue with overwhelming probability.*
2: $\mathscr{H}_{\mathscr{N}} \leftarrow$ Compute-Dual-Vectors$(\mathcal{C}^{\mathscr{N}}\ ;\ w)$ ▷ *Returns a subset of dual vectors of $\mathcal{C}^{\mathscr{N}}$ which are of weight w*
3: $\mathscr{H} \leftarrow \{$Lift$(\mathcal{C}^{\perp}, \mathscr{N},\ \mathbf{h}_{\mathscr{N}})$ for $\mathbf{h}_{\mathscr{N}} \in \mathscr{H}_{\mathscr{N}}\}$ ▷ *Lift those dual vectors $\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}$ to make them dual vectors $\mathbf{h}$ of $\mathcal{C}$ of low weight w on $\mathscr{N}$*
4: $\mathcal{L} \leftarrow [(\mathbf{h}_{\mathscr{P}},\ \langle \mathbf{y}, \mathbf{h}\rangle)$ for $\mathbf{h} \in \mathscr{H}]$
5: **return** $\mathcal{L}$

We refer to Section 4.3 for an overview of the techniques that can be used to compute low-weight dual vectors and that are tailored to the constant-rate regime we focus on.

5.2.3 Solving the LPN problem

In essence, we enumerate all the possible solutions $\mathbf{x}$ for $\mathbf{e}_{\mathscr{P}}$ and keep those candidates $\mathbf{x}$ for $\mathbf{e}_{\mathscr{P}}$ that are sufficiently good, namely such that $\langle \mathbf{y}, \mathbf{h}\rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}}\rangle$ is sufficiently biased toward 0, namely we compute the score function for all $\mathbf{x} \in \mathbb{F}_2^s$ defined as follows.

Definition 33 (LPN score function)**.** *We define the* LPN *score function as*

$$F^{\mathcal{L}}(\mathbf{x}) = \sum_{\mathbf{h} \in \mathscr{H}} (-1)^{\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle}$$

Here, we use the FFT trick, see Section 3.2.3, to compute this score function for all $\mathbf{x} \in \mathbb{F}_2^s$ in time $\widetilde{\mathcal{O}}(\max(2^s, |\mathscr{H}|))$. This allows an exponential gain compared to the cost of the naive computation which is $\widetilde{\mathcal{O}}(2^s \times |\mathscr{H}|)$ where 2^s and $|\mathscr{H}|$ are both exponential. In fact the naive computation could be greatly diminished by using the fact that the secret $\mathbf{e}_{\mathscr{P}} \in \mathbb{F}_2^s$ is known to be of low-weight, so we don't have to exhaust the whole space $\mathbb{F}_2^s$. But we do not consider this variant here and rather delegate leveraging the sparseness of $\mathbf{e}_{\mathscr{P}}$ to the next section.

Finally, we decide that $\mathbf{x}$ is a candidate for the solution $\mathbf{e}_{\mathscr{P}}$ if its associated score function value is superior to a well-chosen threshold T. We store those candidates in a list. Crucially here, we only keep those candidates $\mathbf{x}$ that are how correct weight $t-u$. Indeed, recalling in our main loop we made the bet that $|\mathbf{e}_{\mathscr{N}}| = u$, so in that case we know that $|\mathbf{e}_{\mathscr{P}}| = t-u$. This is important as, as we will see in the analysis later, this filtering allows to exponentially diminish the number of candidates and make the algorithm work.

Definition 34 (Set of candidates)**.** *We define the set of candidates* $\mathcal{S}$ *as*

$$\mathcal{S} \stackrel{def}{=} \{\mathbf{x} \in \mathcal{S}_{t-u}^s \,:\, F^{\mathcal{L}}(\mathbf{x}) > T\}.$$

Algorithm 14 The LPN solver

Name: RLPN-LPN-Solver
Input: $\mathcal{L}$ ▷ *$\mathcal{L}$ is a list of* LPN *samples of the form* $(\mathbf{a}, b)$ *where* $b = \langle \mathbf{a}, \mathbf{s} \rangle + e$ *and* $\mathbf{s} \in \mathbb{F}_2^s$
Parameter: v, T ▷ *v is the weight of the secret of the* LPN *problem given by* $\mathcal{L}$ *and* T *is the threshold from which we keep the candidates.*
1: $F^{\mathcal{L}} \leftarrow$ FFT-LPN-Solver($\mathcal{L}$) ▷ *The procedure is defined in Algorithm 10 and outputs*
2: $\mathcal{S} \leftarrow \{\mathbf{x} \in \mathcal{S}_v^s \,:\, F^{\mathcal{L}}(\mathbf{x}) \geqslant T\}$
3: **return** $\mathcal{S}$

Proposition 35 (Complexity of LPN-Solver)**.** *The time and memory complexity of Algorithm 19 is given by*

$$\mathrm{poly}(n) \max(2^s, |\mathcal{L}|).$$

5.2.4 Recovering the rest of the error

Here we are given a small subset $\mathcal{S}$ of candidates $\mathbf{x}$ for the secret $\mathbf{e}_{\mathscr{P}}$ and we want to test them for the secret. Basically we check if $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ by solving a (smaller) decoding problem onto $\mathcal{C}^{\mathscr{N}}$ at distance $|\mathbf{e}_{\mathscr{N}}| = u$. It returns fail if $\mathbf{x} \neq \mathbf{e}$ and returns the rest of the error, $\mathbf{e}_{\mathscr{N}}$, if $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$. This procedure will have an exponential time complexity, but the rationale is that it should not dominate the complexity of an iteration of RLPN. Recall that $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e} \in \mathcal{S}_t^n$. We leverage the fact that from the knowledge of $\mathbf{e}_{\mathscr{P}} \in \mathcal{S}_{t-u}^s$ we can construct the noisy codeword $\mathbf{y}' = \mathbf{c}^{\mathscr{N}} + \mathbf{e}_{\mathscr{N}}$ for some $\mathbf{c}^{\mathscr{N}} \in \mathcal{C}^{\mathscr{N}}$, we recover $\mathbf{e}_{\mathscr{N}}$ by calling a decoder, say Decoder($\mathcal{C}^{\mathscr{N}}, \mathbf{y}', u$) which decodes $\mathbf{y}'$ at distance u and returns $\perp$ if the problem has no solutions.

Indeed it is readily seen from Eq. (5.3) that $\mathbf{c}$ can be rewritten, for some $\mathbf{c}^{\mathscr{N}} \in \mathcal{C}^{\mathscr{N}}$ as

$$\mathbf{c} = (\mathbf{c}_{\mathscr{P}} \,||\, \mathbf{c}_{\mathscr{P}}\mathbf{R}) + (\mathbf{0} \,||\, \mathbf{c}^{\mathscr{N}})$$

where we recall that $\mathbf{R} \stackrel{\text{def}}{=} \text{Lift}\left(\mathcal{C}^{\perp},\ \mathscr{N}\right)$. Using this rewriting we have that:

$$\begin{aligned} \mathbf{y} &= (\mathbf{c}_{\mathscr{P}} + \mathbf{e}_{\mathscr{P}} \,||\, \mathbf{c}_{\mathscr{P}}\mathbf{R}) + (\mathbf{0} \,||\, \mathbf{c}^{\mathscr{N}} + \mathbf{e}_{\mathscr{P}}), \\ \mathbf{c}_{\mathscr{P}} &= \mathbf{y}_{\mathscr{P}} - \mathbf{e}_{\mathscr{P}}. \end{aligned}$$

As such, if $\mathbf{x}$ is our guess for $\mathbf{e}_{\mathscr{P}}$, we have by constructing

$$\mathbf{y}' = \mathbf{y}_{\mathscr{N}} - (\mathbf{y}_{\mathscr{P}} - \mathbf{x})\,\mathbf{R}$$

that $\mathbf{y}' = \mathbf{c}^{\mathscr{N}} + \mathbf{e}_{\mathscr{N}}$ in the case $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$. If this problem has no solution we have of course a false candidate for $\mathbf{e}_{\mathscr{P}}$. This is summed up in Algorithm 21.

Algorithm 15 Recovering the rest of the error

Name: RLPN-Recover-full-error
Input: $\mathcal{C}$, $\mathscr{P}$, $\mathscr{N}$, $\mathcal{S}$ ▷ *$\mathcal{S}$ is a list of candidates $\mathbf{x} \in \mathcal{S}^{s}_{l-u}$ for $\mathbf{e}_{\mathscr{P}}$*
Parameter: u

```
for x ∈ S do
    R ← Lift(C^⊥, N)
    y' ← y_N − (y_P − x) R
    z ← Sub-Decoder(C^N, y', u)
    if z ≠ ⊥ then
        Construct e such that e_P = x and e_N = z
        return e
return ⊥
```

We have several choices here depending on the number of candidates we will have to test, if this number is poly bounded we can use some very simple but inefficient decoder but we have to be careful about the decoder we choose if this number was to grow. It turns out that in this thesis, all our cases of interests in RLPN will lead to a poly bounded number of false candidates. In this case it is way simpler and sufficient to use for sub-Decoder our proved variant of statistical decoding, namely Algorithm 2. Indeed, it will defacto ensures that this last step does not add more than a polynomial factor to the overall complexity of the RLPN decoder. This comes from the fact that we can directly reuse the dual vectors computed in the Compute-LPN-samples procedure and which we stored in $\mathscr{H}_{\mathscr{N}}$ and plug them in this variant of statistical decoding. The parameters constraint we have on RLPN (i.e. that $N \approx 1/\varepsilon^2$) are then sufficient to prove that this last decoding step succeeds with high probability.

5.2.5 Complexity of the algorithm

The average complexity of our algorithm is straightforwardly given by the next proposition.

Proposition 36. *Let $n, k, t, s, T,\ N_{\text{iter}} \in \mathbb{N}$ be some parameters. The expected time and memory complexity of Algorithm 12 when given an instance of $\mathrm{DP}_{\mathbf{G}}(n, k, t)$ are given by*

$$\begin{aligned}\text{Time} &= \text{poly}(n)\, N_{\text{iter}} \left[2^s + T_{\text{eq}} + (1+S)\ T_{\text{subdec}}\right]\\ \text{Memory} &= \text{poly}(n) \left[2^s + M_{\text{eq}} + M_{\text{subdec}}\right]\end{aligned}$$

where S is the expected number of false candidates, namely

$$S \stackrel{def}{=} \mathbb{E}\left(|\mathcal{S} \setminus \{\mathbf{e}_{\mathscr{P}}\}|\right)$$

and where $T_{\text{eq}},\ M_{\text{eq}}$ are the time and memory complexity of Compute-Short-Dual-Vector *and $T_{\text{subdec}},\ M_{\text{subdec}}$ are the time and memory complexity of* Sub-Decoder.

Because the complexity T_{subdec} of the sub decoder will be exponential in say the distance u it is crucial to correctly estimate the expected number of false candidates. Let us first give an insight onto why this requires some technical arguments.

5.3 Analysis

We analyze our algorithm in the case where essentially all the dual vectors of weight w on $\mathscr{N}$ are produced. We are only interested in the behavior of our algorithm when the bet on the error is valid, namely when $\mathbf{e}_{\mathscr{N}} = u$. Our goal here is to give the conditions on the parameters to recover the secret $\mathbf{e}_{\mathscr{P}}$ of our LPN samples and to give a bound on the number of false candidates. We use in this section the following notation that captures the distributions and quantities encountered in a good iteration of RLPN.

Notation 3. *$n, k, t, w, s, u, T \in \mathbb{N}$ are some parameters. $\mathscr{P}$ and $\mathscr{N}$ are two fixed complementary subsets of $[\![1,\ n]\!]$ of size s and $n-s$ respectively. $\mathcal{C}$ is a linear code of length n such that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e} \in \mathcal{S}_t^n$ is fixed such that*

$$|\mathbf{e}_{\mathscr{N}}| = u.$$

Define the LPN *score function as*

$$F^{\mathcal{L}}(\mathbf{x}) \stackrel{def}{=} \sum_{\mathbf{h} \in \mathscr{H}} (-1)^{\langle \mathbf{y}, \mathbf{h}\rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}}\rangle}, \qquad \text{where} \qquad \mathscr{H} \stackrel{def}{=} \{\mathbf{h} \in \mathcal{C}^{\perp}\ :\ |\mathbf{h}_{\mathscr{N}}| = w\}$$

and define the set of candidates as

$$\mathcal{S} \stackrel{def}{=} \{\mathbf{x} \in \mathcal{S}_{t-u}^s\ :\ F^{\mathcal{L}}(\mathbf{x}) > T\}.$$

Whenever probabilities are involved the code $\mathcal{C}$ is the only random quantity and is taken according to $\mathcal{U}_{\mathbf{G}}(n,\ k)$ conditioned on the event that $\dim(\mathcal{C}_{\mathscr{P}}) = s$.

Lemma 14 (Main quantities involved in the analysis)**.** *The expected number of* LPN *samples is*

$$\mathbb{E}(|\mathscr{H}|) = N \qquad \text{where} \qquad N \stackrel{def}{=} \frac{\binom{n-s}{w}}{2^{k-s}}.$$

The expected value of the score function on the secret is

$$\mathbb{E}\left(F^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}})\right) = N\delta_w^{(n-s)}(u).$$

Because $F^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}}) = \sum_{\mathbf{h}\in\mathscr{H}}(-1)^{\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}}\rangle}$ this lemma shows roughly that the expected bias of the error $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}}\rangle$ of our LPN samples $\langle \mathbf{y}, \mathbf{h}\rangle = \langle \mathbf{e}_{\mathscr{P}}, \mathbf{h}_{\mathscr{P}}\rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}}\rangle$ is $\delta_w^{(n-s)}(u)$. This is basically the bias if we had ignored the code structure. If the LPN sample followed a true LPN distribution we would expect that it is sufficient that $N \geqslant s/\delta_w^{(n-s)}(u)^2$ to be able to recover the secret $\mathbf{e}_{\mathscr{P}}$ with overwhelming probability. This would be done by setting the threshold $T \overset{\text{def}}{=} \frac{1}{2} N \delta_w^{(n-s)}(u)$ to filter out all other candidates. However, modeling our LPN samples as true LPN samples is not always accurate as shown by Fig. 5.3.

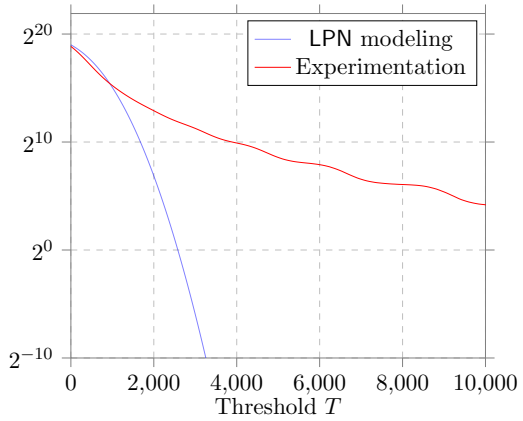


Figure 5.3: Experimental average number of false candidates $\left|\{\mathbf{x} \in \mathbb{F}_2^s : \mathbf{x} \neq \mathbf{e}_{\mathscr{P}} \text{ and } F^{\mathcal{L}}(\mathbf{x}) > T\}\right|$ against its LPN modeled counterpart. $n = 100$, $k = 30$, $s = 20$, $w = 6$.

Remark 14. *We give more details about this figure and the* LPN *model in Section 5.6.2. In particular, we give a description of the first version of* RLPN *that was published in [CDMT22] and that was analyzed using this* LPN *modeling. The key thing to notice is that in this figure we plot* $\left|\{\mathbf{x} \in \mathbb{F}_2^s : \mathbf{x} \neq \mathbf{e}_{\mathscr{P}} \text{ and } F^{\mathcal{L}}(\mathbf{x}) > T\}\right|$ *whereas in our algorithm the candidates in* $\mathcal{S}$ *are filtered by weight.*

However, using a conjecture, that we state later, we show that we can essentially distinguish $\mathbf{e}_{\mathscr{P}}$ from the other candidates at the cost of a slight polynomial strengthening on the condition that $N \geqslant 1/\delta_w^{(n-s)}(u)^2$. Our statement is given as follows.

Proposition 37 (Main proposition of this section)**.** *For any* $k, t, s, w, u \in \mathbb{N}$ *implicit functions of* $n \in \mathbb{N}$ *such that*

$$N \in \frac{\omega\left(n^6\right)}{\delta_w^{(n-s)}(u)^2}$$

and $u < \text{Root}\left(K_w^{(n-s)}\right)$ *and* $t \in \Theta(n)$ *and* $\binom{n}{t}/2^{n-k} \in \widetilde{\mathcal{O}}(1)$ *and* $t < n/2$ *then*

$$\mathbb{P}_{\mathcal{C}}(\mathbf{e}_{\mathscr{P}} \in \mathcal{S}) = 1 - o(1)$$

and, under Conjecture 1 we have that

$$C = \widetilde{\mathcal{O}}(1) \quad \text{where} \quad C \overset{def}{=} \mathbb{E}_{\mathcal{C}}(|\mathcal{S}| \mid \dim(\mathcal{C}) = k) \tag{5.4}$$

and where

$$T \stackrel{def}{=} N\delta_w^{(n-s)}(u)$$

and where the other quantities and distribution are defined in Notation 3 and where we recall that $N \stackrel{def}{=} \frac{\binom{n-s}{w}}{2^k}$.

5.3.1 Main linearity relations

Our analysis relies on the following rewriting of the LPN score function defined in the previous chapter. It comes from the linearity relations between the coordinates of $\mathbf{h}$ which allows us to write $\mathbf{h}_{\mathscr{P}}$ as a linear combination of $\mathbf{h}_{\mathscr{N}}$.

Lemma 15. *For any* $\mathbf{h} \in \mathscr{H}$ *and* $\mathbf{x} \in \mathbb{F}_2^s$ *we have that*

$$\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle = \langle (\mathbf{x} - \mathbf{e}_{\mathscr{P}})\mathbf{R} + \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle, \qquad \forall \mathbf{h} \in \mathcal{C}^{\perp}$$

where $\mathbf{R} \stackrel{def}{=} \text{Lift}\left(\mathcal{C}^{\perp}, \mathscr{N}\right)$ *is the unique* $\mathbb{F}_2^{s\times(n-s)}$ *matrix such that* $\mathbf{h}_{\mathscr{P}} = \mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}$ *for all* $\mathbf{h} \in \mathcal{C}^{\perp}$.

Proof. As $\text{rank}(\mathcal{C}_{\mathscr{P}}) = s$, from Lemma 1, $\mathscr{N}$ is an information set of $\mathcal{C}^{\perp}$ thus $\text{Lift}\left(\mathcal{C}^{\perp}, \mathscr{N}\right)$ is well-defined. Finally

$$\begin{aligned}
\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle &= \langle \mathbf{e}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle \\
&= \langle \mathbf{e}_{\mathscr{P}} - \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle \\
&= \langle \mathbf{e}_{\mathscr{P}} - \mathbf{x}, \mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle \\
&= \langle (\mathbf{x} - \mathbf{e}_{\mathscr{P}})\mathbf{R} + \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle .
\end{aligned}$$

□

Notation 4. *We define*

$$r(\mathbf{x}) \stackrel{def}{=} (\mathbf{x} + \mathbf{e}_{\mathscr{P}})\mathbf{R} + \mathbf{e}_{\mathscr{N}}.$$

This, by the way, gives a very simple explanation to the fact that the LPN model does not hold. Indeed, $\mathbf{h}_{\mathscr{N}}$ lies in an $[n-s, n-k]$ linear code, $\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\perp}\right)_{\mathscr{N}} = \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}$. It is readily seen that the inner product on the right-hand side of the previous equality is biased toward 0 as soon as $r(\mathbf{x})$ is close to a codeword of $\mathbf{c}^{\mathscr{N}} \in \mathcal{C}^{\mathscr{N}}$ as $\left\langle \mathbf{c}^{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \right\rangle = 0$. This by the way gives some simple explanation to the bumps we see in Fig. 5.9b, each corresponds to some $\mathbf{x}$ which are such that $r(\mathbf{x})$ is unusually close to $\mathcal{C}^{\mathscr{N}}$. All in all we can rewrite the score function as follows (for our convenience we add the distribution of each quantity).

Proposition 38 (Distribution of the LPN score function using a related score function)**.** *For any* $\mathbf{x} \in \mathcal{S}_{l-u}^s$ *we have that*

$$F^{\mathcal{L}}(\mathbf{x}) = F_{\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp} \bigcap \mathcal{S}_w^{n-s}}(r(\mathbf{x})), \qquad \text{where } F_{\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp} \bigcap \mathcal{S}_w^{n-s}}(r(\mathbf{x})) \stackrel{def}{=} \sum_{\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp} \bigcap \mathcal{S}_w^{n-s}} (-1)^{\langle \mathbf{h}_{\mathscr{N}}, r(\mathbf{x}) \rangle}$$

Moreover we have that $\mathcal{C}^{\mathscr{N}} \sim \mathcal{U}_{\mathbf{G}}(n-s,\ k-s)$ *and for any fixed* $\mathbf{x}$ *that*

$$\begin{aligned}
r(\mathbf{x}) &\sim \mathcal{U}\left(\mathbb{F}_2^{n-s}\right) \text{ if } \mathbf{x} \neq \mathbf{e}_{\mathscr{P}} \\
r(\mathbf{e}_{\mathscr{P}}) &= \mathbf{e}_{\mathscr{N}}.
\end{aligned}$$

Proof. The distributions follow from Fact 7. □

5.3.2 The minimal condition on the number of needed dual vectors

This last proposition is somewhat enlightening to devise say a minimal set of conditions such that our reduction is sound. Indeed, let us forget for now our algorithm and that we want to find $\mathbf{e}_{\mathscr{P}}$ among a great number of $\mathbf{x} \in \mathcal{S}^s_{t-u}$. We should at least be able to distinguish say $\mathbf{x}_1 = \mathbf{e}_{\mathscr{P}}$ and $\mathbf{x}_2 \neq \mathbf{e}_{\mathscr{P}}$ with polynomial probability. Recalling that we supposed that the bet on the error was valid, namely that $|\mathbf{e}_{\mathscr{N}}| = u$ we have by using the concentration bound on the score function we devised in the previous chapter that the following holds.

Corollary 6 (Of Proposition 38 and Proposition 31)**.** *For any positive function f we have that*

$$\mathbb{P}_{\mathcal{C}}\left(\left|F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\right) - N\delta_w^{(n-s)}\left(u\right)\right| \geqslant f(n)\sqrt{N}\right) \leqslant \frac{1}{f(n)}$$
$$\mathbb{P}_{\mathcal{C}}\left(\left|F^{\mathcal{L}}\left(\mathbf{x}\right)\right| \geqslant f(n)\sqrt{N}\right) \leqslant \frac{1}{f(n)} \qquad \text{if } \mathbf{x} \neq \mathbf{e}_{\mathscr{P}}.$$

Moreover, $N\delta_w^{(n-s)}(u)$ is the expected value of $F^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}})$ and 0 is the expected value of $F^{\mathcal{L}}(\mathbf{x})$. We used Notation 3.

This allows us to conclude that $F^{\mathcal{L}}(\mathbf{x}_1)$ and $F^{\mathcal{L}}(\mathbf{x}_2)$ are sufficiently concentrated around their expectation such that we can distinguish them with probability $1 - o(1)$ as soon as $N = \omega(1)/\delta_w^{(n)}(u)^2$, namely

Corollary 7. *For any k, t, s, w, u implicit functions of $n \in \mathbb{N}$ such that*

$$N \in \omega(1)/\delta_w^{(n)}(u)^2$$

we have that for any fixed $\mathbf{x} \in \mathcal{S}^s_{t-u}$

$$\mathbb{P}_{\mathcal{C}}\left(\left|F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\right)\right| \geqslant \frac{1}{2}N\delta_w^{(n-s)}\left(u\right)\right) = 1 - o(1)$$
$$\mathbb{P}_{\mathcal{C}}\left(\left|F^{\mathcal{L}}\left(\mathbf{x}\right)\right| \geqslant \frac{1}{2}N\delta_w^{(n-s)}\left(u\right)\right) = o(1) \qquad \text{if } \mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$$

and where we recall that $N \overset{def}{=} \frac{\binom{n-s}{w}}{2^{k-s}}$ and where we used Notation 3.

Our intuition is that if N is smaller than this quantity we have essentially no advantage to distinguish $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ from $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$. This would in turn make our reduction useless: depending on the threshold T this leads to either $\mathcal{S}$ containing every $\mathbf{x} \in \mathcal{S}^s_{t-u}$ or to it being completely empty.

5.3.3 Conjecture : the need for stronger concentration bounds

Now of course our second order concentration bounds of Corollary 6 are completely insufficient to prove anything useful about our ability to distinguish $\mathbf{e}_{\mathscr{P}}$ among an exponential number of $\mathbf{x} \in \mathcal{S}^s_{t-u}$. Our goal however is to stay tight to the previously devised minimal condition on $N = \omega(1)/\delta_w^{(n-s)}(u)^2$. Say for simplicity that we keep the threshold as before and count the number of false candidates we get, that is:

Lemma 16. *Let $\mathbf{x} \in \mathbb{F}_2^s$ be any fixed vector such that $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$. We have that the expected number of candidates is*

$$\mathbb{E}\left(|\mathcal{S}| \mid \dim(\mathcal{C}) = k\right) \leqslant \binom{s}{t-u} \mathbb{P}\left(F^{\mathcal{L}}(\mathbf{x}) > \frac{1}{2} N \delta_w^{(n-s)}(u) \,\middle|\, \dim(\mathcal{C}) = k\right) + 1.$$

where we used Notation 3.

Proof. We have that $|\mathcal{S}| = \sum_{\mathbf{z} \in \mathcal{S}_{t-u}^s} \mathbf{1}_{\mathbf{z} \in \mathcal{S}} \leqslant 1 + \sum_{\mathbf{z} \in \mathcal{S}_{t-u}^s : \mathbf{z} \neq \mathbf{e}_{\mathscr{P}}} \mathbf{1}_{\mathbf{z} \in \mathcal{S}}$, thus by linearity of the expectation we get

$$\mathbb{E}(|\mathcal{S}|) \leqslant 1 + \sum_{\mathbf{z} \in \mathcal{S}_{t-u}^s : \mathbf{z} \neq \mathbf{e}_{\mathscr{P}}} \mathbb{P}(\mathbf{z} \in \mathcal{S}) = 1 + \sum_{\mathbf{z} \in \mathcal{S}_{t-u}^s : \mathbf{z} \neq \mathbf{e}_{\mathscr{P}}} \mathbb{P}\left(\left|F^{\mathcal{L}}(\mathbf{z})\right| > T\right) = 1 + \binom{s}{t-u} \mathbb{P}\left(\left|F^{\mathcal{L}}(\mathbf{x})\right| > T\right)$$

we used that from Proposition 38, the distribution of $r(\mathbf{x})$ does not depend on $\mathbf{x}$ when $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$. Conditioning by $\dim(\mathcal{C}) = k$ does not change the argument. □

Usually this is where the LPN modeling comes at play but as we cannot use it here we make a conjecture which is essentially the best exponential strengthening of our second order concentration bound we could hope for.

It is readily seen that $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ will be mistaken for $\mathbf{e}_{\mathscr{P}}$ if for example $r(\mathbf{x}) \stackrel{\text{def}}{=} (\mathbf{x} + \mathbf{e}_{\mathscr{P}})\mathbf{R} + \mathbf{e}_{\mathscr{N}}$ is unusually close to the code $\mathcal{D} = \mathcal{C}^{\mathscr{N}}$, say it is at distance at most $u + o(n)$ (in this case it is readily seen that $F^{\mathcal{L}}(\mathbf{x}) \approx N \delta_w^{(n-s)}(u)$). We simply make the conjecture that this is essentially the only case where this happens. The probability that such an event happens can easily be bounded by $\widetilde{\mathcal{O}}\left(\binom{n-s}{u}/2^{n-k}\right)$. All in all and forgetting about the fact that we are on $\mathcal{C}^{\mathscr{N}}$ to simplify we make the following conjecture.

Conjecture 1. *For any k, t, w implicit functions of $n \in \mathbb{N}$ such that*

$$\frac{\binom{n}{w}}{2^k} \in \frac{\omega\left(n^6\right)}{\delta_w^{(n)}(t)^2}$$

and $t < \mathrm{Root}\left(K_w^{(n)}\right)$ and $t \in \Theta(n)$ and $t < n/2$ then

$$\mathbb{P}\left(|F_{\mathscr{H}}(\mathbf{z})| \geqslant \frac{1}{2} \frac{\binom{n}{w}}{2^k} \delta_w^{(n)}(t)\right) = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{2^{n-k}}\right)$$

where $\mathcal{C} \sim \mathcal{U}(n,\ k)$ and $\mathbf{z} \sim \mathcal{U}(\mathbb{F}_2^n)$ and $\mathscr{H} \stackrel{def}{=} \mathcal{C}^{\perp} \bigcap \mathcal{S}_w^n$ and $F_{\mathscr{H}}(\mathbf{g}) \stackrel{def}{=} \sum_{\mathbf{h} \in \mathscr{H}} (-1)^{\langle \mathbf{g}, \mathbf{h} \rangle}$, $\forall \mathbf{g} \in \mathbb{F}_2^n$.

We give heuristics that this conjecture is true in Section 5.4.

Remark 15. *This conjecture is really a strengthening of the second case of Proposition 31. Indeed, using the condition on $N' \stackrel{def}{=} \frac{\binom{n}{w}}{2^k}$ in Conjecture 3 one can replace $N' \delta_w^{(n)}(t)$ by $\sqrt{f(n)}\sqrt{N'}$ thus the conjecture claims that there exists f and g poly bounded such that:*

$$\mathbb{P}\left(|F_{\mathscr{H}}(\mathbf{z})| \geqslant \frac{\sqrt{f(n)}}{2}\sqrt{N'}\right) \leqslant g(n) \frac{\binom{n}{t}}{2^{n-k}}.$$

Instead of having a polynomial bound (i.e. say $\frac{\sqrt{f(n)}}{2}$ here we rather conjecture that we get an exponential advantage as soon as $t < d_{\mathrm{GV}}(n,\ k)$.

5.3.4 Proof

We are now ready to prove our main proposition.

Proof of Proposition 37 . The fact that $\mathbf{e}_{\mathscr{P}} \in \mathcal{S}$ with probability $1 - o(1)$ directly follows from the second-order concentration bounds given in Corollary 7. The bound on the expected number is a direct corollary of Lemma 16 and Conjecture 1. We can indeed use Conjecture 1 to upper bound the right-hand side of Lemma 16 by using the fact that if $\mathcal{C}$ is distributed according to $\mathcal{U}_{\mathbf{G}}(n,\ k)$ conditioned on the event that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and on the event that $\dim(\mathcal{C}) = k$ then $\mathcal{C}^{\mathscr{N}} \sim \mathcal{U}(n-s,\ k-s)$. As such we get by using Conjecture 1 that

$$\begin{aligned}
\mathbb{E}\left(|\mathcal{S}| \mid \dim(\mathcal{C}) = k\right) &\leqslant 1 + \binom{s}{t-u}\mathbb{P}\left(\left|F^{\mathcal{L}}(\mathbf{z})\right| > T \;\middle|\; \dim(\mathcal{C}) = k\right) \\
&\leqslant 1 + \binom{s}{t-u}\widetilde{\mathcal{O}}\left(\frac{\binom{n-s}{u}}{2^{n-k}}\right) \\
&\leqslant 1 + \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{2^{n-k}}\right) \\
&= \widetilde{\mathcal{O}}(1)
\end{aligned}$$

where the last line follows by assumption in the proposition. □

5.4 Precise behavior of the score function, veryfing the conjecture

Recall that a key step of our analysis in the last section relies on estimating the number of false candidates, which as we showed can be reduced to proving some tail bounds for the score function, namely we used Conjecture 1 that we state again here.

Conjecture 1. *For any k, t, w implicit functions of $n \in \mathbb{N}$ such that*

$$\frac{\binom{n}{w}}{2^k} \in \frac{\omega\left(n^6\right)}{\delta_w^{(n)}(t)^2}$$

and $t < \mathrm{Root}\left(K_w^{(n)}\right)$ and $t \in \Theta(n)$ and $t < n/2$ then

$$\mathbb{P}\left(|F_{\mathscr{H}}(\mathbf{z})| \geqslant \frac{1}{2}\frac{\binom{n}{w}}{2^k}\delta_w^{(n)}(t)\right) = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{2^{n-k}}\right)$$

where $\mathcal{C} \sim \mathcal{U}(n,\ k)$ and $\mathbf{z} \sim \mathcal{U}(\mathbb{F}_2^n)$ and $\mathscr{H} \stackrel{def}{=} \mathcal{C}^{\perp} \bigcap \mathcal{S}_w^n$ and $F_{\mathscr{H}}(\mathbf{g}) \stackrel{def}{=} \sum_{\mathbf{h}\in\mathscr{H}}(-1)^{\langle \mathbf{g},\mathbf{h}\rangle}$, $\forall \mathbf{g} \in \mathbb{F}_2^n$.

In this section we want to give compelling evidence that this conjecture is indeed true and we do that by showing that the distribution of the score function is precisely described with a simple model which implies our conjecture.

5.4.1 Duality formula and the poisson model

Our main tool to study the distribution of $F_{\mathscr{H}}(\mathbf{z})$ is the following dual expression for the score function expressing $F_{\mathscr{H}}(\mathbf{z})$ using the weight enumerator of the coset code $\mathcal{C}+\mathbf{z}$.

Proposition 39 (Main duality tool)**.** *Let $n,k,w\in\mathbb{N}$. Let $\mathcal{C}\in\mathfrak{C}[n,k]$ and $\mathscr{H}=\mathcal{C}^{\perp}\bigcap\mathcal{S}_w^n$ and let $\mathbf{z}\in\mathbb{F}_2^n$. We have*

$$F_{\mathscr{H}}(\mathbf{z})=\sum_{i=0}^{n}N_i(\mathcal{C}+\mathbf{z})\frac{K_w^{(n)}(i)}{2^k} \tag{5.5}$$

where we recall that $N_i(\mathcal{C}+\mathbf{z})$ is the weight enumerator of $\mathcal{C}+\mathbf{z}$, namely

$$N_i(\mathcal{C}+\mathbf{z})\overset{def}{=}\left|(\mathcal{C}+\mathbf{z})\bigcap\mathcal{S}_i^n\right|.$$

Proof. We have that

$$\begin{aligned}F_{\mathscr{H}}(\mathbf{z})&=\sum_{\mathbf{h}\in\mathcal{C}^{\perp}\bigcap\mathcal{S}_w^n}(-1)^{\langle\mathbf{z},\mathbf{h}\rangle}\\&=\sum_{\mathbf{h}\in\mathcal{C}^{\perp}}(-1)^{\langle\mathbf{z},\mathbf{h}\rangle}\mathbf{1}_{\mathbf{h}\in\mathcal{S}_w^n}\\&=\frac{1}{2^k}\sum_{\mathbf{c}\in\mathcal{C}}j(\mathbf{c})\qquad\text{(Poisson summation)}\end{aligned}$$

where the function j is the Fourier transform of the following function $\mathbf{h}\to(-1)^{\langle\mathbf{z},\mathbf{h}\rangle}\mathbf{1}_{\mathbf{h}\in\mathcal{S}_w^n}$. As such we have that

$$\begin{aligned}j(\mathbf{c})&=\sum_{\mathbf{h}\in\mathbb{F}_2^n}\mathbf{1}_{\mathbf{h}\in\mathcal{S}_w^n}(-1)^{\langle\mathbf{z}+\mathbf{c},\mathbf{h}\rangle}\\&=\sum_{\mathbf{h}\in\mathcal{S}_w^n}(-1)^{\langle\mathbf{z}+\mathbf{c},\mathbf{h}\rangle}\\&=K_w^{(n)}(|\mathbf{c}+\mathbf{h}|).\end{aligned}$$

□

The above dual formula allows reducing the study of the distribution of $F_{\mathscr{H}}(\mathbf{z})$ to the study of the distribution of the weight enumerator $N_i(\mathcal{C}+\mathbf{z})$. The problem is that current knowledge of this distribution is insufficient to prove our conjecture, see Section 5.4.3.2 for more details. This leads us to make the weight enumerator following a Poisson variable of the right expected value, see Section 5.4.3.3 for more details.

Model 3 (Poisson model)**.** *Let $\mathcal{C}\sim\mathcal{U}(n,\ k)$ and let $\mathbf{z}\sim\mathcal{U}(\mathbb{F}_2^n)$. We make the model that for any $i\in[\![0,\ n]\!]$*

$$N_i(\mathcal{C}+\mathbf{z})\sim\text{Poisson}\left(\frac{\binom{n}{i}}{2^{n-k}}\right)$$

5.4.2 Heuristic argument that the conjecture is valid

The key thing is that making this model allows us proving Conjecture 1 while observing experimentally that it keeps the distribution of $F_{\mathscr{H}}(\mathbf{z})$ unchanged, namely we will get the following proposition along with Fig. 5.4.

Proposition 40. *Under Model 3, Conjecture 1 is true.*

Proof. The proof is given in Section 5.6.1. □

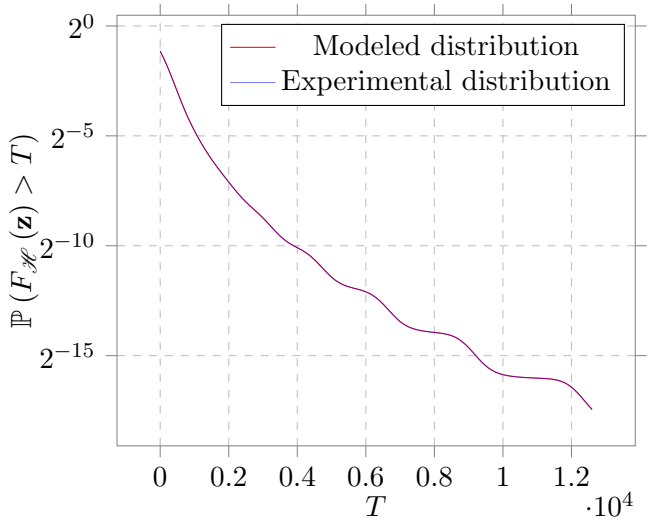


Figure 5.4: Experimental validity of the Poisson model (Model 3). We plot the distribution of the score function $F_{\mathscr{H}}(\mathbf{z})$ against the modeled distribution of $F_{\mathscr{H}}(\mathbf{z})$ under the Poisson model. More precisely, in the first case we drew the distribution of $F_{\mathscr{H}}(\mathbf{z})$ where $\mathcal{C} \sim \mathcal{U}(n,\ k)$ and $\mathbf{z} \sim \mathcal{U}(\mathbb{F}_2^n)$ and in the second case we drew the distribution of $F_{\mathscr{H}}(\mathbf{z})$ when replacing the weight enumerator by Poisson variables given by the model, namely we drew $\sum_{i=0}^{n} \mathbf{X}_i K_w^{(n)}(i)/2^k$ where $\mathbf{X}_i \sim \text{Poisson}\left(\binom{n-s}{i}/2^{n-k}\right)$ are independent Poisson variables. Here $n = 100,\ k = 30,\ s = 30,\ w = 6$

Remark 16. *Both curve in Fig. 5.4 were obtained with Monte-Carlo methods. Namely, to plot the experimental distribution of the score function we drew multiple codes $\mathcal{C}$ and $\mathbf{z}$ according to $\mathcal{U}(n,\ k)$ and $\mathcal{U}(\mathbb{F}_2^n)$ and plotted the resulting experimental distribution. The modeled distribution of the score function was obtained by drawing multiple score functions according to the distribution of the conjecture. Finally, in the modeled distribution the Poisson variables are taken independently so that we can actually estimate numerically the distribution of F but note that this independence is not required for the proof of Proposition 40.*

5.4.3 Discussion

5.4.3.1 Interpreting the conjecture in light of the duality formula

Recall that in Corollary 7 we showed that if $\mathbf{y}$ is say at distance t from $\mathcal{C}$ and as long as $N = \omega(1)/\delta_w^{(n)}(t)^2$ then with probability $1 - o(1)$ we have that

$$F_{\mathscr{H}}(\mathbf{y}) = N\delta_w^{(n)}(t)(1 + o(1)).$$

Our duality formula can be rewritten as

Corollary 8 (Corollary of Proposition 39)**.** *Let* $N \stackrel{def}{=} \binom{n}{w}/2^k$ *we have that*

$$F_{\mathscr{H}}(\mathbf{z}) = \sum_{i=0}^{n} N_i(\mathcal{C} + \mathbf{z}) N\delta_w^{(n)}(i).$$

Thus

$$F_{\mathscr{H}}(\mathbf{y}) = N\delta_w^{(n)}(t) + \sum_{i=0}^{n} N_i(\mathcal{C} \setminus \{0\} + \mathbf{y}) N\delta_w^{(n)}(t).$$

which is seen to be dominated by this first term at least with probability $1 - o(1)$. Essentially our conjecture implies that this is in fact the case with probability $2^{-\Omega(n)}$. A maybe more direct reformulation is that our conjecture says that when $\mathbf{z} \sim \mathcal{U}_{\mathbf{G}}(n,\ k)$ the event "$F_{\mathscr{H}}(\mathbf{z})$ is very big (say superior to $N\delta_w^{(n)}(t)$)" is completely dominated by the existence of an unusually low weight codeword in the coset, say for example $N_t(\mathcal{C} + \mathbf{z}) \neq 0$ and that the other terms the sum do not come into play here.

5.4.3.2 Why we need the Poisson model

Problematically, our proof attempts of this conjecture somewhat reduces to proving exponential concentration bounds on the weight enumerator of random linear codes. More precisely we show in the next section that it is sufficient to prove, when $i > d_{\mathrm{GV}}(n,\ k)$ (namely, when $\frac{\binom{n}{i}}{2^{n-k}} \in 2^{\Theta(n)}$) that there exists some positive poly-bounded function f such that we have say

$$\mathbb{P}\left(|N_i(\mathcal{C} + \mathbf{z}) - \mathbb{E}(N_i(\mathcal{C} + \mathbf{z}))| > f(n)\sqrt{\mathbf{Var}(N_i(\mathcal{C} + \mathbf{z}))}\right) = 2^{-\Omega(n)}\ ? \tag{5.6}$$

We do not know if such bound are true and could be proven. Interestingly enough it could be achieved using Markov style inequalities by tightly bounding some higher order central moments of the weight enumerator (contrary to simply use the second order central moment gives us only a polynomial bound), but, a recent line of work [LM20, Sam24] proved that, when $\mathcal{D} \sim \mathcal{U}_{\mathbf{H}}(n,\ k)$ the higher central moments (from say $m > 4$) $\mathbb{E}\left((N_i(\mathcal{D}) - \mathbb{E}(N_i(\mathcal{D})))^m\right)$ are order of magnitude dramatically bigger than $\mathbf{Var}(N_i(\mathcal{D}))^{m/2}$ so we could nto use their result to prove anything toward this.

5.4.3.3 Rationale behind the Poisson model

Note that a somewhat slightly more natural model would be to model $N_i(\mathcal{C} + \mathbf{z})$ as a Binomial distribution. Indeed, recall that we can write the weight enumerator as $N_i(\mathcal{C} + \mathbf{z}) =$

$\sum_{\mathbf{r}\in\mathcal{S}_i^n} \mathbf{1}_{\mathbf{r}\in\mathcal{C}+\mathbf{z}}$. Supposing that the $\mathbf{1}_{\mathbf{r}\in\mathcal{C}+\mathbf{z}}$ are mutually independent variables we obtain that $N_i(\mathcal{C}+\mathbf{z})$ follows Binomial $\left(\binom{n}{i},\ 1/2^{n-k}\right)$ which is of expected value $\binom{n}{i}/2^{n-k}$. Note that the Binomial distribution converges to the Poisson distribution with the same expected value. And, because we find that the Poisson distribution is easier to handle, we will make the model that the weight enumerator follows a Poisson variable. Notably both variable share the first two moments.

Fact 19. *Let $\mathcal{C}\sim\mathcal{U}(n,\ k)$, let $\mathbf{z}\sim\mathcal{U}(\mathbb{F}_2^n)$ and $i\in[\![1,\ n]\!]$ and $\mathbf{X}\sim$ Poisson $\left(\binom{n}{i}/2^{n-k}\right)$, then we have that*

$$\mathbb{E}(N_i(\mathcal{C}+\mathbf{z})) = \frac{\binom{n}{i}}{2^{n-k}}, \qquad \mathbf{Var}(N_i(\mathcal{C}+\mathbf{z})) \leqslant \frac{\binom{n}{i}}{2^{n-k}}$$
$$\mathbb{E}(\mathbf{X}) = \frac{\binom{n}{i}}{2^{n-k}}, \qquad \mathbf{Var}(\mathbf{X}) = \frac{\binom{n}{i}}{2^{n-k}}$$

5.5 Main theorem and results

5.5.1 Main theorem

Because the number of false candidate is poly bounded in our case it will be beneficial, to simplify the analysis to use our proved variant of statistical decoding Algorithm 2 in our last decoding step: indeed by reusing the already produces dual vectors, we are guaranteed that the complexity of checking the false candidates will not dominate. And the choice of parameters for RLPN guarantees that this decoder succeed with sufficiently good probability.

Theorem 7. *For any $k,t,s,w,u,N_{\text{iter}},T\in\mathbb{N}$ implicit functions of a parameter $n\in\mathbb{N}$ (n is growing to infinity) and any procedure* Compute-Dual-Vectors *that are such that*

1. *(Computing all the dual vectors)*

$$\mathbb{P}_{\mathcal{D}\sim\mathcal{U}_{\mathbf{G}}(n-s,\,k-s)}\left(\text{Compute-Dual-Vectors}(\mathcal{D}) = \mathcal{D}^{\perp}\bigcap\mathcal{S}_w^{n-s}\right)\in 1-o(1),$$

2. *(Main constraint that we have enough dual vectors)*

$$\frac{\binom{n-s}{w}}{2^{k-s}} \in \frac{\omega\left(n^6\right)}{\left(\delta_w^{(n-s)}(u)\right)^2},$$

3. *(Unimportant technical constraints) $u<\text{Root}\left(K_w^{(n-s)}\right)$ and $t<n/2$ and $t\in\Theta(n)$ and $\binom{n}{t}/2^{n-k}\in\widetilde{\mathcal{O}}(1)$ and $k-s\in\omega(1)$ and $n-k\in\omega(1)$.*

4. *(Auxiliary quantities) $N_{\text{iter}} = n\frac{\binom{n}{t}}{\binom{n-s}{u}\binom{s}{t-u}}$ and $T=\frac{1}{2}\delta_w^{(n-s)}(u)\frac{\binom{n-s}{w}}{2^{k-s}}$*

5. *The procedure* Sub-Decoder$(\mathcal{C}^{\mathscr{N}},\ \mathbf{y}',\ u)$ *is our proved variant of statistical decoding, namely Algorithm 2 where we reuse for the set of dual vectors $\mathscr{H}_{\mathscr{N}}$ which is computed in* RLPN *and choose the same threshold T.*

then, under Conjecture 1, there exists an algorithm (Algorithm 12 with the right stopping conditions) that solves $\mathrm{DP}_{\mathbf{G}}(n,k,t)$ *with probability* $1-o(1)$ *in time and memory*

$$\begin{aligned}\textbf{\textit{Time}} &= \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{\binom{n-s}{u}\binom{s}{t-u}}\left(2^{k_{\mathrm{aux}}}+T_{\mathrm{eq}}\right)\right),\\ \textbf{\textit{Memory}} &= \widetilde{\mathcal{O}}\left(\left(2^{k_{\mathrm{aux}}}+M_{\mathrm{eq}}\right)\right),\end{aligned}$$

and where T_{eq}, M_{eq} *are the time and memory complexity of* Compute-Dual-Vectors$(\mathcal{D},w)$ *and where we recall that* $\delta_w^{(n)}(t)$ *is defined in Definition 25 and* $\mathrm{Root}\left(K_w^{(n)}\right)$ *is defined in Definition 26.*

Proof. Concerning the correctness. There exists with probability $1-o(1)$ an iteration such that $\mathbf{e}_{\mathscr{N}}=u$ (choice of N_{iter}) and $\dim(\mathcal{C}_{\mathscr{P}})=s$ (because $k-s\in\omega(1)$) and $\mathscr{H}=\{\mathbf{h}\in\mathcal{C}^{\perp}\,:\,|\mathbf{h}_{\mathscr{N}}|=w\}$ (constraint on the procedure computing the vectors). Now applying the main Proposition 37 we get that in such an iteration we have with probability $1-o(1)$ that $\mathbf{e}_{\mathscr{P}}\in\mathcal{S}$. Clearly, because of the n^6 term in the main constraint $\frac{\binom{n-s}{w}}{2^{k-s}}\in\frac{\omega(n^6)}{\left(\delta_w^{(n-s)}(u)\right)^2}$ we get that the Sub-Decoder procedure succeeds in recovering the rest of the error $\mathbf{e}_{\mathscr{N}}$ with probability $1-o(1)$ (see Algorithm 2). Concerning the complexity, we stop an iteration if the size of $\mathcal{S}$ is greater than $n\,C$ where C is defined in Proposition 37. This allows us to bound the expected time complexity of the algorithm while it is readily seen using Markov's inequality that it does not undermine the probability of success of the algorithm (where we use the additional fact that $\dim(\mathcal{C})=k$ with probability $1-o(1)$ because $n-k\in\omega(1)$). □

5.5.2 Asymptotic expressions

Let us now give the asymptotic counterpart of Theorem 8.

Remark 17. *In what follows the relative quantities and non-relatice quantities are related as* $R=k/n$, $\tau=t/n$, $\sigma=s/n$, $\omega=w/n$, $\mu=u/n$.

Definition 35 (Asymptotic complexity exponent of RLPN)**.** *Let* $R\in]0,\,1[$ *and* $\tau\in]0,\,1/2[$, *such that* $\tau\leqslant\tau_{\mathrm{GV}}(R)$ *and let* α_{eq} *and* β_{eq} *be two bivariate functions (the complexity exponenent of the procedure to compute the dual vectors). For any* $\sigma\in]0,\,R[$ *and* $\omega\in]0,\,(1-\sigma)/2[$ *and* $\mu\in]0,\,1-\sigma[$ *that verify the constraints that*

- *(Main constraint)*

$$(1-\sigma)h\left(\frac{\omega}{1-\sigma}\right)-(1-\sigma)\geqslant-2(1-\sigma)\kappa\left(\frac{\omega}{1-\sigma},\,\frac{\mu}{1-\sigma}\right) \tag{5.7}$$

- *(Auxiliary constraint)*

$$\mu<1/2-\sqrt{\omega(1-\omega)} \tag{5.8}$$

we define the time and memory complexity exponent α_{RLPN} *and* β_{RLPN} *respectively as*

$$\alpha_{\mathsf{RLPN}}(R, \tau\,;\, \sigma,\, \omega,\, \nu; \alpha_{\mathrm{eq}}) \stackrel{def}{=} \pi + \max\left((1-\sigma)\,\alpha_{\mathrm{eq}}(\frac{R-\sigma}{1-\sigma}, \frac{\omega}{1-\sigma}),\, \sigma\right)$$
$$\beta_{\mathsf{RLPN}}(R, \tau\,;\, \sigma,\, \omega,\, \nu; \beta_{\mathrm{eq}}) \stackrel{def}{=} \max\left((1-\sigma)\,\beta_{\mathrm{eq}}(\frac{R-\sigma}{1-\sigma}, \frac{\omega}{1-\sigma}),\, \cdot\right)\sigma$$
$$\pi \stackrel{def}{=} h(\tau) - \sigma h\left(\frac{\tau-\mu}{\sigma}\right) - (1-\sigma)h\left(\frac{\mu}{1-\sigma}\right)$$

where we recall that $\kappa(,\,)$ *is defined in Corollary 5.*

Proposition 41 (Asymptotic performance of RLPN)**.** *For any constants* $R \in]0,1[$, $\tau \in]0, 1/2[$, $\sigma \in]0,\, R[$, $\omega \in]0,\, (1-\sigma)/2[$, $\mu \in]0,\, 1-\sigma[$ *that verify the constraints of Definition 35 and under Conjecture 1 we have that*

1. *(*RLPN *+ Dumer subroutine) there exists an algorithm that solves* $\mathrm{DP}_{\mathbf{G}}(n, \lfloor Rn \rfloor, \lfloor \tau n \rfloor)$ *in time and memory respectively*

   $$\boldsymbol{T} = \widetilde{\mathcal{O}}\left(2^{n\,\alpha_{\mathsf{RLPN}}(R,\tau\,;\,\sigma,\,\omega,\,\mu;\alpha_{dual-Dumer-routine})}\right), \qquad \boldsymbol{M} = \widetilde{\mathcal{O}}\left(2^{n\,\beta_{\mathsf{RLPN}}(R,\tau\,;\,\sigma,\,\omega,\,\mu;\beta_{dual-Dumer-routine})}\right)$$

   where we recall that $\alpha_{dual-Dumer-routine}$ *and* $\beta_{dual-Dumer-routine}$ *are bivariate functions defined in Proposition 13.*

2. *(*RLPN *+ BJMM subroutine) under the heuristic that the procedure given in Proposition 14 outputs all dual vectors of weight* w *with probability* $1-o(1)$ *then there exists an algorithm that solves* $\mathrm{DP}_{\mathbf{G}}(n, \lfloor Rn \rfloor, \lfloor \tau n \rfloor)$ *in time and memory respectively*

   $$\boldsymbol{T} = \widetilde{\mathcal{O}}\left(2^{n\,\alpha_{\mathsf{RLPN}}(R,\tau\,;\,\sigma,\,\omega,\,\mu;\alpha_{\text{dual-BJMM-routine}})}\right), \qquad \boldsymbol{M} = \widetilde{\mathcal{O}}\left(2^{n\,\beta_{\mathsf{RLPN}}(R,\tau\,;\,\sigma,\,\omega,\,\mu;\beta_{\text{dual-BJMM-routine}})}\right)$$

   where we recall that $\alpha_{\text{dual-BJMM-routine}}$ *and* $\beta_{\text{dual-BJMM-routine}}$ *are bivariate functions defined in Proposition 14.*

Remark 18. *The mild heuristic we have to use in the case of* RLPN *+ BJMM subroutine comes from the fact that we only proved in Proposition 14 that the BJMM subroutine outputs a proportion* $1-o(1)$ *of all dual vectors with probability* $1-o(1)$ *(and not all the dual vectors with probability* $1-o(1)$ *as required by our theorem). This is not really a concern for us.*

5.5.3 Results

In this section we optimized the complexity exponent of double-RLPN by choosing a technique to compute the dual vectors, fixing a rate R and the relative decoding distance τ and minimizing (over $\sigma,\, \omega,\, \mu$) the time complexity exponent α_{RLPN} under the constraint that

$$(1-\sigma)h\left(\frac{\omega}{1-\sigma}\right) - (1-\sigma) \geqslant -2(1-\sigma)\kappa\left(\frac{\omega}{1-\sigma},\, \frac{\mu}{1-\sigma}\right)$$

and checking afterwards that the obtained optimal parameters were indeed outside the root region, namely that $\tau < 1/2 - \sqrt{\omega(1-\omega)}$. The curves labelled Genie-Aided RLPN is an ideal scenario were we supposed that each dual vector of weight w on $\mathscr{N}$ can be computed in polynomial time. This gives a lower-bound on the complexity of our algorithm.

5.5.4 At Gilbert-Varshamov distance

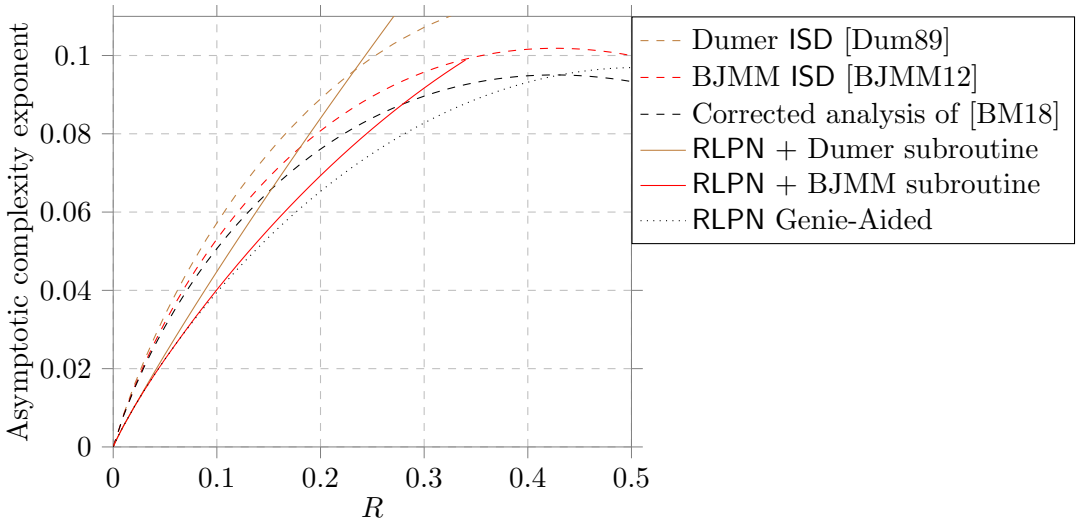


Figure 5.5: Asymptotic complexity exponent, relative to the codelength of some generic decoders when decoding codes of rate R at the Gilbert-Varshamov distance, namely $\tau = \tau_{\mathrm{GV}}(R)$

Remark 19. *The complexity exponent of* RLPN *could be further computed for higher rates but we explain in Section 5.5.6.1 why we have decided to stop it at this specific rate ($R \approx 0.35$ for* RLPN *+ BJMM)*

Interestingly one can notice that the complexities of the ISD's and RLPN when using the same subroutine, say [BJMM12] compared to RLPN + BJMM are not symmetric. We observe that RLPN + BJMM at rate R is worse than the complexity of ISD [BJMM12] at rate $1.0 - R$. We believe that this phenomenon is a consequence of the relatively bad behavior of RLPN in the extremely low rate setting that we mention later.

5.5.4.1 Shape of the parameters

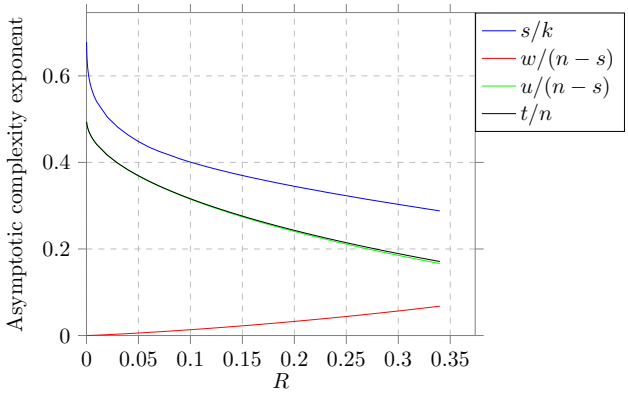


Figure 5.6: Asymptotic parameters,relative to the codelength of RLPN + BJMM

We noticed afterwards that it seems like for our parameters of interest here the gain for making the bet here is extremely limited if not completely null. Indeed, one can see in Fig. 5.6 that the t/n and $u/(n-s)$ are almost equal.

5.5.5 Gaining a square root factor in the low rate regime $R \approx 0.01$

The next Fig. 5.7 show the behavior of our algorithm in the low rate regime. We see that for relatively low rate say $R \in [0.01, 0.02]$ RLPN, regardless if we use Dumer or BJMM subroutine, achieves a complexity of around $\sqrt{2^{R\,n}}$. This beats all the current ISD's which have a complexity just slightly smaller than $2^{R\,n}$.

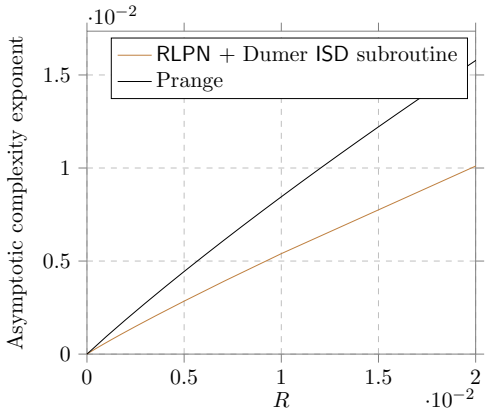


Figure 5.7: Asymptotic complexity exponent, relative to the codelength n of some generic decoders when decoding codes of rate R at the Gilbert-Varshamov distance in the very low rate regime.

5.5.5.1 When the rate tends to 0 the complexity of RLPN is that of the Prange decoder

Recall that for ISD's, the introduction of collision in Dumer [Dum86, Dum91] allowed to gain a square root over the Prange decoder when the rate R tends to 1, going from $2^{(1-R)\,n+o(n)}$ for Prange to $\sqrt{2^{(1-R)n+o(n)}}$ for Dumer. Here we could have expected some kind of dual phenomenon where when R tends toward 0 the complexity of RLPN would gain a square root factor over the Prange decoder, especially in light of the previous Fig. 5.7. However, it is not the case. Indeed we observe in Fig. 5.8 that the complexity of RLPN + Dumer tends, slowly, to the complexity of the naive enumeration, namely some 2^{Rn} when R grows to 0. This stays true even for Genie-Aided RLPN. It remains an open question to understand how we could mitigate this phenomenon and really gain a square root over Prange.

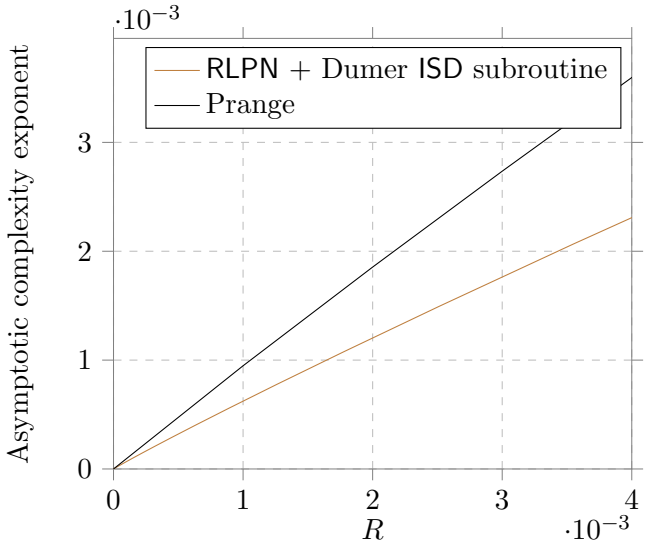


Figure 5.8: Asymptotic complexity exponent, relative to the codelength n of some generic decoders when decoding codes of rate R at the Gilbert-Varshamov distance in the very low rate regime.

5.5.6 Additional results

5.5.6.1 When using an iteration of ISD BJMM

In the case of RLPN it makes sense in the higher rate regime to use a better ISD than the BJMM subroutine. Notably in the first version of the algorithm, in [CDMT22, Section 5], we tried to use a full-fledged iteration of the ISD BJMM [BJMM12] to produce low-weight dual vectors on $\mathscr{N}$, see Section 4.3.3 for more details. The crucial point is that we perform a polynomial filter on the part of the dual vector that is random and intervenes in the error of the LPN sample. However, this strategy was never proven without using the flawed LPN model, and we noticed that, as long as the rate was smaller than 0.35, no difference in performance could be seen between this strategy and simply taking only the BJMM subroutine. The advantage appears only in the higher rate regimes. Because this is already a zone where the best ISD outperforms RLPN, there is no point complicating the algorithm and analysis here.

5.5.6.2 In the sublinear regime

It is easy to see that, when the rate R is constant and the error weight is sublinear in n, namely $t = o(n)$ then, to be remotely competitive and because the cost of the FFT is some $\mathcal{O}(s2^s)$ we have no choice but to take $s = \mathcal{O}(t)$ and thus $s = o(n)$. Consequently, the code $\mathcal{C}^{\mathscr{N}}$ in which we compute the short dual vectors is of length $n - o(n)$ and rate $R - o(n)$ and thus the remark that the only reasonable choice to compute the dual vectors is to use a variation of the Prange decoder applies. In this setting we claim that we could prove that the complexity of RLPN is essentially as bad as the complexity of Statistical decoding in the sublinear regime (see Section 4.3.2 and [DT17a]) namely the complexity would be the square of the Prange decoder, that is

$$2^{-2t\log_2(1-R)}.$$

5.6 Appendices

5.6.1 Proof that the Poisson model implies our conjecture

We prove here Proposition 40. Let us rewrite what we want to prove in a simpler manner.

Conjecture 2 (Rewriting of Conjecture 1)**.** *For any $k, t \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that $t \in \Theta(n)$ and $t < n/2$ and $t < \mathrm{Root}\left(K_w^{(n)}\right)$ and*

$$K_w^{(n)}(t) \in \omega\left(n^3\right)\sqrt{2^k\binom{n}{w}} \tag{5.9}$$

then we conjecture that

$$\mathbb{P}\left(\sum_{i=0}^{n} N_i K_w^{(n)}(i) > K_w^{(n)}(t)\right) = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{2^{n-k}}\right).$$

where $\mathcal{C} \sim \mathcal{U}(n,\ k)$ and $\mathbf{z} \sim \mathcal{U}(\mathbb{F}_2^n)$ and where we recall that $N_i \overset{def}{=} N_i(\mathcal{C} + \mathbf{z})$.

Fact 20. *Conjecture 1 is true if and only if Conjecture 2 is true.*

Proof. This is straightforwardly done by using the duality formula on the score function (Proposition 34) and expanding the quantities in Conjecture 1 to obtain Conjecture 2. □

Consequently, to prove Proposition 40 we only have left to prove the following proposition.

Proposition 42. *Under Model 3 we have that Conjecture 2 is true.*

Proving this proposition is the goal of the rest of this section. We only give the intermediate lemmas for now and prove them later.

Problematically we first observe that bounding each term of the sum directly is really not an option as essentially the innermost terms are already way too big. Indeed, due to the fact that the Krawtchouk polynomial $K_w^{(n)}$ achieves point-wise its L^2 norm between each of its root, we can show easily that there exists $i = n/2 + o(n)$ such that

$$\mathbb{E}(N_i) K_w^{(n)}(i) = 2^{o(n)}\sqrt{2^k}\sqrt{2^k\binom{n}{w}}$$

which is already exponentially bigger than our target

$$K_w^{(n)}(t) = \omega\left(n^6\right)\sqrt{2^k\binom{n}{w}}.$$

Consequently, in order look at each term of the sum individually, we introduce a key centering lemma making use of the orthogonality property of Krawtchouk polynomials and which shows that in the previous sum we can replace the weight enumerator by its centered (around its expectation) counterpart, namely $N_i - \overline{N_i}$, allowing to gain a huge factor.

Lemma 17 (Key centering lemma)**.** *We have that*

$$\sum_{i=0}^{n} N_i K_w^{(n)}(i) = \sum_{i=0}^{n} \left(N_i - \overline{N_i}\right) K_w^{(n)}(i)$$

where $\overline{N_i}$ is the expected value of N_i, namely $\overline{N_i} = \frac{\binom{n}{i}}{2^{n-k}}$.

Proof. This is a direct consequence of Proposition 19 as

$$\begin{aligned}\sum_{i=0}^{n} \overline{N_i} K_w^{(n)}(i) &= \frac{1}{2^{n-k}}\left\langle K_w^{(n)}; K_0^{(n)}\right\rangle \\ &= 0\end{aligned}$$

□

Now we can really look at each term individually with the following.

Lemma 18.

$$\mathbb{P}\left(\sum_{i=0}^{n}\left(N_i - \overline{N_i}\right)K_w^{(n)}(i) > \frac{1}{2}K_w^{(n)}(t)\right) \leqslant n \max_{i=0\cdots n}\mathbb{P}\left(\left|N_i - \overline{N_i}\right| > \frac{1}{2\,n}R_i\right)$$

where

$$R_i \stackrel{def}{=} \left|\frac{K_w^{(n)}(t)}{K_w^{(n)}(i)}\right|.$$

Proof. The proof is straightforward using the triangular inequality along with the bound that $\sum_{i=0}^{n} x_i \leqslant n \max x_i$. □

We now split the indexes in two parts. The first one is composed of indexes essentially smaller than t and which accounts for the rare events where the coset would have very low weight codeword (i.e. the bumps we see on the right hand side of Fig. 5.4). The second one is composed of indexes bigger than t and accounts for the typical "normal like behavior" of the sum, this is where we use the Poisson model (i.e. this accounts for the left hand side of Fig. 5.4).

Bounding these probabilities crucially rely on our parameter constraints which allows relating R_i to the variance of N_i:

Lemma 19. *For any $k, t \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that Eq. (5.9) holds then we have that*

$$R_i \in \omega\left(n^3\right)\sqrt{\overline{N_i}} \quad \forall i \in [\![0,\, n]\!].$$

This allows us to get the following bounds.

Definition 36. *Define the dominant set $\mathcal{D}$ as*

$$\mathcal{D} \stackrel{def}{=} \{i \in [\![1,\, n]\!] \,:\, R_i \leqslant n^3\}.$$

Lemma 20. *For any $k, t \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that Eq. (5.9) holds then for any $i \in \mathcal{D}$ we have that*

$$\mathbb{P}\left(\left|N_i - \overline{N_i}\right| > \frac{1}{2\,n}R_i\right) < \mathbb{P}\left(N_i \neq 0\right)$$

and for any $i \notin \mathcal{D}$ we have that

$$\mathbb{P}\left(\left|N_i - \overline{N_i}\right| > \frac{1}{2\,n}R_i\right) < \mathbb{P}\left(\left|N_i - \overline{N_i}\right| > \frac{n^2}{2}\max\left(\sqrt{\overline{N_i}},\, 1\right)\right). \tag{5.10}$$

We simply bound the probability in the case $i \in \mathcal{D}$ by $\binom{n}{i}/2^{n-k}$ using a simple union bound and bound the probability in the case $i \notin \mathcal{D}$ by some $2^{-\omega(n)}$ by using the Poisson model, which allows writing

Lemma 21. *For any λ positive implicit function of n and any g positive implicit function of n such that $g \in \omega(1)$ we have that*

$$\mathbb{P}_{\mathbf{X}\sim\text{Poisson}(\lambda)}\left(|\mathbf{X} - \lambda| > g\,\max\left(\sqrt{\lambda}, 1\right)\right) = 2^{-\omega(n)}. \tag{5.11}$$

Interestingly enough we observe that the previous bounds on R_i given by Lemma 19 is tight when $i \approx n/2$ but when i decreases there is an exponential gap. As such our Poisson model is really useful precisely in the regime $i \approx n/2$ and we think that for intermediate i's (but not in $\mathcal{D}$) we could tighter bounds to prove directly our result.

All in all, these lemmas allow to prove our main proposition.

Proof of Proposition 42. Let us consider the implicit functions of $n \in \mathbb{N}$, k, t, w that meet the conditions of the proposition. Applying in sequence Lemmas 17, 18 and 20 we only have left to prove the two following equalities

$$\max_{i\in\mathcal{D}} \mathbb{P}\left(N_i \neq 0\right) = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{2^{n-k}}\right), \tag{5.12}$$

$$\max_{i\notin\mathcal{D}} \mathbb{P}\left(\left|N_i - \overline{N_i}\right| > \frac{n^2}{2}\max\left(\sqrt{\overline{N_i}},\, 1\right)\right) = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{2^{n-k}}\right). \tag{5.13}$$

Let us prove the first equality. It is readily seen that

$$\begin{aligned}\mathbb{P}\left(N_i \neq 0\right) &= \mathbb{P}\left(\bigcup_{\mathbf{x}\in\mathcal{S}_i^n} \text{"}\mathbf{x} \in \mathcal{C} + \mathbf{z}\text{"}\right)\\ &\leqslant \frac{\binom{n}{i}}{2^{n-k}}.\end{aligned}$$

Now we only have left to show that $\max_{i\in\mathcal{D}} i \leqslant t + \mathcal{O}(\log n)$, indeed as $t = \Theta(n)$ and $t < n/2$ by assumptions in the proposition, this would directly yield that

$$\begin{aligned}\max_{i\in\mathcal{D}} \mathbb{P}\left(N_i \neq 0\right) &\leqslant \max_{i\in\mathcal{D}} \frac{\binom{n}{i}}{2^{n-k}} \\ &\leqslant \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{2^{n-k}}\right).\end{aligned}$$

Let $i \in \mathcal{D}$ and let us upper bound it. By definition of $\mathcal{D}$ we have that

$$\left|\frac{K_w^{(n)}(t)}{K_w^{(n)}(i)}\right| \leqslant n^3 \tag{5.14}$$

By assumption in the proposition, $t < \text{Root}\left(K_w^{(n)}\right)$, so we have from Proposition 24 we have that

$$\left|K_w^{(n)}(t)\right| \in \tilde{\Omega}\left(2^{\widetilde{\kappa}(w/n,\, t/n)n}\right) \tag{5.15}$$

Moreover, independently of the value of i we always we have that

$$\left|K_w^{(n)}(i)\right| \leqslant \widetilde{\mathcal{O}}\left(2^{\widetilde{\kappa}(w/n,\, i/n)n}\right). \tag{5.16}$$

Combining all these two inequalities this means that

$$n^3 = \tilde{\Omega}\left(2^{(\widetilde{\kappa}(w/n,\, t/n) - \widetilde{\kappa}(w/n,\, i/n))n}\right)$$

Now, from Proposition 25, $\widetilde{\kappa}\left(w/n,\ i/n\right)$ is differentiable and strictly decreasing in i/n. Thus, clearly $i \leqslant t + \mathcal{O}(\log n)$ which concludes the proof of Eq. (5.12).

Let us now prove Eq. (5.13). This is where we use the Poisson model. We have that $\overline{N_i} = \binom{n}{i}/2^{n-k}$ which is also the parameter of the Poisson model, thus we can directly apply Lemma 21 to prove that

$$\mathbb{P}\left(\left|N_i - \overline{N_i}\right| > \frac{1}{2\,n} R_i\right) < \mathbb{P}\left(\left|N_i - \overline{N_i}\right| > \frac{n^2}{2} \max\left(\sqrt{\overline{N_i}},\ 1\right)\right) = 2^{-\omega(n)}$$

This concludes the proof. □

5.6.1.1 Proof of the intermediate lemmas and of the last proposition

Proof of Lemma 19. Recall that we want to lower bound $R_i = \left|\frac{K_w^{(n)}(t)}{K_w^{(n)}(i)}\right|$. From Eq. (5.9) we have that

$$K_w^{(n)}(t) = \omega\left(n^3\right)\sqrt{2^k\binom{n}{w}}$$

We can upper bound $\left|K_w^{(n)}(i)\right|$ using Proposition 19 which gives that $\sum_{j=0}^{n}\binom{n}{j}K_w^{(n)}(j)^2 = 2^n\binom{n}{w}$. As a sum of positive term it's i'th can be upper bounded by the total, namely

$$\binom{n}{i}K_w^{(n)}(i)^2 \leqslant 2^n\binom{n}{w}$$

which yield

$$\left|K_w^{(n)}(i)\right| \geqslant \frac{2^n \binom{n}{w}}{\binom{n}{i}}.$$

As such

$$\begin{aligned}\left|\frac{K_w^{(n)}(t)}{K_w^{(n)}(i)}\right| &\geqslant \omega\left(n^3\right)\sqrt{2^k\binom{n}{w}}\sqrt{\frac{\binom{n}{i}}{2^n\binom{n}{w}}} \\ &\geqslant \omega\left(n^3\right)\sqrt{\frac{\binom{n}{i}}{2^{n-k}}} \\ &\geqslant \omega\left(n^3\right)\sqrt{\overline{N_i}}.\end{aligned}$$

□

Proof of Lemma 20. Let us prove the first inequality. Let $i \in \mathcal{D}$. We only have to prove that $\frac{1}{2\,n}R_i > \overline{N_i}$ and the result follows.

First we upper bound and lower bound R_i by using the fact that $i \in \mathcal{D}$ and Lemma 19, which gives

$$n^3 > R_i > n^3\sqrt{\overline{N_i}}, \tag{5.17}$$

$$\frac{n^2}{2} > \frac{1}{2\,n}R_i > \frac{n^2}{2}\sqrt{\overline{N_i}}. \tag{5.18}$$

Now Eq. (5.17) proves that $\sqrt{\overline{N_i}} < 1$ and thus that $\sqrt{N_i} > N_i$ and Eq. (5.18) proves that $\frac{1}{2\,n}R_i > \sqrt{\overline{N_i}}$ all in all yielding that $\frac{1}{2\,n}R_i > \overline{N_i}$ which concludes the proof.

Let us prove the second inequality. Let $i \notin \mathcal{D}$. By assumption on i we have that

$$R_i \geqslant n^3$$

and with the previous lemma that

$$R_i \geqslant n^3\sqrt{\overline{N_i}}$$

thus

$$\frac{1}{2n}R_i \geqslant \frac{1}{2}n^2 \max\left(\sqrt{\overline{N_i}},\ 1\right).$$

Finally, this yield that

$$\mathbb{P}\left(\left|N_i - \overline{N_i}\right| > \frac{1}{2\,n}R_i\right) \leqslant \mathbb{P}\left(\left|N_i - \overline{N_i}\right| > \frac{1}{2}n^2 \max\left(\sqrt{\overline{N_i}},\ 1\right)\right)$$

□

Proof of Lemma 21. From [Gol17, Prop 11.15] we have

$$\mathbb{P}\left(|\mathbf{X} - \lambda| > r\right) \leqslant 2\,e^{\frac{-\,r^2}{2\,(\lambda + r)}}.$$

Thus,

$$\mathbb{P}\left(|\mathbf{X}-\lambda| > g(n)\max\left(\sqrt{\lambda},1\right)\right) \leqslant 2\, e^{\frac{-\,g(n)^2\max(\lambda,1)}{2\,(\lambda+g(n)\max(\sqrt{\lambda},1))}} \leqslant 2\, e^{\frac{-\,g(n)}{2\left(\frac{\lambda+g(n)\max(\sqrt{\lambda},1)}{g(n)\max(\lambda,1)}\right)}}.$$

We only have left to show that

$$\frac{\lambda+g(n)\max\left(\sqrt{\lambda},1\right)}{g(n)\max\left(\lambda,1\right)} = \mathcal{O}(1)\,.$$

First it is readily seen that we have that $(g(n)=\omega(n))$

$$\frac{\lambda}{g(n)\max\left(\lambda,1\right)} = \mathcal{O}(1)\,,$$

and second,

$$\frac{g(n)\max\left(\sqrt{\lambda},1\right)}{g(n)\max\left(\lambda,1\right)} = \begin{cases} 1=\mathcal{O}(1) & \text{if } \lambda \leqslant 1 \\ \frac{1}{\sqrt{\lambda}}=\mathcal{O}(1) & \text{if } \lambda > 1. \end{cases}$$

□

5.6.2 Problem with the LPN modeling to analyze the algorithm and first version of RLPN.

RLPN was first introduced in [CDMT22] as a slight variant, which we call RLPN0 here and which we analyzed using the following LPN modeling.

Model 4 (LPN modeling.)**.** *Let $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\,k)$ and let $\mathbf{e} \in \mathcal{S}_t^n$. Let $\mathscr{P}$ and $\mathscr{N}$ be two complementary subsets of $[\![1,\,n]\!]$ and let $\mathscr{H} \subset \{\mathbf{h} \in \mathcal{C}^\perp \,:\, |\mathbf{h}_{\mathscr{N}}| = w\}$ and denote by ε the bias of the noise of the* LPN *samples given by $\mathscr{H}$, namely*

$$\varepsilon \overset{def}{=} \mathrm{bias}_{\mathbf{h}\sim\mathcal{U}(\mathscr{H})}\left(\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}}\rangle\right).$$

We make the model that the samples given by $\mathcal{L} = (\mathbf{h}_{\mathscr{P}},\ \langle \mathbf{e}, \mathbf{h}\rangle)_{\mathbf{h}\in\mathscr{H}}$ given by $\mathscr{H}$ are distributed as true LPN *samples of secret $\mathbf{e}_{\mathscr{P}}$, namely that*

$$(\mathbf{h}_{\mathscr{P}},\ \langle \mathbf{e}, \mathbf{h}\rangle)_{\mathbf{h}\in\mathscr{H}} \sim \left(\mathbf{a}^{(i)},\ \left\langle \mathbf{e}_{\mathscr{P}}, \mathbf{a}^{(i)}\right\rangle + e^{(i)}\right)_{i=0\cdots|\mathscr{H}|}$$

where the samples $\mathbf{a}^{(i)} \sim \mathcal{U}\left(\mathbb{F}_2^{|\mathscr{P}|}\right)$ and the noise is distributed as $e^{(i)} \sim \mathrm{Ber}\left(\frac{1-\varepsilon}{2}\right)$, moreover all the variables are ***independent****.*

The RLPN0 algorithm differed from the one we described in this thesis by its way of choosing out some candidates $\mathbf{x}$ from the evaluation of the score function $F^{\mathcal{L}}$. Basically it took, at each iteration the unique $\mathbf{x}^* \in \mathbb{F}_2^s$ that maximized the LPN score function, namely

$$\mathbf{x}^* = \max_{\mathbf{x}\in\mathbb{F}_2^s} F^{\mathcal{L}}(\mathbf{x})$$

and simply concluded that $\mathbf{x}^*$ was $\mathbf{e}_{\mathscr{P}}$ if $F^{\mathcal{L}}(\mathbf{x}^*)$ was superior to a certain well-chosen threshold T and stopped the whole algorithm returning this $\mathbf{x}^*$ a soon as one candidate meeting this condition was found.

The LPN modeling allowed proving that as long as the number of samples N is some $N = \widetilde{\mathcal{O}}(1/\varepsilon^2)$ the preceding procedure recovers the secret $\mathbf{e}_{\mathscr{P}}$ with overwhelming probability and that basically there will not be any false candidates and there is no need for finer filtering the $\mathbf{x} \in \mathbb{F}_2^n$. In turn, we showed that the bias ε of the LPN samples could be precisely estimated as $\varepsilon \approx \delta_w^{(n-s)}(u)$ overall this yielded the simple condition that as long as $N = \omega(1)/\delta_w^{(n-s)}(u)^2$ RLPN0 would succeed.

5.6.2.1 Experimental discrepancy with the LPN modeling

We show here experimentally that this model does not hold and that in all generality this can be problematic in some regimes in RLPN0. First, let us compute the expected behavior of the false candidates under the LPN modeling

Proposition 43. *Under the* LPN *modeling we have that*

$$\mathbb{E}_{LPN}\left(\left|\{\mathbf{x} \in \mathbb{F}_2^s \,:\, \mathbf{x} \neq \mathbf{e}_{\mathscr{P}} \text{ and } F^{\mathcal{L}}(\mathbf{x}) > T\}\right|\right) = (2^s - 1)\mathbb{P}\left(X \geqslant \frac{T + |\mathscr{H}|}{2}\right).$$

where

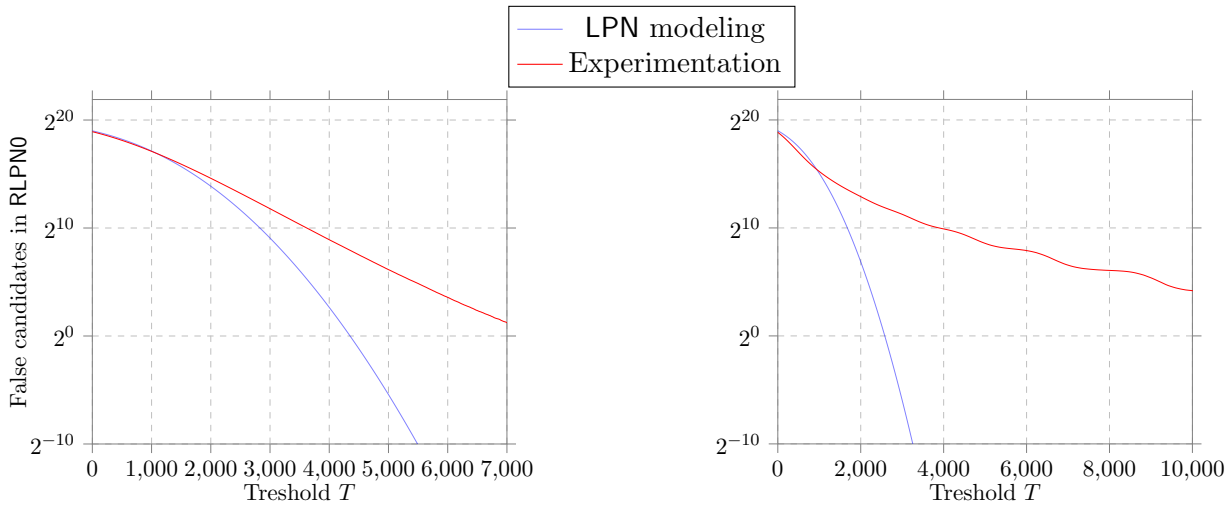
$$X \sim \text{Binomial}\left(|\mathscr{H}|, \frac{1}{2}\right).$$

Proof. Let $\mathbf{x} \in \mathbb{F}_2^s$ be such that $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$. We have that

$$\begin{aligned} F^{\mathcal{L}}(\mathbf{x}) &= \sum_{\mathbf{h} \mathscr{H}} (-1)^{\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle} \\ &= \sum_{\mathbf{h} \mathscr{H}} (-1)^{\langle \mathbf{e}_{\mathscr{P}} - \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle} \\ &= 2|\mathscr{H}_0| - |\mathscr{H}|. \end{aligned}$$

where $\mathscr{H}_0 = \{\mathbf{h} \in \mathscr{H} \,:\, \langle \mathbf{e}_{\mathscr{P}} - \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle = 0\}$. From the LPN modeling, we get that $|\mathscr{H}_0| \sim \text{Binomial}(|\mathscr{H}|, N)$. Our results follow using the linearity of the expected value. □

We plot it against its experimental counterpart in Fig. 5.9. We observe concerning differences in the very low rate regime and significant differences in the high regime rate.



(a) $n = 400,\ k = 30,\ s = 20,\ w = 4$

(b) $n = 100,\ k = 30,\ s = 20,\ w = 6$

Figure 5.9: Experimental average number of false candidates $\left|\{\mathbf{x} \in \mathbb{F}_2^s : \mathbf{x} \neq \mathbf{e}_{\mathscr{P}} \text{ and } F^{\mathcal{L}}(\mathbf{x}) > T\}\right|$ against its LPN modeled counterpart. The experimental curve was obtained by running multiple times one iteration of RLPN0 in which we compute the whole set of dual vectors of weight w on $\mathscr{N}$, namely $\mathscr{H} = \{\mathbf{h} \in \mathcal{C}^{\perp} : |\mathbf{h}_{\mathscr{N}}| = w\}$, each time with a new instance of $\mathrm{DP}_{\mathbf{G}}(n, k, d_{\mathrm{GV}}(n,\ k))$. We plotted the modeled distribution by averaging the expectation of Proposition 43 with the experimental values for $|\mathscr{H}|$.

This shows the need to change slightly RLPN0 by i) considering a set of candidates and not only one and ii) add an exponential filtering step that only keeps candidates that are such that of the right weight $t - u$.

Chapter 6

Dual Attack 3.0 : Reducing sparse-LPN to LPN

Summary

In this chapter we devise a generalization of RLPN that we call double-RLPN. Our main observation is that the LPN problem to which we reduced in RLPN is in fact a $\mathsf{sparse}-\mathsf{LPN}$ problem: its secret is of low Hamming weight as it is some part of the error vector of a decoding problem. However, the FFT-based solver that we use to solve this LPN problem does not leverage this. In double-RLPN we leverage the sparseness of the secret by using the technique of [GJL14] to further reduce the dimension of our LPN problem by compressing the secret with the help of a linear code that we know how to decode. We propose and analyze a concrete instantiation of this decoding technique with a product of a constant number of random linear codes that allows us to compress optimally while incurring only a polynomial overhead in our regimes of interest. We devise a second-order concentration bound to estimate the new noise of our LPN samples. We show that the introduction of this secret compressing technique introduces many false candidates, namely candidates for the secret of our LPN problem that are not related to our original LPN problem secret. However, we are able to precisely bound their number with a conjecture. We show that, in the end, the cost of dealing with those never dominates the complexity of our decoder. This conjecture is a generalization of the conjecture made in RLPN and is about the exponential tail behavior of the score function. We thoroughly verify it experimentally. All in all, this allows us to beat all known decoders when decoding random binary linear codes at the Gilbert-Varshamov distance for constant rates R smaller than 0.42, see Fig. 6.1. This chapter contains [CDMT24] rewritten and with some small additional results.

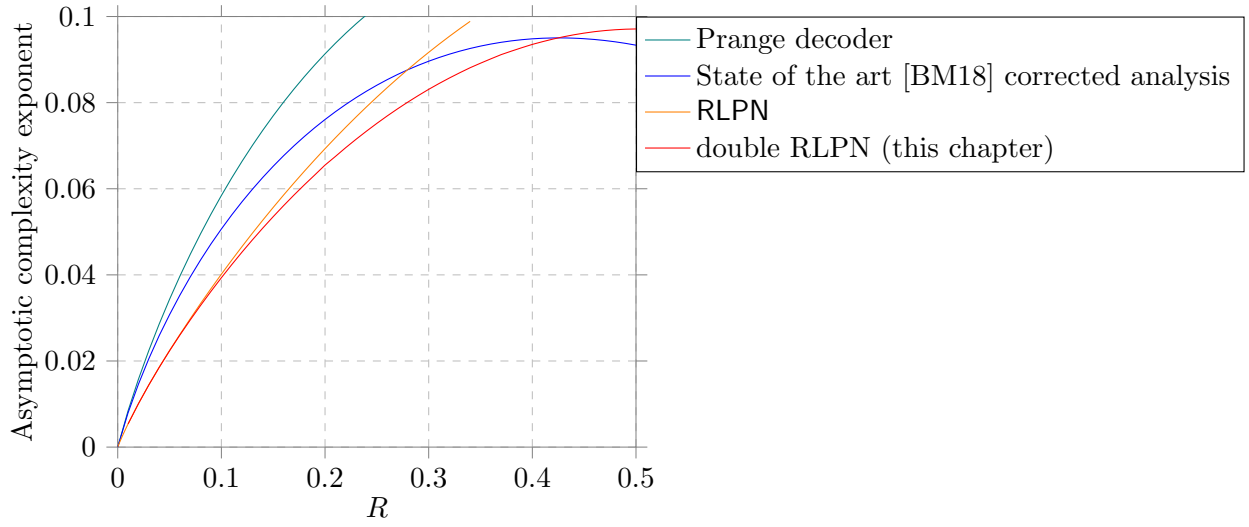


Figure 6.1: Asymptotic complexity exponent, relative to the codelength, of some generic decoders when decoding random codes of rate R at the Gilbert-Varshamov distance. Our corrected analysis of [BM18] is given in Section 2.1.3.4

Contents

6.1 Introduction

6.1.1 The LPN secret in RLPN is sparse

Recall that in the previous chapter we reduced decoding $\mathbf{y} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e} \in \mathcal{S}_t^n$ to an LPN problem. This is done by chosing $\mathscr{P}$ and $\mathscr{N}$ two complementary subsets of $[\![1, n]\!]$ of size say s and $n - s$ respectively and computed many dual vectors $\mathbf{h} \in \mathcal{C}^\perp$ which are of low hamming weight on $\mathscr{N}$, each yielding the following sample of secret $\mathbf{s} = \mathbf{e}_{\mathscr{P}}$:

$$(\mathbf{a},\ \langle \mathbf{a}, \mathbf{s} \rangle + e) \quad \text{where} \quad \begin{cases} \mathbf{a} & = \mathbf{h}_{\mathscr{P}} \\ \mathbf{s} & = \mathbf{e}_{\mathscr{P}} \\ e & = \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle . \end{cases} \tag{6.1}$$

The obtained LPN problem is then solved with an FFT. Here we notice that the LPN problem we have to solve is actually a sparse LPN problem: the secret $\mathbf{e}_{\mathscr{P}}$ is not uniformly distributed among $\mathbb{F}_2^s$ since it is of low weight as it is the restriction of an error vector which is itself of low weight. Unfortunately, the FFT algorithm used for recovering the secret $\mathbf{e}_{\mathscr{P}}$ is unable to exploit this fact. In a sense, what we need here to improve RLPN decoding is an algorithm for solving sparse secret LPN in the very noisy regime, where we recall that very essentially our noise is exponentially small in the error weight $|\mathbf{e}|$.

6.1.2 Reduction from Sparse LPN to Plain LPN

Here we reuse the technique of [GJL14] that compress the secret of the LPN problem with a linear code. This reduction allows to reduce the LPN problem dimension but increases the noise. We recall here the reduction.

6.1.2.1 The reduction

Say we have access to a linear code $\mathcal{C}_{\text{aux}}$ of length s and dimension k_{aux} which we know how to decode efficiently for essentially every word of the space. Say we have an LPN sample $(\mathbf{a}, b) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$ for which the secret $\mathbf{s} \in \mathbb{F}_2^s$ is sparse. The idea is to decode $\mathbf{a} \in \mathbb{F}_2^s$ onto $\mathcal{C}_{\text{aux}}$, yielding say $\mathbf{a} = \mathbf{c}_{\text{aux}} + \mathbf{e}_{\text{aux}}$ where $\mathbf{c}_{\text{aux}} \in \mathcal{C}_{\text{aux}}$ and the error vector $\mathbf{e}_{\text{aux}}$ is of low weight. We notice that this directly yield the sample $(\mathbf{c}_{\text{aux}}, \langle \mathbf{c}_{\text{aux}}, \mathbf{s} \rangle + \langle \mathbf{e}_{\text{aux}}, \mathbf{s} \rangle + e)$ where now $\mathbf{a}$ is replaced by $\mathbf{c}_{\text{aux}}$ which lies in a lower dimension subspace but where the new noise, $\langle \mathbf{e}_{\text{aux}}, \mathbf{s} \rangle + e$, has increased. We can really take advantage of the fact that $\mathbf{c}_{\text{aux}}$ lies in a linear subspace by considering $\mathbf{G}_{\text{aux}}$ a generator matrix of $\mathcal{C}_{\text{aux}}$ and writing $\mathbf{c}_{\text{aux}}$ as $\mathbf{c}_{\text{aux}} = \mathbf{m}_{\text{aux}} \mathbf{G}_{\text{aux}}$, this allows rewriting $\langle \mathbf{c}_{\text{aux}}, \mathbf{s} \rangle = \langle \mathbf{m}_{\text{aux}} \mathbf{G}_{\text{aux}}, \mathbf{s} \rangle = \langle \mathbf{m}_{\text{aux}}, \mathbf{s} \mathbf{G}_{\text{aux}}^\intercal \rangle$ which finally yields the sample

$$\left(\underbrace{\mathbf{m}_{\text{aux}}}_{\in \mathbb{F}_2^{k_{\text{aux}}}},\ \left\langle \mathbf{m}_{\text{aux}}, \underbrace{\mathbf{s} \mathbf{G}_{\text{aux}}^\intercal}_{\text{secret}} \right\rangle + \underbrace{\langle \mathbf{e}_{\text{aux}}, \mathbf{s} \rangle + e}_{\text{noise}} \right)$$

making our new secret $\mathbf{s} \mathbf{G}_{\text{aux}}^\intercal \in \mathbb{F}_2^{k_{\text{aux}}}$. Knowing this new secret reveals some linear combination of $\mathbf{e}_{\mathscr{P}}$.

6.1.2.2 Shape of the reduced LPN samples of RLPN

All in all the new samples are now $(\mathbf{m}_{\text{aux}}, \langle \mathbf{h}, \mathbf{y} \rangle)$ instead of $(\mathbf{h}_{\mathscr{P}}, \langle \mathbf{h}, \mathbf{y} \rangle)$. We have:

$$(\mathbf{a}',\ \langle \mathbf{a}', \mathbf{s}' \rangle + e') \quad \text{where} \quad \begin{cases} \mathbf{a}' & = \mathbf{m}_{\text{aux}} \in \mathbb{F}_2^{k_{\text{aux}}} \\ \mathbf{s}' & = \mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\mathsf{T}} \\ e' & = \langle \mathbf{e}_{\mathscr{P}}, \mathbf{e}_{\text{aux}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle \end{cases} \begin{array}{l} \\ \text{with } \mathbf{h}_{\mathscr{P}} = \mathbf{m}_{\text{aux}} \mathbf{G}_{\text{aux}} + \mathbf{e}_{\text{aux}} \\ \text{and } |\mathbf{e}_{\text{aux}}| \text{ low .} \end{array} \tag{6.2}$$

Before, basically the optimal parameters for RLPN-decoding were such that the cost of FFT decoding the LPN secret, namely $\widetilde{\mathcal{O}}(2^s)$ is of the same order as the cost for computing the samples, say to simplify that this cost is $1/\varepsilon^2$ where ε is the bias of the LPN sample in RLPN and is exponentially decreasing in w, the weight of the dual vectors on $\mathscr{N}$. Here since we do not pay anymore $\widetilde{\mathcal{O}}(2^s)$ for the FFT, decoding the new LPN secret but $\widetilde{\mathcal{O}}\big(2^{k_{\text{aux}}}\big)$ we can take larger values for s which themselves gives a smaller support $\mathscr{N}$ resulting in much smaller weight w on N and thus the bias term coming from $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$ is much smaller. Of course there is an additional noise term now which is $\langle \mathbf{e}_{\mathscr{P}}, \mathbf{e}_{\text{aux}} \rangle$, however, all in all the gain we have by being able to use a much larger s outweighs the additional noise term.

Thus now we have a gain in dimension as we went from sample in dimension s to samples in dimension k_{aux} but with an added noise $\langle \mathbf{s}, \mathbf{e}_{\text{aux}} \rangle + e$ compared to the original noise e. It turns out that this new tredeoff between noise and dimension is extremely beneficial. Forgetting about the fact that $\mathbf{e}_{\text{aux}}$ comes from a decoding problem, we can intuitively think that this added noise $\langle \mathbf{s}, \mathbf{e}_{\text{aux}} \rangle$ acts like a random variable with bias $\delta^{(s)}_{|\mathbf{e}_{\text{aux}}|}(|\mathbf{s}|)$, namely exponentially decreasing in $|\mathbf{e}_{\text{aux}}|$ and $|\mathbf{s}|$. Supposing to simplify that we have access to a code that we can decode at $t_{\text{aux}} = d_{\text{GV}}(s,\ k_{\text{aux}})$, each sample has an added noise of the order $\delta^{(s)}_{t_{\text{aux}}}(|\mathbf{s}|)$ which increases exponentially when k_{aux} decreases.

6.1.3 The double-RLPN algorithm

We call this new algorithm double-RLPN decoding, since it is based on two successive reductions: first we reduce the problem to sparse-LPN, then we reduce the sparse-LPN to a plain-LPN problem as explained above. The algorithm consists in the same steps as the RLPN algorithm but adds this extra reduction step from sparse to plain LPN. Basically we start by splitting the support in two complementary part $\mathscr{P}$ and $\mathscr{N}$ and make a bet that $|\mathbf{e}_{\mathscr{N}}| = u$. We compute dual vectors $\mathbf{h} \in \mathcal{C}^{\perp}$ of low weight on the part $\mathscr{N}$ yielding the LPN samples $(\mathbf{h}_{\mathscr{P}}, \langle \mathbf{y}, \mathbf{h} \rangle)$ as in RLPN. We then make our reduction by choosing an auxiliary code $\mathcal{C}_{\text{aux}}$ of generator matrix $\mathbf{G}_{\text{aux}}$ which we know how to decode efficiently and proceed to compute the reduced LPN samples $(\mathbf{m}_{\text{aux}}, \langle \mathbf{y}, \mathbf{h} \rangle)$ of secret $\mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\mathsf{T}}$ obtained by decoding $\mathbf{h}_{\mathscr{P}}$ onto $\mathcal{C}_{\text{aux}}$ as $\mathbf{h}_{\mathscr{P}} = \mathbf{m}_{\text{aux}} \mathbf{G}_{\text{aux}} + \mathbf{e}_{\text{aux}}$ (where $\mathbf{e}_{\text{aux}}$ is an error vector of low Hamming weight). This reduced LPN problem is then solved with the same FFT based solver as in RLPN returning a score function $F^{\mathcal{L}}(\mathbf{x})$ which is expected to be big if $\mathbf{x} = \mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\mathsf{T}}$ is the secret of the reduced LPN problem. We then filter out these candidates $\mathbf{x}$ for the secret $\mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\mathsf{T}}$ based on this score function value. We then proceed from this set of candidates to recover the whole error vector $\mathbf{e}$ by solving for each candidate $\mathbf{x}$ two successive smaller decoding problems: on to recover $\mathbf{e}_{\mathscr{P}}$ and one to recover $\mathbf{e}_{\mathscr{N}}$, depending on if one decoding problem failed this tells us if $\mathbf{x}$ was a false candidate or not.

6.1.4 Which code to use

In [GJL14] (see its extended version [GJL20]) this compression technique was analyzed when a perfect code or a juxtaposition of perfect codes was used. Here juxtaposition means a Cartesian product. The idea is that perfect codes have good covering properties. This allows, by decoding the sample onto the closest codeword, to maximize the bias. This decoding was done by precomputing a syndrome table by enumerating the errors. In that regard, the juxtaposition strategy allows precomputing independently smaller codes, thus mitigating the cost of the precomputation while only slightly damaging the covering property.

We instantiate our algorithm with a juxtaposition of a *constant* number of *random* linear codes. Note that using random linear codes is essentially as good as using perfect codes (which only exist for a very limited set of parameters). Instead of referring to the covering radius of a code, we prefer to refer here to the slightly more natural quantity which is the distortion level of a code/decoder for a certain channel: we want it as close as possible as the optimal rate-distortion bound. The reason is that we are not really interested in the worst-case scenario but rather in the average case. In that sense really using a random linear code seems like the optimal strategy. For the regime we are interested in this thesis, namely when solving a decoding problem at constant rate and when the error weight grows linearly in the length of the code, the parameters of double-RLPN are such that we end up needing to decode an exponential number of samples (the ambient space is $\mathbb{F}_2^s$ where $s = |\mathscr{P}|$), say $2^{\lambda s}$. Consequently, we only need an amortized time decoder. Essentially we take the juxtaposition of a constant number $b = 1/\lambda$ of random linear codes. The fact that each block is of length λs allows us to conclude that this enumerating step is always smaller than the size of the list we consider, while the constant nature of the number of blocks allows us to prove that these codes are as good as random linear codes. In our regime, this strategy incurs only a polynomial loss compared to the use of a random linear code that we would know how to decode for free.

Juxtapositing a constant number of random codes could also be used for building good Locality-Sensitive-Hashing (LSH) function. In particular these are the building blocks behind recent ISD's to solve a Near-Collision problem. Currently the provable version of these algorithms, see [Car20] (or alternatively [EKZ21]) use the juxtaposition of $o(n)$ random linear codes that are decoded by enumerating the codewords. This incurs a superpolynomial loss.

Alternatively, note that in [Car20], it was proposed to use polar codes (which achieve the same rate-distortion bound as random linear codes) to build an LSH. Moreover, [Car20, Figure 9.3] provided experimental evidence that using polar codes in this context would only produce a polynomial overhead. It is very likely, based on those experimental results, that the polynomial overhead is much smaller than with our construction. However, using polar codes in our context requires finer analysis (experimental or theoretical) to assess its impact on the shape of our LPN samples and on the bias of the noise term appearing in the LPN reduction.

6.1.5 Analysis

We analyze our algorithm in the setting when all the dual vectors of weight w on $\mathscr{N}$ are computed.Our analysis is essentially a generalization of the analysis we made in RLPN and is made in two steps: first we devise a minimal condition such that our reduction even makes sense and then count precisely (using an experimentally verified conjecture) the number of

false candidates we get when our parameters verify these minimal conditions.

6.1.5.1 A minimal condition for our algorithm to make sense

First, the minimal condition comes from the fact that we can show that the bias of the reduced LPN samples is of the order

$$\delta \approx \delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)$$

which is by the way the bias we would obtain by summing two independent variables of bias $\delta_w^{(n-s)}(u)$ and $\delta_{t_{\text{aux}}}^{(s)}(t-u)$ which are respectively themselves the bias of we would obtain for each part of the noise by forgetting about the structure. Our intuition is that we must have at least

$$N \approx 1/\delta^2$$

LPN samples in order for our reduction to make sense. Importantly this stays valid, up to polynomial factors, as long as the number of blocks in the juxtaposition code is constant.

6.1.5.2 Counting the number of candidates from this condition

Second, we count, under this simple condition, how much false candidates we have at each iteration. Similarly, to RLPN our second order concentration bounds on the score function will be insufficient to bound their number, thus as for RLPN we will need a conjecture to conclude. This conjecture is some generalization of what we did for RLPN, it is essentially some exponential strengthening of our second order concentration bounds, this is done by devising a simple lower bound on the concentration of the score function and conjecturing that this lower bound is tight. It stems from a key duality formula for the score function involving the weight enumerator of some coset of $\mathcal{C}$ and $\mathcal{C}_{\text{aux}}^{\perp}$. Our conjecture is basically that a false candidate only occurs if these coset codes contains some unusually low weight codeword. More than simply giving an intuition, this duality formula will be key to showing experimentally that our conjecture holds. Indeed by making a simple model, as in RLPN, that the weight enumerators acts like a Poisson variable of right expected value we are able to prove our conjecture while verifying experimentally that making this model does not change the distribution of the score function. Interestingly, contrary to RLPN we show that because of the additional structure coming from the second reduction we can have an exponential number of false candidates, but we show that in all cases of interest their number is sufficiently small such that the last checking step never dominates.

6.2 The double RLPN decoder

6.2.1 A generic framework for double-RLPN

We give here the main steps of the algorithm. Basically they are exactly the same as RLPN at the difference that we add the extra reduction step from sparse to plain LPN and finish with a slightly more involved procedure to recover the whole error vector $\mathbf{e}$ as now the secret of our reduced LPN samples is some linear combination of $\mathbf{e}_{\mathscr{P}}$. Basically now our algorithm works in 4 step:

1. Compute-LPN-Sample. It computes many dual vectors $\mathbf{h} \in \mathcal{C}^{\perp}$ of low weight w on $\mathscr{N}$ and stores and returns the list, say $\widetilde{\mathcal{L}}$, composed of the LPN samples $(\mathbf{a}, b) = (\mathbf{h}_{\mathscr{P}}, \langle \mathbf{y}, \mathbf{h} \rangle)$ with secret $\mathbf{e}_{\mathscr{P}}$, as described in Eq. (5.1). This procedure is reused from RLPN.

2. Sparse-LPN-to-Plain-LPN($\mathcal{C}$, $\mathscr{N}$; w). Takes as input the previous samples and apply the reduction in the introduction, to produce samples of the form Eq. (6.2) with secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal} \in \mathbb{F}_2^{k_{\text{aux}}}$ and outputs these new samples in a list $\mathcal{L}$ along with the auxiliary code $\mathcal{C}_{\text{aux}}$ with generator matrix $\mathbf{G}_{\text{aux}}$ that was used to make the reduction.

3. LPN-Solver. It takes as input the list of LPN samples and outputs a (small) set of candidates $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ for the secret of the reduced LPN problem $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$.

4. double-RLPN-Recover-Full-Error. It takes as input the set of candidates, if the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ is in the set of candidate we expect that this procedures returns the full error $\mathbf{e}$ or it fails.

As in RLPN we make a bet that $|\mathbf{e}_{\mathscr{N}}| = u$ and iterate this procedure a certain number of times N_{iter} where N_{iter} is chosen big enough so that the bet is valid as least once.

Algorithm 16 double-RLPN algorithm

Name: DOUBLE-RLPN
Input: $\mathcal{C} \in \mathfrak{C}[n, k]$, $\mathbf{y} \in \mathbb{F}_2^n$, t
Parameter: s, w, u and T, N_{iter}

while $i = 1 \dots N_{\text{iter}}$ **do**
  $\mathscr{P} \xleftarrow{\$} \{\mathscr{P} \subset [\![1, n]\!] \, : \, |\mathscr{P}| = s\}$ ▷ *Hope that* $\mathbf{e}_{\mathscr{N}} = u$
  $\mathscr{N} \leftarrow [\![1, n]\!] \setminus \mathscr{P}$
  $\widetilde{\mathcal{L}} \leftarrow$ Create-LPN-Samples($\mathcal{C}$, $\mathscr{N}$; w) ▷ *Returns some LPN samples of the form* $(\mathbf{h}_{\mathscr{P}}, \langle \mathbf{y}, \mathbf{h} \rangle)$
  $\mathcal{L}, \mathcal{C}_{\text{aux}} \leftarrow$ Sparse-LPN-to-Plain-LPN($\widetilde{\mathcal{L}}; s; k_{\text{aux}}; t_{\text{aux}}$) ▷ *Returns the reduced* LPN *samples of secret* $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$
  $\mathcal{S} \leftarrow$ LPN-Solver($\mathcal{L}$; T) ▷ *Returns a set of candidates for the secret of the LPN problem given by* $\mathcal{L}$, T *is a treshold*
  $\mathbf{e} \leftarrow$ double-RLPN-Recover-Full-Error($\mathcal{S}$, $\mathscr{P}$, $\mathscr{N}\mathbf{y}$ u) ▷ *Returns either* $\perp$ *if* $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal} \notin \mathcal{S}$ *but returns* $\mathbf{e}$ *if* $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal} \in \mathcal{S}$
  if $\mathbf{e} \neq \perp$ **then**
    return $\mathbf{e}$

6.2.1.1 Computing the LPN samples

This is done exactly as in RLPN as such we reuse precisely Algorithm 13, which we rewrite here.

Algorithm 17 Computing the LPN samples

Name: CREATE-LPN-SAMPLES
Input: $\mathcal{C} \in \mathfrak{C}[n,k]$, $\mathscr{N}$
Require: A procedure COMPUTE-DUAL-VECTORS$(\mathcal{D}; w)$ which given a linear code $\mathcal{D}$ outputs a subset of the dual vectors of $\mathcal{D}$ of weight w.
Parameter: w

1: **If** Information − Set $\left(\mathcal{C}^{\perp}, \mathscr{N}\right)$ = True **continue else** $i \leftarrow i+1$ and **go to** Line 2 of Algorithm 16 ▷ *Check that $\mathscr{N}$ is an information set of $\mathcal{C}^{\perp}$. Continue with overwhelming probability.*
2: $\mathscr{W}_{\mathscr{N}} \leftarrow$ COMPUTE-DUAL-VECTORS$(\mathcal{C}^{\mathscr{N}}\ ;\ w)$ ▷ *Returns a subset of dual vectors of $\mathcal{C}^{\mathscr{N}}$ which are of weight w*
3: $\mathscr{W} \leftarrow \{\text{LIFT}(\mathcal{C}^{\perp}, \mathscr{N},\ \mathbf{h}_{\mathscr{N}})$ for $\mathbf{h}_{\mathscr{N}} \in \mathscr{W}_{\mathscr{N}}\}$ ▷ *Lift those dual vectors $\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}$ to make them dual vectors $\mathbf{h}$ of $\mathcal{C}$ of low weight w on $\mathscr{N}$*
4: $\widetilde{\mathcal{L}} \leftarrow [(\mathbf{h}_{\mathscr{P}},\ \langle \mathbf{y}, \mathbf{h}\rangle)$ for $\mathbf{h} \in \mathscr{W}]$
5: **return** $\widetilde{\mathcal{L}}$

6.2.1.2 Sparse LPN to Plain LPN

Say we are given a list of LPN samples $\widetilde{\mathcal{L}} = [(\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e)]$ with a sparse secret $\mathbf{s} \in \mathbb{F}_2^s$. We recall here the reduction from $\mathsf{sparse}-\mathsf{LPN}$ in dimension s to plain LPN in dimension k_{aux}, smaller than s that was presented in [GJL14]. This requires a family of auxiliary code which we know how to decode efficiently. Namely we require $\mathcal{F}$ a family of $[s, k_{\text{aux}}]$ linear code that we can sample with a randomized procedure SAMPLE-AUXILARY-CODE$(\mathcal{F})$ which returns some $[s, k_{\text{aux}}]$-linear auxiliary code $\mathcal{C}_{\text{aux}}$ under the form of a generator matrix $\mathbf{G}_{\text{aux}}$. It comes with an efficient decoder DECODE-AUXILARY$(\mathcal{C}_{\text{aux}},\ \mathbf{a}\ ;\ t_{\text{aux}})$ which for each $\mathbf{a} \in \mathbb{F}_2^s$, decodes it onto $\mathcal{C}_{\text{aux}}$ at distance t_{aux}. More precisely it returns a subset $\mathcal{E}$, possibly empty, of errors which is such that

$$\mathcal{E} \subset \{\mathbf{e}_{\text{aux}} \in \mathcal{S}^s_{t_{\text{aux}}}\ :\ \mathbf{a} - \mathbf{e}_{\text{aux}} \in \mathcal{C}_{\text{aux}}\}.$$

Now the reduction works as follows. We sample a code $\mathcal{C}_{\text{aux}}$ at random, then, for each sample $(\mathbf{a}, b)$ we decode $\mathbf{a}$ onto $\mathcal{C}_{\text{aux}}$ to obtain $\mathcal{E} = \text{DECODE}(\mathcal{C}_{\text{aux}},\ \mathbf{a}\ ;\ t_{\text{aux}})$. This allows us, for each $\mathbf{e}_{\text{aux}} \in \mathcal{E}$ to construct the reduced samples $(\mathbf{m}_{\text{aux}}, b)$ where $\mathbf{m}_{\text{aux}}$ is uniquely obtained such that $\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} = \mathbf{a} - \mathbf{e}_{\text{aux}}$. All in all, this gives the reduced sample $(\mathbf{m}_{\text{aux}}, \langle \mathbf{m}_{\text{aux}}, \mathbf{s}\mathbf{G}_{\text{aux}}^{\intercal}\rangle + \langle \mathbf{e}_{\text{aux}}, \mathbf{s}\rangle + e)$ with secret $\mathbf{s}\mathbf{G}_{\text{aux}}^{\intercal}$.

Algorithm 18 Reducing sparse LPN to plain LPN

Name: SPARSE-LPN-TO-PLAIN-LPN
Input: $\widetilde{\mathcal{L}}$ ▷ *$\widetilde{\mathcal{L}}$ is a list of* LPN *samples of the form* $(\mathbf{a}, b)$ *where* $b = \langle \mathbf{a}, \mathbf{s} \rangle + e$ *and* $\mathbf{s} \in \mathbb{F}_2^{k_{\text{aux}}}$
Parameter: s, k_{aux}, t_{aux} ▷ *T is the threshold from which we keep the candidates.*

$\mathcal{C}_{\text{aux}} \leftarrow$ SAMPLE-AUXILARY-CODE$(\mathcal{F})$
$\mathbf{G}_{\text{aux}} \leftarrow \mathbf{G}(\mathcal{C}_{\text{aux}})$
$\mathcal{L} \leftarrow \emptyset$
for $(\mathbf{a}, b) \in \widetilde{\mathcal{L}}$ **do**
  $\mathcal{E} \leftarrow$ DECODE-AUXILARY$(\mathcal{C}_{\text{aux}}, \mathbf{a})$ ▷ *Returns a set of error of small weight* $\mathbf{e}_{\text{aux}} \in \mathcal{S}^s_{t_{\text{aux}}}$ *s.t* $\mathbf{a} - \mathbf{e}_{\text{aux}} \in \mathcal{C}_{\text{aux}}$
  for $\mathbf{e}_{\text{aux}} \in \mathcal{E}$ **do**
    $\mathbf{m}_{\text{aux}}$ is the unique vector such that $\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} = \mathbf{a} - \mathbf{e}_{\text{aux}}$
    $\mathcal{L}$.APPEND$((\mathbf{m}_{\text{aux}}, b))$ ▷ *The new reduced* LPN
return $\mathcal{L}$

6.2.1.2.1 The procedure

6.2.1.2.2 Shape of the reduced samples

We have that

Fact 21. *When used in Algorithm 16 the output list $\mathcal{L}$ of reduced* LPN *samples is of the form*

$$\mathcal{L} = [(\mathbf{m}_{\text{aux}}, \langle \mathbf{y}, \mathbf{h} \rangle) \text{ where } \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} = \mathbf{h}_{\mathscr{P}} - \mathbf{e}_{\text{aux}} \text{ for } (\mathbf{h}, \mathbf{e}_{\text{aux}}) \in \mathscr{H}]$$

where

$$\mathscr{H} \stackrel{def}{=} \{(\mathbf{h}, \mathbf{e}_{\text{aux}}) \in \mathscr{W} \times \mathbb{F}_2^s \, : \, \mathbf{e}_{\text{aux}} \in \text{DECODE-AUXILARY}(\mathcal{C}_{\text{aux}}, \mathbf{h}_{\mathscr{P}})\}.$$

For $(\mathbf{m}_{\text{aux}}, \langle \mathbf{y}, \mathbf{h} \rangle) \in \mathcal{L}$ we have that

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\intercal}, \mathbf{m}_{\text{aux}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{e}_{\mathscr{P}}, \mathbf{e}_{\text{aux}} \rangle$$

where $\mathbf{e}_{\text{aux}}$ is the error vector asssociated to $\mathbf{m}_{\text{aux}}$ and $\mathbf{h}_{\mathscr{P}}$.

6.2.1.3 LPN solver

We get our reduced LPN samples in a list $\mathcal{L}$ of LPN samples of the form $(\mathbf{a}, b)$ and produce with an FFT the score function whose definition is recalled here.

Definition 37 (LPN score function)**.** *Given a list $\mathcal{L}$ of samples of the form $(\mathbf{a}, b)$ we define the* LPN *score function as*

$$F^{\mathcal{L}}(\mathbf{x}) \stackrel{def}{=} \sum_{(\mathbf{a},b) \in \mathcal{L}} (-1)^{b - \langle \mathbf{x}, \mathbf{a} \rangle}$$

This LPN score function is in fact computed exactly as in RLPN with a standard FFT. The only difference from RLPN lies after, in the filtering step where, because we do not have an apriori knowledge on the weight of the secret anymore as we did for RLPN, we basically keep any candidate $\mathbf{z}$ of $\mathbb{F}_2^{k_{\text{aux}}}$ that is such that $F^{\mathcal{L}}(\mathbf{z})$ is superior to a well-chosen threshold T and store it in a set of "undecoded candadidates".

Definition 38 (Set of undecoded candidates)**.** *We define the set of undecoded candidates as*

$$\mathcal{S} \stackrel{def}{=} \{\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}} \, : \, F^{\mathcal{L}}(\mathbf{z}) \geqslant T\}.$$

Algorithm 19 The LPN solver

Name: double-RLPN-LPN-Solver
Input: $\mathcal{L}$ ▷ *$\mathcal{L}$ is a list of* LPN *samples of the form* $(\mathbf{a}, b)$ *where* $b = \langle \mathbf{a}, \mathbf{s} \rangle + e$ *and* $\mathbf{s} \in \mathbb{F}_2^{k_{\text{aux}}}$
Parameter: T ▷ *T is the threshold from which we keep the candidates.*
1: $F^{\mathcal{L}} \leftarrow$ FFT-LPN-Solver$(\mathcal{L})$
2: $\mathcal{S} \leftarrow \{\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}} \ : \ F^{\mathcal{L}}(\mathbf{z}) \geqslant T\}$
3: **return** $\mathcal{S}$

We have that

Fact 22. *Using the same notation as in Fact 21 we have that for* $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$,

$$F^{\mathcal{L}}(\mathbf{z}) = \sum_{\substack{(\mathbf{h}, \mathbf{e}_{\text{aux}}) \in \mathscr{H} \\ \mathbf{m}_{\text{aux}} : \mathbf{m}_{\text{aux}} \mathbf{G}_{\text{aux}} + \mathbf{e}_{\text{aux}} = \mathbf{h}_{\mathscr{P}}}} (-1)^{\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{z}, \mathbf{m}_{\text{aux}} \rangle}.$$

In particular the score function evalutated on the secret is

$$F^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\intercal}) = \sum_{(\mathbf{h}, \mathbf{e}_{\text{aux}}) \in \mathscr{H}} (-1)^{\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{e}_{\mathscr{P}}, \mathbf{e}_{\text{aux}} \rangle}$$

6.2.1.4 Recovering the rest of the error

We are given the set $\mathcal{S}$ containing candidates $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ for the secret, $\mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\intercal}$. From those candidates, we want to recover the whole error $\mathbf{e}$ if the secret is in the list of candidate else it returns fails, namely $\perp$. Importantly in our design rationale this step should not dominate the complexity of double-RLPN: namely it should be exponentially less costly than say the FFT step. It turns out that the procedure we describe next is sufficient for our purpose.

Recall that the set of undecoded candidates $\mathcal{S}$ is composed of candidates $\mathbf{z}$ for $\mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\intercal}$. But note that we can recover $\mathbf{e}_{\mathscr{P}}$ from $\mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\intercal}$ by solving yet another smaller decoding problem. Indeed, if our bet is valid then $|\mathbf{e}_{\mathscr{P}}| = t - u$, thus recovering $\mathbf{e}_{\mathscr{P}}$ from $\mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\intercal}$ is nothing but decoding at distance $t-u$ the syndrome $\mathbf{e}_{\mathscr{P}} \mathbf{G}_{\text{aux}}^{\intercal}$ in the code of parity-check $\mathbf{G}_{\text{aux}}$, namely $\mathcal{C}_{\text{aux}}^{\perp}$. This is done with a procedure Recover-$\mathbf{e}_{\mathscr{P}}(\mathcal{C}_{\text{aux}}^{\perp}, \mathbf{z}, \ t-u)$ that we instantiate later and which is called for each candidate $\mathbf{z} \in \mathcal{S}$. It returns a subset $\mathcal{V} \subset \{\mathbf{x} \in \mathcal{S}_{t-u}^{s} \ : \ \mathbf{x} \mathbf{G}_{\text{aux}}^{\intercal} = \mathbf{z}\}$ but in practice essentially all the set is computed. With this, we construct the set $\mathcal{S}'$ of decoded candidates $\mathbf{x}$ for $\mathbf{e}_{\mathscr{P}}$ by taking the union of the output of the call of the previous decoder.

From this set of candidates for $\mathbf{e}_{\mathscr{P}}$ we proceed to test them and recover $\mathbf{e}_{\mathscr{N}}$ as in RLPN, namely using the procedure RLPN-Recover-full-error which we recall here. Note that, contrary to RLPN our analysis will show that the set $\mathcal{S}'$ of candidates for $\mathbf{e}_{\mathscr{P}}$ can be of exponential size therefore we will need to use some more advanced decoder for this last step. We choose ISDDumer [Dum89] as it is simple and sufficient for our purpose.

Algorithm 20 Recovering the rest of the error

Name: double-RLPN-Recover-full-error
Input: $\mathcal{C}$, $\mathscr{P}$, $\mathscr{N}$, $\mathcal{C}_{\text{aux}}$, $\mathcal{S}$ ▷ *$\mathcal{S}$ is a list of candidates $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ for $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$*
Parameter: u

1: **If** $|\mathcal{S}| > \text{MaxSize}$ **then go to** Line 1 of Algorithm 16 **else continue** ▷ MaxSize *is an additional parameter which is here to ensure that we are not in a pathological situation where the size of $\mathcal{S}$ would be much larger than its expected value.* MaxSize *will be chosen around* $n\mathbb{E}(|\mathcal{S}|)$
2: $\mathcal{S}' \leftarrow \emptyset$
3: **for** $\mathbf{z} \in \mathcal{S}$ **do**
4: $\quad \mathcal{V} \leftarrow \text{Recover-}\mathbf{e}_{\mathscr{P}}(\mathcal{C}_{\text{aux}}, \mathbf{z}, t-u)$
5: $\quad \mathcal{S}' \leftarrow \mathcal{S}' \bigcup \mathcal{V}$
6: **return** RLPN-Recover-full-error$(\mathcal{C}, \mathscr{P}, \mathscr{N}, \mathcal{S}, u)$

Algorithm 21 Recovering the rest of the error

Name: RLPN-Recover-full-error
Input: $\mathcal{C}$, $\mathscr{P}$, $\mathscr{N}$, $\mathcal{S}$ ▷ *$\mathcal{S}$ is a list of candidates $\mathbf{x} \in \mathcal{S}_{t-u}^{s}$ for $\mathbf{e}_{\mathscr{P}}$*
Parameter: u

1: **for** $\mathbf{x} \in \mathcal{S}$ **do**
2: $\quad \mathbf{R} \leftarrow \text{Lift}\left(\mathcal{C}^{\perp}, \mathscr{N}\right)$
3: $\quad \mathbf{y}' \leftarrow \mathbf{y}_{\mathscr{N}} - (\mathbf{y}_{\mathscr{P}} - \mathbf{x})\,\mathbf{R}$
4: $\quad \mathbf{z} \leftarrow \text{ISD-Dumer}(\mathcal{C}^{\mathscr{N}}, \mathbf{y}', u)$ ▷ *This is the* ISD *decoder from [Dum89]*
5: $\quad$ **if** $\mathbf{z} \neq \perp$ **then**
6: $\quad\quad$ Construct $\mathbf{e}$ such that $\mathbf{e}_{\mathscr{P}} = \mathbf{x}$ and $\mathbf{e}_{\mathscr{N}} = \mathbf{z}$
7: $\quad\quad$ **return** $\mathbf{e}$
8: **return** $\perp$

6.2.2 Choosing a good auxiliary code

6.2.2.1 Juxtaposition code with a constant number of blocks

We devise here a simple family of linear codes and a decoder which essentially achieves the rate distortion bound for a BSC channel while having an efficient amortized time decoder. We argue that this is easily done as long as we want to decode an exponential number of elements in the ambient space. Namely, we state the following.

Proposition 44. *Let n, k, t and $\lambda > 0$ be a positive constant. There exists a family of linear codes $\mathcal{F}$ of length n and dimension k and some algorithm which is such that given $\mathcal{C} \sim \mathcal{U}(\mathcal{F})$, a list $\mathcal{L}$ of at least $2^{\lambda n}$ vectors of $\mathbb{F}_2^n$, each taken uniformly at random, and a decoding distance t, outputs for each element $\mathbf{y} \in \mathcal{L}$ a set* $\text{Decode}(\mathcal{C}, \mathbf{y}, t) \subset \{\mathbf{e} \in \mathcal{S}_t^n \,:\, \mathbf{y} - \mathbf{e} \in \mathcal{C}\}$ *which is optimal in the sense that*

$$\mathbb{P}\left(|\text{Decode}(\mathcal{C}, \mathbf{y}, t)| \neq 0\right) = \frac{1}{\text{poly}(n)} \min\left(\frac{\binom{n}{t}}{2^{n-k}}, 1\right)$$

and where the $\text{poly}(n)$ *term is of the order $n^{\mathcal{O}(1/\lambda)}$ for some universal $\mathcal{O}()$. Lastly, provided*

that $\binom{n}{t}/2^{n-k} = \mathrm{poly}(n)$ *the expected time and memory complexity of this algorithm are*

$$\mathrm{poly}(n)\,|\mathcal{L}|\,.$$

We do not provide a complete proof of this proposition because it do not technically be needed for our purpose but we give the main elements in this section. A very simple approach to achieve this would be optimal in terms of rate distortion would be to take a random $[n,k]$-linear code, enumerate all possible errors of weight t and store their associated syndrome in a table. Decoding an element of the list onto the code is then simply searching in this table for a specific syndrome. This algorithm decodes $\mathcal{L}$ in amortized time $\mathrm{poly}(n)$ as long as $\binom{n}{t} \leqslant |\mathcal{L}|$, or, if $t = d_{\mathrm{GV}}(n,\ k)$ as long as $2^{n-k} < |\mathcal{L}|$. The idea then, is simply to take a juxtaposition code with a constant number of blocks b and enumerate all the errors of weight $\frac{t}{b}$ independently on each part of length $\frac{t}{b}$. Now, our decoding algorithm works in amortized time $\mathrm{poly}(n)$ as long as $\sqrt[b]{2^{n-k}} < |\mathcal{L}|$ and the constant nature of b allows us to argue that this error pattern is optimal. As long as is at least $|\mathcal{L}| > 2^{\lambda\, n}$ for some constant $\lambda > 0$ then there exists a constant b verifying this.

6.2.2.1.1 Definition

Definition 39 (Juxtaposition code)**.** *We define the set of juxtaposition codes with b blocks, and of length n and dimension k, namely $\mathfrak{C}^{\mathrm{juxt}}[b,\ n,\ k]$, as the set of linear codes $\mathcal{C}$ such that there exists some constituent code $\mathcal{C}^{(i)} \in \mathfrak{C}\left[n^{(i)}, k^{(i)}\right]$ such that*

$$\mathcal{C} = \mathcal{C}^{(1)} \times \mathcal{C}^{(2)} \times \cdots \times \mathcal{C}^{(b)}$$

where we denote implicitly (in b), for each integer $v \in \mathbb{N}$ its i'th part as:

$$v^{(i)} \stackrel{def}{=} \begin{cases} \lfloor v/b \rfloor + 1 & \text{if } i \leqslant (v \mod b) \\ \lfloor v/b \rfloor & \text{else} \end{cases} \tag{6.3}$$

When $\mathcal{C} \in \mathfrak{C}^{\mathrm{juxt}}[b,\ s,\ k_{\mathrm{aux}}]$ we denote by $\mathcal{C}^{(i)}$ its associated constituent code on the part i.

Fact 23. *For $v \in \mathbb{N}$ we have that $v = \sum_{i=1}^{b} v^{(i)}$. Moreover*

$$\mathfrak{C}^{\mathrm{juxt}}[b,\ n,\ k] \subset \mathfrak{C}[n,k]$$

Notation 5. *When the context is clear we will implicitly denote the i'th part of a vector $\mathbf{x} \in \mathbb{F}_2^s$ relative to the support given by n by $\mathbf{x}^{(i)} \stackrel{def}{=} \mathbf{x}_{\mathscr{I}}$ where $\mathscr{I} \stackrel{def}{=} [\![\sum_{j=1}^{i-1} n^{(j)},\ \sum_{j=1}^{i} n^{(j)}]\!]$. In the same manner, given $\mathcal{C} \in \mathfrak{C}^{\mathrm{juxt}}[b,\ n,\ k]$ we denote by $\mathcal{C}^{(i)}$ its i'th constituent code.*

Definition 40 (Set of admissible errors)**.** *Let $b, n, k, t \in \mathbb{N}$. Let $\mathcal{C} \in \mathfrak{C}^{\mathrm{juxt}}[b,\ n,\ k]$ be a juxtaposition code of length n and dimension k. Let $\mathbf{y} \in \mathbb{F}_2^n$, we define the set of admissible errors as*

$$Dec^{\mathrm{juxt}}(\mathcal{C}, \mathbf{y}, t) \stackrel{def}{=} \{\mathbf{e} \in \mathcal{S}_t^n \ :\ \left|\mathbf{e}^{(i)}\right| = t^{(i)} \text{ and } \mathbf{y}^{(i)} - \mathbf{e}^{(i)} \in \mathcal{C}^{(i)},\ \forall i \in [\![1,\ b]\!]\}.$$

We have in particular that

$$Dec^{\mathrm{juxt}}(\mathcal{C}, \mathbf{y}, t) \subset \{\mathbf{e} \in \mathcal{S}_t^n \ :\ \mathbf{e} + \mathbf{y} \in \mathcal{C}\}.$$

6.2.2.1.2 Optimality of juxtaposition codes We state the following proposition about the optimality of juxtaposition codes and the associted set of admissible errors and give some proof elements.

Proposition 45 (Asymptotic optimality of juxtaposition codes decoded on admissible errors)**.** *Let $n \in \mathbb{N}$ growing to infinity and let $k,\ t, b \in \mathbb{N}$ be implicit function of n such that $b = \mathcal{O}(1)$. Let $\mathcal{C} \sim \mathcal{U}\left(\mathfrak{C}^{\text{juxt}}\left[b,\ n,\ k\right]\right)$ be a uniformly random juxtaposition code of length n and dimension k. Let $\mathbf{y} \sim \mathcal{U}\left(\mathbb{F}_2^n\right)$. We have that*

$$\mathbb{P}\left(\left|Dec^{\text{juxt}}\left(\mathcal{C}, \mathbf{y}, t\right)\right| \neq 0\right) = \frac{1}{\text{poly}\left(n\right)} \min\left(\frac{\binom{n}{t}}{2^{n-k}},\ 1\right).$$

It could be shown by proving the following lemma

Lemma 22. *Using the notations of Proposition 45 we have that*

$$\mathbb{E}\left(\left|Dec^{\text{juxt}}\left(\mathcal{C}, \mathbf{y}, t\right)\right|\right) = \frac{1}{2^{n-k}} \prod_{i=1}^{b} \binom{n^{(i)}}{t^{(i)}},$$

$$\textbf{Var}\left(\left|Dec^{\text{juxt}}\left(\mathcal{C}, \mathbf{y}, t\right)\right|\right) \leqslant \mathbb{E}\left(\left|Dec^{\text{juxt}}\left(\mathcal{C}, \mathbf{y}, t\right)\right|\right) + \sum_{\substack{\mathbf{x} \in \{0,1\}^b \\ \mathbf{x} \neq \mathbf{0} \text{ and } \mathbf{x} \neq \mathbf{1}}} \prod_{i=1}^{b} \left(\frac{\binom{n^{(i)}}{t^{(i)}}}{2^{n^{(i)}-k^{(i)}}}\right)^{x_i+1}.$$

The proof of the proposition then directly follows from applying Byenemé-Chebychev inequality to $\left|\text{Dec}^{\text{juxt}}\left(\mathcal{C}, \mathbf{y}, t\right)\right|$ and using the following fact.

Fact 24. *We use the notation of Proposition 45. Because $b = \mathcal{O}(1)$ we have that*

$$\frac{1}{2^{n-k}} \prod_{i=1}^{b} \binom{n^{(i)}}{t^{(i)}} = \frac{1}{\text{poly}\left(n\right)} \frac{\binom{n}{t}}{2^{n-k}}.$$

6.2.2.1.3 An efficient amortized decoding algorithm Let us now give more precisely the decoder. We have the syndrome precomputation phase.

Algorithm 22 Pre-compute all syndromes

Name: JUXT-PRECOMPUTESYNDROME
Input: $\mathcal{C} \in \mathfrak{C}^{\text{juxt}}\left[b,\ n,\ k\right],\ t$
  for $i \in [\![1,\ b]\!]$ **do**
    $\mathbf{H}^{(i)} \leftarrow \mathbf{H}\left(\mathcal{C}^{(i)}\right)$
    $\mathcal{S}^{(i)} \leftarrow \emptyset$
    for $\mathbf{e}^{(i)} \in \mathcal{S}_{t^{(i)}}^{n^{(i)}}$ **do**
      $\mathbf{s}^{(i)} \leftarrow \mathbf{H}^{(i)}\mathbf{e}^{(i)}$
      $\mathcal{S}^{(i)}$.APPEND$\left(\left(\mathbf{s}^{(i)},\ \mathbf{e}^{(i)}\right)\right)\}$
  $\mathcal{S} \leftarrow \{\left(i, \mathcal{S}^{(i)}\right)\ :\ i \in [\![1,\ b]\!]\}$
  return $\mathcal{S}$ is done in Section 6.3.2 where we verify Conjecture 3 and

Having access to this precomputed hash table one can compute the set of admissible errors as follows.

Algorithm 23 Decode with pre-computation

Name: Juxt-Decode
Input: a pointer to $\mathcal{S}$ the output of Algorithm 22, $\mathbf{y} \in \mathbb{F}_2^n$
Output: $\mathrm{Dec}^{\mathrm{juxt}}(\mathcal{C}, \mathbf{y}, t)$

```
for i ∈ ⟦1, b⟧ do
    H^(i) ← H(C^(i))
    s^(i) ← H^(i) y^(i)
    D^(i) ← ∅
    for e^(i) such that (s^(i), e^(i)) ∈ S^(i) do
        D^(i).append(e^(i))
return D ← {e ∈ F_2^n : e^(1), ..., e^(b) ∈ D^(1) × ... D^(b)}
```

Finally we get the following

Algorithm 24 Decoder

Name: Juxt-Decode-List
Input: $\mathcal{C} \in \mathfrak{C}^{\mathrm{juxt}}[b,\ n,\ k]$, t the sub-decoding distances, $\mathcal{L}$ a list of vectors of $\mathbb{F}_2^n$
Output: $\{(\mathbf{y},\ \mathrm{Dec}^{\mathrm{juxt}}(\mathcal{C}, \mathbf{y}, t)) : \mathbf{y} \in \mathcal{L}\}$ the list $\mathcal{L}$ decoded onto $\mathcal{C}$

```
S ← Juxt-PrecomputeSyndrome(C)
D ← ∅
for y ∈ L do
    V ← Juxt-Decode(a pointer to S, y)
    D.append((y, V))
return D
```

Proposition 46. *Let n be growing to infinity and let $k,\ t, b$ be implicit functions of n and let $\lambda > 0$ be a positive constant. Let $\mathcal{L}$ be a list of size at least $2^{\lambda n}$ of vectors taken uniformly at random in $\mathbb{F}_2^n$. Then, provided that $\binom{n}{t}/2^{n-k} = \mathrm{poly}(n)$ Algorithm 24 with input $\mathcal{C},\ t,\ \mathcal{L}$ outputs $\{(\mathbf{y},\ Dec^{\mathrm{juxt}}(\mathcal{C}, \mathbf{y}, t)) : \mathbf{y} \in \mathcal{L}\}$ with time and memory*

$$\mathrm{poly}(n)\,|\mathcal{L}|\,.$$

The proof is straightforward.

6.2.2.2 DoubleRLPN with juxtaposition codes

In this section we give the instantiation of Algorithm 16 with juxtaposition codes.

Algorithm 3 (DoubleRLPN Algorithm with juxtaposition codes)**.** *We define an instantiation of Algorithm 16 where we have an additionnal parameter $b \in \mathbb{N}$ and where:*

- *The family $\mathcal{F} \subset \mathfrak{C}[s, k_{\mathrm{aux}}]$ of auxiliary codes is defined as*

$$\mathcal{F} = \mathfrak{C}^{\mathrm{juxt}}[b,\ s,\ k_{\mathrm{aux}}]$$

- *We instantiate the procedure* Sample-Auxilary-Code($\mathcal{F}$) *which returns a code $\mathcal{C}_{\mathrm{aux}}$ which was chosen uniformly at random in $\mathfrak{C}^{\mathrm{juxt}}[b,\ s,\ k_{\mathrm{aux}}]$, by choosing each of its constituent code $\mathcal{C}_{\mathrm{aux}}^{(i)} \sim \mathcal{U}(s^{(i)},\ k^{(i)})$, i.e. each of them is uniformly random $[s^{(i)},\ k^{(i)}]$-linear code.*

- *After having chosen the code* $\mathcal{C}_{\text{aux}}$ *we precompute the table of syndromes with a call to* SyndromeTable $\leftarrow$ Juxt-PrecomputeSyndrome$(\mathcal{C}_{\text{aux}}; t_{\text{aux}})$. *Each call to* Auxilary-Decode$(\mathcal{C}_{\text{aux}},\ \mathbf{a}; t_{\text{aux}})$ *is then replaced with one call to* Juxt-Decode(SyndromeTable, $\mathbf{a}$).

- *The* Recover-$\mathbf{e}_{\mathscr{P}}(\mathcal{C}_{\text{aux}}^{\perp},\ \mathbf{z},\ t-u)$ *procedure is some slight adaptation of the previous decoding* Auxilary-Decode *procedure and returns the set*

$$\{\mathbf{x} \in \mathcal{S}_{t-u}^{s} \ :\ \left|\mathbf{x}^{(i)}\right| = t_{\text{aux}}^{(i)} \text{ and } \mathbf{x}^{(i)} \left(\mathbf{G}_{\text{aux}}^{(i)}\right)^{\mathsf{T}} = \mathbf{z}^{(i)}, \quad \forall i \in [\![1,\ b]\!]\}$$

  where $\mathbf{G}_{\text{aux}}^{(i)} \stackrel{def}{=} \mathbf{G}\left(\mathcal{C}_{\text{aux}}^{(i)}\right)$.

Note also that the design rationale behind the reduction from sparse-LPN to plain-LPN is that each of the samples $\mathbf{h}_{\mathscr{P}}$ gets decoded onto one close codeword of $\mathcal{C}_{\text{aux}}$ (preferably the closest), and that in general our algorithm does not gain by decoding on multiple codewords. However, in our instantiation, we fix the decoding distance and each of the samples $(\mathbf{h}_{\mathscr{P}},\ \langle \mathbf{y}, \mathbf{h} \rangle)$ is decoded to the set $\text{Dec}^{\text{juxt}}(\mathcal{C}_{\text{aux}}, \mathbf{h}_{\mathscr{P}}, t_{\text{aux}})$, yielding for each $\mathbf{h}$ possibly many reduced plain-LPN samples. We do this only because it makes the analysis slighty simpler and claim that doing otherwise changes the complexity of our attack by no more than a polynomial factor. Moreover, it is clear that we have no advantage whatsoever of taking t_{aux} significantly bigger than $d_{\text{GV}}(s,\ k_{\text{aux}})$, that is, all our cases of interest are such that

$$\frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} \leqslant \text{poly}(n)$$

and in this regime we can show that the expected size of $\text{Dec}^{\text{juxt}}(\mathcal{C}_{\text{aux}}, \mathbf{h}_{\mathscr{P}}, t_{\text{aux}})$ is poly-bounded.

Lastly, it is readily seen that the cost of precomputing the syndrome to decode $\mathcal{C}_{\text{aux}}$ and precomputing the syndrome to decode $\mathcal{C}_{\text{aux}}^{\perp}$ as it is done in the procedure Recover-$\mathbf{e}_{\mathscr{P}}$ are both upper bounded by some $\text{poly}(n)\, 2^{s/b}$. As such, taking $b = s/k_{\text{aux}}$ is sufficient to make sure that both these costs do not dominate in front of say the cost of the FFT. All in all we can show that we have the following.

Proposition 47 (Complexity)**.** *Let* $n \in \mathbb{N}$ *and let* $k, t \in \mathbb{N}$ *and let* $s,\ k_{\text{aux}},\ t_{\text{aux}},\ u,\ w$ *be some parameters and* N_{iter}, T *be the number of iterations and the threshold respectively, all implicit functions of* n. *Suppose that* $\binom{s}{t_{\text{aux}}}/2^{s-k_{\text{aux}}} = \text{poly}(n)$ *and choose the number of blocks as* $b \stackrel{def}{=} \lceil s/k_{\text{aux}} \rceil$. *The expected time complexity of Algorithm 3, given as input an instance of* $\text{DP}_{\mathbf{G}}(n, k, t)$ *is*

$$\text{poly}(n)\left(\underbrace{N_{\text{iter}}}_{\text{Number iterations}}\left(\underbrace{T_{\text{eq}}}_{\text{Computing the LPN samples}} + \underbrace{2^{k_{\text{aux}}}}_{\text{FFT}} + \underbrace{\mathbb{E}\left(\left|\mathcal{S}'\right|\right) \times T_{\text{dec}}}_{\text{Recovering the rest of the error}}\right)\right)$$

where

1. T_{eq} *is time complexity of* Compute-Short-Vectors.
2. T_{dec} *is the time complexity of [Dum89] to solve* $\text{DP}_{\mathbf{G}}(n-s, k-s, u)$ *with probability* $1 - o(1)$.

6.3 Outline of the analysis and conjecture

6.3.1 Outline of the analysis and key quantities

The goal of this section is to introduce the key quantities and the two main steps intervening in the analysis. We will focus our analysis in the special case when the procedure computing the dual vectors essentially outputs all vectors of weight w on $\mathscr{N}$. More, to simplify the discussion we assume in this introductory section that the number of blocks in the auxiliary code is $b = 1$ so here really $\mathcal{C}_{\text{aux}}$ is taken uniformly at random in $\mathfrak{C}[s, k_{\text{aux}}]$, the set of all $[s, k_{\text{aux}}]$ linear codes. We make the full fledge statements later and show that essentially everything we say remains unchanged as long as $b = \mathcal{O}(1)$. The complete and full analysis (including the proofs) will be made in Section 6.5.3.

Notation 6. *The parameters $n, k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}} \in \mathbb{N}$ are implicit functions of $n \in \mathbb{N}$. The number of blocks is $b = 1$.*

- *$\mathscr{P}$ and $\mathscr{N}$ are any fixed complementary subsets of $[\![1, n]\!]$ of size s and $n-s$ respectively.*
- *$\mathcal{C}$ is a linear code of length n such that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and $\mathcal{C}_{\text{aux}}$ is an $[s, k_{\text{aux}}]$-linear code and $\mathbf{G}_{\text{aux}}$ is any generator matrix of $\mathcal{C}_{\text{aux}}$.*
- *$\mathbf{y} = \mathbf{e}$ where $\mathbf{e} \in \mathcal{S}_t^n$ is fixed such that $|\mathbf{e}_{\mathscr{N}}| = u$*
- *The set of decoded dual vectors is*

$$\mathscr{H} \stackrel{def}{=} \{(\mathbf{h},\ \mathbf{m}_{\text{aux}}) \in \mathcal{C}^{\perp} \times \mathbb{F}_2^{k_{\text{aux}}} \ :\ |\mathbf{h}_{\mathscr{N}}| = w \text{ and } |\mathbf{h}_{\mathscr{P}} + \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}}| = t_{\text{aux}}\}.$$

- *The score function is*

$$\forall \mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}\ F^{\mathcal{L}}(\mathbf{z}) \stackrel{def}{=} \sum_{(\mathbf{h},\ \mathbf{m}_{\text{aux}}) \in \mathscr{H}} (-1)^{\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{m}_{\text{aux}}, \mathbf{z} \rangle}.$$

This section is composed of the three following subsections, the first giving the main quantities intervening in the analysis. The second gives a minimal condition on the parameters to guarantee that the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ of our LPN problem is distinguishable from a bad guess $\mathbf{z} \neq \mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ with good probability. This is done by devising a second order concentration bound on the score function. Last, we devise a lower bound on the probability that $\mathbf{z} \neq \mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ is indistinguishable from the secret and conjecture that this lower bound is tight: this will allow us to estimate the number of false candidates.

6.3.1.1 Key quantities: number of LPN samples and bias of the error

6.3.1.2 Intuition for the result and main constraint

Note that we expect that there are $\binom{n-s}{w}/2^{k-s}$ dual vectors of weight w on $\mathscr{N}$ and each of these dual vectors will be decoded into an expected number of $\binom{s}{t_{\text{aux}}}/2^{s-k_{\text{aux}}}$ codewords.

Lemma 23 (Expected number of LPN samples)**.** *The expected number of* LPN *samples is*

$$\mathbb{E}(|\mathscr{H}|) = \frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}}$$

where $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\ k)$ conditioned on the event that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and $\mathcal{C}_{\text{aux}} \sim \mathcal{U}(s,\ k_{\text{aux}})$.

Let us now give the bias of the LPN samples. Given a decoded dual vector $(\mathbf{h},\ \mathbf{m}_{\text{aux}}) \in \mathscr{H}$ we get the following associated reduced LPN sample:

$$(\mathbf{m}_{\text{aux}},\ \langle \mathbf{y}, \mathbf{h}\rangle) = (\mathbf{m}_{\text{aux}},\ \langle \mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}, \mathbf{m}_{\text{aux}}\rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}}\rangle + \langle \mathbf{e}_{\mathscr{P}}, \mathbf{e}_{\text{aux}}\rangle)$$

where

$$\mathbf{e}_{\text{aux}} \stackrel{\text{def}}{=} \mathbf{h}_{\mathscr{P}} + \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} \in \mathcal{S}^{s}_{t_{\text{aux}}}.$$

Recall that we supposed that

$$|\mathbf{e}_{\mathscr{N}}| = u \qquad \text{and} \qquad |\mathbf{e}_{\mathscr{P}}| = t - u.$$

Making the approximation that we can forget about the code structure, a rough approximation for the bias of the error of the previous LPN sample can be obtained with the Piling-up lemma and supposing that everything is independent and well-distributed

$$\text{bias}\left(\left\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}'_{\mathscr{N}}\right\rangle + \left\langle \mathbf{e}_{\mathscr{P}}, {\mathbf{e}_{\text{aux}}}'\right\rangle\right) = \delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u) \text{ where } \begin{cases} \mathbf{h}'_{\mathscr{N}} \sim \mathcal{U}\left(\mathcal{S}_w^{n-s}\right) \\ {\mathbf{e}_{\text{aux}}}' \sim \mathcal{U}\left(\mathcal{S}^{s}_{t_{\text{aux}}}\right) \\ \mathbf{h}'_{\mathscr{N}} \text{ and } {\mathbf{e}_{\text{aux}}}' \text{ are independent} \end{cases}.$$

So in essence if our LPN sample behaved like a true LPN sample where everything is independent and well-distributed we would expect that it is sufficient that the number of LPN samples considered is greater than the inverse of the bias squared, namely

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} = \frac{\text{poly}(n)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}$$

so that we could information-theoretically recover the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$. This translates into the fact that the score function evaluated on the secret $F^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal})$ is greater than the score function $F^{\mathcal{L}}(\mathbf{z})$ for any bad guesses $\mathbf{z} \neq \mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$. This will be our main constraint on the parameters. Because our LPN samples are not well distributed the analysis is more complicated.

6.3.1.3 Second-order behavior of the score function

In this section we formalize that the above constraint naturally appears in our analysis.

Lemma 24 (Expected value and variance of the score function on the secret)**.** *Let* $N \stackrel{def}{=} \binom{n-s}{w}\binom{s}{t_{\text{aux}}}/2^{k-k_{\text{aux}}}$ *and suppose furthermore that* $\binom{s}{t_{\text{aux}}}/2^{s-k_{\text{aux}}} \leqslant 1$*. We have that*

$$\begin{aligned} \mathbb{E}\left(F^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal})\right) &= N\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u), \\ \mathbf{Var}\left(F^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal})\right) &= \mathcal{O}(N) \end{aligned}$$

where $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n,\ k)$ *conditioned on the event that* $\dim(\mathcal{C}_{\mathscr{P}}) = s$ *and* $\mathcal{C}_{\text{aux}} \sim \mathcal{U}(s,\ k_{\text{aux}})$*.*

This lemma allows us to derive the second-order behavior of the score function on the secret.

Corollary 9. *Under the same conditions and distributions as in Lemma 24 we have that*

$$\mathbb{P}\left(\left|F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)-N\delta_{w}^{(n-s)}\left(u\right)\delta_{t_{\text{aux}}}^{(s)}\left(t-u\right)\right|\geqslant n\sqrt{N}\right)=\mathcal{O}(1/n)$$

In the same manner, we could also similarly show that the score function evaluated on a bad guess $\mathbf{z}\neq\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ satisfies $\mathbb{E}\left(F^{\mathcal{L}}\left(\mathbf{z}\right)\right)=0$ and that $\mathbf{Var}\left(F^{\mathcal{L}}\left(\mathbf{z}\right)\right)=\mathcal{O}(N)$ which allows us to derive that for any positive function f we have that

$$\mathbb{P}\left(\left|F^{\mathcal{L}}\left(\mathbf{z}\right)\right|\geqslant f(n)\sqrt{N}\right)=\mathcal{O}(1/f(n))\qquad\text{when }\mathbf{z}\neq\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}.$$

This allows us to show that we can hope to distinguish the secret from a bad guess with probability $1-o(1)$ as long as

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}}\in\frac{\omega\left(1\right)}{\left(\delta_{w}^{(n-s)}\left(u\right)\delta_{t_{\text{aux}}}^{(s)}\left(t-u\right)\right)^{2}}.$$

This is done by taking the threshold T as

$$T=\frac{1}{2}N\delta_{w}^{(n-s)}\left(u\right)\delta_{t_{\text{aux}}}^{(s)}\left(t-u\right)$$

and keeping in the set of candidates $\mathcal{S}$ only those elements whose evaluations are above T. As in RLPN, our intuition is that if this condition is not met (up to polynomial factors), then we have no usable advantage to distinguish good from bad guesses, and that, depending on the value of T, the set of candidates $\mathcal{S}$ will be either essentially the whole space $\mathbb{F}_{2}^{k_{\text{aux}}}$ or entirely empty.

6.3.1.4 A conjecture on the exponential tail of the score function

So now we have our minimal condition to distinguish the secret against a specific bad candidate $\mathbf{z}\neq\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$, but the problem in end is now to quantify our ability to distinguish the secret from all bad guesses $\mathbf{z}\in\mathbb{F}_{2}^{k_{\text{aux}}}\setminus\{\ \mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\}$. Here our second order concentration bound are useless as the search space is exponentially big, hence we need an exponential strengthening of the shape

$$\mathbb{P}\left(\left|F^{\mathcal{L}}\left(\mathbf{z}\right)\right|\geqslant\text{poly}\left(n\right)\sqrt{N}\right)=2^{-\Omega(k_{\text{aux}})}.$$

We are unable to prove so but devise here a lower bound on this probability and conjecture that this lower bound is tight up to polynomial factors. Due to the linear dependencies between the quantities involved in our LPN samples, we can determine a sufficient condition for $\mathbf{z}$ to be indistinguishable from the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$.

Lemma 25 (Linearity relation)**.** *Let* $(\mathbf{h},\ \mathbf{m}_{\text{aux}})\in\mathscr{H}$ *and let* $\mathbf{e}_{\text{aux}}$ *be the associated auxiliary error, namely* $\mathbf{e}_{\text{aux}}\overset{def}{=}\mathbf{h}_{\mathscr{P}}+\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}}$*. Let* $\mathbf{z}\in\mathbb{F}_{2}^{k_{\text{aux}}}$ *and let* $\mathbf{x}\in\mathbb{F}_{2}^{s}$ *be such that* $\mathbf{x}\mathbf{G}_{\text{aux}}^{\intercal}=\mathbf{z}$*. For any* $(\mathbf{r},\ \mathbf{z})\in\mathcal{D}^{(\mathbf{x})}$ *we have that*

$$\langle\mathbf{y},\mathbf{h}\rangle-\langle\mathbf{z},\mathbf{m}_{\text{aux}}\rangle=\langle\mathbf{r},\mathbf{e}_{\text{aux}}\rangle+\langle\mathbf{z},\mathbf{h}_{\mathscr{N}}\rangle\tag{6.4}$$

where

$$\begin{aligned}\mathcal{D}^{(\mathbf{x})}&\overset{def}{=}\{(\mathbf{r},\mathbf{z})\in\mathbb{F}_{2}^{s}\times\mathbb{F}_{2}^{n-s}\ :\ \mathbf{r}\in\mathcal{C}_{\text{aux}}^{\perp}+\mathbf{x},\quad\mathbf{z}\in\mathcal{C}^{\mathscr{N}}+(\mathbf{r}+\mathbf{e}_{\mathscr{P}})\,\mathbf{R}+\mathbf{e}_{\mathscr{N}}\},\\ \mathbf{R}&\overset{def}{=}\text{Lift}\left(\mathcal{C}^{\perp},\ \mathscr{N}\right).\end{aligned}$$

This lemma is simply a series of rewritings and is proved in Section 6.3.1.4.1. Notably, we see directly that if $\mathcal{D}^{(\mathbf{x})}$ contains a low-weight codeword then $\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{z}, \mathbf{m}_{\text{aux}} \rangle$ will be biased toward 0 and thus $F^{\mathcal{L}}(\mathbf{z})$ will be big.

Definition 41 (Weight enumerator of the code $\mathcal{D}^{(\mathbf{x})}$)**.**

$$N_{j,i}^{(\mathbf{x})} \stackrel{def}{=} \mathcal{D}^{(\mathbf{x})} \bigcap \mathcal{S}_j^s \times \mathcal{S}_i^{n-s}.$$

More precisely, if $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ and $\mathbf{x} \in \mathbb{F}_2^s$ are such that $\mathbf{z} = \mathbf{x}\mathbf{G}_{\text{aux}}^{\intercal}$ we have that

$$\begin{aligned}
&\mathbb{E}\left(F^{\mathcal{L}}(\mathbf{z}) \,\middle|\, \exists(\mathbf{r},\ \mathbf{z}) \in \mathcal{D}^{(\mathbf{x})}\ :\ |\mathbf{r}| = v_1 \text{ and } |\mathbf{a}| = v_2\right) \\
&= \mathbb{E}\left(F^{\mathcal{L}}(\mathbf{z}) \,\middle|\, N_{v_1,v_2} \neq 0\right) \\
&\approx N\delta_w^{(n-s)}(v_2)\,\delta_{t_{\text{aux}}}^{(s)}(v_1).
\end{aligned}$$

Now, we recall that from Lemma 24, when the bet on the error is valid we have that for the secret

$$\mathbb{E}\left(F^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal})\right) = N\,\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u).$$

As such clearly, if $v_1 = t-u$ and $v_2 = u$ then we expect that $\mathbf{z}$ will be mistaken for the secret. But more generally we expect that this is the case if

$$\delta_w^{(n-s)}(v_2)\,\delta_{t_{\text{aux}}}^{(s)}(v_1) \approx \delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u).$$

Our conjecture will be that the event that $\mathbf{z}$ is mistaken for the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ is overwhelmingly dominated by the event that such a low weight codeword exists. We take a small margin around these values to account for the usual variation of the score function.

Conjecture 3. *If*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \Omega\left(\frac{n^8}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}\right), \quad \frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} \in \mathcal{O}(1) \tag{6.5}$$

then we conjecture that

$$\mathbb{P}\left(F^{\mathcal{L}}(\mathbf{x}\mathbf{G}_{\text{aux}}^{\intercal}) \geqslant T\right) \in \widetilde{\mathcal{O}}\big(\mathbb{P}\left(\exists(i,j) \in \mathcal{A}\ :\ N_{i,j} \neq 0\right) + 2^{-n}\big)$$

where we define $\mathbf{x} \sim \mathcal{U}\left(\mathbb{F}_2^s \setminus \{\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{e}_{\mathscr{P}}\}\right)$ *and*

$$\begin{aligned}
\mathcal{A} &\stackrel{def}{=} \{(i,j) \in [\![0,\ n-s]\!] \times [\![0,\ s]\!]\ :\ \delta_w^{(n-s)}(i)\,\delta_{t_{\text{aux}}}^{(s)}(j) \geqslant \frac{\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)}{n^{3.2}}\} \\
T &\stackrel{def}{=} \frac{1}{2}N\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u).
\end{aligned}$$

and where $\mathcal{C}$ *is distributed as* $\mathcal{U}(n,\ k)$ *conditioned on the event that* $\dim(\mathcal{C}_{\mathscr{P}}) = s$*, and* $\mathcal{C}_{\text{aux}} \sim \mathcal{U}(s,\ k_{\text{aux}})$ *and where we used the notations of Notation 6.*

Remark 20. *We verify experimentally this conjecture in Definition 41. We believe however that the results stay true for a much smaller polynomial than the* n^8 *appearing in the condition.*

6.3.1.4.1 Proof of the linearity relations

Proof of Lemma 25. Let $\mathbf{c}^{\mathscr{N}} \in \mathcal{C}^{\mathscr{N}}$ and $\mathbf{c}^{\perp}_{\text{aux}} \in \mathcal{C}^{\perp}_{\text{aux}}$. We have that

$$\begin{aligned}\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{z}, \mathbf{m}_{\text{aux}} \rangle &= \langle \mathbf{e}, \mathbf{h} \rangle - \langle \mathbf{x}\mathbf{G}^{\intercal}_{\text{aux}}, \mathbf{m}_{\text{aux}} \rangle \\ &= \langle \mathbf{e}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} \rangle \\ &= \langle \mathbf{e}, \mathbf{h} \rangle - \left\langle \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} \right\rangle\end{aligned}$$

where in the last line we used the fact that $\left\langle \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} \right\rangle = 0$ as $\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} \in \mathcal{C}_{\text{aux}}$. We will next use the fact that, by construction, $\mathbf{h}_{\mathscr{P}} = \mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal}$ to write that

$$\begin{aligned}\langle \mathbf{e}, \mathbf{h} \rangle - \left\langle \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} \right\rangle &= \langle \mathbf{e}, \mathbf{h} \rangle - \left\langle \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{h}_{\mathscr{P}} + \mathbf{e}_{\text{aux}} \right\rangle \\ &= \left\langle \mathbf{e}_{\mathscr{P}} + \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{h}_{\mathscr{P}} \right\rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \left\langle \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{e}_{\text{aux}} \right\rangle \\ &= \left\langle \mathbf{e}_{\mathscr{P}} + \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{h}_{\mathscr{P}} \right\rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \left\langle \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{e}_{\text{aux}} \right\rangle \\ &= \left\langle \mathbf{e}_{\mathscr{P}} + \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal} \right\rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \left\langle \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{e}_{\text{aux}} \right\rangle \\ &= \left\langle \left(\mathbf{e}_{\mathscr{P}} + \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}} \right)\mathbf{R} + \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \right\rangle + \left\langle \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{e}_{\text{aux}} \right\rangle \\ &= \left\langle \left(\mathbf{e}_{\mathscr{P}} + \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}} \right)\mathbf{R} + \mathbf{e}_{\mathscr{N}} + \mathbf{c}^{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \right\rangle + \left\langle \mathbf{x} + \mathbf{c}^{\perp}_{\text{aux}}, \mathbf{e}_{\text{aux}} \right\rangle\end{aligned}$$

where in the last line we used the fact that $\left\langle \mathbf{c}^{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \right\rangle = 0$ as $\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\perp}\right)_{\mathscr{N}}$ where

$$\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}} = \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}.$$

□

6.3.2 Tail behavior of the score function, verifying the conjecture

The goal of this section is to verify heuristically Conjecture 3 which gives some exponential concentration bounds for our score function. We generalize the technique devised in RLPN to our setting here, namely we make a model on the score function $F^{\mathcal{L}}(\mathbf{z})$, experimentally verify that this model does not change the score function's distribution and then prove that our conjecture holds under this model.

6.3.2.1 Duality formula

Our main tool, as in RLPN, is a dual formula which is basically the continuity of the discussion around the conjecture made in Section 6.3.1.4 and which gives basically the individual contribution of each of the weight enumerator $N^{(\mathbf{x})}_{i,j}$ to the value of the score function $F^{\mathcal{L}}(\mathbf{x}\mathbf{G}^{\intercal}_{\text{aux}})$.

Proposition 48 (Duality formula for the score function)**.** *Using Notation 6 and supposing furthermore that $\mathcal{C}$ is of dimension k we have that for any $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ and any $\mathbf{x} \in \mathbb{F}_2^s$ such that $\mathbf{z} = \mathbf{x}\mathbf{G}^{\intercal}_{\text{aux}}$,*

$$F^{\mathcal{L}}(\mathbf{z}) = \frac{1}{2^{k-k_{\text{aux}}}} \sum_{i=0}^{n-s} \sum_{j=0}^{s} N^{(\mathbf{x})}_{i,j} K^{(n-s)}_{w}(i)\, K^{(s)}_{t_{\text{aux}}}(j). \tag{6.6}$$

Notably, the key quantity $N_{i,j}^{(\mathbf{x})}$ is directly related to more standard weight enumerators of the following random affine codes:

Lemma 26. *Let $\mathbf{x} \in \mathbb{F}_2^s$ we have that*

$$N_{i,j}^{(\mathbf{x})} = \sum_{u=0}^{N_j\left(\mathcal{C}_{\text{aux}}^{\perp}+\mathbf{x}\right)} N_i\left(\mathcal{C}^{\mathscr{N}} + \left(\mathbf{r}^{(u)} + \mathbf{e}_{\mathscr{P}}\right)\mathbf{R} + \mathbf{e}_{\mathscr{N}}\right)$$

where we recall that $N_i(\mathcal{C})$ is the standard weight enumerator of the code $\mathcal{C}$, namely

$$N_i(\mathcal{C}) = \left|\mathcal{C} \bigcap \mathcal{S}_i^n\right|$$

and where $\mathbf{r}^{(u)}$ denotes the u'th codeword of $\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x} \bigcap \mathcal{S}_j^s$.

As in the case of RLPN, here also, we reduce proving our conjecture on the tail distribution of $F^{\mathcal{L}}(\mathbf{z})$ to proving exponential tail bounds on the weight enumerator of the coset codes considered in the previous lemma, namely bounds of the type

$$\mathbb{P}\left(N_i(\mathcal{C}) - \mathbb{E}\left(N_i(\mathcal{C})\right) \geqslant \text{poly}(n)\sqrt{\mathbf{Var}\left(N_i(\mathcal{C})\right)}\right) = 2^{-\Omega(n)}$$

which we could not find in the literature.

6.3.2.2 The Poisson model

To contravene this we will make the model that these weight enumerators are Poisson variables with right expected value, namely we could show that for any fixed $u \in [\![1,\, N_j\left(\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x}\right)]\!]$ we have that

$$\mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x}\right)\right) \leqslant \frac{\binom{s}{j}}{2^{k_{\text{aux}}}}\left(1 + \mathcal{O}\left(2^{-k_{\text{aux}}}\right)\right)$$
$$\mathbb{E}\left(N_i\left(\mathcal{C}^{\mathscr{N}} + \left(\mathbf{r}^{(u)} + \mathbf{e}_{\mathscr{P}}\right)\mathbf{R} + \mathbf{e}_{\mathscr{N}}\right)\right) = \frac{\binom{n-s}{i}}{2^{n-k}}.$$

We model $N_j\left(\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x}\right)$ by a Poisson variable of expected value $\frac{\binom{s}{j}}{2^{k_{\text{aux}}}}$ and, for each u we model $N_i\left(\mathcal{C}^{\mathscr{N}} + \left(\mathbf{r}^{(u)} + \mathbf{e}_{\mathscr{P}}\right)\mathbf{R} + \mathbf{e}_{\mathscr{N}}\right)$ by a Poisson variable of expected value $\frac{\binom{n-s}{i}}{2^{n-k}}$. Additionally, it will be convenient and simpler for our proof to ask that the variables $N_i\left(\mathcal{C}^{\mathscr{N}} + \left(\mathbf{r}^{(u)} + \mathbf{e}_{\mathscr{P}}\right)\mathbf{R} + \mathbf{e}_{\mathscr{N}}\right)$ given by u are independent. All in all, using the fact that the sum of Poisson variable is a Poisson variable of good expected value, we get the following model for $N_{i,j}$.

Model 5 (Poisson Model)**.** *Using the distributions in Conjecture 3 we make the model that*

$$N_{i,j}^{(\mathbf{x})} \sim \text{Poisson}\left(N_j\,\frac{\binom{n-s}{i}}{2^{n-k}}\right),\ \textit{where}\ N_j^{(\mathbf{x})} \sim \text{Poisson}\left(\frac{\binom{s}{j}}{2^{k_{\text{aux}}}}\right).$$

Now, under Poisson Model Model 5, the following proposition proves Conjecture 3 while we can show experimentally that it keeps the distribution of the score function unchanged, see Fig. 6.2.

Proposition 49. *Under Model 5, Conjecture 3 holds.*

The proof is made in Section 6.5.2.

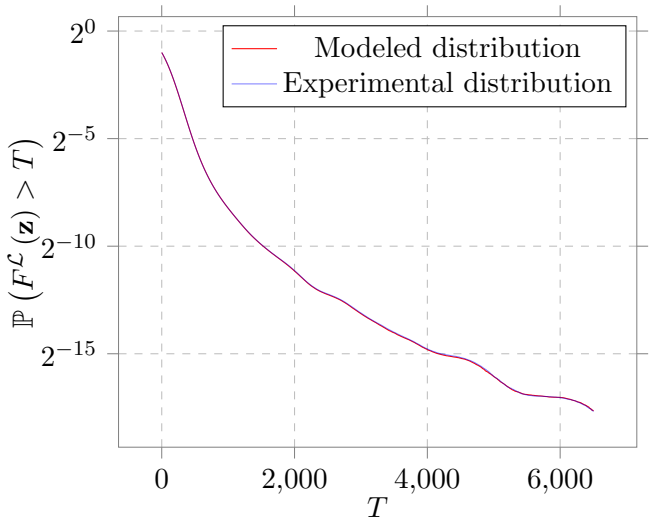


Figure 6.2: Experimental validity of the Poisson model (Model 5). We plot the distribution of the score function $F^{\mathcal{L}}(\mathbf{z})$ of double-RLPN that we obtained experimentally against the modeled distribution of $F^{\mathcal{L}}(\mathbf{z})$ under the Poisson model Model 5. Both curve are plotted using Monte-Carlo method (see Fig. 5.4 for more details). Here $n = 60$, $k = 30$, $s = 28$, $k_{\text{aux}} = 20$, $t_{\text{aux}} = 2$, $w = 5$.

6.3.2.3 Proof of the duality formula

Here we prove Proposition 48. Let us recall that the score function can be rewritten as follows by using the linearity relation between the dual vectors.

Fact 25. *Let $\mathcal{C}$ We have for $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ and any $\mathbf{x} \in \mathbb{F}_2^s$ such that $\mathbf{z} = \mathbf{x}\mathbf{G}_{\text{aux}}^{\intercal}$ that*

$$F^{\mathcal{L}}(\mathbf{z}) = \sum_{\substack{\mathbf{h}_{\mathscr{N}} \in (\mathcal{C}^{\mathscr{N}})^{\perp} \\ |\mathbf{h}_{\mathscr{N}}| = w}} \sum_{\substack{\mathbf{e}_{\text{aux}} \in \mathcal{S}_{t_{\text{aux}}}^s \\ \mathbf{h}_{\mathscr{P}}\mathbf{R}^{\intercal} + \mathbf{e}_{\text{aux}} \in \mathcal{C}_{\text{aux}}}} (-1)^{\langle (\mathbf{x}+\mathbf{e}_{\mathscr{P}})\mathbf{R} + \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{x}, \mathbf{e}_{\text{aux}} \rangle}$$

Proof of Proposition 48. Starting from Fact 25 we get that

$$F^{\mathcal{L}}(\mathbf{z}) = \sum_{\substack{(\mathbf{h}_{\mathscr{N}}, \mathbf{c}_{\text{aux}}) \in (\mathcal{C}^{\perp})_{\mathscr{N}} \times \mathcal{C}_{\text{aux}} \\ |\mathbf{h}_{\mathscr{N}}| = w, |\mathbf{R}\mathbf{h}_{\mathscr{N}} + \mathbf{c}_{\text{aux}}| = t_{\text{aux}}}} (-1)^{\langle (\mathbf{x}+\mathbf{e}_{\mathscr{P}})\mathbf{R} + \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{x}, \mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal} + \mathbf{c}_{\text{aux}} \rangle}$$

Therefore,

$$F^{\mathcal{L}}(\mathbf{z}) = \sum_{(\mathbf{h}_{\mathscr{N}}, \mathbf{c}_{\text{aux}}) \in (\mathcal{C}^{\perp})_{\mathscr{N}} \times \mathcal{C}_{\text{aux}}} f(\mathbf{h}_{\mathscr{N}}, \mathbf{c}_{\text{aux}}) \tag{6.7}$$

where

$$f(\mathbf{h}_{\mathscr{N}}, \mathbf{c}_{\text{aux}}) \stackrel{\text{def}}{=} (-1)^{\langle (\mathbf{x}+\mathbf{e}_{\mathscr{P}})\mathbf{R} + \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{x}, \mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal} + \mathbf{c}_{\text{aux}} \rangle} \mathbf{1}_{\{|\mathbf{h}_{\mathscr{N}}| = w, |\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal} + \mathbf{c}_{\text{aux}}| = t_{\text{aux}}\}}.$$

Using Lemma 1 we have that $\left(\mathcal{C}^{\perp}\right)_{\mathscr{N}} = \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}$ and thus $\left(\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp} \times \mathcal{C}_{\text{aux}}\right)^{\perp} = \mathcal{C}^{\mathscr{N}} \times \mathcal{C}_{\text{aux}}^{\perp}$. By using the Poisson formula Proposition 17 together with the fact that $\dim\left(\mathcal{C}^{\mathscr{N}} \times \mathcal{C}_{\text{aux}}^{\perp}\right) = \dim\left(\mathcal{C}^{\mathscr{N}}\right) + \dim\left(\mathcal{C}_{\text{aux}}^{\perp}\right) = k - k_{\text{aux}}$, we get

$$\sum_{(\mathbf{h}_{\mathscr{N}}, \mathbf{c}_{\text{aux}}) \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp} \times \mathcal{C}_{\text{aux}}} f(\mathbf{h}_{\mathscr{N}}, \mathbf{c}_{\text{aux}}) = \frac{|\mathcal{C}_{\text{aux}}|}{|\mathcal{C}|} \sum_{(\mathbf{c}^{\mathscr{N}}, \mathbf{c}_{\text{aux}}^{\perp}) \in \mathcal{C}^{\mathscr{N}} \times \mathcal{C}_{\text{aux}}^{\perp}} \widehat{f}(\mathbf{c}^{\mathscr{N}}, \mathbf{c}_{\text{aux}}^{\perp}). \tag{6.8}$$

Let us compute the right-hand term. By definition of f, it is readily seen that

$$\begin{aligned}
\widehat{f}(\mathbf{y}_1, \mathbf{y}_2) &= \sum_{\substack{\mathbf{z}_1 \in \mathbb{F}_2^{n-1}, \mathbf{z}_2 \in \mathbb{F}_2^s \\ |\mathbf{z}_1| = w, |\mathbf{z}_1 \mathbf{R}^{\top} + \mathbf{z}_2| = t_{\text{aux}}}} (-1)^{\langle \mathbf{y}_1, \mathbf{z}_1 \rangle + \langle \mathbf{y}_2, \mathbf{z}_2 \rangle} (-1)^{\langle (\mathbf{x} + \mathbf{e}_{\mathscr{P}})\mathbf{R} + \mathbf{e}_{\mathscr{N}}, \mathbf{z}_1 \rangle + \langle \mathbf{x}, \mathbf{z}_1 \mathbf{R}^{\intercal} + \mathbf{z}_2 \rangle} \\
&= \sum_{\mathbf{z}_1 \in \mathbb{F}_2^{n-s} : |\mathbf{z}_1| = w} (-1)^{\langle \mathbf{y}_1 + (\mathbf{x} + \mathbf{e}_{\mathscr{P}} + \mathbf{y}_2)\mathbf{R} + \mathbf{e}_{\mathscr{N}}, \mathbf{z}_1 \rangle} \sum_{\mathbf{z}_2 \in \mathbb{F}_2^s : |\mathbf{z}_1 \mathbf{R}^{\intercal} + \mathbf{z}_2| = t_{\text{aux}}} (-1)^{\langle \mathbf{y}_2 + \mathbf{x}, \mathbf{z}_1 \mathbf{R}^{\intercal} + \mathbf{z}_2 \rangle} \\
&= K_w^{(n-s)}\left(|\mathbf{y}_1 + (\mathbf{y}_2 + \mathbf{x} + \mathbf{e}_{\mathscr{P}})\mathbf{R} + \mathbf{e}_{\mathscr{N}}|\right) \, K_{t_{\text{aux}}}^{(s)}\left(|\mathbf{y}_2 + \mathbf{x}|\right).
\end{aligned}$$

Plugging this into Equation (6.8) and then into Equation (6.7) concludes the proof. □

6.4 Main theorem and results

6.4.1 Main theorem

We are now ready to state the main theorem of this chapter giving the performance of double-RLPN instantiated with juxtaposition code with a constant number of blocks and when furthermore all the dual vectors of weight w on $\mathscr{N}$ are computed.

Theorem 8. *There exists a positive poly-bounded function f such that for any $k, t, s, b, k_{\text{aux}}, t_{\text{aux}}, w, u, N_{\text{iter}}, T \in \mathbb{N}$ implicit functions of a parameter $n \in \mathbb{N}$ (n is growing to infinity) and any procedure* Compute-Dual-Vectors *that are such that*

1. *(Computing all the dual vectors)*

$$\mathbb{P}_{\mathcal{D} \sim \mathcal{U}_{\mathbf{G}}(n-s,\, k-s)}\left(\text{Compute-Dual-Vectors}(\mathcal{D}) = \mathcal{D}^{\perp} \bigcap \mathcal{S}_w^{n-s}\right) \in 1 - o(1),$$

2. *(Linear scaling and constant number of blocks)* $b = \left\lceil \frac{s}{k_{\text{aux}}} \right\rceil$ *and* $b \in \mathcal{O}(1)$,

3. *(Main constraint that we have enough dual vectors)*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \Omega\left(\frac{f(n^b)}{\left(\delta_w^{(n-s)}(u)\, \delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}\right),$$

4. *(Decoding the auxiliary code below Gilbert-Varshamov)* $\binom{s}{t_{\text{aux}}}/2^{s-k_{\text{aux}}} \in \mathcal{O}(1)$,

5. *(Outside Krawtchouk root region)* $u < \text{Root}\left(K_w^{(n-s)}\right)$ *and for all* $i \in [\![1, b]\!]$, $(t-u)^{(i)} < \text{Root}\left(K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\right)$,

6. *(Auxiliary quantities and minor constraints)* $N_{\text{iter}} \in \Omega\left(f(n^b)\frac{\binom{n}{t}}{\binom{n-s}{u}\binom{s}{t-u}}\right)$ *and* $T = \frac{1}{2}\delta_w^{(n-s)}(u)\prod_{i=1}^{b}\delta_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left((t-u)^{(i)}\right)\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}}$ *and* $k-s \in \omega(1)$

then, under Conjecture 3 (and its generalization to the case $b \in \mathcal{O}(1)$*, namely Conjecture 5), there exists an algorithm (Algorithm 3 with the right stoping conditions) that solves* $\mathrm{DP}_{\mathbf{G}}(n,k,t)$ *with probability* $1-o(1)$ *in time and memory*

$$\begin{aligned}
\textbf{\textit{Time}} &= \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{\binom{n-s}{u}\binom{s}{t-u}}\left(2^{k_{\text{aux}}} + T_{\text{eq}} + C\,T_{\text{subdec}}\right)\right),\\
\textbf{\textit{Memory}} &= \widetilde{\mathcal{O}}\left(\left(2^{k_{\text{aux}}} + M_{\text{eq}} + M_{\text{subdec}}\right)\right),
\end{aligned}$$

where the upper bound for the number of false candidates is given by

$$\begin{aligned}
C &\overset{def}{=} \left[\max\left(\frac{\binom{s}{t-u}}{2^{k_{\text{aux}}}},\,1\right) + \binom{s}{t-u}\max_{(i,j)\in\mathcal{A}}\left(\frac{\binom{n-s}{i}\binom{s}{j}}{2^{n-k+k_{\text{aux}}}}\right)\right]\max\left(1,\,\frac{2^{k_{\text{aux}}}}{\binom{s}{t-u}}\right)\\
\mathcal{A} &\overset{def}{=} \{(i,j)\in[\![0,\,n-s]\!]\times[\![0,\,s]\!] \,:\, \delta_w^{(n-s)}(i)\,\delta_{t_{\text{aux}}}^{(s)}(j) \geqslant \frac{\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)}{n^{3.2}}\}
\end{aligned}$$

and where T_{eq}, M_{eq} *are the expected time and memory complexity of* Compute-Dual-Vectors$(\mathcal{D},w)$ *when* $\mathcal{D}\sim\mathcal{U}_{\mathbf{G}}(n-s,\,k-s)$ *and* T_{subdec}, M_{subdec} *are the time and memory complexity of* ISD*Dumer [Dum89] to solve* $\mathrm{DP}_{\mathbf{G}}(n-s,k-s,u)$ *with probability* $1-o(1)$*. We recall that* $\delta_w^{(n)}(t)$ *is defined in Definition 25 and* $\mathrm{Root}\left(K_w^{(n)}\right)$ *is defined in Definition 26.*

We make the full prof of this statement in Section 6.3.2 and the conjecture was verified experimentally in the previous Section 6.3.2.

6.4.2 Results

6.4.2.1 Time complexity exponent

We give the asymptotic counterpart of the previous theorem in Section 6.5.1. This allows us to obtain, after some parameter optimization the following Fig. 6.3 giving the complexity exponent of double-RLPN when decoding at the Gilbert-Varshamov distance. We give more details on how those curves were obtained in Section 6.5.1. Note that we were careful to optimize our complexity formula in a simplified setting first by i) overlooking the cost of checking the false candidates and ii) keeping only the main constraint on the parameters that we have enough dual vectors. Then, from those optimal parameters, we checked that the cost of dealing with the candidates did not dominate the complexity and that the additional constraints were met by our parameters. This allows us to justify that our algorithm performs essentially as well as if the LPN model was valid. Second, after optimization it turns out that all our optimal parameters end-up such that $t_{\text{aux}} = d_{\text{GV}}(s,\,k_{\text{aux}})$.

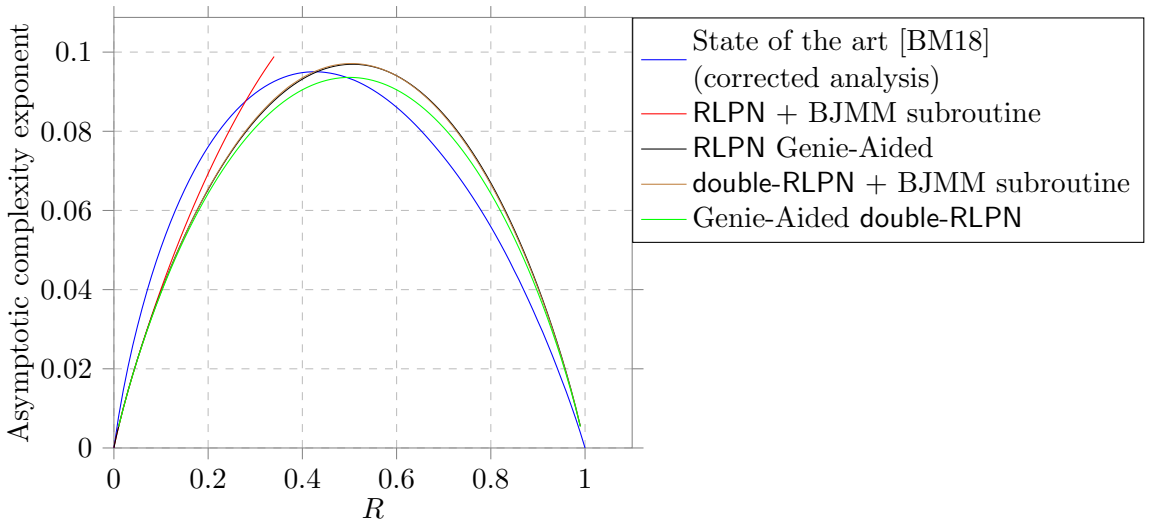


Figure 6.3: Asymptotic complexity exponent, relative to the codelength n of some generic decoders when decoding codes of rate R at the Gilbert-Varshamov distance. The double-RLPN Genie-Aided variant is an idealized variant obtained by supposing that each dual vector of weight w on $\mathscr{N}$ can be obtained in time poly (n). The state of the art of ISD is given by [BM18] with our corrected analysis.

6.4.2.2 Memory complexity exponent

We have optimized the time complexity regardless of the memory, thus, our optimized parameters are not really representative of the memory complexity of our algorithm. Nevertheless, the memory complexity of double-RLPN seems significantly higher than recent ISD's such that [BJMM12, BM18] for small to mid-range rates but seems to be comparable, and possibly better than the aforementioned ISD's for high rates.

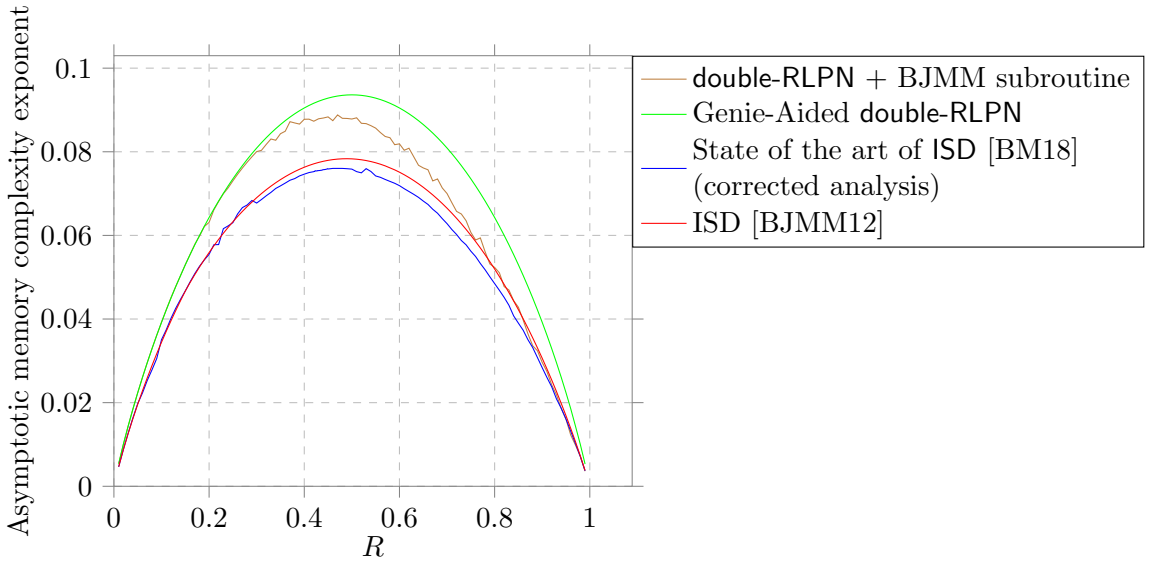


Figure 6.4: Asymptotic memory complexity exponent, relative to the codelength n of some generic decoders when decoding codes of rate R at the Gilbert-Varshamov distance. The asymptotic exponents is obtained by using the same parameters that minimized the time complexity exponent.

6.5 Appendices

6.5.1 Details about the asymptotic results and optimization

Let us define the asymptotic complexity exponent of double-RLPN.

Remark 21. *In the following definition the relative counterpart of the parameters can be seen as* $R = k/n$, $\tau = t/n$, $\sigma = s/n$, $\omega = w/n$, $\mu = u/n$, $R_{\text{aux}} = k_{\text{aux}}/n$, $\tau_{\text{aux}} = t_{\text{aux}}/n$.

Definition 42 (Asymptotic complexity exponent of double-RLPN)**.** *Let* α_{eq} *and* β_{eq} *be two bivariate functions (the time and memory complexity exponents of the procedure computing the dual vectors). Let* $R \in]0,\ 1[$ *and* $\tau \in]0,\ 1/2[$ *and let*

$$\text{Param} \stackrel{def}{=} [\sigma, \omega, \mu, R_{\text{aux}}, \tau_{\text{aux}}]$$

be list of non-negative reals such that

1. *(Main constraint that we have enough dual vectors)*

$$\sigma\, h\left(\frac{\tau_{\text{aux}}}{\sigma}\right) + (1-\sigma)h\left(\frac{\omega}{1-\sigma}\right) - (R - R_{\text{aux}}) \geqslant -2\left(\sigma\kappa\left(\frac{\tau_{\text{aux}}}{\sigma},\ \frac{\tau-\mu}{\sigma}\right) + (1-\sigma)\kappa\left(\frac{\omega}{1-\sigma},\ \frac{\mu}{1-\sigma}\right)\right)$$

2. *(Decoding the Auxiliary code below GV)*

$$\sigma h\left(\frac{R_{\text{aux}}}{\sigma}\right) - (\sigma - R_{\text{aux}}) \leqslant 0$$

3. (Outside Krawtchouk root region)

$$\mu < \frac{1-\sigma}{2} - \sqrt{\omega(1-\omega)}, \qquad \tau - \mu < \frac{\sigma}{2} - \sqrt{t_{\text{aux}}(1-t_{\text{aux}})}$$

4. (Valid domain)

$$\sigma \leqslant R, \quad \tau - \sigma \leqslant \mu \leqslant \tau, \quad \omega \leqslant 1-\sigma,\ R_{\text{aux}} \leqslant \sigma,\ \tau_{\text{aux}} \leqslant \sigma$$

we define the asymptotic time complexity exponent of double-RLPN *as*

$$\alpha_{\text{double-RLPN}}\ (R, \tau : \text{Param}; \alpha_{\text{eq}}) \stackrel{def}{=} -\pi + \max\Bigg((1-\sigma)\, \alpha_{\text{eq}} \left(\frac{R-\sigma}{1-\sigma}, \frac{\omega}{1-\sigma} \right),\ R_{\text{aux}},\ \nu_{\text{candidate}} + (1-\sigma)\alpha_{\text{ISD-Dumer}} \frac{R-\sigma}{1-\sigma} \frac{\mu}{1-\sigma} \Bigg),$$

and we define the asymptotic memory complexity exponent of double-RLPN *as*

$$\beta_{\text{double-RLPN}}\ (R, \tau : \text{Param}; \beta_{\text{eq}}) \stackrel{def}{=} \max\left((1-\sigma)\ \cdot\ \beta_{\text{eq}} \left(\frac{R-\sigma}{1-\sigma}, \frac{\omega}{1-\sigma} \right),\ R_{\text{aux}},\ (1-\sigma)\beta_{\text{ISD-Dumer}} \frac{R-\sigma}{1-\sigma} \frac{\mu}{1-\sigma} \right)$$

where

$$\nu_{\text{candidates}} \stackrel{def}{=} \max\Bigg(0, \sigma h \left(\frac{\tau-\mu}{\sigma} \right) - R_{\text{aux}},\ \max_{(\eta,\zeta)\in\mathcal{A}} \sigma\ \cdot\ h_2 \left(\frac{\zeta}{\sigma} \right) + (1-\sigma)\ \cdot\ h_2 \left(\frac{\eta}{1-\sigma} \right) - (1-R) \Bigg) + \max\left(R_{\text{aux}} - \sigma h \left(\frac{\tau-\mu}{\sigma} \right),\ 0 \right)$$

$$\pi \stackrel{def}{=} h(\tau) - \sigma\ \cdot\ h_2 \left(\frac{\tau-\mu}{\sigma} \right) - (1-\sigma)\ \cdot\ h \left(\frac{\mu}{1-\sigma} \right)$$

with

$$\mathcal{A} \stackrel{def}{=} \Bigg\{ (\eta,\zeta) \in [0, 1-\sigma] \times [0, \sigma]\ :\ \sigma \left[\kappa \left(\frac{t_{\text{aux}}}{\sigma},\ \frac{\tau-\mu}{\sigma} \right) - \kappa \left(\frac{t_{\text{aux}}}{\sigma},\ \frac{\zeta}{\sigma} \right) \right) + (1-\sigma) \left[\kappa \left(\frac{\omega}{1-\sigma},\ \frac{\mu}{1-\sigma} \right) - \kappa \left(\frac{\omega}{1-\sigma},\ \frac{\eta}{1-\sigma} \right) \right] \leqslant 0 \Bigg\}$$

where we recall that $\kappa(,\ )$ is defined in Corollary 5 and where $\alpha_{\text{ISD-Dumer}}$ and $\beta_{\text{ISD-Dumer}}$ are the time and memory complexity exponent of [Dum89] defined in Definition 43.

Proposition 50 (Asymptotic performance of double-RLPN)**.** *Let $n \in \mathbb{N}$ be growing to infinity. For any constants $R, \tau, \sigma, \omega, \mu, R_{\text{aux}}, \tau_{\text{aux}}$ that verify the constraints of Definition 35 and defining*

$$\text{Param} \stackrel{def}{=} [\sigma, \omega, \mu, R_{\text{aux}}, \tau_{\text{aux}}]$$

and under Conjecture 3 (and its generalization to the case $b \in \mathcal{O}(1)$, namely Conjecture 5) we have that

1. *(*double-RLPN *+ Dumer subroutine) there exists an algorithm that solves* $\mathrm{DP}_{\mathbf{G}}(n, \lfloor Rn \rfloor, \lfloor \tau n \rfloor)$ *in time and memory respectively*

$$\boldsymbol{T} = \widetilde{\mathcal{O}}\left(2^{n\,\alpha_{\text{double-RLPN}(R,\tau\,;\text{Param};\alpha_{dual-Dumer-routine})}}\right),\ \boldsymbol{M} = \widetilde{\mathcal{O}}\left(2^{n\,\beta_{\text{double-RLPN}(R,\tau\,;\text{Param};\beta_{dual-Dumer-routine})}}\right)$$

where we recall that $\alpha_{dual-Dumer-routine}$ *and* $\beta_{dual-Dumer-routine}$ *are bivariate functions defined in Proposition 13.*

2. *(*double-RLPN *+ BJMM subroutine) under the heuristic that the procedure given in Proposition 14 outputs all the dual vectors of weight* w *with probability* $1-o(1)$ *then there exists an algorithm that solves* $\mathrm{DP}_{\mathbf{G}}(n, \lfloor Rn \rfloor, \lfloor \tau n \rfloor)$ *in time and memory respectively*

$$\boldsymbol{T} = \widetilde{\mathcal{O}}\left(2^{n\,\alpha_{\text{double-RLPN}(R,\tau\,;\text{Param};\alpha_{\text{dual-BJMM-routine}})}}\right), \quad \boldsymbol{M} = \widetilde{\mathcal{O}}\left(2^{n\,\beta_{\text{double-RLPN}(R,\tau\,;\text{Param};\beta_{\text{dual-BJMM-routine}})}}\right)$$

where we recall that $\alpha_{\text{dual-BJMM-routine}}$ *and* $\beta_{\text{dual-BJMM-routine}}$ *are bivariate functions defined in Proposition 14.*

The curves related to double-RLPN in Fig. 6.3 which expose the complexity exponent of the algorithm were obtained by optimizing the parameters to minimize the complexity of 1. and 2. as given by Proposition 50.

Asymptotic complexity exponent of ISD Dumer Lastly, we give now, for completeness the asymptotic complexity exponent of [Dum89] that appears in Definition 42. This algorithm is exactly using Dumer collision decoder Algorithm 3 inside the framework given by Algorithm 2 as described in our chapter on ISD's.

Definition 43 (Complexity exponent of the ISD[Dum89])**.** *Let* $R \in]0,\ 1[$ *and* $\tau \in]0,\ 1/2[$ *we define*

$$\alpha_{\text{ISD-Dumer}}(R,\ \tau) \stackrel{def}{=} \min_{\omega,\lambda}\left(\pi + \max\left(\frac{R+\lambda}{2} h_2\left(\frac{\omega}{R+\lambda}\right), (R+\lambda)\, h_2\left(\frac{\omega}{R+\lambda}\right) - \lambda\right)\right), \tag{6.9}$$

$$\pi \stackrel{def}{=} h_2(\tau) - (1-R-\lambda) h_2\left(\frac{\tau-\omega}{1-R-\lambda}\right) - (R+\lambda) h_2\left(\frac{\omega}{R+\lambda}\right), \tag{6.10}$$

$$\nu_{\text{sol}} \stackrel{def}{=} \max\left(h_2(\tau) - (1-R), 0\right), \tag{6.11}$$

$$\beta_{\text{ISD-Dumer}}(R,\ \tau) \stackrel{def}{=} \frac{R+\lambda}{2} h_2\left(\frac{\omega}{R+\lambda}\right) \tag{6.12}$$

where λ *and* ω *must verify the following constraints:*

$$0 \leqslant \lambda \leqslant 1-R, \qquad \max\left(R+\lambda+\tau-1, 0\right) \leqslant \omega \leqslant \min\left(\tau, R+\lambda\right).$$

Proposition 51. *For any constants* $R \in]0,\ 1[$ *and* $\tau \in]0,\ 1/2[$ *there exists an algorithm ([Dum89]) that solves* $\mathrm{DP}_{\mathbf{G}}(n, \lfloor Rn \rfloor, \lfloor \tau n \rfloor)$ *with probability* $1-o(1)$ *in time* $\widetilde{\mathcal{O}}\left(2^{n\alpha_{\text{ISD-Dumer}}(R,\ \tau)}\right)$ *and memory* $\widetilde{\mathcal{O}}\left(2^{n\beta_{\text{ISD-Dumer}}(R,\ \tau)}\right)$.

6.5.1.1 Optimization

In Fig. 6.3, the curves related to double-RLPN where obtained by fixing the rate R and the relative decoding distance τ at $\tau_{\mathrm{GV}}(R)$ and minimizing (over $\sigma,\ \omega,\ \mu,\ R_{\mathrm{aux}}, \tau_{\mathrm{aux}}$) the "simplified time complexity" of the algorithm, namely

$$-\pi + \max\left((1-\sigma)\,\alpha_{\mathrm{eq}}\left(\frac{R-\sigma}{1-\sigma}, \frac{\omega}{1-\sigma}\right),\ R_{\mathrm{aux}}\right)$$

under the main constraint that we have enough dual vectors, namely:

$$\sigma\, h\left(\frac{\tau_{\mathrm{aux}}}{\sigma}\right) + (1-\sigma) h\left(\frac{\omega}{1-\sigma}\right) - (R - R_{\mathrm{aux}}) \geqslant -2\left(\sigma\kappa\left(\frac{\tau_{\mathrm{aux}}}{\sigma},\ \frac{\tau-\mu}{\sigma}\right) + (1-\sigma)\kappa\left(\frac{\omega}{1-\sigma},\ \frac{\mu}{1-\sigma}\right)\right)$$

and then checking that the obtained optimal parameters satisfy the other minor constraints and that the cost of the last recovering step does not dominate compared to, say, the cost of the FFT.

6.5.2 Proof that the Poisson model imply the conjecture

In this section we prove Proposition 49. Let us rewrite here slightly what we have to prove.

Using our duality formula given in Proposition 48 it is readily seen that we can simply replace the score function $F^{\mathcal{L}}$ appearing in the conjecture by its dual counterpart.

Conjecture 4 (Rewritting of Conjecture 3)**.** *For any $t, k, s, w, u, k_{\mathrm{aux}}, t_{\mathrm{aux}} \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that*

$$\frac{\binom{s}{t_{\mathrm{aux}}}}{2^{s-k_{\mathrm{aux}}}} \in \mathcal{O}(1) \quad \text{and} \quad \frac{\binom{n-s}{w}\binom{s}{t_{\mathrm{aux}}}}{2^{k-k_{\mathrm{aux}}}} = \frac{\omega\left(n^8\right)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\mathrm{aux}}}^{(s)}(t-u)\right)^2}$$

then we conjecture that

$$\mathbb{P}\left(\sum_{j=0}^{s}\sum_{i=0}^{n-s} K_{t_{\mathrm{aux}}}^{(s)}(j)\, K_w^{(n-s)}(i)\, N_{i,j} \geqslant \frac{1}{2} K_w^{(n-s)}(u)\, K_{t_{\mathrm{aux}}}^{(s)}(t-u)\right) = \widetilde{\mathcal{O}}\left(\mathbb{P}\left(\exists (i,j) \in \mathcal{A} \,:\, N_{i,j} \neq 0\right) + 2^{-n}\right)$$

where

$$N_{j,i} \stackrel{def}{=} \{(\mathbf{r}, \mathbf{z}) \in \left(\mathcal{C}_{\mathrm{aux}}^{\perp} + \mathbf{x}\right) \times \left(\mathcal{C}^{\mathscr{N}} + (\mathbf{r} + \mathbf{e}_{\mathscr{P}})\,\mathbf{R} + \mathbf{e}_{\mathscr{N}}\right) \,:\, |\mathbf{r}| = j,\ |\mathbf{z}| = i\}$$

$$\mathbf{R} \stackrel{def}{=} \mathrm{Lift}\left(\mathcal{C}^{\perp},\ \mathscr{N}\right)$$

$$\mathcal{A} \stackrel{def}{=} \{(i,j) \in [\![0,\ n-s]\!] \times [\![0,\ s]\!] \,:\, \delta_w^{(n-s)}(i)\,\delta_{t_{\mathrm{aux}}}^{(s)}(j) \geqslant \frac{\delta_w^{(n-s)}(u)\,\delta_{t_{\mathrm{aux}}}^{(s)}(t-u)}{n^{3.2}}\}$$

and where $\mathscr{P}$ and $\mathscr{N}$ are any fixed complementary subsets of $[\![1,\ n]\!]$ of size s and $n-s$ respectively and $\mathcal{C}$ is distributed as $\mathcal{U}(n,\ k)$ conditionned on the event that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and $\mathcal{C}_{\mathrm{aux}} \sim \mathcal{U}(s,\ k_{\mathrm{aux}})$ and $\mathbf{e} \sim \mathcal{U}(\mathcal{S}_t^n)$ and $\mathbf{x}$ is taken uniformly at random in $\mathbb{F}_2^s \setminus \{\mathcal{C}_{\mathrm{aux}}^{\perp} + \mathbf{e}_{\mathscr{P}}\}$.

Fact 26. *Conjecture 4 is true if and only if Conjecture 3 is true.*

Proof. The proof is straightforward applying the duality formula given in Proposition 48 to write the score function appearing in the left-hand side of Conjecture 3 and simplifying the expressions. □

As such to prove our main Proposition 49 we only have to prove that the Poisson Model 5 imply Conjecture 4. As for RLPN to prove this proposition we will reduce proving conjecture Conjecture 4 to proving exponential concentration bound on the weight enumerator of random linear codes, these concentrations bounds are derived using the Poisson model. As for RLPN this reduction is done with a key centering trick that we adapt to our case.

Lemma 27 (Centering Lemma.)**.** *We have,*

$$\sum_{j=0}^{s}\sum_{i=0}^{n-s} K_{t_{\text{aux}}}^{(s)}(j)\, K_w^{(n-s)}(i)\, N_{i,j} = \sum_{j=0}^{s}\sum_{i=0}^{n-s} K_{t_{\text{aux}}}^{(s)}(j)\, K_w^{(n-s)}(i)\left(N_{i,j} - V_j\overline{N_i}\right)$$

where

$$V_j \stackrel{def}{=} \left(\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x}\right) \bigcap \mathcal{S}_j^s, \qquad \overline{N_i} \stackrel{def}{=} \frac{\binom{n-s}{i}}{2^{n-k}}.$$

Proof. From the orthogonality of Krawtchouk polynomials Proposition 19 we have that

$$0 = \sum_{i=0}^{n-s} K_w^{(n-s)}(i)\,\overline{N_i} = \sum_{j=0}^{s} V_j\, K_{t_{\text{aux}}}^{(s)}(j) \sum_{i=0}^{n-s} K_w^{(n-s)}(i)\,\overline{N_i} = \sum_{j=0}^{s}\sum_{i=0}^{n-s} K_w^{(n-s)}(i)\, K_{t_{\text{aux}}}^{(s)}(j)\, V_j\,\overline{N_i}.$$

We have our result by substracting this last term to the equation of the lemma. □

With this centering lemma we can rewrite the left-hand side of the probability appearing in Conjecture 4 by bounding the sum by the absolute value of its component as follows.

Corollary 10. *We have,*

$$\mathbb{P}\left(\sum_{j=0}^{s}\sum_{i=0}^{n-s} K_{t_{\text{aux}}}^{(s)}(j)\, K_w^{(n-s)}(i)\, N_{i,j} \geqslant \frac{1}{2} K_w^{(n-s)}(u)\, K_{t_{\text{aux}}}^{(s)}(t-u)\right) \leqslant$$
$$\mathbb{P}\left(\exists (i,j) \in [\![0,\, n-s]\!] \times [\![0,\, s]\!] \,:\, \left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j}\right)$$

where

$$R_{i,j} \stackrel{def}{=} \frac{1}{2\,(n+1)^2}\left|\frac{K_w^{(n-s)}(u)\, K_{t_{\text{aux}}}^{(s)}(t-u)}{K_w^{(n-s)}(i)\, K_{t_{\text{aux}}}^{(s)}(j)}\right|.$$

Proof. The proof is straightforward using the centering lemma and the fact that for the sum to be big at least one of the term must have sufficiently big absolute value and concluding with a union bound. □

We can now state our technical intermediate proposition that basically reduce proving the conjecture to proving concentration bounds on the weight enumerator of random linear code.

Proposition 52. *(Technical proposition.) If*

$$\frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} \in \mathcal{O}(1) \quad \text{and} \quad \frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \frac{\omega\left(n^8\right)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}$$

then for any $(i,j) \in \mathcal{A}$ we have that

$$\mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j}\right) \leqslant \mathbb{P}\left(N_{i,j} \neq 0\right) \tag{6.13}$$

and for any $(i,j) \notin \mathcal{A}$ we have that

$$\mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j}\right) = \mathcal{O}\left(\max_{v \in [\![0,\,M]\!]} \mathbb{P}\left(\left|N_{i,j} - v\overline{N_i}\right| \geqslant n^{1.1}\max\left(\sqrt{v\overline{N_i}},\,1\right) \,\middle|\, V_j = v\right) + \mathbb{P}\left(V_j \geqslant M\right)\right) \tag{6.14}$$

where

$$\overline{V_j} \stackrel{def}{=} \frac{\binom{s}{j}}{2^{k_{\text{aux}}}}, \qquad M \stackrel{def}{=} \overline{V_j} + n^{1.1}\max\left(\sqrt{\overline{V_j}}, 1\right). \tag{6.15}$$

Remark 22. *Recall that $N_{i,j} \stackrel{def}{=} \sum_{u=1}^{V_j} N_i^{(u)}$ where $N_i^{(u)} \stackrel{def}{=} N_i\left(\left(\mathbf{r}^{(u)} + \mathbf{e}_{\mathscr{P}}\right)\mathbf{R} + \mathbf{e}_{\mathscr{N}} + \mathcal{C}^{\mathscr{N}}\right)$ and $\mathbf{r}^{(u)}$ is the u'th codeword of weight j of $\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x}$. From this, we could show we that*

$$\mathbb{E}_{\mathcal{C},\mathcal{C}_{\text{aux}},\mathbf{x}}\left(N_{i,j} \middle|\, V_j = v\right) = v\,\overline{N_i}\,(1 + o(1))$$

and that $\overline{V_j}$ is a good approximation of the expected value of V_j. As such, Equation (6.14) *in the previous proposition can also really be seen as a concentration inequality.*

We prove this technical lemma just after. Note that this technical lemma directly yield our result.

Proof of Proposition 49. Recalling, from Fact 26 we only have to prove that the Poisson model imply Conjecture 4. Let us prove that. Applying successively Corollary 10 and the technical Proposition 52 we get that

$$\begin{aligned}
&\mathbb{P}\left(\sum_{j=0}^{s}\sum_{i=0}^{n-s} K_{t_{\text{aux}}}^{(s)}(j)\,K_w^{(n-s)}(i)\,N_{i,j} \geqslant \frac{1}{2}K_w^{(n-s)}(u)\,K_{t_{\text{aux}}}^{(s)}(t-u)\right) \\
&\leqslant \mathbb{P}\left(\exists (i,j) \in \mathcal{A} \,:\, \left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j}\right) + \mathbb{P}\left(\exists (i,j) \notin \mathcal{A} \,:\, \left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j}\right) \\
&\leqslant \mathbb{P}\left(\exists (i,j) \in \mathcal{A} \,:\, N_{i,j} \neq 0\right) \\
&\qquad + \sum_{(i,j)\notin\mathcal{A}} \mathcal{O}\left(\max_{v \in [\![0,\,M]\!]} \mathbb{P}\left(\left|N_{i,j} - v\overline{N_i}\right| \geqslant n^{1.1}\max\left(\sqrt{v\overline{N_i}},\,1\right) \,\middle|\, V_j = v\right) + \mathbb{P}\left(V_j \geqslant M\right)\right).
\end{aligned}$$

The first term appearing in the last inequality is exactly the first term in the right-hand side of Conjecture 4. We only have left to prove that

$$\sum_{(i,j)\notin\mathcal{A}} \mathcal{O}\left(\max_{v \in [\![0,\,M]\!]} \mathbb{P}\left(\left|N_{i,j} - v\overline{N_i}\right| \geqslant n^{1.1}\max\left(\sqrt{v\overline{N_i}},\,1\right) \,\middle|\, V_j = v\right) + \mathbb{P}\left(V_j \geqslant M\right)\right) = \widetilde{\mathcal{O}}\left(2^{-n}\right). \tag{6.16}$$

This is a consequence of the Poisson model along with Lemma 21 that we devised in the previous chapter and that states that if $\mathbf{X} \sim \text{Poisson}(\lambda)$ then

$$\mathbb{P}\left(|\mathbf{X} - \lambda| > n^{1.1} \max\left(\sqrt{\lambda}, 1\right)\right) = 2^{-\omega(n)}. \tag{6.17}$$

First, recalling using the Poisson Model 5 and because $N_{i,j}$ is a sum of V_j terms, each of expected value $\overline{N_i}$ we have that $(N_{i,j}|V_j = v) \sim \text{Poisson}\left(v\overline{N_i}\right)$ thus

$$\max_{v \in [\![0,\, M]\!]} \mathbb{P}\left(\left|N_{i,j} - v\overline{N_i}\right| \geqslant n^{1.1} \max\left(\sqrt{v\overline{N_i}},\ 1\right) \,\middle|\, V_j = v\right) = \mathcal{O}\left(2^{-n}\right). \tag{6.18}$$

Second, recalling that

$$M \stackrel{\text{def}}{=} \overline{V_j} + n^{1.1} \max\left(\sqrt{\overline{V_j}}, 1\right)$$

and that under the Poisson Model 5, $V_j \sim \text{Poisson}\left(\overline{V_j}\right)$, we have that

$$\mathbb{P}\left(V_j \geqslant M\right) = \mathbb{P}\left(V_j - \overline{V_j} \geqslant n^{1.1} \max\left(\sqrt{\overline{V_j}}\right)\right) = \mathcal{O}\left(2^{-n}\right). \tag{6.19}$$

Finally, plugging Eq. (6.19) and Eq. (6.18) into the left hand-side of Eq. (6.16) yield the right-hand side of the equality. Hence the result is proved. □

6.5.2.1 Proof of the technical proposition

We only have left to prove the technical Proposition 52. The proof of this statement will rely on the following lemma that allows us to relate $R_{i,j}$ appearing in the right-hand side of our probabilities to the expected value of the weight enumerator the related codes.

Lemma 28. *If the parameters are such that*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} = \frac{\omega\left(n^8\right)}{\left(\delta_w^{(n-s)}(u)\, \delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2} \quad \text{and} \quad \frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} = \mathcal{O}(1) \tag{6.20}$$

then we have that

$$\left(\frac{K_w^{(n-s)}(u)\, K_{t_{\text{aux}}}^{(s)}(t-u)}{K_w^{(n-s)}(i)\, K_{t_{\text{aux}}}^{(s)}(j)}\right)^2 = \omega\left(n^8\right) \overline{N_i} \max\left(\overline{V_j}, 1\right)$$

where we recall that

$$\overline{N_i} \stackrel{def}{=} \frac{\binom{n-s}{i}}{2^{n-k}}, \qquad \overline{V_j} \stackrel{def}{=} \frac{\binom{s}{j}}{2^{k_{\text{aux}}}}.$$

Proof. First recalling that $\delta_w^{(n-s)}(u) \stackrel{\text{def}}{=} \frac{K_w^{(n-s)}(u)}{\binom{n-s}{w}}$ we have, rewriting slightly the first term of Eq. (6.20) we have that

$$\left(\frac{K_w^{(n-s)}(u)\, K_{t_{\text{aux}}}^{(s)}(t-u)}{K_w^{(n-s)}(i)\, K_{t_{\text{aux}}}^{(s)}(j)}\right)^2 = \omega\left(n^8\right) \frac{2^{k-k_{\text{aux}}}\binom{s}{t_{\text{aux}}}\binom{n-s}{w}}{\left(K_w^{(n-s)}(i)\, K_{t_{\text{aux}}}^{(s)}(j)\right)^2}. \tag{6.21}$$

First ,let us lower bound $\frac{1}{\left(K_w^{(n-s)}(i)\right)^2}$. From the orthonormality relations of Krawtchouk polynomials Proposition 19 we have that

$$\sum_{v=0}^{n-s}\binom{n-s}{v}\left(K_w^{(n-s)}(v)\right)^2=\binom{n-s}{w}2^{n-s},$$

and thus, as the previous sum is composed of positive terms, we that

$$\frac{1}{\left(K_w^{(n-s)}(i)\right)^2}\geqslant\frac{\binom{n-s}{i}}{\binom{n-s}{w}2^{n-s}}. \tag{6.22}$$

Similarly for $\frac{1}{\left(K_{t_{\text{aux}}}^{(s)}(j)\right)^2}$ we get that

$$\frac{1}{\left(K_{t_{\text{aux}}}^{(s)}(j)\right)^2}\geqslant\frac{\binom{s}{j}}{\binom{s}{t_{\text{aux}}}2^{s}}.$$

Furthermore, from Definition 24 we can also deduce the following inequality

$$\left(K_{t_{\text{aux}}}^{(s)}(j)\right)^2\leqslant\binom{s}{t_{\text{aux}}}^2.$$

Combining the last two equations we get that

$$\frac{1}{\left(K_{t_{\text{aux}}}^{(s)}(j)\right)^2}\geqslant\min\left(\frac{\binom{s}{j}}{\binom{s}{t_{\text{aux}}}2^{s}},\frac{1}{\binom{s}{t_{\text{aux}}}^2}\right). \tag{6.23}$$

Finally, by using Equation (6.22) and (6.23) we get

$$\begin{aligned}\frac{2^{k-k_{\text{aux}}}\binom{s}{t_{\text{aux}}}\binom{n-s}{w}}{\left(K_w^{(n-s)}(i)\,K_{t_{\text{aux}}}^{(s)}(j)\right)^2}&\geqslant 2^{k-k_{\text{aux}}}\binom{s}{t_{\text{aux}}}\binom{n-s}{w}\frac{\binom{n-s}{i}}{\binom{n-s}{w}2^{n-s}}\ \min\left(\frac{\binom{s}{j}}{\binom{s}{t_{\text{aux}}}2^{s}},\frac{1}{\binom{s}{t_{\text{aux}}}^2}\right)\\&=\frac{\binom{n-s}{i}}{2^{n-k}}\min\left(\frac{\binom{s}{j}}{2^{k_{\text{aux}}}},\quad\frac{2^{s-k_{\text{aux}}}}{\binom{s}{t_{\text{aux}}}}\right)\\&=\frac{\binom{n-s}{i}}{2^{n-k}}\min\left(\frac{\binom{s}{j}}{2^{k_{\text{aux}}}},\Omega(1)\right)\quad\text{Eq. (6.20)}\\&=\overline{N_i}\min\left(\overline{V_j},\Omega(1)\right).\end{aligned}$$

Plugging this into Eq. (6.21) completes the proof. □

We are now ready to prove the technical statement that we recall here.

Proposition 52. *(Technical proposition.) If*

$$\frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}}\in\mathcal{O}(1)\quad\text{and}\quad\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}}\in\frac{\omega\left(n^8\right)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}$$

then for any $(i,j) \in \mathcal{A}$ *we have that*

$$\mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j}\right) \leqslant \mathbb{P}\left(N_{i,j} \neq 0\right) \tag{6.13}$$

and for any $(i,j) \notin \mathcal{A}$ *we have that*

$$\mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j}\right) = \mathcal{O}\left(\max_{v \in [\![0,\,M]\!]} \mathbb{P}\left(\left|N_{i,j} - v\overline{N_i}\right| \geqslant n^{1.1}\max\left(\sqrt{v\overline{N_i}},\,1\right) \,\middle|\, V_j = v\right) + \mathbb{P}\left(V_j \geqslant M\right)\right) \tag{6.14}$$

where

$$\overline{V_j} \overset{def}{=} \frac{\binom{s}{j}}{2^{k_{\mathrm{aux}}}}, \qquad M \overset{def}{=} \overline{V_j} + n^{1.1}\max\left(\sqrt{\overline{V_j}}, 1\right). \tag{6.15}$$

Proof of Proposition 52. We prove the previous equality for each case: $(i,j) \in \mathcal{A}$ or $(i,j) \notin \mathcal{A}$. First note that $\mathcal{A}$ can directly be re-written from its definition given in Conjecture 4 as

$$\mathcal{A} \overset{\mathrm{def}}{=} \left\{(i,j) \in [\![0, n-s]\!] \times [\![0,s]\!], \left|\frac{K_w^{(n-s)}(u)\,K_{t_{\mathrm{aux}}}^{(s)}(t-u)}{K_w^{(n-s)}(i)\,K_{t_{\mathrm{aux}}}^{(s)}(j)}\right| \leqslant n^{3.2}\right\}.$$

Case 1: Here we suppose that $(i,j) \in \mathcal{A}$. Let us prove Equation (6.13). Using the law of total probability we have that

$$\mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j}\right) \leqslant \mathbb{P}\left(N_{i,j} \neq 0\right) + \mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j},\ N_{i,j} = 0\right).$$

As $(i,j) \in \mathcal{A}$,

$$\mathbb{P}\left(N_{i,j} \neq 0\right) = \widetilde{\mathcal{O}}\left(\max_{(i^*,j^*)\in\mathcal{A}} \mathbb{P}\left(N_{i^*,j^*} \neq 0\right)\right)$$

we only have left to show that:

$$\mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j},\ \ N_{i,j} = 0\right) = \widetilde{\mathcal{O}}\left(\max_{(i^*,j^*)\in\mathcal{A}} \mathbb{P}\left(N_{i^*,j^*} \neq 0\right) + 2^{-n}\right).$$

We now show the previous equation, by proving that,

$$\mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j},\ \ N_{i,j} = 0\right) = \widetilde{\mathcal{O}}\left(2^{-n}\right). \tag{6.24}$$

We have:

$$\begin{aligned}
\mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j},\ \ N_{i,j} = 0\right) &= \mathbb{P}\left(V_j\,\overline{N_i} \geqslant R_{i,j}\right) \quad \text{(we used that } V_j, \overline{N_i} \geqslant 0) \\
&= \mathbb{P}\left(V_j \geqslant \frac{R_{i,j}}{\overline{N_i}}\right) \\
&= \mathbb{P}\left(V_j - \overline{V_j} \geqslant \frac{R_{i,j}}{\overline{N_i}} - \overline{V_j}\right) \\
&\leqslant \mathbb{P}\left(\left|V_j - \overline{V_j}\right| \geqslant \frac{R_{i,j}}{\overline{N_i}} - \overline{V_j}\right).
\end{aligned}$$

We only have to show that for n big enough we have

$$\frac{R_{i,j}}{\overline{N_i}} - \overline{V_j} \geqslant n^{1.1}\max\left(\sqrt{\overline{V_j},\,1}\right). \tag{6.25}$$

Let us prove Equation (6.25). By definition of $R_{i,j}$ in Corollary 10 and using Lemma 28 we have that

$$\begin{aligned} R_{i,j}^2 &= \frac{1}{4\,(n+1)^4}\left(\frac{K_w^{(n-s)}\,(u)\,K_{t_{\text{aux}}}^{(s)}\,(t-u)}{K_w^{(n-s)}\,(i)\,K_{t_{\text{aux}}}^{(s)}\,(j)}\right)^2 \\ &= \frac{1}{4\,(n+1)^4}\,\omega\left(n^8\right)\overline{N_i}\max\left(\overline{V_j},1\right) \\ &= f(n)\max\left(\overline{N_i}\,\overline{V_j},\overline{N_i}\right) \end{aligned} \tag{6.26}$$

where $f(n)=\omega\left(n^4\right)$. Therefore,

$$\begin{aligned} \frac{R_{i,j}^2}{\overline{N_i}^2}\,\frac{1}{n^{2.4}\max\left(\overline{V_j}^2,\,1\right)} &= \frac{f(n)\max\left(\overline{N_i}\,\overline{V_j},\,\overline{N_i}\right)}{n^{2.4}\,\overline{N_i}^2\max\left(\overline{V_j}^2,\,1\right)} \\ &= \frac{f(n)\max\left(\frac{\overline{V_j}}{\overline{N_i}},\frac{1}{\overline{N_i}}\right)}{n^{2.4}\,\max\left(\overline{V_j}^2,\,1\right)} \\ &= \begin{cases} \frac{1}{n^{2.4}}f(n)\max\left(\frac{1}{\overline{N_iV_j}},\frac{1}{\overline{N_iV_j}^2}\right) & \text{if } \overline{V_j}>1 \\ \frac{1}{n^{2.4}}f(n)\max\left(\frac{\overline{V_j}}{\overline{N_i}},\frac{1}{\overline{N_i}}\right) & \text{if } \overline{V_j}\leqslant 1 \end{cases} \\ &\geqslant \frac{1}{n^{2.4}}f(n)\min\left(\frac{1}{\overline{N_iV_j}},\frac{1}{\overline{N_i}}\right) \\ &= \frac{1}{n^{2.4}}\,\frac{f(n)}{\max\left(\overline{N_iV_j},1\,\overline{N_i}\right)} \\ &= \frac{1}{n^{2.4}}\,\frac{f(n)^2}{R_{i,j}^2} \qquad \text{(By Equation (6.26))} \\ &= \frac{\omega\left(n^{5.6}\right)}{R_{i,j}^2} \qquad \left(f(n)=\omega\left(n^{\alpha+4}\right)\right) \\ &= \omega(1) \end{aligned} \tag{6.27}$$

where in the last line we used the fact that $(i,j)\in\mathcal{A}$: by definition,

$$R_{i,j}=\frac{1}{2\,(n+1)^2}\left|\frac{K_w^{(n-s)}\,(u)\,K_{t_{\text{aux}}}^{(s)}\,(t-u)}{K_w^{(n-s)}\,(i)\,K_{t_{\text{aux}}}^{(s)}\,(j)}\right|\leqslant\frac{1}{2\,(n+1)^2}\,n^{3.2}$$

and thus

$$\frac{1}{R_{i,j}^2}\geqslant\frac{2\,(n+1)^2}{n^{3.2}}.$$

Finally, Equation (6.27) shows that for n big enough

$$\frac{R_{i,j}}{\overline{N_i}}\geqslant n^{1.2}\max\left(\overline{V_j},\,1\right)$$

And as such, for n big enough

$$\begin{aligned}\frac{R_{i,j}}{\overline{N_i}} - \overline{V_j} &\geqslant n^{1.1} \max\left(\overline{V_j},\, 1\right) \\ &\geqslant n^{1.1} \max\left(\sqrt{\overline{V_j}},\, 1\right)\end{aligned}$$

which proves Equation (6.25). Therefore we have just proved Equation (6.13) in the case where $(i,j) \in \mathcal{A}$.

Case 2: . Here we suppose that $(i,j) \notin \mathcal{A}$. Let us prove Equation (6.14) that is prove that:

$$\mathbb{P}\left(\left|N_{i,j} - V_j\,\overline{N_i}\right| \geqslant R_{i,j}\right) = \mathcal{O}\left(\max_{v \in [\![0,\, M]\!]} \mathbb{P}\left(\left|N_{i,j} - v\overline{N_i}\right| \geqslant n^{1.1} \max\left(\sqrt{v\overline{N_i}},\, 1\right) \,\middle|\, V_j = v\right) + \mathbb{P}\left(V_j \geqslant M\right)\right).$$

By the law of total probability we have that

$$\begin{aligned}&\mathbb{P}_{\mathcal{C},\mathcal{C}_{\text{aux}},\mathbf{x}}\left(\left|N_{i,j} - V_j\overline{N_i}\right| > R_{i,j}\right) = \\ &\qquad \sum_{v=0}^{M} \mathbb{P}\left(\left|N_{i,j} - V_j\overline{N_i}\right| > R_{i,j} | V_j = v\right) \mathbb{P}\left(V_j = v\right) \\ &\qquad\quad + \mathbb{P}\left(\left|N_{i,j} - V_j\overline{N_i}\right| > R_{i,j} | V_j > M\right) \mathbb{P}\left(V_j > M\right) \\ &\qquad\qquad \leqslant \max_{v \in [\![0,\, M]\!]} \mathbb{P}\left(\left|N_{i,j} - v\,\overline{N_i}\right| > R_{i,j} | V_j = v\right) + \mathbb{P}\left(V_j > M\right).\end{aligned}$$

Now, to prove Eq. (6.14), we only have left to prove that for any $v \in [\![0, M]\!]$ and for n big enough we have that

$$n^{1.1} \max\left(\sqrt{v\,\overline{N_i}},\, 1\right) \leqslant R_{i,j}. \tag{6.28}$$

Let us show it. Let $v \in [\![0,\, M]\!]$. Using the definition

$$M \stackrel{\text{def}}{=} \overline{V_j} + n^{1.1} \max\left(\sqrt{\overline{V_j}}, 1\right)$$

we get that

$$\begin{aligned}n^{2.2} \max\left(v\,\overline{N_i},\, 1\right) &\leqslant n^{2.2} \max\left(M\,\overline{N_i},\, 1\right) \\ &\leqslant n^{2.2} \max\left(\overline{V_j}\,\overline{N_i} + n^{1.1}\overline{N_i} \max\left(\sqrt{\overline{V_j}}, 1\right),\, 1\right) \\ &\leqslant \max\left(2\,n^{2.2}\,\overline{V_j}\,\overline{N_i},\; 2\,n^{3.3}\overline{N_i}\sqrt{\overline{V_j}},\; 2\,n^{3.3}\overline{N_i},\; n^{2.2}\right)\end{aligned}$$

To show Equation (6.28), we only have left to prove that, for n big enough, each term in the previous maximum is smaller than $R_{i,j}^2$. First let us recall that by definition of $R_{i,j}$ in Corollary 10 and from Lemma 28,

$$R_{i,j}^2 \in \omega\left(\max\left(n^4\overline{N_i}\,\overline{V_j}, n^4\overline{N_i}\right)\right).$$

For n big enough we have that:

$$\begin{aligned}
&2\,n^{2.2}\,\overline{V_j}\,\overline{N_i} && \leqslant n^4\overline{N_i}\,\overline{V_j} \leqslant R_{i,j}^2, \\
&2\,n^{3.3}\overline{N_i}\sqrt{\overline{V_j}} && \leqslant \begin{cases} n^4\overline{N_i} \leqslant R_{i,j}^2 & \text{when } \overline{V_j} \leqslant 1 \\ n^4\overline{N_i}\,\overline{V_j} \leqslant R_{i,j}^2 & \text{when } \overline{V_j} > 1 \end{cases}, \\
&2\,n^{3.3}\,\overline{N_i} && \leqslant n^4\overline{N_i} \leqslant R_{i,j}^2, \\
&n^{2.2} && \leqslant R_{i,j}^2.
\end{aligned}$$

Where in the last equation we used the fact that $(i,j) \notin \mathcal{A}$, thus $R_{i,j} \geqslant \frac{n^{3.2}}{2(n+1)^2}$ and thus $R_{i,j}^2 \geqslant n^{2.3}$ for n big enough. We have shown that

$$n^{2.2}\max\left(v\,\overline{N_i},\,1\right) \leqslant R_{i,j}^2$$

and thus we have shown Equation (6.28). □

6.5.3 Proof of the main theorem

Here we prove our main Theorem 8. We refer the reader to Section 6.3 for the rationale behind our proof. Note that now we make the proof in the general case when the number of blocks is $b = \mathcal{O}(1)$. We recall that each sample $\mathbf{h}_{\mathscr{P}}$ is decoded using the following set

$$\mathrm{Dec}^{\mathrm{juxt}}\left(\mathcal{C}_{\mathrm{aux}}, \mathbf{h}_{\mathscr{P}}, t_{\mathrm{aux}}\right) \stackrel{\mathrm{def}}{=} \{\mathbf{e}_{\mathrm{aux}} \in \mathcal{S}^s_{t_{\mathrm{aux}}} \,:\, \forall i \in [\![1,\,b]\!],\, \mathbf{e}^{(i)}_{\mathrm{aux}} \in \mathcal{S}^{s^{(i)}}_{t^{(i)}_{\mathrm{aux}}} \text{ and } \mathbf{h}^{(i)}_{\mathscr{P}} - \mathbf{e}^{(i)}_{\mathrm{aux}} \in \mathcal{C}^{(i)}_{\mathrm{aux}}\}.$$

And we recall that $\mathbf{e}^{(i)}_{\mathscr{P}}$ refers to the projection of $\mathbf{e}_{\mathscr{P}}$ on the part of the support given by the i'th constituent code of $\mathcal{C}_{\mathrm{aux}} = \mathcal{C}^{(1)}_{\mathrm{aux}} \times \cdots \times \mathcal{C}^{(i)}_{\mathrm{aux}} \times \cdots \times \mathcal{C}^{(b)}_{\mathrm{aux}}$ (see Definition 39 and Notation 5). Notably the noise of our LPN samples is now

$$\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \langle \mathbf{e}_{\mathscr{P}}, \mathbf{h}_{\mathscr{P}} \rangle = \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle + \sum_{i=1}^{b} \left\langle \mathbf{e}^{(i)}_{\mathscr{P}}, \mathbf{e}^{(i)}_{\mathrm{aux}} \right\rangle.$$

The bias of this noise thus finely depends on the weight of $\mathbf{e}_{\mathscr{P}}$ on each of its part. To that extent we make, in addition to the bet that $|\mathbf{e}_{\mathscr{N}}| = u$, the bet that for $i \in [\![1,\,b]\!]$ we have that $\mathbf{e}^{(i)}_{\mathscr{P}}$ is of typical weight $((t-u)/b)$ say equal to $(t-u)^{(i)}$. As long as $b \in \mathcal{O}(1)$ this only incurs a polynomial loss in the probability that the bet is verified and we will show later that the bias of $\sum_{i=1}^{b} \left\langle \mathbf{e}^{(i)}_{\mathscr{P}}, \mathbf{e}^{(i)}_{\mathrm{aux}} \right\rangle$ in that case also polynomially relates to $\delta^{(s)}_{t_{\mathrm{aux}}}(t-u)$ (the bias in the case $b = 1$).

Lemma 29. *There exists a positive poly-bounded function f such that*

$$\mathbb{P}\left(\text{“}\,|\mathbf{e}_{\mathscr{N}}| = u \bigwedge_{i=1}^{b} \left|\mathbf{e}^{(i)}_{\mathscr{P}}\right| = (t-u)^{(i)}\text{”}\right) \geqslant \frac{1}{f(n^b)}\mathbb{P}\left(|\mathbf{e}_{\mathscr{N}}| = u\right)$$

where $\mathscr{P}$ and $\mathscr{N}$ are fixed complementary subsets of $[\![1,\,n]\!]$ and $\mathbf{e} \sim \mathcal{U}\left(\mathcal{S}^n_t\right)$.

All in all we will state our propositions using the following notation that represents the quantities encountered in a "good" iteration of double-RLPN.

Notation 7. *When $n, k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ are parameters and $\mathcal{C}$ is a linear code of length n such that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and $\mathcal{C}_{\text{aux}} \in \mathcal{U}\left(\mathfrak{C}^{\text{juxt}}[b,\ s,\ k_{\text{aux}}]\right)$ and $\mathbf{G}_{\text{aux}}$ is a generator matrix of $\mathcal{C}_{\text{aux}}$ we denote by:*

- *$\mathscr{P}$ and $\mathscr{N}$ are any fixed complementary subsets of $[\![1, n]\!]$ of size s and $n-s$ respectively.*
- *$\mathbf{y} = \mathbf{e}$ where $\mathbf{e} \in \mathcal{S}_t^n$ is any fixed vector such that the bet is valid*

$$|\mathbf{e}_{\mathscr{N}}| = u \text{ and } \left|\mathbf{e}_{\mathscr{P}}^{(i)}\right| = (t-u)^{(i)}\ \forall i \in [\![1,\ b]\!]$$

- *The set of decoded dual vectors is defined as*

$$\mathscr{H} \stackrel{def}{=} \{(\mathbf{h},\ \mathbf{m}_{\text{aux}}) \in \mathcal{C}^{\perp} \times \mathbb{F}_2^{k_{\text{aux}}} : |\mathbf{h}_{\mathscr{N}}| = w \text{ and } \left|(\mathbf{h}_{\mathscr{P}} + \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}})^{(i)}\right| = t_{\text{aux}}^{(i)} \forall i \in [\![1, b]\!]\}.$$

- *The score function is defined as*

$$\forall \mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}\ F^{\mathcal{L}}(\mathbf{z}) \stackrel{def}{=} \sum_{(\mathbf{h},\ \mathbf{m}_{\text{aux}}) \in \mathscr{H}} (-1)^{\langle \mathbf{y}, \mathbf{h}\rangle - \langle \mathbf{m}_{\text{aux}}, \mathbf{z}\rangle}.$$

- *The set of undecoded and decoded candidates are respectively:*

$$\mathcal{S} \stackrel{def}{=} \{\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}\ :\ F^{\mathcal{L}}(\mathbf{z}) > T\}, \qquad \mathcal{S}' \stackrel{def}{=} \bigcup_{\mathbf{z} \in \mathcal{S}} Dec^{\text{juxt}}\left(\mathcal{C}_{\text{aux}}^{\perp}, \mathbf{z}, t-u\right)$$

Proposition 53. *(Second-order behavior of the score function) For any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that*

$$N \in \frac{\omega\left(n^{b+1}\right)}{\delta^2} \quad \text{and} \quad N_{\text{aux}} \in \mathcal{O}(1) \quad \text{and} \quad b \in \mathcal{O}(1)$$

and such that $u < \text{Root}\left(K_w^{(n-s)}\right)$ and for all $i \in [\![1,\ b]\!]$ $(t-u)^{(i)} < \text{Root}\left(K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\right)$ then

$$\mathbb{P}\left(F^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}) \geqslant \frac{1}{2}N\delta\right) = 1 - o(1)$$

where

$$N \stackrel{def}{=} N_{\text{eq}}N_{\text{aux}},\ N_{\text{eq}} \stackrel{def}{=} \frac{\binom{n-s}{w}}{2^{k-s}},\ N_{\text{aux}} \stackrel{def}{=} \frac{\prod_{i=1}^{b}\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s-k_{\text{aux}}}},\ \delta \stackrel{def}{=} \delta_w^{(n-s)}(u)\prod_{i=1}^{b}\delta_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left((t-u)^{(i)}\right)$$

and where $\mathcal{C}$ is distributed according to $\mathcal{U}_{\mathbf{G}}(n,\ k)$ conditioned on the event that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and $\mathcal{C}_{\text{aux}} \sim \mathcal{U}\left(\mathfrak{C}^{\text{juxt}}[b,\ s,\ k_{\text{aux}}]\right)$ and where the other fixed quantities are defined in Notation 7.

We prove this technical proposition in Section 6.5.3.2.

Notation 8. *From now on the threshold T is chosen as*

$$T \stackrel{def}{=} \frac{1}{2}\delta N$$

Using basic properties of Krawtchouk polynomials we can polynomially relate the quantities appearing in Proposition 53 from the case $b \in \mathcal{O}(1)$ to the case $b = 1$.

Lemma 30 (Relating the bias to the case $b = 1$)**.** *There exists a positive poly-bounded function f_1 such that for any $k, t, s, u, t_{\text{aux}}, b \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that*

$$u < \text{Root}\left(K_w^{(n-s)}\right) \quad \text{and} \quad \forall i \in [\![1,\, b]\!]\ (t-u)^{(i)} < \text{Root}\left(K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\right)$$

then we have that

$$\begin{aligned}
\delta &\in \Omega\left(\frac{1}{f_1(n^b)}\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right),\\
N_{\text{aux}} &\in \Omega\left(\frac{1}{f_1(n^b)}\frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}}\right),\\
N &\in \Omega\left(\frac{1}{f_1(n^b)}\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}}\right).
\end{aligned}$$

where $\delta, N_{\text{aux}}, N$ are defined in Proposition 53.

Proof. Recalling that by definition

$$v^{(i)} \stackrel{\text{def}}{=} \begin{cases} \lfloor v/b \rfloor + 1 & \text{if } i \leqslant (v \mod b) \\ \lfloor v/b \rfloor & \text{else} \end{cases} \tag{6.29}$$

and using the condition that we are outside the root region of the Krawtchouk polynomials and Corollary 5 we have that

$$\delta_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left((t-u)^{(i)}\right) \in \tilde{\Omega}\left(\sqrt[b]{\delta_{t_{\text{aux}}}^{(s)}(t-u)}\right).$$

The other equalities are straightforward. □

Consequently it is readily seen that we have the following corollary giving that $\mathbf{e}_{\mathscr{P}}$ is in the final set of decoded candidates $\mathcal{S}'$ with good probability.

Corollary 11. *There exists a positive poly-bounded function f such that for any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ are implicit functions of $n \in \mathbb{N}$ such that*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \Omega\left(\frac{f(n^b)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}\right) \quad \text{and} \quad \frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} \in \mathcal{O}(1) \quad \text{and} \quad b \in \mathcal{O}(1)$$

and such that $u < \text{Root}\left(K_w^{(n-s)}\right)$ and for all $i \in [\![1,\, b]\!]$ $(t-u)^{(i)} < \text{Root}\left(K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\right)$ then

$$\mathbb{P}\left(\mathbf{e}_{\mathscr{P}} \in \mathcal{S}'\right) = 1 - o(1)$$

where the threshold is chosen as $T \stackrel{def}{=} \frac{1}{2}N\delta$ and where the distributions are specified by Proposition 53.

Proposition 54. *(Expected number of candidates when $b = 1$.) For any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \frac{\omega\left(n^8\right)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2} \quad \text{and} \quad \frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} \in \mathcal{O}(1) \quad \text{and} \quad b = 1$$

we have, under Conjecture 3, that

$$\mathbb{E}\left(\left|\mathcal{S}'\right|\right) = \widetilde{\mathcal{O}}(C)$$

where

$$C \overset{def}{=} \left[\max\left(\frac{\binom{s}{t-u}}{2^{k_{\text{aux}}}},\ 1\right) + \binom{s}{t-u} \max_{(i,j)\in\mathcal{A}}\left(\frac{\binom{n-s}{i}\binom{s}{j}}{2^{n-k+k_{\text{aux}}}}\right)\right] \max\left(1,\ \frac{2^{k_{\text{aux}}}}{\binom{s}{t-u}}\right)$$

and where $\mathcal{C}$ is distributed according to $\mathcal{U}(n,\ k)$ conditioned on the event that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and $\mathcal{C}_{\text{aux}} \sim \mathcal{U}\left(\mathfrak{C}^{\text{juxt}}[b,\ s,\ k_{\text{aux}}]\right)$ where all the other quantities are defined in Notation 7 and where we recall that

$$\mathcal{A} \overset{def}{=} \{(i,j) \in [\![0,\ n-s]\!] \times [\![0,\ s]\!] \ :\ \delta_w^{(n-s)}(i)\,\delta_{t_{\text{aux}}}^{(s)}(j) \geqslant \frac{\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)}{n^{3.2}}\}.$$

We prove this proposition in Section 6.5.3.3.

Remark 23. *We believe that the term $\max\left(1,\ \frac{2^{k_{\text{aux}}}}{\binom{s}{t-u}}\right)$ appearing at the end of $\mathbb{E}\left(|\mathcal{S}'|\right)$ could be deleted. We only need it to be able to reuse some of the proofs of [CDMT24] but other than that it is in fact artificial. Moreover, in practice, for our asymptotic parameters of interest it will always be equal to 1.*

Conjecture 5 (Generalization of Conjecture 3 + Proposition 54 to the case $b \in \mathcal{O}(1)$)**.** *We conjecture that there exists a positive poly-bounded function f such that for any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \frac{\omega\left(f(n^b)\right)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2} \quad \text{and} \quad \frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} \in \mathcal{O}(1) \quad \text{and} \quad b \in \mathcal{O}(1)$$

we have that

$$\mathbb{E}\left(\left|\mathcal{S}'\right|\right) = \widetilde{\mathcal{O}}(C)$$

where $\mathcal{C}$ is distributed according to $\mathcal{U}(n,\ k)$ conditioned on the event that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and $\mathcal{C}_{\text{aux}} \sim \mathcal{U}\left(\mathfrak{C}^{\text{juxt}}[b,\ s,\ k_{\text{aux}}]\right)$ and where C is defined in Proposition 54.

6.5.3.1 Proof of the main theorem

Proof of the main theorem. The function f in Theorem 8 is the maximum of all the functions appearing in the previous propositions and lemmas. By the assumptions on the parameters of Theorem 8 clearly there exists with probability $1 - o(1)$ an iteration such that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and such that the fine bet is valid, namely such that $|\mathbf{e}_{\mathscr{N}}| = u$ and $\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right| = (t-u)^{(i)} \forall i \in [\![1, b]\!]$

and such that the procedure computing the set of dual vectors outputs all dual vectors of weight w on $\mathscr{N}$. In such an iteration, we have, using Proposition 53 that $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal} \in \mathcal{S}$ with probability $1 - o(1)$. This automatically implies that $\mathbf{e}_{\mathscr{P}} \in \mathcal{S}'$ (this is due to the decoder of $\mathcal{C}_{\text{aux}}^{\perp}$ defined in Algorithm 3 and the fact that the fine bet on the error weight distribution is valid). Finally, because ISD-DUMER solves $\text{DP}_{\mathbf{G}}(n-s, k-s, u)$ with probability $1-o(1)$, the whole error $\mathbf{e}$ is outputted by the algorithm. Last, for the complexity of the algorithm, there exists a positive poly-bounded function g such that stopping an iteration if $|\mathcal{S}'|$ is bigger than gC (where C is defined in Proposition 54) allows us to ensure that the algorithm has the claimed complexity while not damaging the probability of success. In the case $b = 1$, we obtain this with Proposition 54 and in the case where $b \in \mathcal{O}(1)$, this result is given by Conjecture 5. □

6.5.3.2 Proof of the second-order concentration bounds

Let us prove our concentration bound, we recall Proposition 53 here.

Proposition 53. *(Second-order behavior of the score function) For any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that*

$$N \in \frac{\omega\left(n^{b+1}\right)}{\delta^2} \quad \text{and} \quad N_{\text{aux}} \in \mathcal{O}(1) \quad \text{and} \quad b \in \mathcal{O}(1)$$

and such that $u < \text{Root}\left(K_w^{(n-s)}\right)$ and for all $i \in [\![1,\, b]\!]$ $(t-u)^{(i)} < \text{Root}\left(K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\right)$ then

$$\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) \geqslant \frac{1}{2}N\delta\right) = 1 - o(1)$$

where

$$N \overset{def}{=} N_{\text{eq}}N_{\text{aux}},\ N_{\text{eq}} \overset{def}{=} \frac{\binom{n-s}{w}}{2^{k-s}},\ N_{\text{aux}} \overset{def}{=} \frac{\prod_{i=1}^{b}\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s-k_{\text{aux}}}},\ \delta \overset{def}{=} \delta_w^{(n-s)}(u)\prod_{i=1}^{b}\delta_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left((t-u)^{(i)}\right)$$

and where $\mathcal{C}$ is distributed according to $\mathcal{U}_{\mathbf{G}}(n,\ k)$ conditioned on the event that $\dim(\mathcal{C}_{\mathscr{P}}) = s$ and $\mathcal{C}_{\text{aux}} \sim \mathcal{U}\left(\mathfrak{C}^{\text{juxt}}[b,\ s,\ k_{\text{aux}}]\right)$ and where the other fixed quantities are defined in Notation 7.

The main technical lemma that we will use here is the following giving the first two moments of the score function.

Lemma 31. *(Main technical lemma.) For any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ are implicit functions of $n \in \mathbb{N}$ we have that*

$$\begin{aligned}\mathbb{E}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) &= N\,\delta \\ \mathbf{Var}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) &= \mathcal{O}\left(n^{b+1}\,N\,\max\left(1,\ N_{\text{aux}}\right)\right)\end{aligned}$$

where the quantities and distributions are defined in Proposition 53.

Remark 24. *We believe that the term n^{b+1} appearing in the variance is much more reasonable and is rather some $\mathcal{O}\left(2^b\right)$ in the constant rate regime.*

This allows us to prove our main statement.

Proof of Proposition 53. Suppose that the parameters verify the constraints of the proposition, namely that

$$N \in \frac{\omega\left(n^{b+1}\right)}{\delta^2} \quad \text{and} \quad N_{\text{aux}} \in \mathcal{O}(1)\,.$$

We have that

$$\begin{aligned}\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) \leqslant \frac{1}{2}N\delta\right) &\leqslant \mathbb{P}\left(\left|F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) - N\delta\right| \geqslant \frac{1}{2}N\delta\right)\\ &\leqslant \mathbb{P}\left(\left|F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) - N\delta\right| \geqslant \frac{1}{2}\frac{N\delta}{\sqrt{\mathbf{Var}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right)}}\sqrt{\mathbf{Var}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right)}\right)\end{aligned}$$

Now using the expression of the variance given in the previous Lemma 31 we have that

$$\begin{aligned}\frac{1}{2}\frac{N\delta}{\sqrt{\mathbf{Var}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right)}} &= \Omega\left(\frac{N\delta}{\sqrt{n^{b+1}N\max\left(1,\ N_{\text{aux}}\right)}}\right)\\ &= \omega\left(1\right).\end{aligned}$$

where in the last line we use the fact that the parameters verify the constraints. Applying Byenemé-Chebychev we get that

$$\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) \leqslant \frac{1}{2}N\delta\right) = o(1)$$

and thus

$$\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) \geqslant \frac{1}{2}N\delta\right) = 1 - o(1)$$

which yields our result. □

6.5.3.2.1 Moments of the score function Here we prove Lemma 31 that we recall here.

Lemma 31. *(Main technical lemma.) For any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ are implicit functions of $n \in \mathbb{N}$ we have that*

$$\begin{aligned}\mathbb{E}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) &= N\,\delta\\ \mathbf{Var}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) &= \mathcal{O}\Big(n^{b+1}\ N\ \max\left(1,\ N_{\text{aux}}\right)\Big)\end{aligned}$$

where the quantities and distributions are defined in Proposition 53.

We recall for convenience two lemmas we will be usefull to prove the proposition. We recall here the distribution appearing in the previous expression and which will

Lemma 32 (Distribution of some related quantities)**.** *For any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ we have that*

$$\mathbf{R} \sim \mathcal{U}\left(\mathbb{F}_2^{s\times(n-s)}\right), \tag{6.30}$$

$$\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp} \sim \mathcal{U}_{\mathbf{H}}\left(n-s,\ n-k\right), \tag{6.31}$$

$$\mathbf{R} \text{ and } \mathcal{C}^{\mathscr{N}} \text{ are independent.} \tag{6.32}$$

where $\mathbf{R} \overset{def}{=} \text{Lift}\left(\mathcal{C}^{\perp},\ \mathscr{N}\right)$ and where the other quantities are defined in Proposition 53.

Proof. This is exactly Proposition 38. □

This allows us to get the following lemma

Lemma 33. *We have that*

$$\mathbb{E}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = N\delta,$$
$$\mathbf{Var}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) \leqslant N\left[1+\sum_{\mathbf{c}\in\{0,1\}^b\,:\,\mathbf{c}\neq\mathbf{0}}\prod_{i=1}^{b}\left(\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{c_i}\right]$$

where the quantities are defined in Proposition 53.

Proof. Let us first compute the expected value. Recall that we have that

$$|\mathbf{e}_{\mathscr{N}}| = u,\ \left|\mathbf{e}_{\mathscr{P}}^{(1)}\right| = (t-u)^{(1)},\ \cdots,\ \left|\mathbf{e}_{\mathscr{P}}^{(b)}\right| = (t-u)^{(b)} \tag{6.33}$$

We will show that

$$\mathbb{E}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = N\,\delta.$$

Rewritting the score function we have that

$$F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)$$
$$= \sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_w^{n-s}}\ \sum_{\substack{\mathbf{e}_{\text{aux}}^{(1)}\in\mathcal{S}_{t_{\text{aux}}^{(1)}}^{s^{(1)}}}}\cdots\sum_{\substack{\mathbf{e}_{\text{aux}}^{(b)}\in\mathcal{S}_{t_{\text{aux}}^{(b)}}^{s^{(b)}}}}(-1)^{\langle\mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle}\left[\prod_{i=1}^{b}(-1)^{\left\langle\mathbf{e}_{\mathscr{P}}^{(i)},\mathbf{e}_{\text{aux}}^{(i)}\right\rangle}\right]\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}} \tag{6.34}$$

By linearity of the expected value we have that

$$\mathbb{E}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = \sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_w^{n-s}}\ \sum_{\substack{\mathbf{e}_{\text{aux}}^{(1)}\in\mathcal{S}_{t_{\text{aux}}^{(1)}}^{s^{(1)}}}}\cdots\sum_{\substack{\mathbf{e}_{\text{aux}}^{(b)}\in\mathcal{S}_{t_{\text{aux}}^{(b)}}^{s^{(b)}}}}$$
$$\left[(-1)^{\langle\mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle}\prod_{i=1}^{b}(-1)^{\left\langle\mathbf{e}_{\mathscr{P}}^{(i)},\mathbf{e}_{\text{aux}}^{(i)}\right\rangle}\right]\left[\mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)\right] \tag{6.35}$$

Now from the independence of the indicator variable we get that

$$\begin{aligned}\mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right) &= \mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}}\right)\prod_{i=1}^{b}\mathbb{E}\left(\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)\\ &= \mathbb{P}\left(\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}\right)\prod_{i=1}^{b}\mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}\right).\end{aligned} \tag{6.36}$$

From Lemma 32 we have that $\mathcal{C}^{\mathscr{N}}\sim\mathcal{U}_{\mathbf{G}}\left(n-s,\ k-s\right)$ thus using Proposition 5 we get

$$\mathbb{P}\left(\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}\right) = \frac{1}{2^{k-s}}.$$

From Lemma 32 we have that for each $i \in [\![1,\ b]\!]$, $\mathbf{e}_{\text{aux}}^{(i)} + (\mathbf{h}_{\mathscr{N}}\mathbf{R}^\intercal)^{(i)} \sim \mathcal{U}\left(\mathbb{F}_2^{s^{(i)}}\right)$, thus as $\mathcal{C}_{\text{aux}}^{(i)} \in \mathfrak{C}\left[s^{(i)}, k_{\text{aux}}^{(i)}\right]$ we get that

$$\mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)} + (\mathbf{h}_{\mathscr{N}}\mathbf{R}^\intercal)^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}\right) = \frac{1}{2^{s^{(i)}-k_{\text{aux}}^{(i)}}}. \tag{6.37}$$

Plugging these last equation into Eq. (6.36) yield that

$$\begin{aligned}\mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in(\mathcal{C}^{\mathscr{N}})^\perp}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^\intercal)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right) &= \frac{1}{2^{k-s}}\prod_{i=1}^{b}\frac{1}{2^{s^{(i)}-k_{\text{aux}}^{(i)}}}\\ &= \frac{1}{2^{k-k_{\text{aux}}}}\end{aligned} \tag{6.38}$$

where in the last equation we used the fact that by definition of the i'th part of a vector in Eq. (6.3) we have that

$$\sum_{i=1}^{b} s^{(i)} = s, \qquad \sum_{i=1}^{b} k_{\text{aux}}^{(i)} = k_{\text{aux}}.$$

Finally, plugging this last equality back into Eq. (6.35) gives that

$$\begin{aligned}\mathbb{E}\left(F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^\intercal\right)\right) &= \frac{1}{2^{k-k_{\text{aux}}}}\sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_w^{n-s}}\ \sum_{\mathbf{e}_{\text{aux}}^{(1)}\in\mathcal{S}^{s^{(1)}}_{t_{\text{aux}}^{(1)}}}\cdots\sum_{\mathbf{e}_{\text{aux}}^{(b)}\in\mathcal{S}^{s^{(b)}}_{t_{\text{aux}}^{(b)}}}(-1)^{\langle\mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle}\prod_{i=1}^{b}(-1)^{\left\langle\mathbf{e}_{\mathscr{P}}^{(i)},\mathbf{e}_{\text{aux}}^{(i)}\right\rangle}\\ &= \frac{1}{2^{k-k_{\text{aux}}}}K_w^{(n-s)}\left(|\mathbf{e}_{\mathscr{N}}|\right)\prod_{i=1}^{b}K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left(\left|\mathbf{e}_{\mathscr{P}}^{(i)}\right|\right)\\ &= \frac{1}{2^{k-k_{\text{aux}}}}K_w^{(n-s)}\left(u\right)\prod_{i=1}^{b}K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left((t-u)^{(i)}\right) \qquad (Eq.\ (6.33))\\ &= \frac{\binom{n-s}{w}\prod_{i=1}^{b}\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{k-k_{\text{aux}}}}\frac{K_w^{(n-s)}\left(u\right)\prod_{i=1}^{b}K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left((t-u)^{(i)}\right)}{\binom{n-s}{w}\prod_{i=1}^{b}\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}\\ &= N\delta_w^{(n-s)}\left(u\right)\prod_{i=1}^{b}\delta_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\left(v^{(i)}\right)\\ &= N\delta\end{aligned}$$

where in the last lines we used the definition of N and δ given in Proposition 53. This concludes the computation of the expected value.

Let us now compute the variance of $F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^\intercal\right)$. Starting again from Equation (6.34) and denoting by $A \in \{-1, 1\}$ the value:

$$A\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}}^{(1)},\ \ldots,\ \mathbf{e}_{\text{aux}}^{(b)}\right) \stackrel{\text{def}}{=} (-1)^{\langle\mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle}\left[\prod_{i=1}^{b}(-1)^{\left\langle\mathbf{e}_{\mathscr{P}}^{(i)},\mathbf{e}_{\text{aux}}^{(i)}\right\rangle}\right]$$

we have that

$$F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^\intercal\right) = \sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_w^{n-s}}\ \sum_{\left(\mathbf{e}_{\text{aux}}^{(i)}\in\mathcal{S}^{s^{(i)}}_{t_{\text{aux}}^{(i)}}\right)_{i\in[\![1,\ b]\!]}} A\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}}^{(1)},\ \ldots,\ \mathbf{e}_{\text{aux}}^{(b)}\right)\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in(\mathcal{C}^{\mathscr{N}})^\perp}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^\intercal)^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}.$$

Now we use the fact that we can upper bound the variance of $\sum_i A_i \mathbf{X}_i$ where $\mathbf{X}_i$ are some random variables the $A_i \in \{-1, 1\}$ are some fixed coefficient by upper bounding the covariances as

$$\begin{aligned} Cov\left(A_i\, X_i, A_j\, X_j\right) &= A_i A_j Cov\left(X_i, X_j\right) \\ &\leqslant \left|Cov\left(X_i, X_j\right)\right|. \end{aligned}$$

This observation allows us to write that

$$\mathbf{Var}\left(F\left(\mathbf{e}_{\mathscr{D}} \mathbf{G}_{\text{aux}}^{\intercal}\right)\right) \leqslant V + C \tag{6.39}$$

where

$$V \stackrel{\text{def}}{=} \sum_{\mathbf{h}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \sum_{\substack{\left(\mathbf{e}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}}} \mathbf{Var}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} + \left(\mathbf{h}_{\mathscr{N}} \mathbf{R}^{\intercal}\right)^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}}\right) \tag{6.40}$$

and

$$\begin{aligned} C \stackrel{\text{def}}{=} \sum_{\mathbf{h}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \sum_{\mathbf{g}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \sum_{\substack{\left(\mathbf{e}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}}} \sum_{\substack{\left(\mathbf{z}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}}} \\ \mathbf{1}_{\left(\mathbf{h}_{\mathscr{N}},\, \mathbf{e}_{\text{aux}}\right) \neq \left(\mathbf{g}_{\mathscr{N}},\, \mathbf{z}_{\text{aux}}\right)} \left|\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)\right| \end{aligned} \tag{6.41}$$

where

$$\begin{aligned} &\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right) \\ &\quad \stackrel{\text{def}}{=} \mathbf{Cov}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} + \left(\mathbf{h}_{\mathscr{N}} \mathbf{R}^{\intercal}\right)^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}},\ \mathbf{1}_{\mathbf{g}_{\mathscr{N}} \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{z}_{\text{aux}}^{(i)} + \left(\mathbf{h}_{\mathscr{N}} \mathbf{R}^{\intercal}\right)^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}}\right). \end{aligned} \tag{6.42}$$

Let us first compute the term V. As V is the variance of a Bernoulli distribution, we can upper bound it by the expected value of this Bernouilli:

$$\begin{aligned} \mathbf{Var}\left(\mathbf{1}_{\left(\mathbf{h}_{\mathscr{N}} \mathbf{R}^{\intercal}, \mathbf{h}_{\mathscr{N}}\right) \in \mathscr{W}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} + \left(\mathbf{h}_{\mathscr{N}} \mathbf{R}^{\intercal}\right)^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}}\right) &\leqslant \mathbb{E}\left(\mathbf{1}_{\left(\mathbf{h}_{\mathscr{N}} \mathbf{R}^{\intercal}, \mathbf{h}_{\mathscr{N}}\right) \in \mathscr{W}} \prod_{i=1}^{b} \mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} + \left(\mathbf{h}_{\mathscr{N}} \mathbf{R}^{\intercal}\right)^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}}\right) \\ &= \frac{1}{2^{k - k_{\text{aux}}}} \qquad (Eq.\ (6.38)) \end{aligned}$$

And thus, plugging this last equation in Eq. (6.40) we get

$$\begin{aligned} V &\leqslant \sum_{\mathbf{h}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \sum_{\substack{\left(\mathbf{e}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}}} \frac{1}{2^{k - k_{\text{aux}}}} \\ &= \frac{\binom{n-s}{w} \prod_{i=1}^{b} \binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{k - k_{\text{aux}}}} \\ &= N. \end{aligned}$$

Let us now compute the covariance terms C by first rewriting $\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)$. We have

$$\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right) = \mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\mathbf{1}_{\mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{z}_{\text{aux}}^{(i)}+(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)$$
$$-\mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}},\ \prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)\mathbb{E}\left(\mathbf{1}_{\mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}},\ \prod_{i=1}^{b}\mathbf{1}_{\mathbf{z}_{\text{aux}}^{(i)}+(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right)$$

Thus

$$\begin{aligned}
&\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right) = \\
&= \mathbb{E}\left(\mathbf{1}_{\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}}\mathbf{1}_{\mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}}\prod_{i=1}^{b}\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\mathbf{1}_{\mathbf{z}_{\text{aux}}^{(i)}+(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}}\right) - \left(\frac{1}{2^{k-k_{\text{aux}}}}\right)^{2} \\
&= \mathbb{P}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}\right)\prod_{i=1}^{b}\mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)},\ \mathbf{z}_{\text{aux}}^{(i)}+(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}\right) - \left(\frac{1}{2^{k-k_{\text{aux}}}}\right)^{2}
\end{aligned} \tag{6.43}$$

where in the last line we used the independence of the variables.

1. Case $\mathbf{h}_{\mathscr{N}}\neq\mathbf{g}_{\mathscr{N}}$. Suppose here that $\mathbf{h}_{\mathscr{N}}\neq\mathbf{h}_{\mathscr{N}}$ Let us first compute $\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)$. First, as $\mathcal{C}^{\mathscr{N}}\sim\mathcal{U}_{\mathbf{G}}\left(n-s,\ k-s\right)$ we get from Proposition 5 that

$$\mathbb{P}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}\right) = \left(\frac{1}{2^{k-s}}\right)^{2}$$

Moreover, regardless of the values of $\mathbf{e}_{\text{aux}}^{(i)}$ and $\mathbf{z}_{\text{aux}}^{(i)}$ we have that $\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}$ and $\mathbf{z}_{\text{aux}}^{(i)}+(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}$ are independent and uniformly distributed which yields that

$$\text{if}\quad \mathbf{h}_{\mathscr{N}}\neq\mathbf{g}_{\mathscr{N}}\quad\text{then}\quad \mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)},\ \mathbf{z}_{\text{aux}}^{(i)}+(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}\right) = \left(\frac{1}{2^{s^{(i)}-k_{\text{aux}}^{(i)}}}\right)^{2}.$$

Plugging this last equality back into $\mathbf{C}$ gives that

$$\textbf{If}\quad \mathbf{h}_{\mathscr{N}}\neq\mathbf{g}_{\mathscr{N}}\ \textbf{then}\quad \mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{g}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)=0.$$

Using this fact in Eq. (6.41) gives that

$$C = \sum_{\mathbf{h}_{\mathscr{N}}\in\mathcal{S}_{w}^{n-s}}\ \sum_{\left(\mathbf{e}_{\text{aux}}^{(i)}\in\mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i\in[\![1,\,b]\!]}}\ \sum_{\left(\mathbf{z}_{\text{aux}}^{(i)}\in\mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i\in[\![1,\,b]\!]}} \mathbf{1}_{\mathbf{e}_{\text{aux}}\neq\mathbf{z}_{\text{aux}}}\left|\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)\right| \tag{6.44}$$

2. Case $\mathbf{h}_{\mathscr{N}}=\mathbf{g}_{\mathscr{N}}$. Let us now compute $\left|\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)\right|$. Recall that from Eq. (6.43) we have that

$$\begin{aligned}
&\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right) = \\
&= \mathbb{P}\left(\mathbf{h}_{\mathscr{N}}\in\left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}\right)\prod_{i=1}^{b}\mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)}+(\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)},\ \mathbf{z}_{\text{aux}}^{(i)}+(\mathbf{g}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)}\in\mathcal{C}_{\text{aux}}^{(i)}\right) - \left(\frac{1}{2^{k-k_{\text{aux}}}}\right)^{2}.
\end{aligned}$$

But as we have that

$$\begin{aligned}
&\mathbb{P}\left(\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}\right) \prod_{i=1}^{b} \mathbb{P}\left(\mathbf{e}_{\text{aux}}^{(i)} + (\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)},\ \mathbf{z}_{\text{aux}}^{(i)} + (\mathbf{h}_{\mathscr{N}}\mathbf{R}^{\intercal})^{(i)} \in \mathcal{C}_{\text{aux}}^{(i)}\right) \\
&\leqslant \frac{1}{2^{k-s}} \prod_{i=1}^{b} \left(\frac{1}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{1+\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} \neq \mathbf{z}_{\text{aux}}^{(i)}}} \\
&= \frac{1}{2^{k-k_{\text{aux}}}} \prod_{i=1}^{b} \left(\frac{1}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} \neq \mathbf{z}_{\text{aux}}^{(i)}}}
\end{aligned}$$

This yield that

$$|\mathbf{C}\left(\mathbf{h}_{\mathscr{N}},\ \mathbf{h}_{\mathscr{N}},\ \mathbf{e}_{\text{aux}},\ \mathbf{z}_{\text{aux}}\right)| = \mathcal{O}\left(\frac{1}{2^{k-k_{\text{aux}}}} \prod_{i=1}^{b} \left(\frac{1}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} \neq \mathbf{z}_{\text{aux}}^{(i)}}}\right).$$

Finally

$$\begin{aligned}
C &= \mathcal{O}\left(\sum_{\mathbf{h}_{\mathscr{N}} \in \mathcal{S}_w^{n-s}} \sum_{\left(\mathbf{e}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}} \sum_{\left(\mathbf{z}_{\text{aux}}^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\right)_{i \in [\![1,\, b]\!]}} \mathbf{1}_{\mathbf{e}_{\text{aux}} \neq \mathbf{z}_{\text{aux}}} \frac{1}{2^{k-k_{\text{aux}}}} \prod_{i=1}^{b} \left(\frac{1}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{\mathbf{1}_{\mathbf{e}_{\text{aux}}^{(i)} \neq \mathbf{z}_{\text{aux}}^{(i)}}}\right) \\
&= \mathcal{O}\left(\frac{\binom{n-s}{w} \prod_{i=1}^{b} \binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{k-k_{\text{aux}}}} \sum_{\mathbf{c} \in \{0,1\}^b\,:\,\mathbf{c} \neq \mathbf{0}} \prod_{i=1}^{b} \left(\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{c_i}\right) \\
&= \mathcal{O}\left(N \sum_{\mathbf{c} \in \{0,1\}^b\,:\,\mathbf{c} \neq \mathbf{0}} \prod_{i=1}^{b} \left(\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{c_i}\right).
\end{aligned}$$

Plugging this last upper bound for C along with the expression of V in Eq. (6.40) in Eq. (6.39) allows writing

$$\mathbf{Var}\left(F^{\mathcal{L}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) \leqslant N\left(1 + \mathcal{O}\left(\sum_{\mathbf{c} \in \{0,1\}^b\,:\,\mathbf{c} \neq \mathbf{0}} \prod_{i=1}^{b} \left(\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-k_{\text{aux}}^{(i)}}}\right)^{c_i}\right)\right)$$

□

We can now prove our main lemma.

Proof of Lemma 31. The expected values are already given by Lemma 33, we only have to show that

$$\mathbf{Var}\left(F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) = \mathcal{O}\Big(N\, n^{b+1}\ \max\left(1,\ N_{\text{aux}}\right)\Big).$$

Recall that from Lemma 33, we have that

$$\mathbf{Var}\left(F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) \leqslant N\left[1 + \sum_{\mathbf{c} \in \{0,1\}^b\,:\,\mathbf{c} \neq \mathbf{0}} \prod_{i=1}^{b} \left(\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{c_i}\right]. \tag{6.45}$$

The result will come from the fact that, as for any i, j we have that

$$\begin{aligned} s^{(i)} &= s^{(j)} \pm 1, \\ k_{\text{aux}}^{(i)} &= k_{\text{aux}}^{(j)} \pm 1, \\ t_{\text{aux}}^{(i)} &= t_{\text{aux}}^{(i)} \pm 1, \end{aligned}$$

we can write that

$$\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-k_{\text{aux}}^{(i)}}} = \mathcal{O}\left(\frac{s^{(i)}+1}{t_{\text{aux}}^{(i)}+1}\frac{\binom{s^{(j)}}{t_{\text{aux}}^{(j)}}}{2^{s^{(j)}-k_{\text{aux}}^{(j)}}}\right) \tag{6.46}$$

$$= \mathcal{O}(n)\,\frac{\binom{s^{(j)}}{t_{\text{aux}}^{(j)}}}{2^{s^{(j)}-k_{\text{aux}}^{(j)}}} \tag{6.47}$$

The previous equation yields that, regardless of $\mathbf{c} \in \{0,1\}^b$ in Eq. (6.45) we have that

$$\begin{aligned} \prod_{i=1}^{b}\left(\frac{\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s^{(i)}-t_{\text{aux}}^{(i)}}}\right)^{c_i} &= \mathcal{O}\left(n^b\ \max\left(1,\ \frac{\prod_{i=1}^{b}\binom{s^{(i)}}{t_{\text{aux}}^{(i)}}}{2^{s-k_{\text{aux}}}}\right)\right) \\ &= \mathcal{O}\Big(n^b\ \max\left(1,\ N_{\text{aux}}\right)\Big). \end{aligned}$$

Plugging this last equation in Eq. (6.45) we get:

$$\begin{aligned} \mathbf{Var}\left(F\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)\right) &\leqslant N\left[1+\sum_{\mathbf{c}\in\{0,1\}^b\,:\,\mathbf{c}\neq\mathbf{0}}\mathcal{O}\Big(n^b\ \max\left(1,\ N_{\text{aux}}\right)\Big)\right] \\ &\leqslant N\left[1+\ \mathcal{O}\Big(n^{b+1}\ \max\left(1,\ N_{\text{aux}}\right)\Big)\right] \\ &= \mathcal{O}\Big(N\,n^{b+1}\ \max\left(1,\ N_{\text{aux}}\right)\Big). \end{aligned}$$

Which is our desired result. □

6.5.3.3 Proof of the bound on the number of candidates

Let us prove the proposition giving the number of candidates that we restate here Proposition 54.

Proposition 54. *(Expected number of candidates when $b = 1$.) For any $k, t, s, u, w, k_{\text{aux}}, t_{\text{aux}}, b \in \mathbb{N}$ implicit functions of $n \in \mathbb{N}$ such that*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \frac{\omega\left(n^8\right)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2} \quad \textit{and} \quad \frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} \in \mathcal{O}(1) \quad \textit{and} \quad b = 1$$

we have, under Conjecture 3, that

$$\mathbb{E}\left(\left|\mathcal{S}'\right|\right) = \widetilde{\mathcal{O}}(C)$$

where

$$C \stackrel{def}{=} \left[\max\left(\frac{\binom{s}{t-u}}{2^{k_{\text{aux}}}},\ 1\right) + \binom{s}{t-u} \max_{(i,j)\in\mathcal{A}} \left(\frac{\binom{n-s}{i}\binom{s}{j}}{2^{n-k+k_{\text{aux}}}}\right)\right] \max\left(1,\ \frac{2^{k_{\text{aux}}}}{\binom{s}{t-u}}\right)$$

and where $\mathcal{C}$ *is distributed according to* $\mathcal{U}(n,\ k)$ *conditioned on the event that* $\dim(\mathcal{C}_{\mathscr{P}}) = s$ *and* $\mathcal{C}_{\text{aux}} \sim \mathcal{U}\left(\mathfrak{C}^{\text{juxt}}[b,\ s,\ k_{\text{aux}}]\right)$ *where all the other quantities are defined in Notation 7 and where we recall that*

$$\mathcal{A} \stackrel{def}{=} \{(i,j) \in [\![0,\ n-s]\!] \times [\![0,\ s]\!] \ :\ \delta_w^{(n-s)}(i)\,\delta_{t_{\text{aux}}}^{(s)}(j) \geqslant \frac{\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)}{n^{3.2}}\}.$$

Lemma 34. *For any* $k, t, w, s, u, k_{\text{aux}}, t_{\text{aux}}$ *implicit functions of* $n \in \mathbb{N}$ *we have that*

$$\mathbb{E}\left(\left|\mathcal{S}'\right|\right) = \mathcal{O}\left(E + \frac{\binom{s}{t-u}}{2^{k_{\text{aux}}}}\right) \max\left(1,\ \frac{2^{k_{\text{aux}}}}{\binom{t-u}{s}}\right)$$

where we defined

$$E \stackrel{def}{=} \binom{s}{t-u} \mathbb{P}\left(F^{\mathcal{L}}(\mathbf{z}) > T\right)$$

and where $\mathbf{z} \sim \mathcal{U}\left(\mathbb{F}_2^{k_{\text{aux}}} \setminus \{\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\}\right)$ *and where the other quantities and distributions are defined in Proposition 54.*

Remark 25. *The previous expression for* $\mathbb{E}(|\mathcal{S}'|)$ *is somewhat slightly more messy and less tight than what we could get optimally. However, in practice this is not a problem: basically the only term of interest here is the main dominating term* E*. The term* $\binom{s}{t-u}/2^{k_{\text{aux}}}$ *appears because we upper bounded the probability that the secret* $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ *is in the set of candidates by* 1*, it will be negligible compared to* E*. The multiplicative term* $\max\left(1,\ 2^{k_{\text{aux}}}/\binom{t-u}{s}\right)$ *is a smoothing term which is slightly artificial and comes because of our choice for the distribution of* $\mathbf{z}$*: it could be deleted by taking* $\mathbf{z}$ *distributed as some* $\mathbf{a}\mathbf{G}_{\text{aux}}^{\intercal}$ *where* $\mathbf{a} \sim \mathcal{U}\left(\mathcal{S}_{t-u}^{s} \setminus \{\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{e}_{\mathscr{P}}\}\right)$*. But, we stick with our choice of for the distribution of* $\mathbf{z}$ *as it makes our discussion slightly easier and, in practice, all our optimal parameters are such that* $\binom{s}{t-u}/2^{k_{\text{aux}}} = 2^{\Omega(n)}$ *consequently having this term is really not a problem.*

The proof of Lemma 34 is given in Section 6.5.3.3.2. This directly yields the following corollary.

Corollary 12. *For any* $k, t, w, s, u, k_{\text{aux}}, t_{\text{aux}}, b$ *implicit functions of* $n \in \mathbb{N}$ *such that*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \frac{\omega\left(n^8\right)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2} \quad \text{and} \quad \frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} \in \mathcal{O}(1) \quad \text{and} \quad b = 1$$

then under Conjecture 3 we have that

$$\mathbb{E}\left(\left|\mathcal{S}'\right|\right) = \widetilde{\mathcal{O}}\Big(\mathbb{P}\left(\exists (i,j) \in \mathcal{A} \ :\ N_{i,j}^{(\mathbf{x})} \neq 0\right)\Big)$$

where $\mathbf{x} \sim \mathcal{U}\left(\mathbb{F}_2^s \setminus \{\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{e}_{\mathscr{P}}\ \}\right)$ *and where we recall that*

$$N_{i,j}^{(\mathbf{x})} \stackrel{def}{=} \left|\left\{\left(\mathbf{r}, \mathbf{c}^{\mathscr{N}}\right) \in (\mathbf{x} + \mathcal{C}_{\text{aux}}^{\perp}) \times \mathcal{C}^{\mathscr{N}} \ : |\mathbf{r}| = j \text{ and } \left|(\mathbf{r} + \mathbf{e}_{\mathscr{P}})\,\mathbf{R} + \mathbf{e}_{\mathscr{N}} + \mathbf{c}^{\mathscr{N}}\right| = i\right\}\right|$$

and $\mathbf{R} \stackrel{def}{=} \text{Lift}\left(\mathcal{C}^{\perp},\ \mathscr{N}\right)$ *and where the other distributions and quantities are defined in Proposition 54.*

Proof. This is a direct application of Conjecture 3 as well as using the fact that $\mathbf{G}_{\text{aux}}$ is of full rank k_{aux} to argue that we can switch distribution from $\mathbf{z}$ in Proposition 54 to $\mathbf{x}$ in Corollary 12. □

Lemma 35. *(Distribution of $N_{i,j}$) We have that*

$$\mathbb{P}\left(\exists(i,j)\in\mathcal{A}\ :\ N_{i,j}^{(\mathbf{x})}\neq 0\right)=\widetilde{\mathcal{O}}\left(\max_{(i,j)\in\mathcal{A}}\frac{\binom{n-s}{i}\binom{s}{j}}{2^{n-k+k_{\text{aux}}}}\right)$$

and where all the other quantities and distributions are defined in Corollary 12.

The proof of this statement is made in Section 6.5.3.3.1.

Proof of Proposition 54. This is done by using Corollary 12 along with Lemma 35. □

We only have left to prove the two previous lemmas, namely Lemma 34 and Lemma 35. This is done in the two following sections.

6.5.3.3.1 Proof regarding the distribution of the weight enumerator

This section is dedicated to proving Lemma 35 that we recall here

Lemma 35. *(Distribution of $N_{i,j}$) We have that*

$$\mathbb{P}\left(\exists(i,j)\in\mathcal{A}\ :\ N_{i,j}^{(\mathbf{x})}\neq 0\right)=\widetilde{\mathcal{O}}\left(\max_{(i,j)\in\mathcal{A}}\frac{\binom{n-s}{i}\binom{s}{j}}{2^{n-k+k_{\text{aux}}}}\right)$$

and where all the other quantities and distributions are defined in Corollary 12.

Note that in a sense proposition is trivially true by supposing that the Poisson Model 5 holds. Hence, one does not necessarily have to read this section to be convinced of our results. First, using the union bound and bounding the relevant probability by its expectation we get that

Fact 27. *We have that*

$$\mathbb{P}\left(\exists(i,j)\in\mathcal{A}\ :\ N_{i,j}\neq 0\right)=\mathcal{O}\left(n^2\max_{(i,j)\in\mathcal{A}}\mathbb{E}\left(N_{i,j}\right)\right)$$

Lemma 36. *We have that*

$$\mathbb{E}_{\mathcal{C},\mathcal{C}_{\text{aux}},\mathbf{x}}\left(N_{i,j}\right)=\overline{N_j^{(\mathcal{C}_{\text{aux}}^{\perp})}}\ \overline{N_i^{(\mathcal{C}^{\mathscr{N}})}}$$

where

$$\overline{N_j^{(\mathcal{C}_{\text{aux}}^{\perp})}}\overset{def}{=}\mathbb{E}_{\mathcal{C}_{\text{aux}},\mathbf{x}}\left(N_j\left(\mathcal{C}_{\text{aux}}^{\perp}+\mathbf{x}\right)\right)\quad,\quad\overline{N_i^{(\mathcal{C}^{\mathscr{N}})}}\overset{def}{=}\mathbb{E}_{\mathcal{C}}\left(N_i\left(\mathcal{C}^{\mathscr{N}}+\mathbf{v}\right)\right)$$

and where $\mathbf{v}\sim\mathcal{U}\left(\mathbb{F}_2^{n-s}\right)$ and where we recall that

$$N_{i,j}\overset{def}{=}\left|\left\{\left(\mathbf{r},\mathbf{c}^{\mathscr{N}}\right)\in(\mathbf{x}+\mathcal{C}_{\text{aux}}^{\perp})\times\mathcal{C}^{\mathscr{N}}\ :|\mathbf{r}|=j\ \textit{and}\ \left|(\mathbf{r}+\mathbf{e}_{\mathscr{P}})\,\mathbf{R}+\mathbf{e}_{\mathscr{N}}+\mathbf{c}^{\mathscr{N}}\right|=i\right\}\right|$$

Proof. Rewritting

$$N_{i,j} = \sum_{\mathbf{r} \in \mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x} \bigcap \mathcal{S}_j^s} N_i \left((\mathbf{r} + \mathbf{e}_{\mathscr{P}}) \mathbf{R} + \mathbf{e}_{\mathscr{N}} + \mathcal{C}^{\mathscr{N}} \right)$$

we see that

$$\begin{aligned} N_{i,j} &= \sum_{\mathbf{z} \in \mathcal{S}_j^s} N_i \left((\mathbf{z} + \mathbf{e}_{\mathscr{P}}) \mathbf{R} + \mathbf{e}_{\mathscr{N}} + \mathcal{C}^{\mathscr{N}} \right) \mathbb{1}_{\text{“}\mathbf{z} \in \mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x}''} \\ &= \sum_{\mathbf{z} \in \mathcal{S}_j^s, \mathbf{w} \in \mathcal{S}_i^{n-s}} \mathbb{1}_{\text{“}\mathbf{w} \in (\mathbf{z} + \mathbf{e}_{\mathscr{P}}) \mathbf{R} + \mathbf{e}_{\mathscr{N}} + \mathcal{C}^{\mathscr{N}}\text{”}} \mathbb{1}_{\text{“}\mathbf{z} \in \mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x}''} \end{aligned}$$

We deduce that,

$$\begin{aligned} \mathbb{E}_{\mathcal{C}, \mathcal{C}_{\text{aux}}, \mathbf{x}} \left(N_{i,j} \right) &= \sum_{\mathbf{z} \in \mathcal{S}_j^s, \mathbf{w} \in \mathcal{S}_i^{n-s}} \mathbb{P} \left(\mathbf{w} \in (\mathbf{z} + \mathbf{e}_{\mathscr{P}}) \mathbf{R} + \mathbf{e}_{\mathscr{N}} + \mathcal{C}^{\mathscr{N}},\ \mathbf{z} \in \mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x} \right) \\ &= \sum_{\mathbf{z} \in \mathcal{S}_j^s, \mathbf{w} \in \mathcal{S}_i^{n-s}} \mathbb{P} \left(\mathbf{w} \in (\mathbf{z} + \mathbf{e}_{\mathscr{P}}) \mathbf{R} + \mathbf{e}_{\mathscr{N}} + \mathcal{C}^{\mathscr{N}} \mid \mathbf{z} \in \mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x} \right) \mathbb{P} \left(\mathbf{z} \in \mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x} \right) \end{aligned}$$

Now, as $\mathbf{x} \sim \mathcal{U} \left(\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{e}_{\mathscr{P}} \right)$ we necessarily have that if $\mathbf{z} \in \mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x}$ then $\mathbf{z} \neq \mathbf{e}_{\mathscr{P}}$ and thus $\mathbf{z} + \mathbf{e}_{\mathscr{P}} \neq \mathbf{0}$. We can show in the same manner as in Lemma 32 (which is not directly applicable as the distribution of $\mathcal{C}$ slightly differs) that we have that $\mathbf{R} \sim \mathcal{U} \left(\mathbb{F}_2^{s \times (n-s)} \right)$ and thus $(\mathbf{z} + \mathbf{e}_{\mathscr{P}}) \mathbf{R} + \mathbf{e}_{\mathscr{N}} \sim \mathcal{U} \left(\mathbb{F}_2^{n-s} \right)$. Denoting, as in the lemma statement $\mathbf{v} \sim \mathcal{U} \left(\mathbb{F}_2^{n-s} \right)$ we get

$$\begin{aligned} \mathbb{E}_{\mathcal{C}, \mathcal{C}_{\text{aux}}, \mathbf{x}} \left(N_{i,j} \right) &= \sum_{\mathbf{z} \in \mathcal{S}_j^s, \mathbf{w} \in \mathcal{S}_i^{n-s}} \mathbb{P} \left(\mathbf{w} \in \mathcal{C}^{\mathscr{N}} + \mathbf{v} \right) \mathbb{P} \left(\mathbf{z} \in \mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x} \right) \\ &= \left(\sum_{\mathbf{z} \in \mathcal{S}_j^s} \mathbb{P} \left(\mathbf{z} \in \mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x} \right) \right) \left(\sum_{\mathbf{w} \in \mathcal{S}_i^{n-s}} \mathbb{P} \left(\mathbf{w} \in \mathcal{C}^{\mathscr{N}} + \mathbf{v} \right) \right) \\ &= \overline{N_i^{(\mathcal{C}^{\mathscr{N}})}}\ \overline{N_j^{(\mathcal{C}_{\text{aux}}^{\perp})}} \end{aligned}$$

□

Lemma 37. *Using the same notations as Lemma 36 we have that*

$$\overline{N_i^{(\mathcal{C}^{\mathscr{N}})}} = \frac{\binom{n-s}{i}}{2^{n-k}} \tag{6.48}$$

$$\overline{N_j^{(\mathcal{C}_{\text{aux}}^{\perp})}} \leqslant \frac{\binom{s}{j}}{2^{k_{\text{aux}}}} \left(1 + \mathcal{O} \left(2^{-k_{\text{aux}}} \right) \right) \tag{6.49}$$

Proof. The equality concerning $\overline{N_i^{(\mathcal{C}^{\mathscr{N}})}}$ is clear. Let us now show Equation (6.49). Recall that

$$\overline{N_j^{(\mathcal{C}_{\text{aux}}^{\perp})}} \overset{\text{def}}{=} \mathbb{E}_{\mathcal{C}_{\text{aux}}, \mathbf{x}} \left(N_j \left(\mathcal{C}_{\text{aux}}^{\perp} + \mathbf{x} \right) \right)$$

where $\mathbf{x}$ is taken uniformly at random in $\mathbb{F}_2^s \setminus \{\mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\}$. Let $\mathbf{z} \sim \mathcal{U}(\mathbb{F}_2^s)$. By the law of total expectation we have that

$$\begin{aligned}
\mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^\perp + \mathbf{z}\right)\right) &= \mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^\perp + \mathbf{z}\right) \,\middle|\, \mathbf{z} \in \mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\right)\mathbb{P}\left(\mathbf{z} \in \mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\right) \\
&\quad + \mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^\perp + \mathbf{z}\right) \,\middle|\, \mathbf{z} \notin \mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\right)\mathbb{P}\left(\mathbf{z} \notin \mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\right) \\
&\geqslant \mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^\perp + \mathbf{z}\right) \,\middle|\, \mathbf{z} \notin \mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\right)\mathbb{P}\left(\mathbf{z} \notin \mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\right) \\
&= \mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^\perp + \mathbf{x}\right)\right)\mathbb{P}\left(\mathbf{z} \notin \mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\right)
\end{aligned}$$

Using the fact that

$$\mathbb{P}\left(\mathbf{z} \notin \mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\right) = \frac{2^s - 2^{s-k_{\text{aux}}}}{2^s}$$

we can write that

$$\begin{aligned}
\mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^\perp + \mathbf{x}\right)\right) &\leqslant \frac{\mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^\perp + \mathbf{z}\right)\right)}{\mathbb{P}\left(\mathbf{z} \notin \mathcal{C}_{\text{aux}}^\perp + \mathbf{e}_{\mathscr{P}}\right)} \\
&= \frac{\mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^\perp + \mathbf{z}\right)\right)}{1 - 2^{-k_{\text{aux}}}} \\
&= \mathbb{E}\left(N_j\left(\mathcal{C}_{\text{aux}}^\perp + \mathbf{z}\right)\right)\left(1 + \mathcal{O}\left(2^{-k_{\text{aux}}}\right)\right) \\
&= \frac{\binom{s}{j}}{2^{k_{\text{aux}}}}\left(1 + \mathcal{O}\left(2^{-k_{\text{aux}}}\right)\right).
\end{aligned}$$

This concludes the proof. □

6.5.3.3.2 Expected number of false candidates Here we prove Lemma 34 which we restate here.

Lemma 34. *For any $k, t, w, s, u, k_{\text{aux}}, t_{\text{aux}}$ implicit functions of $n \in \mathbb{N}$ we have that*

$$\mathbb{E}\left(\left|\mathcal{S}'\right|\right) = \mathcal{O}\left(E + \frac{\binom{s}{t-u}}{2^{k_{\text{aux}}}}\right)\max\left(1,\ \frac{2^{k_{\text{aux}}}}{\binom{t-u}{s}}\right)$$

where we defined

$$E \overset{def}{=} \binom{s}{t-u}\mathbb{P}\left(F^{\mathcal{L}}(\mathbf{z}) > T\right)$$

and where $\mathbf{z} \sim \mathcal{U}\left(\mathbb{F}_2^{k_{\text{aux}}} \setminus \{\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^\intercal\}\right)$ and where the other quantities and distributions are defined in Proposition 54.

We make the proof with the two following lemmas.

Lemma 38. *We have that*

$$\mathbb{E}\left(\left|\mathcal{S}'\right|\right) \leqslant \binom{s}{t-u}\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{a}\mathbf{G}_{\text{aux}}^\intercal\right) > T\right)$$

where $\mathbf{a} \sim \mathcal{U}\left(\mathcal{S}_{t-u}^s\right)$.

Proof. Recall that

$$\mathcal{S}' = \bigcup_{\mathbf{z}\in\mathcal{S}} \mathrm{Dec}^{\mathrm{juxt}}\left(\mathcal{C}_{\mathrm{aux}}^{\perp}, \mathbf{z}, t-u\right).$$

By definition and because $b = 1$ we have that

$$\mathrm{Dec}^{\mathrm{juxt}}\left(\mathcal{C}_{\mathrm{aux}}^{\perp}, \mathbf{z}, t-u\right) = \{\mathbf{x} \in \mathcal{S}_{t-u}^{s} \,:\, \mathbf{x}\mathbf{G}_{\mathrm{aux}}^{\intercal} = \mathbf{z}\}.$$

As such

$$\mathcal{S}' = \bigcup_{\mathbf{x}\in\mathcal{S}_{t-u}^{s}} \mathbf{1}_{\mathbf{x}\mathbf{G}_{\mathrm{aux}}^{\intercal}\in\mathcal{S}}$$

and thus

$$\left|\mathcal{S}'\right| = \sum_{\mathbf{x}\in\mathcal{S}_{t-u}^{s}} \mathbf{1}_{\mathbf{x}\mathbf{G}_{\mathrm{aux}}^{\intercal}\in\mathcal{S}}.$$

By linearity of the expected value we have

$$\begin{aligned}
\mathbb{E}\left(\left|\mathcal{S}'\right|\right) &= \sum_{\mathbf{x}\in\mathcal{S}_{t-u}^{s}} \mathbb{P}\left(\mathbf{x}\mathbf{G}_{\mathrm{aux}}^{\intercal} \in \mathcal{S}\right) \\
&= \sum_{\mathbf{x}\in\mathcal{S}_{t-u}^{s}} \mathbb{P}\left(\mathbf{a}\mathbf{G}_{\mathrm{aux}}^{\intercal} \in \mathcal{S} \mid \mathbf{a} = \mathbf{x}\right) \\
&= \sum_{\mathbf{x}\in\mathcal{S}_{t-u}^{s}} \mathbb{P}\left(\mathbf{a}\mathbf{G}_{\mathrm{aux}}^{\intercal} \in \mathcal{S} \mid \mathbf{a} = \mathbf{x}\right) \frac{\mathbb{P}\left(\mathbf{a} = \mathbf{x}\right)}{\mathbb{P}\left(\mathbf{a} = \mathbf{x}\right)} \\
&= \binom{s}{t-u} \mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{a}\mathbf{G}_{\mathrm{aux}}^{\intercal}\right) > T\right)
\end{aligned}$$

where in the last line we used the law of total probability and the fact that $\mathbb{P}\left(\mathbf{a} = \mathbf{x}\right) = 1/\binom{s}{t-u}$. □

Let us now relate the probability $\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{a}\mathbf{G}_{\mathrm{aux}}^{\intercal}\right) > T\right)$ to our distribution of interest, namely we get

Lemma 39. *We have that*

$$\mathbb{P}_{\mathcal{C},\mathcal{C}_{\mathrm{aux}},\mathbf{a}}\left(F^{\mathcal{L}}\left(\mathbf{a}\mathbf{G}_{\mathrm{aux}}^{\intercal}\right) > T\right) = \mathcal{O}\left(\mathbb{P}_{\mathcal{C},\mathcal{C}_{\mathrm{aux}},\mathbf{r}}\left(F^{\mathcal{L}}\left(\mathbf{r}\right) > T\right) \max\left(1,\ \frac{2^{k_{\mathrm{aux}}}}{\binom{t-u}{s}}\right)\right)$$

where $\mathbf{a} \sim \mathcal{U}\left(\mathcal{S}_{t-u}^{s}\right)$ *and* $\mathbf{r} \sim \mathcal{U}\left(\mathbb{F}_2^{k_{\mathrm{aux}}}\right)$.

Proof. Because $\mathbf{G}_{\mathrm{aux}}$ is distributed uniformly at random among matrices of $\mathbb{F}_2^{k_{\mathrm{aux}}\times s}$ of rank k_{aux} we have that $\mathbf{a}\mathbf{G}_{\mathrm{aux}}^{\intercal}$ has the same distribution as $\mathbf{r}$ conditioned on the event that there exists $\mathbf{v} \in \mathcal{S}_{t-u}^{s}$ such that $\mathbf{v}\mathbf{G}_{\mathrm{aux}}^{\intercal} = \mathbf{r}$. Thus, by using Bayes's theorem and bounding by the relevant term we get

$$\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{a}\mathbf{G}_{\mathrm{aux}}^{\intercal}\right) > T\right) \leqslant \frac{\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{r}\right) > T\right)}{\mathbb{P}\left(\exists \mathbf{v} \in \mathcal{S}_{t-u}^{s} \,:\, \mathbf{v}\mathbf{G}_{\mathrm{aux}}^{\intercal} = \mathbf{r}\right)}.$$

Let us upper bound the denominator. If $t-u=0$ we can conclude, else, let us suppose that $t-u\neq 0$. Let $\mathbf{g},\mathbf{b}\in\mathcal{S}^s_{t-u}$ such that $\mathbf{g}\neq\mathbf{b}$. From Proposition 5, and by denoting

$$\begin{aligned}p_1 &\stackrel{\text{def}}{=} \mathbb{P}\left(\mathbf{b}\mathbf{G}^\intercal_{\text{aux}}=\mathbf{r}\right),\\ p_2 &\stackrel{\text{def}}{=} \mathbb{P}\left(\mathbf{b}\mathbf{G}^\intercal_{\text{aux}}=\mathbf{r},\ \mathbf{g}\mathbf{G}^\intercal_{\text{aux}}=\mathbf{r}\right),\end{aligned}$$

we have that

$$\begin{aligned}p_1 &= \Theta\left(\frac{1}{2^{k_{\text{aux}}}}\right),\\ p_2 &= \Theta\left(p_1^2\right).\end{aligned}$$

This allows us to use Bonferroni like inequalities to upper bound our probability. Namely, from [dC97] we get that if $A_{\mathbf{x}}$ is a finite set of event indexed by $\mathbf{x}\in X$ we have that

$$\mathbb{P}\left(\bigcup_{\mathbf{x}\in X}A_{\mathbf{x}}\right)\geqslant\sum_{\mathbf{x}\in X}\frac{\mathbb{P}\left(A_{\mathbf{x}}\right)^2}{\sum_{\mathbf{u}\in X}\mathbb{P}\left(A_{\mathbf{x}}\cap A_{\mathbf{u}}\right)}.$$

Applying it in our case directly yields that

$$\begin{aligned}\mathbb{P}\left(\exists\mathbf{v}\in\mathcal{S}^s_{t-u}\ :\ \mathbf{v}\mathbf{G}^\intercal_{\text{aux}}=\mathbf{r}\right) &\geqslant \frac{\binom{t-u}{s}p_1^2}{p_1+\binom{t-u}{s}p_2}\\ &\geqslant \frac{1}{\frac{1}{\binom{s}{t-u}p_1}+\frac{p_2}{p_1^2}}\\ &= \frac{1}{\frac{1}{\binom{s}{t-u}p_1}+\Theta\left(1\right)}\\ &= \Omega\left(\min\left(\binom{s}{t-u}p_1,\ 1\right)\right).\end{aligned}$$

Finally, this allows us to conclude that

$$\frac{1}{\mathbb{P}\left(\exists\mathbf{v}\in\mathcal{S}^s_{t-u}\ :\ \mathbf{v}\mathbf{G}^\intercal_{\text{aux}}=\mathbf{r}\right)}=\mathcal{O}\left(\max\left(\frac{2^{k_{\text{aux}}}}{\binom{s}{t-u}},\ 1\right)\right)$$

which concludes our proof. □

We can now prove our proposition.

Proof of Lemma 34. Applying successively Lemma 38 and Lemma 39 and we get that

$$\mathbb{E}\left(\left|\mathcal{S}'\right|\right)=\mathcal{O}\left(\binom{s}{t-u}\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{r}\right)>T\right)\max\left(1,\ \frac{2^{k_{\text{aux}}}}{\binom{s}{t-u}}\right)\right)$$

where we recall that $\mathbf{r}\sim\mathcal{U}\left(\mathbb{F}_2^{k_{\text{aux}}}\right)$. By applying the law of total probabilities and bounding the irrelevant terms by 1 we get

$$\begin{aligned}\mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{r}\right)>T\right) &\leqslant \mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{r}\right)>T\ \middle|\ \mathbf{r}\in\mathbb{F}_2^{k_{\text{aux}}}\setminus\{\mathbf{e}_{\mathscr{P}}\mathbf{G}^\intercal_{\text{aux}}\}\right)+\mathbb{P}\left(\mathbf{r}=\mathbf{e}_{\mathscr{P}}\mathbf{G}^\intercal_{\text{aux}}\right)\\ &= \mathbb{P}\left(F^{\mathcal{L}}\left(\mathbf{z}\right)>T\right)+\frac{1}{2^{k_{\text{aux}}}}\end{aligned}$$

where in the last line we used the fact that $\mathbf{z} \sim \mathcal{U}\left(\mathbb{F}_2^{k_{\text{aux}}} \setminus \{\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\}\right)$. This concludes the proof. □

Chapter 7

Fully provable dual attacks

Summary

We devise a variant of double-RLPN (and RLPN) which has, up to polynomial factors, the same performance as double-RLPN but which we can fully analyze without using any conjecture whatsoever. The analysis of our variant only relies on the second-order concentration property of the score function. Apart from simplifying the analysis it also simplifies the complexity statement of double-RLPN: in the last chapter the statement had a parasitical term accounting for the cost of checking the so called false candidates, here we do not have this. The cost of our algorithm really is, up to polynomial factors, the cost of computing the N short dual vectors and the cost of the FFT and it succeeded when N is of the order of $1/\varepsilon^2$ where ε is the bias of the LPN samples. This chapter is not published yet.

Contents

7.1 Introduction

Recall that our analysis of RLPN and double-RLPN needed some conjecture about the exponential tail bounds of some score function related to the LPN samples we compute. Our goal here is to devise a slight variant of double-RLPN which achieves the same performances, up to polynomials factor, as the original algorithm but can be fully proven without using any conjecture. Let us forget about double-RLPN for now and talk only about RLPN, everything we say can be adapted to the more complex setting of double-RLPN.

7.1.1 Technical difficulty in the analysis coming from RLPN

Recall that in RLPN we are given a noisy codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{C}$ and $\mathbf{e} \in \mathcal{S}_t^n$. We choose two complementary subsets $\mathscr{P}$ and $\mathscr{N}$ of the support $[\![1, n]\!]$ and compute many dual vectors $\mathbf{h} \in \mathcal{C}^\perp$ which are of low Hamming weight on $\mathscr{N}$, each yielding the following LPN sample

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{e}_{\mathscr{P}}, \mathbf{h}_{\mathscr{P}} \rangle + \langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle .$$

Finally, we compute a score function $F^{\mathcal{L}}$ which gives us how likely a certain vector $\mathbf{x}$ is the secret $\mathbf{e}_{\mathscr{P}}$ of the LPN samples by encoding how biased the quantity of interest $\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle$ is, namely (we have added explicitly the dependency in $\mathbf{y}$)

$$F_{\mathbf{y}}^{\mathcal{L}}(\mathbf{x}) = \sum_{\mathbf{h}} (-1)^{\langle \mathbf{y}, \mathbf{h} \rangle - \langle \mathbf{x}, \mathbf{h}_{\mathscr{P}} \rangle}.$$

Basically we are able to show, using second order concentration bounds on the score function, i.e. bounds of the type

$$\mathbb{P}\left(\left| F_{\mathbf{y}}^{\mathcal{L}}(\mathbf{x}) - \mathbb{E}\left(F_{\mathbf{y}}^{\mathcal{L}}(\mathbf{x}) \right) \right| \leqslant \operatorname{poly}(n) \sqrt{\mathbf{Var}\left(F_{\mathbf{y}}^{\mathcal{L}}(\mathbf{x}) \right)} \right) \geqslant 1 - 1/\operatorname{poly}(n), \tag{7.1}$$

that we can distinguish a specific $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ from $\mathbf{e}_{\mathscr{P}}$ with probability $1 - 1/\operatorname{poly}(n)$ as long as the number of dual vectors computed are of the order $\operatorname{poly}(n)/\varepsilon^2$ where ε is a bias of the noise term $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$. However, having in mind that we want to stay polynomially tight to this condition, these bounds fall short to say anything about our ability to distinguish $\mathbf{e}_{\mathscr{P}}$ from all of the $\mathbf{x}$ which are different from $\mathbf{e}_{\mathscr{P}}$ as this space is exponentially big ($|\mathscr{P}|$ is linear in t), this is where we needed a conjecture about the exponential behavior of the score function.

7.1.2 The technique

We show here that we can find a workaround so that we only need our second order concentration bound to analysis our new algorithm. From a high level point of view our algorithm computes for each $\mathbf{x} \in \mathbb{F}_2^{|\mathscr{P}|}$ an associated guess for the value of $\mathbf{e}_{\mathscr{N}}$ and then tests this guess. The crucial point being that regardless if $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ or $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ we can then test a guess in polynomial time simply by verifying that $\mathbf{x}$ concatenated with its associated guess for $\mathbf{e}_{\mathscr{N}}$ is indeed the solution to our decoding problem by say verifying that it is of good weight t and has the right syndrome. Our algorithm will be correct at the condition that when $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ the guess on $\mathbf{e}_{\mathscr{N}}$ is valid, and the other cases are consequently completely unimportant. Let us explain how this guess is done. For $\mathbf{x} \in \mathbb{F}_2^{|\mathscr{P}|}$ we make the guess for $\mathbf{e}_{\mathscr{N}}$ one position at a time as follows. The i'th position of $\mathbf{e}_{\mathscr{N}}$, namely $\mathbf{e}_{\mathscr{N}_i}$ is guessed by constructing $\mathbf{y}^{(i)}$ which

is $\mathbf{y}$ but where we flipped $\mathbf{y}_{\mathscr{N}_i}$, this has the effect of flipping $\mathbf{e}_{\mathscr{N}_i}$. Now we for this specific $\mathbf{x}$ make the guess $\mathbf{e}_{\mathscr{N}_i} = 1$ if $F^{\mathcal{L}}_{\mathbf{y}^{(i)}}(\mathbf{x}) > F^{\mathcal{L}}_{\mathbf{y}}(\mathbf{x})$ else we make the guess that $\mathbf{e}_{\mathscr{N}_i} = 0$. Clearly when $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ then

$$F^{\mathcal{L}}_{\mathbf{y}^{(i)}}(\mathbf{e}_{\mathscr{P}}) = \sum_{\mathbf{h}} (-1)^{\langle \mathbf{e}_{\mathscr{N}} + \delta_i, \mathbf{h}_{\mathscr{N}} \rangle}$$

which is expected to be bigger than $F^{\mathcal{L}}_{\mathbf{y}}(\mathbf{x})$ if we flipped an erroneous position as in that case the weight of $\mathbf{e}_{\mathscr{N}} + \delta_i$ is decreased compared to the weight of $\mathbf{e}_{\mathscr{N}}$, hence our decision rationale. Crucially when $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$, our second order concentration bounds are sufficient here as for each position i we compare only two distributions, the one related to $\mathbf{e}_{\mathscr{N}_i}$ being 0 and the one for $\mathbf{e}_{\mathscr{N}_i}$ being 1. Of course, we must apply these bounds for each $i \in [\![1, n]\!]$, but this only incurs a polynomial loss compared to stronger bounds. We make no claim whatsoever regarding the other cases corresponding to $\mathbf{x} \neq \mathbf{e}_{\mathscr{P}}$ as they will be naturally discarded by our checking phase.

7.1.3 Why we have the same performance

While it is clear that second order concentration bounds are the only tool required to prove the previously introduced algorithm, one could argue that we have somehow lost something as the distribution we are trying to distinguish could be closer compared to the original distribution we were trying to distinguish in RLPN. However, this is essentially not the case. Recall that in RLPN wet bet that $|\mathbf{e}_{\mathscr{N}}| = u$ and make our analysis in the setting where all dual vectors of weight w on $\mathscr{N}$ are computed. In this case the bias of the LPN samples in RLPN is approximately $\delta_w^{(n-s)}(u)$ where $|\mathscr{N}| = n - s$ and the conclusion is that we needed

$$N = \frac{\mathrm{poly}(n)}{\left(\delta_w^{(n-s)}(u)\right)^2}$$

dual vectors for our algorithm to work as expected. The intuition boiled down to applying Eq. (7.1) using the fact that our score function $F^{\mathcal{L}}(\mathbf{x})$ had a variance of N and was expected to be $N\delta_w^{(n-s)}(u)$ when $\mathbf{x} = \mathbf{e}_{\mathscr{P}}$ and 0 otherwise. Simplifying a bit, here in our case, the distribution we are comparing are somewhat slightly closer as basically we are trying to distinguish between two variables with expected value $N\delta_w^{(n-s)}(u-1)$ and $N\delta_w^{(n-s)}(u+1)$, but their distances are comparable as essentially in the non-oscillatory regime, $\delta_w^{(n)}(t + \mathcal{O}(1))$ is polynomially relatable to $\delta_w^{(n)}(t)$, namely

Lemma 40 (Difference of bias)**.** *Let $n, w, t \in \mathbb{N}$, we have that*

$$\delta_w^{(n)}(t-1) - \delta_w^{(n)}(t) = 2\frac{w}{n}\delta_{w-1}^{(n-1)}(t-1).$$

As such clearly it is sufficient for our algorithm to compute, as before, $\mathrm{poly}(n) / \left(\delta_w^{(n-s)}(u)\right)^2$ dual vectors to be correct.

7.2 Fully provable double-RLPN

Let us now reuse the idea presented in the introduction but adapt it to double-RLPN. We basically need 3 ingredients

1. A procedure DOUBLE-RLPN-SCOREFUNCTION which given a vector $\mathbf{y}$ computes the double-RLPN score function associated to the LPN problem generated by $\mathbf{y}$ (and some given subpart $\mathscr{P}$ and $\mathscr{N}$ of the support).

2. A guessing procedure which will, for each vector in the secret space of the underlying LPN problem make a guess for the value of $\mathbf{e}_{\mathscr{N}}$ by calling the previous procedure multiple times.

3. A checking procedure which will check each guess for $\mathbf{e}_{\mathscr{N}}$ and will try to reconstruct the whole error vector $\mathbf{e}$. Importantly each guess is tested in polynomial time.

Our full algorithm is in fact basically choosing $\mathscr{P}$ and $\mathscr{N}$ and calling the guessing procedure and then the checking procedure. As in, double-RLPN we will iterate it a certain number N_{iter} of times in order to make sure that there exists an iteration such that $\mathbf{e}_{\mathscr{N}}$ is of sufficiently low weight. The whole algorithm is presented at the end of the section.

7.2.1 Computing the score function

Here basically we reuse the core part of the double-RLPN algorithm which serves to compute the score function. We break it down into two procedures, one that computes the small weight dual vectors and decode them into an auxiliary code, and one that from this set of decoded dual vectors and a given vector $\mathbf{y}$ computes the associated score function. The first one DECODED-DUAL-VECTOR-DOUBLE-RLPN$(\mathcal{C}, \mathscr{P}, \mathscr{N}, \mathcal{C}_{\text{aux}})$ takes as input a linear code $\mathcal{C}$ of length n, $\mathscr{P}$ and $\mathscr{N}$ two complementary subparts of $[\![1, n]\!]$ and $\mathcal{C}_{\text{aux}}$ an $[s, k_{\text{aux}}]$ linear code of generator matrix $\mathbf{G}_{\text{aux}}$. It outputs a set

$$\mathscr{H} \subset \{(\mathbf{h}, \mathbf{m}_{\text{aux}})\, \mathcal{C}^{\perp} \times \mathbb{F}_2^{k_{\text{aux}}} \ : \ |\mathbf{h}_{\mathscr{N}}| = w \text{ and } \mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} + \mathbf{h}_{\mathscr{P}} \in \mathcal{S}^s_{t_{\text{aux}}}\}$$

of decoded dual vectors as in RLPN. It is given in Algorithm 25.

Algorithm 25

Name: DECODED-DUAL-VECTOR-DOUBLE-RLPN$(\mathcal{C}, \mathscr{P}, \mathscr{N}, \mathcal{C}_{\text{aux}})$
Input: $\mathcal{C}$, $\mathscr{P}$, $\mathscr{N}$, $\mathcal{C}_{\text{aux}}$
Parameter: $w, s, k_{\text{aux}}, t_{\text{aux}}$

$\mathscr{W}_{\mathscr{N}} \leftarrow$ COMPUTE-DUAL-VECTORS$(\mathcal{C}^{\mathscr{N}}\ ;\ w)$ ▷ *Returns a subset of dual vectors of $\mathcal{C}^{\mathscr{N}}$ which are of weight w*
$\mathscr{W} \leftarrow \{$LIFT$(\mathcal{C}^{\perp}, \mathscr{N},\ \mathbf{h}_{\mathscr{N}})$ for $\mathbf{h}_{\mathscr{N}} \in \mathscr{W}_{\mathscr{N}}\}$ ▷ *Lift those dual vectors $\mathbf{h}_{\mathscr{N}} \in \left(\mathcal{C}^{\mathscr{N}}\right)^{\perp}$ to make them dual vectors $\mathbf{h}$ of $\mathcal{C}$ of low weight w on $\mathscr{N}$*
$\mathcal{C}_{\text{aux}} \leftarrow$ SAMPLE-AUXILARY-CODE$(\mathcal{F})$
$\mathbf{G}_{\text{aux}} \leftarrow \mathbf{G}\,(\mathcal{C}_{\text{aux}})$
$\mathscr{H} \leftarrow \emptyset$
for $\mathbf{h} \in \mathscr{W}$ **do**
    $\mathcal{E} \leftarrow$ DECODE-AUXILARY$(\mathcal{C}_{\text{aux}}, \mathbf{h}_{\mathscr{P}})$ ▷ *Returns a set of error of small weight $\mathbf{e}_{\text{aux}} \in \mathcal{S}^s_{t_{\text{aux}}}$ $s.t$ $\mathbf{h}_{\mathscr{P}} - \mathbf{e}_{\text{aux}} \in \mathcal{C}_{\text{aux}}$*
    for $\mathbf{e}_{\text{aux}} \in \mathcal{E}$ **do**
        $\mathbf{m}_{\text{aux}}$ is the unique vector such that $\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} = \mathbf{h}_{\mathscr{P}} - \mathbf{e}_{\text{aux}}$
        $\mathscr{H}$.APPEND$((\mathbf{h},\ \mathbf{m}_{\text{aux}}))$ ▷ *The set of decoded dual vectors*
return $\mathscr{H}$

From the set $\mathscr{H}$ of decoded dual vectors and a vector $\mathbf{y} \in \mathbb{F}_2^n$ the procedure Score-function($\mathbf{y}$, $\mathscr{H}$) computes its associated score function.

Definition 44. *For all $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ we define*

$$F_{\mathbf{y}}^{\mathcal{L}}(\mathbf{z}) \stackrel{def}{=} \sum_{(\mathbf{h},\mathbf{m}_{\text{aux}})\in\mathscr{H}} (-1)^{\langle \mathbf{y},\mathbf{h}\rangle - \langle \mathbf{m}_{\text{aux}},\mathbf{z}\rangle}.$$

The procedure to compute this score is recalled in Algorithm 26.

Algorithm 26

Name: Score-function($\mathbf{y}$, $\mathscr{H}$)
Input: $\mathbf{y}$, $\mathscr{H}$
1: $\mathcal{L} \leftarrow$ the list of samples $(\mathbf{m}_{\text{aux}}, \langle \mathbf{y}, \mathbf{h}\rangle)$ for $(\mathbf{h}, \mathbf{m}_{\text{aux}}) \in \mathscr{H}$
2: $F_{\mathbf{y}}^{\mathcal{L}} \leftarrow$ FFT-LPN-Solver($\mathcal{L}$) ▷ *The procedure is defined in Algorithm 10*
3: **return** $F_{\mathbf{y}}^{\mathcal{L}}$

7.2.2 Guessing phase

Recalling that $\mathbb{F}_2^{k_{\text{aux}}}$ the space in which lives the secret of the underlying LPN problem $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}$, our goal here is for each $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$, to make a guess on the value of $\mathbf{e}_{\mathscr{N}}$, we make this guess bit by bit. We will exploit the fact that when $\mathbf{z}$ is secret of our LPN samples, namely $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ we have that

Fact 28.

$$F_{\mathbf{y}}^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}) = \sum_{(\mathbf{h},\,\mathbf{e}_{\text{aux}})\in\mathscr{H}} (-1)^{\langle \mathbf{e}_{\mathscr{N}},\mathbf{h}_{\mathscr{N}}\rangle + \langle \mathbf{e}_{\mathscr{P}},\mathbf{e}_{\text{aux}}\rangle}.$$

Observe that flipping the positions of the received word $\mathbf{y}$ is exactly flipping the positions of the error vector $\mathbf{e}$. Thus, we have an impact on the score function by flipping the i'th bit of $\mathbf{y}$ in $\mathscr{N}$.

Definition 45. *For any $i \in [\![1, n]\!]$ we denote by $\mathbf{y}^{(i)} \in \mathbb{F}_2^n$ the vector defined as $\left(\mathbf{y}^{(i)}\right)_{\mathscr{P}} \stackrel{def}{=} \mathbf{y}_{\mathscr{P}}$ and $\left(\mathbf{y}^{(i)}\right)_{\mathscr{N}} \stackrel{def}{=} \mathbf{y}_{\mathscr{N}} + \delta_i$ where $\delta_i \in \mathbb{F}_2^{n-s}$ is the vector which is zero everywhere except on its i'th coordinate.*

Fact 29.

$$F_{\mathbf{y}^{(i)}}^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}) = \sum_{(\mathbf{h},\,\mathbf{e}_{\text{aux}})\in\mathscr{H}} (-1)^{\langle \mathbf{e}_{\mathscr{N}}+\delta_i,\mathbf{h}_{\mathscr{N}}\rangle + \langle \mathbf{e}_{\mathscr{P}},\mathbf{e}_{\text{aux}}\rangle}.$$

It is readily seen that this last quantity is expected to be bigger than original $F_{\mathbf{y}}^{\mathcal{L}}(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal})$ if we flipped an erroneous position, namely if $(\mathbf{e}_{\mathscr{N}})_i = 1$. This is our decision rationale for our guess on $\mathbf{e}_{\mathscr{N}}$.

7.2.2.1 Algorithm

We compute for each $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$ and associated guess for $\mathbf{e}_{\mathscr{N}}$, call it $G(\mathbf{z}) \in \mathbb{F}_2^{n-s}$ by successively flipping the bits of $\mathbf{y}$ as described above. More precisely the i'th bit of $G(\mathbf{z})$ is determined by computing a reference score function $F_{\mathbf{y}}^{\mathcal{L}}$ and computing the flipped score function $F_{\mathbf{y}^{(i)}}^{\mathcal{L}}$ and comparing them.

Definition 46. *For each* $\mathbf{x} \in \mathbb{F}_2^{k_{\text{aux}}}$ *we denote by* $G(\mathbf{z})$ *the vector of* $\mathbb{F}_2^{n-s}$ *whose* i*'th coordinate is equal to*

$$G(\mathbf{z})_i \stackrel{def}{=} \begin{cases} 1 & \text{if} \quad F^{\mathcal{L}}_{\mathbf{y}^{(i)}}(\mathbf{z}) > F^{\mathcal{L}}_{\mathbf{y}}(\mathbf{z}) \\ 0 & \text{else.} \end{cases}.$$

We call $G(\mathbf{z})$ *the guess for* $\mathbf{e}_{\mathscr{N}}$ *related to* $\mathbf{z}$*.*

We then store these guesses in a set $\mathcal{G} = \{(\mathbf{z},\ G(\mathbf{z}))\ :\ \mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}\}$ and outputs it.

Algorithm 27

Name: Guessing-$\mathbf{e}_{\mathscr{N}}(\mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N})$
Input: $\mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N}$
1: $\mathscr{H} \leftarrow$ Decoded-dual-vector-double-RLPN$(\mathcal{C}, \mathscr{P}, \mathscr{N})$
2: $F^{\mathcal{L}}_{\mathbf{y}} \leftarrow$ Score-Function$(\mathbf{y}, \mathscr{H})$ ▷ *Reference value of the score function*
3: **while** $i = 1 \dots n - s$ **do**
4: $\quad \mathbf{y}^{(i)} \leftarrow \mathbf{y} + \delta_{\mathscr{N}_i}$ ▷ *This flips the* i*'th bit of* $\mathbf{e}_{\mathscr{N}}$
5: $\quad F^{\mathcal{L}}_{\mathbf{y}^{(i)}} \leftarrow$ Score-Function$(\mathbf{y}^{(i)}, \mathscr{H})$ ▷ *Comparative score function*
6: Compute $G(\mathbf{z})$ (Definition 46) for all $\mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}$
7: **return** $\{(\mathbf{z},\ G(\mathbf{z}))\ :\ \mathbf{z} \in \mathbb{F}_2^{k_{\text{aux}}}\}$

7.2.3 Checking phase

Here we want to check each guess for $\mathbf{e}_{\mathscr{N}}$. It is important that each guess is checked in polynomial time as we range over the whole secret space $\mathbb{F}_2^{k_{\text{aux}}}$, and we want this step to no dominate in front of the FFT say. Note that in the case of double-RLPN the secret of the LPN samples is not $\mathbf{e}_{\mathscr{P}}$ but rather some linear combination of $\mathbf{e}_{\mathscr{P}}$, say $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ where $\mathbf{G}_{\text{aux}}$ is the generator matrix of the code used in the reduction from $\mathsf{sparse}-\mathsf{LPN}$ to $\mathsf{plain}-\mathsf{LPN}$. Consequently, given $\mathbf{z}$ a candidate for $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ and an associated guess for $\mathbf{e}_{\mathscr{N}}$ we cannot easily verify if this couple is indeed the solution to our decoding problem as we could have done if we had access to $\mathbf{e}_{\mathscr{P}}$. Notably, trying to recover $\mathbf{e}_{\mathscr{P}}$ from $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ would be exponentially harmful in here as we observe that in practice our double-RLPN optimal parameters are such that this compression of $\mathbf{e}_{\mathscr{P}}$ is extremely lossy, namely the Hamming sphere in which $\mathbf{e}_{\mathscr{P}}$ lives is exponentially larger than this arrival space. We believe there are workarounds for this issue but to keep our new algorithm the most simple possible we restrict ourselves to the setting where the position given by $\mathscr{N}$ are an information set of the code $\mathcal{C}$ with good probability. In this case everything is simple as a guess $\mathbf{z}$ for $\mathbf{e}_{\mathscr{N}}$ can be verified in polynomial time by simply computing the unique codeword $\mathbf{c}$ of $\mathcal{C}$ which is equal to $\mathbf{y}_{\mathscr{N}} - \mathbf{z}$ on the part $\mathscr{N}$ and then check that $\mathbf{y} - \mathbf{c}$ is of good Hamming weight t (the weight of the error). Importantly, denoting by $s \stackrel{\text{def}}{=} |\mathscr{P}|$, the setting where $\mathscr{N}$ is an information set with good probability, is when

$$n - s \geqslant k$$

which are trivially verified by all our parameters when the rate R of the code $\mathcal{C}$ is smaller than 0.5 as $s \leqslant k$. This is more than enough to account for the interesting parameters for which we beat the ISD's. The checking algorithm is described in Algorithm 28.

Algorithm 28

Name: CHECKING-$\mathbf{e}_{\mathscr{N}}(\mathcal{G}, \mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N})$
Input: $\mathcal{G}, \mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N}$
Parameter: t

while $\mathbf{e}_{\mathscr{N}}^{(\mathbf{z})} \in \mathcal{G}$ **do**
  $\mathbf{c}_{\mathscr{N}} \leftarrow \mathbf{y}_{\mathscr{N}} - \mathbf{e}_{\mathscr{N}}^{(\mathbf{z})}$
  $\mathbf{c} \leftarrow \text{Lift}\left(\mathcal{C},\ \mathscr{N},\ \mathbf{c}_{\mathscr{N}}\right)$
  if $|\mathbf{y} - \mathbf{c}| = t$ **then**
    $\mathbf{e} \leftarrow \mathbf{y} - \mathbf{c}$
    return $\mathbf{e}$

It is readily seen that we have that

Fact 30. *If the guess for $\mathbf{e}_{\mathscr{N}}$ related to the secret $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}$ is good, namely if $G\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) = \mathbf{e}_{\mathscr{N}}$ and if $\mathscr{N}$ contains an information set of $\mathcal{C}$ then* CHECKING-$\mathbf{e}_{\mathscr{N}}$ *outputs* $\mathbf{e}$.

7.2.4 Whole algorithm

Finally, our whole algorithm retakes the main loop of double-RLPN by choosing at random two complementary subsets $\mathscr{P}$ and $\mathscr{N}$ of $[\![1,\ n]\!]$ and an auxiliary code $\mathcal{C}_{\text{aux}}$ at random among a family $\mathscr{F}$ and running the checking and guessing procedure. We loop N_{iter} in order to ensure that there exists at least one iteration such that the weight distribution of the error $\mathbf{e}$ on its subpart verifies some bet.

Algorithm 29 double-RLPN algorithm

Name: DOUBLE-RLPN
Input: $\mathcal{C} \in \mathfrak{C}\left[n, k\right],\ \mathbf{y} \in \mathbb{F}_2^n,\ t$
Parameter: s, w, u and N_{iter}

while $i = 1 \ldots N_{\text{iter}}$ **do**
  $\mathscr{P} \xleftarrow{\$} \{\mathscr{P} \subset [\![1,\ n]\!]\ :\ |\mathscr{P}| = s\}$ ▷ *Hope that* $\mathbf{e}_{\mathscr{N}} = u$
  $\mathscr{N} \leftarrow [\![1,\ n]\!] \setminus \mathscr{P}$
  $\mathcal{G} \leftarrow$GUESSING-$\mathbf{e}_{\mathscr{N}}(\mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N})$
  $\mathbf{e} \leftarrow$ CHECKING-$\mathbf{e}_{\mathscr{N}}(\mathcal{G},\ \mathcal{C}, \mathbf{y}, \mathscr{P}, \mathscr{N}, t)$
  if $\mathbf{e} \neq \perp$ **then**
    return $\mathbf{e}$

7.3 Instantiation with a juxtaposition codes

As done in the chapter presenting double-RLPN we instantiate it using juxtaposition codes and their decoder.

Algorithm 4. *We define an instantiation of Algorithm 29 taking b as an additionnal parameter where*

- *The family $\mathcal{F} \subset \mathfrak{C}\left[s, k_{\text{aux}}\right]$ of auxiliary codes is defined as*

$$\mathcal{F} = \mathfrak{C}^{\text{juxt}}\left[b,\ s,\ k_{\text{aux}}\right]$$

- *The procedure* Sample-Auxilary-Code($\mathcal{F}$) *which returns a code* $\mathcal{C}_{\text{aux}}$ *which was chosen uniformly at random in* $\mathfrak{C}^{\text{juxt}}[b,\ s,\ k_{\text{aux}}]$ *by choosing each of its constituent codes* $\mathcal{C}_{\text{aux}}^{(i)} \sim \mathcal{U}\left(s^{(i)},\ k^{(i)}\right)$.
- *After having chosen the code* $\mathcal{C}_{\text{aux}}$ *we precompute the table of syndromes with a call to* SyndromeTable $\leftarrow$ Juxt-PrecomputeSyndrome($\mathcal{C}_{\text{aux}}; t_{\text{aux}}$). *Each call to* Auxilary-Decode($\mathcal{C}_{\text{aux}},\ \mathbf{a}; t_{\text{aux}}$) *is then replaced with one call to* Juxt-Decode($*$SyndromeTable, $\mathbf{a}$).

It is readily seen, that this instantiation has the following complexity.

Proposition 55 (Complexity of fully provable double-RLPN with juxtaposition codes)**.** *Let* n *and let* $k, t, s, k_{\text{aux}}, t_{\text{aux}}, N_{\text{iter}}, b$ *be the parameters of Algorithm 4, all implicit functions of* n. *Suppose that* $b \geqslant \lceil s/k_{\text{aux}} \rceil$ *and that* $\binom{s}{t_{\text{aux}}}/2^{s-k_{\text{aux}}} = \text{poly}(n)$. *Then, given an instance of* $\text{DP}_{\mathbf{G}}(n, k, t)$, *the expected time and memory complexity of Algorithm 4 is given by*

$$\boldsymbol{Time} = \widetilde{\mathcal{O}}\Big(N_{\text{iter}}\left(T_{\text{eq}} + 2^{k_{\text{aux}}}\right)\Big), \qquad \boldsymbol{Memory} = \widetilde{\mathcal{O}}\Big(\left(M_{\text{eq}} + 2^{k_{\text{aux}}}\right)\Big)$$

where T_{eq} *and* M_{eq} *are respectively the expected time and memory complexity of one call to* Compute-Dual-Vector($\mathcal{C}$) *given* $\mathcal{C} \sim \mathcal{U}_{\mathbf{G}}(n-s,\ k-s)$.

7.4 Performance of the algorithm, main theorem

We give here the main theorem giving the complexity and correctness of the algorithm, we prove it in the next Section 7.5. Our theorem is stated when the number of blocks in the juxtaposition code is constant. Furthermore, we make our statement in the case where essentially all the dual vectors are computed and the number of blocks in the juxtaposition code is constant.

Theorem 9. *There exists a positive poly-bounded function* f *such that for any* $k, t, s, b, k_{\text{aux}}, t_{\text{aux}}, w, u \in \mathbb{N}$ *implicit functions of a parameter* $n \in \mathbb{N}$ *(*n *is growing to infinity) and any procedure* Compute-Dual-Vectors *that are such that*

1. *(Computing all the dual vectors)*

$$\mathbb{P}_{\mathcal{D}\sim\mathcal{U}_{\mathbf{G}}(n-s,\ k-s)}\left(\text{Compute-Dual-Vectors}(\mathcal{D}) = \mathcal{D}^{\perp} \bigcap \mathcal{S}_w^{n-s}\right) \in 1 - o(1)\,,$$

2. *(Linear scaling and constant number of blocks)* $b = \left\lceil \frac{s}{k_{\text{aux}}} \right\rceil$ *and* $b \in \mathcal{O}(1)$,

3. *(Main constraint that we have enough dual vectors)*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \in \Omega\left(\frac{f(n^b)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2}\right),$$

4. *(Decoding the auxiliary code below Gilbert-Varshamov)* $\binom{s}{t_{\text{aux}}}/2^{s-k_{\text{aux}}} \in \mathcal{O}(1)$,

5. *(*$\mathscr{N}$ *is an information set of the code* $\mathcal{C}$*)* $n - s - k \in \omega(1)$,

6. *(Small technical constraints)* $u < \mathrm{Root}\left(K_w^{(n-s)}\right)$ *and* $(t-u)^{(i)} < \mathrm{Root}\left(K_{t_{\mathrm{aux}}^{(i)}}^{(s^{(i)})}\right)$ *for* $i \in [\![1,\, b]\!]$ *and* $k - s \in \omega(1)$

then there exists N_{iter} *an implicit function of* n *such that Algorithm 4 solve* $\mathrm{DP}_{\mathbf{G}}(n,k,t)$ *with probability* $1-o(1)$ *in time and memory*

$$\boldsymbol{Time} = \widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{\binom{s}{t-u}\binom{n-s}{u}}\left(2^{k_{\mathrm{aux}}} + T_{\mathrm{eq}}\right)\right), \qquad \boldsymbol{Memory} = \widetilde{\mathcal{O}}\Big(\left(2^{k_{\mathrm{aux}}} + M_{\mathrm{eq}}\right)\Big)$$

where T_{eq} *and* M_{eq} *are respectively the expected time and memory complexity of one call to the procedure* Compute-Dual-Vectors$(\mathcal{D})$ *when* $\mathcal{D} \sim \mathcal{U}_{\mathbf{G}}(n-s,\, k-s)$ *and where we recall that* $\delta_w^{(n)}(t)$ *is defined in Definition 25 and* $\mathrm{Root}\left(K_w^{(n)}\right)$ *is defined in Definition 26.*

In particular, this gives the following corollary.

Corollary 13. *Any complexity claim made under Conjecture 3 concerning Algorithm 3 with parameters verifying the constraints and choices of Theorem 8 are provably achieved, up to polynomial factors, by Algorithm 4 when the rate of the code* R *is smaller than* 0.5.

While we believe it is true, we refrain to say that in general, the performance of this new algorithm is equivalent, up to polynomial factors to the performance of double-RLPN algorithm, this is the reason why restricted the previous corollary to parameters verifying constraints given by Theorem 8. Indeed, in double-RLPN because we allowed some false candidates we have more margin on the choice of the parameters. However we think that this margin is of a polynomial nature. In contrary, while we believe that false candidates are never an issue in double-RLPN, we do not know in general if there are some pathological optimal parameters which would be such that the cost of checking false candidates dominate, even if we conjecture that it is never the case.

7.5 Analysis, proof of the main theorem

Here our goal is to prove Theorem 9. Again, to simplify further statements we use the following notation which corresponds to the quantities handled by our algorithm when we are in a good iteration. More precisely we suppose $\dim(\mathcal{C}_{\mathscr{P}}) = s$ (this happens with probability $1-o(1)$ as $n-s-k = \Omega(\log_2 k)$) and that all the dual vectors of weight w on $\mathscr{N}$ are computed (this also happens with probability $1-o(1)$). Last in fact in the proof of Theorem 9 we choose N_{iter} as some well-chosen $\widetilde{\mathcal{O}}\left(\frac{\binom{n}{t}}{\binom{s}{t-u}\binom{n-s}{u}}\right)$ so that we are guaranteed with probability $1-o(1)$ that there exists some iteration which is such that the bet on the error is valid (as well as the finer bet due to the use of juxtaposition codes):

$$|\mathbf{e}_{\mathscr{N}}| = u \bigwedge_{i=1}^{b} \left|\mathbf{e}_{\mathscr{P}}^{(i)}\right| = (t-u)^{(i)}.$$

This is true because $b = \mathcal{O}(1)$. Our main proposition will be that, the guess on $\mathbf{e}_{\mathscr{N}}$ related to the secret of the LPN problem $\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\mathrm{aux}}^{\intercal}$ is valid with good probability, namely that we have that $G(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\mathrm{aux}}^{\intercal}) = \mathbf{e}_{\mathscr{N}}$ with probability $1-o(1)$.

Proposition 56. *There exists a positive poly-bounded function f such that for any $t, k, s, k_{\text{aux}}, t_{\text{aux}}, w, u, b \in \mathbb{N}$ implicit functions of n such that $u < \text{Root}\left(K_w^{(n-s)}\right)$ and $(t-u)^{(i)} < \text{Root}\left(K_{t_{\text{aux}}^{(i)}}^{(s^{(i)})}\right)$ for all $i \in [\![1,\, b]\!]$ and*

$$\frac{\binom{n-s}{w}\binom{s}{t_{\text{aux}}}}{2^{k-k_{\text{aux}}}} \geqslant \frac{f\left(n^b\right)}{\left(\delta_w^{(n-s)}(u)\,\delta_{t_{\text{aux}}}^{(s)}(t-u)\right)^2} \quad \text{and} \quad \frac{\binom{s}{t_{\text{aux}}}}{2^{s-k_{\text{aux}}}} \in \mathcal{O}(1) \quad \text{and} \quad b \in \mathcal{O}(1)$$

then we have that

$$\mathbb{P}\left(G\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right) = \mathbf{e}_{\mathscr{N}}\right) = 1 - o(1)\,.$$

where

- *$\mathscr{N}$ and $\mathscr{P}$ are two fixed complementary subsets of $[\![1,\, n]\!]$.*
- *$\mathcal{C}$ is distributed according to the distribution $\mathcal{U}_{\mathbf{G}}(n,\, k)$ conditioned on the event that $\dim\left(\mathcal{C}_{\mathscr{P}}\right) = s$ and $\mathcal{C}_{\text{aux}} \sim \mathcal{U}\left(\mathfrak{C}^{\text{juxt}}[b,\, s,\, k_{\text{aux}}]\right)$ and $\mathbf{G}_{\text{aux}} \in \mathbb{F}_2^{k_{\text{aux}} \times s}$ is any generator matrix of $\mathcal{C}_{\text{aux}}$.*
- *$\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \sim \mathcal{U}(\mathcal{C})$ and $\mathbf{e} \in \mathcal{S}_t^n$ is a fixed vector such that the bet is valid, namely*

$$|\mathbf{e}_{\mathscr{N}}| = u \bigwedge_{i=1}^{b} \left|\mathbf{e}_{\mathscr{P}}^{(i)}\right| = (t-u)^{(i)}\,.$$

- *The set $\mathscr{H}$ is*

$$\mathscr{H} \overset{def}{=} \{(\mathbf{h}, \mathbf{m}_{\text{aux}}) \in \mathcal{C}^{\perp} \times \mathbb{F}_2^{k_{\text{aux}}} \;:\; |\mathbf{h}_{\mathscr{N}}| = w,\; \forall i \in [\![1,\, b]\!], (\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}} + \mathbf{h}_{\mathscr{P}})^{(i)} \in \mathcal{S}_{t_{\text{aux}}^{(i)}}^{s^{(i)}}\}.$$

- *The other quantities appearing $\mathbf{y}^{(i)}$, $F_{\mathbf{y}}^{\mathcal{L}}(\mathbf{z})$ and $G(\mathbf{z})$ are the one as defined in Section 7.2.*

One can convince himself that this last proposition imply the main Theorem 9. The goal of this section is to prove this proposition. Let us recall the quantities quickly. By definition $G(\mathbf{z})$ is a vector of $\mathbb{F}_2^{n-s}$ defined coordinate by coordinate as

$$G(\mathbf{z})_i \overset{\text{def}}{=} \begin{cases} 1 & \text{if} \quad F_{\mathbf{y}^{(i)}}^{\mathcal{L}}(\mathbf{z}) > F_{\mathbf{y}}^{\mathcal{L}}(\mathbf{z}) \\ 0 & \text{else.} \end{cases}$$

where we recall that $\mathbf{y}^{(i)}$ is $\mathbf{y}$ where we flipped the i'th bit of $\mathbf{y}_{\mathscr{N}}$. Now to prove that the guess is good with probability $1 - o(1)$ we in fact prove that the guess on each coordinate is valid with probability $1 - o(1/n)$, namely that we have that $G\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}^{\intercal}\right)_i = (\mathbf{e}_{\mathscr{N}})_i$ with probability $1 - o(1/n)$ and conclude with a union bound argument. The main tool we use are the second order concentration bounds on the score function devised in the previous chapter: applying Byenemé-Chebyshev inequality on the moment of the score function given by Lemma 31 we have the following.

Lemma 41 (Corollary of Lemma 31)**.** *There exists a positive poly-bounded function f such that for any $t, k, s, k_{\mathrm{aux}}, t_{\mathrm{aux}}, w, u, b \in \mathbb{N}$ implicit functions of n such that $\binom{s}{t_{\mathrm{aux}}}/2^{s-k_{\mathrm{aux}}} \in \mathcal{O}(1)$ then for any $i \in [\![1,\, n-s]\!]$ we have that*

$$\mathbb{P}\left(\left|F^{\mathcal{L}}_{\mathbf{y}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}^{\intercal}_{\mathrm{aux}}\right) - E\left(0\right)\right| \geqslant f_1(n^b)\sqrt{N}\right) = \mathcal{O}\left(\frac{1}{n^2}\right),$$
$$\mathbb{P}\left(\left|F^{\mathcal{L}}_{\mathbf{y}^{(i)}}\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}^{\intercal}_{\mathrm{aux}}\right) - E\left((-1)^{(\mathbf{e}_{\mathscr{N}})_i}\right)\right| \geqslant f_1(n^b)\sqrt{N}\right) = \mathcal{O}\left(\frac{1}{n^2}\right)$$

where

$$E\left(x\right) \stackrel{def}{=} N\delta^{(n-s)}_{w}\left(u+x\right)\prod_{j=1}^{b}\delta^{(s^{(j)})}_{t^{(j)}_{\mathrm{aux}}}\left((t-u)^{(j)}\right), \qquad N \stackrel{def}{=} \frac{\binom{n-s}{w}\prod_{j=1}^{b}\binom{s^{(j)}}{t^{(j)}_{\mathrm{aux}}}}{2^{k-k_{\mathrm{aux}}}}.$$

Thus, using the notations and quantities of the previous propositions, we make the good guess for each coordinate with probability $1-o\left(1/n^2\right)$ as long as

$$\begin{cases} E\left(-1\right) - E\left(0\right) > f_1(n^b)\sqrt{N} & \text{if } \left(\mathbf{e}_{\mathscr{N}}\right)_i = 1 \\ E\left(0\right) - E\left(1\right) > f_1(n^b)\sqrt{N} & \text{if } \left(\mathbf{e}_{\mathscr{N}}\right)_i = 0. \end{cases}$$

Let us upper bound the minimum value of these differences. We have that

Lemma 42. *There exists a positive poly-bounded function f_2 such that for any $t, k, s, k_{\mathrm{aux}}, t_{\mathrm{aux}}, w, u, b \in \mathbb{N}$ implicit functions of n such that $u < \mathrm{Root}\left(K^{(n-s)}_{w}\right)$ and $(t-u)^{(i)} < \mathrm{Root}\left(K^{(s^{(i)})}_{t^{(i)}_{\mathrm{aux}}}\right)$ for all $i \in [\![1,\, b]\!]$ then*

$$\min\left(E\left(-1\right) - E\left(0\right),\ E\left(0\right) - E\left(1\right)\right) \geqslant \frac{1}{f_2(n)} N\delta^{(n-s)}_{w}\left(u\right)\prod_{j=1}^{b}\delta^{(s^{(j)})}_{t^{(j)}_{\mathrm{aux}}}\left((t-u)^{(j)}\right)$$

We prove this lemma in Section 7.5.1. Using the two previous lemmas we get the following proposition.

Proposition 57. *There exists a positive poly-bounded function f_3 such that for any $t, k, s, k_{\mathrm{aux}}, t_{\mathrm{aux}}, w, u, b \in \mathbb{N}$ implicit functions of n such that $u < \mathrm{Root}\left(K^{(n-s)}_{w}\right)$ and $(t-u)^{(i)} < \mathrm{Root}\left(K^{(s^{(i)})}_{t^{(i)}_{\mathrm{aux}}}\right)$ for all $i \in [\![1,\, b]\!]$ and*

$$N \geqslant \frac{f_3\left(n^b\right)}{\left(\delta^{(n-s)}_{w}\left(u\right)\prod_{j=1}^{b}\delta^{(s^{(j)})}_{t^{(j)}_{\mathrm{aux}}}\left((t-u)^{(j)}\right)\right)^2}$$

then we have that

$$\mathbb{P}\left(G\left(\mathbf{e}_{\mathscr{P}}\mathbf{G}^{\intercal}_{\mathrm{aux}}\right) = \mathbf{e}_{\mathscr{N}}\right) = 1 - o(1).$$

Finally, using the Lemma 30 about the fact that all these quantities relate polynomially to the case $b = 1$ as long as $b = \mathcal{O}(1)$, we finally get the main Proposition 56 we wanted to prove.believe

7.5.1 Proof of the intermediate lemmas

Here we prove Lemma 42. Recall that by definition in Lemma 41 we have that

$$E(-1) - E(0) = N\left(\delta_w^{(n-s)}(u-1) - \delta_w^{(n-s)}(u)\right)\prod_{j=1}^{b}\delta_{t_{\text{aux}}^{(j)}}^{(s^{(j)})}\left((t-u)^{(j)}\right) \tag{7.2}$$

$$E(0) - E(1) = N\left(\delta_w^{(n-s)}(u) - \delta_w^{(n-s)}(u+1)\right)\prod_{j=1}^{b}\delta_{t_{\text{aux}}^{(j)}}^{(s^{(j)})}\left((t-u)^{(j)}\right). \tag{7.3}$$

We use the following lemma for the proof

Lemma 43 (Difference of bias)**.** *Let $n, w, t \in \mathbb{N}$, we have that*

$$\delta_w^{(n)}(t-1) - \delta_w^{(n)}(t) = 2\frac{w}{n}\delta_{w-1}^{(n-1)}(t-1).$$

Proof. Recall that

$$\delta_w^{(n)}(t) = \frac{K_w^{(n)}(t)}{\binom{n}{w}}.$$

From the recurrence relations of Proposition 20 we have that

$$\begin{aligned} K_w^{(n)}(t-1) &= K_w^{(n-1)}(t-1) + K_{w-1}^{(n-1)}(t-1), \\ K_w^{(n)}(t) &= K_w^{(n-1)}(t-1) - K_{w-1}^{(n-1)}(t-1). \end{aligned}$$

Now, using the fact that

$$\binom{n}{w} = \frac{n}{w}\binom{n-1}{w-1}$$

we get

$$\begin{aligned} \delta_w^{(n)}(t-1) - \delta_w^{(n)}(t-1) &= \frac{2\,K_{w-1}^{(n-1)}(t-1)}{\binom{n}{w}} \\ &= 2\,\frac{w}{n}\frac{K_{w-1}^{(n-1)}(t-1)}{\binom{n-1}{w-1}}. \end{aligned}$$

□

Using this lemma inside Eq. (7.2) yield that

$$E(-1) - E(0) = N2\frac{w}{n}\delta_{w-1}^{(n-s-1)}(u-1)\prod_{j=1}^{b}\delta_{t_{\text{aux}}^{(j)}}^{(s^{(j)})}\left((t-u)^{(j)}\right) \tag{7.4}$$

$$E(0) - E(1) = N2\frac{w}{n}\delta_{w-1}^{(n-s-1)}(u)\prod_{j=1}^{b}\delta_{t_{\text{aux}}^{(j)}}^{(s^{(j)})}\left((t-u)^{(j)}\right). \tag{7.5}$$

Now from the fact that we supposed in the proposition that we are ultimately outside the root region of Krawtchouk polynomials we have that the bias is decreasing in this region so we can write that ultimately, for n big enough,

$$\begin{aligned}\min\left(E\left(-1\right)-E\left(0\right),\ E\left(0\right)-E\left(1\right)\right) &= E\left(0\right)-E\left(1\right)\\ &= N2\frac{w}{n}\delta_{w-1}^{(n-s-1)}\left(u\right)\prod_{j=1}^{b}\delta_{t_{\text{aux}}^{(j)}}^{(s^{(j)})}\left(\left(t-u\right)^{(j)}\right).\end{aligned}$$

Using once again the fact that we are outside the root region of Krawtchouk polynomials and Corollary 5 (along the fact that κ is derivable) we get that

$$\delta_{w-1}^{(n-s-1)}\left(u\right)=\tilde{\Omega}\left(\delta_{w}^{(n-s)}\left(u\right)\right)$$

where the $\tilde{\Omega}\left(\right)$ term does not depend on the functions w,u,s. This concludes the proof.

Chapter 8

Dual attacks in lattice-based cryptography and their analysis

Summary

In this section we give a very brief introduction to the LWE problem and lattice-based cryptography. We draw a quick comparison between the recent lattice-based dual attacks and the code-based dual attacks we devised in this thesis. Some of the most recent attacks [GJ21, MAT22] claim to diminish the security of Kyber but their analysis relies on independence assumptions that were recently shown to be false in [DP23b]. This leaves open the question of whether these attacks work as expected and how to actually analyze them. As a contribution in this chapter we provide tools for this analysis. These results were published, in a slightly different form in [CDMT24, Section 8].

Contents

8.1 The LWE problem and lattice-based cryptography

In his seminal work, [Reg05] built an encryption scheme whose security relied on the hardness of the so-called Learning With Errors (LWE) problem. This problem is conjectured to be quantumly hard to solve and its hardness is now at the heart of the security of many schemes, such as the NIST standards Kyber and Dilithium. As such, the complexity of the best algorithms for solving it allows to correctly choose the parameters of those schemes.

8.1.1 The LWE problem

The LWE problem is essentially a generalization of the decoding problem presented before, the difference being that the considered linear system lies in a much larger ring $\mathbb{Z}_q$ with say $q > 2$ instead of the binary field $\mathbb{F}_2$ and the metric changes from the Hamming weight to a much finer one: the coordinates of the error $\mathbf{e} \in \mathbb{Z}_q^n$ are chosen to be small compared to q. Here we qualify "small" by identifying $\mathbb{Z}_q$ with the set of integers (say q is odd) $\{ -(q-1)/2, \cdots 0, \cdots, (q-1)/2\}$.

Definition 47 (LWE problem, generator matrix)**.** *Let $n, k, q \in \mathbb{N}$ with $k \leqslant n$ and let χ be a distribution with support in $\mathbb{Z}_q$. Given n, k, q, χ and $(\mathbf{G},\ \mathbf{y})$ where*

- $\mathbf{G}$ *is taken uniformly at random in* $(\mathbb{Z}_q)^{k\times n}$,
- $\mathbf{y} = \mathbf{sG} + \mathbf{e}$ *where* $\mathbf{e} \sim \chi^n$ *and* $\mathbf{s} \sim \mathcal{U}\left(\mathbb{Z}_q^k\right)$,

the goal is to find the vector $\mathbf{e}$*. We sometime additionally refer to k as the dimension of the problem and n as the length or number of samples.*

Remark 26. *To fix the intuition about the hardness of this problem, one can think of* Kyber*'s hardest mode whose security essentially relies on solving this* LWE *problem with parameters $q = 3329$, $n = 2048$, $k = 1024$ and χ is a centered binomial distribution with support $[\![-2, 2]\!]$. This represents a regime which is deeply in the injective regime, where we do not expect any other non-planted solution. The current estimations (i.e. the complexity of the best attack) predict that solving this problem is as hard as breaking AES-256.*

In practice χ is often symmetric and concentrated around 0, and is a discrete Gaussian distribution in Regev's original work. This naturally involves the Euclidean metric as in that case one can argue that the error $\mathbf{e}$ is roughly uniformly distributed in some Euclidean ball of small known radius and the problem really becomes to find the $\mathbf{e}$ of small Euclidean norm $||\mathbf{e}||$ such that $\mathbf{b} - \mathbf{e}$ is a codeword of the code generated by $\mathbf{G}$. Here we embedded $\mathbb{Z}_q^n$ with the standard Euclidean norm $||\mathbf{x}|| = \sum_{i=0}^n x_i^2$ where $\mathbf{x} \in \mathbb{Z}_q^n$ is seen, we recall, as an integer vector with entries smaller or equal than $(q-1)/2$.

Now, note that Regev's scheme can be viewed as the lattice counterpart of the code-based Alekhnovich scheme. With various improvements it eventually yielded Kyber. One of which came from the introduction, for efficiency purposes, of an underlying polynomial ring structure, but it is not believed to degrade the hardness of the problem by more than polynomial factors in practice. This will be the reason why we stick, as it is traditionally done for generic attacks, to the study of the hardness of the problem defined over $\mathbb{Z}_q$.

Contrary to the standard decoding problem we presented earlier, taking a higher modulus and using a finer norm at the same time somewhat changes the source of hardness of this problem in terms of both the underlying security reduction and the techniques used to solve it. Let us explain both of these points, each in the two following subsections.

8.1.2 Lattices

Regev [Reg05] also gave arguments for the security of his scheme by showing that the average LWE problem was quantumly harder than some worst-case lattice problems that were difficult in practice.

Definition 48 (Lattice)**.** *A lattice Λ is a discrete subgroup of $\mathbb{R}^n$.*

Fact 31. *If Λ is a lattice of $\mathbb{R}^n$ there exists a rank $d \leqslant n$ and basis $\mathbf{B} \in \mathbb{R}^{d \times n}$ of rank d such that $\Lambda = \mathbb{Z}^d\mathbf{B}$. We denote by*

$$\Lambda(\mathbf{B}) \stackrel{def}{=} \{ \mathbf{xB} \,:\, \mathbf{x} \in \mathbb{Z}^n \}$$

the lattice generated by the basis $\mathbf{B}$.

More precisely, Regev showed that there was a polynomial quantum reduction from the problem of approximating (up to a polynomial factor $\sqrt{n}$) the length of the shortest non-zero vector in any given lattice to the LWE problem. And, while [AR05] showed that this last approximation problem was in $\mathrm{NP} \cap \mathrm{CoNP}$, it is still believed to be hard in practice, as the best-known algorithms solving this problem are sieving algorithms, starting with [AKS01], that run in time $2^{\mathcal{O}(n)}$. Roughly speaking, his reduction works when the modulus and the error distribution are sufficiently large and wide respectively (compared to n). It was left as an open question whether or not such a reduction could exist in the binary case.

8.1.3 Primal attacks for solving LWE

First, recall that LWE is really a code problem. Say q is prime, then $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ can be seen as the generator matrix of a q-ary code $\mathcal{C}$, and $\mathbf{y} = \mathbf{sG} + \mathbf{e}$ as some noisy codeword we want to decode. However, it is readily seen that the techniques used to solve the standard binary decoding problem and the LWE problem are somewhat different. For example, one can convince oneself that the Prange bet that k positions of the error are all 0 becomes less and less interesting as the metric gets finer (whereas it's a very good bet in $\mathbb{Z}_q$ with the Hamming metric). Solving LWE is rather often done by reducing it to a lattice problem. Namely, one can embed $\mathcal{C}$ onto a lattice with Construction A, defined as follows.

Definition 49 (Construction A)**.** *Let $\mathcal{C}$ be a q-ary linear code of length n. The lattice obtained by Construction A applied to $\mathcal{C}$ is given by*

$$\begin{aligned} \Lambda_q(\mathcal{C}) &\stackrel{def}{=} \Lambda = \mathcal{C} + q\mathbb{Z}^n && (8.1) \\ &= \{\mathbf{x} \in \mathbb{Z}^n \,:\, (\mathbf{x} \bmod q) \in \mathcal{C}\}\,. && (8.2) \end{aligned}$$

We say furthermore that $\Lambda_q(\mathcal{C})$ is a q-ary lattice.

In that case, recovering the error $\mathbf{e}$ from $\mathbf{y} = \mathbf{c} + \mathbf{e}$ is really finding the point of the lattice Λ which is the closest to $\mathbf{y}$, seen as a vector in $\mathbb{R}^n$. One can easily turn this problem into the problem of finding the shortest vector in a closely related lattice. This can be solved by enumeration techniques in superpolynomial time [SE94, Kan83], but more recent sieving-style algorithms [AKS01, NV08, MV] allow solving the problem in time $2^{\mathcal{O}(n)}$. These are called primal attacks because they manipulate primal vectors, i.e. vectors in Λ. These sievers can be sped up by using some near-neighbor subroutine [BGJ15, Laa15, BDGL16]. The best in terms of asymptotic time complexity is given by [BDGL16], whose technique allows one to heuristically find the shortest vector in time $2^{0.292n(1+o(1))}$.

8.1.4 Length of the shortest vector

One natural question when studying this shortest vector problem is to actually estimate the length of the shortest vector in a random lattice. The fundamental quantity underlying this question is its volume.

Definition 50 (volume)**.** *The volume $\mathcal{V}(\Lambda)$ of a lattice Λ is the Euclidean volume of its fundamental domain*

$$\mathcal{V}(\Lambda) \stackrel{def}{=} \mathcal{V}(\mathrm{span}(\Lambda)/\Lambda) = \sqrt{\det(\mathbf{B}^\intercal \mathbf{B})}$$

where $\mathbf{B}$ is any basis of Λ and where $\mathcal{V}()$ on the right side of the equality is the Euclidean volume.

It is easy to compute the expected number of lattice points in a ball from a random coset of a fixed lattice.

Fact 32. *Let Λ be a full rank lattice in $\mathbb{R}^n$. Let $\mathbf{y} \sim \mathcal{U}(\mathbb{R}^n/\Lambda)$. We have for any $r > 0$ that*

$$\mathbb{E}\left((\Lambda + \mathbf{y}) \bigcap \mathsf{Ball}_r^n\right) = \frac{\mathcal{V}(\mathsf{Ball}_r^n)}{\mathcal{V}(\Lambda)}$$

where we recall that $\mathcal{V}(\mathsf{Ball}_r^n) = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} r^n$.

The situation is slightly more tricky when we want to compute the expected number of lattice points in a ball for a random lattice (without taking a random coset). A classical result of [Rog56] yields that for lattices taken at random according to Siegel's measure [Sie45], we really have that

$$\mathbb{E}\left((\Lambda \setminus \{\mathbf{0}\}) \bigcap \mathsf{Ball}_w^n\right) = \frac{\mathcal{V}(\mathsf{Ball}_r^n)}{\mathcal{V}(\Lambda)}.$$

The same result is slightly less simple for random q-ary lattices as computing the expected number of codewords of $\mathcal{C}$ within a Euclidean ball of a certain radius is easy but involves the number of points of $\mathbb{Z}_q^n$ inside a certain Euclidean ball whose exact formula is rather complicated. However, as q grows one can reasonably approximate this quantity by the volume of the Euclidean ball, see [Zam14, Lemma 7.9.2] for a more precise statement. To simplify the expressions, from Stirling's formula we have the classical asymptotic equivalent $\Gamma\left(\frac{n}{2}+1\right) \sim \sqrt{\pi n}\left(\frac{n}{2e}\right)^{n/2}$, that leads to the following heuristic.

Heuristic 1 (Gaussian Heuristic)**.** *When Λ is a (random) full-rank lattice in $\mathbb{R}^n$ we say that we use the Gaussian heuristic when we approximate $|(\Lambda \setminus \{\mathbf{0}\}) \bigcap \mathsf{Ball}_r^n|$ by*

$$\frac{r^n}{\mathcal{V}(\Lambda)} \frac{(2\pi e)^{n/2}}{\sqrt{n\pi} n^{n/2}}.$$

Note that for q-ary lattices the volume is easily computable.

Fact 33. *If q is prime and $\mathcal{C}$ is a q-ary linear code of length n and dimension k then the volume of the q-ary lattice given by $\mathcal{C}$ is $\mathcal{V}(\Lambda_q(\mathcal{C})) = q^{n-k}$.*

Combining the Gaussian heuristic with the previous fact we can estimate that the expected length of the smallest non-zero vector in $\mathcal{C}$ or equivalently $\Lambda_q(\mathcal{C})$ is around (neglecting $\sqrt{n\pi}^{1/n}$)

$$q^{(n-k)/n} \sqrt{\frac{n}{2\pi e}}$$

and this estimation gets better as q increases.

8.2 Dual attacks

Similar to coding theory, dual attacks against LWE were first uncompetitive compared to primal attacks, but were successively improved [ADPS16b, Alb17, EJK20], up until [GJ21, MAT22], which eventually claimed to weaken the security of Kyber and Dilithium. These latter claims were seriously questioned recently in [DP23b], that invalidated some key independence heuristics used in the analysis of these attacks. We return to the LWE problem later and focus for now just to give a brief history on the origins of dual attacks [AR04].

8.2.1 A bit of history

Basically, a tool was devised there to prove that the approximate closest vector problem was in NP $\cap$ CoNP. In this problem, given a lattice Λ and some point $\mathbf{y} \in \mathbb{R}^n$, the goal is to decide whether $\mathbf{y}$ is closer or further than some thresholds. The tools developed there led to a procedure to actually solve the problem efficiently for any $\mathbf{y} \in \mathbb{R}^n$, provided that we make a preprocessing step on Λ. Of course, the whole point here was to target a regime where the problem was believed to be hard without this preprocessing step (or the so-called succinct witness). This succinct approximation is based on a key duality using the Poisson summation formula to express distance to the primal lattice in terms of a sum of vectors over the dual lattice.

Definition 51 (Dual lattice)**.** *The dual of a lattice $\Lambda \subset \mathbb{R}^n$ is the lattice of $\mathbb{R}^n$ defined as*

$$\Lambda^\vee \stackrel{def}{=} \{ \mathbf{x} \in \mathrm{span}(\Lambda) \ : \ \langle \mathbf{x}, \mathbf{c} \rangle \in \mathbb{Z} \ \forall \mathbf{c} \in \Lambda \}.$$

Without going into the details, this was possible by relaxing the distance function into some closely related approximate form.

Proposition 58 (Page 10 [AR04])**.** *Let Λ be a lattice of $\mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^n$. We have that*

$$\frac{1}{\alpha} \sum_{\mathbf{x} \in \Lambda} e^{-\pi ||\mathbf{x}-\mathbf{y}||^2} = \mathbb{E}_{\mathbf{h}} \left(\cos \left(2\pi \langle \mathbf{y}, \mathbf{h} \rangle \right) \right)$$

where $\mathbf{h}$ is taken at random in the dual lattice $\Lambda^\vee$ with probability $e^{-\pi ||\mathbf{h}||^2}/\beta$ where $\beta = \sum_{\mathbf{h} \in \Lambda^\vee} e^{-\pi ||\mathbf{h}||^2}$ is a normalization constant and where $\alpha \stackrel{def}{=} \sum_{\mathbf{x} \in \Lambda} e^{-\pi ||\mathbf{x}||^2}$.

Basically, one can note that the left-hand side of the equality approximates the distance of $\mathbf{y}$ to the lattice Λ due to the exponentially decaying nature of $e^{-\pi ||\mathbf{x}-\mathbf{y}||^2}$. Very notably, then, it is possible with a pointwise approximation lemma to argue that the right-hand term of the equality can be sufficiently well approximated by sampling independently a certain number of dual vectors according to this probability distribution. These are the succinct approximations of the distance function.

8.2.2 Idea behind dual attacks against LWE

While dual attacks against LWE can be viewed purely as a problem of finding the closest point to a vector using construction A, these attacks can be explained without lattices. Let us explain the basic principle behind these attacks and to simplify say that we rather target the closely related decision variant of the LWE problem: given either $(\mathbf{G}, \ \mathbf{y} = \mathbf{s}\mathbf{G} + \mathbf{e})$ taken

as an LWE instance or $(\mathbf{G},\ \mathbf{y})$ where $\mathbf{G}$ is uniformly random as in LWE but $\mathbf{y}$ is now also uniformly random in $\mathbb{Z}_q^n$, and the goal is to decide whether it was taken as a uniform instance or an LWE instance. Note that in our setting, namely $q = \text{poly}(n)$ and q prime, there are polynomial search-to-decision reductions [Reg09, Lemma 4.2]. Denoting by $\mathcal{C}$ the code with generator matrix $\mathbf{G}$ one can notice that a dual vector $\mathbf{h} \in \mathcal{C}^\perp$ of small norm can be used to get information about the error by leveraging the fact that in the uniform case $\langle \mathbf{y}, \mathbf{h} \rangle$ is uniformly distributed in $\mathbb{Z}_q$ while in the LWE instance case

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{sG} + \mathbf{e}, \mathbf{h} \rangle = \langle \mathbf{e}, \mathbf{h} \rangle$$

is more biased toward small values of $\mathbb{F}_q$ as the norm of $\mathbf{e}$ and $\mathbf{h}$ are small. The idea is to compute many short dual vectors and make a decision based on the values of the $\langle \mathbf{y}, \mathbf{h} \rangle$'s. Many are needed to make a meaningful decision, and this number grows with the amount of noise, that is with the norm of $\mathbf{e}$ and $\mathbf{h}$. As a side note, the design rationale behind most of those dual attacks is no longer driven by the duality equality of Proposition 58 but rather by seeing the $\langle \mathbf{y}, \mathbf{h} \rangle$'s as independent variables, while this is imprecise this allows to get a first rough idea on the number of dual vectors that will be required to make a meaningful decision. In practice this number is exponential.

8.2.3 Computing the small dual vectors

Depending on the regime of the LWE problem there are different ways of producing the dual vectors. In fact when the number of samples n is big (the LWE problem approaches the LPN setting where we have an unbounded number of samples) combinatorial techniques such as variations of the BKW algorithm (that was originally designed against LPN but that can be adapted to LWE) prevail [BKW03, KF15, GJS15]. However, in the regime we target in this thesis, that is say when the dimension k grows linearly in the number of samples n the most efficient technique is done by embedding $\mathcal{C}^\perp$ onto the q-ary lattice $\Lambda_q\left(\mathcal{C}^\perp\right)$ and finding short vectors in that lattice using lattice tools. Note that this is completely sound as each small vector of the q-ary lattice $\Lambda_q\left(\mathcal{C}^\perp\right)$ yields a unique small vector of $\mathcal{C}^\perp$ by considering its coefficient modulo q. Moreover, in all the regimes we consider in this thesis, this is really a one-to-one correspondence in the sense that the norm of the produced dual vectors is well-below q: the vectors produced really lie deep inside the Euclidean ball of radius $q/2$. Note that alternatively, and to have a common vocabulary between dual attacks against LWE and dual attacks to decode onto a general lattice, it is actually completely equivalent to produce dual vectors of the q-ary lattice constructed with $\mathcal{C}$ as we have the following fact.

Fact 34. *Let $\mathcal{C}$ be a q-ary linear code. We have that*

$$\Lambda_q(\mathcal{C})^\vee = \frac{1}{q}\Lambda_q\left(\mathcal{C}^\perp\right).$$

8.2.3.1 Sieving in a sublattice

Starting from [ADPS16b], all the recent dual attacks [ADPS16b, LW21, GJ21, MAT22] compute those dual vectors with a sieving algorithm [AKS01, NV08, MV, BGJ15, Laa15, BDGL16]. At first those sievers were designed to find the shortest vector in a lattice but in the process they naturally compute an exponential number of small dual vectors that can be used. Recalling, sieving a lattice $\Lambda \subset \mathbb{R}^n$ is done by computing a starting list of N relatively

small vectors (say by enumerating a BKZ or LLL reduced basis) and then say at each iteration we combine (add or substract) all the pairs that lead to a resulting vector of smaller norm. This iteration is repeated several times. Crucially, there is a geometric argument coming from [NV08] that we must have take

$$N = (4/3)^d$$

where d is rank of the lattice (the dimension of the subspace of $\mathbb{R}^n$ spanned by Λ) in order for the list size to stay constant over the iterations, taking anything less than this leads to an unwanted exponential decay of the list. One can then use the $(4/3)^d$ vectors coming from the last list to get a large batch of small vectors of Λ.

In practice in the case of dual attacks, because the lattice coming from the LWE problem $\Lambda = \Lambda_q(\mathcal{C})^\vee \in \mathbb{R}^n$ is of full rank $d = n$ (with overwhelming probability), performing a sieve directly on Λ, we call that a full-sieve, is completely overkill, and leads to computing too much vectors than necessary. To equilibrate the costs those attacks consider $\Lambda' \subset \Lambda$ a sublattice of Λ of rank $d < n$ and rather perform a sieve in that sublattice, bringing down the list size to $(4/3)^d$ and hence the complexity of procedure. In [EJK20, GJ21, MAT22] Λ' is selected by first applying a BKZ reduction on a basis of Λ and select the d best vectors to form a basis of Λ'.

8.2.4 Making a decision with a simple score function

One standard and simple enough way of quantifying how biased $\langle \mathbf{y}, \mathbf{h} \rangle$ when $\mathbf{h} \in \mathcal{C}^\perp$ is to compute $\cos(2\pi \langle \mathbf{y}, \mathbf{h} \rangle / q)$. Clearly it is expected to be 0 in the uniform case and that is expected to be bigger in the LWE instance case: this can heuristically be understood by recalling that for small values of $\langle \mathbf{e}, \mathbf{h} \rangle$ we have by using a Taylor expansion that $\cos(2\pi \langle \mathbf{e}, \mathbf{h} \rangle / q) \approx 1 - \mathcal{O}\left((\langle \mathbf{e}, \mathbf{h} \rangle / q)^2\right)$. Basically those dual attacks computes many small dual vectors $\mathbf{h} \in \mathcal{C}^\perp$ that are stored in a list $\mathscr{W}$ and aggregate those statistics in a score function. It is slightly more convenient for our discussion here to rather consider $\mathscr{W}$ as a set of short vectors of $\Lambda_q(\mathcal{C})^\vee$, hence the following definition where q does not appear (see Fact 34).

Definition 52 (Simple score function)**.** *Let $\mathscr{W}$ a set of vectors of $\mathbb{R}^n$ and $\mathbf{y}$ a vector of $\mathbb{R}^n$ we define the score function as*

$$F_{\mathscr{W}}(\mathbf{y}) = \sum_{\mathbf{h} \in \mathscr{W}} \cos(2\pi \langle \mathbf{y}, \mathbf{h} \rangle).$$

We can then decide that we are in the LWE case if the score F is higher than a certain well-chosen threshold. Here traditionally a weight [AR04] can be applied to give more importance to lower weights of dual vectors but, as mentioned in [LW21], this simpler score function is sufficient for our cases of interest since the norm of the dual vector produced naturally concentrates around some value.

Note additionally that, when the coordinates of the error are distributed as a discrete Gaussian distribution and making the assumption that the $\langle \mathbf{y}, \mathbf{h} \rangle$ are realization of some independent variables, then this function is closely related to the optimal Neyman–Pearson test as shown by [LW21, Corollary 2 and Section 4.4] or by [GMW21, Appendix A] in a closely related context. Note that [EJK20] also showed that under these independence assumptions this score function allowed to make a good decision as long as

$$|\mathscr{W}| = \mathrm{poly}(n) / \varepsilon^2$$

where ε is the "bias" in the LWE case, namely $\varepsilon \stackrel{\text{def}}{=} \mathbb{E}\left(\cos\left(2\pi \left\langle \mathbf{e}, \mathbf{h} \right\rangle\right)\right)$. The dual interpretation of this score function (with weights) can be traced back to the previous duality formula given by [AR04], but note that it applies only in the very special case when the dual vectors are produced according to the distribution of the lemma.

8.2.5 Comparing modern dual attacks in code and lattice-based cryptography

Here our goal here is to derive a quick analogy between the recent dual attacks in lattices and the dual attacks we devised in this thesis. In particular, we disagree with the claim made in [PS24, Page 3 and Appendix A] that some of these recent lattice-based attacks and our code based attacks are fundamentally different: particularly we argue that the splitting strategy used [Alb17, EJK20, GJ21, MAT22, PS24] is exactly the same splitting strategy that we used in our attack and that can be traced back independently in lattice to the attack devised in [Alb17] (in some other context) and in codes to a remark made in [DT17a].

8.2.5.1 LWE and small LWE

Say $(\mathbf{G},\ \mathbf{y} = \mathbf{sG} + \mathbf{e})$ is an LWE problem in dimension k and with n samples as defined in Definition 47 with some secret $\mathbf{s}$ uniformly distributed in $\mathbb{Z}_q^k$. One can transform it into its syndrome variant $(\mathbf{H},\ \mathbf{b} = \mathbf{He}^\intercal)$ where $\mathbf{H}$ is a parity-check matrix of the code generated by $\mathbf{G}$. Additionally, by computing $\mathbf{H}$ in systematic form, say $\mathbf{H} = \begin{pmatrix}\mathbf{I}_{(n-k)} & \mathbf{A}\end{pmatrix}$ it is then possible to rewrite this last LWE problem as a "small secret" LWE sample $\left(\mathbf{A},\ \mathbf{b} = \mathbf{A}\mathbf{e}_{[\![n-k+1,\, n]\!]} + \mathbf{e}_{[\![1,\, n-k]\!]}\right)$, all these reductions are standard. In the case of Kyber and Dilithium, the problem are regularly given in this last form but this is completely equivalent to the first generator matrix form. In general one can transform a small LWE problem onto a standard generator matrix LWE problem as long as the coordinates of the error and the secret of that small LWE problem have the same distribution. In some other context such as for Homomorphic encryption [BGV12, CGGI20] however the situation is different because the small LWE problem considered is such that the secret and the error coordinates do not necessarily have the same distribution, and therefore we cannot directly transform it into a standard generator matrix LWE problem as given in Definition 47, but provided that we accept to tweak the previous definition by allowing that the coordinates of the error are not distributed identically this is not a problem, in essence what matters in the following discussion is that these coordinates are biased toward the small values of $\mathbb{Z}_q$.

8.2.5.2 Recent dual attacks

So, say we have an LWE problem $(\mathbf{G},\ \mathbf{y} = \mathbf{sG} + \mathbf{e})$ and denote by $\mathcal{C}$ the code generated by $\mathbf{G}$. Starting from [Alb17] in the small secret LWE context and then in [EJK20] for the standard LWE problem those attacks start by splitting the support $[\![1,\, n]\!]$ in two complementary parts $\mathscr{P}$ and $\mathscr{N}$ and reduce solving the original LWE problem to another LWE problem with decreased dimension but increased noise, the secret in all these variants is related to $\mathbf{e}_{\mathscr{P}}$. For example in [Alb17, EJK20, MAT22] this is done by computing dual vectors $\mathbf{h} \in \mathcal{C}^\perp$ which are of small norm on $\mathscr{N}$ (and arbitrary on $\mathscr{P}$) that yield the following new LWE sample

$$\left\langle \mathbf{y}, \mathbf{h} \right\rangle = \left\langle \mathbf{e}_{\mathscr{P}}, \mathbf{h}_{\mathscr{P}} \right\rangle + \left\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \right\rangle$$

where the secret is given by $\mathbf{e}_{\mathscr{P}}$ and the noise is given by $\langle \mathbf{e}_{\mathscr{N}}, \mathbf{h}_{\mathscr{N}} \rangle$. This introduces the same tradeoff that we benefited from in coding theory, namely as $|\mathscr{P}|$ increases, it allows decreasing the norm of $\mathbf{h}_{\mathscr{N}}$, hence naturally decreasing the number of required dual vectors needed to recover the secret, but on the contrary we now have to solve this new small LWE problem of dimension $|\mathscr{P}|$. In [Alb17, EJK20] this problem is solved by naively enumerating (with some tweaks to gain polynomial factors in [EJK20]) all the possible small candidates $\mathbf{r}$ for the secret $\mathbf{e}_{\mathscr{P}}$ and computing naively for each of them an associated score and keeping only the one whose associated score function is the highest. Problematically it is not efficient to use directly an FFT, costing $\widetilde{\mathcal{O}}\left(q^{|\mathscr{P}|}\right)$ operations, to compute the score in batch. Indeed in all these use cases because the secret $\mathbf{e}_{\mathscr{P}}$ is small and the modulus q large, in practice this leads to a huge number of useless computations. This is the reason why when the error is sufficiently small the naive enumeration of the small candidates prevails. In [GJ21] a new technique was introduced that allowed reducing the cost of the FFT step to $\widetilde{\mathcal{O}}\left(2^{|\mathscr{P}|}\right)$ operations by guessing only the coordinate of the secret $\mathbf{e}_{\mathscr{P}}$ modulo 2 at the cost of an increased noise. It was claimed there that this new attack reduced the security of Kyber and Dilithium. Later, a follow-up work by [MAT22] claimed to improve this last attack by achieving the same goal but with a different modulus switching technique.

8.2.6 Usual analysis of these dual attacks

The analysis of the simple distinguisher presented above relies on understanding the distribution of the score function $F_{\mathscr{W}}(\mathbf{y})$ where $\mathscr{W} \subset \Lambda_q(\mathcal{C})^{\vee}$ depending on whether $\mathbf{y}$ is uniform or $\mathbf{y}$ is planted and close to $\Lambda_q(\mathcal{C})$. Note that from now on we assume that $\mathbf{y}$ is distributed uniformly in $\mathbb{R}^n/\Lambda_q(\mathcal{C})$ instead of being uniformly distributed in $\mathbb{Z}_q^n$. Even though these attacks only deal with sublattices of q-ary lattices, [DP23b] made this simplifying assumption to stay general in their discussion, because q is large we do not expect that this changes anything in the conclusion.

The standard assumptions used to make the analysis of this dual attack and basically all subsequent newer dual attacks as [DP23b] points out is to assume that the terms in the sum

$$F_{\mathscr{W}}(\mathbf{y}) = \sum_{\mathbf{h} \in \mathscr{W}} \cos\left(2\pi \langle \mathbf{y}, \mathbf{h} \rangle\right)$$

are independent of each other. We focus in the following discussion only on the distribution when $\mathbf{y}$ is uniform.

Heuristic 2 (Independence heuristics (See [DP23b, Heuristic 3]))**.** *Let Λ be a full rank lattice of $\mathbb{R}^n$. Let $\mathscr{W} \subset \Lambda^{\vee}$ be a fixed set and let $\mathbf{y}$ be taken uniformly at random in $\mathbb{R}^n/\Lambda$ the random variables $(\langle \mathbf{h}, \mathbf{y} \rangle)_{\mathbf{h} \in \mathscr{W}}$ are mutually independent.*

Because the number N of dual vectors considered is big (exponential), one can now simply from this assumption compute exponential tail bounds for F in both cases. Because the number N of dual vectors considered is big (exponentially large), one can now simply from this assumption compute exponential tail bounds for F in both cases. For example, if $\mathbf{y} \sim \mathcal{U}(\mathbb{R}^n/\Lambda)$ then we can easily show that $\mathbb{E}_{\mathbf{y}}\left(\cos\left(2\pi \langle \mathbf{y}, \mathbf{h} \rangle\right)\right) = 0$ and $\mathbf{Var}_{\mathbf{y}}\left(\cos\left(2\pi \langle \mathbf{y}, \mathbf{h} \rangle\right)\right) = 1/2$ and derive, from the central limit theorem the following normal approximation (See for example [LW21, Lemma 6]).

Model 6 (Usual model in the uniform target case when analyzing dual attacks)**.** *Let Λ be a lattice. Let $\mathscr{W} \subset \Lambda^\vee$ be a set of N dual vectors and let $\mathbf{y} \sim \mathcal{U}(\mathbb{R}^n/\Lambda)$. The score function is modeled as*

$$F_{\mathscr{W}}(\mathbf{y}) \approx \mathcal{N}(0,\ N/2)\,.$$

8.2.6.1 Contradictory regime

[DP23b] note that there is a regime where those assumptions are clearly false as there is a small probability that $\mathbf{y}$ when taken uniformly at random in $\mathbb{R}^n/\Lambda$ ends up unusually close to Λ, say closer than what would typically be expected when $\mathbf{y}$ is an LWE instance. Clearly, this phenomenon is not captured by this assumption; see for example the following figure showing the survival function of the score function $\mathbb{P}(F_{\mathscr{W}}(\mathbf{y}) \geqslant x)$ when $\mathscr{W}$ is the output of a full sieve over $\Lambda^\vee$.

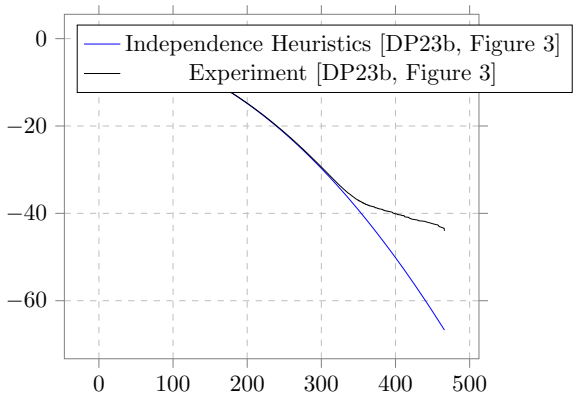


Figure 8.1: Survival function of the score function

The first zone where both curves match is called the waterfall zone and the second zone when $\mathbf{y}$ is unusually close to a vector of Λ is called the waterfloor zone, this zone is not well predicted by the independence assumption.

Now the key argument of [DP23b] is to notice that for dual attacks using a sieve, this contradictory regime appears in practice well before what we see in Fig. 8.1, which was obtained by performing a full sieve in $\Lambda^\vee$. Indeed, recall that for efficiency purposes, as stated above, sieving is in fact done in a sublattice $(\Lambda^\vee)' \subset \Lambda^\vee$ of much smaller rank. As a consequence

$$\langle \mathbf{y}, \mathbf{h} \rangle = \langle \mathbf{y}', \mathbf{h} \rangle$$

where $\mathbf{y}'$ is the orthogonal projection onto the space span $\left((\Lambda^\vee)'\right)$ and the question now is really to distinguish whether $\mathbf{y}'$ is close to a vector of the dual of $(\Lambda^\vee)'$. But now, because we are in a lower dimensional subspace, the probability that a uniform target is relatively closer to this lattice than an LWE target is much higher.

Coming back to the most recent attack [GJ21, MAT22] against LWE, [DP23b] argue their analysis reduces to some variant of the distinguishing problem we described above where one is given many uniform targets (the bad guesses coming from the splitting strategy) and one coming from an LWE instance and the attack succeeds if we can identify which comes from the LWE instance. [DP23b] then argue, looking at the parameters of the attack, that there

will be with overwhelming probability some uniform target lying closer to the true LWE instance.

8.2.6.2 Analyzing dual attacks when Gaussian sampling is used

Following the work of [DP23b], [PS24] proposed to use a Gaussian sampler [WL19] to produce dual vectors. Basically this procedure computes small vectors of the whole lattice Λ whose weight is distributed according to a certain Gaussian distribution (essentially as in the duality Proposition 58). This allows the suppression of the issue that comes from producing dual vectors in a sublattice. With this and using a slight variation of the duality formula of Proposition 58, [PS24] is able to fully prove some simplified variant of the most recent dual attacks (essentially [MAT22] but by dropping modulus switching to make the analysis tractable). Using the Gaussian sampler and sticking to a simpler variant does not make the attack competitive but offers a theoretical framework for a rigorous analysis.

8.3 Contribution: predicting the behavior of the score function

In this contribution section we predict the behavior of the score function that was shown to be badly predicted by the traditional independence assumptions, see Fig. 8.1.

Concurrent work. Note that, concurrently and independently to this work published in [CDMT24] (in a slightly different form), [DP23a] posted a preprint predicting the behavior of the score function using essentially the same tools, a.k.a Bessel function appears, but with a different reasoning. Their work is more in depth. For example, we only focus here only on predicting the behavior of the score function when $\mathbf{y}$ is uniform in $\mathbb{R}^n/\Lambda$.

8.3.1 Model

Say for now and to simplify that we want to predict the behavior of the score function $F_{\widetilde{\mathscr{W}}}(\mathbf{y})$ when all the dual vectors of norm smaller than a certain w are considered. Note that in practice a full sieve does not return all the vectors of a lattice in an Euclidean Ball but is often parametrized to return about a constant fraction of them, we will take that into account later. Our reasoning is based on the following duality formula for making intervene Bessel functions of the first kind. It is similar to the Poisson duality Proposition 58 but tailored to the output distribution considered here.

Fact 35. *We have that*

$$\sum_{\mathbf{h}\in\Lambda^\vee\,:\,||\mathbf{h}||\leqslant w}\cos\left(2\pi\left\langle\mathbf{y},\mathbf{h}\right\rangle\right)=\sum_{\mathbf{h}\in\Lambda^\vee}e^{-2i\pi\langle\mathbf{y},\mathbf{h}\rangle}\mathbf{1}_{\mathsf{Ball}_w^n}\left(\mathbf{h}\right)$$

Proof. By definition of cos, $\sum_{\mathbf{h}\in\Lambda^\vee\,:\,||\mathbf{h}||\leqslant w}\cos\left(2\pi\left\langle\mathbf{y},\mathbf{h}\right\rangle\right)=(1/2)\sum_{\mathbf{h}\in\Lambda^\vee\,:\,||\mathbf{h}||\leqslant w}e^{-2i\pi\langle\mathbf{y},\mathbf{h}\rangle}+e^{2i\pi\langle\mathbf{y},\mathbf{h}\rangle}$. The result follows by using the fact that $\mathbf{h}\in\Lambda\Leftrightarrow-\mathbf{h}\in\Lambda^\vee$. □

This can be seen as the lattice-based counterpart of the duality formula Proposition 39 we devised in coding theory that makes intervene Krawtchouk polynomials. Recalling, for

a Schwartz function (that is sufficiently regular, in particular continuous) f, the Poisson summation formula [SW71, Chapter VII, Corollary 2.6] states that

$$\sum_{\mathbf{h}\in\Lambda^\vee} e^{-2i\pi\langle\mathbf{y},\mathbf{h}\rangle} f(\mathbf{h}) = \frac{1}{\mathcal{V}(\Lambda^\vee)} \sum_{\mathbf{c}\in\Lambda+\mathbf{y}} \widehat{f}(\mathbf{c}). \tag{8.3}$$

Now our function of interest here is the indicator of an Euclidean ball whose Fourier transform classically relate to Bessel function of the first kind.

Fact 36 ([DDRT23, Fact 4.11])**.** *For any $\mathbf{x}\in\mathbb{R}^n$ we have that*

$$\widehat{\mathbf{1}_{\mathsf{Ball}_w^n}}(\mathbf{y}) = \left(\frac{w}{||\mathbf{y}||}\right)^{n/2} J_{n/2}(2\pi w\,||\mathbf{y}||)$$

where $\mathsf{Ball}_w^n \stackrel{def}{=} \{\mathbf{x}\in\mathbb{R}^n : ||\mathbf{x}|| \leqslant w\}$ and where the Fourier transform of $f:\mathbb{R}^n\to\mathbb{C}$ is defined as $\widehat{f}(\mathbf{x}) \stackrel{def}{=} \int_{\mathbb{R}^n} f(\mathbf{z})\, e^{-2i\pi\langle\mathbf{x},\mathbf{z}\rangle} d\mathbf{z}$ and where J_n is the Bessel function of the first kind of order n.

In particular, because the indicator is not continuous we cannot directly apply Poisson summation formula, we could convolute it with a concentrated Gaussian as done in [Pin08, Section 4.5.2] at the cost of a small correction factor, but we rather simply make the approximation that we have the equality.

Approximation 1. *Let $n\in\mathbb{N}$ and $w\in\mathbb{R}$. Let Λ be a lattice in $\mathbb{R}^n$ and let $\widetilde{\mathscr{W}} \stackrel{def}{=} \Lambda^\vee \bigcap \{\mathbf{x}\in\mathbb{R}^n : ||\mathbf{x}|| \leqslant w\}$ and let $F_{\widetilde{\mathscr{W}}}(\mathbf{y}) \stackrel{def}{=} \sum_{\mathbf{w}\in\widetilde{\mathscr{W}}} \cos(2\pi\langle\mathbf{y},\mathbf{h}\rangle)$. We make the approximation that*

$$F_{\widetilde{\mathscr{W}}}(\mathbf{y}) = \frac{1}{Vol(\Lambda^\vee)} \sum_{\mathbf{x}\in\Lambda+\mathbf{y}} \left(\frac{w}{||\mathbf{x}||}\right)^{n/2} J_{n/2}(2\pi w\,||\mathbf{x}||)$$

where J_n is the Bessel function of the first kind of order n.

Very generally we argue heuristically that when Λ or $\mathbf{y}$ are random this score function can be modeled as the sum of two simple random variables: first, the term of the sum accounting for the smallest vector in $\Lambda+\mathbf{y}$, this accounts for the closest lattice vector to $\mathbf{y}$. Second there are some small variations around it accounting for the normal like behavior of the rest of the sum.

One of the rationale for the following model stems from assuming that when computed by a full sieve, the N dual vectors are produced uniformly at random in a certain ball intersected with the lattice. We give more details on how this model was precisely derived in Section 8.3.4.

Model 7. *Let Λ be some random full-rank lattice of $\mathbb{R}^n$ of volume V. Let $\mathscr{W}$ be taken uniformly at random from the subset of $\{\mathbf{x}\in\mathbb{R}^n : ||\mathbf{x}|| \leqslant w\}$ of size N. Let $\mathbf{y}\sim\mathcal{U}(\mathbb{R}^n/\Lambda)$. We make the model that*

$$F_{\mathscr{W}}(\mathbf{y}) = X_{\text{floor}} + X_{\text{fall}}$$

where

$$X_{\text{floor}} = N\sqrt{n\pi}\left(\frac{n}{2\pi e w x_{\min}}\right)^{n/2} J_{n/2}(2\pi w x_{\min})$$
$$X_{\text{fall}} \sim \mathcal{N}(0,\ N/2)$$

and where

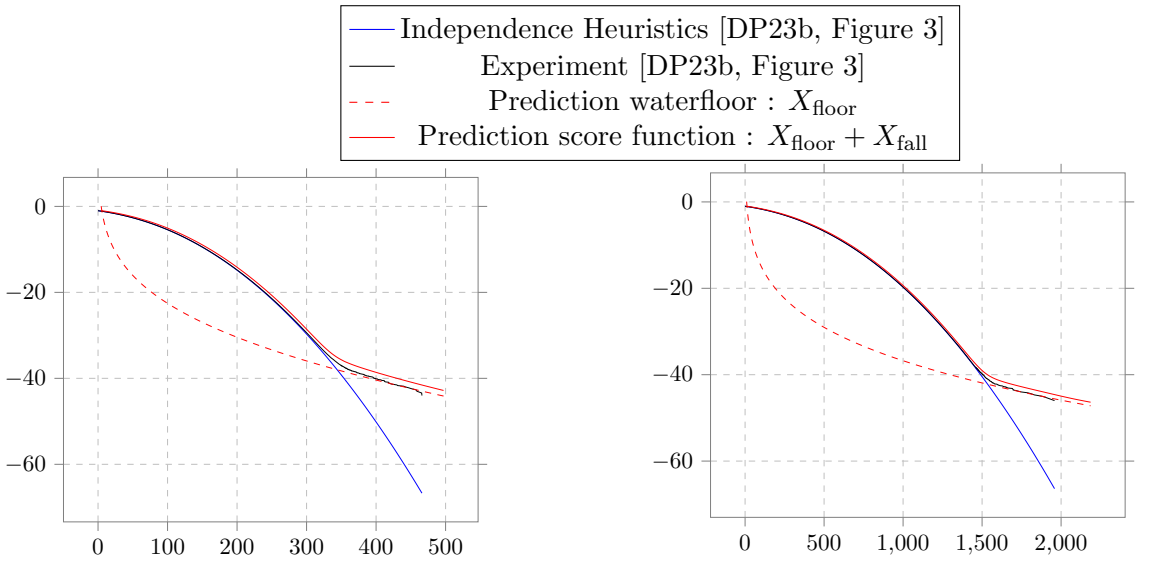
$$x_{\min}^n \sim \text{Exponential}\left(\frac{V}{\mathcal{V}\left(\mathsf{Ball}_1^n\right)}\right)$$

where we recall that $\mathcal{V}\left(\mathsf{Ball}_1^n\right) = \pi^{n/2}/\Gamma\left(1+n/2\right)$.

We show in Fig. 8.2 that this model seems to reasonably predict the experimental survival function of the score function in the uniform target setting that was observed by [DP23b, Figure 3]. We give here some details on how we instantiate the model to make the actual prediction. First in [DP23b, Figure 3] the lattice Λ (In [DP23b] this lattice is called Λ') in which the small weight dual vectors are computed is in fact a sparsification (this was done as in [GJ21]) of a random q-ary lattice coming from the original LWE problem. The only important remark here is that Λ is random but has a fixed volume V that can either easily be computed from the parameters or directly from its basis $\mathbf{B}$ given in the experimental results of [DP23b].

8.3.2 Results

Second, we reused the datasets of [DP23b, Figure 3] and extracted the number of dual vectors N and heuristically chose w as the mean length of the dual vectors experimentally obtained. This is only a rough estimate that does not exactly correspond to what w means in our model but we see in practice that it is sufficient. In the next chapter we will refine this much more and model w theoretically.



(a) $n = 60$, $w = 0.0320$, $q = 3329$, $N = 5040$, $V = 6.12 \times 10^{110}$

(b) $n = 80$, $w = 0.0376$, $q = 3329$, $N = 89494$, $V = 8.19 \times 10^{146}$

Figure 8.2: Prediction of the score function in the uniform target case.

8.3.3 Discussion

The reason why Bessel functions did not appear in previous analysis was because the bias of $\langle \mathbf{y}, \mathbf{h} \rangle$ was computed over the choice of $\mathbf{y}$ and $\mathbf{h}$ was considered as a fixed vector. Heuristically

because $\mathbf{y}$ is random but fixed once and for all when computing the score function it seems more natural to consider $\mathbf{h}$ as the random vector. In that case say $\mathbf{h}$ is taken uniformly at random in Ball_w^n (Or in $\mathsf{Ball}_w^n \bigcap \Lambda^\vee$ for a random lattice Λ with the right distribution) then the bias of the main quantity is

$$\mathbb{E}_{\mathbf{h}}\left(\cos\left(2\pi\left\langle \mathbf{y},\mathbf{h}\right\rangle\right)\right)=\frac{\widehat{\mathbf{1}_{\mathsf{Ball}_w^n}}\left(\mathbf{y}\right)}{\mathcal{V}\left(\mathsf{Ball}_w^n\right)}$$

and where the Fourier transform is defined and given in Fact 36 and makes intervene Bessel functions. This is the lattice-based equivalent of Krawtchouk polynomials intervening in the bias of $\langle \mathbf{y},\mathbf{h}\rangle$ in coding theory. Indeed, recalling, we have that when $\mathbf{h}$ is taken uniformly in $\mathcal{S}_w^n \stackrel{\text{def}}{=} \{\ \mathbf{x}\in\mathbb{F}_2^n\ :\ |\mathbf{x}|=w\}$ the Hamming sphere of radius w, then

$$\mathbb{E}\left((-1)^{\langle \mathbf{y},\mathbf{h}\rangle}\right)=\frac{\widehat{\mathbf{1}_{\mathcal{S}_w^n}}\left(\mathbf{y}\right)}{|\mathcal{S}_w^n|}=\frac{K_w^{(n)}\left(t\right)}{\binom{n}{w}}$$

where $\widehat{f}\left(\mathbf{y}\right)\stackrel{\text{def}}{=}\sum_{\mathbf{x}\in\mathbb{F}_2^n} f(\mathbf{x})(-1)^{\langle \mathbf{x},\mathbf{y}\rangle}$ is the discrete Fourier transform here.

8.3.4 Justification for the model

We achieve this model through a series of approximations. First we make the approximation that the score function is equal to its conditional expectation where the expected value is taken over the choice of $\mathscr{W}$ (see Model 7) that is:

$$F_{\mathscr{W}}\left(\mathbf{y}\right)\approx\mathbb{E}_{\mathscr{W}}\left(F_{\mathscr{W}}\left(\mathbf{y}\right)\right)$$

where $\widetilde{\mathscr{W}}\stackrel{\text{def}}{=}\Lambda^\vee\bigcap\{\mathbf{x}\in\mathbb{R}^n\ :\ ||\mathbf{x}||\leqslant w\ \}$. This allows us to write that

$$F_{\mathscr{W}}\left(\mathbf{y}\right)\approx\frac{N}{\left|\widetilde{\mathscr{W}}\right|}F_{\widetilde{\mathscr{W}}}(\mathbf{y})\,.$$

Now, using the Gaussian Heuristic Heuristic 1 we heuristically estimate that

$$\left|\widetilde{\mathscr{W}}\right|\approx\frac{w^n}{\mathcal{V}\left(\Lambda^\vee\right)}\frac{(2\pi e)^{n/2}}{\sqrt{n\pi}n^{n/2}}$$

Now, using the fact that we always have that

$$\mathcal{V}\left(\Lambda\right)=\frac{1}{\mathcal{V}\left(\Lambda^\vee\right)}$$

and using all those previous approximations with the fact that from Approximation 1 we have that

$$F_{\widetilde{\mathscr{W}}}(\mathbf{y})\approx\frac{1}{\mathcal{V}\left(\Lambda^\vee\right)}\sum_{\mathbf{x}\in\Lambda+\mathbf{y}}\left(\frac{w}{||\mathbf{x}||}\right)^{n/2}J_{n/2}\left(2\pi w\,||\mathbf{x}||\right)$$

we get the following approximation for F:

$$F_{\mathscr{W}}\left(\mathbf{y}\right)\approx N\sqrt{n\pi}\sum_{\mathbf{x}\in\Lambda+\mathbf{y}}\left(\frac{n}{2\pi e w\,||\mathbf{x}||}\right)^{n/2}J_{n/2}\left(2\pi w\,||\mathbf{x}||\right).$$

Prediction of the waterfloor (X_{floor}) Basically we argue that the waterfloor phenomenon can be encompassed by considering the smallest vector of $\Lambda + \mathbf{y}$ and considering the term in the sum related to the smallest vector

$$N\sqrt{n\pi}\left(\frac{n}{2\pi e w x_{\min}}\right)^{n/2} J_{n/2}\left(2\pi w x_{\min}\right)$$

where

$$x_{\min} \stackrel{\text{def}}{=} \min_{\lambda\in\Lambda} ||\lambda + \mathbf{y}|| .$$

We make the classical model that $x_{\min}$ follows an exponential distribution of good expected value. This comes from the fact that, refining the Gaussian heuristic slightly we can actually make the model that the number of lattice point inside an Euclidean ball follows a Poisson distribution of right expected value:

$$\left|(\Lambda \setminus \{\ \mathbf{0}\}) \bigcap \mathsf{Ball}_t^n\right| \sim \text{Poisson}\left(\frac{\mathcal{V}\left(\mathsf{Ball}_t^n\right)}{\mathcal{V}\left(\Lambda\right)}\right).$$

This model is motivated by the fact it was shown in [Rog56] that this quantity indeed weakly converges to this Poisson distribution for random lattices in the sense of [Sie45]. With this model, it is clear that the survival function of the shortest vector can be written as follows:

$$\begin{aligned}
\mathbb{P}\left(x_{\min}^n > z^n\right) &= \mathbb{P}\left(x_{\min} > z\right) \\
&= \mathbb{P}\left((\Lambda \setminus \{\ \mathbf{0}\}) \bigcap \mathsf{Ball}_z^n = 0\right) \\
&= e^{-\frac{\mathcal{V}\left(\mathsf{Ball}_z^n\right)}{\mathcal{V}(\Lambda)}} \\
&= e^{-\left(\frac{\mathcal{V}\left(\mathsf{Ball}_1^n\right)}{\mathcal{V}(\Lambda)}\right) z^n}.
\end{aligned}$$

This is exactly the survival function of an exponential distribution, hence we model the floor as

$$x_{\min}^n \sim \text{Exponential}\left(\frac{\mathcal{V}\left(\Lambda\right)}{\mathcal{V}\left(\mathsf{Ball}_1^n\right)}\right).$$

Prediction of the waterfall (X_{fall}) We observe in practice that the waterfall zone is well-predicted by the independence heuristic, hence in that zone we keep the standard normal model

$$X_{\text{fall}} \sim \mathcal{N}\left(0,\ N/2\right).$$

Predicting the whole score function A natural idea to predict the experimental curve on the whole support is therefore to take the convolution of these two distributions X_{floor} and X_{fall}. Indeed, we observe by comparing our model to the experiments that for any support point, there is always one distribution which exponentially dominates the other one.

Chapter 9

Assessing the impact of a variant of MATZOV's dual attack

Summary

The dual attacks on the Learning With Errors (LWE) problem are currently a subject of controversy. In particular, the results of [MAT22], which claim to significantly lower the security level of Kyber [SAB+20], a lattice-based cryptosystem currently being standardized by NIST, are not widely accepted. The analysis behind their attack depends on a series of assumptions that, in certain scenarios, have been shown to contradict established theorems or well-tested heuristics [DP23b].

In this chapter, we introduce a new dual lattice attack on LWE, drawing from ideas in coding theory. Our approach revisits the dual attack proposed by [MAT22], replacing modulus switching with an efficient decoding algorithm. This decoding is achieved by using polar codes over $\mathbb{Z}_q$, and we confirm their strong distortion properties through benchmarks. This modification enables a reduction from small-LWE to plain-LWE, with a notable decrease in the secret dimension. Additionally, we replace the enumeration step in the attack by assuming the secret is zero for the portion being enumerated, iterating this assumption over various choices for the enumeration part.

We make an analysis of our attack in the spirit of what was done in [CDMT24].

Lastly, we assess the complexity of our attack on Kyber by showing that the security levels for Kyber-512/768/1024 are 3.5/11.9/12.3 bits below the NIST requirements (143/207/272 bits) in the same nearest-neighbor cost model as in [SAB+20, MAT22]. All in all the cost of our attack matches and even slightly beat in some cases the complexities originally claimed by the attack of [MAT22].

Disclaimer 1. *This chapter is exactly the paper [CMST25] with no rewriting. In particular, in contrast with the rest of this thesis, the vectors in this chapter are column vectors.*

Contents

9.1 Introduction

9.1.1 Background

9.1.1.1 The LWE problem

The Learning With Errors (LWE) problem originally introduced by Regev in [Reg05]. It revolves around the task of finding $\mathbf{s} \in \mathbb{Z}_q^n$ given $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ with $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ where $\mathbf{e}$ is of small Euclidean norm. This problem can be seen as the decoding problem for the code $\mathcal{C}(\mathbf{A})$ generated by the columns of $\mathbf{A}$ (i.e. $\mathcal{C}(\mathbf{A}) \stackrel{\text{def}}{=} \{\mathbf{A}\mathbf{x},\ \mathbf{x} \in \mathbb{Z}_q^n\}$) and the Euclidean distance, where we are asked to find the codeword (i.e. the element of $\mathcal{C}$) which is close in Euclidean distance to $\mathbf{b}$. We are particularly interested in the case where $\mathbf{s}$ is short too which is called the small LWE problem. This problem has emerged as a fundamental challenge in cryptography. Notably, it underpins the construction of various cryptographic primitives and is conjectured to withstand attacks from quantum computers [LPR10]. Our motivation for exploring this problem stems particularly from the need to gauge the security level of KYBER, a lattice-based cryptosystem that is being standardized by the NIST[1].

9.1.1.2 Dual attacks

The most efficient cryptanalysis techniques against LWE(-like) problems are "primal" and "dual" lattice attacks. The primal attack corresponds to lattice reduction being performed on the "primal" lattice $\Lambda_q(\mathbf{A})$ which is Construction A of the q-ary lattice obtained from $\mathcal{C}(\mathbf{A})$, namely

$$\begin{aligned} \Lambda_q(\mathbf{A}) &= \{\mathbf{y} \in \mathbb{Z}^m \ :\ \exists \mathbf{c} \in \mathcal{C}(\mathbf{A}) \text{ such that } \mathbf{y} = \mathbf{c} \mod q\} \\ &= \{\mathbf{y} \in \mathbb{Z}^m \ :\ \exists \mathbf{s} \in \mathbb{Z}_q^n \text{ such that } \mathbf{y} = \mathbf{A}\mathbf{s} \mod q\}. \end{aligned}$$

Dual attacks mean that lattice reduction is performed over the dual lattice $\Lambda_q(\mathbf{A})^\vee$, which in this case, up to a q-multiplicative factor, is nothing but Construction A applied to the dual code $\mathcal{C}(\mathbf{A})^\perp \stackrel{\text{def}}{=} \{\mathbf{x} \in \mathbb{Z}_q^m \ :\ \mathbf{A}^\intercal \mathbf{x} = \mathbf{0}\}$:

$$\begin{aligned} \Lambda_q(\mathbf{A})^\vee &\stackrel{\text{def}}{=} \{\mathbf{x} \in \mathbb{R}^m \ :\ \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z},\ \forall \mathbf{y} \in \Lambda_q(\mathbf{A})\} \\ &= \tfrac{1}{q}\{\mathbf{x} \in \mathbb{Z}^m \ :\ \mathbf{A}^\intercal \mathbf{x} = \mathbf{0} \mod q\}. \end{aligned}$$

The lattice $\{\mathbf{x} \in \mathbb{Z}^m \ :\ \mathbf{A}^\intercal \mathbf{x} = \mathbf{0} \mod q\}$ is known under the name of the orthogonal lattice of $\mathbf{A}$, i.e. $\Lambda_q^\perp(\mathbf{A}) \stackrel{\text{def}}{=} \{\mathbf{x} \in \mathbb{Z}^m \ :\ \mathbf{A}^\intercal \mathbf{x} = \mathbf{0} \mod q\}$.

Dual attacks were introduced in [AR04]. In its simplest form, a dual attack is a distinguisher attack which is given either $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$ or $(\mathbf{A}, \mathbf{u})$ where $(\mathbf{A}, \mathbf{u})$ are uniform and $\mathbf{e}$ is short, and answers if we are in the first or the second case. It starts by computing many short $\mathbf{x}_j$'s in the dual lattice $\Lambda_q(\mathbf{A})^\vee$ and the associated $\langle \mathbf{x}_j, \mathbf{b} \rangle$'s. Those short vectors are obtained by lattice reduction of $\Lambda_q(\mathbf{A})^\vee$ or what is basically equivalent $\Lambda_q^\perp(\mathbf{A})$. In the second case, we expect that these scalar products are uniformly distributed in $\mathbb{Z}_q$. On the other hand in the first case since $\langle \mathbf{x}_j, \mathbf{b} \rangle = \langle \mathbf{x}_j, \mathbf{A}\mathbf{s} + \mathbf{e} \rangle = \langle \mathbf{A}^\intercal \mathbf{x}_j, \mathbf{s} \rangle + \langle \mathbf{x}_j, \mathbf{e} \rangle = \langle \mathbf{x}_j, \mathbf{e} \rangle \mod q$, we get scalar products of small vectors which are tilted towards small entries.

There have been a sequence of developments in dual attacks, for instance by combining these attacks with a guessing stage [Alb17] consisting in splitting the support of $\mathbf{s}$ in two

[1] `https://csrc.nist.gov/projects/post-quantum-cryptography`

parts $\begin{pmatrix}\mathbf{s}_{\mathsf{enu}}\\ \mathbf{s}_{\mathsf{lat}}\end{pmatrix}$ and guessing one part of this support. $\mathbf{A}$ is split accordingly $\mathbf{A} = [\mathbf{A}_{\mathsf{enu}} \quad \mathbf{A}_{\mathsf{lat}}]$. This allows to only perform lattice reduction on Construction A of the code generated by $\begin{bmatrix}\mathbf{I}_m\\ \mathbf{A}_{\mathsf{lat}}^\intercal\end{bmatrix}$. That means we only look for short vectors $\begin{pmatrix}\mathbf{x}\\ \mathbf{y}_{\mathsf{lat}}\end{pmatrix}$ such that $\mathbf{A}_{\mathsf{lat}}{}^\intercal\mathbf{x} = \mathbf{y}_{\mathsf{lat}} \mod q$. Define $\mathbf{y}_{\mathsf{enu}}$ by $\mathbf{y}_{\mathsf{enu}} = \mathbf{A}_{\mathsf{enu}}^\intercal\mathbf{x}$. The point is that looking for such vectors is faster because of the dimension reduction. We then guess $\mathbf{s}_{\mathsf{enu}}$ and check whether the $\langle\mathbf{x},\mathbf{b}\rangle - \langle\mathbf{y}_{\mathsf{enu}},\mathbf{s}_{\mathsf{enu}}\rangle$'s are tilted towards small values or not. This comes from the fact that

$$\begin{aligned}\langle\mathbf{x},\mathbf{b}\rangle &= \langle\mathbf{x},\mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}} + \mathbf{A}_{\mathsf{lat}}\mathbf{s}_{\mathsf{lat}} + \mathbf{e}\rangle = \langle\mathbf{A}_{\mathsf{enu}}^\intercal\mathbf{x},\mathbf{s}_{\mathsf{enu}}\rangle + \langle\mathbf{A}_{\mathsf{lat}}^\intercal\mathbf{x},\mathbf{s}_{\mathsf{lat}}\rangle + \langle\mathbf{x},\mathbf{e}\rangle\\ &= \langle\mathbf{y}_{\mathsf{enu}},\mathbf{s}_{\mathsf{enu}}\rangle + \langle\mathbf{y}_{\mathsf{lat}},\mathbf{s}_{\mathsf{lat}}\rangle + \langle\mathbf{x},\mathbf{e}\rangle \mod q \end{aligned} \tag{9.1}$$

In [EJK20] this was generalized to broader secret distributions paired with additional improvements on the exhaustive search. [GJ21] applied a Fast Fourier Transform-style algorithm to the search over $\mathbf{s}_{\mathsf{enu}}$ and the search space is significantly reduced by roughly considering only the most significant bits of $\mathbf{s}_{\mathsf{enu}}$. [MAT22] replaced this step with "modulus switching" [BV11, AFFP14], yielding significant performance gains. The algorithm can be described as follows.

The MATZOV template. Let $n_{\mathsf{enu}}, n_{\mathsf{fft}}, n_{\mathsf{lat}}$ be some positive integers such that $n_{\mathsf{enu}} + n_{\mathsf{fft}} + n_{\mathsf{lat}} = n$. The matrix $\mathbf{A}$ and the secret $\mathbf{s}$ are divided accordingly:

$$\mathbf{A} \stackrel{\text{def}}{=} \begin{bmatrix}\mathbf{A}_{\mathsf{enu}} & \mathbf{A}_{\mathsf{fft}} & \mathbf{A}_{\mathsf{lat}}\end{bmatrix} \in \mathbb{Z}_q^{m\times n_{\mathsf{enu}}} \times \mathbb{Z}_q^{m\times n_{\mathsf{fft}}} \times \mathbb{Z}_q^{m\times n_{\mathsf{lat}}}, \tag{9.2}$$

$$\mathbf{s} \stackrel{\text{def}}{=} \begin{pmatrix}\mathbf{s}_{\mathsf{enu}}\\ \mathbf{s}_{\mathsf{fft}}\\ \mathbf{s}_{\mathsf{lat}}\end{pmatrix} \in \mathbb{Z}_q^{n_{\mathsf{enu}}} \times \mathbb{Z}_q^{n_{\mathsf{fft}}} \times \mathbb{Z}_q^{n_{\mathsf{lat}}}. \tag{9.3}$$

The objective is to guess the part $\mathbf{s}_{\mathsf{enu}}$ of the secret vector but to do that, we need to be able to distinguish between a right and a wrong guess. By performing lattice reduction on the lattice generated by $\begin{bmatrix}\mathbf{I}_m & \mathbf{0}\\ \mathbf{A}_{\mathsf{lat}}^\intercal & q\mathbf{I}_{n_{\mathsf{lat}}}\end{bmatrix}$, we obtain a set $\mathscr{S}$ of vectors $\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}$ such that $\mathbf{x}$ and $\mathbf{y}$ are of small Euclidean norm in $\mathbb{Z}_q^m$ and $\mathbb{Z}_q^{n_{\mathsf{lat}}}$ respectively and are such that $\mathbf{y} = \mathbf{A}_{\mathsf{lat}}^\intercal\mathbf{x}$. Similarly to Equation (9.1), we have

$$\langle\mathbf{x},\mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}}\rangle - \left\langle\mathbf{A}_{\mathsf{fft}}^\intercal\mathbf{x},\mathbf{s}_{\mathsf{fft}}\right\rangle = \langle\mathbf{x},\mathbf{e}\rangle + \langle\mathbf{y},\mathbf{s}_{\mathsf{lat}}\rangle\,. \tag{9.4}$$

Since $(\mathbf{x},\mathbf{y})$ and $(\mathbf{e},\mathbf{s}_{\mathsf{lat}})$ are short, Equation (9.4) is biased towards zero, raising the issue of finding efficiently $\mathbf{s}_{\mathsf{enu}}$ and $\mathbf{s}_{\mathsf{fft}}$ such that the left-hand term in (9.4) is small.

One can notice that (9.4) actually gives us many small-LWE samples[2] $(\mathbf{a}',b') \in \mathbb{Z}_q^{n_{\mathsf{fft}}} \times \mathbb{Z}_q$ with secret $\mathbf{s}' \in \mathbb{Z}_q^{n_{\mathsf{fft}}}$ and error $e' \in \mathbb{Z}_q$ where

$$\mathbf{a}' \stackrel{\text{def}}{=} \mathbf{A}_{\mathsf{fft}}^\intercal\mathbf{x}, \tag{9.5}$$

$$b' \stackrel{\text{def}}{=} \langle\mathbf{x},\mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}}\rangle\,, \tag{9.6}$$

$$\mathbf{s}' \stackrel{\text{def}}{=} \mathbf{s}_{\mathsf{fft}}, \tag{9.7}$$

$$e' \stackrel{\text{def}}{=} \langle\mathbf{x},\mathbf{e}\rangle + \langle\mathbf{y},\mathbf{s}_{\mathsf{lat}}\rangle\,. \tag{9.8}$$

[2] By using N samples $\{(\mathbf{a}'_i,b'_i)\}_{i\in[\![1,N]\!]}$, we can rewrite the LWE instance with a matrix $\mathbf{A}' \stackrel{\text{def}}{=} \begin{bmatrix}\mathbf{a}'_1 & \cdots & \mathbf{a}'_N\end{bmatrix}$ and a vector $\mathbf{b}' \stackrel{\text{def}}{=} (b'_1 \cdots b'_N)^\intercal$.

Thus, we can distinguish a correct and an incorrect guess for $\mathbf{s}_{\mathsf{enu}}$ and recover $\mathbf{s}_{\mathsf{fft}}$ by solving this small-LWE instance. Since n_{enu} is small enough, $\mathbf{s}_{\mathsf{enu}}$ can be exhaustively searched. Once $\mathbf{s}_{\mathsf{enu}}$ is identified, $\mathbf{s}_{\mathsf{fft}}$ is recovered by solving the above small-LWE instance. Specifically, this involves exhaustively searching for a $\mathbf{z} \in \mathbb{Z}_q^{n_{\mathsf{fft}}}$ such that the vector $\left(\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}}\rangle - \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}, \mathbf{z}\right\rangle\right)_{(\mathbf{x},\mathbf{y})\in\mathscr{S}}$ has a small Euclidean norm. One approach to speed up the search is to use a Fourier transform. The method introduces an evaluation function E, which assigns a real value to each guess $\widetilde{\mathbf{s}_{\mathsf{enu}}} \in \mathbb{Z}_q^{n_{\mathsf{enu}}}$:

$$E(\widetilde{\mathbf{s}_{\mathsf{enu}}}) \stackrel{\text{def}}{=} \max_{\mathbf{z}\in\mathbb{Z}_q^{n_{\mathsf{fft}}}} F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{z}) \tag{9.9}$$

where

$$F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{z}) \stackrel{\text{def}}{=} \sum_{(\mathbf{x},\mathbf{y})\in\mathscr{S}} \cos\left(\tfrac{2\pi}{q}\left(\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\rangle - \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}, \mathbf{z}\right\rangle\right)\right). \tag{9.10}$$

MATZOV's algorithm essentially consists in finding $\widetilde{\mathbf{s}_{\mathsf{enu}}} \in \mathbb{Z}_q^{n_{\mathsf{enu}}}$ such that $E(\widetilde{\mathbf{s}_{\mathsf{enu}}}) \geqslant T$, where $T \in \mathbb{N}$ is a threshold chosen around the expected value of E when evaluated on the correct guess, namely $T \approx E(\mathbf{s}_{\mathsf{enu}})$. The key idea is that we expect $E(\mathbf{s}_{\mathsf{enu}})$ to be significantly larger than $E(\widetilde{\mathbf{s}_{\mathsf{enu}}})$ when $\widetilde{\mathbf{s}_{\mathsf{enu}}}$ is incorrect, i.e. when $\widetilde{\mathbf{s}_{\mathsf{enu}}} \neq \mathbf{s}_{\mathsf{enu}}$. Indeed, for the correct guess, Equation (9.4) shows that $\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}}\rangle - \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}, \mathbf{s}_{\mathsf{fft}}\right\rangle \bmod q$ is biased toward zero. Consequently, each term in the sum in Equation (9.10) will be biased toward 1, resulting in a large total value for $\mathbb{E}\left(F_{\mathbf{s}_{\mathsf{enu}}}(\mathbf{s}_{\mathsf{fft}})\right)$. On the other hand, if $\widetilde{\mathbf{s}_{\mathsf{enu}}} \neq \mathbf{s}_{\mathsf{enu}}$, then for all $\mathbf{z} \in \mathbb{Z}_q^{n_{\mathsf{fft}}}$, $\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\rangle - \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}, \mathbf{z}\right\rangle \bmod q$ is uniformly distributed over $\mathbb{Z}_q$, meaning that $\mathbb{E}\left(F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{z})\right) = 0$.

Modulus switching. The point of using the evaluation function E is that $F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}$ can be computed efficiently with a Fast Fourier Transform (FFT). However, the large input space $\mathbb{Z}_q^{n_{\mathsf{fft}}}$ makes the FFTcostly. This size can be reduced, though at the cost of slightly weakening the bias of $F_{\mathbf{s}_{\mathsf{enu}}}(\mathbf{s}_{\mathsf{fft}})$.

To do that, MATZOV proposes to reduce the size of the field $\mathbb{Z}_q$ by using a modulus switching technique: instead of considering $F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}$, they consider

$$\begin{array}{rccc} F^{(\mathsf{ms})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}: & \mathbb{Z}_p^{n_{\mathsf{fft}}} & \longrightarrow & \mathbb{R} \\ & \mathbf{z} & \longmapsto & \displaystyle\sum_{(\mathbf{x},\mathbf{y})\in\mathscr{S}} \cos\left(\tfrac{2\pi}{p}\left(\tfrac{p}{q}\left\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\right\rangle - \left\langle \left\lfloor \tfrac{p}{q}\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}\right\rceil, \mathbf{z}\right\rangle\right)\right) \end{array}$$

where $p \leqslant q$ is a smaller modulus and $\lfloor\cdot\rceil$ stands for the integer rounding operation. Thus, for the wrong guess $\widetilde{\mathbf{s}_{\mathsf{enu}}} \neq \mathbf{s}_{\mathsf{enu}}$ and for all $\mathbf{z} \in \mathbb{Z}_p^{n_{\mathsf{fft}}}$, $F^{(\mathsf{ms})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{z})$ is still expected to be 0 whereas for the good guess $\mathbf{s}_{\mathsf{enu}}$ we have, from Equation (9.4),

$$\begin{aligned} F^{(\mathsf{ms})}_{\mathbf{s}_{\mathsf{enu}}}(\mathbf{s}_{\mathsf{fft}} \bmod p) &= \sum_{(\mathbf{x},\mathbf{y})\in\mathscr{S}} \cos\left(\tfrac{2\pi}{p}\left(\tfrac{p}{q}\langle \mathbf{x}, \mathbf{e}\rangle + \tfrac{p}{q}\langle \mathbf{y}, \mathbf{s}_{\mathsf{lat}}\rangle + \left\langle \tfrac{p}{q}\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \left\lfloor \tfrac{p}{q}\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}\right\rceil, \mathbf{s}_{\mathsf{fft}}\right\rangle\right)\right) \\ &= \sum_{(\mathbf{x},\mathbf{y})\in\mathscr{S}} \cos\left(\tfrac{2\pi}{q}\left(\langle \mathbf{x}, \mathbf{e}\rangle + \langle \mathbf{y}, \mathbf{s}_{\mathsf{lat}}\rangle + \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \tfrac{q}{p}\left\lfloor \tfrac{p}{q}\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}\right\rceil, \mathbf{s}_{\mathsf{fft}}\right\rangle\right)\right). \end{aligned}$$

Even if it means reducing the threshold a little, we can still expect $F^{(\mathsf{ms})}_{\mathbf{s}_{\mathsf{enu}}}(\mathbf{s}_{\mathsf{fft}} \bmod p) \geqslant T$ since the additional term $\left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \frac{q}{p}\left\lfloor \frac{p}{q}\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}\right\rceil, \mathbf{s}_{\mathsf{fft}}\right\rangle$ is also biased toward zero. Indeed, it is the scalar product of two short vectors; in particular:

$$d_{\mathsf{ms}} \stackrel{\text{def}}{=} \mathbb{E}\left(\left|\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \tfrac{q}{p}\left\lfloor \tfrac{p}{q}\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}\right\rceil\right|\right) \approx \tfrac{q}{p}\sqrt{\tfrac{n_{\mathsf{fft}}}{12}} \tag{9.11}$$

since we can make the approximation that each of the coordinates of $\frac{p}{q}\mathbf{A}_{\mathsf{fft}}^{\intercal}\mathbf{x} - \left\lfloor \frac{p}{q}\mathbf{A}_{\mathsf{fft}}^{\intercal}\mathbf{x} \right\rceil$ are drawn uniformly at random in $[-0.5, 0.5]$.

Finally, [MAT22] claims that the security level of NIST candidates like KYBER could be significantly lowered. However, this result is not widely accepted, as the analysis relies on assumptions which turn out to be false according to [DP23b].

9.1.1.3 Dual attacks in code-based cryptography their analysis

Dual attacks in lattice based cryptography can be viewed as the lattice based analogue of statistical decoding in code-based cryptography which dates back to Al Jabri [Jab01]. In this case, short codewords $\mathbf{h}$ in the dual code are computed, where short is with respect to the Hamming distance. The goal is to solve the decoding problem $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ where $\mathbf{b}$ and $\mathbf{A}$ are given whereas $\mathbf{s}$ and $\mathbf{e}$ are unknown and $\mathbf{e}$ is of small Hamming weight (rather than of small Euclidean norm for the LWE problem). The issue is to recover $\mathbf{s}$. Here too, the inner product $\langle \mathbf{h}, \mathbf{b} \rangle$ is biased towards zero. This is used to solve the decoding problem very much in the same way as it is used to solve the LWE problem. Similarly to what happened in lattice based cryptography, dual attacks became much more effective through a splitting strategy which allowed to look for short codewords in a smaller code [CDMT22]. In an analogous way to what was done in lattice based cryptography, these attacks were analyzed by making assumptions and in particular independence assumptions [CDMT22, Ass. 3.7] which are close to the independence assumptions made for analyzing MATZOV's attack [MAT22, Ass. 4.4, Ass. 5.8]. In the lattice based case, these assumptions were shown to contradict some theorems in certain regimes or well-tested heuristics in some other regimes [DP23b].

Note that it was already noticed in [CDMT22, §3.4] that the i.i.d. Bernoulli model used for analyzing dual attacks in code-based cryptography is not always accurate. However, it was conjectured there that the difference between this ideal model and experiments has no impact on the asymptotic analysis of the decoding based on this model. This was proved to be wrong in [MT23]. However, this paper gave at the same time an approach for analyzing rigorously dual attacks in coding theory by bringing in a duality equation [MT23, Prop. 1.3] shedding some light on the fundamental quantities manipulated by the decoder. This allowed to obtain a proof of the correctness of a slightly modified version of the dual attack proposed in [CDMT22]. Dual attacks for solving the decoding problem have been further improved in [CDMT24] and the approach of [MT23] has been carried over to this improved dual attack. [CDMT24] can be viewed as a somewhat improved version of the MATZOV attack in the context of codes, where the modulus switching part is replaced by an optimal lossy source encoder. It is worthwhile to note that [CDMT24, Section 8] shows that the fundamental duality equation [MT23, Prop. 1.3] used to analyze dual attacks for codes also carries over to the lattice setting and could serve as a tool to analyze dual attacks for lattices. Moreover, at the same time, a series of papers [DP23a, PS24] provide some new ideas to properly analyze dual attacks in lattices. In particular, concurrently to [CDMT24, Section 8], [DP23a] provided, in a more in-depth work and using comparable but not identical reasoning, similar heuristics to predict the behavior of these attacks.

9.1.2 Our contribution

Our purpose here is to come up with a variation of the MATZOV algorithm, that improves it and which also helps analyzing it. This is obtained by a lattice based analogue of [CDMT24],

replacing the modulus switching by lossy source encoding/using the relevant quantizer based on polar coding. We use the duality approach [MT23, CDMT24, DP23a] to analyze our new dual attack to avoid relying on independence assumptions. This duality approach allows us to derive a new simple heuristic that is backed up by experimental evidence and which is essentially a generalization of the heuristics made in [CDMT24, DP23a]. All in all, it turns out that the complexities claimed in [MAT22] can be achieved and even slightly surpassed with our algorithm, which dents the parameters of KYBER by a few bits, as predicted in [MAT22].

9.1.2.1 A lossy source coding approach

Similarly to [CDMT24] we observe that the FFT approach factorizing the common computations for computing all the $F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{z})$ for $\mathbf{z} \in \mathbb{Z}_q^{n_{\mathsf{fft}}}$ is probably suboptimal, since the $\mathbf{z}$ such that $F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{z})$ is maximum is likely to be attained for $\mathbf{z} = \mathbf{s}_{\mathsf{fft}}$ which is of rather small norm. The problem is that the fast FFT algorithm does not leverage the fact that we only need to compute it for small $\mathbf{z}$'s which have the same Euclidean norm as $\mathbf{s}_{\mathsf{fft}}$. In a sense, the modulus switching approach of [MAT22] is a way to alleviate this phenomenon since $\mathbf{s}_{\mathsf{fft}} \mod p$ is more uniformly distributed in $\mathbb{Z}_p^{n_{\mathsf{fft}}}$ than $\mathbf{s}$ is in $\mathbb{Z}_q^{n_{\mathsf{fft}}}$. A further refinement of this method, in the spirit of [GJS15, Section 5.5], involves approximating the relevant $\mathbf{z} \in \mathbb{Z}_q^{n_{\mathsf{fft}}}$'s by close enough codewords. This also allows to reduce the size $q^{n_{\mathsf{fft}}}$ of the space over which the fast Fourier transform is applied.

Basically the lossy source/quantizing approach can be explained as follows. We choose a linear code $\mathcal{C}_{\mathsf{lsc}}$ generated by the matrix $\mathbf{G} \in \mathbb{Z}_q^{n_{\mathsf{fft}} \times k_{\mathsf{fft}}}$, i.e. $\mathcal{C}_{\mathsf{lsc}} \stackrel{\text{def}}{=} \{\mathbf{G}\mathbf{u}_{\mathsf{lsc}} : \mathbf{u}_{\mathsf{lsc}} \in \mathbb{Z}_q^{k_{\mathsf{fft}}}\}$ so that we can find efficiently, for any $\mathbf{y}_{\mathsf{fft}} \in \mathbb{Z}_q^{n_{\mathsf{fft}}}$, a $\mathbf{u}_{\mathsf{lsc}} \in \mathbb{Z}_q^{k_{\mathsf{fft}}}$ such that $\mathbf{G}\mathbf{u}_{\mathsf{lsc}}$ is close to $\mathbf{y}_{\mathsf{fft}}$. $\mathbf{G}\mathbf{u}_{\mathsf{lsc}}$ can be viewed as a "quantization" of $\mathbf{y}_{\mathsf{fft}}$ and $\mathcal{C}_{\mathsf{lsc}}$ as a lossy source code or code used for quantization. We apply this quantization to all pairs of short dual vectors $(\mathbf{x}, \mathbf{y})$ in $\mathscr{S}$ and compute for all such $\mathbf{x}$'s a corresponding $\mathbf{u}_{\mathsf{lsc}} \in \mathbb{Z}_q^{k_{\mathsf{fft}}}$ such that

$$\left|\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathbf{G}\mathbf{u}_{\mathsf{lsc}}\right| \approx d_{\mathsf{lsc}} \tag{9.12}$$

where d_{lsc} is the decoding distance of the lossy source code. The point is that the left hand term in (9.4) can be rewritten as

$$\begin{aligned} \langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}} \rangle - \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}, \mathbf{s}_{\mathsf{fft}} \right\rangle &= \langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}} \rangle - \langle \mathbf{G}\mathbf{u}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}} \rangle - \langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}} \rangle & (9.13) \\ &= \langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}} \rangle - \langle \mathbf{u}_{\mathsf{lsc}}, \mathbf{G}^{\mathsf{T}}\mathbf{s}_{\mathsf{fft}} \rangle - \langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}} \rangle & (9.14) \end{aligned}$$

where $\mathbf{e}_{\mathsf{lsc}} \stackrel{\text{def}}{=} \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathbf{G}\mathbf{u}_{\mathsf{lsc}}$. If we use (9.4) we see that

$$\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}} \rangle - \langle \mathbf{u}_{\mathsf{lsc}}, \mathbf{G}^{\mathsf{T}}\mathbf{s}_{\mathsf{fft}} \rangle = \langle \mathbf{x}, \mathbf{e} \rangle + \langle \mathbf{y}, \mathbf{s}_{\mathsf{lat}} \rangle + \langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}} \rangle . \tag{9.15}$$

So we expect that the left-hand term $\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}} \rangle - \langle \mathbf{u}_{\mathsf{lsc}}, \mathbf{G}^{\mathsf{T}}\mathbf{s}_{\mathsf{fft}} \rangle$ is still small. In other words, we once again performed a reduction to an LWE problem. Indeed, Equation (9.15) can be interpreted as an LWE sample $(\mathbf{a}', b') \in \mathbb{Z}_q^{k_{\mathsf{fft}}} \times \mathbb{Z}_q$, with secret $\mathbf{s}' \in \mathbb{Z}_q^{k_{\mathsf{fft}}}$ and error term $e' \in \mathbb{Z}_q$, where

$$\begin{aligned} \mathbf{a}' &\stackrel{\text{def}}{=} \mathbf{u}_{\mathsf{lsc}}, & (9.16) \\ b' &\stackrel{\text{def}}{=} \langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}} \rangle , & (9.17) \\ \mathbf{s}' &\stackrel{\text{def}}{=} \mathbf{G}^{\mathsf{T}}\mathbf{s}_{\mathsf{fft}}, & (9.18) \\ e' &\stackrel{\text{def}}{=} \langle \mathbf{x}, \mathbf{e} \rangle + \langle \mathbf{y}, \mathbf{s}_{\mathsf{lat}} \rangle + \langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}} \rangle . & (9.19) \end{aligned}$$

The advantage of this reduction is that it drastically reduces the dimension of the problem. However, it is important to note that the secret $\mathbf{s}'$ is no longer small, but uniformly distributed over $\mathbb{Z}_q^{k_{\mathsf{fft}}}$; thus, this new problem becomes a plain-LWE problem. We can now solve it using a technique similar to the one previously described. This motivates to replace $F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}$ by

$$\begin{array}{rccc} F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}: & \mathbb{Z}_q^{k_{\mathsf{fft}}} & \longrightarrow & \mathbb{R} \\ & \mathbf{z} & \longmapsto & \displaystyle\sum_{(\mathbf{x},\mathbf{y})\in\mathscr{S}} \cos\left(\frac{2\pi}{q}\left(\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\rangle - \langle \mathbf{u}_{\mathsf{lsc}}, \mathbf{z}\rangle\right)\right). \end{array} \tag{9.20}$$

Again, for the wrong guess $\widetilde{\mathbf{s}_{\mathsf{enu}}} \neq \mathbf{s}_{\mathsf{enu}}$ and for all $\mathbf{z} \in \mathbb{Z}_p^{k_{\mathsf{fft}}}$, $F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{z})$ is still expected to be 0 whereas for the good guess $\mathbf{s}_{\mathsf{enu}}$ we have, from Equation (9.15), that

$$F^{(\mathsf{lsc})}_{\mathbf{s}_{\mathsf{enu}}}\left(\mathbf{G}^{\intercal}\mathbf{s}_{\mathsf{fft}}\right) = \sum_{(\mathbf{x},\mathbf{y})\in\mathscr{S}} \cos\left(\frac{2\pi}{q}\left(\langle \mathbf{x}, \mathbf{e}\rangle + \langle \mathbf{y}, \mathbf{s}_{\mathsf{lat}}\rangle + \langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}}\rangle\right)\right) \tag{9.21}$$

which is still expected to be large because the additional term $\langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}}\rangle$ is biased towards zero.

The pro and cons of this approach is that on the positive side:
– It allows us to choose large values for n_{fft}, since we are now not limited by the $q^{n_{\mathsf{fft}}}$ term in the complexity coming from evaluating $F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}$ but by a smaller $q^{k_{\mathsf{fft}}}$ term. This in turn allows us to decrease the n_{lat} term and therefore the cost of lattice reduction.
– It allows us to solve radically the problem that $\mathbf{s}_{\mathsf{fft}}$ is not uniformly distributed in $\mathbb{Z}_q^{n_{\mathsf{fft}}}$. If k_{fft} is low enough, it can namely be verified that the likely argmax $\mathbf{G}^{\intercal}\mathbf{s}_{\mathsf{fft}}$ of $F^{(\mathsf{lsc})}_{\mathbf{s}_{\mathsf{enu}}}(\mathbf{z})$ is uniformly distributed in $\mathbb{Z}_q^{k_{\mathsf{fft}}}$. This solves one source of suboptimality of this approach.
On the negative side, it increases the noise term in the right-hand side of (9.15) expressing the quantity $\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\mathbf{s}_{\mathsf{enu}}\rangle - \langle \mathbf{u}_{\mathsf{lsc}}, \mathbf{G}^{\intercal}\mathbf{s}_{\mathsf{fft}}\rangle$. This is due to the additional term $\langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}}\rangle$ which appears there. We therefore expect $F^{(\mathsf{lsc})}_{\mathbf{s}_{\mathsf{enu}}}\left(\mathbf{G}^{\intercal}\mathbf{s}_{\mathsf{fft}}\right)$ to be smaller than $F_{\mathbf{s}_{\mathsf{enu}}}\left(\mathbf{s}_{\mathsf{fft}}\right)$ and will need to make $\mathscr{S}$ bigger to distinguish it from other values of $F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{z})$.

The last point above makes it clear that we want to make the additional noise term $\langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}}\rangle$ as small as possible. But of course for a given k_{fft}, there is a lower bound of what we can achieve. It can namely be proven (see Equation (9.34) in Section 9.2) that the best we can do is to choose, for a given k_{fft}, the decoding distance d_{lsc} such that

$$d_{\mathsf{lsc}} \approx q^{1-\frac{k_{\mathsf{fft}}}{n_{\mathsf{fft}}}} \cdot \sqrt{\frac{n_{\mathsf{fft}}}{2\pi e}}. \tag{9.22}$$

This can be compared to the modulus switching approach. For a same FFT complexity, meaning roughly that k_{fft} is such that $p^{n_{\mathsf{fft}}} = q^{k_{\mathsf{fft}}}$, we have

$$d_{\mathsf{ms}} \approx \frac{q}{p}\sqrt{\frac{n_{\mathsf{fft}}}{12}} \approx 0.28867\frac{q\sqrt{n_{\mathsf{fft}}}}{p} \tag{9.23}$$

$$d_{\mathsf{lsc}} \approx \frac{q}{p}\sqrt{\frac{n_{\mathsf{fft}}}{2\pi e}} \approx 0.24197\frac{q\sqrt{n_{\mathsf{fft}}}}{p} \tag{9.24}$$

Equations (9.23) and (9.24) show that the lossy source coding approach yields a smaller value for $|\langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}}\rangle|$ than modulus switching, as $d_{\mathsf{lsc}} < d_{\mathsf{ms}}$. However, unlike modulus switching, we need to construct a code that can be decoded efficiently up to the optimal decoding

distance d_{lsc}. A classical approach involves using a Cartesian product of small random codes. This solution achieves asymptotically the optimal decoding distance d_{lsc} but with a sub-exponential complexity that is not so negligible (super-polynomial). In Section 9.3.3, we propose an alternative solution using polar codes, which can achieve a decoding distance very close to d_{lsc} in quasi-linear time.

It should be noted that the modulus switching strategy can really be viewed as a quantizing approach. In the modulus switching technique, we approximate $\mathbf{A}_{\mathsf{fft}}^{\intercal}\mathbf{x}$ by $\frac{q}{p}\left\lfloor \frac{p}{q}\mathbf{A}_{\mathsf{fft}}^{\intercal}\mathbf{x} \right\rceil$, i.e. we quantize/approximate a point in $\mathbb{R}^{n_{\mathsf{fft}}}$ by a point in the lattice $\frac{q}{p}\mathbb{Z}^{n_{\mathsf{fft}}}$. On the other hand, in the case of the lossy source code approach we approximate a point in $\mathbb{Z}_q^{n_{\mathsf{fft}}}$ by a codeword in $\mathcal{C}_{\mathsf{lsc}}$. If we express this as a quantizer, this means that we quantize/approximate a point in $\mathbb{R}^{n_{\mathsf{fft}}}$ by a lattice point in Construction A applied to $\mathcal{C}_{\mathsf{lsc}}$. The second quantizer just turns out to be much better than the first quantizer in terms of the distortion/quantizing distance which is achieved.

9.1.2.2 Getting rid of independence assumptions and results

In [MAT22], it is argued that dual attacks can substantially lower the security level of certain NIST candidates, such as Kyber. However, this claim remains contested, as their analysis relies on assumptions that, according to [DP23b], have been proven to be incorrect. Specifically, [DP23b] highlights a flawed independence assumption, which can be stated as follows:

Assumption 3 (Independence Assumption)**.** *Let Λ be a full-rank lattice of dimension n, and let $\mathbf{r}$ be a random variable distributed according to* $\mathrm{Unif}(\mathbb{R}^n/\Lambda)$ *or ${\mathcal{B}_\alpha}^n$, where the $\mathcal{B}_\alpha$ are i.i.d. centered binomial variables. Assume that $\left(e^{2i\pi\langle \mathbf{w},\mathbf{r}\rangle}\right)_{\mathbf{w}\in\Lambda}$ are mutually independent.*

In dual attacks, evaluation functions involve sums of terms like those described in the assumption. Assuming independence when the terms are not independent can lead to significant miscalculations in estimating false positives passing the evaluation function. This issue is highlighted in [DP23b], which shows that predictions of such scoring functions are inaccurate in certain regimes.

Recent papers [DP23a, MT23, CDMT24] present new approaches to analyze dual attacks accurately, without relying on the independence assumption. Specifically, let

$$F(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{\mathbf{w}\in\Lambda\cap\mathscr{B}} e^{2i\pi\langle \mathbf{w},\mathbf{r}(\mathbf{x})\rangle} \tag{9.25}$$

define a scoring function, where Λ is a full-rank lattice of dimension n, $\mathscr{B}\subseteq\mathbb{R}^n$ a set of small vectors, and $\mathbf{r}(\mathbf{x})\in\mathbb{R}^n$, all depending on the specific dual attack method under consideration. It can be observed that $F(\mathbf{x})$ remains invariant when any vector from the dual lattice of Λ is added to $\mathbf{r}(\mathbf{x})$. Indeed, for any $\mathbf{w}^\vee\in\Lambda^\vee$, we have

$$\left\langle \mathbf{w}, \mathbf{r}(\mathbf{x})+\mathbf{w}^\vee\right\rangle = \langle \mathbf{w},\mathbf{r}(\mathbf{x})\rangle + \left\langle \mathbf{w},\mathbf{w}^\vee\right\rangle = \langle \mathbf{w},\mathbf{r}(\mathbf{x})\rangle\,. \tag{9.26}$$

Thus, it is reasonable to conclude that $\langle \mathbf{w},\mathbf{r}(\mathbf{x})\rangle$ depends on the structure of the coset $\Lambda^\vee+\mathbf{r}(\mathbf{x})$, and in particular, on its shortest vector. One way to see this is by making the following approximation coming from the Poisson summation formula (see the discussion in Section 8.3.1)

$$F(\mathbf{x}) \approx \frac{1}{\mathcal{V}(\Lambda)}\cdot \sum_{\mathbf{w}^\vee\in\Lambda^\vee+\mathbf{r}(\mathbf{x})} \widehat{\mathbb{1}_{\mathscr{B}}}\left(\mathbf{w}^\vee\right). \tag{9.27}$$

In [DP23a, CDMT24], it was observed that the Fourier transform of the indicator function can be expressed in terms of Bessel functions, depending only on the input's length. Consequently, $F(\mathbf{x})$ is essentially related to the length enumerator of the vectors in $\Lambda^\vee + \mathbf{r}(\mathbf{x})$, particularly the shortest. We distinguish between correct and incorrect candidates by noting that $\mathbf{r}(\mathbf{x})$ is notably small when $\mathbf{x}$ is the desired vector, and random otherwise. Therefore, for the correct guess, the shortest vector in $\Lambda^\vee + \mathbf{r}(\mathbf{x})$ is $\mathbf{r}(\mathbf{x})$, while for incorrect guesses, it corresponds to the typical shortest vector in a random coset of a random lattice.

This approach allowed us to analyze our evaluation functions thoroughly (see Section 9.4). However, to simplify calculations, we make approximations that we validate through simulations. Finally, we control the number of false positives in our attack, ensuring they remain negligible. By doing so, we achieve results close to those of MATZOV (see Section 9.5). Furthermore, as they did previously, we apply our attack to the parameters of KYBER and confirm the claim in [MAT22], which asserts that KYBER's security does not meet the NIST's requirements.

9.2 Notation and preliminaries

9.2.1 Basic notation

We denote vectors by bold lowercase letters and matrices by bold uppercase letters, e.g. $\mathbf{v}$ and $\mathbf{M}$. We consider the vectors as column vectors and therefore row vectors are denoted $\mathbf{v}^\intercal$. The concatenation of vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted as $(\mathbf{x}, \mathbf{y})$. The components of the vector $\mathbf{x} \in \mathbb{R}^n$ are denoted by x_i for $i \in [\![1, n]\!]$ where $[\![a, b]\!]$ are the integers between a and b.

For any $x \in \mathbb{Z}_q$, denote by $\widehat{x} \in x + q\mathbb{Z}$ the unique integer such that $|\widehat{x}| \leqslant \frac{q-1}{2}$. We extend this notion to vectors $\mathbf{x} \in \mathbb{Z}_q^n$ componentwise. In other words, $\widehat{\mathbf{x}}$ is the lift from $\mathbb{Z}_q$ to $\mathbb{Z}$ centered on $\mathbf{0}$. We define $|\mathbf{x}|$ for $\mathbf{x} \in \mathbb{Z}_q^n$ as

$$|\mathbf{x}| \stackrel{\text{def}}{=} |\widehat{\mathbf{x}}| \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^{n} \widehat{x_i}^2} \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^{n} \operatorname*{argmin}_{j \in \mathbb{Z}} \left((x_i + jq)^2\right)} \tag{9.28}$$

9.2.2 The Learning With Error problem

Let us define more formally the LWE problem here. It starts by defining an LWE oracle that produces samples according to the following distribution:

Definition 53 (LWE oracle)**.** *Let $q, n, m \in \mathbb{N}$, and let χ_s, χ_e be distributions over $\mathbb{Z}_q$. We first draw $\mathbf{s} \in \mathbb{Z}_q^n$ with coordinates drawn independently from each other according to the distribution χ_s and then draw m* LWE *samples $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where the $\mathbf{a}_i$'s are drawn uniformly at random in $\mathbb{Z}_q^n$ and $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i$ where the e_i's are drawn independently according to the distribution χ_e. We let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be the matrix where the i-th row is $\mathbf{a}_i$. The pair $(\mathbf{A}, \mathbf{b})$ is the output of the oracle and satisfies $\mathbf{b} \stackrel{def}{=} \mathbf{A}\mathbf{s} + \mathbf{e}$.*

We then define the Search-LWE problem as follows:

Problem 1 (Search-LWE)**.** *Given a sample $(\mathbf{A}, \mathbf{b})$ drawn from an $LWE(q, n, m, \chi_s, \chi_e)$-oracle, the goal is to recover the secret vector $\mathbf{s}$.*

In the literature, the LWE oracle is sometimes defined by replacing the matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ by a vector $\mathbf{a} \in \mathbb{Z}_q^n$. Then the LWE problems are stated for an arbitrary number of calls to the oracle. If the LWE oracle is called m times, then the situation is actually the same as above.

9.2.3 The centered binomial distribution

In the LWE oracle, the distributions χ_s and χ_e depend on the context. Historically, the secret vector $\mathbf{s}$ was distributed uniformly in $\mathbb{Z}_q^n$ and the noise vector $\mathbf{e}$ was short. It is quite common today to consider the case where $\mathbf{s}$ is also short; we are then talking about the Small-LWE problem. In recent cryptosystems, particularly those involved in the NIST Post-Quantum Standardization Process, the distributions for χ_s and χ_e are centered binomial distributions. Note that in KYBER, the distribution for the secret vector and the error vector is the same.

Definition 54 (Centered Binomial Distribution)**.** *The centered binomial distribution $\mathcal{B}_\alpha$ of parameter $\alpha \in \left[\!\left[0, \frac{q-1}{2}\right]\!\right]$ is defined as $\mathcal{B}_\alpha \sim \sum_{i=1}^{\alpha}(X_i - Y_i)$ where the X_i's and Y_i's are i.i.d. as uniform over $\{0,1\}$. In particular, for all $i \in [\![-\alpha, \alpha]\!]$, we have $\mathbb{P}\left(\mathcal{B}_\alpha = i\right) = 2^{-2\alpha}\binom{2\alpha}{\alpha+i}$. Note that $\mathcal{B}_\alpha$ has mean 0 and standard deviation $\sigma \overset{def}{=} \sqrt{\frac{\alpha}{2}}$.*

9.2.4 Further lattice background

Recall that we can construct a lattice from a linear code through Construction A:

Definition 55 (Construction A)**.** *Let $\mathcal{C}$ be an $[n,k]_q$-code. The lattice Λ obtained by Construction A applied to $\mathcal{C}$ is given by*

$$\Lambda(\mathcal{C}) \overset{def}{=} \{\mathbf{x} \in \mathbb{R}^n \ : \ \mathbf{x} \equiv \mathbf{c} \bmod q, \ \mathbf{c} \in \mathcal{C}\}. \tag{9.29}$$

Note that if $\mathcal{C}$ is defined by a systematic generator matrix $\mathbf{G} \overset{\text{def}}{=} \begin{bmatrix}\mathbf{I}_k \\ \mathbf{A}\end{bmatrix} \in \mathbb{Z}_q^{n \times k}$, then the lattice $\Lambda(\mathcal{C})$ that is obtained through Construction A is also the lattice $\Lambda(\mathbf{B})$ generated by

$$\mathbf{B} \overset{\text{def}}{=} \left[\begin{array}{c|c} \mathbf{I}_k & \mathbf{0} \\ \hline \mathbf{A} & q\mathbf{I}_{n-k} \end{array}\right]. \tag{9.30}$$

Clearly finding the closest point in Euclidean distance to some $\mathbf{y} \in \mathbb{Z}_q^n$ in $\mathcal{C}$ also amounts to find the closest lattice point in $\Lambda(\mathcal{C})$ of $\mathbf{y}$. The algorithm for performing this task when $\mathbf{y}$ belongs to $\mathbb{R}^n$ is known as a mean-squared-error (MSE) quantizer for $\Lambda(\mathcal{C})$. To analyze its performance, first notice that the fundamental Voronoï region V of a lattice $\Lambda(\mathcal{C})$ associated to code $\mathcal{C}$ of dimension k over $\mathbb{Z}_q^n$ has volume $\mathcal{V}(V) \overset{\text{def}}{=} \mathcal{V}(\Lambda(\mathcal{C})) = q^{n-k}$ [CS88]. The average decoding distance ω provided by the mean-square quantizer for Λ can be assessed by the normalized second moment $G \overset{\text{def}}{=} G(\Lambda(\mathcal{C}))$, which is defined as

$$G \overset{\text{def}}{=} \frac{1}{n \cdot \mathcal{V}(V)^{\frac{2}{n}}} \int_V \frac{|\mathbf{v}|^2}{\mathcal{V}(V)} d\mathbf{v}. \tag{9.31}$$

It is known that

$$G \geqslant \frac{1}{2\pi e} \tag{9.32}$$

and is achieved asymptotically for lattices generated by Construction A from q-ary random codes when q gets big [ZF96]. The case where the equality is achieved in Equation (9.32) really corresponds to the case when the equality

$$|\mathcal{C}| \cdot \mathcal{V}\left(\mathsf{Ball}_{\omega}^{n}\right) = q^{n} \tag{9.33}$$

is met, i.e. when the average decoding distance is

$$\omega \stackrel{\text{def}}{=} \sqrt{\int_{V} \frac{|\mathbf{v}|^{2}}{\mathcal{V}(V)} d\mathbf{v}} = \sqrt{\frac{n}{2\pi e}} \cdot q^{1-\frac{k}{n}} + o()\,1. \tag{9.34}$$

It corresponds therefore to the lattice analogue of the Gilbert-Varshamov distance.

9.2.5 Short vector sampler

Dual attacks heavily depend on lattice reduction algorithms, such as BKZ, to find short vectors in a lattice or its dual. Our improvements in this chapter do not address these algorithms. Instead, we will use Algorithm 30 to obtain short vectors and refer the reader to [GJ21, MAT22] for implementation details.

Algorithm 30 Short Vectors Sampling Procedure [GJ21]

Input: A basis $\mathbf{B} = \begin{bmatrix}\mathbf{b}_0 & \dots & \mathbf{b}_{d-1}\end{bmatrix}$ for a lattice and $2 \leqslant \beta_{\mathsf{bkz}}, \beta_{\mathsf{sieve}} \in \mathbb{Z} \leqslant d$.

Output: A list of $N_{\mathsf{sieve}}(\beta_{\mathsf{sieve}}) \stackrel{\text{def}}{=} \left(\sqrt{\frac{4}{3}}\right)^{\beta_{\mathsf{sieve}}}$ vectors from the lattice.

1: Randomize the basis $\mathbf{B}$.
2: Run BKZ-β_{bkz} to obtain a reduced basis $\mathbf{b}'_0, \dots, \mathbf{b}'_{d-1}$.
3: Run a sieve in dimension β_{sieve} on the sublattice spanned by $\mathbf{b}'_0, \dots, \mathbf{b}'_{\beta_{\mathsf{sieve}}-1}$ to obtain a list L of $N_{\mathsf{sieve}}(\beta_{\mathsf{sieve}})$ vectors.
4: **return** L

The parameter β_{bkz} controls the block size in the BKZ algorithm, with an exponential cost in β_{bkz}. The sieving algorithm outputs $N_{\mathsf{sieve}}(\beta_{\mathsf{sieve}})$ short vectors in the lattice and its complexity also scales exponentially with β_{sieve}. The magnitude $N_{\mathsf{sieve}}(\beta_{\mathsf{sieve}})$ also grows exponentially with β_{sieve} but slower than the cost of sieving. We will write $T_{\mathsf{BKZ}}(d, \beta_{\mathsf{bkz}})$ for the cost of running BKZ-β_{bkz} in dimension d and $T_{\mathsf{sieve}}(\beta_{\mathsf{sieve}})$ for the cost of sieving in dimension β_{sieve}. One possible instantiation of the lattice sieve algorithm is [BDGL16] which has a cost of $2^{0.292\,\beta_{\mathsf{sieve}}+o(\beta_{\mathsf{sieve}})}$. Thus, according to the best known algorithms we have $T_{\mathsf{BKZ}}(d, \beta_{\mathsf{bkz}}) \in \mathrm{poly}\,(d) \cdot 2^{\Theta(\beta_{\mathsf{bkz}})}$ and $T_{\mathsf{sieve}}(\beta_{\mathsf{sieve}}) \in 2^{\Theta(\beta_{\mathsf{sieve}})}$. More specifically, we take these complexities from [MAT22, Lemma 4.1, Assumption 7.3].

Lemma 44 (Short Vectors Sampling Complexity)**.** *Let $\mathbf{B}$ be a basis of a d-dimensional lattice. Then, the running time T_{sample} of Algorithm 30 to output $N_{\mathsf{sieve}}(\beta_{\mathsf{sieve}})$ short vectors is*

$$T_{\mathsf{sample}}\left(d, \beta_{\mathsf{bkz}}, \beta_{\mathsf{sieve}}\right) = T_{\mathsf{BKZ}}(d, \beta_{\mathsf{bkz}}) + T_{\mathsf{sieve}}(\beta_{\mathsf{sieve}}) \tag{9.35}$$

where

- $T_{\mathsf{BKZ}}(d, \beta_{\mathsf{bkz}}) = C_{\mathsf{prog}}^{2} \cdot (d - \beta_{\mathsf{bkz}} + 1) \cdot T_{\mathsf{NNS}}(\beta_{\mathsf{bkz}}^{\mathsf{eff}})$,
- $T_{\mathsf{sieve}}(\beta_{\mathsf{sieve}}) = C_{\mathsf{prog}} \cdot T_{\mathsf{NNS}}\left(\beta_{\mathsf{sieve}}\right)$,

- ▷ $N_{\mathsf{sieve}}(\beta) = \left(\sqrt{\frac{4}{3}}\right)^{\beta}$ *is the expected number of sieve results,*
- ▷ $T_{\mathsf{NNS}}(\beta)$ *is the time complexity for finding all close pairs in dimension* β *(see [AGPS20a] with improvement of MATZOV [MAT22, Section 6]),*
- ▷ $C_{\mathsf{prog}} = 1/\left(1 - 2^{-0.292}\right)$ *is the number of close pairs search to run,*
- ▷ *and* β^{eff} *is the optimal sieve dimension to use for solving the Shortest Vector Problem (SVP) for lattices in dimension* β*.*

Note that in [Duc18], it is estimated that $\beta^{\mathsf{eff}} = \beta - \frac{\beta \log(4/3)}{\log(\beta/(2\pi e))}$.

Lengths of the short vectors produced Assuming that the Gaussian Heuristic (GH) and the Geometric Series Assumption[3] (GSA) [Sch03] hold for a d-dimensional lattice, applying BKZ-β to it produces vectors $\mathbf{x}$ of average length [Che13]:

$$|\mathbf{x}| \approx \delta(\beta)^d \cdot \mathcal{V}(\Lambda)^{\frac{1}{d}}, \tag{9.36}$$

where $\delta(\beta) = \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}}$ is the root-Hermite factor[4]. With these assumptions, the expected length of the returned short vectors is given by:

Lemma 45 (Length of the sampled short vectors [MAT22, Lemma 4.2])**.** *Let* Λ *be a* d*-dimensional lattice. Then, Algorithm 30 outputs at least* N *vectors of expected length* ℓ *given by*

$$\ell \overset{def}{=} \mathcal{V}(\Lambda)^{1/d} \cdot \sqrt{\frac{4}{3}} \cdot \delta(\beta_{\mathsf{sieve}})^{\beta_{\mathsf{sieve}}-1} \cdot \delta(\beta_{\mathsf{bkz}})^{d-\beta_{\mathsf{sieve}}}. \tag{9.37}$$

In our attack, we select β_{bkz} and β_{sieve} such that $T_{\mathsf{BKZ}}(d, \beta_{\mathsf{bkz}}) = T_{\mathsf{sieve}}(\beta_{\mathsf{sieve}})$. Under the same GH and GSA assumptions, we derive the following lemma:

Lemma 46. *The short vectors produced by Algorithm 30 are in a sublattice* Λ' *of dimension* β_{sieve} *and expected volume*

$$\mathcal{V}(\Lambda') = \left(\mathcal{V}(\Lambda)^{\frac{1}{d}} \cdot \delta(\beta_{\mathsf{bkz}})^{d-\beta_{\mathsf{sieve}}}\right)^{\beta_{\mathsf{sieve}}}. \tag{9.38}$$

9.2.6 Fast Fourier Transform

Dual lattice attacks widely use discrete Fourier transforms:

Definition 56 (Discrete Fourier Transform)**.** *The discrete Fourier transform* $\widehat{f}$ *of a function* $f\colon \mathbb{Z}_q^n \longrightarrow \mathbb{C}$ *is defined as*

$$\widehat{f}(\mathbf{x}) = \sum_{\mathbf{a}\in\mathbb{Z}_q^n} f(\mathbf{a}) e^{-\frac{2i\pi}{q}\langle \mathbf{x},\mathbf{a}\rangle}. \tag{9.39}$$

[3] The GSA may lead us to underestimate the final complexity of a few bits (see [DP23b, Appendix A.3]).

[4] Experiments in [AD221] show that these assumptions hold for $d > \beta$ and $\beta \to \infty$ and in particular, it hold with good accuracy for $\beta > 50$.

The effectiveness of our dual attacks is heavily dependent on the speed at which we can compute discrete Fourier transforms. In [MAT22, Ass. 7.4], it is estimated that a Fast Fourier Transform (FFT) over $\mathbb{Z}_q^n$ requires $n \cdot q^{n+1}$ multiplications. Note that this is clearly suboptimal for large prime q's: while this is not a problem in [MAT22] as q is reduced thanks to modulus switching, here, we drop modulus switching resulting in q being big and imposed by the original LWE instance (i.e. $q = 3329$ for KYBER). This motivates us to give the following finer estimation for the cost of an FFT over $\mathbb{Z}_{3329}^n$.

Proposition 59 (Complexity of the FFT)**.** *Let $q = 3329$. There exists an FFT over $\mathbb{Z}_q$ with complexity given by:*

$$N_{FFT}^{(\mathsf{add})} = 240500 \quad \textit{and} \quad N_{FFT}^{(\mathsf{mul})} = 115928, \tag{9.40}$$

the number of additions and multiplications, respectively.

By using the algorithm in [DM90, §2.3.2] that reduces the calculation of an FFT over $\mathbb{Z}_q^n$ to FFT's over $\mathbb{Z}_q$, we deduce that the total cost to perform a discrete Fourier transform over $\mathbb{Z}_q^n$ is

$$n\, q^{n-1}\, N_{FFT}^{(\mathsf{add})} \quad \textit{and} \quad n\, q^{n-1}\, N_{FFT}^{(\mathsf{mul})} \tag{9.41}$$

additions and multiplications, respectively. Finally, by supposing as in [MAT22, Ass. 7.4] that the cost of an addition and a multiplication are

$$C_{\mathsf{add}} = 160 \quad \textit{and} \quad C_{\mathsf{mul}} = 1024, \tag{9.42}$$

respectively, the total cost of an FFT over $\mathbb{Z}_q^n$ is

$$C_{FFT} = C_{\mathsf{add}}\, n\, q^{n-1}\, N_{FFT}^{(\mathsf{add})} + C_{\mathsf{mul}}\, n\, q^{n-1}\, N_{FFT}^{(\mathsf{mul})}. \tag{9.43}$$

We obtained the number of additions and multiplications required for an FFT over $\mathbb{Z}_{3329}$, namely $N_{\mathsf{FFT}}^{(\mathsf{add})}$ and $N_{\mathsf{FFT}}^{(\mathsf{mul})}$, by slightly modifying the FFTW software. This software basically allows one to enumerate a large number of FFT's and we simply selected the one that minimized the cost C_{FFT}[5].

9.3 Our algorithm

In this section, we present our algorithm for solving the search LWE problem. Unlike MATZOV, we do not use modulus switching to reduce the modulus size. As noted in the introduction, we expect that a good lossy source code/quantizer results in a smaller additional noise term $\langle \mathbf{e}_{\mathsf{fft}}, \mathbf{s}_{\mathsf{fft}} \rangle$ compared to modulus switching. However, modulus switching may still offer benefits: it has certain advantages, such as improved FFT efficiency when the modulus is a power of two, which is not the case in the starting LWE problem for KYBER. Additionally, modulus switching provides more flexibility in choosing the parameter k_{fft}. Although a hybrid approach combining both techniques is possible, we do not pursue it in this work due to the complexity it would add to the analysis.

[5] We give more details on how this result was obtained in `https://github.com/kevin-carrier/CodedDualAttack/tree/main/claimFFT`

9.3.1 Overview

Our algorithm begins by partitioning the set of coordinates $I \stackrel{\text{def}}{=} [\![1, n]\!]$ of the secret $\mathbf{s}$ into I_{enu}, I_{fft} and I_{lat}, with sizes n_{enu}, n_{fft} and n_{lat}, respectively. The three parts of the vector $\mathbf{s}$ are denoted as $\mathbf{s}_{\mathsf{enu}}$, $\mathbf{s}_{\mathsf{fft}}$ and $\mathbf{s}_{\mathsf{lat}}$ respectively. Similarly, we divide the columns of $\mathbf{A}$ to obtain $\mathbf{A}_{\mathsf{enu}}$, $\mathbf{A}_{\mathsf{fft}}$ and $\mathbf{A}_{\mathsf{lat}}$. As described in the introduction, our algorithm basically tries to guess a value $\widetilde{\mathbf{s}_{\mathsf{enu}}}$ for $\mathbf{s}_{\mathsf{enu}}$ by computing an associated score for $\widetilde{\mathbf{s}_{\mathsf{enu}}}$ namely the maximum value of the score function, $\max_{\mathbf{z} \in \mathbb{Z}_q^{k_{\mathsf{fft}}}} F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{z})$ and making a decision on this value. Apart from the lossy source code approach described above, we depart from the MATZOV algorithm in the two following ways: firstly, we will make the bet that $\mathbf{s}_{\mathsf{enu}} = \mathbf{0}$, that is, we only consider $\widetilde{\mathbf{s}_{\mathsf{enu}}} = \mathbf{0}$ whereas MATZOV enumerates and tests several values $\widetilde{\mathbf{s}_{\mathsf{enu}}}$ taken by decreasing likelihood. Secondly, in our algorithm, we will select I_{lat} once and for all (instead of changing it at each iteration), and then iterate R times with different choices for I_{enu} and I_{fft} on the remaining positions, this allows us to reuse the computed dual vectors for each iteration. The skeleton of the whole algorithm is given in Algorithm 31.

Algorithm 31 The code based dual attack to solve LWE

Input: a sample $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$ produced by an $LWE\,(q, n, m, \mathcal{B}_\alpha, \mathcal{B}_\alpha)$ oracle.

Parameters: some positive integers R, T, β_{bkz}, β_{sieve}, n_{enu}, n_{fft}, k_{fft}, n_{lat}, d_{lsc} and an $[n_{\mathsf{fft}}, k_{\mathsf{fft}}]_q$ linear code with generator matrix $\mathbf{G}$.

Output: the secret vector $\mathbf{s}$.

choose $I_{\mathsf{lat}} \subseteq [\![1, n]\!]$ such that $|I_{\mathsf{lat}}| = n_{\mathsf{lat}}$;
$\mathscr{S} \leftarrow$ SET_OF_SHORT_LATTICE_VECTORS$(\mathbf{A}, I_{\mathsf{lat}})$;
while $i = 1 \cdots R$ **do**
  choose a partition $I_{\mathsf{enu}} \cup I_{\mathsf{fft}}$ of $[\![1, n]\!] \setminus I_{\mathsf{lat}}$ with $|I_{\mathsf{enu}}| = n_{\mathsf{enu}}$, $|I_{\mathsf{fft}}| = n_{\mathsf{fft}}$;
  $\mathbf{A}_{\mathsf{enu}}, \mathbf{A}_{\mathsf{fft}} \leftarrow$ select the columns of $\mathbf{A}$ indexed by I_{enu} and I_{fft} respectively;
  $\mathscr{L} \leftarrow$ LWE_SAMPLES$(\mathscr{S}, \mathbf{A}_{\mathsf{fft}}, \mathbf{G})$;
  V $\leftarrow$ SOLVE_LWE_WITH_FFT$(\mathscr{L})$;
  if V $\geqslant T$ **then**
    $\mathbf{s}_{\mathsf{enu}} \leftarrow \mathbf{0}$
    $(\mathbf{s}_{\mathsf{fft}}, \mathbf{s}_{\mathsf{lat}}) \leftarrow$ SUB_LWE_SOLVER$(\begin{bmatrix}\mathbf{A}_{\mathsf{fft}} & \mathbf{A}_{\mathsf{lat}}\end{bmatrix}, \mathbf{b})$
    return $(\mathbf{s}_{\mathsf{enu}}, \mathbf{s}_{\mathsf{fft}}, \mathbf{s}_{\mathsf{lat}}, I_{\mathsf{enu}}, I_{\mathsf{fft}}, I_{\mathsf{lat}})$

9.3.1.1 Finding short vectors

We now give details on the procedures SET_OF_SHORT_LATTICE_VECTORS, LWE_SAMPLES, SOLVE_LWE_WITH_FFT and SUB_LWE_SOLVER. This function outputs a set $\mathscr{S}$ of pairs of short vectors $(\mathbf{x}, \mathbf{y})$ obtained by performing the short vectors sampler Algorithm 30 with the appropriate choice of parameters on the lattice $\Lambda(\mathbf{B})$ where the lattice basis $\mathbf{B} \in \mathbb{R}^{(m+n_{\mathsf{lat}}) \times (m+n_{\mathsf{lat}})}$ is given by Construction A:

$$\mathbf{B} \stackrel{\text{def}}{=} \left[\begin{array}{c|c} \mathbf{I}_m & \mathbf{0} \\ \hline \mathbf{A}_{\mathsf{lat}}^{\mathsf{T}} & q\mathbf{I}_{n_{\mathsf{lat}}} \end{array}\right]. \tag{9.44}$$

This gives pairs $(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^m \times \mathbb{Z}^{n_{\mathsf{lat}}}$ for which both $\mathbf{x}$ and $\mathbf{y}$ are short and satisfy $\mathbf{y} = \mathbf{A}_{\mathsf{lat}}^{\mathsf{T}}\mathbf{x}$ mod q. For the rest of this chapter we denote by N the expected size of $\mathscr{S}$. From Lemma 44, we have

Notation 9.

$$N \overset{def}{=} N_{\mathsf{sieve}}(\beta_{\mathsf{sieve}}) \overset{def}{=} \left(\sqrt{\frac{4}{3}}\right)^{\beta_{\mathsf{sieve}}}. \tag{9.45}$$

9.3.1.2 Reducing the dimension with a linear code

We give more details on the LWE_Samples procedure here. The idea behind the dimension reduction technique using a linear code that we use here originates from Coded-BKW [GJS15]. Each pair $(\mathbf{x}, \mathbf{y})$ of short vectors in $\mathscr{S}$ yields one LWE sample by decoding $\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}$ in the code $\mathcal{C}_{\mathsf{lsc}}$ generated by $\mathbf{G}$. This yields an output $\mathbf{u}_{\mathsf{lsc}} \in \mathbb{Z}_q^{k_{\mathsf{fft}}}$ such that $\mathbf{e}_{\mathsf{lsc}} \overset{\text{def}}{=} \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathbf{G}\mathbf{u}_{\mathsf{lsc}}$ is of small norm, say close to some d_{lsc}.The new LWE sample is given by the pair $(\mathbf{u}_{\mathsf{lsc}}, \langle \mathbf{x}, \mathbf{b}\rangle)$. It is readily seen that, when $\mathbf{s}_{\mathsf{enu}} = \mathbf{0}$ we have

$$\langle \mathbf{x}, \mathbf{b}\rangle = \langle \mathbf{u}_{\mathsf{lsc}}, \mathbf{G}^{\mathsf{T}}\mathbf{s}_{\mathsf{fft}}\rangle + \langle \mathbf{x}, \mathbf{e}\rangle + \langle \mathbf{y}, \mathbf{s}_{\mathsf{lat}}\rangle + \langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}}\rangle \tag{9.46}$$

which corresponds to an LWE sample with secret $\mathbf{G}^{\mathsf{T}}\mathbf{s}_{\mathsf{fft}}$ and noise $e' = \langle \mathbf{x}, \mathbf{e}\rangle + \langle \mathbf{y}, \mathbf{s}_{\mathsf{lat}}\rangle + \langle \mathbf{e}_{\mathsf{lsc}}, \mathbf{s}_{\mathsf{fft}}\rangle$. All these samples $(\mathbf{u}_{\mathsf{lsc}}, \langle \mathbf{x}, \mathbf{b}\rangle)$ are then put in a list $\mathscr{L}$ which is the output of LWE_Samples.

9.3.1.3 Solving the LWE problem with a fast Fourier transform

Here we give more details on the Solve_LWE_with_FFT procedure. This procedure outputs a real number V which indicates to us how noisy the aforementioned LWE samples are. This is done by searching exhaustively for the solution $\mathbf{G}^{\mathsf{T}}\mathbf{s}_{\mathsf{fft}}$ by computing the score function for all $\mathbf{z} \in \mathbb{Z}_q^{k_{\mathsf{fft}}}$:

$$F_{\mathbf{0}}^{(\mathsf{lsc})}(\mathbf{z}) \overset{\text{def}}{=} \sum_{(\mathbf{u}_{\mathsf{lsc}}, b) \in \mathscr{L}} \cos\left(\tfrac{2\pi}{q}\left(b - \langle \mathbf{u}_{\mathsf{lsc}}, \mathbf{z}\rangle\right)\right), \tag{9.47}$$

and returning its maximum value, $V = \max_{\mathbf{z} \in \mathbb{Z}_q^{k_{\mathsf{fft}}}} F_{\mathbf{0}}^{(\mathsf{lsc})}(\mathbf{z})$. If $\mathbf{s}_{\mathsf{enu}} = \mathbf{0}$ we expect that this maximum value is achieved for $\mathbf{z} = \mathbf{G}^{\mathsf{T}}\mathbf{s}_{\mathsf{fft}}$. The score function is efficiently computed with an FFT as follows. First we compute a function $f_{\mathbf{0}}^{(\mathsf{lsc})}$ defined for $\mathbf{a} \in \mathbb{Z}_q^{k_{\mathsf{fft}}}$ as

$$f_{\mathbf{0}}^{(\mathsf{lsc})}(\mathbf{a}) \overset{\text{def}}{=} \sum_{(\mathbf{a}, b) \,:\, (\mathbf{a}, b) \in \mathscr{L}} e^{\frac{2i\pi}{q} b}, \tag{9.48}$$

then we compute the FFT of $f_{\mathbf{0}}^{(\mathsf{lsc})}$ and take its real part. It is readily seen that

$$F_{\mathbf{0}}^{(\mathsf{lsc})} = \mathfrak{Re}\left(\widehat{f_{\mathbf{0}}^{(\mathsf{lsc})}}\right). \tag{9.49}$$

9.3.1.4 Recovering the rest of the secret

Here we give more details about the sub_LWE_solver procedure. At this point, we expect that, under the condition that our parameters are well-chosen, if $V \geqslant T$ (here T will be chosen around the expected value of $F_{\mathbf{0}}^{(\mathsf{lsc})}(\mathbf{G}^{\mathsf{T}}\mathbf{s}_{\mathsf{fft}})$ when $\mathbf{s}_{\mathsf{enu}} = \mathbf{0}$) then with overwhelming probability, $\mathbf{s}_{\mathsf{enu}} = \mathbf{0}$. In other words, we recovered n_{enu} positions of the secret from the original LWE problem $(\mathbf{A}, \mathbf{b})$ of dimension $m \times n$. Note that when $\mathbf{s}_{\mathsf{enu}} = \mathbf{0}$ we can write

$$\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} = \begin{bmatrix}\mathbf{A}_{\mathsf{fft}} & \mathbf{A}_{\mathsf{lat}}\end{bmatrix}\begin{pmatrix}\mathbf{s}_{\mathsf{fft}} \\ \mathbf{s}_{\mathsf{lat}}\end{pmatrix} + \mathbf{e}. \tag{9.50}$$

As such, if indeed $\mathbf{s}_{\mathsf{enu}} = \mathbf{0}$, we can recover $\mathbf{s}_{\mathsf{fft}}$ and $\mathbf{s}_{\mathsf{lat}}$ by solving the new LWE problem given by $\left(\begin{bmatrix}\mathbf{A}_{\mathsf{fft}} & \mathbf{A}_{\mathsf{lat}}\end{bmatrix}, \mathbf{b}\right)$, which has strictly smaller dimension $m \times (n - n_{\mathsf{enu}})$. We do not specify a particular algorithm for the SUB_LWE_SOLVER routine, since it will be called at most once, and solving an LWE instance with smaller dimension and with error distribution $\mathcal{B}_\alpha$ has complexity negligible compared to the other computations in Algorithm 31.

9.3.2 Correctness of the algorithm

The following lemma provides the conditions on the parameters for Algorithm 31 to succeed.

Lemma 47 (Correctness)**.** *Suppose* $(\mathbf{A}, \mathbf{b})$ *is sampled from an* $LWE\,(q, n, m, \mathcal{B}_\alpha, \mathcal{B}_\alpha)$ *oracle, and* SUB_LWE_SOLVER *returns* $\mathbf{s}_{\mathsf{fft}}$ *and* $\mathbf{s}_{\mathsf{lat}}$ *with probability* $1 - \mu$ *when the bet* $\mathbf{s}_{\mathsf{enu}} = \mathbf{0}$ *is valid and the secret meets the threshold, namely* $F_{\mathbf{0}}^{(\mathsf{lsc})}(\mathbf{G}^\intercal \mathbf{s}_{\mathsf{fft}}) \geqslant T$*. The probability that our algorithm succeeds in recovering the secret* $\mathbf{s}$*, is lower bounded by*

$$P_{\text{success}} \overset{def}{=} \eta \cdot P_{\mathsf{good}} \cdot (1 - \mu) - \varepsilon \tag{9.51}$$

where

$$\varepsilon \overset{def}{=} R \cdot q^{k_{\mathsf{fft}}} \cdot P_{\mathsf{wrong}}, \tag{9.52}$$

$$\eta \overset{def}{=} \left(1 - \sum_{t=0}^{n_{\mathsf{enu}}+n_{\mathsf{fft}}} \left(1 - \frac{\binom{t}{n_{\mathsf{enu}}}}{\binom{n_{\mathsf{enu}}+n_{\mathsf{fft}}}{n_{\mathsf{enu}}}}\right)^R \binom{n_{\mathsf{enu}} + n_{\mathsf{fft}}}{t} p_0^t (1 - p_0)^{n_{\mathsf{enu}}+n_{\mathsf{fft}}-t}\right), \tag{9.53}$$

$$P_{\mathsf{good}} \overset{def}{=} \mathbb{P}\left(F_{\mathbf{0}}^{(\mathsf{lsc})}\left(\mathbf{G}^\intercal \mathbf{s}_{\mathsf{fft}}\right) \geqslant T \middle| \mathbf{s}_{\mathsf{enu}} = \mathbf{0}\right), \tag{9.54}$$

$$P_{\mathsf{wrong}} \overset{def}{=} \mathbb{P}\left(F_{\mathbf{0}}^{(\mathsf{lsc})}\left(\mathbf{z}\right) \geqslant T \middle| \mathbf{s}_{\mathsf{enu}} \neq \mathbf{0}\right), \tag{9.55}$$

$$p_0 \overset{def}{=} \mathbb{P}\left(\mathcal{B}_\alpha = 0\right) = 2^{-2\alpha}\binom{2\alpha}{\alpha} \tag{9.56}$$

and where $\mathbf{z}$ *is taken uniformly at random in* $\mathbb{Z}_q^{k_{\mathsf{fft}}}$*. We will use Approximations 5 and 6 to estimate* P_{good} *and* P_{wrong}*, respectively.*

Proof. For $i \in [\![1, R]\!]$, let us denote the following events for each iteration i:

- A_i: "$s_{\mathsf{enu}} = \mathbf{0}$";
- B_i: "$F_{\mathbf{0}}^{(\mathsf{lsc})}(\mathbf{G}^\intercal \mathbf{s}_{\mathsf{fft}}) \geqslant T$";
- C_i: "sub-LWE finds the secret";
- D_i: "$\exists \mathbf{z} \in \mathbb{Z}_q^{k_{\mathsf{fft}}},\ F_{\mathbf{0}}^{(\mathsf{lsc})}(\mathbf{z}) \geqslant T$";

Using the union bound on the probability that the algorithm fails and taking the complement of this event yields that the probability that our algorithm succeeds is lower bounded by

$$\mathbb{P}\left(\left(\bigcup_i (A_i \cap B_i \cap C_i)\right) \cap \left(\bigcap_i (A_i \cup \overline{D_i})\right)\right) \geqslant \mathbb{P}\left(\bigcup_i (A_i \cap B_i \cap C_i)\right) - \mathbb{P}\left(\bigcup_i (\overline{A_i} \cap D_i)\right)$$

Next, applying the union bound again and considering that we run R iterations, we can upper bound $\mathbb{P}\left(\bigcup_i \left(\overline{A_i} \cap D_i\right)\right)$ by $Rq^{k_{\mathsf{fft}}}P_{\mathsf{wrong}}$, which gives the $-\varepsilon$ term in the inequality stated in the lemma. On the other hand, it is straightforward to lower bound the first term:

$$\begin{aligned}
\mathbb{P}\left(\bigcup_i (A_i \cap B_i \cap C_i)\right) &= \mathbb{P}\left(\bigcup_i (A_i \cap B_i \cap C_i) \mid \bigcup_i A_i\right) \cdot \mathbb{P}\left(\bigcup_i A_i\right) \\
&\geqslant \mathbb{P}\left(B_{i_0} \cap C_{i_0} \mid A_{i_0}\right) \cdot \mathbb{P}\left(\bigcup_i A_i\right) \\
&= \mathbb{P}\left(C_{i_0} \mid A_{i_0}, B_{i_0}\right) \cdot \mathbb{P}\left(B_{i_0} \mid A_{i_0}\right) \cdot \mathbb{P}\left(\bigcup_i A_i\right) \\
&= (1-\mu) \cdot P_{\mathsf{good}} \cdot \mathbb{P}\left(\bigcup_i A_i\right)
\end{aligned}$$

Let n_0 denote the number of zeros in $\mathbf{s}_{[\![1,n]\!]\setminus I_{\mathsf{lat}}}$. Then, we have

$$\begin{aligned}
\mathbb{P}\left(\bigcup_i A_i\right) &= 1 - \mathbb{P}\left(\bigcap_i \overline{A_i}\right) \\
&= 1 - \sum_{t=0}^{n_{\mathsf{enu}}+n_{\mathsf{fft}}} \mathbb{P}\left(\bigcap_i \overline{A_i} \mid n_0 = t\right)^R \cdot \mathbb{P}\left(n_0 = t\right) \\
&= 1 - \sum_{t=0}^{n_{\mathsf{enu}}+n_{\mathsf{fft}}} \left(1 - \frac{\binom{t}{n_{\mathsf{enu}}}}{\binom{n_{\mathsf{enu}}+n_{\mathsf{fft}}}{n_{\mathsf{enu}}}}\right)^R \cdot \mathbb{P}\left(n_0 = t\right)
\end{aligned}$$

Clearly, n_0 follows a binomial distribution with parameter $n_{\mathsf{enu}} + n_{\mathsf{fft}}$ and p_0. This concludes the proof. □ □

Note that Lemma 47 does not impose any constraints on N and T. However, we must choose them carefully. First, if $R \cdot q^{k_{\mathsf{fft}}} \cdot P_{\mathsf{wrong}} \geqslant 1$ or if P_{good} is too small, then the lower bound we obtain for the probability of success of our dual attack is 0, which means we cannot actually guarantee a success. This is why, in Section 9.5, we select N and T that ensure a high probability of success. In particular, we set $\eta \geqslant 0.62$ and ε close to 0 (see Appendix 9.4), ensuring a success probability of at least $0.3(1-\mu)$. Furthermore, it is worth noting that the condition "$R \cdot q^{k_{\mathsf{fft}}} \cdot P_{\mathsf{wrong}}$ is much smaller than 1" resolves the indistinguishability issue raised in [DP23b].

9.3.3 Choice for the auxiliary code used

In Algorithm 31, one could ask: which code should we use for $\mathcal{C}_{\mathsf{lsc}}$? In terms of the decoding distance alone, the answer would be, just use a random code of dimension k_{fft} in $\mathbb{Z}_q^{n_{\mathsf{fft}}}$. In this case, we would obtain the decoding distance $d \approx q^{1-\frac{k_{\mathsf{fft}}}{n_{\mathsf{fft}}}}\sqrt{\frac{n_{\mathsf{fft}}}{2\pi e}}$ (see Equation (9.34)) attaining the bound (9.32) or (9.33). However, the decoding algorithm we could use in this case would be too complex for our purpose. We could instead use a product code structure as in [BDGL16]. Contrarily to what happens in the latter case, where spherical codes can be used, we are in a situation where more structured codes could do the job better. A natural answer is given here by polar codes.

The generator matrix of such a code in length n which is a power of 2 and an arbitrary code dimension k is obtained as follows. We define a generator matrix $\mathbf{G}$ for our code as $\mathbf{G} \stackrel{\text{def}}{=} \mathbf{K}_1 \otimes \cdots \otimes \mathbf{K}_{\log_2(n)} \cdot \mathbf{F}$ where $\otimes$ stands for the Kronecker product, $\mathbf{K}_i \stackrel{\text{def}}{=} \begin{bmatrix} 1 & 1 \\ \alpha_i & 0 \end{bmatrix}$ and α_i's are some invertible elements chosen uniformly at random in $\mathbb{Z}_q^*$. The matrix $\mathbf{F} \in \mathbb{Z}_q^{n \times k}$ is an expansion matrix such that for all $\mathbf{m} \in \mathbb{Z}_q^k$, $\mathbf{Fm}$ is exactly $\mathbf{m}$ on k positions and 0 on the others (that are the frozen positions). Since the code length we need is not necessarily a power of 2, we adjust it by puncturing the code (i.e. we remove as many rows of the generator matrix as needed).

Furthermore, the Successive Cancellation (SC) decoder which is classically used to decode polar codes in the error correction scenario can be turned with a simple modification into an algorithm for finding a close codeword (but not necessarily the closest one) [KU10]. This is precisely what is needed in our context. It needs a noise model to instantiate it, and this can be done for our Euclidean metric by using the Gaussian noise model. To get even closer to the optimal decoding distance, we have improved the decoding process: firstly we turn the SC algorithm into a probabilistic decoder, then we call it several times to get a list of codeword candidates and choose the closest one from this list (see Appendix 9.6.1 to get more details about our list decoder). This procedure induces a new small constant factor L in the whole complexity which is the size of the decoding list; but in return this additional cost allows us to achieve a better decoding distance. The complexity for decoding is given by:

Lemma 48 (Decoding polar codes)**.** *List decoding an $[n_{\mathsf{fft}}, k_{\mathsf{fft}}]_q$ polar code by using L probabilistic SC decoders can be done with time complexity of order*

$$T_{\mathsf{dec}}(q, n_{\mathsf{fft}}, k_{\mathsf{fft}}) = 3 \cdot L \cdot \left(C_{\text{mul}} \cdot N_{FFT}^{(\text{mul})}(q) + C_{\text{add}} \cdot N_{FFT}^{(\text{add})}(q) \right) \cdot n'_{\mathsf{fft}} \log_2(n'_{\mathsf{fft}}) \tag{9.57}$$

where n'_{fft} is the smallest power of 2 greater than n_{fft} and $N_{FFT}^{(\text{mul})}(q)$ is the number of multiplications we need to achieve a discrete Fourier transform over $\mathbb{Z}_q$.

The proof of this lemma directly follows from Lemma 49.

By using similar arguments as in [KU10], it can be proven that for n_{fft} tending to infinity and constant $\frac{k_{\mathsf{fft}}}{n_{\mathsf{fft}}}$, the average distance achieved by our decoder is

$$d_{\mathsf{lsc}} = \sqrt{\tfrac{n_{\mathsf{fft}}}{2\pi e}} \cdot q^{1 - \frac{k_{\mathsf{fft}}}{n_{\mathsf{fft}}}} \cdot (1 + o()\,1) \tag{9.58}$$

This result is essentially due to the polarization phenomenon (see Appendix 9.6.1). However, Equation (9.58) is not precise enough to accurately estimate the full complexity of our dual attack due to the $o(1)$ term. For this reason, we provide a C implementation[6] and present experimental results demonstrating that polar codes are perfectly suited to our case. The experiments we conducted use the exact codes required for our dual attacks, and our optimization suggests choosing $L = 1$. To justify the use of polar codes in this context, we verified that the total complexity of our dual attack closely matches the ideal scenario, where decoding at the distance $d_{\mathsf{lsc}} \stackrel{\text{def}}{=} \sqrt{\frac{n_{\mathsf{fft}}}{2\pi e}} \cdot q^{1 - \frac{k_{\mathsf{fft}}}{n_{\mathsf{fft}}}}$ would incur the same cost as using polar codes.

[6] `https://github.com/kevin-carrier/CodedDualAttack/tree/main/PolarCodeOverZq`

9.4 Analysis

9.4.1 Complexity

A general formula for the complexity of our algorithm using polar codes for $\mathcal{C}_{\mathsf{lsc}}$ is given by the following theorem:

Theorem 10 (Complexity of Algorithm 31)**.** *Using the same notations as in the correctness Lemma 47 and by supposing that the cost of one call to* Sub_LWE_Solver *is negligible, the average time complexity of Algorithm 31 is upper bounded by*

$$T_{\mathsf{sample}} + R \cdot (N \cdot T_{\mathsf{dec}} + T_{\mathit{FFT}}) \tag{9.59}$$

where

- ▷ $T_{\mathsf{sample}} \overset{def}{=} T_{\mathsf{sample}}(m + n_{\mathsf{lat}}, \beta_{\mathsf{bkz}}, \beta_{\mathsf{sieve}})$ *is the cost to produce* $N = N_{\mathsf{sieve}}(\beta_{\mathsf{sieve}})$ *short vectors in* $\Lambda(\mathbf{B})$*. This cost is given by Lemma 44;*
- ▷ $T_{\mathsf{dec}} \overset{def}{=} T_{\mathsf{dec}}(q, n_{\mathsf{fft}}, k_{\mathsf{fft}})$ *is the cost for decoding a random vector in* $\mathbb{Z}_q^{n_{\mathsf{fft}}}$ *in* $\mathcal{C}_{\mathsf{lsc}}$ *generated by* $\mathbf{G}$*. This cost is given by Lemma 48;*
- ▷ T_{FFT} *is the cost of an* FFT *over* $\mathbb{Z}_q^{k_{\mathsf{fft}}}$ *and is given by Proposition 59;*

9.4.2 Modelling the distribution of the score function

In this subsection, we provide an accurate estimation of the probabilities P_{good} and P_{wrong} that appear in Theorem 10. First, let us recall the expression of the score function:

$$F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{G}^{\intercal}\widetilde{\mathbf{s}_{\mathsf{fft}}}) \overset{\text{def}}{=} \sum_{(\mathbf{x},\mathbf{y})\in\mathscr{S}} \cos\left(\tfrac{2\pi}{q}\left(\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\rangle - \langle \mathbf{u}_{\mathsf{lsc}}, \mathbf{G}^{\intercal}\widetilde{\mathbf{s}_{\mathsf{fft}}}\rangle\right)\right) \tag{9.60}$$

$$= \sum_{(\mathbf{x},\mathbf{y})\in\mathscr{S}} \cos\left(\tfrac{2\pi}{q}\left(\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\rangle - \langle \mathbf{c}_{\mathsf{lsc}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}\rangle\right)\right) \tag{9.61}$$

where $\mathscr{S}$ is a set of N short vectors drawn from $\{(\mathbf{x},\mathbf{y}) \in \Lambda(\mathbf{B}) \,:\, |(\mathbf{x},\mathbf{y})| \leqslant d_{\mathsf{lat}}\}$. Here, $\mathbf{c}_{\mathsf{lsc}} \overset{\text{def}}{=} \mathbf{G}\mathbf{u}_{\mathsf{lsc}}$ is a codeword in $\mathcal{C}_{\mathsf{lsc}} \subseteq \mathbb{Z}_q^{n_{\mathsf{fft}}}$, obtained by decoding $\mathbf{A}_{\mathsf{fft}}^{\intercal}\mathbf{x}$ using our polar code decoder. It is important to note that, according to Lemma 46, the short lattice vectors in $\mathscr{S}$ generated by Algorithm 30 are not uniformly distributed within $\Lambda(\mathbf{B}) \cap \mathsf{Ball}_{d_{\mathsf{lat}}}^{m+n_{\mathsf{lat}}}$; instead, they belong to a β_{sieve}-dimensional sublattice $\Lambda(\mathbf{B}') \subseteq \Lambda(\mathbf{B})$, where $\mathbf{B}' \in \mathbb{R}^{(m+n_{\mathsf{lat}})\times\beta_{\mathsf{sieve}}}$. More precisely, we make the following assumption regarding the distribution of vectors in $\mathscr{S}$:

Assumption 4. *We assume that the set* $\mathscr{S}$ *consists of* N *vectors uniformly sampled from*

$$\left\{\mathbf{w} \in \Lambda(\mathbf{B}') \,:\, |\mathbf{w}| \leqslant d_{\mathsf{lat}}\right\},$$

where $\Lambda(\mathbf{B}')$ *is a* β_{sieve}*-dimensional sublattice of* $\Lambda(\mathbf{B})$ *with basis* $\mathbf{B}' \in \mathbb{R}^{(m+n_{\mathsf{lat}})\times\beta_{\mathsf{sieve}}}$*, and volume as stated in Lemma 46. The radius* $d_{\mathsf{lat}} \overset{def}{=} \frac{\ell(\beta_{\mathsf{sieve}}+1)}{\beta_{\mathsf{sieve}}}$ *corresponds to that of a* β_{sieve}*-dimensional Euclidean ball, within which the average vector length* ℓ *is given in Lemma 45.*

Additionally, we assume that the probability distribution of the output of our polar code decoder has radial symmetry. This means that the probability depends only on the distance between the returned codeword and the word to be decoded, not on the specific direction. Through experimentation, we observed that this distance closely follows a normal distribution. Based on these observations, we make the following assumption:

Assumption 5. *Let* $\mathbf{u}$ *be any vector from* $\mathbb{Z}_q^{n_{\mathsf{fft}}}$*, and let* $\mathrm{Dec}(\mathbf{u})$ *represent the random variable corresponding to the output of our polar code decoding algorithm. The distribution of* $|\mathbf{u} - \mathrm{Dec}(\mathbf{u})|$ *does not depend on* $\mathbf{u}$ *and we assume that it can be smoothed and approximated by the normal distribution* $\mathcal{N}(\mu_{\mathsf{lsc}}, \sigma_{\mathsf{lsc}}^2)$*, where* μ_{lsc} *is the mean and* σ_{lsc} *is the standard deviation, both determined through simulations. We also assume that the conditional distribution of* $\mathbf{u} - \mathrm{Dec}(\mathbf{u})$ *given that* $|\mathbf{u} - \mathrm{Dec}(\mathbf{u})| = d_{\mathsf{lsc}}$ *is uniform over* $(\Lambda(\mathcal{C}_{\mathsf{lsc}}) + \mathbf{u}) \cap \mathsf{Sphere}_{d_{\mathsf{lsc}}}^{n_{\mathsf{fft}}}$.

9.4.2.1 First-level approximation

The score function $F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}^{(\mathsf{lsc})}(\mathbf{G}^\intercal \widetilde{\mathbf{s}_{\mathsf{fft}}})$, as recalled in Equation (9.61), is a random variable influenced by multiple sources of randomness:

(i) the randomness in the short vector sampling Algorithm 30 that generates $\mathscr{S}$,

(ii) the inherent randomness in the polar code decoder Dec,

(iii) the randomness in the LWE instance, particularly in the choice of the matrix $\mathbf{A}$,

(iv) the randomness of the guess $\widetilde{\mathbf{s}_{\mathsf{enu}}}$, which primarily arises from the selection of I_{enu},

(v) the randomness of the guess $\widetilde{\mathbf{s}_{\mathsf{fft}}}$.

We denote by $\mathbb{E}_{\mathscr{S},\mathrm{Dec}}(\cdot)$ the conditional expectation over the randomness sources (i) and (ii). Thus, a first-level approximation of the score function is:

Approximation 2 (First-Level Approximation)**.** *The score function* $F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}^{(\mathsf{lsc})}(\mathbf{G}^\intercal \widetilde{\mathbf{s}_{\mathsf{fft}}})$ *can be approximated by its conditional expectation*

$$\mathbb{E}_{\mathscr{S},\mathrm{Dec}}\left(F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}^{(\mathsf{lsc})}(\mathbf{G}^\intercal \widetilde{\mathbf{s}_{\mathsf{fft}}})\right) \tag{9.62}$$

where the expectation is taken over the randomness of the polar code decoder and the short vector sampler.

We make the following approximation that we justify heuristically next.

Approximation 3 (Dual approximation)**.** *The conditional expectation in Approximation 2 can be expressed as*

$$\mathbb{E}_{\mathscr{S},\mathrm{Dec}}\left(F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}^{(\mathsf{lsc})}(\mathbf{G}^\intercal \widetilde{\mathbf{s}_{\mathsf{fft}}})\right) = \sum_{\substack{(\mathbf{w}_{\mathsf{lat}}^\vee, \mathbf{w}_{\mathsf{lsc}}^\vee) \\ \in \Lambda(\mathbf{B}_{\mathsf{global}}^{\mathsf{tmp}})^\vee + \frac{\left(\mathbf{B}'^\intercal \mathbf{r}_{\mathsf{lat}}, \mathbf{B}_{\mathsf{lsc}}^\intercal \mathbf{r}_{\mathsf{lsc}}\right)}{q}}} \widehat{f_{\mathsf{lat}}}\left(\mathbf{w}_{\mathsf{lat}}^\vee\right) \cdot \widehat{f_{\mathsf{lsc}}}\left(\mathbf{w}_{\mathsf{lsc}}^\vee\right), \tag{9.63}$$

where

$$f_{\mathsf{lat}}(\mathbf{w}_{\mathsf{lat}}) \overset{def}{=} \mathbb{P}\left(\mathbf{B}'\mathbf{w}_{\mathsf{lat}} \in \mathscr{S}\right), \quad f_{\mathsf{lsc}}(\mathbf{w}_{\mathsf{lsc}}) \overset{def}{=} \mathbb{P}\left(\mathbf{u} - \mathrm{Dec}(\mathbf{u}) = \mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}\right), \tag{9.64}$$

$$(\mathbf{r}_{\mathsf{lat}}, \mathbf{r}_{\mathsf{lsc}}) \overset{def}{=} \mathbf{r} \overset{def}{=} (\mathbf{e} + \mathbf{A}_{\mathsf{enu}}(\mathbf{s}_{\mathsf{enu}} - \widetilde{\mathbf{s}_{\mathsf{enu}}}) + \mathbf{A}_{\mathsf{fft}}(\mathbf{s}_{\mathsf{fft}} - \widetilde{\mathbf{s}_{\mathsf{fft}}}), \mathbf{s}_{\mathsf{lat}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}), \tag{9.65}$$

and

$$\mathbf{B}^{\mathsf{tmp}}_{\mathsf{global}} \overset{def}{=} \begin{bmatrix} \mathbf{I}_{\beta_{\mathsf{sieve}}} & \mathbf{0} \\ \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{B}'_{[m]} & \mathbf{I}_{n_{\mathsf{fft}}} \end{bmatrix}. \tag{9.66}$$

Here, $\mathbf{r}_{\mathsf{lat}}$ corresponds to the first $m + n_{\mathsf{lat}}$ coordinates of $\mathbf{r}$, while $\mathbf{r}_{\mathsf{lsc}}$ corresponds to its last n_{fft} coordinates. Additionally, under Assumption 5, we have that the random variable $\mathbf{u} - \mathrm{Dec}(\mathbf{u})$ is independent of $\mathbf{u}$ and follows the distribution induced by the polar code decoder.

Heuristic justification. For any vector in $\Lambda(\mathbf{B}')$, let $\mathbf{x}$ and $\mathbf{y}$ denote its first m coordinates and its next n_{lat} coordinates, respectively. Taking the conditional expectation over the randomness of both the polar code decoder and the short vector sampler, we obtain:

$$\begin{aligned}
\mathbb{E}_{\mathscr{S},\mathrm{Dec}}\left(F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{G}^{\mathsf{T}}\widetilde{\mathbf{s}_{\mathsf{fft}}})\right) &= \sum_{\substack{(\mathbf{x},\mathbf{y})\in\Lambda(\mathbf{B}') \\ \mathbf{c}_{\mathsf{lsc}}\in\Lambda(\mathcal{C}_{\mathsf{lsc}})}} \mathbb{P}\left((\mathbf{x},\mathbf{y}) \in \mathscr{S} \text{ and } \mathrm{Dec}\left(\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}\right) = \mathbf{c}_{\mathsf{lsc}}\right) \\
&\qquad \cdot \cos\left(\tfrac{2\pi}{q}\left(\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\rangle - \langle \mathbf{c}_{\mathsf{lsc}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}\rangle\right)\right) \\
&= \sum_{\substack{(\mathbf{x},\mathbf{y})\in\Lambda(\mathbf{B}') \\ \mathbf{w}_{\mathsf{lsc}}\in\Lambda(\mathcal{C}_{\mathsf{lsc}})+\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}}} \mathbb{P}\left((\mathbf{x},\mathbf{y}) \in \mathscr{S} \text{ and } \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathrm{Dec}\left(\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}\right) = \mathbf{w}_{\mathsf{lsc}}\right) \\
&\qquad \cdot \cos\left(\tfrac{2\pi}{q}\left(\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\rangle - \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathbf{w}_{\mathsf{lsc}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}\right\rangle\right)\right) \\
&= \sum_{\substack{(\mathbf{x},\mathbf{y})\in\Lambda(\mathbf{B}') \\ \mathbf{w}_{\mathsf{lsc}}\in\Lambda(\mathcal{C}_{\mathsf{lsc}})+\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}}} \mathbb{P}\left((\mathbf{x},\mathbf{y}) \in \mathscr{S}\right) \cdot \mathbb{P}\left(\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathrm{Dec}\left(\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}\right) = \mathbf{w}_{\mathsf{lsc}}\right) \\
&\qquad \cdot \cos\left(\tfrac{2\pi}{q}\left(\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\rangle - \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathbf{w}_{\mathsf{lsc}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}\right\rangle\right)\right)
\end{aligned}$$

where the last equality holds because of the independence of the decoding noise $\mathbf{u} - \mathrm{Dec}(\mathbf{u})$ from $\mathbf{u}$. Next, we simplify the inner expression inside the cosine function:

$$\begin{aligned}
&\langle \mathbf{x}, \mathbf{b} - \mathbf{A}_{\mathsf{enu}}\widetilde{\mathbf{s}_{\mathsf{enu}}}\rangle - \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathbf{w}_{\mathsf{lsc}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}\right\rangle \\
&= \langle \mathbf{x}, \mathbf{e} + \mathbf{A}_{\mathsf{enu}}(\mathbf{s}_{\mathsf{enu}} - \widetilde{\mathbf{s}_{\mathsf{enu}}}) + \mathbf{A}_{\mathsf{lat}}\mathbf{s}_{\mathsf{lat}} + \mathbf{A}_{\mathsf{fft}}\mathbf{s}_{\mathsf{fft}}\rangle - \left\langle \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathbf{w}_{\mathsf{lsc}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}\right\rangle \\
&= \langle \mathbf{x}, \mathbf{e} + \mathbf{A}_{\mathsf{enu}}(\mathbf{s}_{\mathsf{enu}} - \widetilde{\mathbf{s}_{\mathsf{enu}}}) + \mathbf{A}_{\mathsf{fft}}(\mathbf{s}_{\mathsf{fft}} - \widetilde{\mathbf{s}_{\mathsf{fft}}})\rangle + \left\langle \mathbf{A}_{\mathsf{lat}}^{\mathsf{T}}\mathbf{x}, \mathbf{s}_{\mathsf{lat}}\right\rangle + \langle \mathbf{w}_{\mathsf{lsc}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}\rangle \\
&= \langle (\mathbf{x}, \mathbf{y}, \mathbf{w}_{\mathsf{lsc}}), \mathbf{r}\rangle \bmod q
\end{aligned}$$

where $\mathbf{r}$ is as defined in Equation (9.65). So, we obtain

$$\mathbb{E}_{\mathscr{S},\mathrm{Dec}}\left(F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{G}^{\mathsf{T}}\widetilde{\mathbf{s}_{\mathsf{fft}}})\right) = \sum_{\substack{(\mathbf{x},\mathbf{y})\in\Lambda(\mathbf{B}') \\ \mathbf{w}_{\mathsf{lsc}}\in\Lambda(\mathcal{C}_{\mathsf{lsc}})+\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}}} \mathbb{P}\left((\mathbf{x},\mathbf{y}) \in \mathscr{S}\right) \cdot \mathbb{P}\left(\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x} - \mathrm{Dec}\left(\mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}\right) = \mathbf{w}_{\mathsf{lsc}}\right) \cdot \cos\left(2\pi\left\langle (\mathbf{x}, \mathbf{y}, \mathbf{w}_{\mathsf{lsc}}), \tfrac{\mathbf{r}}{q}\right\rangle\right).$$

We observe that if $(\mathbf{x}, \mathbf{y}) \in \Lambda(\mathbf{B}')$ and $\mathbf{w}_{\mathsf{lsc}} \in \Lambda(\mathcal{C}_{\mathsf{lsc}}) + \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}$, then it follows that $(-\mathbf{x}, -\mathbf{y}) \in \Lambda(\mathbf{B}')$ and $-\mathbf{w}_{\mathsf{lsc}} \in \Lambda(\mathcal{C}_{\mathsf{lsc}}) - \mathbf{A}_{\mathsf{fft}}^{\mathsf{T}}\mathbf{x}$. Furthermore, the probabilities involved in the formula are invariant under negation. This allows us to express the cosine function in its exponential

form, which yields

$$\begin{aligned}\mathbb{E}_{\mathscr{S},\mathrm{Dec}}\left(F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}\left(\mathbf{G}^{\intercal}\widetilde{\mathbf{s}_{\mathsf{fft}}}\right)\right) &= \sum_{\substack{(\mathbf{x},\mathbf{y})\in\Lambda(\mathbf{B}')\\ \mathbf{w}_{\mathsf{lsc}}\in\Lambda(\mathcal{C}_{\mathsf{lsc}})+\mathbf{A}_{\mathsf{fft}}^{\intercal}\mathbf{x}}} \mathbb{P}\left((\mathbf{x},\mathbf{y})\in\mathscr{S}\right)\cdot\mathbb{P}\left(\mathbf{A}_{\mathsf{fft}}^{\intercal}\mathbf{x}-\mathrm{Dec}\left(\mathbf{A}_{\mathsf{fft}}^{\intercal}\mathbf{x}\right)=\mathbf{w}_{\mathsf{lsc}}\right)\cdot e^{2i\pi\left\langle(\mathbf{x},\mathbf{y},\mathbf{w}_{\mathsf{lsc}}),\frac{\mathbf{r}}{q}\right\rangle}\\ &= \sum_{(\mathbf{w}_{\mathsf{lat}},\mathbf{w}_{\mathsf{lsc}})\in\Lambda(\mathbf{B}^{\mathsf{tmp}}_{\mathsf{global}})} f_{\mathsf{lat}}\left(\mathbf{w}_{\mathsf{lat}}\right)\cdot f_{\mathsf{lsc}}\left(\mathbf{w}_{\mathsf{lsc}}\right)\cdot e^{2i\pi\left\langle(\mathbf{B}'\mathbf{w}_{\mathsf{lat}},\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}),\frac{\mathbf{r}}{q}\right\rangle}.\end{aligned}$$

Note that the function f_{lsc} is defined in terms of a vector $\mathbf{u}$, which in this case depends on $\mathbf{w}_{\mathsf{lat}}$. However, the error vector produced by our polar code decoder is independent of the vector being decoded; that means the distribution f_{lsc} does not depend on $\mathbf{u}$. Finally, we heuristically suppose that we can apply Poisson summation formula to conclude. □

9.4.2.2 Second-level approximation

By estimating the Fourier transforms $\widehat{f_{\mathsf{lat}}}$ and $\widehat{f_{\mathsf{lsc}}}$ from Approximation 3, we derive a new approximation of the score function:

Approximation 4 (Second-Level Approximation)**.** *Based on Approximation 2, Assumptions 4 and 5, and assuming the Gaussian Heuristic holds, we have*

$$F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}\left(\mathbf{G}^{\intercal}\widetilde{\mathbf{s}_{\mathsf{fft}}}\right)\approx N\cdot\sum_{\substack{i\geqslant 0\\ j\geqslant 0}} N_{i,j}\cdot\int_0^{\infty}\psi_{\mathsf{lsc}}(d_{\mathsf{lsc}})\cdot\Phi_{d_{\mathsf{lsc}}}(i,j)dd_{\mathsf{lsc}} \tag{9.67}$$

where

$$\Phi_{d_{\mathsf{lsc}}}(i,j)\overset{def}{=}\Upsilon_{\frac{\beta_{\mathsf{sieve}}}{2}}\left(\tfrac{2\pi}{q}d_{\mathsf{lat}}i\right)\cdot\Upsilon_{\frac{n_{\mathsf{fft}}}{2}-1}\left(\tfrac{2\pi}{q}d_{\mathsf{lsc}}j\right), \tag{9.68}$$

$$\Upsilon_n(x)\overset{def}{=}\frac{\Gamma\left(n+1\right)J_n(x)}{(x/2)^n}=\sum_{\ell=0}^{+\infty}\frac{(-1)^{\ell}(x/2)^{2\ell}}{\ell!\prod_{s=1}^{\ell}(n+s)}, \tag{9.69}$$

$$\begin{aligned}N_{i,j}\overset{def}{=}\Big|\Big\{(\mathbf{w}_{\mathsf{lat}},\mathbf{w}_{\mathsf{lsc}})\in q\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)^{\vee}\ &+\mathbf{r}_{\mathsf{proj}}\subseteq\mathrm{span}\left(\mathbf{B}'\right)\times\mathbb{R}^{n_{\mathsf{fft}}}\\ &:\ |\mathbf{w}_{\mathsf{lat}}|=i\ \textit{and}\ |\mathbf{w}_{\mathsf{lsc}}|=j\Big\}\Big|,\end{aligned} \tag{9.70}$$

with $\mathbf{r}_{\mathsf{proj}}$*, the orthogonal projection on* $\mathrm{span}(\mathbf{B}_{\mathsf{global}})=\mathrm{span}(\mathbf{B}')\times\mathbb{R}^{n_{\mathsf{fft}}}$ *of*

$$\mathbf{r}\overset{def}{=}(\mathbf{e}+\mathbf{A}_{\mathsf{enu}}(\mathbf{s}_{\mathsf{enu}}-\widetilde{\mathbf{s}_{\mathsf{enu}}})+\mathbf{A}_{\mathsf{fft}}(\mathbf{s}_{\mathsf{fft}}-\widetilde{\mathbf{s}_{\mathsf{fft}}}),\mathbf{s}_{\mathsf{lat}},\widetilde{\mathbf{s}_{\mathsf{fft}}}), \tag{9.71}$$

and

$$\mathbf{B}_{\mathsf{global}}\overset{def}{=}\begin{array}{c}\overset{\beta_{\mathsf{sieve}}}{\longleftrightarrow}\ \overset{n_{\mathsf{fft}}}{\longrightarrow}\\ \left[\begin{array}{c|c}\mathbf{B}' & \mathbf{0}\\ \hline \mathbf{A}_{\mathsf{fft}}^{\intercal}\cdot\mathbf{B}'_{[m]} & \mathbf{B}_{\mathsf{lsc}}\end{array}\right]\begin{array}{l}\updownarrow m+n_{\mathsf{lat}}\\ \updownarrow n_{\mathsf{fft}}\end{array}\end{array}$$

Here:

- $\mathbf{B}' \in \mathbb{R}^{(m+n_{\mathsf{lat}})\times\beta_{\mathsf{sieve}}}$ *is a basis of the sublattice where the sampled short vectors lie,*
- $\mathbf{B}'_{[m]}$ *consists of the first* m *rows of* $\mathbf{B}'$*,*
- *and* $\mathbf{B}_{\mathsf{lsc}}$ *is a basis of the lattice* $\Lambda(\mathcal{C}_{\mathsf{lsc}})$*.*

To justify the above approximation, we first show that under Assumptions 4 and 5, and assuming the Gaussian Heuristic holds, the following two approximations hold:

$$\widehat{f_{\mathsf{lat}}}\left(\mathbf{w}_{\mathsf{lat}}^{\vee}\right) \approx N \cdot \Upsilon_{\frac{\beta_{\mathsf{sieve}}}{2}}\left(2\pi d_{\mathsf{lat}}\left|\mathbf{B}'\left(\mathbf{B}'^{\intercal}\mathbf{B}'\right)^{-1}\mathbf{w}_{\mathsf{lat}}^{\vee}\right|\right), \tag{9.72}$$

and

$$\widehat{f_{\mathsf{lsc}}}\left(\mathbf{w}_{\mathsf{lsc}}^{\vee}\right) \approx \int_0^{\infty} \psi_{\mathsf{lsc}}(d_{\mathsf{lsc}}) \cdot \Upsilon_{\frac{n_{\mathsf{fft}}}{2}-1}\left(\tfrac{2\pi}{q} d_{\mathsf{lsc}}\left|\mathbf{B}_{\mathsf{lsc}}^{-\top}\mathbf{w}_{\mathsf{lsc}}^{\vee}\right|\right) dd_{\mathsf{lsc}}. \tag{9.73}$$

On the one hand, under Assumption 4 and the Gaussian Heuristic, we can smooth and approximate f_{lat} by

$$f_{\mathsf{lat}}\left(\mathbf{w}_{\mathsf{lat}}\right) \approx \mathbb{1}_{\leqslant d_{\mathsf{lat}}}\left(\mathbf{B}'\mathbf{w}_{\mathsf{lat}}\right) \cdot N \cdot \frac{\mathcal{V}(\Lambda(\mathbf{B}'))}{\mathcal{V}\left(\mathsf{Ball}_{d_{\mathsf{lat}}}^{\beta_{\mathsf{sieve}}}\right)}. \tag{9.74}$$

Using the radial nature of both $\mathbb{1}_{\leqslant d_{\mathsf{lat}}}$ and $\widehat{\mathbb{1}_{\leqslant d_{\mathsf{lat}}}}$, and the facts that $\left|\mathbf{B}'\mathbf{w}_{\mathsf{lat}}\right| = \left|\sqrt{\mathbf{B}'^{\intercal}\mathbf{B}'}\mathbf{w}_{\mathsf{lat}}\right|$ and $\left|\sqrt{\mathbf{B}'^{\intercal}\mathbf{B}'}^{-\top}\mathbf{w}_{\mathsf{lat}}^{\vee}\right| = \left|\mathbf{B}'\left(\mathbf{B}'^{\intercal}\mathbf{B}'\right)^{-1}\mathbf{w}_{\mathsf{lat}}^{\vee}\right|$, we have

$$\begin{aligned}
\widehat{f_{\mathsf{lat}}}\left(\mathbf{w}_{\mathsf{lat}}^{\vee}\right) &\approx \frac{N\cdot\mathcal{V}(\Lambda(\mathbf{B}'))}{\mathcal{V}\left(\mathsf{Ball}_{d_{\mathsf{lat}}}^{\beta_{\mathsf{sieve}}}\right)} \int_{\mathbb{R}^{\beta_{\mathsf{sieve}}}} \mathbb{1}_{\leqslant d_{\mathsf{lat}}}\left(\mathbf{B}'\mathbf{w}_{\mathsf{lat}}\right) e^{-2i\pi\left\langle \mathbf{w}_{\mathsf{lat}}, \mathbf{w}_{\mathsf{lat}}^{\vee}\right\rangle} d\mathbf{w}_{\mathsf{lat}} \\
&= \frac{N\cdot\mathcal{V}(\Lambda(\mathbf{B}'))}{\mathcal{V}\left(\mathsf{Ball}_{d_{\mathsf{lat}}}^{\beta_{\mathsf{sieve}}}\right)} \int_{\mathbb{R}^{\beta_{\mathsf{sieve}}}} \mathbb{1}_{\leqslant d_{\mathsf{lat}}}\left(\sqrt{\mathbf{B}'^{\intercal}\mathbf{B}'}\mathbf{w}_{\mathsf{lat}}\right) e^{-2i\pi\left\langle \mathbf{w}_{\mathsf{lat}}, \mathbf{w}_{\mathsf{lat}}^{\vee}\right\rangle} d\mathbf{w}_{\mathsf{lat}} \\
&= \frac{N\cdot\mathcal{V}(\Lambda(\mathbf{B}'))}{\mathcal{V}\left(\mathsf{Ball}_{d_{\mathsf{lat}}}^{\beta_{\mathsf{sieve}}}\right)\cdot\sqrt{\det(\mathbf{B}'^{\intercal}\mathbf{B}')}} \cdot \int_{\mathbb{R}^{\beta_{\mathsf{sieve}}}} \mathbb{1}_{\leqslant d_{\mathsf{lat}}}\left(\mathbf{v}\right) e^{-2i\pi\left\langle \sqrt{\mathbf{B}'^{\intercal}\mathbf{B}'}^{-1}\mathbf{v}, \mathbf{w}_{\mathsf{lat}}^{\vee}\right\rangle} d\mathbf{v} \\
&= \frac{N}{\mathcal{V}\left(\mathsf{Ball}_{d_{\mathsf{lat}}}^{\beta_{\mathsf{sieve}}}\right)} \cdot \widehat{\mathbb{1}_{\leqslant d_{\mathsf{lat}}}}\left(\sqrt{\mathbf{B}'^{\intercal}\mathbf{B}'}^{-\top}\mathbf{w}_{\mathsf{lat}}^{\vee}\right) \\
&= \frac{N}{\mathcal{V}\left(\mathsf{Ball}_{d_{\mathsf{lat}}}^{\beta_{\mathsf{sieve}}}\right)} \cdot \widehat{\mathbb{1}_{\leqslant d_{\mathsf{lat}}}}\left(\mathbf{B}'\left(\mathbf{B}'^{\intercal}\mathbf{B}'\right)^{-1}\mathbf{w}_{\mathsf{lat}}^{\vee}\right)
\end{aligned}$$

where the Fourier transform of the indicator function of a ball can be expressed in term of the Bessel function:

$$\begin{aligned}
\widehat{\mathbb{1}_{\leqslant d_{\mathsf{lat}}}}\left(\mathbf{w}\right) &= \left(\frac{d_{\mathsf{lat}}}{|\mathbf{w}|}\right)^{\frac{\beta_{\mathsf{sieve}}}{2}} \cdot J_{\frac{\beta_{\mathsf{sieve}}}{2}}\left(2\pi d_{\mathsf{lat}}\left|\mathbf{w}\right|\right) \\
&= \mathcal{V}\left(\mathsf{Ball}_{d_{\mathsf{lat}}}^{\beta_{\mathsf{sieve}}}\right) \cdot \Upsilon_{\frac{\beta_{\mathsf{sieve}}}{2}}\left(2\pi d_{\mathsf{lat}}\left|\mathbf{w}\right|\right).
\end{aligned}$$

On the other hand, based on Assumption 5, we state the following approximation for f_{lsc}:

$$\begin{aligned}
f_{\mathsf{lsc}}\left(\mathbf{w}_{\mathsf{lsc}}\right) &\overset{\text{def}}{=} \mathbb{P}\left(\mathbf{u} - \mathrm{Dec}\left(\mathbf{u}\right) = \mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}\right) \\
&= \mathbb{P}\left(\left|\mathbf{u} - \mathrm{Dec}\left(\mathbf{u}\right)\right| = \left|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}\right|\right) \cdot \mathbb{P}\left(\mathbf{u} - \mathrm{Dec}\left(\mathbf{u}\right) = \mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}} \mid \left|\mathbf{u} - \mathrm{Dec}\left(\mathbf{u}\right)\right| = \left|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}\right|\right) \\
&\approx \left(\int_{\left|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}\right|}^{\sqrt{\left|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}\right|^2+1}} \psi_{\mathsf{lsc}}\left(d_{\mathsf{lsc}}\right) dd_{\mathsf{lsc}}\right) \cdot \frac{\mathcal{V}(\Lambda(\mathcal{C}_{\mathsf{lsc}}))}{\mathcal{V}\left(\mathsf{Ball}_{\sqrt{\left|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}\right|^2+1}}^{n_{\mathsf{fft}}}\right) - \mathcal{V}\left(\mathsf{Ball}_{\left|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}\right|}^{n_{\mathsf{fft}}}\right)}
\end{aligned}$$

where ψ_{lsc} is the probability density function of the normal distribution $\mathcal{N}\left(\mu_{\mathsf{lsc}}, \sigma_{\mathsf{lsc}}^2\right)$. In the context of our dual attack on KYBER, the decoding distance $|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|$ takes values in the hundreds or even thousands, as shown in Appendix 9.6.3, Table 9.3. Consequently, the difference $\sqrt{|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|^2 + 1} - |\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|$ is small, allowing us to approximate $f_{\mathsf{lsc}}(\mathbf{w}_{\mathsf{lsc}})$ by

$$\begin{aligned} f_{\mathsf{lsc}}(\mathbf{w}_{\mathsf{lsc}}) &\approx \frac{\left(\sqrt{|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|^2 + 1} - |\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|\right) \cdot \psi_{\mathsf{lsc}}\left(|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|\right) \cdot \mathcal{V}\left(\Lambda(\mathcal{C}_{\mathsf{lsc}})\right)}{\left(\sqrt{|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|^2 + 1} - |\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|\right) \cdot \mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|}\right)} \\ &= \psi_{\mathsf{lsc}}\left(|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|\right) \cdot \frac{\mathcal{V}\left(\Lambda(\mathcal{C}_{\mathsf{lsc}})\right)}{\mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|}\right)} \end{aligned}$$

Note that we must have

$$\sum_{\mathbf{w}_{\mathsf{lsc}} \in \mathbb{Z}^{n_{\mathsf{fft}}}} \mathbb{P}\left(\mathbf{u} - \mathrm{Dec}\left(\mathbf{u}\right) = \mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}\right) = 1.$$

Thus, the smoothed approximation of f_{lsc} should represent a probability density function, as we can verify by the following[7]:

$$\begin{aligned} \int_{\mathbb{R}^{n_{\mathsf{fft}}}} \psi_{\mathsf{lsc}}\left(|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|\right) \cdot \frac{\mathcal{V}(\Lambda(\mathcal{C}_{\mathsf{lsc}}))}{\mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|}\right)} d\mathbf{w}_{\mathsf{lsc}} &= \int_{\mathbb{R}^{n_{\mathsf{fft}}}} \psi_{\mathsf{lsc}}\left(|\mathbf{w}|\right) \cdot \frac{\mathcal{V}(\Lambda(\mathcal{C}_{\mathsf{lsc}}))}{\mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{|\mathbf{w}|}\right)} \cdot \frac{1}{\det(\mathbf{B}_{\mathsf{lsc}})} d\mathbf{w} \\ &= \int_{\mathbb{R}^{n_{\mathsf{fft}}}} \psi_{\mathsf{lsc}}\left(|\mathbf{w})|\right) \cdot \frac{1}{\mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{|\mathbf{w}|)}\right)} d\mathbf{w} \\ &= \int_0^{\infty} \psi_{\mathsf{lsc}}\left(d_{\mathsf{lsc}}\right) \int_{\mathsf{Sphere}^{n_{\mathsf{fft}}}_{d_{\mathsf{lsc}}}} \frac{1}{\mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{d_{\mathsf{lsc}}}\right)} d\sigma(\mathbf{s}) dd_{\mathsf{lsc}} \\ &= \int_0^{\infty} \psi_{\mathsf{lsc}}\left(d_{\mathsf{lsc}}\right) dd_{\mathsf{lsc}} \\ &\approx 1 \end{aligned}$$

where $d\sigma(\mathbf{s})$ represents the classical Lebesgue measure on the sphere $\mathsf{Sphere}^{n_{\mathsf{fft}}}_{d_{\mathsf{lsc}}}$. Finally, by using [CE01, Prop.2.1], we have

$$\begin{aligned} \widehat{f_{\mathsf{lsc}}}\left(\mathbf{w}^{\vee}_{\mathsf{lsc}}\right) &\approx \int_{\mathbb{R}^{n_{\mathsf{fft}}}} \psi_{\mathsf{lsc}}\left(|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|\right) \cdot \frac{\mathcal{V}(\Lambda(\mathcal{C}_{\mathsf{lsc}}))}{\mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{|\mathbf{B}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}}|}\right)} \cdot e^{-2i\pi\left\langle \mathbf{w}^{\vee}_{\mathsf{lsc}}, \mathbf{w}_{\mathsf{lsc}} \right\rangle} d\mathbf{w}_{\mathsf{lsc}} \\ &= \int_{\mathbb{R}^{n_{\mathsf{fft}}}} \psi_{\mathsf{lsc}}\left(|\mathbf{w}|\right) \cdot \frac{\mathcal{V}(\Lambda(\mathcal{C}_{\mathsf{lsc}}))}{\mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{|\mathbf{w}|}\right)} \cdot e^{-2i\pi\left\langle \mathbf{w}^{\vee}_{\mathsf{lsc}}, \mathbf{B}^{-1}_{\mathsf{lsc}}\mathbf{w}_{\mathsf{lsc}} \right\rangle} \cdot \frac{1}{\det(\mathbf{B}_{\mathsf{lsc}})} d\mathbf{w} \\ &= \int_{\mathbb{R}^{n_{\mathsf{fft}}}} \psi_{\mathsf{lsc}}\left(|\mathbf{w}|\right) \cdot \frac{1}{\mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{|\mathbf{w}|}\right)} \cdot e^{-2i\pi\left\langle \mathbf{B}^{-\top}_{\mathsf{lsc}}\mathbf{w}^{\vee}_{\mathsf{lsc}}, \mathbf{w}_{\mathsf{lsc}} \right\rangle} d\mathbf{w} \\ &= \frac{2\pi}{\left|\mathbf{B}^{-\top}_{\mathsf{lsc}}\mathbf{w}^{\vee}_{\mathsf{lsc}}\right|^{\frac{n_{\mathsf{fft}}}{2}-1}} \int_0^{\infty} \psi_{\mathsf{lsc}}(d_{\mathsf{lsc}}) \cdot \frac{\mathcal{V}(\Lambda(\mathcal{C}_{\mathsf{lsc}}))}{\mathcal{V}\left(\mathsf{Sphere}^{n_{\mathsf{fft}}}_{d_{\mathsf{lsc}}}\right)} \cdot d_{\mathsf{lsc}}^{\frac{n_{\mathsf{fft}}}{2}} \cdot J_{\frac{n_{\mathsf{fft}}}{2}-1}\left(2\pi d_{\mathsf{lsc}} \left|\mathbf{B}^{-\top}_{\mathsf{lsc}}\mathbf{w}^{\vee}_{\mathsf{lsc}}\right|\right) dd_{\mathsf{lsc}} \\ &= \int_0^{\infty} \psi_{\mathsf{lsc}}(d_{\mathsf{lsc}}) \cdot \Upsilon_{\frac{n_{\mathsf{fft}}}{2}-1}\left(2\pi d_{\mathsf{lsc}} \left|\mathbf{B}^{-\top}_{\mathsf{lsc}}\mathbf{w}^{\vee}_{\mathsf{lsc}}\right|\right) dd_{\mathsf{lsc}}. \end{aligned}$$

[7] We verified that $\int_{-\infty}^{0} \psi_{\mathsf{lsc}}(r)dr$ is negligible.

Finally, by leveraging Approximation 2 and Approximation 3, together with Approximations (9.72) and (9.73) for $\widehat{f_{\mathsf{lat}}}$ and $\widehat{f_{\mathsf{lsc}}}$, we can complete the justification for Approximation 4. First, we observe that for all $(\mathbf{w}_{\mathsf{lat}}^{\vee}, \mathbf{w}_{\mathsf{lsc}}^{\vee}) \in \Lambda(\mathbf{B}_{\mathsf{global}}^{\mathsf{tmp}})^{\vee} + \frac{(\mathbf{B}'^{\intercal}\mathbf{r}_{\mathsf{lat}}, \mathbf{B}_{\mathsf{lsc}}^{\intercal}\mathbf{r}_{\mathsf{lsc}})}{q}$, we have

$$q\left(\mathbf{B}'\left(\mathbf{B}'^{\intercal}\mathbf{B}'\right)^{-1}\mathbf{w}_{\mathsf{lat}}^{\vee}, \mathbf{B}_{\mathsf{lsc}}^{-\top}\mathbf{w}_{\mathsf{lsc}}^{\vee}\right) \in q\Lambda(\mathbf{B}_{\mathsf{global}})^{\vee} + \mathbf{r}_{\mathsf{proj}},$$

And so, combining all the terms and making the appropriate variable change, we obtain

$$F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}^{(\mathsf{lsc})}(\mathbf{G}^{\intercal}\widetilde{\mathbf{s}_{\mathsf{fft}}}) \approx N \cdot \int_0^{\infty} \psi_{\mathsf{lsc}}(d_{\mathsf{lsc}}) \cdot \sum_{\substack{(\mathbf{w}_{\mathsf{lat}}^{\vee}, \mathbf{w}_{\mathsf{lsc}}^{\vee}) \\ \in q\Lambda(\mathbf{B}_{\mathsf{global}})^{\vee} + \mathbf{r}_{\mathsf{proj}}}} \Upsilon_{\frac{\beta_{\mathsf{sieve}}}{2}}\left(\tfrac{2\pi}{q} d_{\mathsf{lat}} \left|\mathbf{w}_{\mathsf{lat}}^{\vee}\right|\right) \cdot \Upsilon_{\frac{n_{\mathsf{fft}}}{2}-1}\left(\tfrac{2\pi}{q} d_{\mathsf{lsc}} \left|\mathbf{w}_{\mathsf{lsc}}^{\vee}\right|\right) dd_{\mathsf{lsc}}.$$

We conclude by noting that the inner sum depends only on the lengths of $\mathbf{w}_{\mathsf{lat}}^{\vee}$ and $\mathbf{w}_{\mathsf{lsc}}^{\vee}$, that we denote i and j, respectively.

9.4.2.3 Third-level approximation

In Appendix 9.6.2, we provide an initial intuition for why the good guess can be distinguished from the wrong ones, based on Approximation 4. However, here we present more precise calculations.

Recall that Approximation 4 gives

$$F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}^{(\mathsf{lsc})}(\mathbf{G}^{\intercal}\widetilde{\mathbf{s}_{\mathsf{fft}}}) \approx N \cdot \int_0^{\infty} \psi_{\mathsf{lsc}}(d_{\mathsf{lsc}}) \cdot \sum_{i,j} N_{i,j} \cdot \Phi_{d_{\mathsf{lsc}}}(i,j) dd_{\mathsf{lsc}} \tag{9.75}$$

where $N_{i,j}$ is a random variable representing the number of pairs $(\mathbf{w}_{\mathsf{lat}}^{\vee}, \mathbf{w}_{\mathsf{lsc}}^{\vee}) \in q\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)^{\vee} + \mathbf{r}_{\mathsf{proj}} \subseteq \mathrm{span}\left(\mathbf{B}'\right) \times \mathbb{R}^{n_{\mathsf{fft}}}$ such that $|\mathbf{w}_{\mathsf{lat}}^{\vee}| = i$ and $|\mathbf{w}_{\mathsf{lsc}}^{\vee}| = j$.

Describing the distribution in Equation (9.75) is quite complex. To simplify this, we propose the following model:

Model 8. *We assume that* $F_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}^{(\mathsf{lsc})}(\mathbf{G}^{\intercal}\widetilde{\mathbf{s}_{\mathsf{fft}}})$ *approximately follows the same distribution as* $\mathcal{D} + \mathcal{N}(0, N/2)$*, where* $\mathcal{N}(0, N/2)$ *denotes a normal distribution with mean* 0 *and standard deviation* $\sqrt{N/2}$*, and*

$$\mathcal{D} \overset{def}{=} N \cdot \int_0^{\infty} \psi_{\mathsf{lsc}}(d_{\mathsf{lsc}}) \cdot \left(\max_{i,j\,:\,N_{i,j}=1} \left(\Phi_{d_{\mathsf{lsc}}}(i,j)\right)\right) dd_{\mathsf{lsc}}. \tag{9.76}$$

We recall that ψ_{lsc} *refers to the probability density function of* $\mathcal{N}(\mu_{\mathsf{lsc}}, \sigma_{\mathsf{lsc}}^2)$*.*

Based on Approximation 4 and Model 8, we can make the following two approximations, with probabilities calculated over the randomness of the guesses I_{enu} and $\widetilde{\mathbf{s}_{\mathsf{fft}}}$, as well as over the randomness of the LWE instance:

Approximation 5 (Good Guess)**.** *If we make the good guess* $(\widetilde{\mathbf{s}_{\mathsf{enu}}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}) = (\mathbf{s}_{\mathsf{enu}}, \mathbf{s}_{\mathsf{fft}})$ *and if we choose* T *around the expectation of* $F_{\mathbf{s}_{\mathsf{enu}}}^{(\mathsf{lsc})}(\mathbf{G}^{\intercal}\mathbf{s}_{\mathsf{fft}})$*, namely by defining*

$$\frac{T}{N} = \frac{\exp\left(\frac{-\alpha(\pi\mu_{\mathsf{lsc}}/q)^2}{1+2\alpha(\pi\sigma_{\mathsf{lsc}}/q)^2}\right)}{\sqrt{1+2\alpha(\pi\sigma_{\mathsf{lsc}}/q)^2}} \cdot \int_0^1 \beta_{\mathsf{sieve}} \cdot t^{\beta_{\mathsf{sieve}}-1} \cdot e^{-\alpha\left(\frac{\pi d_{\mathsf{lat}} t}{q}\right)^2} dt, \tag{9.77}$$

then

$$P_{\mathsf{good}} \overset{def}{=} \mathbb{P}\left(F_{\mathbf{s}_{\mathsf{enu}}}^{(\mathsf{lsc})}(\mathbf{G}^{\intercal}\mathbf{s}_{\mathsf{fft}}) \geqslant T\right) \approx 0.5 \tag{9.78}$$

Approximation 6 (Wrong Guess)**.** *If we make the wrong guess* $(\widetilde{\mathbf{s}_{\mathsf{enu}}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}) \neq (\mathbf{s}_{\mathsf{enu}}, \mathbf{s}_{\mathsf{fft}})$*, then*

$$P_{\mathsf{wrong}} \overset{def}{=} \mathbb{P}\left(F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}\left(\mathbf{G}^{\intercal}\widetilde{\mathbf{s}_{\mathsf{fft}}}\right) \geqslant T\right) \tag{9.79}$$

$$\approx \int_{-\infty}^{+\infty}\int_{0}^{+\infty} \min\left(1, \int_{\mathcal{E}(T-t)} \lambda(x)\mu(y)d(x,y)\right) \cdot \frac{e^{\frac{-t^2}{N} - \frac{(d_{\mathsf{lsc}}-\mu_{\mathsf{lsc}})^2}{2\sigma^2_{\mathsf{lsc}}}}}{\pi\sigma_{\mathsf{lsc}}\sqrt{2N}} dd_{\mathsf{lsc}}\, dt \tag{9.80}$$

where

$$\mathcal{E}(T-t) \overset{def}{=} \left\{(x,y) \in \mathbb{R}^2_+ \ : \ N \cdot \Phi_{d_{\mathsf{lsc}}}(x,y) \geqslant T - t\right\}, \tag{9.81}$$

$$\lambda(x) \overset{def}{=} \frac{2 \cdot \delta\left(\beta_{\mathsf{bkz}}\right)^{\beta_{\mathsf{sieve}}(m+n_{\mathsf{lat}}-\beta_{\mathsf{sieve}})} \cdot \pi^{\frac{\beta_{\mathsf{sieve}}}{2}} \cdot x^{\beta_{\mathsf{sieve}}-1}}{q^{\beta_{\mathsf{sieve}}\cdot\frac{m}{m+n_{\mathsf{lat}}}} \cdot \Gamma\left(\frac{\beta_{\mathsf{sieve}}}{2}\right)} \qquad \textit{and} \qquad \mu(y) \overset{def}{=} \frac{2 \cdot \pi^{\frac{n_{\mathsf{fft}}}{2}} \cdot y^{n_{\mathsf{fft}}-1}}{q^{k_{\mathsf{fft}}} \cdot \Gamma\left(\frac{n_{\mathsf{fft}}}{2}\right)}. \tag{9.82}$$

9.4.2.3.1 Justification of Approximation 5. The rationale behind Approximation 5 is that if we make the good guess $(\widetilde{\mathbf{s}_{\mathsf{enu}}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}) = (\mathbf{s}_{\mathsf{enu}}, \mathbf{s}_{\mathsf{fft}})$, then there exists an element

$$\mathbf{r}_{\mathsf{proj}} \overset{\text{def}}{=} (\mathbf{P}(\mathbf{e}, \mathbf{s}_{\mathsf{lat}}), \mathbf{s}_{\mathsf{fft}}) \in q\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)^{\vee} + \mathbf{r}_{\mathsf{proj}} \tag{9.83}$$

which has particularly small length. Specifically, the quantities $\frac{|\mathbf{P}(\mathbf{e},\mathbf{s}_{\mathsf{lat}})|}{\sqrt{\frac{\alpha}{2}}}$ and $\frac{|\mathbf{s}_{\mathsf{fft}}|}{\sqrt{\frac{\alpha}{2}}}$ approximately follow a χ-distribution[8] with degrees of freedom β_{sieve} and n_{fft}, respectively. Therefore, we obtain

$$\begin{aligned}
\mathbb{E}_{\chi_{\beta_{\mathsf{sieve}}},\chi_{n_{\mathsf{fft}}}}\left(F^{(\mathsf{lsc})}_{\mathbf{s}_{\mathsf{enu}}}\left(\mathbf{G}^{\intercal}\mathbf{s}_{\mathsf{fft}}\right)\right) &\approx \mathbb{E}_{\chi_{\beta_{\mathsf{sieve}}},\chi_{n_{\mathsf{fft}}}}(\mathcal{D}) \\
&\approx N \cdot \mathbb{E}_{d_{\mathsf{lsc}}}\left(\mathbb{E}_{\chi_{\beta_{\mathsf{sieve}}},\chi_{n_{\mathsf{fft}}}}\left(\Phi_{d_{\mathsf{lsc}}}\left(\sqrt{\tfrac{\alpha}{2}}\chi_{\beta_{\mathsf{sieve}}}, \sqrt{\tfrac{\alpha}{2}}\chi_{n_{\mathsf{fft}}}\right)\right)\right) \\
&= N \cdot \mathbb{E}_{\chi_{\beta_{\mathsf{sieve}}}}\left(\Upsilon_{\frac{\beta_{\mathsf{sieve}}}{2}}\left(\tfrac{2\pi}{q}d_{\mathsf{lat}}\sqrt{\tfrac{\alpha}{2}}\chi_{\beta_{\mathsf{sieve}}}\right)\right) \\
&\quad \cdot \mathbb{E}_{\chi_{n_{\mathsf{fft}}},d_{\mathsf{lsc}}}\left(\Upsilon_{\frac{n_{\mathsf{fft}}}{2}-1}\left(\tfrac{2\pi}{q}d_{\mathsf{lsc}}\sqrt{\tfrac{\alpha}{2}}\chi_{n_{\mathsf{fft}}}\right)\right)
\end{aligned}$$

where $\mathbb{E}_{d_{\mathsf{lsc}}}(\cdot)$ denotes the expectation with respect to the random variable d_{lsc}, which follows a normal distribution $\mathcal{N}(\mu_{\mathsf{lsc}}, \sigma^2_{\mathsf{lsc}})$.

Using Equation (9.69), each term in the above equation can be expressed in terms of the

[8] Strictly speaking, it is not an exact χ-distribution since the coordinates of $\mathbf{r}_{\mathsf{proj}}$ are not precisely normally distributed.

moments of the χ^2-distribution:

$$\begin{aligned}
\mathbb{E}_{\chi_{n_{\mathsf{fft}}},d_{\mathsf{lsc}}}\left(\Upsilon_{\frac{n_{\mathsf{fft}}}{2}-1}\left(\tfrac{2\pi}{q}d_{\mathsf{lsc}}\sqrt{\tfrac{\alpha}{2}}\chi_{n_{\mathsf{fft}}}\right)\right) &= \mathbb{E}_{d_{\mathsf{lsc}}}\left(\sum_{\ell=0}^{+\infty}\frac{\left(-\alpha\left(\frac{\pi d_{\mathsf{lsc}}}{q}\right)^2\right)^\ell}{\ell!}\cdot\frac{\mathbb{E}\left(\chi_{n_{\mathsf{fft}}}^{2\ell}\right)}{2^\ell\prod_{s=1}^{\ell}\left(\frac{n_{\mathsf{fft}}}{2}-1+s\right)}\right)\\
&= \mathbb{E}_{d_{\mathsf{lsc}}}\left(\sum_{\ell=0}^{+\infty}\frac{\left(-\alpha\left(\frac{\pi d_{\mathsf{lsc}}}{q}\right)^2\right)^\ell}{\ell!}\right)\\
&= \mathbb{E}_{d_{\mathsf{lsc}}}\left(e^{-\alpha\left(\frac{\pi d_{\mathsf{lsc}}}{q}\right)^2}\right)\\
&= \frac{\exp\left(\frac{-\alpha(\pi\mu_{\mathsf{lsc}}/q)^2}{1+2\alpha(\pi\sigma_{\mathsf{lsc}}/q)^2}\right)}{\sqrt{1+2\alpha(\pi\sigma_{\mathsf{lsc}}/q)^2}}
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}_{\chi_{\beta_{\mathsf{sieve}}}}\left(\Upsilon_{\frac{\beta_{\mathsf{sieve}}}{2}}\left(\tfrac{2\pi}{q}d_{\mathsf{lat}}\sqrt{\tfrac{\alpha}{2}}\chi_{\beta_{\mathsf{sieve}}}\right)\right) &= \sum_{\ell=0}^{+\infty}\frac{\left(-\alpha\left(\frac{\pi d_{\mathsf{lat}}}{q}\right)^2\right)^\ell}{\ell!}\cdot\frac{\mathbb{E}\left(\chi_{\beta_{\mathsf{sieve}}}^{2\ell}\right)}{2^\ell\prod_{s=1}^{\ell}\left(\frac{\beta_{\mathsf{sieve}}}{2}+s\right)}\\
&= \sum_{\ell=0}^{+\infty}\frac{\left(-\alpha\left(\frac{\pi d_{\mathsf{lat}}}{q}\right)^2\right)^\ell}{\ell!}\cdot\frac{\prod_{s=0}^{\ell-1}\left(\frac{\beta_{\mathsf{sieve}}}{2}+s\right)}{\prod_{s=0}^{\ell-1}\left(\frac{\beta_{\mathsf{sieve}}}{2}+1+s\right)}\\
&= \int_0^1\beta_{\mathsf{sieve}}\cdot t^{\beta_{\mathsf{sieve}}-1}\cdot e^{-\alpha\left(\frac{\pi d_{\mathsf{lat}}t}{q}\right)^2}dt
\end{aligned}$$

where the last equality is a well-known result concerning generalized hypergeometric functions[9].

Finally, in Equation (9.77), we chose T as the expected score $\mathbb{E}\left(F_{\mathbf{s}_{\mathsf{enu}}}^{(\mathsf{lsc})}\left(\mathbf{G}^\intercal\mathbf{s}_{\mathsf{fft}}\right)\right)$ of the good guess. Through experimentation, we verified that the expectation of this score is approximately equal to its median, which justifies Approximation 5.

9.4.2.3.2 Justification of Approximation 6. On the other hand, Approximation 6 is obtained by estimating the length of the short vectors in $q\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)^\vee+\mathbf{r}_{\mathsf{proj}}$, where $\mathbf{r}_{\mathsf{proj}}$ is no longer the shortest vector in the lattice coset. For i and j small enough, we can make the approximation that

$$\mathbb{P}\left(N_{i,j}>0\right)\approx\mathbb{P}\left(N_{i,j}=1\right). \tag{9.84}$$

[9]Note that $\int_0^1\beta_{\mathsf{sieve}}\cdot t^{\beta_{\mathsf{sieve}}-1}e^{-\alpha\left(\frac{\pi d_{\mathsf{lat}}t}{q}\right)^2}dt=\mathbb{E}_{t\sim\mathrm{Unif}\left(\mathsf{Ball}_{d_{\mathsf{lat}}}^{\beta_{\mathsf{sieve}}}\right)}\left(e^{-\alpha\left(\frac{\pi t}{q}\right)^2}\right)$.

Thus, the survival function of $\mathcal{D}$, knowing that the achieved decoding distance is d_{lsc}, can be approximated by

$$\mathbb{P}\left(\mathcal{D} > T \mid d_{\mathsf{lsc}}\right) \approx \mathbb{P}\left(\exists (i,j) \in \mathcal{E}(T) \ : \ N_{i,j} > 0\right) \tag{9.85}$$

$$\approx \min\left(1, \mathbb{E}\left(\sum_{\substack{(i,j)\in\mathcal{E}(T)\\(i^2,j^2)\in\mathbb{N}^2}} N_{i,j}\right)\right) \tag{9.86}$$

where

$$\mathcal{E}(T) \stackrel{\text{def}}{=} \left\{(i,j) \in \mathbb{R}_+^2 \ : \ N \cdot \Phi_{d_{\mathsf{lsc}}}(i,j) \geqslant T\right\}. \tag{9.87}$$

We have already observed that when we make the correct guess, the probability $\mathbb{P}\left(N_{i,j} > 0\right)$ is particularly high for a pair (i,j) close to $\left(\sqrt{\frac{\alpha\beta_{\mathsf{sieve}}}{2}}, \sqrt{\frac{\alpha n_{\mathsf{lsc}}}{2}}\right)$. Now, in the case where we make a wrong guess – that is $(\widetilde{\mathbf{s}_{\mathsf{enu}}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}) \neq (\mathbf{s}_{\mathsf{enu}}, \mathbf{s}_{\mathsf{fft}})$ – then we have

$$\mathbb{E}\left(\sum_{\substack{(i,j)\in\mathcal{E}(T)\\(i^2,j^2)\in\mathbb{N}^2}} N_{i,j}\right) \approx \frac{\int_{\mathcal{E}(T)} \mathcal{V}\left(\mathsf{Sphere}_x^{\beta_{\mathsf{sieve}}}\right) \cdot \mathcal{V}\left(\mathsf{Sphere}_y^{n_{\mathsf{fft}}}\right) d(x,y)}{\mathcal{V}\left(q\Lambda(\mathbf{B}_{\mathsf{global}})^\vee\right)}$$

$$= \int_{\mathcal{E}(T)} \frac{\mathcal{V}\left(\mathsf{Sphere}_x^{\beta_{\mathsf{sieve}}}\right)\cdot\mathcal{V}(\Lambda(\mathbf{B}'))}{q^{\beta_{\mathsf{sieve}}}} \cdot \frac{\mathcal{V}\left(\mathsf{Sphere}_y^{n_{\mathsf{fft}}}\cdot\mathcal{V}(\Lambda(\mathbf{B}_{\mathsf{lsc}}))\right)}{q^{n_{\mathsf{fft}}}} d(x,y)$$

The volume of $\Lambda(\mathbf{B}')$ is provided in Lemma 46, while the volume of $\Lambda(\mathbf{B}_{\mathsf{lsc}})$ is $q^{n_{\mathsf{fft}}-k_{\mathsf{fft}}}$. Meanwhile, the integral can be evaluated numerically. Note that the Gaussian Heuristic is necessary here, as q-ary lattices only behave approximately like random lattices.

Finally, under Model 8, P_{wrong} is the convolution product of the probability density function of the normal distribution $\mathcal{N}\left(0, N/2\right)$ and the survival function of $\mathcal{D}$, that is given by

$$\mathbb{P}\left(\mathcal{D} > T\right) = \int_0^\infty \psi_{\mathsf{lsc}}(d_{\mathsf{lsc}}) \cdot \mathbb{P}\left(\mathcal{D} > T \mid d_{\mathsf{lsc}}\right) dd_{\mathsf{lsc}}$$

9.4.2.4 Validating our analysis with simulations

We verify here the soundness of Approximation 6 for P_{wrong}, which is crucial for estimating the number of false candidates. To this end, we implemented and ran Algorithm 31, computing an experimental value for P_{wrong}, namely $\frac{\left|\{\mathbf{z}\in\mathbb{Z}_q^{k_{\mathsf{fft}}} : F_{\mathbf{0}}^{(\mathsf{lsc})}(\mathbf{z})\geqslant T\}\right|}{q^{k_{\mathsf{fft}}}}$ for different values of T. We plotted it against its theoretical approximation in Figure 9.1. Notably, we found that the experimental and theoretical estimates are in agreement, though the plot on the right suggests that our analysis may be slightly optimistic.

We used the g6k library [ADH+19] to generate short vectors in a lattice and used polar codes, along with the decoder described in Section 9.3.3, for the auxiliary code $\mathcal{C}_{\mathsf{lsc}}$. We provide the program used to generate Figure 9.1 in the GitHub repository[10].

[10] `https://github.com/kevin-carrier/CodedDualAttack/tree/main/verifyModel`

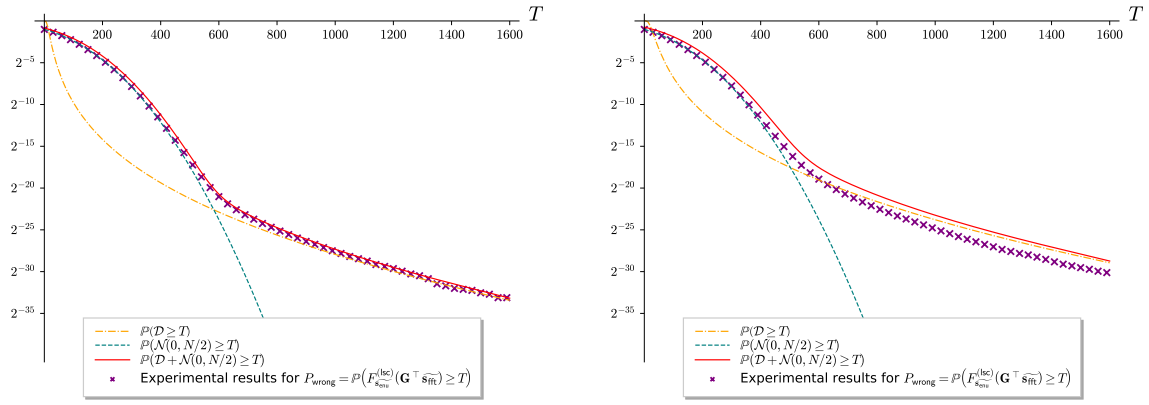


Figure 9.1: Experimental validation of Approximation 6 for P_{wrong}. The solid lines represent our theoretical model, while the crosses indicate results obtained from simulations. The theoretical model is drawn using the observed values for d_{lat} and d_{lsc}. In particular, d_{lsc} follows a normal distribution $\mathcal{N}(\mu_{\mathsf{lsc}}, \sigma_{\mathsf{lsc}}^2)$ with mean μ_{lsc} and standard deviation σ_{lsc} that are derived from the observed decoding distances of the $[n_{\mathsf{fft}}, k_{\mathsf{fft}}]_q$ polar codes used in the experiments. The experimental data were obtained by running 4000 iterations of Algorithm 31, with each iteration using an input $(\mathbf{A}, \mathbf{b})$ taken uniformly at random in $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. The parameters used are:
– on the left: $q = 241$, $m = 40$, $n = 43$, $n_{\mathsf{lat}} = 35$, $n_{\mathsf{enu}} = 0$, $n_{\mathsf{fft}} = 8$, $k_{\mathsf{fft}} = 3$, $N = 25971$, $\beta_{\mathsf{bkz}} = 32$, $\beta_{\mathsf{sieve}} = 44$, $d_{\mathsf{lat}} = 42.00$, $\mu_{\mathsf{lsc}} = 23.94$ and $\sigma_{\mathsf{lsc}} = 3.38$,
– on the right: $q = 241$, $m = 40$, $n = 50$, $n_{\mathsf{lat}} = 42$, $n_{\mathsf{enu}} = 0$, $n_{\mathsf{fft}} = 8$, $k_{\mathsf{fft}} = 3$, $N = 25970$, $\beta_{\mathsf{bkz}} = 35$, $\beta_{\mathsf{sieve}} = 41$, $d_{\mathsf{lat}} = 58.60$, $\mu_{\mathsf{lsc}} = 23.87$ and $\sigma_{\mathsf{lsc}} = 3.30$.

9.5 Application

In this section we give estimates for the cost of our attack against LWE problems from the literature. In particular, we consider NIST PQC standardized candidate Kyber [SAB+20]. We summarize in Table 9.1 the parameters of Kyber and outline the security level required by NIST, along with the attack complexities claimed by MATZOV. Note that MATZOV findings are not widely agreed upon in the cryptographic community as the analysis is based on independence assumptions which were strongly questioned in [DP23b].

The models C0, CC, and CN refer to different cost models for lattice reduction. These models are consistent with those presented in [AS22], and they can be described as follows:

C0. Cost estimates in the "Core-SVP" cost model [ADPS16a] for Algorithm 30 using [BDGL16] as the sieving oracle. This model assumes a single SVP call suffices to reduce a lattice. It furthermore assumes that all lower-order terms in the exponent are zero.

CC. Cost estimates in a classical circuit model [AGPS20b, SAB+20, MAT22] for Algorithm 30 using [BDGL16] as the sieving oracle. We derive these estimates by implementing the cost estimates from [MAT22], those tagged "asymptotic" (cf. [MAB+22]). This is the most detailed cost estimate available in the literature. However, we caution that these estimates, too, ignore the cost of memory access and thus may significantly underestimate the true cost. That is RAM access is not "free" (cf. [MAB+22]). This cost model is called "list_decoding-classical" in [AGPS20b].

CN. Cost estimates in a query model for Algorithm 30 using [BDGL16] as the sieving oracle. We include this cost model for completeness. This cost model is called "list_decoding-naive_classical" in [AGPS20b].

These various models rely on [BDGL16] for nearest-neighbor search, using fuzzy hashing functions based on product codes. It should be added that [Duc22] pointed out that the original analysis of [BDGL16] assumes an ideal case, underestimating the decoding cost and overlooking the suboptimal rate-distortion of product codes. Specifically, [Duc22] estimates that for a sieving dimension of 380, the complexity of the nearest-neighbor search increases by a factor of about 2^6. Here we kept the same way of computing the costs as in Matzov's paper [MAT22] to keep a fair comparison with their work. Delving further into this topic and replacing for instance the simple product codes with a better approach is outside the scope of this work.

For the same reasons, we only consider the classical RAM model. Adapting our analysis to a more realistic memory access model, such as the one presented in [Jaq24], is left as future work. Furthermore, we do not compare our results with the state-of-the-art primal attacks. A fair comparison would require substantial effort, as it involves accounting for many confounding factors, such as those listed in [DP23b, Appendix A]. Achieving a more precise estimate – down to the gate count level, as done for instance in [SAB+20] – would demand a significantly more in-depth analysis, which we leave as future work. Once again, our goal here is to provide a proof of concept for dual lattice attacks and to help resolve the controversy surrounding them. In line with this objective, we deliberately chose to work with a relatively simple and likely suboptimal dual attack algorithm. There are several potential avenues for improving it, including: (i) allowing false positives and adding a verification step to filter them out, (ii) combining modulus switching with our technique, as it would enable computations in characteristic 2, thereby allowing for a much more efficient fast Fourier transform, (iii) replacing the naive product code in the sieving procedure by a better quantizer... We leave these improvements for future work, as they fall outside the scope of this work.

9.5.1 Evaluating the complexity of our dual attack

We optimize the time complexity of Algorithm 31, as established in Theorem 10, under the assumptions outlined in Lemma 47. We assume that $1 - \mu \approx 1$, and we constrain the parameters N and T such that $\varepsilon := R\,P_{\mathsf{wrong}}\,q^{k_{\mathsf{fft}}}$ remains close to 1; in particular, in our setting, η is always greater than 0.62. Additionally, we select T following Approximation 5, ensuring that $P_{\mathsf{good}} \approx \frac{1}{2}$. At the same time, we ensure that ε remains close to 0. These choices guarantee that the overall success probability[11] of our algorithm is lower-bounded by approximately $0.31 - \varepsilon \approx 0.31$.

Our complexity results are summarized in the last three columns of Table 9.1. The associated parameters are given in Appendix 9.6.3, Table 9.3 and some relevant intermediate quantities are summarized in Appendix 9.6.3, Table 9.4. Note that the quantity d_{lat} is defined from the other parameters as in Lemma 45. The mean μ_{lsc} of the decoding distance d_{lsc} and its standard deviation σ_{lsc} are computed by choosing an $[n_{\mathsf{fft}}, k_{\mathsf{fft}}]_q$ polar code and by decoding many (a thousand) random words of $\mathbb{Z}_q^{n_{\mathsf{fft}}}$. For all instances of KYBER, we used a list size of $L = 1$ in the decoder (see Section 9.3.3, Lemma 48). This list size allows us to achieve a decoding distance close enough to the optimal decoding distance $d_{\mathrm{GV}} \stackrel{\mathrm{def}}{=} \sqrt{\frac{n_{\mathsf{fft}}}{2\pi e}} q^{1-\frac{k_{\mathsf{fft}}}{n_{\mathsf{fft}}}}$ without

[11] Note that in [MAT22], this success probability was approximately 0.25.

incurring additional cost in the complexity. Finally, we provide in the GitHub repository[12] a file verifying that our parameters achieve the complexity claims and verify all the constraints.

Scheme	LWE parameters			Security level required by NIST	MATZOV Complexity			Complexity of our Algorithm 31		
	q	n	α		C0	CC	CN	C0	CC	CN
Kyber-512	3329	512	3	AES-128 (143 bits)	115.4	139.2	134.4	**121.8**	**139.5**	**134.5**
Kyber-768	3329	768	2	AES-192 (207 bits)	173.7	196.1	190.6	**173.0**	**195.1**	**189.8**
Kyber-1024	3329	1024	2	AES-256 (272 bits)	241.8	262.4	256.1	**239.0**	**259.7**	**254.6**

Table 9.1: The LWE parameters for Kyber, the security level required by NIST, the claimed $\log_2$ complexity of MATZOV attack as given in [AS22, Table 2], and the $\log_2$ complexity of our dual attack Algorithm 31.

9.6 Appendices

9.6.1 Polar codes over a ring of integers

In this section we give more details about the construction of polar codes over $\mathbb{Z}_q$ that is mentioned in Subsection 9.3.3. We then verify that, when decoding random words, it is possible, to achieve a typical decoding distance which is very close to the lattice analogue of the Gilbert-Varshamov distance that we recall to be

$$\omega \approx \sqrt{\frac{n}{2\pi e}} q^{1-\frac{k}{n}}. \tag{9.88}$$

where n and k are respectively the length and the dimension of the polar code.

9.6.1.1 Construction

Let assume a codeword in $\mathbb{Z}_q^n$ of which each symbol is transmitted through a Gaussian channel of standard deviation $\sigma \stackrel{\text{def}}{=} \frac{\omega}{\sqrt{n}}$. The polar code construction basically consists of transforming those n Gaussian channels into n virtual channels that are, for most of them, either of maximal or minimal entropy. The idea then is to fix (or in other words *freeze*) the information that will transit via the bad channels and involve the good channels for the k symbols of useful information.

Our construction essentially follows the papers [ŞTA09, Chi14, Sav21]. We refer to those articles for more details about polar codes. It is a recursive construction which can be described as follows.

[12] `https://github.com/kevin-carrier/CodedDualAttack/tree/main/OptimizeCodedDualAttack`

Definition 57 ($(U+V, \alpha U)$-construction)**.** *Let U and V be two linear codes of the same length n over $\mathbb{Z}_q$ and let $\alpha \in \mathbb{Z}_q^*$ be an invertible scalar. The $(U+V, \alpha V)$ is a $\mathbb{Z}_q$-linear code of length $2n$ defined by*

$$(U+V, \alpha U) \stackrel{def}{=} \{(\mathbf{u}+\mathbf{v}, \alpha\mathbf{u}) : \mathbf{u} \in U \text{ and } \mathbf{v} \in V\} \tag{9.89}$$

A polar code of length $n \stackrel{\text{def}}{=} 2^m$ and dimension k is then defined by

Definition 58 (polar code)**.** *Let F be a subset of $\{0,1\}^m$ of size $2^m - k$ and let α be a function mapping the binary words of length $< m$ to $\mathbb{Z}_q^*$. The polar code of length 2^m associated to F and α is defined recursively by*

$$C \stackrel{def}{=} U_\varepsilon \tag{9.90}$$

where the $U_{\mathbf{x}}$ are codes of length 1 for all $\mathbf{x} \in \{0,1\}^m$ (we denote by ε the empty binary word) and are given by

$$U_{\mathbf{x}} = \begin{cases} \{0\} & \text{if } \mathbf{x} \in F \\ \mathbb{Z}_q & \text{otherwise} \end{cases} \tag{9.91}$$

and the other $U_{\mathbf{x}}$'s where $\mathbf{x}$ is a binary word of length $< m$ are defined recursively by

$$U_{\mathbf{x}} \stackrel{def}{=} \left(U_{0||\mathbf{x}} + U_{1||\mathbf{x}}, \alpha(\mathbf{x}) U_{0||\mathbf{x}}\right). \tag{9.92}$$

Thus, a polar code is fully defined by the set F of frozen positions and the $\alpha(\mathbf{x})$'s. In [Chi14], it is admitted that choosing the $\alpha(\mathbf{x})$'s uniformly at random in $\mathbb{Z}_q^*$ is good enough. However, in [Sav21], it is shown that those coefficients can be optimized. Thereafter, we do not use the optimization technique from [Sav21] but simply try several polar codes then choose the best of them. On another hand, we classically determine the optimal frozen positions F using Monte-Carlo simulation: we run many times a genie-aided decoder for estimating the probability distribution of each virtual channel then selecting the worst of them; that are the $2^m - k$ virtual channels for which the error probabilities are the highest.

9.6.1.2 Decoding algorithm

The Successive Cancelation (SC) decoding algorithm (see [ŞTA09, Chi14, Sav21]) can be described as a recursive decoding algorithm. For each code $U_{\mathbf{x}} \subseteq \mathbb{Z}_q^{2^{m-t}}$ such that $\mathbf{x} \in \{0,1\}^t$ and $t \in [\![0, m-1]\!]$, we decode a noisy codeword in this code by using recursively the decoders of $U_{0||\mathbf{x}}$ and $U_{1||\mathbf{x}}$.

Let $\mathbf{c} \stackrel{\text{def}}{=} (c_1, \cdots, c_{2^{m-t}}) \stackrel{\text{def}}{=} (\mathbf{u}+\mathbf{v}, \alpha(\mathbf{x})\mathbf{u})$ be a codeword in $U_{\mathbf{x}}$; *i.e.* $\mathbf{u} \stackrel{\text{def}}{=} (u_1, \cdots, u_{2^{m-t-1}})$ and $\mathbf{v} \stackrel{\text{def}}{=} (v_1, \cdots, v_{2^{m-t-1}})$ are respectively in $U_{0||\mathbf{x}}$ and $U_{1||\mathbf{x}}$. Let assume that $\mathbf{c}$ is transmitted through a channel $W^{(\mathbf{x})}$; let

$$\mathbf{y} \stackrel{\text{def}}{=} (y_1, \cdots, y_{2^{m-t}}) \stackrel{\text{def}}{=} (\mathbf{y}_\ell, \mathbf{y}_r) \in \mathbb{Z}_q^{2^{m-t-1}} \times \mathbb{Z}_q^{2^{m-t-1}} \tag{9.93}$$

be the received word. We assume that for each position $i \in [\![1, 2^{m-t}]\!]$ and symbol $s \in \mathbb{Z}_q$, we know the probability that the transmitted symbol is s knowing that the received one is y_i:

$$\Pi_i^{(\mathbf{x})}(s) \stackrel{\text{def}}{=} \mathbb{P}\left(c_i = s | y_i\right) \tag{9.94}$$

Instead of decoding directly $\mathbf{y}$, we decode first $\mathbf{y}_\ell - \alpha(\mathbf{x})^{-1}\mathbf{y}_r$ expecting to find $\mathbf{v} \in U_{1||\mathbf{x}}$. The virtual channel through which $\mathbf{v}$ has transited is then the serialization of two $W^{(\mathbf{x})}$ channels that we denote by $W^{(1||\mathbf{x})}$. Thus for each coordinate $i \in [\![1, 2^{m-t-1}]\!]$ and symbol $s \in \mathbb{Z}_q$, we have the probability

$$\begin{aligned}
\Pi_i^{(1||\mathbf{x})}(s) &\stackrel{\text{def}}{=} \mathbb{P}\left(v_i = s | y_i, y_{i+2^{m-t-1}}\right) && (9.95)\\
&= \sum_{s' \in \mathbb{Z}_q} \Pi_i^{(\mathbf{x})}(s + s') \cdot \Pi_{i+2^{m-t-1}}^{(\mathbf{x})}(\alpha(\mathbf{x}) \cdot s') && (9.96)\\
&= \sum_{s' \in \mathbb{Z}_q} \Pi_i^{(\mathbf{x})}(s - s') \cdot \Pi_{i+2^{m-t-1}}^{(\mathbf{x})}(-\alpha(\mathbf{x}) \cdot s') && (9.97)\\
&= \left(\Pi_i^{(\mathbf{x})} * \overline{\Pi}_{i+2^{m-t-1}}^{(\mathbf{x})}\right)(s) && (9.98)
\end{aligned}$$

where $\overline{\Pi}_i^{(\mathbf{x})}(s) \stackrel{\text{def}}{=} \Pi_i^{(\mathbf{x})}(-\alpha(\mathbf{x}) \cdot s)$.

On another hand, let us assume that the decoding of $\mathbf{y}_\ell - \alpha(\mathbf{x})^{-1}\mathbf{y}_r$ has led us to the vector $\widetilde{\mathbf{v}}$ that we expect to be $\mathbf{v}$ (for the genie-aided decoder used for the construction of the code, we actually take $\widetilde{\mathbf{v}} = \mathbf{v}$, regardless of the result of the decoding of $\mathbf{y}_\ell - \alpha(\mathbf{x})^{-1}\mathbf{y}_r$). We now have two independent noisy versions of the same vector $\mathbf{u}$ that are $\alpha(\mathbf{x})^{-1}\mathbf{y}_r$ and $\mathbf{y}_\ell - \widetilde{\mathbf{v}}$. In other words, supposing $\widetilde{\mathbf{v}} = \mathbf{v}$, the vector $\mathbf{u}$ has been sent twice through the channel $W^{(\mathbf{x})}$; we denote by $W^{(0||\mathbf{x})}$ the resulting channel and for each coordinate $i \in [\![1, 2^{m-t-1}]\!]$ and symbol $s \in \mathbb{Z}_q$, we have the probability

$$\begin{aligned}
\Pi_i^{(0||\mathbf{x})}(s) &\stackrel{\text{def}}{=} \mathbb{P}\left(u_i = s | y_i, y_{i+2^{m-t-1}}\right) && (9.99)\\
&= \frac{1}{\eta} \cdot \Pi_i^{(\mathbf{x})}(s + \widetilde{v}_i) \cdot \Pi_{i+2^{m-t-1}}^{(\mathbf{x})}(\alpha(\mathbf{x}) \cdot s) && (9.100)
\end{aligned}$$

where $\eta \stackrel{\text{def}}{=} \sum_{s' \in \mathbb{Z}_q} \Pi_i^{(\mathbf{x})}(s' + \widetilde{v}_i) \cdot \Pi_{i+2^{m-t-1}}^{(\mathbf{x})}(\alpha(\mathbf{x}) \cdot s')$ is a normalization factor.

Finally, for decoding a received word $\mathbf{y} \in \mathbb{Z}_q^{2^m}$ in the code U_ε that has been sent through a Gaussian channel of standard deviation σ, one essentially has to compute recursively the vector probabilities $\Pi_i^{(\mathbf{x})}$ for all $t \in [\![1, m]\!]$, $\mathbf{x} \in \{0,1\}^t$ and $i \in [\![1, 2^{m-t}]\!]$ using the Equations (9.98) and (9.100). Note that the initial channel $W^{(\varepsilon)}$ is the original Gaussian channel; so for all $i \in [\![1, 2^m]\!]$ and $s \in \mathbb{Z}_q$, we have

$$\Pi_i^{(\varepsilon)}(s) = \mathbb{P}\left(\mathcal{G}_{\mathbb{Z}_q,\sigma} = y_i - s\right) \tag{9.101}$$

where $\mathcal{G}_{\mathbb{Z}_q,\sigma}$ is the modular Gaussian distribution defined as follows:

Definition 59 (Discrete Gaussian Distribution)**.** *Let $\sigma > 0$ and let $\mathcal{S} \subset \mathbb{R}$ be a discrete set. The discrete Gaussian distribution $\mathcal{G}_{\mathcal{S},\sigma}$ over $\mathcal{S}$ is defined by:*

$$\mathbb{P}\left(\mathcal{G}_{\mathcal{S},\sigma} = x\right) \stackrel{\text{def}}{=} \frac{\rho_\sigma(x)}{\sum_{y \in \mathcal{S}} \rho_\sigma(y)} \tag{9.102}$$

where $\rho_\sigma(x) \stackrel{\text{def}}{=} \exp(-x^2/2\sigma^2)$ is the probability density function of the normal distribution $N(0, \sigma^2)$.

In particular, if $\mathcal{S} \overset{def}{=} \mathbb{Z}_q$ *then we speak of modular Gaussian distribution and for all* $x \in \mathbb{Z}_q$*, we have*

$$\mathbb{P}\left(\mathcal{G}_{\mathbb{Z}_q,\sigma} = x\right) = \mathbb{P}\left(\mathcal{G}_{\mathbb{Z},\sigma} \in x + q\mathbb{Z}\right) = \frac{\sum_{u \in x+q\mathbb{Z}} \rho_\sigma(u)}{\sum_{y \in \mathbb{Z}} \rho_\sigma(y)} \tag{9.103}$$

where x *is assimilated to any of its representatives.*

When arriving to the codes on the leaves – that are the codes $U_{\mathbf{x}}$ such that $\mathbf{x} \in \{0,1\}^m$ – then we can exhaustively decode $U_{\mathbf{x}}$:

1. if $\mathbf{x} \in F$ (meaning the corresponding symbol is frozen) then the only possible codeword in $U_{\mathbf{x}}$ is the symbol 0,
2. if $\mathbf{x} \notin F$, then we choose the maximum likelihood codeword in $U_{\mathbf{x}}$ that is the symbol s for which $\Pi_1^{(\mathbf{x})}(s)$ is the greatest.

The running time of Successive Cancelation decoding is given by the following lemma:

Lemma 49 (Complexity of the SC decoder)**.** *Assuming* q *is a power of* 2*. The running time for decoding a word in a polar code of length* 2^m *and dimension* k *over* $\mathbb{Z}_q$ *is:*

$$T_{\mathrm{SC}} \leqslant 3 \cdot \left(C_{\mathrm{add}} \cdot N_{\mathit{FFT}}^{(\mathrm{add})}(q) + C_{\mathrm{mul}} \cdot N_{\mathit{FFT}}^{(\mathrm{mul})}(q)\right) \cdot m \cdot 2^m \tag{9.104}$$

where C_{add} *and* C_{mul} *are the costs of an addition and a multiplication, respectively, and* $N_{\mathit{FFT}}^{(\mathrm{add})}(q)$ *and* $N_{\mathit{FFT}}^{(\mathrm{mul})}(q)$ *are the number of additions and multiplications needed to achieve a discrete Fourier transform over* $\mathbb{Z}_q$ *(Proposition 59 provides those numbers for* $q = 3329$*, which were computed using the* FFTW *software [FJ05]).*

Proof. For all $t \in [\![1, m]\!]$, $\mathbf{x} \in \{0,1\}^t$ and $i \in [\![1, 2^{m-t}]\!]$ – *i.e.* for $m \cdot 2^m$ triplets $(t, \mathbf{x}, i)$ – we can compute the vector of probabilities $\Pi_i^{(\mathbf{x})}$ with at most $3 \cdot q \cdot \log_2(q)$ multiplications. Indeed, we either have to compute Equation (9.98) or Equation (9.100). In the first case, it is a convolution; this can be done with the help of three fast Fourier transforms, with each FFT requiring $N_{\mathsf{FFT}}^{(\mathrm{add})}(q)$ additions and $N_{\mathsf{FFT}}^{(\mathrm{mul})}(q)$ multiplications. In the second case, we only have to do $2.q$ multiplications and q additions, which is less than the cost of a convolution. □

Remark 27. *We could reduce the cost of the SC decoder by considering the vectors of LLR (Log Likelihood Ratio) instead of the vectors of probabilities. This trick allows to transform multiplications into additions.*

9.6.1.3 List-Decoding

We can modify the above SC decoder to obtain a probabilistic decoder. To this end, when decoding non-frozen symbols in the codes on the leaves $U_{\mathbf{x}}$ where $\mathbf{x} \in \{0,1\}^m \setminus F$, then output the symbol s according to the distribution $\Pi_1^{(\mathbf{x})}$ instead of returning the one with the best probability. Note that as m tends to infinity and $\frac{k}{2^m}$ remains constant, for $\mathbf{x} \in \{0,1\}^m \setminus F$, the channels $W^{(\mathbf{x})}$'s have capacity very close to 1 and for $\mathbf{x} \in F$, they have capacity very close to 1. Because of this polarization phenomenon, we can prove similarly to [KU10] that our probabilistic SC decoder achieves an average decoding distance

$$d = \sqrt{\tfrac{2^m}{2\pi e}} \cdot q^{1-\frac{k}{2^m}} \cdot (1 + o()\,1) \tag{9.105}$$

when decoding a random vector in $\mathbb{Z}_q^{2^m}$.

To turn this probabilistic SC decoder into a list decoder, one only has to running it L times then choosing the codeword that minimizes the decoding distance. The complexity of a such algorithm is essentially L times the complexity of the SC decoder given by Lemma 49. Note that, contrary to some more classical list decoders of polar codes, the L decoding procedures of our algorithm can be trivially parallelized.

9.6.1.4 Puncturing

The polar codes construction above is about codes of length that are a power of 2. In our case, we may require codes of other length. A simple way for reducing the length of a code without changing its dimension is to puncture it. Let n, k be two positive integers. We build a linear code of length n and dimension k by puncturing a polar code of length 2^m and dimension k where $m \stackrel{\text{def}}{=} \lceil \log_2(n) \rceil$. Let denote by $\ell \stackrel{\text{def}}{=} 2^m - n$ the number of symbol to puncture. The puncturing operation essentially consists of ignoring the ℓ first symbols of the codeword; that is equivalent to suppose that the ℓ first physical channels through which transit the codewords are of maximal entropy:

$$\Pi_i^{(\varepsilon)}(s) \stackrel{\text{def}}{=} \frac{1}{q} \qquad \forall i \in [\![1, \ell]\!] \tag{9.106}$$

Note that we made this assumption both for the decoder and also for the genie-aided decoder used to determine the frozen positions.

9.6.2 A rough reason on why we are outside the contradictory regime

Here we explain in an informal way why it is reasonable, given the parameters of our attack, that the good guess is distinguishable from the bad guesses. Of course, we refer the reader to Section 9.4.2 for the complete formal analysis.

An important insight from Approximation 4 is that the score $F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{G}^\intercal \widetilde{\mathbf{s}_{\mathsf{fft}}})$ depends on the length enumerator of the vectors in the coset $q\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)^\vee + \mathbf{r}_{\mathsf{proj}}$. Specifically, for the good guess $(\widetilde{\mathbf{s}_{\mathsf{enu}}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}) = (\mathbf{s}_{\mathsf{enu}}, \mathbf{s}_{\mathsf{fft}})$, this lattice coset contains a particularly short vector $\mathbf{r}_{\mathsf{proj}} = (\mathbf{P}(\mathbf{e}, \mathbf{s}_{\mathsf{lat}}), \mathbf{s}_{\mathsf{fft}})$ where $\mathbf{P} \stackrel{\text{def}}{=} \mathbf{B}'\left(\mathbf{B}'^\intercal \mathbf{B}'\right)^{-1} \mathbf{B}'^\intercal$ is the orthogonal projection onto $\mathrm{span}(\mathbf{B}')$. In contrast, for wrong guesses where $(\widetilde{\mathbf{s}_{\mathsf{enu}}}, \widetilde{\mathbf{s}_{\mathsf{fft}}}) \neq (\mathbf{s}_{\mathsf{enu}}, \mathbf{s}_{\mathsf{fft}})$, the shortest vector is no longer $\mathbf{r}_{\mathsf{proj}}$. This observation provides a preliminary answer to the indistinguishability question posed in [DP23b]. Indeed, we cannot distinguish the good guess from the wrong ones if the length of $(\mathbf{P}(\mathbf{e}, \mathbf{s}_{\mathsf{lat}}), \mathbf{s}_{\mathsf{fft}})$ is greater than the length of the shortest vector in $q\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)^\vee + \mathbf{r}_{\mathsf{proj}}$. Given that the coordinates of $(\mathbf{e}, \mathbf{s}_{\mathsf{lat}}, \mathbf{s}_{\mathsf{fft}})$ are i.i.d. random variables following a centered binomial distribution with parameter α, we estimate the length of $(\mathbf{P}(\mathbf{e}, \mathbf{s}_{\mathsf{lat}}), \mathbf{s}_{\mathsf{fft}})$ to be

$$|(\mathbf{P}(\mathbf{e}, \mathbf{s}_{\mathsf{lat}}), \mathbf{s}_{\mathsf{fft}})| \approx \sqrt{\tfrac{\alpha(\beta_{\mathsf{sieve}} + n_{\mathsf{fft}})}{2}}. \tag{9.107}$$

On the other hand, if $\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)$ is treated as a random lattice with volume $V_{\mathsf{global}} \stackrel{\text{def}}{=} \mathcal{V}\left(\Lambda(\mathbf{B}_{\mathsf{global}})\right) = \mathcal{V}\left(\Lambda(\mathbf{B}')\right) \cdot \mathcal{V}\left(\Lambda(\mathbf{B}_{\mathsf{lsc}})\right)$, then, using the Gaussian Heuristic, we can estimate the length of the shortest vector in the lattice coset to be

$$\lambda_1\left(q\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)^\vee + \mathbf{r}_{\mathsf{proj}}\right) \approx \frac{q}{V_{\mathsf{global}}^{\frac{1}{\beta_{\mathsf{sieve}} + n_{\mathsf{fft}}}}} \cdot \sqrt{\tfrac{\beta_{\mathsf{sieve}} + n_{\mathsf{fft}}}{2\pi e}}. \tag{9.108}$$

However, we do not consider just one such coset, but rather $M \stackrel{\text{def}}{=} R \cdot q^{k_{\mathsf{fft}}}$. Therefore, the probability of having an even smaller shortest vector in one of the cosets is not negligible. In [DP23b, Section 4.3], the shortest vector, across all cosets, is estimated to be

$$\approx \frac{\lambda_1\left(q\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)^{\vee} + \mathbf{r}_{\mathsf{proj}}\right)}{M^{\frac{1}{\beta_{\mathsf{sieve}}+n_{\mathsf{fft}}}}}. \tag{9.109}$$

Table 9.2 compares (9.107) and (9.109) for the parameters derived for KYBER in Section 9.5.

Scheme	**C0**		**CC**		**CN**	
	Eq. (9.107)	Eq.(9.109)	Eq. (9.107)	Eq.(9.109)	Eq. (9.107)	Eq.(9.109)
KYBER-512	25.42	26.61	25.66	27.21	25.57	26.58
KYBER-768	25.63	26.70	25.82	27.23	25.75	27.19
KYBER-1024	30.25	31.17	30.38	31.99	30.48	32.34

Table 9.2: Comparison between the estimated length of $(\mathbf{P}(\mathbf{e}, \mathbf{s}_{\mathsf{lat}}), \mathbf{s}_{\mathsf{fft}})$ and the estimated length of the shortest vector in all the cosets $q\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)^{\vee} + \mathbf{r}_{\mathsf{proj}}$, based on the parameters provided in Table 9.3 (Appendix 9.6.3). We are outside the contradictory regime raised in [DP23b] as soon as $Eq.(9.107) < Eq.(9.109)$.

In Subsection 9.4.2, we refine these calculations to provide an accurate approximation of $F^{(\mathsf{lsc})}_{\widetilde{\mathbf{s}_{\mathsf{enu}}}}(\mathbf{G}^{\intercal}\widetilde{\mathbf{s}_{\mathsf{fft}}})$, which we validate through simulations. In particular, we no longer assume that $\Lambda\left(\mathbf{B}_{\mathsf{global}}\right)$ is a random lattice of volume V_{global}; instead, we separately analyze the first β_{sieve} coordinates and the last n_{fft} coordinates.

9.6.3 Parameter tables related to our complexity claim

Here we give the parameters related to the complexities given in Table 9.1.

C0:

Scheme	m	β_{bkz}	β_{sieve}	n_{enu}	n_{fft}	k_{fft}	n_{lat}	d_{lat}	μ_{lsc}	σ_{lsc}	$\log_2(N)$	$\log_2(T)$
Kyber-512	488	393	393	1	38	8	473	2882.14	975.17	42.11	81.55	43.82
Kyber-768	667	588	588	6	69	12	693	4401.02	1741.80	48.28	122.02	64.69
Kyber-1024	920	815	815	9	100	17	915	5190.00	2134.08	45.25	169.13	88.47

CC:

Scheme	m	β_{bkz}	β_{sieve}	n_{enu}	n_{fft}	k_{fft}	n_{lat}	d_{lat}	μ_{lsc}	σ_{lsc}	$\log_2(N)$	$\log_2(T)$
Kyber-512	475	384	387	5	52	9	455	2706.86	1528.73	47.38	80.31	43.31
Kyber-768	636	581	574	6	93	14	669	4015.56	2410.56	46.47	119.12	63.01
Kyber-1024	802	811	792	10	131	19	883	4683.71	2956.65	50.05	164.35	85.86

CN:

Scheme	m	β_{bkz}	β_{sieve}	n_{enu}	n_{fft}	k_{fft}	n_{lat}	d_{lat}	μ_{lsc}	σ_{lsc}	$\log_2(N)$	$\log_2(T)$
Kyber-512	457	384	388	4	48	9	460	2830.06	1292.55	39.26	80.51	43.42
Kyber-768	682	583	577	8	86	13	674	4083.09	2309.88	57.72	119.74	63.50
Kyber-1024	782	816	797	6	132	19	886	4678.80	2997.51	46.56	165.39	86.37

Table 9.3: Parameters to obtain Table 9.1.

C0:

Scheme	$\log_2(P_{\mathsf{wrong}})$	$\log_2(R)$	$\log_2(T_{\mathsf{sample}})$	$\log_2(N \cdot T_{\mathsf{dec}})$	$\log_2(T_{\mathsf{FFT}})$	η	$log_2(\varepsilon)$
Kyber-512	−103.25	2.84	115.76	118.95	112.13	0.91	−6.81
Kyber-768	−220.73	9.49	172.70	160.64	159.52	0.69	−70.83
Kyber-1024	−295.30	13.74	238.98	207.75	218.53	0.62	−82.65

CC:

Scheme	$\log_2(P_{\mathsf{wrong}})$	$\log_2(R)$	$\log_2(T_{\mathsf{sample}})$	$\log_2(N \cdot T_{\mathsf{dec}})$	$\log_2(T_{\mathsf{FFT}})$	η	$log_2(\varepsilon)$
Kyber-512	−119.57	9.39	139.51	117.71	124.00	0.66	−4.87
Kyber-768	−177.79	9.49	194.81	157.74	183.15	0.73	−4.49
Kyber-1024	−244.03	15.15	259.35	204.17	242.09	0.63	−6.57

CN:

Scheme	$\log_2(P_{\mathsf{wrong}})$	$\log_2(R)$	$\log_2(T_{\mathsf{sample}})$	$\log_2(N \cdot T_{\mathsf{dec}})$	$\log_2(T_{\mathsf{FFT}})$	η	$log_2(\varepsilon)$
Kyber-512	−120.51	7.71	143.30	117.91	124.00	0.71	−7.49
Kyber-768	−225.52	12.32	189.78	158.36	171.34	0.63	−61.09
Kyber-1024	−240.59	9.49	254.44	205.21	242.09	0.76	−8.78

Table 9.4: Intermediate results for Table 9.1. We recall that $P_{\mathsf{good}} \approx 0.5$. η and ε are defined in Lemma 47.

Chapter 10

Documentation for a software supporting our claims

Summary

This chapter contains, as is, the documentation for a software that is available at

`https://artifacts.iacr.org/eurocrypt/2024/a10/`

The software allows to:

1. Validate experimentally the Poisson model (see Model 5) used in the analysis of double-RLPN. It can plot figures similar to Fig. 6.2 comparing the experimental distribution of the score function against that given by the Poisson model. This is done with a C++ implementation of double-RLPN.

2. Verify the asymptotic complexity claims made for double-RLPN. This can be used to plot figures similar to Fig. 6.3 that gives the asymptotic complexity exponent of double-RLPN. It contains a dataset with some already optimized asymptotic parameters and a script computing the complexities from these parameters.

3. Verify our model (see Model 7) that predicts the simple score function for lattice-based dual attacks. This allows to plot figures closely related to the prediction of the score function that we gave in Fig. 8.2 of Chapter 8.

It was published as an associated software to verify the complexity claims of [CDMT24], all the notations used in this documentation are thus the ones of [CDMT24].

Documentation of the repository of the paper "Reduction from sparse LPN to LPN, Dual Attack 3.0"

June 7, 2024

Contents

1 Overview of the repository

References to Proposition, Figure or Model point to the eprint version of the article uploaded on December 4th:

`https://eprint.iacr.org/archive/2023/1852/1701452846.pdf`

Summary of each folder

- "Verify_Poisson_Model": A program to show that the poisson Model 1 is valid, it reproduces a figure close to Figure 2. It contains in particular parts of doubleRLPN implemented in C++. Documented in Section 2.
- "Lattice_Prediction": A program to show that we can predict the distribution of the score function of dual attacks in lattices. It essentially reproduces Figure 3 and Figure 4. Documented in Section 3.
- "Complexity_Claim": A program to verify the complexity claims relative to doubleRLPN. It contains in particular a dataset with the optimized asymptotic parameters of doubleRLPN to decode at the relative Gilbert-Varshamov distance. Documented in Section 4.

Dependencies

For "Verify_Poisson_Model"

- gcc/g++, available at `https://gcc.gnu.org`. Tested with version 13.1 but an older version with support for C++20 should suffice.

For "Verify_Poisson_Model" and "Complexity_Claim"

- python3, available at `https://www.python.org/downloads/`. Tested with version 3.11.3. Modules needed:
  - Python 3 standard Library
  - NumPy (Tested with version 1.24.3)
  - Scipy (Tested with version 1.10.1)
  - Matplotlib (Tested with version 3.7.1)

For "Lattice_Prediction"

- Jupyter notebook, available at `https://jupyter.org/`. Tested with version 6.5.4.
- SageMath, available at `https://www.sagemath.org/`. Tested with version 10.0.
- unzip.

Everything was tested on a 64 bit Arch-Linux distribution.

Acknowledgement

We would like to warmly thank the anonymous reviewers of Eurocrypt 2024's artifacts whose comments allowed to greatly improve the quality of this artifact.

2 Verification of the Poisson Model

In folder

`Verify_Poisson_Model/`

The goal here is to verify the Poisson Model which is used to bound the expected number of false candidates in Proposition 5, namely the quantity

$$\mathbb{E}\left(\left|\{\mathbf{x}\in\mathbb{F}_2^{k_{\text{aux}}}\setminus\{\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}\}\ :\ \widehat{f_{\mathbf{y},\mathscr{H},\mathbf{G}_{\text{aux}}}}(\mathbf{x})\geq T\}\right|\right).$$

The goal is to show that the expected number of false candidates is the same experimentally and by supposing that the Poisson Model is true.

Remark: This section does not exactly reproduce Figure 2 of the article. The latter was generated in the case where the set $\mathscr{H}$ of LPN samples is a random subset of $\widetilde{\mathscr{H}}$ of size N. While here we focus on the framework of Proposition 5, that is when $\mathscr{H} = \widetilde{\mathscr{H}}$, which is much simpler and shows in the same manner that the Poisson Model is valid.

Overview of the folder

- "plot.py": The main script. Plot the number of false candidates given by doubleRLPN against the number of false candidates given by the Poisson Model. This script uses scripts (that can be run independently) contained in the following two folders.
- "doubleRLPN": contains parts of doubleRLPN implemented in C++. Allow to compute the expected number of false candidates in doubleRLPN. Documented in Section 2.1.
- "Poisson_Model": computes the expected number of false candidates under the Poisson Model. Documented in Section 2.2.

How to run and what is does

`-python3 plot.py [--options]` $w\ t_{\texttt{aux}}\ k_{\texttt{aux}}\ s\ k\ n\ t\ N_{\texttt{iter}}$

Options:

- --d1. Create a dataset containing the expected number of false candidates given experimentally by doubleRLPN. More specifically, it runs the script documented in Section 2.1 with the same parameters.
- --d2. Create a dataset containing the expected number of false candidates given by the Poisson Model. More specifically, it runs the script documented in Section 2.2 with the same parameters.
- --plot. Combine the two previous datasets into a plot. This option must be either combined with option d1 and d2 if the corresponding datasets do not already exist, or can be used alone if the datasets already exist.

Alternatively, the script can be run without options which is equivalent to run it with --d1, --d2 and --plot all together.
Example:

```
-python3 plot.py 5 2 20 28 30 60 8 100
```

which is equivalent to running

```
-python3 plot.py --d1 --d2 --plot 5 2 20 28 30 60 8 100
```

Executing this command can take a few hours.

Typical output

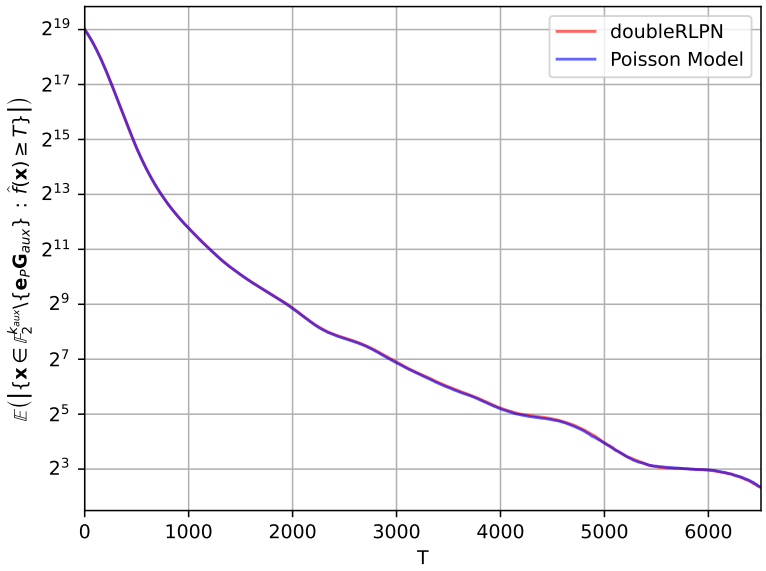
An image in

`plot/plot_`w`_`t_{aux}`_`k_{aux}`_`s`_`k`_`n`_`N_{iter}`.pdf`

Example with

`plot/plot_5_2_20_28_30_60_8_100.pdf`

The limit on the T axis of the plot is set to T such that the number of false candidates is equal to $\frac{500}{N_{\text{iter}}}$, this prevents the two curve from diverging from each other due to lack of data. Consider increasing N_{iter} to get information for larger T's. N_{iter} is advised to be more than 1000.



2.1 Number of false candidates in doubleRLPN

In folder

`Verify_Poisson_Model/doubleRLPN/`

What it does

Gives an empirical value for the expected number of false candidates in each iteration of doubleRLPN for different values of threshold T. More precisely: given the parameters of the algorithm $w, t_{\text{aux}}, k_{\text{aux}}, s, k, n, t$ and N_{iter} it runs a number N_{iter} of times the following procedure:

- **Do:**
  - Take $\mathcal{C}$ and $\mathcal{C}_{\text{aux}}$ uniformly at random in $[n,k]$ and $[s,k_{\text{aux}}]$ respectively by choosing two generator matrices $\mathbf{G}$ and $\mathbf{G}_{\text{aux}}$ uniformly at random among matrices of $\mathbb{F}_2^{k\times n}$ of rank k and matrices of $\mathbb{F}_2^{k_{\text{aux}}\times s}$ of rank k_{aux}. Compute $\mathbf{y}=\mathbf{c}+\mathbf{e}$ where $\mathbf{c}$ and $\mathbf{e}$ are taken uniformly at random in $\mathcal{C}$ and $\{\mathbf{x}\in\mathbb{F}_2^n \ : |\mathbf{x}|=t\}$ respectively. Choose uniformly at random two complementary subsets of $[\![1,n]\!]$, $\mathscr{P}$ and $\mathscr{N}$ of size s and $n-s$ respectively.

  While $\mathcal{C}_{\mathscr{P}}$ is not of dimension s.

- Compute the set of false candidates

$$\{\mathbf{x}\in\mathbb{F}_2^{k_{\text{aux}}}\setminus\{\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}\} \ : \widehat{f_{\mathbf{y},\widetilde{\mathscr{H}},\mathbf{G}_{\text{aux}}}}(\mathbf{x})\geq T\}$$

  where for $\mathbf{x}\in\mathbb{F}_2^{k_{\text{aux}}}$,

$$\widehat{f_{\mathbf{y},\widetilde{\mathscr{H}},\mathbf{G}_{\text{aux}}}}(\mathbf{x})=\sum_{(\mathbf{h},\mathbf{m}_{\text{aux}})\in\widetilde{\mathscr{H}}}(-1)^{\langle\mathbf{y},\mathbf{h}\rangle-\langle\mathbf{x},\mathbf{m}_{\text{aux}}\rangle}$$

  and

$$\widetilde{\mathscr{H}}=\{(\mathbf{h},\mathbf{m}_{\text{aux}})\in\mathcal{C}^{\perp}\times\mathcal{C}_{\text{aux}} \ : |\mathbf{h}_{\mathscr{N}}|=w \text{ and } |\mathbf{h}_{\mathscr{P}}+\mathbf{m}_{\text{aux}}\mathbf{G}_{\text{aux}}|=t_{\text{aux}}\}.$$

It outputs a file containing, for different values of T, the experimental average (computed over the N_{iter} iterations) number of false candidates.

How to run

`-python3 doubleRLPN.py` w $t_{\texttt{aux}}$ $k_{\texttt{aux}}$ s k n t $N_{\texttt{iter}}$

Example :

```
-python3 doubleRLPN.py 5 2 20 28 30 60 8 100
```

N_{iter} is advised to be more than 1000 if possible to get the most accurate estimation as possible.

Typical output

An output file in

`data/doubleRLPN_`w`_`$t_{\texttt{aux}}$`_`$k_{\texttt{aux}}$`_`s`_`k`_`n`_`$N_{\texttt{iter}}$`.csv`

of the format

T_1`,`y_{T_1}
T_2`,`y_{T_2}
...

where y_{T_i} is the average number of false candidates for the threshold T_i.

2.2 Number of false candidates under the Poisson Model

In folder

`Verify_Poisson_Model/Poisson_Model`

What it does

Gives an estimate of the expected number of false candidates under the Poisson Model. More precisely, similarly to Lemma 5 we can show that the expected number of false candidates can be rewritten as

$$\mathbb{E}_{\mathcal{C},\mathcal{C}_{\text{aux}}}\left(\left|\{\mathbf{x}\in\mathbb{F}_2^{k_{\text{aux}}}\setminus\{\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}\}\ :\widehat{f_{\mathbf{y},\widetilde{\mathscr{H}},\mathbf{G}_{\text{aux}}}}(\mathbf{x})\geq T\}\right|\right)= \left(2^{k_{\text{aux}}}-1\right)\mathbb{P}_{\mathcal{C},\mathcal{C}_{\text{aux}},\mathbf{x}}\left(\widehat{f_{\mathbf{y},\widetilde{\mathscr{H}},\mathbf{G}_{\text{aux}}}}(\mathbf{x})\geq T\right)$$

where $\mathcal{C}$ and $\mathcal{C}_{\text{aux}}$ uniformly at random in $[n,k]$ and $[s,k_{\text{aux}}]$ respectively and $\mathbf{x}$ is taken uniformly at random in $\mathbb{F}_2^{k_{\text{aux}}}\setminus\{\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}\}$. Using Lemma 1 and Proposition 4 we have that

$$\widehat{f_{\mathbf{y},\widetilde{\mathscr{H}},\mathbf{G}_{\text{aux}}}}=\frac{1}{2^{k-k_{\text{aux}}}}\sum_{i=0}^{n-s}\sum_{j=0}^{s}N_{i,j}K_w^{(n-s)}(i)\,K_{t_{\text{aux}}}^{(s)}(j)\,.$$

Then, under the Poisson model (replacing $N_{i,j}$ by a compound Poisson variable) we have that

$$\mathbb{E}\left(\left|\{\mathbf{x}\in\mathbb{F}_2^{k_{\text{aux}}}\setminus\{\mathbf{e}_{\mathscr{P}}\mathbf{G}_{\text{aux}}\}\ :\widehat{f_{\mathbf{y},\widetilde{\mathscr{H}},\mathbf{G}_{\text{aux}}}}(\mathbf{x})\geq T\}\right|\right)=\left(2^{k_{\text{aux}}}-1\right)\mathbb{P}\left(Z\geq T\right)\tag{1}$$

where

$$Z=\frac{1}{2^{k-k_{\text{aux}}}}\sum_{i=0}^{n-s}\sum_{j=0}^{s}\widetilde{N_{i,j}}K_w^{(n-s)}(i)\,K_{t_{\text{aux}}}^{(s)}(j)$$

and

$$\widetilde{N_{i,j}}\sim\text{Poisson}\left(\widetilde{N_j}\frac{\binom{n-s}{i}}{2^{n-k}}\right)\text{ and }\widetilde{N_j}\sim\text{Poisson}\left(\frac{\binom{s}{j}}{2^{k_{\text{aux}}}}\right)$$

and where the variables are independent.

Given the parameters of the algorithm $w, t_{\text{aux}}, k_{\text{aux}}, s, k, n, t$ and N_{iter}, this script estimates Equation (1) by a monte-carlo method: it draws N_{iter} $2^{k_{\text{aux}}}$ variables Z to heuristically estimate $\mathbb{P}\left(Z\geq T\right)$.

How to run

`-python3 PoissonModel.py` w t_{aux} k_{aux} s k n t N_{iter}

Example :

```
-python3 PoissonModel.py 5 2 20 28 30 60 8 100
```

N_{iter} is advised to be more than 1000 if possible to get the most accurate estimation as possible. This part is usually the longest and can take several hours with the parameters given as example. Consider parallelizing the code.

Typical output

An output file in

`data/PoissonModel_`w`_`t_{aux}`_`k_{aux}`_`s`_`k`_`n`_`N_{iter}`.csv`

of the format

T_1,y_{T_1}
T_2,y_{T_2}
...

where y_{T_i} is the average number of false candidates for the threshold T_i under the Poisson Model.

3 Prediction of lattice score function

In folder

`Lattice_Prediction/`

Overview of the folder

- `prediction_lattices.ipynb`
  - Reproduces Figure 3 and 4 for different parameters as described in Section 8 of the article. w appearing in Equation (19) is taken here as the average length of the short dual vectors returned by the sieve. They are stored in the following file.
- `out_n`**X**`_fft`**Y**`_enum`**Z**`.txt`
  - File containing information about the lattice and short dual vectors returned by the sieve. This file was created by showing the variables "Bprime" (before the call to the "reduce_and_sieve" function) and "dual_db" of `https://github.com/ludopulles/DoesDualSieveWork/tree/main/code/unif_score.py` with input $n =$ **X**, fft= **Y**, enum= **Z** (q is set to default to 3329).
- `Data_DP23/`
  - is taken from `https://github.com/ludopulles/DoesDualSieveWork/tree/main/data`

How to run

First, unzip the following compressed dataset:

`-unzip out_n90_fft22_enum26.zip`

then, run the notebook:

`-jupyter notebook prediction_lattices.ipynb`

4 Verification of complexity claims

In folder

`Complexity_Claim/`

The files are meant to verify the complexity claims relative to doubleRLPN.

Overview of the folder

- `doubleRLPN_BJMM12.csv`
  - Contains, for different code rates R, the optimized relative parameters and the associated complexity of the doubleRLPN decoder to decode at the relative Gilbert-Varshamov distance when using BJMM12 technique to compute low-weight parity-checks. These parameters are used in Proposition 9 to compute the asymptotic complexity of the algorithm. The file contains, for different rates R the values of $\sigma, R_{\text{aux}}, \nu, \omega, \tau$ along with $\lambda_1, \lambda_2, \pi_1, \pi_2$, the later 4 parameters are used in Proposition 11 to compute the complexity of computing the parity-checks using BJMM12 technique. All the parameters (even $\lambda_1, ...$) are written relatively to n. τ_{aux} is implicitly set to be equal to $\sigma h_2^{-1}\left(1 - \frac{R_{\text{aux}}}{\sigma}\right)$ and N_{aux} is implicitly set to be equal to 1. The parameters relative to the two subroutines Dumer-Decoder and Solve-SubProblem will be computed on the fly in the following file.
- `complexity_doubleRLPN_BJMM12.py`
  - Using the relative parameters contained in the parameter file, this script re-computes, using the formula in Proposition 9, the time complexity exponent ($\alpha_{\text{doubleRLPN}}$) of the doubleRLPN decoder. This script also assert that the parameters meet the constraints of Proposition 9 and Proposition 11 (executions fails if one constraint is not verified).

How to run

```
-python3 complexity_doubleRLPN_BJMM12.py
```

Typical output

A list of complexity exponent

```
Rate: 0.01000; Complexity: 0.00539
Rate: 0.02000; Complexity: 0.01009
...
```

Conclusion

We significantly developed code-based dual attacks. Our best dual attack significantly asymptotically beats all previously known generic decoders for rates smaller than 0.42 at the Gilbert-Varshamov distance and is provable without using any assumptions. Second, we devised a new lattice-based dual attack that improves upon previous lattice-based dual attacks. We developed new tools to analyze lattice-based dual attacks without using the flawed independence assumptions. In particular, we show that our attack diminishes the security of the NIST standard Kyber. This positively answers the recent controversy that questioned whether a lattice-based dual-"sieve" attack could really be competitive. Here are some future works we would like to investigate in relation to our code-based dual attacks.

- Note that double-RLPN could be improved in several ways. In a previous version of [CDMT24] that we never published but that was submitted in October 2022 to Eurocrypt 2023 we added an extra Prange bet that $\mathbf{e}_{\mathscr{P}}$ was zero on a few coordinates. This allows us to diminish the dimension of the underlying LPN problem and we can iterate this bet ($\mathscr{P}$ stays fixed but at each iteration we bet that some random positions of $\mathscr{P}$ are error free). The gain comes from the fact that we can reuse the dual vectors of low weight on $\mathscr{N}$ at each iteration. We did not present this strategy here but it allows a slight gain in the complexity exponent. Second, we only studied our algorithm when the Dumer or BJMM subroutine are used to compute the low-weight dual vectors but surely using more advanced techniques such as an iteration of [BM18] or maybe the recent sieving algorithms will yield better results.

- In Chapter 6, we quickly mentioned that the complexity of double-RLPN at rate R when the error weight $t = o(n)$ is sublinear is $2^{-t\log_2(1-R)(1+o(1))n}$. So, and contrary to Statistical decoding and RLPN, this seems like a reasonable algorithm in the sublinear regime. Indeed, the first-order term $-t\log_2(1-R)$ is that of all ISD's in this regime. The question remains to evaluate in practice its complexity against schemes like HQC.

- We would like to investigate the potential of dual attacks against variants of the decoding problem we targeted in this thesis. For example when the code is quasi-cyclic. We would also like to investigate the Hamming q-ary decoding problem and variants when the error has a regular shape, or when the error lies in a subgroup of $\mathbb{F}_q^n$ as in the recent CROSS signature scheme.

- We reduced the analysis of RLPN and double-RLPN to devising exponential concentration bounds on the weight enumerator of random linear codes, see Eq. (5.6). We would like to investigate to what extent these exponential concentration bounds are true.

Bibliography

[AAB+22a] Carlos Aguilar Melchor, Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Santosh Ghosh, Shay Gueron, Tim Güneysu, Rafael Misoczki, Edoardo Persichetti, Jan Richter-Brockmann, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zémor. BIKE. Round 4 Submission to the NIST Post-Quantum Cryptography Call, v. 5.1, October 2022.

[AAB+22b] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jurjen Bos, Jean-Christophe Deneuville, Arnaud Dion, Philippe Gaborit, Jérôme Lacan, Edoardo Persichetti, Jean-Marc Robert, Pascal Véron, Gilles Zémor, and Jurjen Bos. HQC. Round 4 Submission to the NIST Post-Quantum Cryptography Call, October 2022. `https://pqc-hqc.org/`.

[ABC+22] Martin Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Mizoczki, Ruben Niederhagen, Edoardo Persichetti, Kenneth Paterson, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wang Wen. Classic McEliece (merger of Classic McEliece and NTS-KEM). `https://classic.mceliece.org`, November 2022. Fourth round finalist of the NIST post-quantum cryptography call.

[AD221] Lattice Attacks on NTRU and LWE: A History of Refinements, page 15–40. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021.

[ADH+19] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In Yuval Ishai and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2019, pages 717–746, Cham, 2019. Springer International Publishing.

[ADPS16a] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, USENIX Security 2016, pages 327–343. USENIX Association, August 2016.

[ADPS16b] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - New Hope. In Thorsten Holz and Stefan Savage, editors,

25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016., pages 327–343. USENIX Association, 2016.

[AFFP14] Martin R. Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. modulus switching for the BKW algorithm on LWE. In Hugo Krawczyk, editor, PKC 2014, volume 8383 of LNCS, pages 429–445, Heidelberg, March 2014. Springer.

[AGPS20a] Martin Albrecht, Vlad Gheorghiu, Eamonn Postlethwaite, and John Schanck. Estimating Quantum Speedups for Lattice Sieves, pages 583–613. 12 2020.

[AGPS20b] Martin R. Albrecht, Vlad Gheorghiu, Eamonn W. Postlethwaite, and John M. Schanck. Estimating quantum speedups for lattice sieves. In Shiho Moriai and Huaxiong Wang, editors, ASIACRYPT 2020, Part II, volume 12492 of LNCS, pages 583–613. Springer, Heidelberg, December 2020.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, STOC '01, page 601–610, New York, NY, USA, 2001. Association for Computing Machinery.

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II, volume 10211 of Lecture Notes in Computer Science, pages 103–129, 2017.

[Ale03] Alekhnovich, Michael. More on Average Case vs Approximation Complexity. In 44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cam pages 298–307. IEEE Computer Society, 2003.

[Alm19] Josh Alman. An Illuminating Algorithm for the Light Bulb Problem. In Jeremy T. Fineman and Michael Mitzenmacher, editors, 2nd Symposium on Simplicity in Algorithms (SOSA 2019), volume 69 of Open Access Series in Informatics (OASIcs), pages 2:1–2:11, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[AMBB+24] Carlos Aguilar Melchor, Slim Bettaieb, Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Nicolas Gama, Shay Gueron, James Howe, Andreas Hülsing, David Joseph, Antoine Joux, Mukul Kulkarni, Edoardo Persichetti, Tovohery Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDitH. Round 2 Additional Signatures to the NIST Post-Quantum Cryptography: Digital Signature Schemes Call, May 2024.

[AR04] Dorit Aharonov and Oded Regev. Lattice problems in NP cap coNP. In 45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings, pages 362–371. IEEE Computer Society, 2004.

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in NP cap coNP. J. ACM, 52(5):749–765, 2005.

[Arı09] Erdal Arıkan. Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. IEEE Trans. Inform. Theory, 55(7):3051–3073, 2009.

[AS22] Martin R. Albrecht and Yixin Shen. Quantum augmented dual attack. Cryptology ePrint Archive, Paper 2022/656, 2022.

[AZ23] Josh Alman and Hengjie Zhang. Generalizations of matrix multiplication can solve the light bulb problem. In 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS), pages 1471–1495, 2023.

[Bar97] Alexander Barg. Complexity issues in coding theory. Electronic Colloquium on Computational Complexity, October 1997.

[BC07] Marco Baldi and Franco Chiaraluce. Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In Proc. IEEE Int. Symposium Inf. Theory - ISIT, pages 2591–2595, Nice, France, June 2007.

[BCJ11] Anja Becker, Jean-Sébastien Coron, and 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, pages 364–385, 2011.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, pages 10–24. SIAM, 2016.

[BE98] Simon Berkovich and Eyas El-Qawasmeh. Reversing the error-correction scheme for a fault-tolerant indexing. In Proceedings DCC '98 Data Compression Conference, page 527, March 1998.

[BGJ15] Anja Becker, Nicolas Gama, and Antoine Joux. Speeding-up lattice sieving without increasing the memory, using sub-quadratic nearest neighbor search. IACR Cryptology ePrint Archive, Report 2015/522, 2015. `http://eprint.iacr.org/2015/522`.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In Advances in Cryptology - EUROCRYPT 2012, LNCS. Springer, 2012.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. Journal of the ACM (JACM), 50(4):506–519, 2003.

[BM17] Leif Both and Alexander May. Optimizing BJMM with Nearest Neighbors: Full Decoding in $2^{2/21n}$ and McEliece Security. In WCC Workshop on Coding and Cryptography, September 2017.

[BM18] Leif Both and Alexander May. Decoding linear codes with high error rate and its impact for LPN security. In Tanja Lange and Rainer Steinwandt, editors, Post-Quantum Cryptography 2018, volume 10786 of LNCS, pages 25–46, Fort Lauderdale, FL, USA, April 2018. Springer.

[BMvT78] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. IEEE Trans. Inform. Theory, 24(3):384–386, May 1978.

[BTV15] Sonia Bogos, Florian Tramer, and Serge Vaudenay. On solving LPN using BKW and variants. IACR Cryptology ePrint Archive, Report2015/049, 2015. `http://eprint.iacr.org/`.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011, pages 97–106. IEEE Computer Society, 2011.

[BV16] Sonia Bogos and Serge Vaudenay. Optimization of LPN solving algorithms. In Jung Hee Cheon and Tsuyoshi Takagi, editors, Advances in Cryptology - ASIACRYPT 2016, pages 703–728, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[Car20] Kevin Carrier. Recherche de presque-collisions pour le décodage et la reconnaissance de codes c Theses, Sorbonne Université, June 2020.

[CDMT22] Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Statistical decoding 2.0: Reducing decoding to LPN. In Advances in Cryptology - ASIACRYPT 2022, LNCS. Springer, 2022.

[CDMT24] Kévin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. Reduction from sparse LPN to LPN, dual attack 3.0. In Marc Joye and Gregor Leander, editors, Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI, volume 14656 of LNCS, pages 286–315. Springer, 2024. Artifact available at `https://artifacts.iacr.org/eurocrypt/2024/a10/`.

[CE01] Henry Cohn and Noam Elkies. New upper bounds on sphere packings i. Annals of Mathematics, 157, 10 2001.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: Fast fully homomorphic encryption over the torus. J. Cryptol., 33(1):34–91, January 2020.

[Che13] Yuanmi Chen. Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe. 2013.

[Chi14] Mao-Ching Chiu. Non-binary polar codes with channel symbol permutations. In 2014 International Symposium on Information Theory and its Applications, pages 433–437, 2014.

[CMST25] Kevin Carrier, Charles Meyer-Hilfiger, Yixin Shen, and Jean-Pierre Tillich. Assessing the impact of a variant of matzov's dual attack on kyber. 2025. To appear in Advances in Cryptology - CRYPTO 2025. ePrint `https://eprint.iacr.org/2022/1750`.

[CS88] John H. Conway and Neil J. A. Sloane. Sphere Packings, Lattices and Groups, volume 290 of Grundlehren der mathematischen Wissenschaften. Springer, 1988.

[CS15] Rodolfo Canto-Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight, 2015. preprint.

[CS16] Rodolfo Canto-Torres and Nicolas Sendrier. Analysis of information set decoding for a sub-linear error weight. In Post-Quantum Cryptography 2016, LNCS, pages 144–161, Fukuoka, Japan, February 2016.

[CST22] Kevin Carrier, Yixin Shen, and Jean-Pierre Tillich. Faster dual lattice attacks by using coding theory. Cryptology ePrint Archive, Paper 2022/1750, 2022. `https://eprint.iacr.org/archive/2022/1750/20221220:184709`.

[CT65] James Cooley and John Tukey. An algorithm for the machine calculation of complex fourier series. Mathematics of Computation, 19(90), 1965.

[dC97] D. de Caen. A lower bound on the probability of a union. Discrete Math., 169(1–3):217–220, May 1997.

[DDRT23] Thomas Debris-Alazard, Léo Ducas, Nicolas Resch, and Jean-Pierre Tillich. Smoothing codes and lattices: Systematic study and new bounds. IEEE Trans. Inform. Theory, 69(9):6006–6027, 2023.

[Deb23] Thomas Debris-Alazard. Code-based cryptography: Lecture notes, 2023. `https://arxiv.org/abs/2304.03541`.

[DEEK24] Léo Ducas, Andre Esser, Simona Etinski, and Elena Kirshanova. Asymptotics and improvements of sieving for codes. In Marc Joye and Gregor Leander, editors, Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI, volume 14656 of Lecture Notes in Computer Science, pages 151–180. Springer, 2024.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. IEEE transactions on Information Theory, 22(6):644–654, 1976.

[DHL+94] Danny Dolev, Yuval Harari, Nathan Linial, Noam Nisan, and Michal Parnas. Neighborhood preserving hashing and approximate queries. In Daniel Dominic Sleator, editor, Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. 23-25 January 1994, Arlington, Virginia, USA., pages 251–259. ACM/SIAM, 1994.

[DM90] Dan E. Dudgeon and Russell M. Mersereau. Multidimensional Digital Signal Processing. Prentice Hall Professional Technical Reference, 1990.

[DMN12] Nico Döttling, Jörn Müller-Quade, and Anderson C. A. Nascimento. IND-CCA secure cryptography based on a variant of the LPN problem. In Advances in Cryptology - ASIACRYPT 2012, volume 7658 of LNCS, pages 485–503, Beijing, China, 2012. Springer.

[DP12] Ivan Damgård and Sunoo Park. Is public-key encryption based on LPN practical? IACR Cryptol. ePrint Arch., page 699, 2012.

[DP23a] Léo Ducas and Ludo N. Pulles. Accurate score prediction for dual attacks. preprint, November 2023. preprint.

[DP23b] Léo Ducas and Ludo N. Pulles. Does the dual-sieve attack on learning with errors even work? In Helena Handschuh and Anna Lysyanskaya, editors, Advances in Cryptology - CRYPTO 2023, volume 14083 of LNCS, pages 37–69, Santa Barbara, CA, USA, August 2023. Springer.

[DT17a] Thomas Debris-Alazard and Jean-Pierre Tillich. Statistical decoding. preprint, January 2017. arXiv:1701.07416.

[DT17b] Thomas Debris-Alazard and Jean-Pierre Tillich. Statistical decoding. In Proc. IEEE Int. Symposium Inf. Theory - ISIT 2017, pages 1798–1802, Aachen, Germany, June 2017.

[Dub10] Moshe Dubiner. Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem. IEEE Trans. Inform. Theory, 56(8):4166–4179, 2010.

[Duc18] Léo Ducas. Shortest vector from lattice sieving: A few dimensions for free. In Jesper Buus Nielsen and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2018, pages 125–145, Cham, 2018. Springer International Publishing.

[Duc22] Léo Ducas. Estimating the hidden overheads in the bdgl lattice sieving algorithm. In Post-Quantum Cryptography: 13th International Workshop, PQCrypto 2022, Virtual Event, September 28–30, 2022, Proceedings, page 480–497, Berlin, Heidelberg, 2022. Springer-Verlag.

[Dum86] Ilya Dumer. On syndrome decoding of linear codes. In Proceedings of the 9th All-Union Symp. on Redundancy in Information Systems, abstracts of papers (in russian), Part 2, pages 157–159, Leningrad, 1986.

[Dum89] Il'ya Dumer. Two decoding algorithms for linear codes. Probl. Inf. Transm., 25(1):17–23, 1989.

[Dum91] Ilya Dumer. On minimum distance decoding of linear codes. In Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory, pages 50–52, Moscow, 1991.

[DV13] Alexandre Duc and Serge Vaudenay. HELEN: A public-key cryptosystem based on the LPN and the decisional minimal distance problems. In Progress in Cryptology - AFRICACRYPT 2013, volume 7918 of LNCS, pages 107–126. Springer, 2013.

[EJK20] Thomas Espitau, Antoine Joux, and Natalia Kharchenko. On a dual/hybrid approach to small secret LWE - A dual/enumeration technique for learning with errors and application to security estimates of FHE schemes. In Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran, editors, Progress in Cryptology - INDOCRYPT 2020 - 21st International Conference on Cryptology in India, Bangalore, India, December 13-16, 2020, Proceedings, volume 12578 of Lecture Notes in Computer Science, pages 440–462. Springer, 2020.

[EKM17] Andre Esser, Robert Kübler, and Alexander May. LPN decoded. In Jonathan Katz and Hovav Shacham, editors, Advances in Cryptology - CRYPTO 2017, volume 10402 of LNCS, pages 486–514, Santa Barbara, CA, USA, August 2017. Springer.

[EKZ21] Andre Esser, Robert Kübler, and Floyd Zweydinger. A faster algorithm for finding closest pairs in Hamming metric. In Mikolaj Bojanczyk and Chandra Chekuri, editors, 41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference, volume 213 of LIPIcs, pages 20:1–20:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Ess23] Andre Esser. Revisiting nearest-neighbor-based information set decoding. In Elizabeth A. Quaglia, editor, Cryptography and Coding, volume 14421 of LNCS, pages 34–54. Springer, 2023.

[FA25] Hiroki Furue and Yusuke Aikawa. An improved both-may information set decoding algorithm: Towards more efficient time-memory trade-offs. In Ruben Niederhagen and Markku-Juhani O. Saarinen, editors, Post-Quantum Cryptography, pages 104–128, Cham, 2025. Springer Nature Switzerland.

[FJ05] M. Frigo and S.G. Johnson. The design and implementation of fftw3. Proceedings of the IEEE, 93(2):216–231, 2005.

[FKI07] Marc P. C. Fossorier, Kazukuni Kobara, and Hideki Imai. Modeling bit flipping decoding based on nonorthogonal check sums with application to iterative decoding attack of McEliece cryptosystem. IEEE Trans. Inform. Theory, 53(1):402–411, 2007.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A.M. Odlyzko, editor, Advances in Cryptology - CRYPTO '86, volume 263 of LNCS, pages 186–194. Springer, 1987.

[Gal63] Robert G. Gallager. Low Density Parity Check Codes. M.I.T. Press, Cambridge, Massachusetts, 1963.

[GJ21] Qian Guo and Thomas Johansson. Faster dual lattice attacks for solving LWE with applications to CRYSTALS. In Mehdi Tibouchi and Huaxiong Wang, editors, Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV, volume 13093 of Lecture Notes in Computer Science, pages 33–62. Springer, 2021.

[GJL14] Qian Guo, Thomas Johansson, and Carl Löndahl. Solving LPN using covering codes. In Advances in Cryptology - ASIACRYPT 2014, volume 8873 of LNCS, pages 1–20. Springer, 2014.

[GJL20] Qian Guo, Thomas Johansson, and Carl Löndahl. Solving lpn using covering codes. J. Cryptol., 33(1):1–33, January 2020.

[GJN24] Qian Guo, Thomas Johansson, and Vu Nguyen. A new sieving-style information-set decoding algorithm. IEEE Trans. Inform. Theory, 70(11):8303–8319, 2024.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Advances in Cryptology - CRYPTO 2015, volume 9215 of LNCS, pages 23–42, Santa Barbara, CA, USA, August 2015. Springer.

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. A key recovery attack on MDPC with CCA security using decoding errors. In Jung Hee Cheon and Tsuyoshi Takagi, editors, Advances in Cryptology - ASIACRYPT 2016, volume 10031 of LNCS, pages 789–815, 2016.

[GMO10] Daniel M Gordon, Victor S Miller, and Peter Ostapenko. Optimal hash functions for approximate matches on the n-cube. IEEE Trans. Inform. Theory, 56(3):984 – 991, March 2010.

[GMW21] Qian Guo, Erik Mårtensson, and Paul Stankovski Wagner. On the sample complexity of solving lwe using bkw-style algorithms. In 2021 IEEE International Symposium on Information Theory (ISIT), pages 2405–2410, 2021.

[Gol17] Oded Goldreich. Introduction to Property Testing. Cambridge University Press, 2017.

[Gre66] Richard R. Green. A serial orthogonal decoder. JPL Space Programs Summary, 37-39-IV:247–253, 1966.

[GRS08] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. : Increasing the security and efficiency of. In Nigel Smart, editor, Advances in Cryptology - EUROCRYPT 2008, pages 361–378, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[HB01] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, Advances in Cryptology - ASIACRYPT 2001, pages 52–66, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[HJ10] Nicholas Howgrave-Graham and Antoine Joux. New generic algorithms for hard knapsacks. In Henri Gilbert, editor, Advances in Cryptology - EUROCRYPT 2010, volume 6110 of LNCS. Sringer, 2010.

[HKL+12] Stephan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on Ring-LPN. In Anne Canteaut, editor, Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, M volume 7549 of LNCS, pages 346–365, Washington DC, United States, 2012. Springer.

[HP03] W. Cary Huffman and Vera Pless. Fundamentals of error-correcting codes. Cambridge University Press, Cambridge, 2003.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[IS98] Mourad E.H. Ismail and Plamen Simeonov. Strong asymptotics for Krawtchouk polynomials. Journal of Computational and Applied Mathematics, pages 121–144, 1998.

[Jab01] Abdulrahman Al Jabri. A statistical decoding algorithm for general linear block codes. In Bahram Honary, editor, Cryptography and coding. Proceedings of the 8th IMA International Conference, volume 2260 of LNCS, pages 1–8, Cirencester, UK, December 2001. Springer.

[Jaq24] Samuel Jaques. Memory adds no cost to lattice sieving for computers in 3 or more spatial dimensions. IACR Communications in Cryptology, 10 2024.

[JW05] Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In Victor Shoup, editor, Advances in Cryptology - CRYPTO 2005, pages 293–308, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, STOC '83, page 193–206, New York, NY, USA, 1983. Association for Computing Machinery.

[KF15] Paul Kirchner and Pierre-Alain Fouque. An improved bkw algorithm for lwe with applications to cryptography and lattices. In Rosario Gennaro and Matthew Robshaw, editors, Advances in Cryptology - CRYPTO 2015, pages 43–62, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[KKK18] Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. ACM Trans. Algorithms, 14(3), June 2018.

[KKKOC16] Matti Karppa, Petteri Kaski, Jukka Kohonen, and Padraig Ó Catháin. Explicit Correlation Amplifiers for Finding Outlier Correlations in Deterministic Subquadratic Time. In Piotr Sankowski and Christos Zaroliagis, editors, 24th Annual European Symposium on Algorithms (ESA 2016), volume 57 of Leibniz International Proceedings in Informatics (LIPIcs), pages 52:1–52:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[KL95] Gil Kalai and Nathan Linial. On the distance distribution of codes. IEEE Trans. Inform. Theory, 41(5):1467–1472, September 1995.

[KPC+11] Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In Kenneth G. Paterson, editor, Advances in Cryptology - EUROCRYPT 2011, pages 7–26, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[KS21] Naomi Kirshner and Alex Samorodnitsky. A moment ratio bound for polynomials and some extremal properties of krawchouk polynomials and hamming spheres. IEEE Trans. Inform. Theory, 67(6):3509–3541, 2021.

[KU10] Satish B. Korada and Rüdiger Urbanke. Polar codes are optimal for lossy source coding. IEEE Trans. Inform. Theory, 56(4):1751–1768, 2010.

[Laa15] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Advances in Cryptology - CRYPTO 2015, volume 9215 of LNCS, pages 3–22, Santa Barbara, CA, USA, August 2015. Springer.

[LB88] Pil J. Lee and Ernest F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In Advances in Cryptology - EUROCRYPT'88, volume 330 of LNCS, pages 275–280. Springer, 1988.

[LF06] Éric Levieil and Pierre-Alain Fouque. An improved LPN algorithm. In Proceedings of the 5th international conference on Security and Cryptography for Networks, volume 4116 of LNCS, pages 348–359. Springer, 2006.

[LM20] Nati Linial and Jonathan Mosheiff. On the weight distribution of random binary linear codes. Random Structures & Algorithms, 56(1):5–36, 2020.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Advances in Cryptology - EUROCRYPT2010, volume 6110 of LNCS, pages 1–23. Springer, 2010.

[LW21] Thijs Laarhoven and Michael Walter. Dual lattice attacks for closest vector problems (with preprocessing). In Topics in Cryptology – CT-RSA 2021: Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17–20, 2021, Proceedings, page 478–502, Berlin, Heidelberg, 2021. Springer-Verlag.

[MAB+22] Matzov, Daniel Apon, Daniel J. Bernstein, Carl Mitchell, Léo Ducas, Martin Albrecht, and Chris Peikert. Improved Dual Lattice Attack. `https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/Fm4cDfsx65s`, 2022.

[MAT22] MATZOV. Report on the Security of LWE: Improved Dual Lattice Attack, April 2022.

[McE78] Robert J. McEliece. A Public-Key System Based on Algebraic Coding Theory, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $O(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors,

Advances in Cryptology - ASIACRYPT 2011, volume 7073 of LNCS, pages 107–124. Springer, 2011.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In E. Oswald and M. Fischlin, editors, Advances in Cryptology - EUROCRYPT 2015, volume 9056 of LNCS, pages 203–228. Springer, 2015.

[MS86] Florence J. MacWilliams and Neil J. A. Sloane. The Theory of Error-Correcting Codes. North–Holland, Amsterdam, fifth edition, 1986.

[MT23] Charles Meyer-Hilfiger and Jean-Pierre Tillich. Rigorous foundations for dual attacks in coding theory. In Theory of Cryptography Conference, TCC 2023, volume 14372 of LNCS, pages 3–32. Springer Verlag, December 2023.

[MV] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem, pages 1468–1480.

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. Journal of Mathematical Cryptology, 2(2):181–207, 2008.

[OTD10] Ayoub Otmani, Jean-Pierre Tillich, and Léonard Dallot. Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. Special Issues of Mathematics in Computer Science, 3(2):129–140, January 2010.

[Ove06] Raphael Overbeck. Statistical decoding revisited. In Reihaneh Safavi-Naini Lynn Batten, editor, Information security and privacy : 11^{th} Australasian conference, ACISP 2006, volume 4058 of LNCS, pages 283–294. Springer, 2006.

[Pin08] M.A. Pinsky. Introduction to Fourier Analysis and Wavelets. Graduate studies in mathematics. American Mathematical Society, 2008.

[Pra62] Eugene Prange. The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory, 8(5):5–9, 1962.

[PS24] Amaury Pouly and Yixin Shen. Provable dual attacks on learning with errors. In Marc Joye and Gregor Leander, editors, Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI, volume 14656 of LNCS, pages 256–285. Springer, 2024.

[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, pages 84–93, 2005.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. Extended version of [Reg05], dated May 2009, 2009.

[Rog56] C. A. Rogers. The number of lattice points in a set. Proceedings of the London Mathematical Society, s3-6(2):305–320, 04 1956.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21(2):120–126, 1978.

[RU08] Tom Richardson and Ruediger Urbanke. Modern Coding Theory. Cambridge University Press, 2008.

[SAB+20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[Sam24] Alex Samorodnitsky. Weight distribution of random linear codes and krawtchouk polynomials. Random Structures & Algorithms, 65(2):261–274, 2024.

[Sav21] Valentin Savin. Non-binary polar codes for spread-spectrum modulations. In 2021 11th International Symposium on Topics in Coding (ISTC), pages 1–5, 2021.

[Sch03] Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, STACS 2003, pages 145–156, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[SE94] C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program., 66(2):181–199, September 1994.

[Sen11] Nicolas Sendrier. Decoding one out of many. In Post-Quantum Cryptography 2011, volume 7071 of LNCS, pages 51–67, 2011.

[Sha48] Claude E. Shannon. A mathematical theory of communication. Bell System Technical Journal, 27(3):379–423, 1948.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In S. Goldwasser, editor, FOCS, pages 124–134, 1994.

[Sie45] Carl Ludwig Siegel. A mean value theorem in geometry of numbers. Annals of Mathematics, 46(2):340–347, 1945.

[ŞTA09] Eren Şaşoğlu, Emre Telatar, and Erdal Arıkan. Polarization for arbitrary discrete memoryless channels. In Proc. IEEE Inf. Theory Workshop- ITW, pages 144–149, October 2009.

[Ste88] Jacques Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, Coding Theory and Applications, volume 388 of LNCS, pages 106–113. Springer, 1988.

[Ste93] Jacques Stern. A new identification scheme based on syndrome decoding. In D.R. Stinson, editor, Advances in Cryptology - CRYPTO'93, volume 773 of LNCS, pages 13–21. Springer, 1993.

[SW71] Elias M. Stein and Guido Weiss. Introduction to Fourier Analysis on Euclidean Spaces (PMS-32). Princeton University Press, 1971.

[TV15] Ido Tal and Alexander Vardy. List decoding of polar codes. IEEE Trans. Inform. Theory, 61(5):2213–2226, 2015.

[Val15] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. J. ACM, 62(2), May 2015.

[WL19] Zheng Wang and Cong Ling. Lattice gaussian sampling by markov chain monte carlo: Bounded distance decoding and trapdoor sampling. IEEE Transactions on Information Theory, 65(6):3630–3645, 2019.

[YS16] Yu Yu and John Steinberger. Pseudorandom functions in almost constant depth from low-noise lpn. In Marc Fischlin and Jean-Sébastien Coron, editors, Advances in Cryptology - EUROCRYPT 2016, pages 154–183, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[Zam14] Ram Zamir. Lattice Coding for Signals and Networks: A Structured Coding Approach to Quantization, Modulation, and Multiuser Information Theory. Cambridge University Press, USA, 2014.

[ZF96] Ram Zamir and Meir Feder. On lattice quantization noise. IEEE Trans. Inform. Theory, 42(4):1152–1159, 1996.

[ZJW16] Bin Zhang, Lin Jiao, and Mingsheng Wang. Faster algorithms for solving lpn. In Marc Fischlin and Jean-Sébastien Coron, editors, Advances in Cryptology - EUROCRYPT 2016, pages 168–195, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

Abstract

In this thesis, we design and analyze dual attacks for solving the decoding problem. In code-based cryptography dual attacks, a.k.a statistical decoding, were introduced by Al-Jabri in 2001. However, it turned out that his algorithm was not competitive against Information Set Decoders (ISD). Starting with the Prange algorithm in 1962, the ISD's have been the dominant family of decoders for the last 60 years and are used to parameterize code-based schemes.

Our main contribution is to dramatically improve dual attacks and show that they can beat the best ISD's for some parameter regimes. In particular, our best attack asymptotically significantly beats all previously known decoders for codes of constant rates smaller than 0.42 at the Gilbert-Varshamov distance. This result was obtained by revisiting Al-Jabri's statistical decoding algorithm and generalizing it using a splitting strategy to reduce decoding to a problem that is essentially the Learning Parity with Noise (LPN) problem. We then solve it using standard solvers. Part of our work also lies in the development of tools to analyze these attacks: we do not use the traditional independence assumptions that were used to analyze statistical decoding and which must be used with great care. Our tools are based on the Poisson summation formula and a model for the distribution of the weight enumerator of random linear codes. We base the analysis of our attacks on this model and verify it experimentally. We also devise a variant of our most advanced attack that has, up to polynomial factors, the same performance but which we can fully prove without using this model.

The second part of this work is dedicated to devising and analyzing dual attacks in lattice-based cryptography. Here, the problem that we target is the Learning With Errors (LWE) problem. In this case, dual attacks have been recently vastly improved, partly using a similar splitting strategy. In particular, two recent attacks, first by Guo & Johansson in 2021 and then by Matzov in 2022 claimed to reduce the security of the NIST standard Kyber (ML-KEM). However, Ducas & Pulles showed in 2023 that the key independence assumptions used in the analysis of those recent lattice-based attacks were flawed. This left open the question of how to analyze these attacks and whether they could really work as expected. We devise a slight variant of these recent dual attacks and provide new tools for analyzing them without these assumptions. In particular, we show that our attack comes dangerously close to Kyber's security claims and that we slightly beat those recent attacks. This settles the controversy over whether a dual-sieve attack can really work as expected.