Applications of Quantum Fourier Sampling and the Dihedral Hidden Subgroup Problem

Maxime Remaud

THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITÉ

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Maxime Remaud

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Applications of Quantum Fourier Sampling and the Dihedral Hidden Subgroup Problem

soutenue publiquement le 17 novembre 2023

devant le jury composé de :

María Naya-Plasencia	Inria de Paris	Présidente du jury
Omar Fawzi	Inria, ENS de Lyon	Rapporteur
Damien Stehlé	CryptoLab	Rapporteur
Alex Bredariol Grilo	LIP6	Examinateur
Mehdi Mhalla	LIG	Examinateur
Jean-Pierre Tillich	Inria de Paris	Directeur

Remerciements

Jean-Pierre, je ne saurais te remercier assez pour ces trois ans, qui ont été, tu en conviendras probablement, particulièrement atypiques. Je n'ai que de la gratitude à exprimer pour ton aide constante, pour ta patience, mais également pour m'avoir épargné bien des tracas à plusieurs reprises. Je suis d'autant plus reconnaissant pour tout ce que j'ai pu apprendre grâce à toi, scientifiquement ou non, et suis fier de t'avoir eu comme directeur de thèse.

Simon, je mesure ma chance d'avoir effectué mon stage de fin d'études, puis d'avoir poursuivi en doctorat, sous ton regard toujours bienveillant. Bien que mes travaux soient finalement restés assez éloignés de tes domaines d'expertise, tu m'as toujours apporté de précieux conseils. Travailler et discuter avec toi est un plaisir en plus d'être enrichissant.

Merci à Omar Fawzi et à Damien Stehlé pour avoir endossé le rôle de rapporteur, ainsi qu'à Alex Bredariol Grilo, Mehdi Mhalla et María Naya-Plasencia pour avoir accepté de faire partie de mon jury.

Je tiens également à remercier Thomas et André, que j'ai eu la chance d'avoir comme co-auteurs. Thomas, ton sens de la pédagogie et la passion que tu mets dans ton travail sont particulièrement admirables. André, nous avons principalement collaboré à distance, mais je peux témoigner de ton enthousiasme et de la qualité du travail que tu abats de manière terriblement efficace.

Je souhaite également exprimer ma reconnaissance envers Cyril. Vous m'avez accordé votre confiance pour réaliser ce doctorat, puis l'avez renouvelée pour la suite. J'espère être à la hauteur des tâches qui m'attendent !

Ces trois années ont été rendues d'autant plus agréables grâce à mes collègues à Atos d'une part et à l'INRIA d'autre part. Je ne pourrais citer toutes les personnes que j'ai croisées, avec qui j'ai eu le plaisir de travailler ou de discuter au cours d'un repas ou d'une pause café. Évoluer dans ces deux environnements différents mais complémentaires a été à la fois instructif et plaisant.

J'ai une pensée particulière pour les professeurs qui m'ont conduit sur la route des mathématiques, puis sur un chemin de plus en plus spécifique pour en arriver jusqu'ici. En particulier, j'aimerais mentionner François Arnault, avec qui j'ai eu la chance de

découvrir l'informatique quantique lors d'un stage entre mes deux années de Master, et Philippe Gaborit, pour avoir notamment contribué à rendre cette thèse possible.

Maxime, un grand merci pour avoir relu un des chapitres de cette thèse et m'avoir donné de très bons conseils, mais également pour nos conversations, toujours très intéressantes et agréables.

Il n'y a pas que des docteurs en informatique théorique que je souhaite remercier. J'ai une reconnaissance infinie envers Dr. Laigle-Donadey en particulier, mais également tous les autres spécialistes que j'ai été amené à rencontrer. Peut-être ai-je contribué à faire avancer la science en médecine durant ces trois ans, bien malgré moi ?

Anaïs, Caroline, Koray, merci pour les moments partagés ensemble. Même s'ils se font plus rares qu'en Licence ou en Master, ils sont inestimables et me rendent souvent nostalgique de ce qu'on a pu vivre ensemble à la Roche, à Limoges ou ailleurs.

Olivier, meda akpe na wò ŋuto. Wòe nye xõnye vevito eye melé wò de nye dzi ŋu kplikplikpli. Mawu nayra wò gede. Èno anyi nam le ɣeyiɣi nyui kple võwo me, èdo alom hede dzi fo nam. Akpe gede na wò, èle vevie nam ŋuto.

A mes parents, merci de m'avoir fait confiance, accompagné dans mes choix et encouragé à suivre ma route en m'en donnant toujours les moyens. A toute ma famille, ne vous inquiétez pas, malgré l'éloignement, la Vendée est toujours dans mon coeur (certains peuvent en témoigner, j'en parle souvent...), tout comme vous l'êtes.

Nana Yaa, aseda nnooso, nanso mennya asɛmfua foforo biara. Meda wo ase sɛ mowo ha bere nyinaa. Meda wo ase sɛ woboa me bere nyinaa. Meda wo ase sɛ woyɛ wo. Yɛrefa asetra mu anammon a emu yɛ den yiye mu, nanso yɛbɛbom ayɛ. ɛrenkyɛ, yebetumi anya asetra mu anigye sɛnea ɛfata. Onyankopon nhyira wo pii. Medo wo.

Enfin, merci à vous tous. Que vous le sachiez déjà ou non, ces trois années ont été intriquées avec de nombreuses épreuves médicales. Votre soutien ou ne serait-ce que votre bienveillance ou un sourire ont rendu ce doctorat plus facile qu'il n'aurait dû l'être dans les circonstances données. Merci.

Contents

Introduction (Français)

Problème de sous-groupe caché. Abrégé par HSP pour "Hidden Subgroup Problem", le problème de sous-groupe caché est un problème fondamental en informatique théorique consistant à trouver un sous-groupe inconnu H au sein d'un groupe G à l'aide uniquement d'une fonction qui est constante et distincte sur les classes de H. Ce problème de sous-groupe caché a de nombreuses applications en cryptographie, ce qui en fait un problème important à étudier, puisque la sécurité de certains systèmes cryptographiques repose notamment sur la difficulté à résoudre ce problème. Les algorithmes quantiques interviennent alors, car il a été montré que des avantages significatifs peuvent être obtenus pour résoudre efficacement certaines instances de HSP difficiles à résoudre pour les ordinateurs classiques. Une technique bien connue se cache derrière cela, c'est l'échantillonnage de Fourier quantique.

Échantillonnage de Fourier quantique. Quantum Fourier Sampling en anglais, l'échantillonnage de Fourier quantique est en effet une technique algorithmique quantique qui exploite la transformée de Fourier quantique (QFT, pour "Quantum Fourier Transform", qui est l'analogue quantique de la transformée de Fourier discrète: d'une certaine manière, elle mesure les fréquences présentes dans l'état d'entrée et les exprime dans l'état transformé). L'échantillonnage de Fourier quantique tire donc parti de cette propriété pour résoudre efficacement les problèmes qui peuvent être représentés à l'aide d'une fonction avec périodicité. La méthode est simple: il suffit tout d'abord de construire une superposition d'entrées, appliquer la fonction en question sur cette superposition, puis appliquer une QFT sur le registre contenant les images ainsi calculées et le mesurer. Nous obtenons alors une superposition d'éléments d'une classe de l'orthogonal du sous-groupe caché, nous permettant par conséquent d'obtenir de l'information sur ce dernier.

Chapitre 1. L'algorithme de Shor est un célèbre et excellent exemple de la manière dont l'échantillonnage de Fourier quantique peut être appliqué à un problème de sous-groupe caché. Il utilise en effet cette technique pour trouver la période d'une fonction modulaire, qui est un exemple concret de HSP. La factorisation de grands nombres (un problème difficile classiquement) pouvant être réduite à la recherche de la période d'une

fonction modulaire et cette fonction modulaire pouvant être calculée efficacement de manière quantique, l'algorithme de Shor résout le problème de factorisation efficacement (c'est-à-dire, en temps polynomial). En tirant parti de la puissance de l'échantillonnage de Fourier quantique, qui hérite lui-même des propriétés de la QFT, il a ainsi été montré qu'à l'image de l'algorithme de Shor, il est possible de résoudre des problèmes de sous-groupe caché ayant un impact important en cryptographie. Le Chapitre 1 introduit plus largement le problème de sous-groupe caché ainsi que la méthode d'échantillonnage de Fourier quantique, en allant de l'algorithme de Simon jusqu'à l'algorithme générique permettant de résoudre le HSP dans n'importe quel groupe abélien en temps polynomial, en passant par l'algorithme de Shor.

Chapitre 2. La cryptographie post-quantique est un domaine en pleine évolution qui cherche à développer des primitives cryptographiques sûres contre les attaques cryptanalytiques exécutées sur des ordinateurs quantiques. La sécurité de ces primitives post-quantiques est basée sur des problèmes mathématiques considérés comme difficiles pour les ordinateurs classiques et quantiques, tels que le problème du vecteur le plus court (SVP, pour "Shortest Vector Problem") et le problème de l'apprentissage avec erreurs (LWE, pour "Learning With Errors") en cryptographie à base de réseaux, qui est une approche populaire de la cryptographie post-quantique se basant sur la métrique euclidienne. Le Chapitre 2 donne les bases et notions élémentaires de cryptographie à base de codes, une autre approche majeure en cryptographie post-quantique se basant quant à elle sur la métrique de Hamming. Les équivalents de SVP et LWE, respectivement le problème de recherche de mot de petit poids et le problème de décodage, sont notamment clairement définis et leur difficulté est discutée. Enfin, nous introduisons et discutons également de la métrique rang, qui tend à être de plus en plus étudiée et investiguée dans le cadre de la recherche de primitives post-quantiques (quelques soumissions en métrique rang, en plus de celles plus nombreuses en métrique euclidienne et de Hamming, sont à noter au programme de standardisation du NIST, voir [NIS16; NIS22]).

Chapitre 3. Ce chapitre correspond à l'article [DRT23]. Le but de notre travail est de montrer comment calculer avec un ordinateur quantique un mot de faible poids de Hamming dans un code aléatoire à partir d'un algorithme permettant de décoder son dual. C'est la première fois qu'une telle réduction (classique ou quantique) pour la métrique de Hamming a été obtenue. En fait, ce travail fournit une adaptation aux codes linéaires de la réinterprétation de Stehlé-Steinfeld-Tanaka-Xagawa [Ste+09] de la réduction quantique de Regev [Reg05] de SVP approximé dans le pire cas au problème LWE. La métrique de Hamming est une métrique beaucoup plus grossière que la métrique euclidienne et cette adaptation a nécessité plusieurs nouveaux ingrédients pour fonctionner. Par exemple, pour obtenir une réduction significative, il est nécessaire, avec la métrique de Hamming, de choisir un rayon de décodage très grand: dans de nombreux

cas, il faut aller au-delà du rayon où le décodage est unique. Une autre étape cruciale pour l'analyse de la réduction est le choix des erreurs pour l'algorithme de décodage. En réseaux, les erreurs sont généralement échantillonnées selon une distribution gaussienne. Cependant, il s'avère que la distribution de Bernoulli (l'analogue de la gaussienne pour les codes) est trop étalée et ne peut pas être utilisée, en tant que telle, pour la réduction avec des codes. Nous avons traité cette difficulté pour obtenir le résultat susmentionné en considérant une distribution originale dans ce contexte, c'est-à-dire une distribution de Bernoulli tronquée. En outre, notre travail montre également une connexion entre l'approche de Regev et une notion intéressante de "distance duale" qui est impliquée dans la première borne de programmation linéaire en théorie des codes [McE+77] ou celle [Lev79; CE03] d'empilement des sphères dans $\mathbb{R}^n$.

Chapitre 4. Comme vu dans le premier chapitre, la méthode d'échantillonnage de Fourier quantique permet de résoudre le problème de sous-groupe caché dans n'importe quel groupe abélien en temps polynomial. Mais son intérêt ne s'arrête pas aux seuls groupes abéliens: il a également été montré qu'elle peut être utile pour résoudre ce problème dans un groupe diédral. Cette fois-ci elle ne permet pas une accélération exponentielle comme pour les groupes abéliens mais une accélération sous-exponentielle, tout de même. Dans ce chapitre nous reprenons les principaux algorithmes qui ont échelonné l'histoire de la résolution du problème de classe diédrale suivant le sous-groupe caché (DCP, pour "Dihedral Coset Problem"), problème quantique qui peut s'exprimer relativement simplement et auquel se réduit le problème de sous-groupe caché dans un groupe diédral. A celui-ci se réduit la sécurité de nombreux cryptosystèmes, qu'ils soient asymétriques ou symétriques, et plus généralement de nombreux autres problèmes, notamment utilisés en cryptographie à base de réseaux ou à base d'isogénies, faisant du DCP un problème de prime importance. Pour $N \in \mathbb{N}$, il est défini comme suit.

Problème (DCP). *Etant donné un oracle générant aléatoirement des nombres k tirés uniformément dans $\mathbb{Z}_N$ et leur état quantique associé $|\psi_k\rangle$, déterminer $s \in \mathbb{Z}_N$, avec:*

$$|\psi_k\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{\frac{2i\pi sk}{N}} |1\rangle \right).$$

Il existe deux familles d'algorithmes pour le résoudre, toutes deux tournant en temps sous-exponentiel: la première est essentiellement composée du premier algorithme de Kuperberg [Kup05], qui résout le problème de manière directe et uniquement grâce à des portes CNOT et des mesures, tandis que l'autre est notamment composée de l'algorithme de Regev [Reg04a] et du deuxième algorithme de Kuperberg [Kup13], qui réduisent le problème à un problème de somme de sous-ensembles (*subset-sum problem*). Ces trois algorithmes sont présentés dans ce chapitre et leur complexité est donnée, après que le problème de sous-groupe caché dans un groupe diédral et le DCP soient définis, ainsi que

l'algorithme de Ettinger et Høyer [EH99], le premier algorithme résolvant ce problème, présenté.

Chapitre 5. En combinant une idée due à Regev pour résoudre le DCP dans $\mathbb{Z}_N$ [Reg04a] et la méthode d'échantillonnage de Fourier quantique, nous introduisons dans ce chapitre un nouveau genre d'algorithme pour résoudre le problème de sous-groupe caché dans un groupe diédral. Nous réduisons en effet la résolution de ce problème à celle d'un problème *quantique* de somme de sous-ensembles, ce qui offre une nouvelle alternative aux deux autres principaux modes de résolutions déjà connus et cités ci-dessus. Pour cela, nous reprenons l'idée initiale de l'algorithme de Regev mais au lieu de mesurer une superposition de valeurs cibles pour un problème de somme de sous-ensembles et de résoudre le problème classiquement, nous conservons la superposition et utilisons un algorithme quantique pour résoudre le problème quantiquement. Nous obtenons alors un algorithme qui ne nécessite plus de faire qu'un nombre linéaire (c'est-à-dire en $O(\log N)$) de requêtes à l'oracle, là où la version économe en requêtes de celui de Regev nécessitait un nombre quadratique (en $O((\log N)^2)$). De plus, la complexité en temps est celle de l'algorithme quantique de somme de sous-ensembles au lieu de celui classique. Nous obtenons ainsi le premier algorithme depuis celui de Ettinger et Høyer à effectuer un nombre linéaire de requêtes à l'oracle, et nous en améliorons l'exposant: là où celui de Ettinger-Høyer tourne en temps en $O(N)$, le notre tourne approximativement en $O(N^{0.418})$. Enfin, nous donnons également une interpolation très naturelle sur le nombre de requêtes entre ce nouvel algorithme et le deuxième de Kuperberg, ainsi qu'une étude affinée d'algorithmes pour résoudre le problème de somme de sous-ensembles quantique. Ce chapitre correspond à l'article [RST23].

Chapitre 6. Finalement, nous donnons dans notre dernier chapitre plusieurs pistes pour résoudre le DCP dans $\mathbb{Z}_N$ sans avoir à se réduire à un problème de somme de sous-ensembles, nous plaçant ainsi dans la voie du premier algorithme de Kuperberg, ce dernier ayant néanmoins le défaut majeur de nécessiter un espace sous-exponentiel (classique et quantique) en $2^{O(\sqrt{\log N})}$ pour fonctionner. En nous inspirant d'une réécriture de celui-ci proposée par Bonnetain et Naya-Plasencia [BN18], nous présentons un nouvel algorithme pour résoudre le DCP dans $\mathbb{Z}_N$. Celui-ci bat tous les algorithmes existants en terme d'usage d'espace classique comme quantique puisqu'il nécessite *au maximum* $\lceil \log N \rceil$ qubits et sa complexité en temps est en $O(N^{0.415})$ et est la plus basse parmi tous les algorithmes utilisant un espace linéaire, mais en contrepartie, il nécessite de faire un nombre de requêtes exponentiel. De par son mode de fonctionnement, il est très facile de généraliser la méthode employée à l'élaboration de cet algorithme et de l'adapter pour un nombre quelconque de qubits.

Introduction (English)

Hidden Subgroup Problem. Abbreviated HSP, the hidden subgroup problem is a fundamental problem in theoretical computer science consisting in finding an unknown subgroup H within a group G using only a function that is constant and distinct over the cosets of H. This hidden subgroup problem has many applications in cryptography, making it an important problem to study, since the security of some cryptographic systems relies in particular on the difficulty of solving this problem. Quantum algorithms come into play here, as it has been shown that significant advantages can be gained in efficiently solving certain HSP instances that are difficult for classical computers to solve. A well-known technique behind this is quantum Fourier sampling.

Quantum Fourier sampling. Quantum Fourier Sampling is in fact a quantum algorithmic technique that exploits the quantum Fourier transform (QFT, which is the quantum analog of the discrete Fourier transform: in a way, it measures the frequencies present in the input state and expresses them in the transformed state). Quantum Fourier sampling therefore takes advantage of this property to efficiently solve problems that can be represented using a function with periodicity. The method is simple: first construct a superposition of inputs, apply the function in question to this superposition, then apply a QFT to the register containing the images thus calculated and measure it. We obtain a superposition of elements of a coset of the orthogonal of the hidden subgroup, enabling us to obtain information about the latter.

Chapter 1. Shor's algorithm is a famous and excellent example of how quantum Fourier sampling can be applied to a hidden subgroup problem. It uses this technique to find the period of a modular function, which is a concrete example of HSP. Since factorization of large numbers (a classically difficult problem) can be reduced to finding the period of a modular function, and since this modular function can be quantumly computed efficiently, Shor's algorithm solves the factorization problem efficiently (*i.e.*, in polynomial time). By taking advantage of the power of quantum Fourier sampling, which itself inherits the properties of QFT, it has thus been shown that, like Shor's algorithm, it is possible to solve hidden subgroup problems that have a major impact

in cryptography. Chapter 1 provides a broader introduction to the hidden subgroup problem and the quantum Fourier sampling method, ranging from Simon's algorithm via Shor's algorithm to the generic algorithm for solving the HSP in any abelian group in polynomial time.

Chapter 2. Post-quantum cryptography is an evolving field that seeks to develop cryptographic primitives secure against cryptanalytic attacks executed on quantum computers. The security of these post-quantum primitives is based on mathematical problems considered difficult for both classical and quantum computers, such as the Shortest Vector Problem (SVP) and the Learning With Errors Problem (LWE) in lattice-based cryptography, which is a popular approach to post-quantum cryptography based on the Euclidean metric. The Chapter 2 gives the basics and notions of code-based cryptography, another major approach to post-quantum cryptography based on the Hamming metric. In particular, the equivalents of SVP and LWE, respectively the short codeword search problem and the decoding problem, are defined and their difficulty discussed. Finally, we also introduce and discuss the rank metric, which tends to be increasingly studied and investigated in the search for post-quantum primitives (a few submissions in rank metric, in addition to the more numerous ones in Euclidean and Hamming metrics, are to be noted at the NIST standardization process, see [NIS16; NIS22]).

Chapter 3. This chapter corresponds to the article [DRT23]. The aim of our work is to show how to compute with a quantum computer a low-weight Hamming word in a random code from an algorithm allowing to decode its dual. This is the first time that such a reduction (classical or quantum) for the Hamming metric has been obtained. In fact, this work provides an adaptation to linear codes of Stehlé-Steinfeld-Tanaka-Xagawa's reinterpretation [Ste+09] of Regev's quantum reduction [Reg05] from worst-case approximate SVP to LWE. The Hamming metric is a much coarser metric than the Euclidean one, and this adaptation required several new ingredients to work. For example, to achieve a significant reduction, it is necessary, with the Hamming metric, to choose a very large decoding radius: in many cases, we have to go beyond the radius where decoding is unique. Another crucial step in reduction analysis is the choice of errors that are passed on to the decoding algorithm. With lattices, errors are generally sampled according to a Gaussian distribution. However, it turns out that the Bernoulli distribution (the Gaussian analogue for codes) is too spread out and cannot be used, as such, for a reduction with codes. We have addressed this difficulty to obtain the above result by considering an original distribution in this context, *i.e.*, a truncated Bernoulli distribution. In addition, our work also shows a connection between Regev's approach and an interesting notion of "dual distance" which is involved in the first linear

programming bound in code theory [McE+77] or that [Lev79; CE03] of sphere stacking in $\mathbb{R}^n$.

Chapter 4. As seen in the first chapter, the quantum Fourier sampling method can be used to solve the hidden subgroup problem in any abelian group in polynomial time. But the method's interest is not limited to abelian groups: it has also been shown that it can be useful for solving this problem in a dihedral group. This time, it does not allow exponential acceleration as for abelian groups, but it does allow subexponential acceleration. In this chapter, we review the main algorithms that have staggered the history of the solution of the Dihedral Coset Problem (DCP), a quantum problem that can be expressed relatively simply and to which the hidden subgroup problem in a dihedral group is reduced. In turn, to this problem is reduced the security of many cryptosystems, whether asymmetric or symmetric, and more generally of many other problems, notably used in lattice-based or isogeny-based cryptography, making the DCP a problem of prime importance. For $N \in \mathbb{N}$, it is defined as follows.

Problem (DCP). *Given an oracle generating numbers k drawn uniformly at random from $\mathbb{Z}_N$ and their associated quantum state $|\psi_k\rangle$, determine $s \in \mathbb{Z}_N$, where:*

$$|\psi_k\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{\frac{2i\pi sk}{N}} |1\rangle \right).$$

There are two families of algorithms for solving this problem, both running in subexponential time: the first is essentially made up of Kuperberg's first algorithm [Kup05], which solves the problem directly and solely using CNOT gates and measurements, while the other is notably made up of Regev's algorithm [Reg04a] and Kuperberg's second algorithm [Kup13], which reduce the problem to a classical *subset-sum problem*. These three algorithms are presented in this chapter and their complexity is given, after the hidden subgroup problem in a dihedral group and the DCP are defined, and Ettinger and Høyer's algorithm [EH99], the historically first algorithm proposed to solve this problem, presented.

Chapter 5. Combining an idea due to Regev for solving the DCP in $\mathbb{Z}_N$ [Reg04a] and the quantum Fourier sampling method, we introduce in this chapter a new kind of algorithm for solving the hidden subgroup problem in a dihedral group. In fact, we reduce the resolution of this problem to that of a quantum subset-sum problem, offering a new alternative to the two other main methods of resolution already known and cited above. To do this, we take up the original idea of Regev's algorithm, but instead of measuring a superposition of target values for a subset-sum problem and solving the problem classically, we retain the superposition and use a quantum algorithm to solve the problem quantum-wise. We then obtain an algorithm that only requires a linear

number (*i.e.*, in $O(\log N)$) of queries to the oracle, whereas the low-queries version of Regev's algorithm required a quadratic number (in $O((\log N)^2)$). The time complexity is that of the quantum subset-sum algorithm instead of the classical one. We thus obtain the first algorithm since Ettinger and Høyer's to perform a linear number of queries to the oracle, and we improve its exponent: where Ettinger-Høyer's runs in $O(N)$ time, ours runs in approximately $O(N^{0.418})$. Finally, we also give a very natural interpolation on the number of queries between this new algorithm and Kuperberg's second algorithm, as well as a refined study of algorithms for solving the quantum subset-sum problem. This chapter corresponds to the article [RST23].

Chapter 6. Finally, in our last chapter, we present several ways of solving the DCP in $\mathbb{Z}_N$ without having to reduce it to a subset-sum problem, thus following in the footsteps of Kuperberg's first algorithm, which nevertheless has the major drawback of requiring a subexponential space (classical and quantum) in $2^{O\left(\sqrt{\log N}\right)}$ to work. Inspired by a rewriting of the latter proposed by Bonnetain and Naya-Plasencia [BN18], we present a new algorithm for solving DCP in $\mathbb{Z}_N$. This beats all existing algorithms in terms of use of both classical and quantum space, since it requires *at most* $\lceil \log N \rceil$ qubits and its time complexity is in $O(N^{0.415})$ and is the lowest among all algorithms using linear space, but on the other hand, it requires making an exponential number of queries. Because of the way it works, it is very easy to generalize the method used to design this algorithm and adapt it to any number of qubits.

Preprints and Publications

[DRT23] Thomas Debris-Alazard, Maxime Remaud, and Jean-Pierre Tillich. "Quantum Reduction of Finding Short Code Vectors to the Decoding Problem". In: *IEEE transactions on Information Theory* (2023) (cit. on pp. vi, x, 23).

[RST23] Maxime Remaud, André Schrottenloher, and Jean-Pierre Tillich. "Time and Query Complexity Tradeoff for the Dihedral Coset Problem". In: *PQCrypto 2023*. LNCS. Springer, 2023 (cit. on pp. viii, xii, 83).

[RT23] Maxime Remaud and Jean-Pierre Tillich. "Linear Space Algorithm for the Dihedral Coset Problem". In: (2023). Preprint (cit. on p. 103).

Notation and Acronyms

General notation. Vectors are in *row notation* and they will be written with lowercase bold letters. Uppercase bold letters are used to denote matrices.

Logarithms. $\log(x)$ (or simply $\log x$) will denote the logarithm in base 2 of x.

Set of integers. For a and b integers with $a \leq b$, we denote by $[\![a, b]\!]$ the set of integers $\{a, a+1, \ldots, b\}$. We extend this notation when a and b are not integers to the set of integers in $[a, b]$.

Polynomial quantity. $\mathsf{poly}(n)$ will denote a quantity which is an $O(n^a)$ for some constant a.

Inner product. $\mathbf{x} \cdot \mathbf{y} \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i y_i$ for $\mathbf{x} = (x_i)_{i=1}^{n}$ and $\mathbf{y} = (y_i)_{i=1}^{n}$.

Shannon entropy. $h_q(x) \stackrel{\text{def}}{=} -(1-x)\log_q(1-x) - x\log_q\left(\frac{x}{q-1}\right)$ for $x \in (0,1)$.

Quantum Fourier Transform (QFT). Let the prime p be the characteristic of $\mathbb{F}_q$, with $q = p^s$. Let $\omega_p \stackrel{\text{def}}{=} e^{\frac{2i\pi}{p}}$. The QFT of $|\psi\rangle \stackrel{\text{def}}{=} \sum_{\mathbf{x}\in\mathbb{F}_q^n} \alpha_{\mathbf{x}} |\mathbf{x}\rangle$ is defined as

$$\widehat{|\psi\rangle} \stackrel{\text{def}}{=} \frac{1}{\sqrt{q^n}} \sum_{\mathbf{y}\in\mathbb{F}_q^n} \left(\sum_{\mathbf{x}\in\mathbb{F}_q^n} \alpha_{\mathbf{x}} \omega_p^{\mathrm{Tr}(\mathbf{x}\cdot\mathbf{y})} \right) |\mathbf{y}\rangle \qquad \text{where} \qquad \mathrm{Tr}(a) \stackrel{\text{def}}{=} \sum_{i=0}^{s-1} a^{p^i}.$$

Acronyms.

CBC	Code-Based Cryptography
DCP	Dihedral Coset Problem
DHSP	Dihedral Hidden Subgroup Problem
DLP	Discrete Logarithm Problem
DP	Decoding Problem
HSP	Hidden Subgroup Problem
LPN	Learning Parity with Noise
LWE	Learning With Errors
SCP	Short Codeword Problem
SIS	Short Integer Solution
SS	Subset-Sum
SVP	Shortest Vector Problem
uSVP	unique Shortest Vector Problem

Chapter 1
Introduction to the Hidden Subgroup Problem

Let G be a known group and H be an unknown subgroup of G. We will refer to H as the hidden subgroup of G and finding it is a problem known as the Hidden Subgroup Problem (HSP). This problem is of the greatest importance in mathematics and theoretical computer science, because it encompasses many known problems used in particular in cryptography to construct primitives. For example, we can mention the problems of period finding, discrete logarithm, graph isomorphism and shortest vector as special cases of hidden subgroup problems. Among all the cryptographic primitives built relying on the hardness of finding a hidden subgroup in a group, we can obviously cite the RSA cryptosystem [RSA78] and the Diffie-Hellman key exchange protocol [DH76], built on the period-finding problem and the discrete logarithm problem respectively. More interestingly for our purpose, we can mention CSIDH [Cas+18] and more generally any cryptosystem based on the isogeny problem [CJS14], any cryptosystem based on the Unique Shortest Vector Problem (uSVP) in lattice-based cryptography (such as [AD97; Reg04b]) or on any of the many problems that can be reduced to uSVP , and even symmetric primitives (Poly1305-AES [Ber05; BN18] for example).

But the HSP is also paramount in quantum computing because it was responsible for the first major advances in the field in the early 1990s. Indeed, Simon's and Shor's algorithms are well-known examples of quantum algorithms that solve hidden subgroup problems, and what makes them particularly interesting is the fact that they work in polynomial time, whereas no classical algorithm does.

In order to solve the HSP, we are given a function which is said to "hide" the hidden subgroup, since it has the property to be constant and distinct on the left cosets of H:

Definition 1.1 (Hiding function)**.** Let G be a group and H be a subgroup of G. We say that a function $f : G \to S$ (where S is a finite set) hides H if

$$\forall g, g' \in G, \quad f(g) = f(g') \iff gH = g'H.$$

We can now properly define the HSP, as follows.

Problem 1.1. *Hidden Subgroup Problem (HSP). The hidden subgroup problem is defined as:*

- *Input: a function $f : G \to S$ that hides an unknown subgroup H of a group G, S being a finite set,*
- *Output: (a generating set of) H.*

Note that in practice, H might be a large subset of G and finding it (*i.e.*, finding all its elements) would be tedious, but finding a generating set of it will be enough for our purpose.

In Table 1.1, we give examples of problems which can be casted as a HSP and the group in which this HSP has to be solved.

Problem	Group
Simon's	$\mathbb{Z}_2^n$
Period Finding	$\mathbb{Z}_N$
Discrete Logarithm	$\mathbb{Z}_N \times \mathbb{Z}_N$
Graph Isomorphism	S_N
Shortest Vector	D_N

Table 1.1: Examples of problems which can be casted as a HSP in the specified group. S_N is the symmetric group on N elements and D_N is the dihedral group, the group of the symmetries of a polygon with N sides, which will be the subject of the next chapter. See [KLM06] for more details.

It turns out that when G is abelian, there is a well-known polynomial-time quantum algorithm solving HSP, which, roughly speaking, consists in building a superposition over the elements of G, computing $f(G)$ in an ancillary register, applying a Quantum Fourier Transform (QFT) on the first register and finally sampling it. Simon's and Shor's algorithms are specific cases of this algorithm. On the other hand, when G is not abelian, there is no general algorithm for solving this problem but some results exist for particular non-abelian groups like the symmetric and the dihedral groups, to which are linked the Graph Isomorphism Problem and the Shortest Vector Problem, respectively.

In this chapter, we will focus on algorithms for solving the HSP in abelian groups. Namely, we will give an overview of Simon's algorithm, then of Shor's algorithm for the Discrete Logarithm Problem, and finally of the "standard algorithm" for solving any HSP in a finite abelian group in polynomial time. The HSP in a dihedral group will be the subject of Chapter 5 and Chapter 6 of this manuscript.

Contents

1 Simon's Algorithm

In the early 1990s, Simon was one of the first to present a quantum algorithm that could solve a certain problem using a polynomial number of queries where a classical algorithm would use an exponential number [Sim94]. The problem in question, which is simply called "Simon's problem", was designed specifically to show that there is a quantum advantage over classical algorithms, but it later turned out that it has practial applications in symmetric cryptography (see for example [Bon+19; CLS22]), making Simon's work all the more interesting. More precisely, Simon's problem is defined as follows.

Problem 1.2. *Simon's problem. Let $f\colon \mathbb{Z}_2^n \to S$ (where S is a finite set) be a function that hides $H = \{\mathbf{0}, \mathbf{s}\} \subseteq \mathbb{Z}_2^n$. Simon's problem asks to find $\mathbf{s}$ given f.*

The promise on f means that we have the following property:

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{Z}_2^n, \quad f(\mathbf{x}) = f(\mathbf{y}) \iff \mathbf{x} = \mathbf{y} \oplus \mathbf{h} \quad \text{where } \mathbf{h} \in H = \{\mathbf{0}, \mathbf{s}\}\,. \tag{1.1}$$

In this problem, finding H boils down to finding the secret vector $\mathbf{s}$.

It can be shown that any classical algorithm solving Simon's problem with probability at least $2/3$ will have a query complexity in $\Omega\left(2^{\frac{n}{3}}\right)$ (see [KLM06], Theorem 6.5.1). On the other hand, Simon came up with a quantum algorithm which solves this same problem with a query complexity in $O(n)$, exponentially improving over its classical counterparts (see [KLM06], Theorem 6.5.2).

Algorithm 1 provides a pseudo-code implementation of Simon's algorithm. It assumes that the function f that hides the subgroup is efficiently implementable in a quantum way, which will always be assumed for all algorithms presented in this chapter. Namely, we assume that we have a unitary U_f for implementing f:

$$U_f\colon |\mathbf{x}\rangle\,|\mathbf{b}\rangle \mapsto |\mathbf{x}\rangle\,|\mathbf{b} \oplus f(\mathbf{x})\rangle\,.$$

Remarks on Simon's algorithm.

- The uniform superposition over $\mathbb{Z}_2^n$ is efficiently produced thanks to Hadamard gates applied to n qubits initialized to 0.
- The first measurement is not mandatory but it simplifies the algorithm's analysis. By this measure, an image of f is chosen and the first register is projected on a

Algorithm 1 Simon's algorithm for Simon's Problem

Require: A unitary U_f to quantumly compute f.
Ensure: The secret vector $\mathbf{s}$ as defined in Equation 1.1.

1: Initialize a list $L = \emptyset$.
2: **repeat**
3: Prepare a uniform superposition over $\mathbb{Z}_2^n$ and use U_f to compute f in an ancillary register

$$\frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in \mathbb{Z}_2^n} |\mathbf{x}\rangle |f(\mathbf{x})\rangle$$

4: Measure the second register and discard it

$$\frac{1}{\sqrt{2}} (|\mathbf{y}\rangle + |\mathbf{y} + \mathbf{s}\rangle)$$

5: Apply $H^{\otimes n}$

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{\mathbf{x} \in Z_N} (-1)^{\mathbf{x} \cdot \mathbf{y}} (1 + (-1)^{\mathbf{x} \cdot \mathbf{s}}) |\mathbf{x}\rangle$$

6: Measure the register and place the resulting vector in L
7: **until** the dimension of the span of the vectors in L equals $n - 1$.
8: Solve the linear system formed by $\ell \cdot \mathbf{s} = 0, \quad \forall \ell \in L$ and output the non-zero solution.

superposition of the preimages by f of the image that has been measured. With our notation, an image $f(\mathbf{y})$ has two antecedents: $\mathbf{y}$ and $\mathbf{y} + \mathbf{s}$.

- If we look at the amplitude of the vector $\mathbf{x}$ just before the second measurement operator, we can see that it is zero if and only if $\mathbf{x} \cdot \mathbf{s} = 1$ or conversely that it is non-zero if and only if $\mathbf{x} \cdot \mathbf{s} = 0$. It means that we can write

$$\begin{aligned} \frac{1}{\sqrt{2^{n+1}}} \sum_{\mathbf{x} \in Z_N} (-1)^{\mathbf{x} \cdot \mathbf{y}} (1 + (-1)^{\mathbf{x} \cdot \mathbf{s}}) |\mathbf{x}\rangle &= \frac{1}{\sqrt{2^{n-1}}} \sum_{\substack{\mathbf{x} \in Z_N : \\ \mathbf{x} \cdot \mathbf{s} = 0}} (-1)^{\mathbf{x} \cdot \mathbf{y}} |\mathbf{x}\rangle \\ &= \frac{1}{\sqrt{2^{n-1}}} \sum_{\mathbf{x} \in H^\perp} (-1)^{\mathbf{x} \cdot \mathbf{y}} |\mathbf{x}\rangle \end{aligned}$$

  where we recall that

$$H^\perp \stackrel{\text{def}}{=} \{\mathbf{x} \in \mathbb{Z}_2^n : \mathbf{x} \cdot \mathbf{h} = 0 \quad \forall h \in H\} = \{\mathbf{x} \in \mathbb{Z}_2^n : \mathbf{x} \cdot \mathbf{s} = 0\}.$$

  The vector that will be measured will thus be a vector orthogonal to the secret one we are looking for.

- The dimension of H being 1, the dimension of $H^\perp$ has to be $n - 1$. Thus, if we have $n - 1$ vectors orthogonal to $\mathbf{s}$ forming a basis of $H^\perp$, we will be able to solve

the linear system in polynomial time in n by Gaussian elimination in the last step of the algorithm. This linear system has exactly two solutions, $\mathbf{0}$ and $\mathbf{s}$.

2 Shor's Algorithm

In this section, we will outline Shor's quantum algorithm for solving the Discrete Logarithm Problem (DLP) in polynomial time [Sho94]. Shor's algorithm is surely best known for allowing to efficiently solve the factorization problem by reducing it to the period finding problem that it then solves thanks to a quantum computer, but this will not be our point here. We will focus on the algorithm for DLP which is more straightforward and can easily be seen to be, in some way, on the path between Simon's algorithm and the standard algorithm for solving HSP for any finite abelian group. We start by recalling DLP:

Problem 1.3. *Discrete Logarithm Problem (*DLP*). Let $G = \langle g \rangle$ be a finite cyclic group of order N and $x \in G$. The* $\mathsf{DLP}_{g,x,N}$ *asks to find the smallest positive integer α such that*

$$g^\alpha = x, \tag{1.2}$$

i.e., $\alpha = \log_g x$.

With these notations, N will typically be equal to $p-1$ where p is a prime number and G will be $\mathbb{Z}_p^\times$. We assume that x is different from g, otherwise the problem is trivial. We also assume that we know the order N of the group since if we do not, we can simply use Shor's algorithm for period finding to determine N.

There have actually been many classical algorithms introduced for solving the DLP in an arbitrary group. Some are deterministic, as for examples the Baby-Step/Giant-Step algorithm [Sha73] or Pohlig-Hellman algorithm [PH78], and some are probabilistic, such as Pollard-ρ algorithm [Pol78] or the distinguished points method [vW99]. In any case, all these algorithms have a query complexity which is exponential in $\log N$. For some non-arbitrary groups, some improvements have been exhibited such as Adleman's method, for the case where the group in which we intend to solve the DLP is a finite field [Adl79]. The query complexity of all such specific algorithms is not better than subexponential but in [Bar+14], a heuristic algorithm providing a quasi-polynomial complexity when N has a specific form was introduced.

When Shor came up with his quantum algorithm, the impact was great because he showed that it is possible to efficiently solve the DLP with a query complexity polynomial in $\log N$, the size of the considered group. We first show how he first reduced the DLP to a HSP in $\mathbb{Z}_N \times \mathbb{Z}_N$ and then give his algorithm for solving this latter problem right after. For more details on classical algorithms for the DLP, we refer to [Bar13].

So indeed, the DLP can be written as a Hidden Subgroup Problem. An appropriate definition of f to fall back on a HSP is the following:

$$\begin{aligned} f\colon \mathbb{Z}_N \times \mathbb{Z}_N &\to G \\ (\alpha, \beta) &\mapsto x^\alpha g^\beta \end{aligned}$$

and this function f hides a subgroup H of $\mathbb{Z}_N \times \mathbb{Z}_N$ defined by

$$H \stackrel{\text{def}}{=} \{(\alpha, -\alpha \log_g x)\colon \alpha \in \mathbb{Z}_N\}$$

and its cosets

$$(0,\delta) + H = \{(\alpha, \delta - \alpha \log_g x)\colon \alpha \in \mathbb{Z}_N\}, \qquad \delta \in \mathbb{Z}_N$$

since we have that every elements in a coset $(0,\delta) + H$ for $\delta \in \mathbb{Z}_N$ has the same image by f and this image is different for each coset:

$$\forall \alpha, \delta \in \mathbb{Z}_N, \quad f(\alpha, \delta - \alpha \log_g x) = x^\alpha g^{-\alpha \log_g x} g^\delta = g^\delta.$$

Shor's algorithm is actually nothing more than Simon's algorithm adaptated from $\mathbb{Z}_2^n$ to $\mathbb{Z}_N \times \mathbb{Z}_N$ in the case of solving the DLP (and to $\mathbb{Z}_N$ in the case of solving the period finding problem). The Walsh-Hadamard transforms $H^{\otimes n}$ are simply replaced by QFT's on $\mathbb{Z}_N \times \mathbb{Z}_N$ ($\mathbb{Z}_N$, respectively). Similarily, it assumes that the hiding function f is efficiently implementable in a quantum way, which is true since we know how to efficiently do an exponentiation with a quantum circuit (we will refer here to [VBE96] but several papers improving the adder circuit which is used as a subroutine in the exponentiation algorithm have followed). Namely, we assume that we have a unitary U_f for implementing f:

$$U_f\colon \left|\mathbf{x}\right\rangle \left|\mathbf{b}\right\rangle \mapsto \left|\mathbf{x}\right\rangle \left|\mathbf{b} \oplus f(\mathbf{x})\right\rangle.$$

Hereafter, Algorithm 2 provides a pseudo-code implementation of Shor's algorithm for the DLP.

Remarks on Shor's algorithm.

- The QFT over $\mathbb{Z}_N$ can efficiently be approximated (*e.g.*, [HH00]) or even replaced by a QFT over $\mathbb{Z}_{2^n}$ where $n \stackrel{\text{def}}{=} \lceil \log N \rceil$, for example. Note that it can also be useful to decompose N by the fundamental theorem of arithmetic and thus uniquely write $N = \prod_{i=1}^m p_i^{n_i}$ where the p_i's are distinct prime numbers and the n_i's are integers. The QFT over $\mathbb{Z}_N$ is then equivalent to the tensor product of the QFT over the cyclic group $\mathbb{Z}_{p_i^{n_i}}$ for i going from 1 to m.

- By the first measurement, an image of f is chosen, let say g^δ, and the first register is projected on a superposition of the preimages (σ, τ) by f of this g^δ. Since we have $g^\delta \stackrel{\text{def}}{=} x^\sigma g^\tau = g^{\tau + \sigma \log_g x}$, we have $\tau = \delta - \sigma \log_g x$. Thus, we get a superposition of the couples $(\sigma, \delta - \sigma \log_g x)$ for $\sigma \in \mathbb{Z}_N$ where δ is unknown.

Algorithm 2 Shor's algorithm for the DLP

Require: A unitary U_f to quantumly compute f.
Ensure: The discrete logarithm $\log_g x$ as defined in Equation 1.2.

1: Prepare a uniform superposition over $\mathbb{Z}_N \times \mathbb{Z}_N$ and use U_f to compute f

$$\frac{1}{N} \sum_{\alpha,\beta \in \mathbb{Z}_N} |\alpha, \beta, f(\alpha, \beta)\rangle = \frac{1}{N} \sum_{\alpha,\beta \in \mathbb{Z}_N} \left|\alpha, \beta, x^\alpha g^\beta\right\rangle$$

2: Measure the last register and discard it

$$\frac{1}{\sqrt{N}} \sum_{\alpha \in \mathbb{Z}_N} \left|\alpha, \delta - \alpha \log_g x\right\rangle$$

3: Apply the Fourier transform over $\mathbb{Z}_N \times \mathbb{Z}_N$

$$\frac{1}{N\sqrt{N}} \sum_{\alpha,\rho,\sigma \in Z_N} \omega_N^{\rho\alpha + \sigma(\delta - \alpha \log_g x)} |\rho, \sigma\rangle$$

4: Measure the two registers to obtain a couple of values $(\sigma \log_g x, \sigma)$ for some $\sigma \in \mathbb{Z}_N$
5: Compute $\sigma^{-1} \mod N$, multiply it to $\sigma \log_g x$ and output the result.

- If we look at the amplitude of the couple (α, β) just before the second measurement operator, we have

$$\begin{aligned}
&\frac{1}{N\sqrt{N}} \sum_{\sigma,\alpha,\beta \in Z_N} \omega_N^{\alpha\sigma + \beta(\delta - \sigma \log_g x)} |\alpha, \beta\rangle \\
=&\frac{1}{N\sqrt{N}} \sum_{\alpha,\beta \in Z_N} \omega_N^{\beta\delta} \left(\sum_{\sigma \in \mathbb{Z}_N} \omega_N^{\sigma(\alpha - \beta \log_g x)} \right) |\alpha, \beta\rangle \\
=&\frac{1}{\sqrt{N}} \sum_{\beta \in Z_N} \omega_N^{\beta\delta} \left|\beta \log_g x, \beta\right\rangle
\end{aligned}$$

  where we used the following fact:

$$\sum_{\sigma \in \mathbb{Z}_N} \omega_N^{\sigma(\alpha - \beta \log_g x)} = \begin{cases} N & \text{if } \alpha = \beta \log_g x \\ 0 & \text{otherwise.} \end{cases}$$

  So as stated in the penultimate step of the algorithm, we will obtain as a result of the measurement step a couple $(\beta \log_g x, \beta)$ for some $\beta \in \mathbb{Z}_N$.

- The value β we obtain has an inverse modulo N with good probability, since β and N are coprime with probability $\frac{\phi(N)}{N} = \Omega\left(\frac{1}{\log \log N}\right)$ where ϕ is Euler's totient function, which allows us to retrieve $\log_g x$ by repeating around $\log \log N$ times the algorithm.

3 Standard Algorithm

It turns out that Simon's algorithm as well as Shor's algorithm, which respectively solve the hidden subgroup problem in $\mathbb{Z}_2^n$ and in $\mathbb{Z}_N$ or $\mathbb{Z}_N \times \mathbb{Z}_N$, can be seen as special cases of a more general algorithm. In the literature, it is often referred to as the *standard algorithm* and is mostly attributed to Kitaev [Kit95]. The method behind this algorithm is called "coset sampling". It consists in the construction of a superposition of the elements of a coset of the hidden subgroup using the function that hides it, followed by the application of a quantum Fourier transform, which produces a superposition on the elements of the dual of the hidden subgroup. Thus, a measurement of this superposition will give an element of the latter, and thus of the hidden subgroup itself. This algorithm turns out to efficiently solve the hidden subgroup problem in any abelian group G. Indeed, we have the following theorem.

Theorem 1.1 (Th. 2.2 [EH99])**.** *Let $f\colon G \to S$ be a function that fulfills the subgroup promise with respect to a subgroup H. There exists a quantum algorithm that outputs a subset $X \subset G$ such that X generates H with probability at least $1 - 1/|G|$. The algorithm uses $O(\log|G|)$ evaluations of f and runs in time polynomial in $\log|G|$ and in the time required to compute f.*

A brief aside is made on the theory of characters (see next page), which will be useful for understanding and analyzing Algorithm 3, the algorithm in question for solving HSP, as well as others in the chapters to come.

Algorithm 3 Standard algorithm for abelian groups

1: Prepare the uniform superposition over $G \times S$ and compute f

$$\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle\,|f(x)\rangle$$

2: Measure the second register and discard it

$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |y + h\rangle$$

3: Apply the Fourier transform over G

$$\frac{1}{\sqrt{|G||H|}} \sum_{h \in H} \sum_{g \in G} \chi_{y+h}(g)\,|g\rangle$$

4: Measure the register to get a random element from $H^{\perp}$.

A word on character theory. Before giving some information which will be necessary to us on character theory, we must begin with representation theory. Essentially, it provides an important tool called "representation" to study abstract algebraic structures, by representing the elements of these structures by linear transformations of vector spaces. A representation is more precisely defined as follows.

Definition 1.2. Let G be a finite group, V be a vector space over a field F and $GL(V)$ be the general linear group on V. A representation of G is a group homomorphism from G to $GL(V)$.

In particular, the elements of a multiplicative group G will be represented by complex invertible matrices and thus we will be able to use linear algebra tools, which is particularly interesting. For more information on the theory of representations, we refer the reader to [Sch21].
We now come to the character theory, which, in short, provides us with another tool called "character" that allows us to characterize the elements of a group by the trace of the matrix that represents them. The information given by the representation is not lost, it is simply compressed. Formally, a character is defined as follows.

Definition 1.3. Let G be a finite group, V be a vector space over a field F and $GL(V)$ be the general linear group on V. The character of a representation $\rho\colon G \to GL(V)$ is

$$\begin{aligned} \chi_\rho(g)\colon G &\to F \\ g &\mapsto \mathrm{Tr}(\rho(g)) \end{aligned}$$

From this new notion, we can directly define the dual group $G^\perp$ of G as the set of characters of G equipped with pointwise multiplication. In the same way, the dual $H^\perp$ of a subgroup H of G can be defined as follows.

$$H^\perp \stackrel{\text{def}}{=} \left\{\chi_g \in G^\perp\colon \quad \chi_g(h) = 1 \quad \forall h \in H\right\}.$$

To finish with this brief aside on characters, we give two properties that will be valuable to us later, on several occasions:

$$|x\rangle \xmapsto{\mathsf{QFT}\text{ over } G} \frac{1}{\sqrt{|G|}} \sum_{g\in G} \chi_x(g)\,|g\rangle \tag{1.3}$$

$$\sum_{h\in H} \chi_g(h) = \begin{cases} |H| & \text{if } \chi_g \in H^\perp \\ 0 & \text{otherwise.} \end{cases} \tag{1.4}$$

Once again we refer to [Sch21] and to [Wol19] for more details on character theory and on the link with the quantum Fourier transform, respectively.

Remarks on the standard algorithm.

- The first remark to be made concerns the QFT. How can it be implemented for any finite abelian group G? One solution is to use the fundamental theorem of finite abelian groups. Indeed, it says that G is isomorphic to the direct sum of cyclic subgroups of order a power of a prime number. Thus we can write that $G \simeq \mathbb{Z}_{p_1^{n_1}} \times \cdots \times \mathbb{Z}_{p_m^{n_m}}$ where the p_i's are distinct prime numbers and the n_i's are integers. Therefore, the QFT over G is the tensor product of the QFT over the cyclic group $\mathbb{Z}_{p_i^{n_i}}$ for i going from 1 to m.

- By the first measurement, an image $f(y)$ is chosen and the first register collapses on a superposition of the preimages of $f(y)$, i.e. of the elements of the coset of y, $y + H$.

- If we look at the amplitude of an element g just before the second measurement operator, we have

$$\begin{aligned}
&\frac{1}{\sqrt{|G||H|}} \sum_{h\in H} \sum_{g\in G} \chi_{y+h}(g) \,|g\rangle \\
=&\frac{1}{\sqrt{|G||H|}} \sum_{g\in G} \chi_y(g) \left(\sum_{h\in H} \chi_h(g) \right) |g\rangle \\
=&\frac{1}{\sqrt{|G||H|}} \sum_{g\in G} \chi_y(g) \left(\sum_{h\in H} \chi_g(h) \right) |g\rangle \\
=&\sqrt{\frac{|H|}{|G|}} \sum_{\substack{g\in G: \\ \chi_g \in H^\perp}} \chi_y(g) \,|g\rangle \qquad \text{(by Equation 1.4)}
\end{aligned}$$

  So, as stated in the last step of the algorithm, we will obtain as a result of the measurement step an element g such that $\chi_g \in H^\perp$.

Chapter 2
Introduction to Code-Based Cryptography

The theory of error-detecting and error-correcting codes makes it possible to ensure the integrity of information when exchanged during a communication over a channel where errors can occur, or of data stored on a physical medium, such as a CD or a hard disk, where errors can also occur. The principle is simple: redundancy is added to the transmitted or stored information. Any alteration of this information can thus be detected and even corrected, depending on the amount of redundancy.

This step during which one takes a k bits message (which contains the information to be transmitted or stored) and one adds redundancy to it to form a n bits *coded message* (n being necessarily greater than k), is called *encoding*. Encoding corresponds in fact to the multiplication of a vector $\mathbf{m}$ (representing the message) by a matrix $\mathbf{G}$ of size $k \times n$. The vector space given by the row space of this matrix is called a *code*. The recipient of the coded message $\mathbf{mG}$ will in fact recover a noisy version due to the alterations generated by the means of communication or storage, i.e. a vector $\mathbf{y} = \mathbf{mG} + \mathbf{e}$ of n bits where $\mathbf{e}$ is a vector representing the perturbations. Finding the original message $\mathbf{m}$ from $\mathbf{y}$ is then a task called *decoding*. This operation will be more or less complicated, depending on the vector $\mathbf{e}$ and on the encoding method used (whether more or less redundancy was added or whether a structure exists in the encoding method).

In the framework described so far, the objective is to make the decoding as simple as possible, preferably of polynomial complexity in n, in order to preserve the integrity of the transmitted information and especially to recover it easily. This is notably made possible by using codes with a strong structure, i.e. a structured matrix $\mathbf{G}$. But error-correcting codes have found a second important use in cryptography, since it turns out that when $\mathbf{G}$ is drawn randomly, the decoding task has an exponential cost in n. It is in 1978 that McEliece proposed the first cryptosystem based on the theory of codes, opening the way to code-based cryptography. Widely studied since then, no generic algorithm with polynomial complexity is known, neither classical nor quantum, to attack cryptographic primitives built from codes, making it one of the most popular branches of post-quantum cryptography.

For an extensive introduction to error-correcting codes, we refer the reader to the well-

known book by Huffman and Pless [HP03], which provides an extensive introduction to error correcting codes. We also mention other interesting resources for the reader interested in codes and which were good supports for writing this chapter. First, a chapter book written by Raphael Overbeck and Nicolas Sendrier [OS09] and some lecture notes by Alain Couvreur [Cou20] and Thomas Debris-Alazard [Deb21a; Deb21b]. There are also several PhD theses: by Nicolas Aragon [Ara20], Maxime Bros [Bro22], Thomas Debris-Alazard [Deb19], Adrien Hauteville [Hau17], Matthieu Lequesne [Leq21] and Rocco Mora [Mor23]. Finally, we can cite the French accreditation to supervise research of Pierre Loidreau [Loi07].

Contents

1 Definitions

We define here some basic key terms of code theory in our context. The notation $\mathbb{F}$ will stand for a finite field in this chapter. Recall that a finite field contains q^m elements, where q is a prime number and m a positive integer.

1.1 Codes and their representations

We start by formally defining what a code is: a code of length n is simply a subset of $\mathbb{F}^n$. Not convenient to use since it requires an exhaustive list of their elements, it is common to restrict ourselves to the case of linear codes (*i.e.*, subspaces of $\mathbb{F}^n$), which will be the kind of codes we will work on exclusively.

Definition 2.1 (Linear code $\mathcal{C}$)**.** Let n and k be integers with $k \leq n$. A linear code of *length* n and *dimension* k over $\mathbb{F}$ is a k-dimensional vector space of $\mathbb{F}^n$ and its elements are called *codewords*. We will refer to it as an $[n, k]_{|\mathbb{F}|}$-code and it will commonly be denoted by $\mathcal{C}$.

For the sake of clarity, we will simply refer to linear codes as codes in what follows.

An important quantity to look at when considering a code is its rate. It quantifies the proportion of information contained in a coded message.

Definition 2.2 (Code rate R)**.** Let $\mathcal{C}$ be an $[n,k]_{|\mathbb{F}|}$-code. The quantity $R \stackrel{\text{def}}{=} \frac{k}{n}$ is referred to as the *rate* of a code.

A code is intimately linked to its dual, defined as follows.

Definition 2.3 (Dual code $\mathcal{C}^\perp$)**.** Let $\mathcal{C}$ be an $[n,k]_{|\mathbb{F}|}$-code. The *dual code* $\mathcal{C}^\perp$ of $\mathcal{C}$ in $\mathbb{F}^n$ is a code of dimension $n-k$, defined as

$$\mathcal{C}^\perp \stackrel{\text{def}}{=} \{\mathbf{u} \in \mathbb{F}^n \colon \mathbf{u} \cdot \mathbf{v} = 0\}\,.$$

The dual code $\mathcal{C}^\perp$ of a linear code $\mathcal{C}$ over $\mathbb{F}$ can also be defined from the characters as follows:

$$\mathcal{C}^\perp \stackrel{\text{def}}{=} \{\mathbf{u} \in \mathbb{F}^n \colon \forall \mathbf{c} \in \mathcal{C},\ \chi_{\mathbf{u}}(\mathbf{c}) = 1\}$$

As briefly described in the introduction, a code can be represented by a matrix, called a generator matrix:

Definition 2.4 (Generator matrix $\mathbf{G}$)**.** Let $\mathcal{C}$ be an $[n,k]_{|\mathbb{F}|}$-code. A matrix $\mathbf{G} \in \mathbb{F}^{k\times n}$ such that the rows of $\mathbf{G}$ form a basis for $\mathcal{C}$ is a *generator matrix* for $\mathcal{C}$.

In other words, any codeword can be expressed as a linear combination of the rows of a generator matrix of the code (note that a code can be represented by more than one generator matrix); and a linear code can be defined as the k-dimensional vector space $\mathcal{C}$ spanned by the rows of one of its generator matrices $\mathbf{G}$:

$$\mathcal{C} \stackrel{\text{def}}{=} \left\{\mathbf{uG} \colon \mathbf{u} \in \mathbb{F}^k\right\}. \tag{2.1}$$

The generator matrix of the dual code $\mathcal{C}^\perp$ of a code $\mathcal{C}$ is called a parity-check matrix of $\mathcal{C}$.

Definition 2.5 (Parity-check matrix $\mathbf{H}$)**.** Let $\mathcal{C}$ be an $[n,k]_{|\mathbb{F}|}$-code. A matrix $\mathbf{H} \in \mathbb{F}^{(n-k)\times n}$ such that the rows of $\mathbf{H}$ form a basis for $\mathcal{C}^\perp$ is a *parity-check matrix* for $\mathcal{C}$.

In other words, any vector that is orthogonal to all the codewords of $\mathcal{C}$ is a linear combination of the rows of a parity-check matrix of the code (note that a code can be represented by more than one parity-check matrix) and a linear code $\mathcal{C}$ can be defined as the kernel of one of its parity-check matrices $\mathbf{H}$:

$$\mathcal{C} \stackrel{\text{def}}{=} \left\{\mathbf{c} \in \mathbb{F}^n \colon \mathbf{Hc}^\intercal = \mathbf{0}\right\}. \tag{2.2}$$

We have briefly mentioned that the encoding of a message $\mathbf{m}$ is done by computing $\mathbf{mG}$ where $\mathbf{G}$ is a generating matrix of the code. Decoding can be done in a similar way by multiplying the vector $\mathbf{mG}$ by the transpose of a parity check matrix, since $\mathbf{GH}^\intercal = \mathbf{0}$ by definition. Applying this decoding method to any vector of the field in which we work gives us what is called a syndrome.

Definition 2.6 (Syndrome)**.** Let $\mathbf{H} \in \mathbb{F}^{(n-k)\times n}$. The *syndrome* of a vector $\mathbf{y} \in \mathbb{F}^n$ with respect to $\mathbf{H}$ is defined as the vector $\mathbf{y}\mathbf{H}^\intercal$.

Since $\mathbf{y}\mathbf{H}^\intercal = (\mathbf{c}+\mathbf{e})\mathbf{H}^\intercal = \mathbf{e}\mathbf{H}^\intercal$ only depends on the error in the noisy encoded message $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c} \in \mathcal{C}$, the syndrome is a useful tool in the decoding process.

It turns out that the syndromes with respect to a code are representative of what are known as the cosets of the code.

Definition 2.7 (Coset)**.** Let $\mathcal{C}$ be an $[n,k]_{|\mathbb{F}|}$-code. The *coset* of a vector $\mathbf{y} \in \mathbb{F}^n$ with respect to $\mathcal{C}$ is defined as the set

$$\mathbf{y} + \mathcal{C} \stackrel{\text{def}}{=} \{\mathbf{y} + \mathbf{c} \colon \mathbf{c} \in \mathcal{C}\}\,.$$

1.2 Norm and distance

To work with error-correcting codes and to be able to specify this notion of decoding as well as that of error detection and correction, it is more than useful to equip our vector space with a norm, which we will call here a *weight function* and usually denote by $|\cdot|$. The *distance* induced by this norm will be denoted by d and is such that

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{F}^n,\ d(\mathbf{x},\mathbf{y}) = |\mathbf{x}-\mathbf{y}|.$$

In what follows, the weight functions we will consider will typically be the Hamming weight for vectors and the rank weight for matrices and will be defined in Section 2. For the time being, we give generic definitions that apply to any distance d, starting with that of ball, which is simply the set of elements whose distance to a certain point, the center, is less than or equal to a certain value, the radius.

Definition 2.8 (Ball $\mathscr{B}_r$ of radius r)**.** Let $n \in \mathbb{N}$, $\mathbf{x}$ be a vector of $\mathbb{F}^n$ and $r \leq n$ be a nonnegative integer. The (closed) *ball* $\mathscr{B}_r(\mathbf{x})$ of center $\mathbf{x}$ and *radius* r is defined as the subset of $\mathbb{F}^n$ constituted of vectors at distance at most r from $\mathbf{x}$, *i.e.*,

$$\mathscr{B}_r(\mathbf{x}) \stackrel{\text{def}}{=} \{\mathbf{y} \in \mathbb{F}^n \colon d(\mathbf{x},\mathbf{y}) \leq r\}\,.$$

We will denote by $\mathscr{B}_r$ a ball of radius r and B_r its volume.

We similarly define the notion of sphere, which is just the boundary of a ball:

Definition 2.9 (Sphere $\mathscr{S}_r$ of radius r)**.** Let $n \in \mathbb{N}$, $\mathbf{x}$ be a vector of $\mathbb{F}^n$ and $r \leq n$ be a nonnegative integer. The *sphere* $\mathscr{S}_r(\mathbf{x})$ of center $\mathbf{x}$ and *radius* r is defined as the subset of $\mathbb{F}^n$ constituted of vectors at distance r from $\mathbf{x}$, *i.e.*,

$$\mathscr{S}_r(\mathbf{x}) \stackrel{\text{def}}{=} \{\mathbf{y} \in \mathbb{F}^n \colon d(\mathbf{x},\mathbf{y}) = r\}\,.$$

We will denote by $\mathscr{S}_r$ a sphere of radius r and S_r its volume.

It straightforwardly follows that a ball $\mathscr{B}_r$ is just the union of all the spheres $\mathscr{S}_j$ with $j \leq r$:

$$\mathscr{B}_r = \bigcup_{j=0}^{r} \mathscr{S}_j.$$

Keeping in mind the decoding objective, it is interesting to work with a code such that the balls centered on its codewords span as much as possible $\mathbb{F}^n$, without intersecting, since an intersection would mean that for some vectors of $\mathbb{F}^n$, there are several possible decodings, which is absolutely not desirable. Naturally, we must be interested in the smallest distance between two codewords in order to determine the largest radius we can take so that the balls do not overlap.

Definition 2.10 (Minimum distance $d_{\min}$)**.** Let $\mathcal{C}$ be a code. Its *minimum distance* is defined as

$$d_{\min}(\mathcal{C}) \stackrel{\text{def}}{=} \min_{\mathbf{c},\mathbf{c}' \in \mathcal{C}:\ \mathbf{c} \neq \mathbf{c}'} \left\{ d(\mathbf{c}, \mathbf{c}') \right\}.$$

The *relative minimum distance* is $\delta_{\min}(\mathcal{C}) \stackrel{\text{def}}{=} \frac{d_{\min}(\mathcal{C})}{n}$. When the context is clear, we will simply note $d_{\min}$ and $\delta_{\min}$.

Note that by linearity of the code, we can in fact prove that $d_{\min}(\mathcal{C}) = \min_{\mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}} \{|\mathbf{c}|\}$. From there, taking $r = \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$, it is straightforward that the balls $\mathscr{B}_r(\mathbf{c})$ for $\mathbf{c} \in \mathcal{C}$ are pairwise disjoint, yielding unique decoding. It is now interesting to see how the quantity $d_{\min}$ behaves.

It turns out that standard probabilistic arguments can be used to show that the minimum distance of a random linear code (*i.e.*, a code $\mathcal{C}$ obtained as in Equation 2.1 by a generator matrix $\mathbf{G}$ chosen uniformly at random in $\mathbb{F}^{k \times n}$) is with overwhelming probability equal, up to an additive constant, to a quantity known as the *Gilbert-Varshamov distance* $d_{\mathrm{GV}}(n, k)$ (or simply d_{GV} if there is no ambiguity).

Definition 2.11 (Gilbert-Varshamov distance d_{GV})**.** The Gilbert-Varshamov distance $d_{\mathrm{GV}}(n, k)$ of an $[n, k]_{|\mathbb{F}|}$-code is defined as the largest radius r for which

$$B_r \leq |\mathbb{F}|^{n-k}.$$

This Gilbert-Varshamov distance also happens to quantify the region where we *typically* have unique decoding. More precisely, it turns out that the solution to the decoding problem for a random linear code is unique with probability $1 - 2^{-\Omega(n)}$ as long as for fixed positive ε,

$$r \leq (1 - \varepsilon) d_{\mathrm{GV}}(n, k) \tag{2.3}$$

when n goes to infinity. The Gilbert-Varshamov distance is crucial and will be very useful later on.

1.3 Computational Problems

The security of code-based cryptosystems relies on the hardness of computational problems such as the Decoding Problem and Short Codeword Problem. It is worth noting that efficient algorithms are known for solving these problems on some very specific families of linear codes, but they are still difficult to solve on average, for codes uniformly drawn at random. We precisely define these two problems in what follows.

- The *Decoding Problem* is a fundamental problem in coding theory. Given a linear code $\mathcal{C}$ and a noisy codeword $\mathbf{y} \in \mathbb{F}^n$ (*i.e.*, a codeword to which an error has been added), the decoding problem (sometimes also referred as the *nearest codeword problem*) is to find the nearest codeword in $\mathcal{C}$ to $\mathbf{y}$ regarding the considered distance. In the case of a random linear code $\mathcal{C}$, which is the standard case, this problem can be expressed as follows:

  Problem 2.1. *Decoding Problem (*DP*). The decoding problem with parameters $n, k, t \in \mathbb{N}$, which will be denoted by $\mathsf{DP}_{|\mathbb{F}|}(n, k, t)$, is defined as:*

  - *Given:* $(\mathbf{G}, \mathbf{mG} + \mathbf{e})$ *where* $\mathbf{G} \in \mathbb{F}^{k \times n}$ *and* $\mathbf{m} \in \mathbb{F}^k$ *are sampled uniformly at random over their domain and* $\mathbf{e} \in \mathbb{F}^n$ *over the words of weight* t*,*
  - *Find:* $\mathbf{e}$*.*

  In fact, it really corresponds to decoding the code generated by the rows of $\mathbf{G}$, as in Equation 2.1.

  The best algorithms for solving the decoding problem have exponential complexity in n as soon as t is linear in n and the code rate R is bounded away from 0 and 1.

  The Decoding Problem can be stated in an equivalent manner, known as the *Syndrome Decoding Problem*. It is based on the fact that the parity-check matrix of a linear code $\mathcal{C}$ can be used to detect errors in a received word $\mathbf{y}$ by computing its syndrome. If $\mathbf{y}$ contains no errors, its syndrome is zero. Otherwise, the syndrome reveals information about the location and type of the errors.

- The *Short Codeword Problem* is another fundamental problem in coding theory. Given a linear code $\mathcal{C}$ and an integer w, the short codeword problem is to find a codeword $\mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}$ of weight at most w (regarding the weight induced by the considered distance). This problem is also sometimes referred to as the Low Weight Codeword Problem. In the case of a random linear code, this problem can be expressed as follows:

  Problem 2.2. *Short Codeword Problem (*SCP*). The Short Codeword Problem with parameters $n, k, w \in \mathbb{N}$, which will be denoted by $\mathsf{SCP}_{|\mathbb{F}|}(n, k, w)$, is defined as:*

– *Given:* $\mathbf{H} \in \mathbb{F}^{(n-k)\times n}$ *which is sampled uniformly at random,*

– *Find:* $\mathbf{c} \in \mathbb{F}^n$ *such that* $\mathbf{H}\mathbf{c}^\intercal = \mathbf{0}$ *and the weight of* $\mathbf{c}$ *belongs to* $(0, w]$.

In fact, we are looking for a non-zero codeword $\mathbf{c}$ of weight $\leq w$ in the code defined by the so-called parity-check matrix $\mathbf{H}$, as in Equation 2.2.

The Short Codeword Problem becomes easy when the weight w is above a certain range. The reason is that the code is a vector space of dimension k: by solving a linear system, we can produce codewords with $k-1$ entries equal to 0, which gives good candidates for having a small weight. This strategy produces in polynomial time codewords of weight $\approx \omega_{\text{easy}}(n,k)n$, where $\omega_{\text{easy}}(n,k)$ is some important quantity that we will explicit in details in the next section. Below this quantity $\omega_{\text{easy}}(n,k)$, the best known algorithms for solving the SCP have an exponential complexity for a fixed rate R and a fixed ratio $\omega = \frac{w}{n}$. Obtaining larger weights is also readily obtained by choosing only part of the $k-1$ entries to be equal to 0.

2 Weights

We will focus in this manuscript on two weights, which are also often called *metrics*, namely the Hamming weight and the rank weight. The former was introduced more than seventy years ago by Richard Hamming [Ham50] and is the one that was historically the first used in code-based cryptography, while the latter has been introduced in the context of matrix codes by Philippe Delsarte [Del78]. Other weights can be used to build error-correcting codes such as the Lee weight, due to William Lee [Lee58], but it will not be studied here.

2.1 Hamming Weight

The most commonly used weight in coding theory is the Hamming weight. The Hamming weight w_H of a vector is defined as the number of its non-zero coordinates, implying that the Hamming distance d_H measures the number of positions at which two vectors of equal length differ. Formally, we have

Definition 2.12 (Hamming weight w_H)**.** Given a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$, its *Hamming weight* is defined as

$$w_H(\mathbf{x}) = |\{i \in [\![1, n]\!] : x_i \neq 0\}|.$$

2.1.1 Properties for the Hamming weight

Let $\mathbb{F}_q$ be a fixed field. We give a collection of properties that will be useful in Chapter 3, starting with the volume of a sphere.

Lemma 2.1 (Volume S_r of a sphere $\mathscr{S}_r$)**.** *Let $n \in \mathbb{N}$ and r be a positive integer. The volume of $\mathscr{S}_r$ is*

$$S_r = \binom{n}{r}(q-1)^r.$$

Lemma 2.2 (Bounding S_r (Lemma 3.11 of [Cou20]))**.** *Let $n \in \mathbb{N}$ and $p \in \left[0, 1 - \frac{1}{q}\right]$ such that $pn \in \mathbb{N}$. Then $S_{pn} \leq q^{nh_q(p)}$.*

The Gilbert-Varshamov distance is more precisely defined as follows for Hamming metric.

Lemma 2.3 (Relative Gilbert-Varshamov distance δ_{GV})**.** *The* relative Gilbert-Varshamov distance δ_{GV} *of a random $[n,k]_q$-code is*

$$\delta_{\mathrm{GV}} = h_q^{-1}\left(1 - \frac{k}{n}\right) + O\left(\frac{1}{n}\right).$$

We end with a theorem giving a simple bound involving explicitly the rate of a code and the Gilbert-Varshamov bound.

Theorem 2.1 (Theorem 4.10 of [Cou20])**.** *There exists a sequence of linear codes $(\mathcal{C}_i)_i$ over a fixed field $\mathbb{F}_q$ whose lengths sequence tends to infinity, rates sequence converges to R and relative distance sequence converges to δ and such that*

$$R \geq 1 - h_q(\delta).$$

Finally, we give an expression for the bound $\omega_{\mathrm{easy}}(n,k)$ under which it is difficult to find short codewords by solving a linear system where we aim to produce codewords with $k-1$ entries equal to 0.

Definition 2.13 (Short Codewords Bound)**.** The bound $\omega_{\mathrm{easy}}(n,k)$ of a random $[n,k]_q$-code is defined as

$$\omega_{\mathrm{easy}}(n,k) \stackrel{\mathrm{def}}{=} \frac{q-1}{q}\left(1 - \frac{k}{n}\right).$$

2.1.2 Hardness of DP and SCP in the context of Hamming weight

The decoding problem for random linear codes has been studied for a long time and despite many efforts on this issue, the best algorithms are exponential in the codelength n in the regime where t and k are linear in n. They almost all use the same technique, known as Information Set Decoding, which was introduced in 1962 by Eugene Prange [Pra62] in the case $\mathbb{F} = \mathbb{F}_2$. Many improvements have later on been proposed [LB88; Leo88; Ste88; FS09; BLP11; MMT11; Bec+12; MO15; BM17]. On the other hand, a technique known as Statistical Decoding and introduced in [Jab01] was very recently

improved up to the point that it beats Information Set Decoding algorithms when $\mathbb{F} = \mathbb{F}_2$ and the rate of the code is smaller than 0.3 in [Car+23].

Nonetheless, the decision version of this problem is NP-complete [BMT78]. In 1978, Robert McEliece [McE78] proposed the first public key encryption scheme based on coding theory and relying on the difficulty of decoding a random linear code. Many other public key encryption schemes [Ale03; Mis+12] and schemes such as authentication protocols [Ste93] or pseudorandom generators [FS96] followed, built relying on the hardness of this task.

Decoding and looking for short codewords are problems that have been conjectured to be extremely close. They have been studied for a long time [Pra62; Ste88; Dum89; MMT11; Bec+12; MO15; BM18; Car+23], and for instance in the regime of parameters where the rate $R = \frac{k}{n}$ is fixed in $(0, 1)$, the best algorithms for solving them are the same (namely Information Set Decoding). A reduction from decoding to the problem of finding short codewords is known but in an LPN context [App+17; Bra+19; Yu+19; DR22]. However, even in an LPN context, no reduction is known in the other direction. These problems can be viewed in some sense as a code version of the LWE and SIS problems respectively in lattice-based cryptography [Reg09].

The decision version of the short codeword problem has also been proved to be NP-complete [Var97] and while the security of many code-based cryptosystems relies on the hardness of the decoding problem, it can also be based on finding a "short" codeword, as in [Mis+12] or in [App+17; Bra+19; Yu+19] to build collision resistant hash functions for example.

2.2 Rank Weight

In recent years, there has been growing interest in developing cryptographic primitives based on the rank weight, a mathematical concept that measures the dimension of the vector space spanned by the rows of a matrix. It is thus tempting to define codes embedded with this metric: these codes are named *matrix codes* and their codewords are matrices. The rank metric has been extensively studied in the area of network coding, where it is well-suited for the kind of errors that occur.

Definition 2.14 (Matrix code)**.** Let $\mathscr{M}_{m\times n}(\mathbb{F}_q)$ be the set of $m \times n$ matrices with coefficients in $\mathbb{F}_q$. An $[m \times n, K]_q$-code is a K-dimensional linear subspace of $\mathscr{M}_{m\times n}(\mathbb{F}_q)$ called a *matrix code*.

The set of codes on $\mathbb{F}_{q^m}$, which are known as $\mathbb{F}_{q^m}$*-linear codes*, is a subset of the class of matrix codes (see [FLP08]). Indeed, we can associate a matrix from $\mathbb{F}_q^{m\times n}$ to any vector from $\mathbb{F}_{q^m}^n$ by using a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ and thus associate a matrix code to a $\mathbb{F}_{q^m}$-linear code.

Definition 2.15 (Matrix code associated to a $\mathbb{F}_{q^m}$-linear code)**.** Let $\beta = (\beta_1, \ldots, \beta_m)$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ and $\mathcal{C}$ be a $\mathbb{F}_{q^m}$-linear code. To each word $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{C}$ we can associate a $m \times n$ matrix $\mathbf{M}(\mathbf{x}) = (m_{ij})_{\substack{i \in [\![1,m]\!] \\ j \in [\![1,n]\!]}}$ with entries in $\mathbb{F}_q$ such that for any j in $[\![1, n]\!]$:

$$x_j = \sum_{i=1}^{m} m_{ij}\beta_i.$$

The set $\{\mathbf{M}(\mathbf{x}) \colon \mathbf{x} \in \mathcal{C}\}$ is the matrix code over $\mathbb{F}_q$ associated to $\mathcal{C}$. The rank of $\mathbf{x}$ is then defined as the rank of $\mathbf{M}(\mathbf{x})$:

$$\operatorname{rank}(\mathbf{x}) \stackrel{\text{def}}{=} \operatorname{rank}(\mathbf{M}(\mathbf{x})).$$

The rank distance d_R follows from this definition. For $\mathbf{x}, \mathbf{y} \in \mathbb{F}_{q^m}^n$,

$$d_R(\mathbf{x}, \mathbf{y}) = \operatorname{rank}(\mathbf{x} - \mathbf{y}).$$

Note that the representation of a $\mathbb{F}_{q^m}$-linear code is cheaper than the one of the corresponding matrix code, making the former particularly interesting.

Remark 2.1. Let us take a $[n, k]_{q^m}$-code and its associated matrix code, which is a $[mn, mk]_q$-code. We will need $k(n-k)m \log q$ bits to represent the former while $mk(mn-mk) \log q$ bits are required to represent the latter. It is therefore more appropriate to work with $\mathbb{F}_{q^m}$-linear codes as they offer a storage gain of a factor m over their matrix code counterpart.

2.2.1 Properties for the rank weight

We can now give a closed formula for the size of a sphere of radius r. It is equal to the number of matrices in $\mathbb{F}_q^{m \times n}$ of rank r.

Lemma 2.4 (Volume S_r of a sphere $\mathscr{S}_r$)**.** *Let r be a positive integer. We have*

$$S_r \stackrel{def}{=} |\mathscr{S}_r| = \left(\prod_{j=0}^{r-1} (q^m - q^j) \right) \begin{bmatrix} n \\ r \end{bmatrix}_q.$$

Where the Gaussian coefficient $\begin{bmatrix} n \\ r \end{bmatrix}_q$ is defined by

$$\begin{bmatrix} n \\ r \end{bmatrix}_q \stackrel{\text{def}}{=} \begin{cases} \prod_{j=0}^{r-1} \frac{q^n - q^j}{q^r - q^j} & \text{if } r \leq n \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

We have some bounds on the volume of a sphere and thus on the one of a ball:

Lemma 2.5. *We have*

$$q^{(m+n-2)r-r^2} \leq S_r \leq q^{(m+n+1)r-r^2} \tag{2.5}$$

$$q^{(m+n-2)r-r^2} \leq B_r \leq q^{(m+n+1)r-r^2+1} \tag{2.6}$$

Using the fact that $\prod_{i=1}^{\infty}(1-q^{-i})$ is some constant depending on q, we can prove the following asymptotic expressions for Gaussian coefficients and for the volume of a sphere:

Lemma 2.6 (Asymptotic expressions)**.** *As $n \to +\infty$,*

$$\begin{bmatrix} n \\ r \end{bmatrix}_q = \Theta\left(q^{r(n-r)}\right) \tag{2.7}$$

$$S_r = \Theta\left(q^{r(m+n-r)}\right) \tag{2.8}$$

From Equation 2.8 we deduce that

$$\frac{S_{u+1}}{S_u} = \Theta\left(\frac{q^{(u+1)(m+n-u-1)}}{q^{u(m+n-u)}}\right) = \Theta\left(q^{m+n-2u-1}\right). \tag{2.9}$$

We can take back the Definition 2.11 of the Gilbert-Varshamov distance and approximate the ball $\mathscr{B}_r$ by its boundary $\mathscr{S}_r$. Injecting Equation 2.8 in the expression then yields a useful and simple relation on the Gilbert-Varshamov distance, stated in the following Lemma for any matrix code.

Lemma 2.7 (Gilbert-Varshamov distance [FLP08])**.** *The relative Gilbert-Varshamov distance δ_{GV} of an $[m \times n, K]_q$-code (with $m \geq n$) satisfies the relation*

$$K \leq (m - \delta_{\mathrm{GV}})(n - \delta_{\mathrm{GV}}).$$

2.2.2 Hardness of DP and SCP in the context of rank weight

Contrary to their counterparts in Hamming metric, the decision versions of the decoding and short codeword problems are not known to be NP-complete. However, Gaborit and Zémor proved that the decoding problem in Hamming metric, which is an NP-complete problem, can be reduced by a randomized reduction to the decoding problem in rank metric [GZ16].

In addition, some cryptosystems have been proposed, based on these problems. Namely, we can cite the first one, introduced in [GPT91] and notable for the fact that it is a translation in rank metric of the McEliece cryptosystem. It suffered from several attacks, *e.g.*, in [Gib96; Ove05]. It is worth noticing that two rank-based cryptosystems made it to the second round of the NIST Post-Quantum Standardization Process, namely

ROLLO [Ara+19] and RQC [Agu+20] and several rank-based signature schemes have been presented to the NIST first round of the Standardiation Process of additional signatures, such as MIRA [Ara+23b], MiRitH [Adj+23] or RYDE [Ara+23a] for example.

Regarding attacks on these problems, for a long time combinatorial attacks were the only ones considered useful and consist in some Information Set Decoding equivalents [CS96; GRS16; Ara+18]. However, algebraic attacks have made their way and have recently been shown to be more effective than the former [OJ02; Bar+20; Bar+20; Bar+22]. These algebraic attacks have notably impacted NIST's decision not to move ROLLO and RQC to the third round of the Standardization Process (as well as the signature schemes Rainbow and GeMSS, in multivariate cryptography).

Finally, it is important to note that the key sizes of rank metric cryptosystems are generally smaller than those in Hamming metric, since the complexity of rank metric attacks is higher than in Hamming metric. On the other hand, the Hamming metric has been much more widely studied than the rank metric, which is more recent.

Chapter 3
From Decoding to Finding Short Codewords

Decoding and looking for short codewords are problems that have been conjectured to be extremely close. They have been studied for a long time [Pra62; Ste88; Dum89; MMT11; Bec+12; MO15; BM18; Car+23], and a reduction from decoding to the problem of finding short codewords is known but in an LPN context [App+17; Bra+19; Yu+19; DR22]. However, even in an LPN context, no reduction is known in the other direction. These problems can be viewed in some sense as a code version of the LWE and SIS problems respectively in lattice-based cryptography [Reg09]. Our contribution in this chapter, based on [DRT23], is precisely to give the code-based version of this reduction, namely a quantum reduction from finding short codewords to decoding. This problem was open for quite some time. To simplify the statements, we will state it in the regime of parameters where the rate R is fixed in $(0, 1)$, but actually it also works in the LPN setting (but needs to be adapted in several places where we use exponential bounds in n).

There is a fundamental difficulty of reducing the research of low weight codewords to decoding a linear code which is due to the fact that the nature of these two problems is very different. Decoding concentrates on a region of parameters where there is typically just one solution, whereas finding low weight codewords concentrates on a region of parameters where there are solutions (and typically an exponential number of solutions). This makes these problems inherently very different. This was also the case for the reduction from SIS to LWE and the fact that we can have a reduction from one to another by looking for quantum reductions instead of classical reductions was really a breakthrough at that time.

Regev's quantum reduction strategy adapted to coding theory. In [Reg05] (see also the extended version [Reg09]) Regev showed how to transform a random oracle solving the decoding problem in a lattice into a quantum algorithm outputting a rather small vector in the dual lattice. Our aim is to show here that the natural translation of this approach in coding theory gives an algorithm that outputs a rather small vector in the dual code. Roughly speaking Regev's approach relies on a fundamental result about the Fourier transform.

Proposition 3.1. *Consider an Abelian group G and a function $f : G \mapsto \mathbb{C}$ that is constant on the cosets of a subgroup H of G. Then the Fourier transform $\widehat{f}$ is constant on the dual subgroup $H^\perp$.*

This innocent looking fact, together with the fact that the quantum Fourier transform can be performed in polylog time when the group G is Abelian, is arguably the key to several remarkable quantum algorithms solving in polynomial time the period finding in a vectorial Boolean function [Sim94], the factoring problem or the discrete logarithm problem [Sho94]. All of these problems can be rephrased in terms of the hidden Abelian subgroup problem, where one is given such a function f that is constant (and distinct) on the cosets of an unknown subgroup H and one is asked to recover H. This is achieved by

(*i*) creating the uniform superposition $\frac{1}{\sqrt{|G|}} \sum_{x \in G} |x\rangle |f(x)\rangle$,

(*ii*) measuring the second register and discarding it, yielding a quantum state of the form $\frac{1}{\sqrt{|H|}} \sum_{h \in H} |x + h\rangle$,

(*iii*) applying the QFT to it yielding a superposition of elements in the dual subgroup $H^\perp$ (and therefore gaining information on H in this way).

Proposition 3.1 is used in a similar way in Regev's reduction. Translating Regev's reduction in coding theory would use this framework by considering that the linear code $\mathcal{C}$ we want to decode plays the role of the aforementioned H. From now on we will assume that this code is of dimension k and length n over $\mathbb{F}_q$. The algorithm would basically look as follows for reducing the search of small codewords in the dual code $\mathcal{C}^\perp = \left\{ \mathbf{c}^\perp \in \mathbb{F}_q^n : \mathbf{c} \cdot \mathbf{c}^\perp = 0, \ \forall \mathbf{c} \in \mathcal{C} \right\}$ (where $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^n x_i y_i$ is the standard inner product in $\mathbb{F}_q^n$) to decoding errors of weight t in $\mathcal{C}$.

Step 1. Use a quantized version of the decoding algorithm to prepare the state

$$\frac{1}{\sqrt{Z}} \sum_{\mathbf{c} \in \mathcal{C}, \mathbf{e} \in \mathbb{F}_q^n} \pi_{\mathbf{e}} |\mathbf{c} + \mathbf{e}\rangle$$

where Z is a normalizing constant and $(|\pi_{\mathbf{e}}|^2)_{\mathbf{e}}$ is a probability distribution on errors that concentrate around the weight t we are able to decode. This is done

(*i*) by preparing first a superposition of codewords and errors,

$$\frac{1}{\sqrt{Z}} \sum_{\mathbf{c} \in \mathcal{C}} \sum_{\mathbf{e} \in \mathbb{F}_q^n} \pi_{\mathbf{e}} |\mathbf{c}\rangle |\mathbf{e}\rangle ,$$

(*ii*) then adding the second register to the first one to get the entangled state

$$\frac{1}{\sqrt{Z}} \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \pi_{\mathbf{e}} \left|\mathbf{c}+\mathbf{e}\right\rangle \left|\mathbf{e}\right\rangle$$

(*iii*) and finally disentangling it thanks to a quantized version of the decoding algorithm, which from $\mathbf{c}+\mathbf{e}$ recovers $\mathbf{e}$ and subtracts it from the second register to get the state

$$\frac{1}{\sqrt{Z}} \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \pi_{\mathbf{e}} \left|\mathbf{c}+\mathbf{e}\right\rangle \left|\mathbf{0}\right\rangle .$$

Step 2. Apply the QFT on $\mathbb{F}_q^n$ to obtain a superposition of elements $\mathbf{c}^\perp$ in the dual code

$$\sum_{\mathbf{c}^\perp\in\mathcal{C}^\perp} \alpha_{\mathbf{c}^\perp} \left|\mathbf{c}^\perp\right\rangle .$$

Step 3. Measure the register to output $\mathbf{c}^\perp$ of rather small norm in $\mathcal{C}^\perp$.

The second step is a direct consequence of Proposition 3.1. The last one raises the issue of whether or not the QFT concentrates the weight of the vector output by this algorithm on weights t' for which finding a codeword in $\mathcal{C}^\perp$ is not known to be easy, as it is the case for Regev's reduction on lattices equipped with the Euclidean metric.

On the difficulty of translating Regev's reduction to the Hamming metric. The natural analog in the Hamming metric case of the Gaussian noise model used in Regev's reduction [Reg09] is the q-ary symmetric channel. Its associated quantum state is given by

$$\left|\pi^{\mathrm{SC}}\right\rangle \stackrel{\text{def}}{=} \sum_{\mathbf{e}\in\mathbb{F}_q^n} (1-\tau)^{\frac{n-|\mathbf{e}|}{2}} \left(\frac{\tau}{q-1}\right)^{\frac{|\mathbf{e}|}{2}} \left|\mathbf{e}\right\rangle = \left(\sqrt{1-\tau}\left|0\right\rangle + \sum_{\alpha\in\mathbb{F}_q^*} \sqrt{\frac{\tau}{q-1}}\left|\alpha\right\rangle\right)^{\otimes n}$$

where $|\mathbf{e}|$ stands for the Hamming weight of $\mathbf{e}$, n for the length of $\mathbf{e}$ and τ is the crossover probability of the q-ary symmetric channel. Indeed, measuring such a state yields an error distributed like a q-ary symmetric channel of crossover probability τ. In both cases (be it for the Gaussian noise or the q-ary symmetric channel), the Fourier transform yields a dual noise which is again Gaussian or q-ary symmetric respectively and the quantum state corresponding to the error is a product state which considerably simplifies the computation. In the case of the q-ary symmetric noisy chanel, applying the QFT on $\left|\pi^{\mathrm{SC}}\right\rangle$ yields the quantum state

$$\sum_{\mathbf{e}\in\mathbb{F}_q^n} (1-\tau^\perp)^{\frac{n-|\mathbf{e}|}{2}} \left(\frac{\tau^\perp}{q-1}\right)^{\frac{|\mathbf{e}|}{2}} \left|\mathbf{e}\right\rangle = \left(\sqrt{1-\tau^\perp}\left|0\right\rangle + \sum_{\alpha\in\mathbb{F}_q^*} \sqrt{\frac{\tau^\perp}{q-1}}\left|\alpha\right\rangle\right)^{\otimes n} \stackrel{\text{def}}{=} \widehat{\left|\pi^{\mathrm{SC}}\right\rangle}$$

where (see Fact 1)

$$\tau^\perp \stackrel{\text{def}}{=} \frac{\left(\sqrt{(q-1)(1-\tau)} - \sqrt{\tau}\right)^2}{q}.$$

This new quantum state represents a q-ary symmetric channel of parameter $\tau^\perp$. If we measure $\left|\pi^{\text{SC}}\right\rangle$ we get relative weights $\approx \tau$ whereas if we measure $\widehat{|\pi^{\text{SC}}\rangle}$ we get relative weights around $\tau^\perp$.

It would thus be tempting to conclude that the "ideal" version of the algorithm presented above will output dual codewords (in Step 3) of relative Hamming weight $\approx \tau^\perp < \omega_{\text{easy}}$, *i.e.*, in the regime where there is a chance that it is difficult to produce such words. However, this natural approach runs into the following problem. The parameter τ of the Bernoulli noise has to be chosen so that the typical error weight τn is equal to or slightly below the weight t we can decode. Such a τ is therefore at most the relative Gibert-Varshamov δ_{GV}. However, it can be proved that in this case the most likely relative weight we measure at Step 3 is *typically zero* if $\tau^\perp < \omega_{\text{easy}}$. In other words, the straightforward application of Regev's approach to coding theory fails to give a useful reduction.

We will give in Remark 3.1 another explanation for the failure of this approach. It can be summarized by saying that the Bernoulli noise model is not concentrated enough on its typical weight τn.

Fact 1. *Let* $\left|\pi^{SC}\right\rangle \stackrel{def}{=} \left(\sqrt{1-\tau}\,|0\rangle + \sum_{\alpha\in\mathbb{F}_q^*}\sqrt{\frac{\tau}{q-1}}\,|\alpha\rangle\right)^{\otimes n}$, *where* $\tau \in \left[0, \frac{q-1}{q}\right]$, *then*

$$\widehat{|\pi^{SC}\rangle} = \left(\sqrt{1-\tau^\perp}\,|0\rangle + \sum_{\alpha\in\mathbb{F}_q^*}\sqrt{\frac{\tau^\perp}{q-1}}\,|\alpha\rangle\right)^{\otimes n}, \quad \textit{where} \tag{3.1}$$

$$\tau^\perp \stackrel{def}{=} \frac{\left(\sqrt{(q-1)(1-\tau)} - \sqrt{\tau}\right)^2}{q}. \tag{3.2}$$

Proof. Let $|\psi\rangle \stackrel{\text{def}}{=} \sqrt{1-\tau}\,|0\rangle + \sum_{\alpha\in\mathbb{F}_q^*}\sqrt{\frac{\tau}{q-1}}\,|\alpha\rangle$. Then, it is readily verified that

$$\widehat{|\pi^{\text{SC}}\rangle} = \left(\widehat{|\psi\rangle}\right)^{\otimes n} \qquad \text{where} \qquad \widehat{|\psi\rangle} = \frac{1}{\sqrt{q}}\left(\beta_0 + \sum_{y\in\mathbb{F}_q^*}\beta_y\,|y\rangle\right) \qquad \text{with}$$

$$\beta_0 = \sqrt{1-\tau} + (q-1)\sqrt{\frac{\tau}{q-1}} \qquad \text{and} \qquad \beta_y = \sqrt{1-\tau} + \sqrt{\frac{\tau}{q-1}}\sum_{x\in\mathbb{F}_q^*}\chi_y(x).$$

It is easy to verify that for any $y \in \mathbb{F}_q^*$ we have for any $x_0 \in \mathbb{F}_q^*$:

$$\sum_{x\in\mathbb{F}_q}\chi_y(x) = \sum_{x\in\mathbb{F}_q}\chi_y(x_0\cdot x) = \chi_y(x_0)\sum_{x\in\mathbb{F}_q}\chi_y(x)$$

Since there exists $x_0 \in \mathbb{F}_q^*$ such that $\chi_y(x_0) \neq 1$, we deduce that $\sum_{x\in\mathbb{F}_q} \chi_y(x) = 0$. Now since $\sum_{x\in\mathbb{F}_q} \chi_y(x) = 1 + \sum_{x\in\mathbb{F}_q^*} \chi_y(x)$, we have $\sum_{x\in\mathbb{F}_q^*} \chi_y(x) = -1$ for any y in $\mathbb{F}_q^*$ and therefore

$$\begin{aligned}
\frac{\beta_y}{\sqrt{q}} &= \sqrt{\frac{1-\tau}{q}} - \sqrt{\frac{\tau}{q(q-1)}} \\
&= \sqrt{\frac{(q-1)(1-\tau)}{q(q-1)}} - \sqrt{\frac{\tau}{q(q-1)}} \\
&= \frac{\sqrt{\frac{(q-1)(1-\tau)}{q}} - \sqrt{\frac{\tau}{q}}}{\sqrt{q-1}} \\
&= \frac{\sqrt{\tau^\perp}}{\sqrt{q-1}}.
\end{aligned}$$

It is then clear that $\frac{\beta_0}{\sqrt{q}} = \sqrt{1-\tau^\perp}$ from $\sum_{y\in\mathbb{F}_q} |\beta_y|^2 = 1$, concluding the proof. □

Our approach. To tackle this issue, it would therefore be natural to choose the most concentrated noise model on the weight t, namely:

$$\left|\pi^{\text{unif}}\right\rangle = \sum_{\mathbf{e}:\ |\mathbf{e}|=t} \frac{1}{\sqrt{S_t}} |\mathbf{e}\rangle$$

where S_t is the cardinality of the sphere of radius t in the Hamming metric, *i.e.*, $S_t = (q-1)^t \binom{n}{t}$. Understanding of which weight w is the outcome after measuring the state $\sum_{\mathbf{c}^\perp\in\mathcal{C}^\perp} \alpha_{\mathbf{c}^\perp} \left|\mathbf{c}^\perp\right\rangle$ in Step 3 is more difficult in the constant weight error model than in the Bernoulli noise model. In particular, it involves properties of Krawtchouk polynomials. However, it can be shown that when $\omega \stackrel{\text{def}}{=} \frac{w}{n}$ lies in a whole interval $\left[\tau^\perp, \tau_+^\perp\right]$ where $\tau_+^\perp \stackrel{\text{def}}{=} \frac{\left(\sqrt{(q-1)(1-\tau)}+\sqrt{\tau}\right)^2}{q}$, we have many points where the probability of measuring a word of weight w is actually $\frac{1}{\mathsf{poly}(n)}$. The "dual" weight distribution is not really concentrated on a single value but spread over a large interval. This would provide a useful reduction when we use a decoding algorithm that succeeds on a non-negligible set of inputs.

Unfortunately, to be relevant in a cryptographic context, we must consider the case where decoding succeeds only for a potentially very low probability ε, the aim being to turn our decoding algorithm into an algorithm that produces a low weight codeword from the dual with some probability $\mathsf{poly}(\varepsilon)$. This cannot be obtained with the uniform distribution on the sphere of radius t. Indeed, the "ideal" version of the algorithm we presented before (where we assume we always succeed with our decoding algorithm) describes a state we obtain in Step 2 that is not completely orthogonal (the scalar product is bounded from below by a quantity $\mathsf{poly}(\varepsilon)$) to the "real" state after applying this approximate decoding process and the QFT. If we were to measure this state directly (starting from

the uniform noise model over the sphere of radius t), we would only be sure to measure a word of some relative weight lying in the interval $\left[\tau^{\perp}, \tau_{+}^{\perp}\right]$ with probability $\mathsf{poly}(\varepsilon)$, since the "ideal" state concentrates its relative weight distribution in all this interval. In this way, we cannot ensure the measurement of a dual codeword of smallest possible relative weight, namely $\tau^{\perp}$, it could be $1/2 \in \left[\tau^{\perp}, \tau_{+}^{\perp}\right]$. We really need here a distribution that is rather sharply concentrated around the decoding radius of our decoding algorithm, but whose Fourier transform is also sharply concentrated around a certain weight.

So far, two noise models have been considered for the reduction to work, each with an advantage and a drawback:

- the q-ary symmetric noise $\left|\pi^{\mathrm{SC}}\right\rangle$ is not concentrated enough on its typical weight τn *but* its dual noise is sufficiently concentrated on $\tau^{\perp} n$,
- the uniform noise $\left|\pi^{\mathrm{unif}}\right\rangle$ on the sphere of radius τn is sufficiently concentrated *but* its dual noise is spread out on the whole interval $\left(\tau^{\perp} n, \tau_{+}^{\perp} n\right)$.

Interestingly, the issues with these two distributions are opposite. Fortunately, it turns out that we have a natural noise model to get a best-of-both-worlds model: *truncating* the q-ary symmetric noisy channel. More precisely, consider the following noise model (for some small enough constant $\varepsilon > 0$),

$$\left|\pi^{\mathrm{Trunc}}\right\rangle = \frac{1}{\sqrt{N}} \sum_{\substack{\mathbf{e}: \\ |\mathbf{e}| \in [(1-\varepsilon)t, (1+\varepsilon)t]}} (1-\tau)^{\frac{n-|\mathbf{e}|}{2}} \left(\frac{\tau}{q-1}\right)^{\frac{|\mathbf{e}|}{2}} |\mathbf{e}\rangle$$

where N is a normalizing factor. This noise model solves our above issues with $\left|\pi^{\mathrm{SC}}\right\rangle$ and $\left|\pi^{\mathrm{unif}}\right\rangle$ because it verifies our two constraints for the reduction to work:

(i) its weight distribution is sufficiently concentrated around τn,

(ii) its dual noise after applying the Fourier transform is concentrated on the relative weight $\tau^{\perp} n$.

Contrary to (i), assertion (ii) may seem unclear. It relies on the following equality as we will show in Lemma 3.11

$$\left\| \left|\pi^{\mathrm{Trunc}}\right\rangle - \left|\pi^{\mathrm{SC}}\right\rangle \right\| = 2^{-\Omega(n)}$$

where $\|\cdot\|$ stands for the norm of the Hilbert space in which the quantum states are embedded. Therefore, applying the Fourier transform (which is an isometry for $\|\cdot\|$) on $\left|\pi^{\mathrm{Trunc}}\right\rangle$ will yield a quantum state which is $2^{-\Omega(n)}$-close of $\widehat{|\pi^{\mathrm{SC}}\rangle}$.

With our approach and the truncated q-ary symmetric channel, we transform through the QFT a decoding algorithm correcting τn errors into an algorithm outputting with non-negligible probability words of weight $\approx \tau^{\perp} n$ in the dual code. The distance $\tau^{\perp}$ is clearly a decreasing function of τ and the issue is now whether or not there exists a $\tau < \delta_{\mathrm{GV}}(n,k)$ (this is the biggest value for which we can hope that decoding is successful with probability $1-o(1)$) such that $\tau^{\perp} < \omega_{\mathrm{easy}}(n, n-k)$ (here we want to find short codewords in the dual code $\mathcal{C}^{\perp}$ which is of dimension $n-k$), since this would yield a useful reduction. It turns out that in many cases we have to choose $\tau > \delta_{\mathrm{GV}}(n,k)/2$, meaning that we are not in the regime where decoding necessarily has at most one solution. This complicates the proof of the reduction somewhat since with a quantized version of the decoding algorithm, we will not be able to produce at Step 1 the state $\frac{1}{\sqrt{Z}} \sum_{\mathbf{c} \in \mathcal{C}} \sum_{\mathbf{e} \in \mathbb{F}_q^n} \pi_{\mathbf{e}} |\mathbf{c} + \mathbf{e}\rangle$ (since decoding fails for some $\mathbf{e}$) but we will show that as long as $\tau < \delta_{\mathrm{GV}}(n,k)$, we will get a state close to it. This will be enough for our purpose.

By putting all these ingredients together, we are able to prove the following result.

Theorem 3.1 (informal)**.** *The short codeword problem* $\mathsf{SCP}(q, n, n-k, w)$ *reduces to the decoding problem* $\mathsf{DP}(q, n, k, t)$ *for* $w = \tau^{\perp} n + O(1)$ *where*

$$\tau \overset{def}{=} \frac{t}{n} \qquad \text{and} \qquad \tau^{\perp} \overset{def}{=} \frac{\left(\sqrt{(q-1)(1-\tau)} - \sqrt{\tau}\right)^2}{q}.$$

It will turn out that for $q = 2$ (see Section 2) we can find for any rate $R = \frac{k}{n}$ in $(0,1)$ a $t < d_{\mathrm{GV}}(n,k)$ for which the corresponding w is below $\omega_{\mathrm{easy}}(n, n-k)n$ (the reduction is useful in this case). It also corresponds to parameter ranges that are relevant for certain cryptographic applications (see [AFS05] whose security relies on the SCP problem in a parameter range which is covered by our reduction and [Ste93] which is an identification scheme whose security actually relies on the decoding problem just below the Gilbert-Varshamov distance). Unfortunately, this is not true anymore when $q \geq 5$, where there is always a range for R for which w is above $\omega_{\mathrm{easy}}(n, n-k)n$, for any choice of $t < d_{\mathrm{GV}}(n,k)$. The reduction then starts to become useless since the range in question gets larger when q grows. Roughly speaking, when q grows, the Hamming metric gets coarser (we have only $n+1$ different values for the metric on $\mathbb{F}_q^n$, whereas the size of the ambient space gets bigger) and this results in the range of values of R for which this reduction is useful becoming smaller.

Considering other metrics. The whole approach we have followed here (properly choosing the error distribution and going beyond the unique decoding radius for decoding if necessary) can of course be adapted to other metrics. It is easy for instance to apply it to the rank metric which is becoming increasingly popular in code-based cryptography, see for instance [Ara+19; Agu+20; Bel+19; Bel+20]. This metric is even coarser than the Hamming metric: on $\mathbb{F}_q^{m \times n}$ there are only $1 + \min(m,n)$ different values for the

rank weight (given a matrix, it is defined as its rank). In this case, as we will see, the reduction is always useless (*i.e.*, reduces to weights which are always easy to produce for a random linear code).

Contents

1 Quantum Reduction from Sampling Short Codewords to Decoding

1.1 A general result

We assume here that we have a probabilistic algorithm $\mathscr{A}$ that solves (sometimes) the decoding problem at distance t. Its inputs are a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ of a code $\mathcal{C} \subseteq \mathbb{F}_q^n$ (*i.e.*, $\mathcal{C} = \left\{\mathbf{uG} \colon \mathbf{u} \in \mathbb{F}_q^k\right\}$) and a noisy codeword $\mathbf{c}+\mathbf{e}$ where $\mathbf{c}$ belongs to $\mathcal{C}$. We denote by $\mathbf{r} \in \mathbb{F}_2^\ell$ the internal coins of $\mathscr{A}$. It outputs with a certain probability ε, the "right" $\mathbf{e}$ when being fed with $\mathbf{c}+\mathbf{e}$ where $\mathbf{c}$ and $\mathbf{e}$ are uniformly chosen at random in $\mathcal{C}$ and among the errors of weight t respectively:

$$\varepsilon \stackrel{\text{def}}{=} \mathbb{P}_{\mathbf{G},\mathbf{c},\mathbf{e},\mathbf{r}}\left(\mathscr{A}(\mathbf{G},\mathbf{c}+\mathbf{e},\mathbf{r}) = \mathbf{e}\right). \tag{3.3}$$

The quantum reduction starts by building the initial superposition

$$\frac{1}{\sqrt{2^\ell q^k}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{r}\in\mathbb{F}_2^\ell} \pi_{\mathbf{e}} \left|\mathbf{e}\right\rangle \left|\mathbf{c}\right\rangle \left|\mathbf{r}\right\rangle$$

where $|\pi\rangle \stackrel{\text{def}}{=} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \pi_{\mathbf{e}} |\mathbf{e}\rangle$ is some quantum superposition of errors. It then acts as follows.

Algorithm of the quantum reduction.

$$\begin{aligned}
&\text{Initial state} && = && \frac{1}{\sqrt{2^\ell q^k}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{r}\in\mathbb{F}_2^\ell} \pi_{\mathbf{e}} |\mathbf{e}\rangle |\mathbf{c}\rangle |\mathbf{r}\rangle \\
&\text{adding } \mathbf{e} \text{ to } \mathbf{c}\text{:} && \mapsto && \frac{1}{\sqrt{2^\ell q^k}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{r}\in\mathbb{F}_2^\ell} \pi_{\mathbf{e}} |\mathbf{e}\rangle |\mathbf{c}+\mathbf{e}\rangle |\mathbf{r}\rangle \\
&\text{applying } \mathscr{A}\text{:} && \stackrel{\mathscr{A}}{\mapsto} && \frac{1}{\sqrt{2^\ell q^k}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{r}\in\mathbb{F}_2^\ell} \pi_{\mathbf{e}} |\mathbf{e}-\mathscr{A}(\mathbf{G},\mathbf{c}+\mathbf{e})\rangle |\mathbf{c}+\mathbf{e}\rangle |\mathbf{r}\rangle \stackrel{\text{def}}{=} |\psi_{\mathscr{A}}\rangle \qquad (3.4) \\
&\mathsf{QFT} \text{ on 2nd reg:} && \mapsto && \widehat{|\psi_{\mathscr{A}}\rangle} \qquad (3.5) \\
&\text{measuring the state:} && \mapsto && |\mathbf{e}\rangle \left|\mathbf{c}^\perp\right\rangle |\mathbf{r}\rangle \qquad (3.6)
\end{aligned}$$

We will now give a general theorem about an algorithm of this kind and will show that it produces a codeword of the dual code $\mathcal{C}^\perp$ of some weight u with probability $\mathsf{poly}(\varepsilon)$ when certain conditions are met. We recall that S_u denotes the volume of a sphere of radius u (see Definition 2.9).

Theorem 3.2. *Assume that $|\pi\rangle$ is radial and non-negative, i.e., $\pi_{\mathbf{e}} = f(|\mathbf{e}|)$ for some non-negative function f. Assume that $\widehat{|\pi\rangle} = \sum_{\mathbf{e}\in\mathbb{F}_q^n} \widehat{\pi_{\mathbf{e}}} |\mathbf{e}\rangle$ is radial too[(1)] and let $\widehat{f}(w) = \widehat{\pi_{\mathbf{e}}}$ for any element $\mathbf{e}$ of $\mathbb{F}_q^n$ of weight w. Furthermore, assume that there exists an interval*

[(1)] In other words, we assume that the Fourier transform is radially preserving. This property depends on the characters chosen to define the Fourier transform and the metric. Recall that a radial function is a function which is constant on spheres centered around 0. This property clearly holds for functions $f\colon \mathbb{F}_q^n \to \mathbb{C}$ with the characters chosen here and the Hamming metric. We give this more general statement in order to apply it in other cases of interest, for instance the rank metric.

$\mathscr{W} \subseteq [\![0, n]\!]$ such that:

(Concentration of π) $$\frac{\langle \pi | \mathbf{1} \rangle^2}{q^{n-k}} = 2^{-\Omega(n)}, \tag{C1}$$

(Exponentially many dual codewords of weight $u \in \mathscr{W}$) $$\sum_{u \in \mathscr{W}} \frac{q^k}{S_u} = 2^{-\Omega(n)}, \tag{C2}$$

(Concentration of the dual distribution $\widehat{\pi}$ on $\mathscr{W}$) $$\sum_{u \in \mathscr{W}} S_u \left| \widehat{f}(u) \right|^2 = 1 - 2^{-\Omega(n)} \tag{C3}$$

with $|\mathbf{1}\rangle$ being the (unnormalized) superposition of errors : $|\mathbf{1}\rangle \stackrel{def}{=} \sum_{\mathbf{e} \in \mathbb{F}_q^n} |\mathbf{e}\rangle$.
Suppose that there exists an algorithm $\mathscr{A}$ solving the decoding problem $\mathsf{DP}(q, n, k, t)$ with success probability ε. Then, there exists a quantum algorithm which takes as input a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ of $\mathcal{C}$ and outputs a codeword of weight $u \in \mathscr{W}$ in $\mathcal{C}^\perp$ with probability greater than $\frac{p_t^2 \varepsilon^3}{16} - O(p_t^4 \varepsilon^5) - 2^{-\Omega(n)} - O\left(q^{-\min(k, n-k)}\right)$ where $p_t \stackrel{def}{=} \sum_{\mathbf{e}:\, |\mathbf{e}|=t} |\pi_{\mathbf{e}}|^2 = S_t f(t)^2$.

Remark 3.1.

- We will use this theorem for Hamming and rank metrics, but it can be applied to *any* metric for which the Fourier transform is radially preserving.
- When $\pi_{\mathbf{e}}$ is non-negative, Condition (*C*1) basically requires the probability distribution on $\mathbb{F}_q^n$, $\mu \stackrel{\text{def}}{=} (\pi_{\mathbf{e}}^2)_{\mathbf{e} \in \mathbb{F}_q^n}$, to be sufficiently concentrated. This quantity can be expressed as $q^k(1 - H^2(\mu, U))^2$, where U stands for the uniform distribution over $\mathbb{F}_q^n$ and $H(\mathbf{p}, \mathbf{q}) \stackrel{\text{def}}{=} \sqrt{1 - \sum_i \sqrt{p_i q_i}}$ is the Hellinger distribution between two probability distributions $\mathbf{p}$ and $\mathbf{q}$ defined over a same probability space.
  1. It is clearly maximal for the uniform probability distribution over $\mathbb{F}_q^n$, and
  2. on the other hand, when considering the Hamming metric and $|\pi\rangle = \left|\pi^{\mathrm{SC}}\right\rangle$, we have
  $$\frac{\left\langle \pi^{\mathrm{SC}} \middle| \mathbf{1} \right\rangle^2}{q^{n-k}} \;=\; \frac{q^k}{q^n} \left| \sum_{\mathbf{y} \in \mathbb{F}_q^n} \chi_{\mathbf{y}}(\mathbf{0}) \pi_{\mathbf{y}} \right|^2 \;=\; q^k \left| \widehat{f}(0) \right|^2 \;=\; q^k (1 - \tau^\perp)^n.$$
  It can be verified that there is no way to choose τ such that at the same time:
  (*i*) $\tau n \leq d_{\mathrm{GV}}(n, k)$ (otherwise there is no hope to decode correctly most of the time),

(*ii*) $\tau^\perp \leq \omega_{\text{easy}}$ (otherwise finding codewords in $\mathcal{C}^\perp$ of weight $\tau^\perp n$ is easy),
(*iii*) $q^k(1-\tau^\perp)^n = o(1)$.

The quantity $\frac{\langle \pi^{\text{SC}} | \mathbf{1} \rangle^2}{q^{n-k}}$ is just too big, or in other words, the distribution of μ is too much spread out and not concentrated enough on its typical weight.

- Condition ($C2$) expresses that u lies in a subset of values for which a random $[n, n-k]$-code has an exponential expected number of codewords. Indeed, the expected number of codewords of weight u is equal to $\frac{S_u}{q^k}$.

- Finally, Condition ($C3$) expresses that the dual probability distribution $\widehat{\mu} \stackrel{\text{def}}{=} (|\widehat{\pi}_{\mathbf{e}}|^2)_{\mathbf{e} \in \mathbb{F}_q^n}$ is almost completely supported on $\mathscr{W}$ up to an exponentially small vanishing term.

1.2 Outline of the proof of Theorem 3.2

Let us first give a general outline of the proof before detailing each step.

Step 1. We prove that after applying $\mathscr{A}$ in the reduction, $|\psi_{\mathscr{A}}\rangle$ is close enough to the "disentangled" state

$$|\psi_{\text{ideal}}\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{Z}} \sum_{\mathbf{e} \in \mathbb{F}_q^n} \sum_{\mathbf{c} \in \mathcal{C}} \sum_{\mathbf{r} \in \mathbb{F}_2^\ell} \pi_{\mathbf{e}} |\mathbf{0}_n\rangle |\mathbf{c} + \mathbf{e}\rangle |\mathbf{r}\rangle \tag{3.7}$$

where Z is a normalizing constant.

Step 2. We then analyze the effect of the QFT on the "ideal state" $|\psi_{\text{ideal}}\rangle$ and a subsequent measurement of it. We namely prove that measuring it produces a codeword $\mathbf{c}^\perp \in \mathcal{C}^\perp$ of weight u with probability $\left|\widehat{f}(u)\right|^2$, up to a normalizing factor.

Step 3. Then we prove that the number of codewords of weight u in $\mathcal{C}^\perp$ is typically very close to $\frac{S_u}{q^k}$. With Step 2 and the assumptions of Theorem 3.2, we infer that the probability of observing a dual codeword of weight in the set $\mathscr{W}$ after measuring $\widehat{|\psi_{\text{ideal}}\rangle}$ is exponentially close to 1.

Step 4. We upper-bound the statistical distance between the probability distribution of the states after measuring $\widehat{|\psi_{\mathscr{A}}\rangle}$ and $\widehat{|\psi_{\text{ideal}}\rangle}$ respectively by using Step 1 and the properties of the trace distance given in Fact 2 below.

Let us give more details about these steps.

Step 1. For this purpose, we use the trace distance between quantum states (as in [Ste+09] where this has been used in the lattice setting). It is defined as follows :

$$D_{\text{tr}}(|\phi\rangle, |\psi\rangle) \stackrel{\text{def}}{=} \sqrt{1 - |\langle \phi | \psi \rangle|^2}. \tag{3.8}$$

This distance meets the following properties that will prove useful in our context:

Fact 2.

(I) *It can never increase after a quantum evolution [NC16, §9, Th. 9.1];*

(II) *The pair of probability distributions (p_m, q_m) of the measurement outcome m of any quantum measurement performed on the pair of states $(|\phi\rangle, |\psi\rangle)$ satisfies [NC16, §9, Th. 9.2]*

$$D_{\text{stat}}(p_m, q_m) \leq D_{\text{tr}}(|\phi\rangle, |\psi\rangle) \tag{3.9}$$

where D_{stat} is the statistical distance (also called the total variation distance) between two probability distributions. It is defined by:

$$D_{\text{stat}}(p, q) \stackrel{def}{=} \frac{1}{2} \sum_{x \in \mathscr{X}} |p(x) - q(x)|$$

where p and q are two discrete probability distributions on $\mathscr{X}$.

With this notion we can prove that

Proposition 3.2. *With probability greater than $1 - \frac{\langle \pi | \mathbf{1} \rangle^2}{q^{n-k}} - O\left(q^{-\min(k, n-k)}\right)$ over the choices of $\mathbf{G}$ we have:*

$$D_{\text{tr}}(|\psi_{\mathscr{A}}\rangle, |\psi_{ideal}\rangle) \leq \sqrt{1 - \frac{p_t^2}{2} \varepsilon_{\mathbf{G}}^2},$$

where $\varepsilon_{\mathbf{G}}$ is the probability that $\mathscr{A}$ returns the right error $\mathbf{e}$ when the input matrix is $\mathbf{G}$, i.e.,

$$\varepsilon_{\mathbf{G}} \stackrel{def}{=} \mathbb{P}_{\mathbf{c},\mathbf{e},\mathbf{r}} \left(\mathscr{A}(\mathbf{G}, \mathbf{c} + \mathbf{e}, \mathbf{r}) = \mathbf{e} \right). \tag{3.10}$$

The proof of this result follows immediately from three lemmas (whose proof is in Section 3.1). The first one bounds the trace distance in terms of $\varepsilon_{\mathbf{G}}$ and Z, and the second one gives a tight upper-bound on the expected value of Z for a related probabilistic model. The latter is used to derive the third one which is fundamental and states that it is very unlikely for Z to be much greater than the "natural" constant $2^\ell q^k$:

Lemma 3.1. *We have:*

$$D_{\text{tr}}(|\psi_{\mathscr{A}}\rangle, |\psi_{ideal}\rangle) \leq \sqrt{1 - \frac{2^\ell q^k p_t^2}{Z} \varepsilon_{\mathbf{G}}^2}.$$

Lemma 3.2. *Assume that $\mathcal{C}$ is chosen by uniformly drawing at random a parity-check matrix $\mathbf{H}$ for it. We have:*

$$\mathbb{E}(Z) \leq 2^\ell q^k \left(1 + \frac{\langle \pi | \mathbf{1} \rangle^2}{q^{n-k}} \right). \tag{3.11}$$

Lemma 3.3. *Let $\eta > 0$. We have:*

$$\mathbb{P}_{\mathbf{G}}(Z > 2^{\ell} q^{k}(1+\eta)) \leq \frac{1}{\eta} \frac{\langle \pi | \mathbf{1} \rangle^2}{q^{n-k}} + O\left(q^{-\min(k,n-k)}\right).$$

Proposition 3.2 immediately follows by using $\eta = 1$ in Lemma 3.3 and plugging this bound on Z in Lemma 3.1. The quantity $2^{\ell} q^{k}$ is the natural value for Z since it is what we can expect when all the $\mathbf{c} + \mathbf{e}$ terms (taking all $\mathbf{c}$ in $\mathcal{C}$ and all typical $\mathbf{e}$) are different. The constant Z increases precisely when there are many collisions for the $\mathbf{c} + \mathbf{e}$ terms. However, in this case, we do not expect to be able to solve the decoding problem anymore.

Step 2. More precisely, we prove that

Lemma 3.4. *If the Fourier transform is radially preserving, meaning that it transforms a radial function into a radial function, then after measuring $|\widehat{\psi_{ideal}}\rangle$ we obtain a state $|\mathbf{0}_n\rangle \left|\mathbf{c}^{\perp}\right\rangle |\mathbf{r}\rangle$ with $\mathbf{c}^{\perp} \in \mathcal{C}^{\perp}$ of weight u with probability $\frac{2^{\ell} q^{2k}}{Z} N_u^{\perp} \left|\widehat{f}(u)\right|^2$ where $\widehat{f}(u) \stackrel{def}{=} \widehat{\pi}_{\mathbf{e}}$ for an arbitrary $\mathbf{e}$ of weight u and $N_u^{\perp}$ is the number of codewords of weight u in $\mathcal{C}^{\perp}$.*

The proof is given in Section 3.2.

Step 3. This step consists in quantifying how close to 1 the probability of observing a dual codeword of weight in the set $\mathscr{W}$ after measuring $|\widehat{\psi_{\text{ideal}}}\rangle$ is. More specifically, we have

Proposition 3.3. *Under the assumptions made in Theorem 3.2, the probability of obtaining a codeword $\mathbf{c}^{\perp} \in \mathcal{C}^{\perp}$ of weight $u \in \mathscr{W}$ when measuring $|\widehat{\psi_{ideal}}\rangle$ is $\geq 1 - \alpha(\pi)$ for a proportion $\geq 1 - \beta(\pi)$ of matrices $\mathbf{G}$, where:*

$$\alpha(\pi) \stackrel{def}{=} \sum_{u \in \mathscr{W}} \left(\frac{q^k}{S_u}\right)^{1/4} + \sqrt{\frac{\langle \pi | \mathbf{1} \rangle^2}{q^{n-k}}} - 2^{-\Omega(n)},$$

$$\beta(\pi) \stackrel{def}{=} (q-1) \sum_{u \in \mathscr{W}} \sqrt{\frac{q^k}{S_u}} + \sqrt{\frac{\langle \pi | \mathbf{1} \rangle^2}{q^{n-k}}} + O\left(q^{-\min(k,n-k)}\right).$$

This proposition is proved in Section 3.3.

Step 4. We first prove the following point

Lemma 3.5. *Call $\mathscr{G}$ the set of "good matrices" $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ that satisfy at the same time:*

(i) $\varepsilon_{\mathbf{G}} \geq \varepsilon/2$ *(where ε and $\varepsilon_{\mathbf{G}}$ are defined in Equation 3.3 and Equation 3.10),*

(ii) $Z \leq 2^{\ell+1} q^{k}$.

The proportion of good matrices is at least $\varepsilon/2 - \delta(\pi)$ *where* $\delta(\pi) \stackrel{def}{=} \frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}} + O\left(q^{-\min(k,n-k)}\right)$.

Proof. By definition,

$$\varepsilon = \frac{1}{q^{kn}} \sum_{\mathbf{G}\in\mathbb{F}_q^{k\times n}} \varepsilon_{\mathbf{G}}.$$

Let $\mathscr{B}$ be the set of matrices $\mathbf{G}$ that are not good, namely for which (a) $\varepsilon_{\mathbf{G}} < \varepsilon/2$ or (b) $Z > 2^{\ell+1}q^k$. By Lemma 3.3, the density of matrices verifying (b) is smaller than $\delta(\pi)$. Therefore,

$$\varepsilon \leq \frac{1}{q^{kn}} \sum_{\mathbf{G}\notin\mathscr{B}} 1 + \delta(\pi)\,\frac{\varepsilon}{2} \leq \frac{1}{q^{kn}} \sum_{\mathbf{G}\notin\mathscr{B}} 1 + \delta(\pi) + \frac{\varepsilon}{2}$$

which concludes the proof. □

We use this lemma to prove that the statistical distance between the weight distributions obtained by measuring $\widehat{|\psi_{\mathscr{A}}\rangle}$ and $\widehat{|\psi_{\text{ideal}}\rangle}$ cannot be too far away:

Lemma 3.6. *Let* P*, respectively* Q*, be the distribution of the weights* $\left|\mathbf{c}^\perp\right|$ *of the state* $|\mathbf{e}\rangle\left|\mathbf{c}^\perp\right\rangle$ *obtained by measuring the state* $\widehat{|\psi_{\mathscr{A}}\rangle}$*, respectively* $\widehat{|\psi_{ideal}\rangle}$*. We have*

$$D_{\text{stat}}(P,Q) \leq 1 - \frac{p_t^2\varepsilon^3}{16} + O\left(p_t^4\varepsilon^5\right) + \delta(\pi).$$

Proof. Let,

$$P_{\mathbf{G}}(u) \stackrel{\text{def}}{=} \mathbb{P}_{\mathbf{c},\mathbf{e}}\left(\text{measuring } \left|\mathbf{c}^\perp\right\rangle \text{ of weight } u \text{ in the 2nd register of } \widehat{|\psi_{\mathscr{A}}\rangle} \text{ for a code choice } \mathbf{G}\right)$$
$$Q_{\mathbf{G}}(u) \stackrel{\text{def}}{=} \mathbb{P}_{\mathbf{c},\mathbf{e}}\left(\text{measuring } \left|\mathbf{c}^\perp\right\rangle \text{ of weight } u \text{ in the 2nd register of } \widehat{|\psi_{\text{ideal}}\rangle} \text{ for a code choice } \mathbf{G}\right)$$

We start the proof by noticing that

$$\begin{aligned}
D_{\text{stat}}(P,Q) &= \frac{1}{2}\sum_u |P(u)-Q(u)| = \frac{1}{2}\sum_u \left|\sum_{\mathbf{G}\in\mathbb{F}_q^{k\times n}} \frac{1}{q^{kn}}\left(P_{\mathbf{G}}(u)-Q_{\mathbf{G}}(u)\right)\right| \\
&\leq \frac{1}{q^{kn}}\sum_{\mathbf{G}\in\mathbb{F}_q^{k\times n}} \frac{1}{2}\sum_u |P_{\mathbf{G}}(u)-Q_{\mathbf{G}}(u)| \\
&= \frac{1}{q^{kn}}\sum_{\mathbf{G}\in\mathbb{F}_q^{k\times n}} D_{\text{stat}}\left(P_{\mathbf{G}},Q_{\mathbf{G}}\right) \\
&= \sum_{\mathbf{G}\in\mathscr{G}} \frac{D_{\text{stat}}\left(P_{\mathbf{G}},Q_{\mathbf{G}}\right)}{q^{kn}} + \sum_{\mathbf{G}\notin\mathscr{G}} \frac{D_{\text{stat}}\left(P_{\mathbf{G}},Q_{\mathbf{G}}\right)}{q^{kn}} \\
&\leq \sum_{\mathbf{G}\in\mathscr{G}} \frac{D_{\text{tr}}\left(|\psi_{\mathscr{A}}\rangle,|\psi_{\text{ideal}}\rangle\right)}{q^{kn}} + \sum_{\mathbf{G}\notin\mathscr{G}} \frac{1}{q^{kn}} \qquad \text{(by Equation 3.9)} \\
&\leq \sum_{\mathbf{G}\in\mathscr{G}} \frac{\sqrt{1-\frac{p_t^2\varepsilon^2}{4}}}{q^{kn}} + \sum_{\mathbf{G}\notin\mathscr{G}} \frac{1}{q^{kn}} \qquad \text{(by Proposition 3.2)} \\
&\leq \sqrt{1-\frac{p_t^2\varepsilon^2}{4}}\left(\varepsilon/2-\delta(\pi)\right)+1-\varepsilon/2+\delta(\pi) \qquad \text{(by Lemma 3.5)} \\
&\leq \left(\varepsilon/2-\delta(\pi)\right)\left(1-\frac{p_t^2\varepsilon^2}{8}+O\left(p_t^4\varepsilon^4\right)\right)+1-\varepsilon/2+\delta(\pi) \\
&\leq 1-\frac{p_t^2\varepsilon^3}{16}+O\left(p_t^4\varepsilon^5\right)+\delta(\pi)
\end{aligned}$$

which concludes the proof. □

1.3 Proof of Theorem 3.2

We are now ready to prove Theorem 3.2. By Proposition 3.3 we know that

$$\sum_{u\in\mathscr{W}} Q(u) \geq (1-\alpha(\pi))(1-\beta(\pi)) \geq 1-\alpha(\pi)-\beta(\pi).$$

But now we have the following computation,

$$\begin{aligned}
\sum_{u\in\mathscr{W}} P(u) &\geq \sum_{u\in\mathscr{W}} Q(u) - D_{\text{stat}}(P,Q) \\
&\geq 1-\alpha(\pi)-\beta(\pi)-1+\frac{p_t^2\varepsilon^3}{16}-O\left(p_t^4\varepsilon^5\right)-\delta(\pi) \\
&= \frac{p_t^2\varepsilon^3}{16}-O\left(p_t^4\varepsilon^5\right)-\alpha(\pi)-\beta(\pi)-\delta(\pi)
\end{aligned}$$

which concludes the proof by definition of $\alpha(\pi)$, $\beta(\pi)$ and $\delta(\pi)$.

1.4 Application to the Hamming metric

The assumptions of Theorem 3.2 will be satisfied for the Hamming metric for weights u close to $\tau^{\perp} n$ (where $\tau^{\perp}$ is given in Equation 3.2) and we will prove that

Theorem 3.3. *Suppose that there exists an algorithm $\mathscr{A}$ solving with success probability ε the decoding problem $\mathsf{DP}(q,n,k,t)$ at Hamming distance $1 \leq t \overset{def}{=} \tau n \leq (1-\delta) d_{\mathrm{GV}}(n,k)$ for any arbitrary $\delta > 0$. Then, there exists a quantum algorithm which takes as input a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ of a code $\mathcal{C} \subseteq \mathbb{F}_q^n$ and outputs $\mathbf{c}^{\perp} \in \mathcal{C}^{\perp}$ of weight $u \in [\![(1-\alpha)\tau^{\perp} n, (1+\alpha)\tau^{\perp} n]\!]$ (where α is any arbitrary constant > 0) with probability over a uniform choice of $\mathbf{G}$ given by a $\Omega\left(\frac{\varepsilon^3}{n} - 2^{-\Omega(n)}\right)$ where:*

$$\tau^{\perp} \overset{def}{=} \frac{1}{q}\left(\sqrt{(q-1)(1-\tau)} - \sqrt{\tau}\right)^2. \tag{3.12}$$

The proof of this theorem relies on Theorem 3.2, for a suitable choice of quantum state $|\pi\rangle$. This is done by choosing $|\pi\rangle = \left|\pi^{\mathrm{Trunc}}\right\rangle$ which represents a truncated q-ary symmetric channel of crossover probability τ. All its weights are in an interval $[\![(1-\eta)t, (1+\eta)t]\!]$ where η is some positive constant which will be chosen later on. More precisely, let us first define the (untruncated) quantum state representing the q-ary symmetric channel of crossover probability τ:

$$\left|\pi^{\mathrm{SC}}\right\rangle \overset{\mathrm{def}}{=} \left(\sqrt{1-\tau}\,|0\rangle + \sqrt{\tau/(q-1)} \sum_{\alpha \in \mathbb{F}_q^*} |\alpha\rangle\right)^{\otimes n}.$$

Indeed, since $\left|\pi^{\mathrm{SC}}\right\rangle$ can also be written as

$$\left|\pi^{\mathrm{SC}}\right\rangle = \sum_{\mathbf{e} \in \mathbb{F}_q^n} \pi_{\mathbf{e}} |\mathbf{e}\rangle \quad \text{with} \quad \pi_{\mathbf{e}} \overset{\mathrm{def}}{=} \sqrt{(1-\tau)^{n-|\mathbf{e}|}\left(\frac{\tau}{q-1}\right)^{|\mathbf{e}|}},$$

measuring $\left|\pi^{\mathrm{SC}}\right\rangle$ mimics the error we have in a q-ary symmetric channel of crossover probability τ, *i.e.*,

$$\mathbb{P}(\text{measurement outputs } \mathbf{e}) = |\pi_{\mathbf{e}}|^2 = \left(\frac{\tau}{q-1}\right)^{|\mathbf{e}|} (1-\tau)^{n-|\mathbf{e}|}.$$

We will be interested in the truncated version given by

$$\left|\pi^{\mathrm{Trunc}}\right\rangle \overset{\mathrm{def}}{=} \sum_{\substack{\mathbf{e} \in \mathbb{F}_q^n : \\ |\mathbf{e}| \in [\![(1-\eta)t, (1+\eta)t]\!]}} \pi_{\mathbf{e}}^{\mathrm{Trunc}} |\mathbf{e}\rangle \quad \text{with} \quad \pi_{\mathbf{e}}^{\mathrm{Trunc}} \overset{\mathrm{def}}{=} \begin{cases} \frac{\pi_{\mathbf{e}}}{\sqrt{N}} & \text{if } |\mathbf{e}| \in [\![(1-\eta)t, (1+\eta)t]\!] \\ 0 & \text{otherwise} \end{cases}$$

where N is the normalizing constant given by

$$N \stackrel{\text{def}}{=} \sum_{\substack{\mathbf{e}\in\mathbb{F}_q^n \\ |\mathbf{e}|\in[\![(1-\eta)t,(1+\eta)t]\!]}} |\pi_{\mathbf{e}}|^2. \tag{3.13}$$

It will be helpful to notice that for all $\eta > 0$, N is exponentially close to 1:

Lemma 3.7. *For all $\eta > 0$, we have*

$$N = 1 - 2^{-\Omega(n)}.$$

Proof. Notice that by Equation 3.13,

$$\begin{aligned} 1 - N = \sum_{\substack{\mathbf{e}\in\mathbb{F}_q^n \\ |\mathbf{e}|\notin[\![(1-\eta)t,(1+\eta)t]\!]}} |\pi_{\mathbf{e}}|^2 &= \mathbb{P}_{\mathbf{e}}(|\mathbf{e}| \notin [\![(1-\eta)t, (1+\eta)t]\!]) \\ &= \mathbb{P}_{\mathbf{e}}(|\mathbf{e}| < (1-\eta)\tau n) \;+\; \mathbb{P}_{\mathbf{e}}(|\mathbf{e}| > (1+\eta)\tau n) \end{aligned} \tag{3.14}$$

where $|\mathbf{e}|$ is the sum of n independent (binary) Bernoulli random variables of parameter τ. Therefore, by Hoeffding's inequality, we have for all $\eta > 0$,

$$\mathbb{P}_{\mathbf{e}}(|\mathbf{e}| < (1-\eta)\tau n) \le e^{-2\eta^2\tau^2 n} \quad \text{and} \quad \mathbb{P}_{\mathbf{e}}(|\mathbf{e}| > (1+\eta)\tau n) \le e^{-2\eta^2\tau^2 n}$$

which concludes the proof by plugging this in Equation 3.14. □

Theorem 3.3 is proved by showing that $\left|\pi^{\text{Trunc}}\right\rangle$ satisfies all the requirements of Theorem 3.2 when η is small enough.

Step 1: Verification of Condition $(C1)$.

This amounts to proving the following lemma

Lemma 3.8. *For $\eta > 0$ small enough, we have,*

$$\frac{\left\langle \pi^{Trunc} \middle| \mathbf{1} \right\rangle^2}{q^{n-k}} = 2^{-\Omega(n)}.$$

Before proving this result, it will be helpful to notice that:

Lemma 3.9. *If $u \le (1-\delta)d_{\text{GV}}(n,k)$ for some $\delta > 0$, then*

$$\frac{S_u}{q^{n-k}} = 2^{-\Omega(n)}.$$

Proof. Recall that the size B_u of the Hamming ball of radius u is of the form

$$B_u = q^{n\, h_q(\mu)(1+o(1))}$$

where $\mu \stackrel{\text{def}}{=} u/n$. From this we obtain

$$\begin{aligned}\frac{S_u}{q^{n-k}} &\leq \frac{B_u}{B_{d_{\mathrm{GV}}}} \qquad (\text{since } S_u \leq B_u \text{ and } B_{d_{\mathrm{GV}}} \leq q^{n-k}) \\ &\leq q^{n(h_q(\mu)-h_q(\delta_{\mathrm{GV}})+o(1))} \\ &\leq q^{n(h_q((1-\delta)\delta_{\mathrm{GV}})-h_q(\delta_{\mathrm{GV}})+o(1))}.\end{aligned}$$

We finish the proof by noticing that $h_q((1-\delta)\delta_{\mathrm{GV}}) - h_q(\delta_{\mathrm{GV}}) < 0$. □

We are now ready to prove Lemma 3.8.

Proof of Lemma 3.8. We have the following computation,

$$\begin{aligned}\left\langle \pi^{\mathrm{Trunc}} \middle| \mathbf{1} \right\rangle &= \sum_{\substack{\mathbf{e}\in\mathbb{F}_q^n \\ |\mathbf{e}|\in[\![(1-\eta)t,(1+\eta)t]\!]}} \pi_{\mathbf{e}}^{\mathrm{Trunc}} \\ &= \sum_{r\in[\![(1-\eta)t,(1+\eta)t]\!]} \sum_{\substack{\mathbf{e}\in\mathbb{F}_q^n \\ |\mathbf{e}|=r}} \frac{\pi_{\mathbf{e}}}{\sqrt{N}} \\ &= \frac{1}{\sqrt{N}} \sum_{r\in[\![(1-\eta)t,(1+\eta)t]\!]} \sqrt{\binom{n}{r}(q-1)^r} \sqrt{\binom{n}{r}(q-1)^r\,(1-\tau)^{n-r}\left(\frac{\tau}{q-1}\right)^r} \\ &\leq \frac{1}{\sqrt{N}} \sum_{r\in[\![(1-\eta)t,(1+\eta)t]\!]} \sqrt{\binom{n}{r}(q-1)^r} \qquad (3.15)\end{aligned}$$

where in the last line we used that $\binom{n}{u}(q-1)^u \left(\frac{\tau}{q-1}\right)^u (1-\tau)^{n-u} \leq 1$ for any $u \in [\![0,n]\!]$. Therefore, using Equation 3.15, we have:

$$\left\langle \pi^{\mathrm{Trunc}} \middle| \mathbf{1} \right\rangle^2 \leq \frac{1}{N} \left(\sum_{r=\lceil(1-\eta)t\rceil}^{\lfloor(1+\eta)t\rfloor} \sqrt{\binom{n}{r}(q-1)^r} \right) \left(\sum_{r'=\lceil(1-\eta)t\rceil}^{\lfloor(1+\eta)t\rfloor} \sqrt{\binom{n}{r'}(q-1)^{r'}} \right)$$

and since $\sharp\,[\![(1-\eta)t,(1+\eta)t]\!] \leq n+1$,

$$\left\langle \pi^{\mathrm{Trunc}} \middle| \mathbf{1} \right\rangle^2 \leq \frac{(n+1)^2}{N} \max_{r,r'\in[\![(1-\eta)t,(1+\eta)t]\!]} \sqrt{\binom{n}{r}\binom{n}{r'}(q-1)^r(q-1)^{r'}}.$$

The max is reached for $r = r' = (1+\eta)t$, so:

$$\left\langle \pi^{\mathrm{Trunc}} \middle| \mathbf{1} \right\rangle^2 \leq \frac{(n+1)^2}{N} \binom{n}{(1+\eta)t} (q-1)^{(1+\eta)t} = \frac{(n+1)^2}{N}\, S_{(1+\eta)t}.$$

Using this last inequality, Lemma 3.7 and Lemma 3.9, we obtain

$$\frac{\left\langle \pi^{\mathrm{Trunc}} \middle| \mathbf{1} \right\rangle^2}{q^{n-k}} = O\left((n+1)^2 \, \frac{S_{(1+\eta)t}}{q^{n-k}}\right) = 2^{-\Omega(n)}$$

where we choose η small enough such that $(1+\eta)t \leq (1-\delta')d_{\mathrm{GV}}(n,k)$ for some $\delta' > 0$ (recall that by assumption $t \leq (1-\delta)d_{\mathrm{GV}}(n,k)$ for $\delta > 0$). □

Remark 3.2. As explained in the introduction and Remark 3.1, Lemma 3.8 is not satisfied by $\left|\pi^{\mathrm{SC}}\right\rangle$. Here truncating the error distribution is essential to verify this concentration lemma.

Step 2: Verification of Conditions $(C2)$ and $(C3)$.

We prove here that the Conditions $(C2)$ and $(C3)$ of Theorem 3.2 are met by $\left|\pi^{\mathrm{Trunc}}\right\rangle$. This results from a combination of arguments: (i) these two conditions are met for $\left|\pi^{\mathrm{SC}}\right\rangle$ (ii) $\left|\pi^{\mathrm{SC}}\right\rangle$ and $\left|\pi^{\mathrm{Trunc}}\right\rangle$ are very close and so are $|\widehat{\pi^{\mathrm{SC}}}\rangle$ and $|\widehat{\pi^{\mathrm{Trunc}}}\rangle$ (because they are obtained from the first pair by applying a QFT, which is unitary).

More precisely we are going to prove that

Lemma 3.10. *Let* $t^\perp \stackrel{def}{=} \frac{(\sqrt{(q-1)(n-t)}-\sqrt{t})^2}{q}$ *and* $\alpha > 0$ *be some constant small enough. We let* $|\widehat{\pi^{Trunc}}\rangle = \sum_{\mathbf{e}\in\mathbb{F}_q^n} \widehat{\pi_{\mathbf{e}}^{Trunc}} |\mathbf{e}\rangle$. *This state is radial and we let* $\widehat{f^{Trunc}}(w) = \widehat{\pi_{\mathbf{e}}^{Trunc}}$ *for any element* $\mathbf{e}$ *of* $\mathbb{F}_q^n$ *of Hamming weight* w. *We have*

$$p_t = \Omega\left(\frac{1}{\sqrt{n}}\right),$$

$$\forall u \in \left[\!\left[t^\perp(1-\alpha), t^\perp(1+\alpha)\right]\!\right], \quad \frac{q^k}{S_u} = 2^{-\Omega(n)} \quad \textit{and} \sum_{u=\lceil t^\perp(1-\alpha)\rceil}^{\lfloor t^\perp(1+\alpha)\rfloor} S_u \left|\widehat{f^{Trunc}}(u)\right|^2 = 1-2^{-\Omega(n)}.$$

As explained above, to prove this result we will rely on the following lemma

Lemma 3.11. *For all* $\eta > 0$, *we have*

$$\left\| \left|\pi^{Trunc}\right\rangle - \left|\pi^{SC}\right\rangle \right\| = 2^{-\Omega(n)} \quad \textit{and} \quad \left\| \widehat{\left|\pi^{Trunc}\right\rangle} - \widehat{\left|\pi^{SC}\right\rangle} \right\| = 2^{-\Omega(n)} \tag{3.16}$$

Proof. We have the following computation,

$$\begin{aligned}
\left\| \left|\pi^{\mathrm{Trunc}}\right\rangle - \left|\pi^{\mathrm{SC}}\right\rangle \right\|^2 &= \sum_{\mathbf{e}\in\mathbb{F}_q^n} (\pi_{\mathbf{e}} - \pi_{\mathbf{e}}^{\mathrm{Trunc}})^2 \\
&= \sum_{\substack{\mathbf{e}\in\mathbb{F}_q^n:\\ |\mathbf{e}|\notin[\![(1-\eta)t,(1+\eta)t]\!]}} \pi_{\mathbf{e}}^2 + \sum_{\substack{\mathbf{e}\in\mathbb{F}_q^n:\\ |\mathbf{e}|\in[\![(1-\eta)t,(1+\eta)t]\!]}} \left(\pi_{\mathbf{e}} - \frac{\pi_{\mathbf{e}}}{\sqrt{N}}\right)^2 \\
&= 1 - N + \frac{(1-\sqrt{N})^2}{N} \sum_{\substack{\mathbf{e}\in\mathbb{F}_q^n\\ |\mathbf{e}|\in[\![(1-\eta)t,(1+\eta)t]\!]}} \pi_{\mathbf{e}}^2 \quad \text{(by Equation 3.13)} \\
&= 1 - N + (1-\sqrt{N})^2 \quad \text{(by Equation 3.13)} \\
&\le 2^{-\Omega(n)} \quad \text{(by Lemma 3.7).}
\end{aligned}$$

The second relation follows, since the QFT is an isometry with respect to $\|\cdot\|$. □

With this lemma at hand, we are ready to prove Lemma 3.10.

Proof of Lemma 3.10. By definition,

$$p_t = \frac{1}{N} \sum_{\mathbf{e}:\ |\mathbf{e}|=t} (1-\tau)^{n-t} \left(\frac{\tau}{q-1}\right)^t = \frac{\binom{n}{t}(q-1)^t\, q^{nh_q(\tau)}}{N} = \Omega\left(\frac{1}{\sqrt{n}}\right)$$

where in the last equality we used Lemma 3.7 and Stirling's formula.
The equality $\frac{q^k}{S_u} = 2^{-\Omega(n)}$ is verified when u is sufficiently close to $t^\perp$ (α small enough), because it is readily verified that there exists some constant $\beta > 0$ such that $t^\perp \ge (1+\beta)d_{\mathrm{GV}}(n, n-k)$. This together with $t^\perp \le \frac{(q-1)n}{q}$ implies that $\frac{q^k}{S_u} = 2^{-\Omega(n)}$ for any u in $[\![t^\perp(1-\alpha), t^\perp(1+\alpha)]\!]$.
The untruncated distribution $\left|\pi^{\mathrm{SC}}\right\rangle = \sum_{\mathbf{e}} \pi_{\mathbf{e}} |\mathbf{e}\rangle$ is radial, and so is its Fourier transform $\widehat{|\pi^{\mathrm{SC}}\rangle} = \sum_{\mathbf{e}} \widehat{\pi}_{\mathbf{e}} |\mathbf{e}\rangle$. We let $\widehat{f}(u) = \widehat{\pi}_{\mathbf{e}}$ where $\mathbf{e}$ is any $\mathbf{e} \in \mathbb{F}_q^n$ of Hamming weight u. We notice that

$$\sum_{u=\lceil t^\perp(1-\alpha)\rceil}^{\lfloor t^\perp(1+\alpha)\rfloor} S_u \left|\widehat{f}(u)\right|^2 = \sum_{\mathbf{e}\in\mathbb{F}_q^n:\ |\mathbf{e}|\in\left[(1-\alpha)t^\perp,(1+\alpha)t^\perp\right]} |\widehat{\pi}_{\mathbf{e}}|^2 = \mathbb{P}_{\mathbf{e}}\left(|\mathbf{e}| \in \left[(1-\alpha)t^\perp n, (1+\alpha)t^\perp n\right]\right),$$

where $\tau^\perp = \frac{t^\perp}{n}$ and $|\mathbf{e}|$ is the sum of n independent (binary) Bernoulli random variables of parameter $\tau^\perp$. Therefore, by using Hoeffding's bound again we obtain that

$$\sum_{u\in[(1-\alpha)t^\perp,(1+\alpha)t^\perp]} S_u \left|\widehat{f}(u)\right|^2 = 1 - 2^{-\Omega(n)}. \tag{3.17}$$

meaning that $\widehat{f}$ concentrates around vectors of weight $t^\perp$. Consider the projection of $\widehat{|\pi^{\mathrm{SC}}\rangle}$ and $\widehat{|\pi^{\mathrm{Trunc}}\rangle}$ on the space spanned by the states $|\mathbf{e}\rangle$ for $|\mathbf{e}| \in [\![(1-\alpha)t^\perp, (1+\alpha)t^\perp]\!]$:

$$\widetilde{|\pi^{\mathrm{SC}}\rangle} \stackrel{\mathrm{def}}{=} \sum_{\mathbf{e}\in\mathbb{F}_q^n \,:\, |\mathbf{e}|\in[(1-\alpha)t^\perp,(1+\alpha)t^\perp]} \widehat{\pi}_{\mathbf{e}} \,|\mathbf{e}\rangle$$

$$\widetilde{|\pi^{\mathrm{Trunc}}\rangle} \stackrel{\mathrm{def}}{=} \sum_{\mathbf{e}\in\mathbb{F}_q^n \,:\, |\mathbf{e}|\in[(1-\alpha)t^\perp,(1+\alpha)t^\perp]} \widehat{\pi^{\mathrm{Trunc}}_{\mathbf{e}}} \,|\mathbf{e}\rangle$$

Since a projection can only reduce the norm, we have

$$\left\| \widetilde{|\pi^{\mathrm{SC}}\rangle} - \widetilde{|\pi^{\mathrm{Trunc}}\rangle} \right\| \leq \left\| \widehat{|\pi^{\mathrm{SC}}\rangle} - \widehat{|\pi^{\mathrm{Trunc}}\rangle} \right\| = 2^{-\Omega(n)}. \tag{3.18}$$

We deduce from the triangle inequality that

$$\left\| \widetilde{|\pi^{\mathrm{Trunc}}\rangle} \right\| \geq \left\| \widetilde{|\pi^{\mathrm{SC}}\rangle} \right\| - \left\| \widetilde{|\pi^{\mathrm{SC}}\rangle} - \widetilde{|\pi^{\mathrm{Trunc}}\rangle} \right\|, \tag{3.19}$$

and then from Equation 3.18 and Equation 3.17 $\left(\text{which says } \left\| \widetilde{|\pi^{\mathrm{SC}}\rangle} \right\|^2 = 1 - 2^{-\Omega(n)}\right)$ that $\left\| \widetilde{|\pi^{\mathrm{Trunc}}\rangle} \right\| \geq 1 - 2^{-\Omega(n)}$. Since $\left\| \widetilde{|\pi^{\mathrm{Trunc}}\rangle} \right\| \leq \left\| \widehat{|\pi^{\mathrm{Trunc}}\rangle} \right\| = 1$ we finally obtain

$$\left\| \widetilde{|\pi^{\mathrm{Trunc}}\rangle} \right\| = 1 - 2^{-\Omega(n)}.$$

This directly implies $\sum_{u\in[\![(1-\alpha)t^\perp,(1+\alpha)t^\perp]\!]} S_u \left| \widehat{f^{\mathrm{Trunc}}}(u) \right|^2 = \left\| \widetilde{|\pi^{\mathrm{Trunc}}\rangle} \right\|^2 = 1 - 2^{-\Omega(n)}$. □

Proof of Theorem 3.3.

This immediately follows from Lemma 3.8 and Lemma 3.10 which show that the relevant assumptions of Theorem 3.2 are verified for the choice $|\pi\rangle = \left|\pi^{\mathrm{Trunc}}\right\rangle$ for η small enough.

1.5 Application to the rank metric

The assumptions of Theorem 3.2 will also be satisfied in the context of codes $\mathcal{C} \subseteq \mathbb{F}_q^{m\times n}$ embedded with the rank metric (given some matrix in $\mathbb{F}_q^{m\times n}$, its weight is defined as its rank). The Gilbert-Varshamov distance $d_{\mathrm{GV}}(m,n,k)$ is defined in a similar way, but it depends on three parameters here and corresponds to the largest radius t of a ball in the rank metric for which

$$q^k B_t \leq q^{m\times n}.$$

We will be able to prove that

Theorem 3.4. *Suppose that there exists an algorithm $\mathscr{A}$ solving with success probability ε the decoding problem at rank distance $1 \leq t < d_{\mathrm{GV}}(m,n,k)$ where $m \geq n$. Then, there exists a quantum algorithm which takes as input a generator matrix of $\mathcal{C} \subseteq \mathbb{F}_q^{m\times n}$ and outputs $\mathbf{c}^\perp \in \mathcal{C}^\perp$ of weight $u \in (t^\perp - \eta n, t^\perp]$ where $t^\perp \overset{def}{=} n - t$ (for any arbitrary constant $\eta > 0$) with probability over a uniform choice of the generator matrix given by a $\Omega\left(\varepsilon^3 - 2^{-\Omega(n)}\right)$.*

Remark 3.3. Our assumption that $m \geq n$ can be done without loss of generality. In the case where $n > m$ we can just consider the transposed code $\mathcal{C}^\intercal \overset{\text{def}}{=} \{\mathbf{M}^\intercal : \mathbf{M} \in \mathcal{C}\}$: taking the transpose is a linear automorphism and can be used to transform any algorithm decoding $\mathcal{C}$ into an algorithm decoding $\mathcal{C}^\intercal$ with the same complexity.

As in the Hamming case, Theorem 3.4 will be a consequence of Theorem 3.2. Therefore we first have to choose appropriately a quantum state $|\pi\rangle$ that will model the noise distribution.

Step 1 : Choosing $|\pi\rangle$.

Let,

$$|\pi\rangle \overset{\text{def}}{=} \frac{1}{\sqrt{N}} \sum_{\substack{V \leq \mathbb{F}_q^n \\ \dim V = t}} |\pi_V\rangle \quad \text{where} \tag{3.20}$$

$$|\pi_U\rangle \overset{\text{def}}{=} \left(\frac{1}{\sqrt{q^{\dim U}}} \sum_{\mathbf{u}\in U} |\mathbf{u}\rangle \right)^{\otimes m} \tag{3.21}$$

and N is a normalizing constant. $|\pi_U\rangle$ can be viewed as a uniform superposition of matrices whose rows belong to U. These matrices have rank at most $\dim U$ and $|\pi\rangle$ is close to a uniform distribution of all matrices of rank t. We also have the following alternative description for $|\pi\rangle$.

Lemma 3.12. *$|\pi\rangle$ is radial, i.e., we may write $|\pi\rangle$ as*

$$|\pi\rangle = \sum_{\mathbf{E}\in\mathbb{F}_q^{m\times n} :\, |\mathbf{E}|\leq t} \pi_{\mathbf{E}} |\mathbf{E}\rangle$$

with $|\mathbf{E}\rangle \overset{def}{=} |\mathbf{E}_1\rangle \otimes \cdots \otimes |\mathbf{E}_m\rangle$ where the $\mathbf{E}_i$'s denote the rows of $\mathbf{E}$ and $\pi_{\mathbf{E}} = f(|\mathbf{E}|)$ where

$$f(u) = \begin{cases} \frac{\begin{bmatrix} n-u \\ t-u \end{bmatrix}_q}{\sqrt{q^{mt}N}} & \text{if } u \leq t, \\ 0 & \text{otherwise.} \end{cases}$$

This lemma is proved in Section 4.1, as well as the following one, which gives estimations for N and p_t in order to apply Theorem 3.2.

Lemma 3.13. *We have:*

$$N = \Theta\left(\begin{bmatrix} n \\ t \end{bmatrix}_q\right) \quad \text{and} \quad p_t = \Theta(1).$$

Step 2: Verification that $\widehat{|\pi\rangle}$ is radial.

The following proposition states that $\widehat{|\pi\rangle}$ has actually the same form as $|\pi\rangle$ where t is replaced by $n-t$:

Proposition 3.4. *We have,*

$$\widehat{|\pi\rangle} = \frac{1}{\sqrt{N}} \sum_{\substack{W \leq \mathbb{F}_q^n \\ \dim W = n-t}} |\pi_W\rangle.$$

Proof. We apply the QFT on $|\pi\rangle$, defined in Equation 3.20. It gives,

$$\widehat{|\pi\rangle} = \frac{1}{\sqrt{N}} \sum_{\substack{V \leq \mathbb{F}_q^n \\ \dim V = t}} \left(\frac{1}{\sqrt{q^{n+t}}} \sum_{\mathbf{y} \in \mathbb{F}_q^n} \left(\sum_{\mathbf{v} \in V} \chi_{\mathbf{y}}(\mathbf{v}) \right) |\mathbf{y}\rangle \right)^{\otimes m}$$

By distinguishing the cases where $\mathbf{y} \in V^\perp$ (the dual of V with the standard inner product) or not:

$$\widehat{|\pi\rangle} = \frac{1}{\sqrt{N}} \sum_{\substack{V \leq \mathbb{F}_q^n \\ \dim V = t}} \left(\frac{1}{\sqrt{q^{n+t}}} \sum_{\mathbf{y} \in V^\perp} q^t |\mathbf{y}\rangle \right)^{\otimes m} = \frac{1}{\sqrt{N}} \sum_{\substack{W \leq \mathbb{F}_q^n \\ \dim W = n-t}} \left(\frac{1}{\sqrt{q^{n-t}}} \sum_{\mathbf{y} \in W} |\mathbf{y}\rangle \right)^{\otimes m}$$

which concludes the proof. □

We can now straightforwardly apply Lemma 3.12 on $\widehat{|\pi\rangle}$ and obtain

Lemma 3.14. *The state $\widehat{|\pi\rangle}$ is radial and can be written as $\sum_{\mathbf{E} \in \mathbb{F}_q^{m \times n} : |\mathbf{E}| \leq n-t} \widehat{\pi_{\mathbf{E}}} |\mathbf{E}\rangle$. If we let $\widehat{f}(u) \stackrel{def}{=} \widehat{\pi_{\mathbf{E}}}$ for any $\mathbf{E} \in \mathbb{F}_q^{m \times n}$ of rank u, we have*

$$\widehat{f}(u) = \begin{cases} \frac{\begin{bmatrix} n-u \\ n-t-u \end{bmatrix}_q}{\sqrt{q^{m(n-t)} N}} & \text{if } u \leq n-t, \\ 0 & \text{otherwise.} \end{cases}$$

Step 3: Verification of Conditions $(C1)$, $(C2)$ and $(C3)$.

This is achieved in the following lemmas that are proved in Section 4.2.

Lemma 3.15. *We have,*

$$\frac{S_t}{q^{mn-k}} = q^{-\Omega(n)} \quad \text{and} \quad \frac{\langle \pi | \mathbf{1} \rangle^2}{q^{mn-k}} = q^{-\Omega(n)}.$$

Lemma 3.16. *For any $\eta > 0$, we have,*

$$\forall u \in [\![(1-\eta)n - t, n-t]\!], \ \frac{q^k}{S_u} = q^{-\Omega(n)} \quad \text{and} \quad \sum_{u \in [\![(1-\eta)n-t, n-t]\!]} S_u \left| \widehat{f}(u) \right|^2 = 1 - q^{-\Omega(n)}.$$

Proof of Theorem 3.4.

This follows from Lemma 3.12, Lemma 3.14, Lemma 3.15 and Lemma 3.16 that allow to apply Theorem 3.2, completing the proof.

2 About the usefulness of our reduction.

It is now interesting to look at the parameters for which our reduction is useful for both the Hamming and rank metrics.

2.1 Hamming case

A lower-bound on τ is obtained with the following arguments. First, if one wants to compute dual codewords (via the quantum measure) for which no poly-time algorithm is known, one has to ensure that $\tau^\perp < \omega_{\text{easy}}(n, n-k) = \frac{q-1}{q}\frac{k}{n}$. But notice that $\tau \mapsto \tau^\perp$ is a decreasing involution on $\left[0, \frac{1}{2}\right]$. Therefore, for the reduction to be meaningful, it is necessary that

$$\begin{aligned} \tau > \omega_{\text{easy}}(n, n-k)^\perp = \frac{1}{q}\left(\sqrt{(q-1)\left(1 - \frac{q-1}{q}\frac{k}{n}\right)} - \sqrt{\frac{q-1}{q}\frac{k}{n}}\right)^2 \\ = \frac{q-1}{q^2}\left(\sqrt{q - (q-1)\frac{k}{n}} - \sqrt{\frac{k}{n}}\right)^2 \end{aligned} \tag{3.22}$$

Furthermore, according to Theorem 3.3, the relative decoding distance τ has to verify

$$\tau < \frac{d_{\text{GV}}(n,k)}{n} = h_q^{-1}\left(1 - \frac{k}{n}\right) + O\left(\frac{1}{n}\right) = \delta_{\text{GV}}(n,k) + O\left(\frac{1}{n}\right).$$

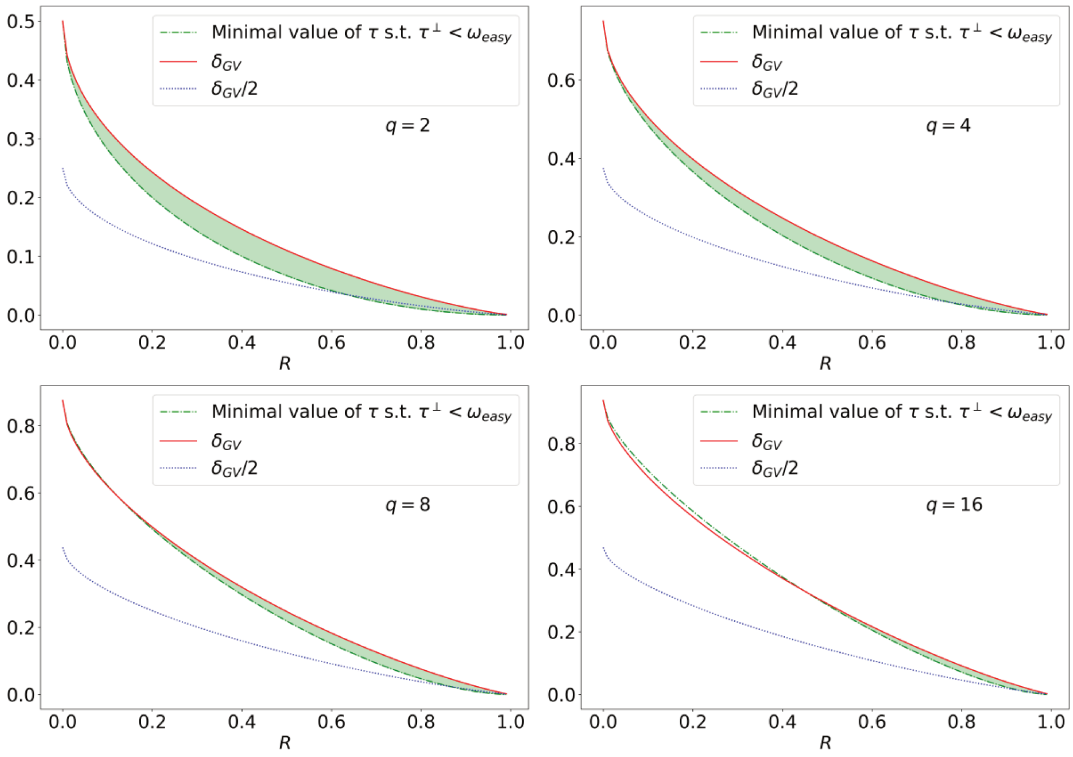

Figure 3.1: Range of values for τ as function of R.

Roughly speaking, it gives the tightest upper-bound for which we can expect to correctly decode with an overwhelming probability. Combining this with Equation 3.22 leads to a whole interval in which τ needs to lie for the reduction to work and be meaningful.

In Figure 3.1, we draw (asymptotically in n) this range of values of τ as function of $R \stackrel{\text{def}}{=} \frac{k}{n}$ for different values of q (the green area).

Two important remarks can be made from these graphs:

1. the range of interesting values for τ (*i.e.*, values such that we solve a hard instance of SCP) shrinks as q grows and depending on R;

2. the lower bound on τ corresponding to $\tau^\perp < \omega_{\text{easy}}(n, n-k)$ is almost always above $\delta_{\text{GV}}(n,k)/2$, meaning that in order to solve a hard instance of SCP, it is a necessity that τ goes beyond the unique decoding radius.

2.2 Rank Case

It turns out that unfortunately, for the rank metric (which is coarser than the Hamming metric), we always reduce decoding t errors to finding dual codewords of weight $t^\perp = n-t$

where $t^\perp$ belongs to a range of values for which it is always easy to find codewords of this weight as we now show.

To verify this point, consider a linear code $\mathcal{C} \subseteq \mathbb{F}_q^{m\times n}$ of dimension K (with $m \geq n$). It is easy to find short codewords if they are above a certain range. To produce codewords of small weight, we use the fact that the dual code is a vector space of dimension $nm - K$. Thus, we can just produce codewords with $nm - K - 1$ entries equal to 0 that will be good candidates for having a small weight by solving a linear system. The entries are chosen so as to fill columns with zeroes. It is straightforward that this strategy produces in polynomial time codewords of weight $\approx Rn$ (since in our case $n \leq m$) where R is the rate of $\mathcal{C}$ defined by $R \stackrel{\text{def}}{=} \frac{K}{mn}$.

Notice now that $t^\perp$ is a decreasing function of the decoding distance t. The largest value for which we can hope to decode is the Gilbert-Varshamov distance $d_{\text{GV}}(m, n, K)$. The relative Gilbert-Varshamov distance $\delta_{\text{GV}}(m, n, K) \stackrel{\text{def}}{=} \frac{d_{\text{GV}}(m,n,K)}{n}$ satisfies the relation

$$R = 1 - \delta_{\text{GV}}(1 + \nu - \delta_{\text{GV}})$$

where $\nu \stackrel{\text{def}}{=} \frac{m}{n} \geq 1$. However, we have (where $\delta_{\text{GV}}^\perp$ is defined as $t^\perp/n$ when $t/n = \delta_{\text{GV}}(m, n, K)$)

$$\frac{t^\perp}{n} \geq \delta_{\text{GV}}^\perp = 1 - \delta_{\text{GV}} = R + \delta_{\text{GV}}(\nu - \delta_{\text{GV}}) \geq R.$$

In other words, we are always in a regime where finding codewords of relative weight $t^\perp/n$ is easy.

3 Proof of Theorem 3.2

3.1 Step 1: Proof of Lemma 3.1 and Lemma 3.3

Let us recall Lemma 3.1 first:

Lemma 3.1. *We have:*

$$D_{\text{tr}}(|\psi_{\mathscr{A}}\rangle, |\psi_{ideal}\rangle) \leq \sqrt{1 - \frac{2^\ell q^k p_t^2}{Z}\varepsilon_{\mathbf{G}}^2}.$$

Proof. Let $\mathscr{G}$ be the set of $(\mathbf{c}, \mathbf{e})$'s that correspond to inputs of weight t to $\mathscr{A}$ that are correctly decoded:

$$\mathscr{G} \stackrel{\text{def}}{=} \left\{(\mathbf{c}, \mathbf{e}, \mathbf{r}) \in \mathcal{C} \times \mathscr{S}_t \times \mathbb{F}_2^\ell \colon \mathscr{A}(\mathbf{G}, \mathbf{c} + \mathbf{e}, \mathbf{r}) = \mathbf{e}\right\}.$$

Let us recall that

$$|\psi_{\mathscr{A}}\rangle = \frac{1}{\sqrt{2^\ell q^k}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{r}\in\mathbb{F}_2^\ell} \pi_{\mathbf{e}} |\mathbf{e} - \mathscr{A}(\mathbf{G}, \mathbf{c} + \mathbf{e})\rangle |\mathbf{c} + \mathbf{e}\rangle |\mathbf{r}\rangle$$

and

$$|\psi_{\text{ideal}}\rangle = \frac{1}{\sqrt{Z}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \sum_{\mathbf{c}\in\mathcal{C}} \sum_{\mathbf{r}\in\mathbb{F}_2^\ell} \pi_{\mathbf{e}} |\mathbf{0}_n\rangle |\mathbf{c}+\mathbf{e}\rangle |\mathbf{r}\rangle$$

From this we deduce by using the non-negativity of $\pi_{\mathbf{e}}$ that

$$\begin{aligned}\langle\psi_{\mathscr{A}}|\psi_{\text{ideal}}\rangle &\geq \frac{1}{\sqrt{2^\ell q^k Z}} \sum_{(\mathbf{c},\mathbf{e},\mathbf{r})\in\mathscr{G}} \pi_{\mathbf{e}}^2 \\ &= \sqrt{\frac{2^\ell q^k}{Z}} S_t f(t)^2 \frac{\sharp\mathscr{G}}{2^\ell q^k S_t} \\ &= \sqrt{\frac{2^\ell q^k}{Z}} p_t \, \varepsilon_{\mathbf{G}}.\end{aligned}$$

□

Remark 3.4. Here we do not have as in the lattice case [Ste+09] to make the assumption that the decoder is "strongly solution independent". In our case we can indeed have a uniform superposition over all the codewords and we can just use the way our error probability is defined, namely as the ratio $\frac{\sharp\mathscr{G}}{2^\ell q^k S_t}$.

All the probabilistic results of this section are easier to prove if, instead of choosing a code $\mathcal{C}$ by picking uniformly at random a generator matrix $\mathbf{G}$ for it, we slightly change the probabilistic model by picking uniformly at random a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ for it, *i.e.*,

$$\mathcal{C} = \left\{\mathbf{x}\in\mathbb{F}_q^n \colon \mathbf{H}\mathbf{x}^\intercal = \mathbf{0}\right\}.$$

We will denote $\mathbb{P}_{\mathbf{G}}$ and $\mathbb{P}_{\mathbf{H}}$ respectively the probabilities in the initial model and the probabilities in the new model. The two probability distributions are closely related: the first model always produces linear codes of dimension $\leq k$ and codes of dimension $= k$ with probability $1 - O\left(q^{-(n-k)}\right)$ whereas the second model always produces linear codes of dimension $\geq k$ and codes of dimension $= k$ with probability $1 - O\left(q^{-k}\right)$. This relationship is expressed by the following lemma.

Lemma 3.17. *Let $\mathcal{E}$ be an ensemble of linear codes of length n in $\mathbb{F}_q$. We have*

$$\mathbb{P}_{\mathbf{G}}(\mathcal{E}) \leq \mathbb{P}_{\mathbf{H}}(\mathcal{E}) + O\left(q^{-\min(k,n-k)}\right).$$

With this new probabilistic model, the expected value of Z is given by:

Lemma 3.2. *Assume that $\mathcal{C}$ is chosen by uniformly drawing at random a parity-check matrix $\mathbf{H}$ for it. We have:*

$$\mathbb{E}(Z) \leq 2^\ell q^k \left(1 + \frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}\right). \tag{3.11}$$

Proof. Computing $\mathbb{E}_{\mathbf{H}}(Z)$ with this alternate probabilistic model is straightforward. We have

$$\begin{aligned} Z &= \left\| \sum_{\mathbf{c}\in\mathcal{C},\mathbf{e}\in\mathbb{F}_q^n,\mathbf{r}\in\mathbb{F}_2^\ell} \pi_{\mathbf{e}} \left|\mathbf{0}_n\right\rangle \left|\mathbf{c}+\mathbf{e}\right\rangle \left|\mathbf{r}\right\rangle \right\|^2 \\ &= 2^\ell q^k \sum_{\mathbf{e}\in\mathbb{F}_q^n} |\pi_{\mathbf{e}}|^2 + 2^\ell \sum_{\substack{(\mathbf{c},\mathbf{e})\neq(\mathbf{c}',\mathbf{e}'):\\ \mathbf{c}+\mathbf{e}=\mathbf{c}'+\mathbf{e}'}} \pi_{\mathbf{e}}\pi_{\mathbf{e}'} \\ &= 2^\ell q^k \left(1 + \sum_{\mathbf{e}\neq\mathbf{e}'\,:\,\mathbf{H}(\mathbf{e}-\mathbf{e}')^\intercal=\mathbf{0}} \pi_{\mathbf{e}}\pi_{\mathbf{e}'}\right) \end{aligned}$$

where $\mathbf{H}$ is an arbitrary-parity check matrix for $\mathcal{C}$. Let

$$X \stackrel{\text{def}}{=} \sum_{\mathbf{e}\neq\mathbf{e}'\,:\,\mathbf{H}(\mathbf{e}-\mathbf{e}')^\intercal=\mathbf{0}} \pi_{\mathbf{e}}\pi_{\mathbf{e}'}.$$

The point of the probabilistic model where the parity-check matrix $\mathbf{H}$ is uniformly drawn at random is that for non-zero element $\mathbf{x}\in\mathbb{F}_q^n$ we have

$$\mathbb{P}_{\mathbf{H}}(\mathbf{x}\in\mathcal{C}) = \mathbb{P}_{\mathbf{H}}(\mathbf{H}\mathbf{x}^\intercal=\mathbf{0}) = \frac{1}{q^{n-k}}.$$

From this we deduce

$$\begin{aligned} \mathbb{E}_{\mathbf{H}}(X) &= \sum_{\mathbf{e}\neq\mathbf{e}'} \pi_{\mathbf{e}}\pi_{\mathbf{e}'}\mathbb{P}_{\mathbf{H}}((\mathbf{e}-\mathbf{e}')\in\mathcal{C}) \\ &= \sum_{\mathbf{e}\neq\mathbf{e}'} \frac{\pi_{\mathbf{e}}\pi_{\mathbf{e}'}}{q^{n-k}} \\ &\leq \sum_{\mathbf{e},\mathbf{e}'} \frac{\pi_{\mathbf{e}}\pi_{\mathbf{e}'}}{q^{n-k}} \\ &= \frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}. \end{aligned}$$

From this inequality we conclude the proof. □

With the help of these two lemmas, we can upper-bound the probability for Z to be bigger than $2^\ell q^k(1+\eta)$ for any $\eta>0$ and prove Lemma 3.3, that we recall:

Lemma 3.3. *Let $\eta>0$. We have:*

$$\mathbb{P}_{\mathbf{G}}(Z > 2^\ell q^k(1+\eta)) \leq \frac{1}{\eta}\,\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}} + O\left(q^{-\min(k,n-k)}\right).$$

Proof. We take back the notation from the proof of Lemma 3.2. We have

$$\begin{aligned}\mathbb{P}_{\mathbf{G}}(Z > 2^{\ell}q^{k}(1+\eta)) &= \mathbb{P}_{\mathbf{G}}(X > \eta)\\ &\leq \mathbb{P}_{\mathbf{H}}(X > \eta) + O\left(q^{-\min(k,n-k)}\right) \quad \text{(by Lemma 3.17)}\\ &\leq \frac{1}{\eta}\,\mathbb{E}_{\mathbf{H}}(X) + O\left(q^{-\min(k,n-k)}\right) \quad \text{(Markov inequality)}\\ &\leq \frac{1}{\eta}\,\frac{\langle\pi|\mathbf{1}\rangle^{2}}{q^{n-k}} + O\left(q^{-\min(k,n-k)}\right)\end{aligned}$$

which concludes the proof. □

3.2 Step 2: Proof of Lemma 3.4

If we apply a QFT on the second register of $|\psi_{\text{ideal}}\rangle$ (given in Equation 3.7), we obtain:

$$\widehat{|\psi_{\text{ideal}}\rangle} \stackrel{\text{def}}{=} (\mathbf{I}\otimes\mathsf{QFT}\otimes\mathbf{I})\,|\psi_{\text{ideal}}\rangle = \frac{q^{k}}{\sqrt{Z}}\sum_{\mathbf{c}^{\perp}\in\mathcal{C}^{\perp}}\sum_{\mathbf{r}\in\mathbb{F}_{2}^{\ell}}\widehat{\pi}_{\mathbf{c}^{\perp}}\,|\mathbf{0}_{n}\rangle\left|\mathbf{c}^{\perp}\right\rangle|\mathbf{r}\rangle\,,$$

where $\widehat{|\pi\rangle} = \sum_{\mathbf{e}\in\mathbb{F}_{q}^{n}}\widehat{\pi}_{\mathbf{e}}\,|\mathbf{e}\rangle$ is the QFT of $|\pi\rangle$. We use this remark to prove Lemma 3.4, that we recall:

Lemma 3.4. *If the Fourier transform is radially preserving, meaning that it transforms a radial function into a radial function, then after measuring $\widehat{|\psi_{ideal}\rangle}$ we obtain a state $|\mathbf{0}_{n}\rangle\left|\mathbf{c}^{\perp}\right\rangle|\mathbf{r}\rangle$ with $\mathbf{c}^{\perp}\in\mathcal{C}^{\perp}$ of weight u with probability $\frac{2^{\ell}q^{2k}}{Z}N_{u}^{\perp}\left|\widehat{f}(u)\right|^{2}$ where $\widehat{f}(u)\stackrel{def}{=}\widehat{\pi}_{\mathbf{e}}$ for an arbitrary $\mathbf{e}$ of weight u and $N_{u}^{\perp}$ is the number of codewords of weight u in $\mathcal{C}^{\perp}$.*

Proof. For $\mathbf{e}\in\mathbb{F}_{q}^{n}$, let

$$|\mathbf{1}_{\mathcal{C}+\mathbf{e}}\rangle \stackrel{\text{def}}{=} \sum_{\mathbf{c}\in\mathcal{C}}|\mathbf{c}+\mathbf{e}\rangle\,.$$

We have

$$\begin{aligned}\widehat{|\mathbf{1}_{\mathcal{C}+\mathbf{e}}\rangle} &= \sum_{\mathbf{c}\in\mathcal{C}}\frac{1}{\sqrt{q^{n}}}\sum_{\mathbf{y}\in\mathbb{F}_{q}^{n}}\chi_{\mathbf{y}}(\mathbf{c}+\mathbf{e})\,|\mathbf{y}\rangle\\ &= \frac{1}{\sqrt{q^{n}}}\sum_{\mathbf{y}\in\mathbb{F}_{q}^{n}}\chi_{\mathbf{y}}(\mathbf{e})\sum_{\mathbf{c}\in\mathcal{C}}\chi_{\mathbf{y}}(\mathbf{c})\,|\mathbf{y}\rangle\\ &= \frac{q^{k}}{\sqrt{q^{n}}}\sum_{\mathbf{c}^{\perp}\in\mathcal{C}^{\perp}}\chi_{\mathbf{c}^{\perp}}(\mathbf{e})\left|\mathbf{c}^{\perp}\right\rangle \quad \left(\text{since }\sum_{\mathbf{c}\in\mathcal{C}}\chi_{\mathbf{y}}(\mathbf{c}) = 0 \text{ if } \mathbf{y}\notin\mathcal{C}^{\perp} \text{ and } q^{k} \text{ otherwise}\right)\end{aligned}$$

Therefore

$$\begin{aligned}\widehat{\left|\sum_{\mathbf{e}\in\mathbb{F}_q^n,\mathbf{c}\in\mathcal{C}} \pi_{\mathbf{e}} \left|\mathbf{c}+\mathbf{e}\right\rangle\right\rangle} &= \sum_{\mathbf{e}\in\mathbb{F}_q^n} \pi_{\mathbf{e}} \widehat{|\mathbf{1}_{\mathcal{C}+\mathbf{e}}\rangle} \\ &= \frac{q^k}{\sqrt{q^n}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \pi_{\mathbf{e}} \sum_{\mathbf{c}^\perp\in\mathcal{C}^\perp} \chi_{\mathbf{c}^\perp}(\mathbf{e}) \left|\mathbf{c}^\perp\right\rangle \\ &= q^k \sum_{\mathbf{c}^\perp\in\mathcal{C}^\perp} \frac{1}{\sqrt{q^n}} \sum_{\mathbf{e}\in\mathbb{F}_q^n} \pi_{\mathbf{e}} \chi_{\mathbf{c}^\perp}(\mathbf{e}) \left|\mathbf{c}^\perp\right\rangle \\ &= q^k \sum_{\mathbf{c}^\perp\in\mathcal{C}^\perp} \widehat{\pi}_{\mathbf{c}^\perp} \left|\mathbf{c}^\perp\right\rangle .\end{aligned}$$

It follows that

$$\widehat{|\psi_{\text{ideal}}\rangle} = \frac{q^k}{\sqrt{Z}} \sum_{\mathbf{c}^\perp\in\mathcal{C}^\perp} \sum_{\mathbf{r}\in\mathbb{F}_2^\ell} \widehat{\pi}_{\mathbf{c}^\perp} \left|\mathbf{0}_n\right\rangle \left|\mathbf{c}^\perp\right\rangle \left|\mathbf{r}\right\rangle .$$

After measurement we get a state $\left|\mathbf{0}_n\right\rangle \left|\mathbf{c}^\perp\right\rangle \left|\mathbf{r}\right\rangle$ with probability $\frac{q^{2k}}{Z}\left|\widehat{f}(\left|\mathbf{c}^\perp\right|)\right|^2$. By summing over all $\mathbf{r}\in\mathbb{F}_2^\ell$ and $\mathbf{c}^\perp\in\mathcal{C}^\perp$ of weight u, we conclude the proof. □

3.3 Step 3 : Proof of Proposition 3.3

We first need for this a good estimation of $\widehat{|\psi_{\text{ideal}}\rangle}$'s amplitudes. This will be a consequence of the following lemma.

Lemma 3.18. *If the generator matrix* $\mathbf{G}$ *of a code* $\mathcal{C}$ *is chosen uniformly at random in* $\mathbb{F}_q^{k\times n}$ *then the number* $N_u^\perp$ *of codewords of weight* u *in* $\mathcal{C}^\perp$ *satisfies*

$$\mathbb{P}_{\mathbf{G}}\left(\left|N_u^\perp - \frac{S_u}{q^k}\right| \geq \left(\frac{S_u}{q^k}\right)^{3/4}\right) \leq (q-1)\sqrt{\frac{q^k}{S_u}}.$$

Proof. Let $\mathbb{1}_{\mathbf{x}}$ be the indicator function of the event "$\mathbf{x}\in\mathcal{C}^\perp$". By definition,

$$N_u^\perp = \sum_{\mathbf{x}\in\mathscr{S}_u} \mathbb{1}_{\mathbf{x}}. \tag{3.23}$$

We have $\mathbb{E}(\mathbb{1}_{\mathbf{x}}) = \mathbb{P}_{\mathbf{G}}(\mathbf{x}\in\mathcal{C}^\perp) = \frac{1}{q^k}$, implying that $\mathbb{E}(N_u^\perp) = \frac{S_u}{q^k}$. By using Bienaymé-

Tchebychev's inequality, we obtain:

$$\begin{aligned}
\mathbb{P}\left(\left|N_u^\perp - \frac{S_u}{q^k}\right| \geq a\right) &\leq \frac{\mathbf{Var}(N_u^\perp)}{a^2} \\
&= \frac{1}{a^2}\left(\sum_{\mathbf{x}\in\mathscr{S}_u} \mathbf{Var}(\mathbb{1}_\mathbf{x}) + \sum_{\substack{\mathbf{x},\mathbf{y}\in\mathscr{S}_u \\ \mathbf{x}\neq\mathbf{y}}} \mathbb{E}(\mathbb{1}_\mathbf{x}\mathbb{1}_\mathbf{y}) - \mathbb{E}(\mathbb{1}_\mathbf{x})\mathbb{E}(\mathbb{1}_\mathbf{y})\right) \\
&\leq \frac{1}{a^2}\left(\sum_{\mathbf{x}\in\mathscr{S}_u} \mathbb{E}(\mathbb{1}_\mathbf{x}) + \sum_{\substack{\mathbf{x},\mathbf{y}\in\mathscr{S}_u \\ \mathbf{x}\neq\mathbf{y}}} \mathbb{E}(\mathbb{1}_\mathbf{x}\mathbb{1}_\mathbf{y}) - \mathbb{E}(\mathbb{1}_\mathbf{x})\mathbb{E}(\mathbb{1}_\mathbf{y})\right) \\
&= \frac{1}{a^2}\left(\frac{S_u}{q^k} + \sum_{\substack{\mathbf{x},\mathbf{y}\in\mathscr{S}_u \\ \mathbf{x}\neq\mathbf{y}}} \mathbb{E}(\mathbb{1}_\mathbf{x}\mathbb{1}_\mathbf{y}) - \mathbb{E}(\mathbb{1}_\mathbf{x})\mathbb{E}(\mathbb{1}_\mathbf{y})\right) \qquad (3.24)
\end{aligned}$$

where we used that $\mathbf{Var}(\mathbb{1}_\mathbf{x}) \leq \mathbb{E}(\mathbb{1}_\mathbf{x}^2) = \mathbb{E}(\mathbb{1}_\mathbf{x})$. Let us now upper-bound the second term of the inequality. It is readily verified that:

$$\mathbb{E}(\mathbb{1}_\mathbf{x}\mathbb{1}_\mathbf{y}) = \begin{cases} 1/q^k & \text{if } \mathbf{x} \text{ and } \mathbf{y} \text{ are colinear,} \\ 1/q^{2k} & \text{otherwise.} \end{cases}$$

Therefore, we deduce that:

$$\begin{aligned}
\sum_{\substack{\mathbf{x},\mathbf{y}\in\mathscr{S}_u \\ \mathbf{x}\neq\mathbf{y}}} \mathbb{E}(\mathbb{1}_\mathbf{x}\mathbb{1}_\mathbf{y}) - \mathbb{E}(\mathbb{1}_\mathbf{x})\mathbb{E}(\mathbb{1}_\mathbf{y}) &= \sum_{\mathbf{x}\in\mathscr{S}_u} \sum_{\substack{\mathbf{y}\in\mathscr{S}_u\setminus\mathbf{x}: \\ \text{colinear to } \mathbf{x}}} \frac{1}{q^k} - \frac{1}{q^{2k}} \\
&\leq \sum_{\mathbf{x}\in\mathscr{S}_u} \sum_{\substack{\mathbf{y}\in\mathscr{S}_u\setminus\mathbf{x}: \\ \text{colinear to } \mathbf{x}}} \frac{1}{q^k} \\
&\leq \frac{(q-2)S_u}{q^k} \qquad (3.25)
\end{aligned}$$

It gives by plugging Equation 3.25 in Equation 3.24:

$$\mathbb{P}_\mathbf{G}\left(\left|N_u^\perp - \frac{S_u}{q^k}\right| \geq a\right) \leq \frac{1}{a^2}\left(\frac{S_u}{q^k} + \frac{(q-2)S_u}{q^k}\right) = \frac{(q-1)S_u}{a^2 q^k}$$

which concludes the proof by choosing $a = \left(\frac{S_u}{q^k}\right)^{3/4}$. □

We are ready to prove Proposition 3.3 which we now recall:

Proposition 3.3. *Under the assumptions made in Theorem 3.2, the probability of obtaining a codeword $\mathbf{c}^\perp \in \mathcal{C}^\perp$ of weight $u \in \mathscr{W}$ when measuring $|\widetilde{\psi_{ideal}}\rangle$ is $\geq 1 - \alpha(\pi)$*

for a proportion $\geq 1-\beta(\pi)$ *of matrices* $\mathbf{G}$*, where:*

$$\alpha(\pi) \stackrel{def}{=} \sum_{u\in\mathscr{W}} \left(\frac{q^k}{S_u}\right)^{1/4} + \sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}} - 2^{-\Omega(n)},$$
$$\beta(\pi) \stackrel{def}{=} (q-1)\sum_{u\in\mathscr{W}} \sqrt{\frac{q^k}{S_u}} + \sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}} + O\left(q^{-\min(k,n-k)}\right).$$

Proof. Let $\mathscr{Q}$ be the quantum algorithm starting from $|\psi\rangle$ which computes $(\mathbf{I}\otimes\mathsf{QFT}\otimes\mathbf{I})|\psi\rangle$. This algorithm succeeds when measuring a dual codeword $\mathbf{c}^\perp\in\mathcal{C}^\perp$ of weight $u\in\mathscr{W}$. When starting with $|\psi_{\text{ideal}}\rangle$, the probability of success of $\mathscr{Q}$ is equal to $\sum_{u\in\mathscr{W}}\frac{2^\ell q^{2k}N_u^\perp}{Z}\left|\widehat{f}(u)\right|^2$ by Lemma 3.4. Let

$$\mathscr{B} \stackrel{\text{def}}{=} \left\{\mathbf{G}\in\mathbb{F}_q^{k\times n}\colon Z > 2^\ell q^k\left(1+\sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}}\right)\right\} \quad \text{and}$$
$$\mathcal{E}_u \stackrel{\text{def}}{=} \left\{\mathbf{G}\in\mathbb{F}_q^{k\times n}\colon \left|N_u^\perp - \frac{S_u}{q^k}\right| \geq \left(\frac{S_u}{q^k}\right)^{3/4}\right\}.$$

By Lemma 3.3 and Lemma 3.18 we have that

$$\mathbb{P}\left(\mathbf{G}\in\mathscr{B}\cup\bigcup_{u\in\mathscr{W}}\mathcal{E}_u\right) \leq \beta(\pi) = (q-1)\sum_{u\in\mathscr{W}}\sqrt{\frac{q^k}{S_u}} + \sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}} + O\left(q^{-\min(k,n-k)}\right).$$

Therefore, for a proportion $\geq 1-\beta(\pi)$ of codes (over matrices $\mathbf{G}$):

(*i*) $Z \leq 2^\ell q^k\left(1+\sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}}\right)$ and $Z\geq 2^\ell q^k$ (this is true for any $\mathbf{G}$ as $\pi_{\mathbf{e}}\geq 0$ for any $\mathbf{e}$),

(*ii*) for all u in $\mathscr{W}$, $\left|\frac{q^kN_u^\perp}{S_u}-1\right| \leq \left(\frac{q^k}{S_u}\right)^{1/4}$.

We deduce that for a proportion $\geq 1-\beta(\pi)$ of codes and for all u in $\mathscr{W}$:

$$\frac{1-\left(\frac{q^k}{S_u}\right)^{1/4}}{1+\sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}}} \leq \frac{q^kN_u^\perp}{S_u}\,\frac{2^\ell q^k}{Z} \leq 1+\left(\frac{q^k}{S_u}\right)^{1/4}.$$

This implies that for a proportion $\geq 1-\beta(\pi)$ of codes and for all u in $\mathscr{W}$ we have that

$$\frac{1-\sum_{u\in\mathscr{W}}\left(\frac{q^k}{S_u}\right)^{1/4}}{1+\sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}}} \leq \frac{2^\ell q^{2k}N_u^\perp}{S_uZ} \leq 1+\sum_{u\in\mathscr{W}}\left(\frac{q^k}{S_u}\right)^{1/4},$$

from which we deduce that under the same conditions we also have

$$S_u\left|\widehat{f}(u)\right|^2\left(\frac{1-\sum_{u\in\mathscr{W}}\left(\frac{q^k}{S_u}\right)^{1/4}}{1+\sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}}}\right)$$
$$\leq\sum_{u\in\mathscr{W}}\frac{2^\ell q^{2k}N_u^\perp}{Z}\left|\widehat{f}(u)\right|^2\leq S_u\left|\widehat{f}(u)\right|^2\left(1+\sum_{u\in\mathscr{W}}\left(\frac{q^k}{S_u}\right)^{1/4}\right). \quad (3.26)$$

Now,

$$\frac{1-\sum_{u\in\mathscr{W}}\left(\frac{q^k}{S_u}\right)^{1/4}}{1+\sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}}}\geq 1-\sum_{u\in\mathscr{W}}\left(\frac{q^k}{S_u}\right)^{1/4}-\sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}}$$

Therefore, by plugging this in Equation 3.26 we have for a proportion $\geq 1-\beta(\pi)$ of codes:

$$(1-\delta)\sum_{u\in\mathscr{W}}S_u\left|\widehat{f}(u)\right|^2\leq\sum_{u\in\mathscr{W}}\frac{2^\ell q^{2k}N_u^\perp}{Z}\left|\widehat{f}(u)\right|^2\leq(1+\delta)\sum_{u\in\mathscr{W}}S_u\left|\widehat{f}(u)\right|^2 \quad (3.27)$$

where $\delta\stackrel{\text{def}}{=}\sum_{u\in\mathscr{W}}\left(\frac{q^k}{S_u}\right)^{1/4}+\sqrt{\frac{\langle\pi|\mathbf{1}\rangle^2}{q^{n-k}}}$. We finish the proof by applying Lemma 3.4: we namely know that after measuring the state obtained by $\mathscr{Q}$, we obtain a dual codeword of weight u in $\mathscr{W}$ with probability $\sum_{u\in\mathscr{W}}\frac{2^\ell q^{2k}N_u^\perp}{Z}\left|\widehat{f}(u)\right|^2$. □

4 Proof of Theorem 3.4

4.1 Proofs of Lemma 3.12 and Lemma 3.13

The following lemma will be very helpful in what follows.

Lemma 3.19. *[BCN89, §9.3, Lem. 9.3.2] Let V be a subspace of dimension s, then there are exactly $q^{(t-\ell)(s-\ell)}\begin{bmatrix}n-s\\t-\ell\end{bmatrix}_q\begin{bmatrix}s\\\ell\end{bmatrix}_q$ subspaces W of dimension t such that $\dim(V\cap W)=\ell$.*

Let us now recall Lemma 3.12:

Lemma 3.12. *$|\pi\rangle$ is radial, i.e., we may write $|\pi\rangle$ as*

$$|\pi\rangle=\sum_{\mathbf{E}\in\mathbb{F}_q^{m\times n}:\,|\mathbf{E}|\leq t}\pi_{\mathbf{E}}\,|\mathbf{E}\rangle$$

with $|\mathbf{E}\rangle\stackrel{def}{=}|\mathbf{E}_1\rangle\otimes\cdots\otimes|\mathbf{E}_m\rangle$ where the $\mathbf{E}_i$'s denote the rows of $\mathbf{E}$ and $\pi_{\mathbf{E}}=f(|\mathbf{E}|)$ where

$$f(u)=\begin{cases}\frac{\begin{bmatrix}n-u\\t-u\end{bmatrix}_q}{\sqrt{q^{mt}N}} & \text{if } u\leq t,\\ 0 & \text{otherwise.}\end{cases}$$

Proof. Let U be the $\mathbb{F}_q$-space generated by the $\mathbf{E}_i$'s. We denote by u the dimension of U. We have

$$\pi_{\mathbf{E}} = \frac{1}{\sqrt{q^{mt}N}} \sharp \left\{ V \leq \mathbb{F}_q^n \colon \dim V = t \text{ and } U \subseteq V \right\} = \frac{\begin{bmatrix} n-u \\ t-u \end{bmatrix}_q}{\sqrt{q^{mt}N}},$$

where we used Lemma 3.19 for the last equality. It concludes the proof. □

Another asymptotic expression for N and an estimate for p_t are given by:

Lemma 3.13. *We have:*

$$N = \Theta\left(\begin{bmatrix} n \\ t \end{bmatrix}_q \right) \quad \text{and} \quad p_t = \Theta(1).$$

This lemma will be a consequence of the following lemmas.

Lemma 3.20. *For any $V, W \leq \mathbb{F}_q^n$ such that $V \neq W$ and $\dim V = \dim W = t$ we have,*

$$\langle \pi_V | \pi_W \rangle = q^{m(\dim(V \cap W) - t)}.$$

Proof. Recall that,

$$|\pi_U\rangle = \left(\frac{1}{\sqrt{q^{\dim U}}} \sum_{\mathbf{u} \in U} |\mathbf{u}\rangle \right)^{\otimes m}$$

Therefore we have,

$$\langle \pi_V | \pi_W \rangle = \left(\frac{1}{q^t} \sum_{\mathbf{v} \in V} \sum_{\mathbf{w} \in W} \langle \mathbf{v} | \mathbf{w} \rangle \right)^m = \left(\frac{1}{q^t} \sharp (V \cap W) \right)^m$$

which concludes the proof. □

Lemma 3.21. *We have,*

$$\sum_{\substack{V \leq \mathbb{F}_q^n \\ \dim V = t}} \sum_{\substack{W \leq \mathbb{F}_q^n \\ \dim W = t \\ W \neq V}} \langle \pi_V | \pi_W \rangle = O\left(\begin{bmatrix} n \\ t \end{bmatrix}_q \right).$$

Proof. We have

$$\begin{aligned}
\sum_{\substack{V\leq\mathbb{F}_q^n\\ \dim V=t}}\sum_{\substack{W\leq\mathbb{F}_q^n\\ \dim W=t\\ W\neq V}}\langle\pi_V|\pi_W\rangle &= \sum_{\substack{V\leq\mathbb{F}_q^n\\ \dim V=t}}\sum_{\substack{W\leq\mathbb{F}_q^n\\ \dim W=t\\ W\neq V}} q^{m(\dim(V\cap W)-t)}\\
&= \sum_{\substack{V\leq\mathbb{F}_q^n\\ \dim V=t}}\sum_{\ell=0}^{t-1}\sum_{\substack{W\leq\mathbb{F}_q^n\\ \dim W=t\\ \dim(W\cap V)=\ell}}\frac{1}{q^{m(t-\ell)}}\\
&= \sum_{\substack{V\leq\mathbb{F}_q^n\\ \dim V=t}}\sum_{\ell=0}^{t-1}\frac{1}{q^{m(t-\ell)}}\,q^{(t-\ell)^2}\begin{bmatrix}t\\ \ell\end{bmatrix}_q\begin{bmatrix}n-t\\ t-\ell\end{bmatrix}_q \qquad (3.28)\\
&= \begin{bmatrix}n\\ t\end{bmatrix}_q\sum_{\ell=0}^{t-1}q^{(t-\ell-m)(t-\ell)}\begin{bmatrix}t\\ \ell\end{bmatrix}_q\begin{bmatrix}n-t\\ t-\ell\end{bmatrix}_q
\end{aligned}$$

where in Equation 3.28 we used Lemma 3.19. Now, there exists some constant $c>0$ such that:

$$\begin{bmatrix}n\\ \ell\end{bmatrix}_q \leq cq^{\ell(n-\ell)}.$$

Then, for some constant $C>0$,

$$\begin{aligned}
\sum_{\substack{V\leq\mathbb{F}_q^n\\ \dim V=t}}\sum_{\substack{W\leq\mathbb{F}_q^n\\ \dim W=t\\ W\neq V}}\langle\pi_V|\pi_W\rangle &\leq C\begin{bmatrix}n\\ t\end{bmatrix}_q\sum_{\ell=0}^{t-1}q^{(t-\ell-m)(t-\ell)+\ell(t-\ell)+(t-\ell)(n-2t+\ell)}\\
&= C\begin{bmatrix}n\\ t\end{bmatrix}_q\sum_{\ell=0}^{t-1}q^{(t-\ell)(t-\ell-m+\ell+n-2t+\ell)}\\
&= C\begin{bmatrix}n\\ t\end{bmatrix}_q\sum_{\ell=0}^{t-1}q^{(t-\ell)(-t+\ell-m+n)}\\
&\leq C\begin{bmatrix}n\\ t\end{bmatrix}_q\sum_{\ell=0}^{t-1}q^{-(t-\ell)^2}\quad(\text{since } n\leq m)
\end{aligned}$$

which concludes the proof. □

We are now ready to prove Lemma 3.13.

Proof of Lemma 3.13. By definition of N we have:

$$N=\left\|\sum_{\substack{V\leq\mathbb{F}_q^n\\ \dim V=t}}\pi_V\right\|^2=\sum_{\substack{V\leq\mathbb{F}_q^n\\ \dim V=t}}\|\pi_V\|^2+\sum_{\substack{V\leq\mathbb{F}_q^n\\ \dim V=t}}\sum_{\substack{W\leq\mathbb{F}_q^n\\ \dim W=t\\ W\neq V}}\langle\pi_V,\pi_W\rangle$$

and by definition of π_V:

$$\sum_{\substack{V \leq \mathbb{F}_q^n \\ \dim V = t}} \|\pi_V\|^2 = \begin{bmatrix} n \\ t \end{bmatrix}_q \left(\frac{1}{q^t} \sum_{v \in V} 1 \right)^m = \begin{bmatrix} n \\ t \end{bmatrix}_q$$

This concludes the proof that $N = \begin{bmatrix} n \\ t \end{bmatrix}_q$ by using Lemma 3.21. Now, by definition of p_t, we have:

$$\begin{aligned} p_t &= S_t f(t)^2 \\ &= S_t \frac{\begin{bmatrix} n-t \\ 0 \end{bmatrix}_q^2}{q^{mt} N} \quad \text{(by Lemma 3.12)} \\ &= \frac{S_t}{\Theta(S_t)} \quad \text{(by using the estimate for } N \text{ and Equation 2.8)} \end{aligned}$$

allowing us to conclude that $p_t = \Theta(1)$. □

4.2 Proofs of Lemma 3.15 and Lemma 3.16

Recall Lemma 3.15 first:

Lemma 3.15. *We have,*

$$\frac{S_t}{q^{mn-k}} = q^{-\Omega(n)} \quad \text{and} \quad \frac{\langle \pi | \mathbf{1} \rangle^2}{q^{mn-k}} = q^{-\Omega(n)}.$$

Proof. In order to prove the first equation, note that $d_{\mathrm{GV}}(m,n,k)$ is defined such that $\frac{S_{d_{\mathrm{GV}}}}{q^{mn-k}} \leq 1$. From this and Equation 2.9 we deduce

$$\frac{S_{d_{\mathrm{GV}}-1}}{q^{mn-k}} = \frac{S_{d_{\mathrm{GV}}-1}}{S_{d_{\mathrm{GV}}}} \frac{S_{d_{\mathrm{GV}}}}{q^{mn-k}} \leq \Theta\left(q^{-(m+n-2d_{\mathrm{GV}}-1)}\right) = \Theta\left(q^{-\Omega(n)}\right)$$

where the last equality follows from the fact that (see for instance [Loi06])

$$d_{\mathrm{GV}}(n,m,k) = \frac{m+n-\sqrt{(m-n)^2+4k}}{2}(1+o(1)). \tag{3.29}$$

Now, for the second equation, we have

$$\begin{aligned} \frac{\langle \pi | \mathbf{1} \rangle^2}{q^{mn}} &= \left| \widehat{f}(0) \right|^2 \\ &= \frac{\begin{bmatrix} n \\ n-t \end{bmatrix}_q^2}{q^{m(n-t)} N} \quad \text{(by Lemma 3.14)} \\ &= \Theta\left(\frac{q^{mt} \begin{bmatrix} n \\ t \end{bmatrix}_q}{q^{mn}} \right) \quad \text{(By Lemma 3.13)} \\ &= \Theta\left(\frac{S_t}{q^{mn}} \right) \quad \text{(By Equation 2.7 and Equation 2.8)} \end{aligned}$$

Thanks to the first equation, we complete the proof. □

The second lemma we need is recalled here:

Lemma 3.16. *For any $\eta > 0$, we have,*

$$\forall u \in [\![(1-\eta)n - t, n - t]\!]\,,\ \frac{q^k}{S_u} = q^{-\Omega(n)} \quad \textit{and} \quad \sum_{u \in [\![(1-\eta)n-t, n-t]\!]} S_u \left|\widehat{f}(u)\right|^2 = 1 - q^{-\Omega(n)}.$$

Proof. The first equation can be proved in the same way as Lemma 3.15. For the second identity, first notice (by Lemma 3.14) that

$$\sum_{u \in [\![n-t-\eta n, n-t]\!]} S_u \left|\widehat{f}(u)\right|^2 = 1 - \sum_{u < n-t-\eta n} S_u \left|\widehat{f}(u)\right|^2.$$

Now, we have the following computation:

$$\begin{aligned}
\sum_{u<n-t-\eta n} S_u \left|\widehat{f}(u)\right|^2 &= \sum_{u<n-t-\eta n} S_u \, \frac{\begin{bmatrix} n-u \\ n-u-t \end{bmatrix}_q^2}{N\, q^{m(n-t)}} \quad \text{(by Lemma 3.15)} \\
&= \Theta\left(\sum_{u<n-t-\eta n} S_u \, \frac{\begin{bmatrix} n-u \\ n-u-t \end{bmatrix}_q^2}{\begin{bmatrix} n \\ t \end{bmatrix}_q q^{m(n-t)}} \right) \quad \text{(by Lemma 3.13)} \\
&= \Theta\left(\sum_{u<n-t-\eta n} q^{u(m+n-u)} \, \frac{q^{2(n-t-u)t}}{q^{t(n-t)}\, q^{m(n-t)}} \right) \quad \text{(By Equation 2.7 and 2.8)} \\
&= \Theta\left(q^{\max_{u<n-t-\eta n} u(m+n-u-2t)} \, q^{(t-m)(n-t)} \right)
\end{aligned}$$

Let $g(u) \stackrel{\text{def}}{=} u(m+n-u-2t)$. Then, $g'(u) = m+n-2(t+u) \geq 0$ as $t+u \leq n \leq m$. Therefore, g is an increasing function and by setting $u = n-t-\eta n$, we obtain

$$\begin{aligned}
\sum_{u<n-t-\eta n} S_u \left|\widehat{f}(u)\right|^2 &\leq n\ \Theta\left(q^{(n-t-\eta n)(m-t+\eta n)} \, q^{(t-m)(n-t)} \right) \\
&= n\ \Theta\left(q^{(n-t)(t-m+m-t+\eta n)} \, q^{-\eta n(m-t+\eta n)} \right) \\
&= n\ \Theta\left(q^{-\eta n(m-t+\eta n-n+t)} \right) \\
&= n\ \Theta\left(q^{-\eta n(m-n+\eta n)} \right) \\
&= q^{-\Omega(n)}
\end{aligned}$$

as $n \leq m$ by assumption. It concludes the proof. □

Chapter 4
Introduction to the Dihedral Hidden Subgroup Problem

The dihedral group of order $2N$, which will be denoted D_N, is simply defined as the group of symmetries (rotations and reflections) of a regular N-gon. Formally:

$$D_N \stackrel{\text{def}}{=} \left\langle u, v \colon u^N = v^2 = (uv)^2 = 1 \right\rangle$$

Here u can be thought as a rotation of the N-gon by an angle $2\pi/N$ and v as a reflection about some fixed axis. A simple example is given in Figure 4.1 of the action of the dihedral group on a regular pentagon.

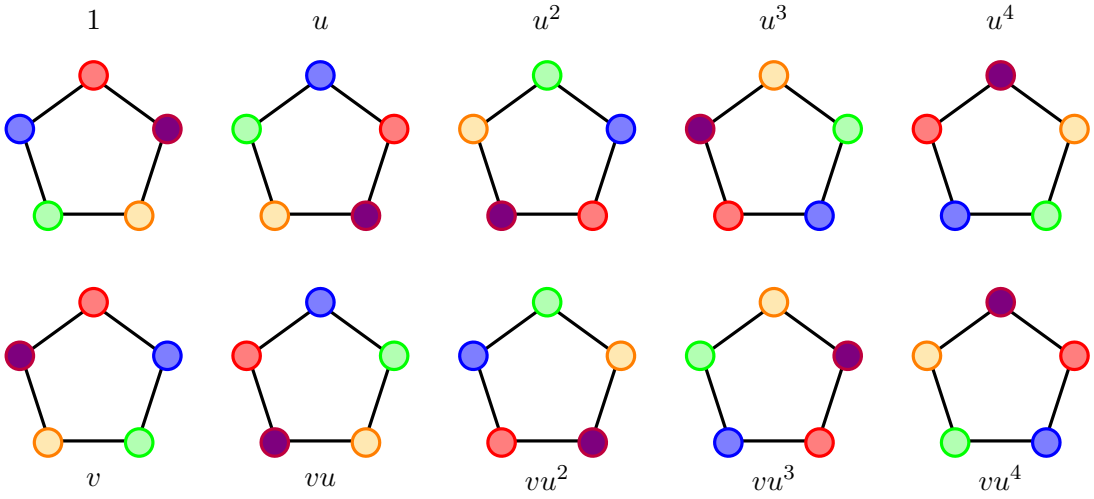

Figure 4.1: Action of the dihedral group D_5 of order 10 on a regular pentagon. The first row shows the effect of the rotation u, the second one shows the effect of a vertical reflection v applied to the first row. We denote by 1 the identity.

It results from the definition we gave that any group element can be written in the form $u^x v^b$ where $x \in \mathbb{Z}_N$ and $b \in \mathbb{Z}_2$. Thus we can equivalently think of the group as consisting of elements $(x, b) \in \mathbb{Z}_N \times \mathbb{Z}_2$, where x defines the angle of the rotation applied and b if we apply a reflection or not. For more information about dihedral groups, we refer to [Chi22; Con23].

In this chapter, we look at the HSP in a dihedral group (DHSP). While the HSP in an abelian group is quantumly easy to solve, as we saw in Chapter 1, many post-quantum primitives are related to the DHSP. This is the case of cryptosystems based on the Unique Shortest Vector Problem (uSVP) in lattice-based cryptography (such as [AD97; Reg04b]) or on any problem that can be reduced to the uSVP (because of a chain of reductions between several problems [Reg02; LM09; Ste+09]). More concretely, the security of several primitives reduces to the DHSP. The most prominent example is the isogeny-based post-quantum key-exchange CSIDH [Cas+18], which is similar to the Diffie-Hellman protocol [DH76] except that it does not rely on the period-finding problem in an abelian group (which is solvable in quantum polynomial time), but on the difficulty to invert the group action. As it has been shown in [BS20; Pei20; Chá+22], a better understanding of the security of CSIDH comes from a careful analysis of quantum algorithms solving the DHSP. Several related constructions [Ala+20] such as the signature schemes SeaSign [DG19; DPV19] and CSI-FiSh [BKV19] also rely on this problem. It should be noted that these isogeny-based cryptosystems are the only major contenders for which the quantum attacker enjoys more than a quadratic speedup, as opposed to the lattice- and hash-based finalists of the NIST post-quantum standardization process [NIS16; Ala+22].

In Section 1, we will see how to reduce the DHSP, which is a purely classical problem, to a quantum problem, known as the Dihedral Coset Problem (DCP). The following sections will then introduce the milestone algorithms, from the first algorithm to solve the DCP in sub-exponential time to Kuperberg's second algorithm, which is the state of the art, via Regev's algorithm, which was the first algorithm to solve the DCP with a polynomial number of qubits.

Contents

1 From Finding a Hidden Subgroup to Solving a Quantum Problem

We properly define here the Dihedral Hidden Subgroup Problem and the Dihedral Coset Problem. We show how to reduce the former to the latter, by the intermediate of another problem which is the hidden subgroup problem in a dihedral group with a promise on the structure of this subgroup.

1.1 Dihedral Hidden Subgroup Problem

The Dihedral Hidden Subgroup Problem, DHSP for short, is simply the Hidden Subgroup Problem in a dihedral group D_N for some integer N. We take it to be greater than 2 in what follows, since D_N is non-abelian in this case. We then have here an example of a group with a relatively simple structure for which the standard algorithm solving the abelian HSP cannot be applied.

We will therefore be interested in the structure that the hidden subgroup in a dihedral group can take, *i.e.*, the possible subgroups of a dihedral group. But first we start by proving that an exhaustive search of the hidden subgroup in D_N is not possible when N is too big. The following lemma states the number of subgroups S_N of the dihedral group D_N:

Lemma 4.1 ([Mil14])**.** *The number S_N of subgroups of D_N is equal to $\sigma(N) + \tau(N)$ where $\sigma(N)$ is the sum of the divisors of N and $\tau(N)$ is the number of divisors of N:*

$$\sigma(N) \overset{def}{=} \sum_{d\colon d|N} d \qquad \text{and} \qquad \tau(N) \overset{def}{=} \sum_{d\colon d|N} 1.$$

From this lemma, we can then deduce a lower bound on S_N, which we find by considering that N is a prime number since it is trivially the case where S_N will be the smallest. In this case, $\sigma(N) = N + 1$ and $\tau(N) = 2$, since the only divisors of N are 1 and N itself. It follows that when N is exponential, so is an exhaustive search to find a hidden subgroup in D_N.

We will now look at the structure of the subgroups of a dihedral group. In fact, it can be shown that its subgroups are either dihedral themselves or cyclic, as stated in the following lemma:

Lemma 4.2 (Theorem 3.1 [Con23])**.** *The possible subgroups of D_N are of the form*

- $\langle (x, 0) \rangle$ *where $x \in \mathbb{Z}_N$ is some divisor of N (cyclic subgroups, normal in D_N),*
- $\langle (x, 0), (y, 1) \rangle$ *where $x \in \mathbb{Z}_N$ is some divisor of N and $0 \leq y < x$ (dihedral subgroups).*

Remark 4.1. More precisely, Miller actually proved in [Mil14] that $\tau(N)$ is the number of cyclic subgroups and $\sigma(N)$ is the number of dihedral subgroups.

It was proved by Ettinger and Høyer that it is possible to reduce the general DHSP (in which the hidden subgroup could be any of the possible ones) to a DHSP with the promise that the hidden subgroup is of a certain form.

Lemma 4.3 (Proof of Theorem 2 [EH99])**.** *The* DHSP *in* D_N *reduces to a* DHSP *where the hidden subgroup is either the trivial subgroup or* $\langle (y,1)\rangle = \{(0,0),(y,1)\}$ *for some* $y \in \mathbb{Z}_N$.

Proof. Let H be a subgroup of D_N. We have from Lemma 4.2 that H can be either cyclic or dihedral:

$$\begin{cases} \langle u^x \rangle & \text{with } x|N \\ \langle u^x, u^y v \rangle & \text{with } x|N \text{ and } y \in [\![0, x-1]\!] \end{cases}$$

Let us show that any of these cases reduce to the particular case where the subgroup is of the form $\langle u^y v\rangle$, or trivial. We distinguish the case where $x = N$ (*i.e.* $x = 0$) from the others.

- When $x \neq N$, the hidden subgroup H necessarily has a cyclic part and therefore a cyclic subgroup, namely $\langle u^x\rangle$. This subgroup being normal in D_N, we can compute the quotient group $D_N/\mathbb{Z}_x \simeq D_{N/x}$. The hidden subgroup in this quotient group is then $H/\mathbb{Z}_x$, which does not have any cyclic subgroup, meaning that it is either the trivial subgroup or a subgroup of the form $\langle u^y v\rangle$.
- When $x = N$, we are in the same case where H can be either trivial or dihedral with no cyclic subgroup.

In conclusion, it does not matter what form the subgroup H takes, it will reduce to finding a subgroup of the form $\langle u^y v\rangle$, *i.e.* a subgroup of the form $\langle (y,1)\rangle$. □

1.2 Dihedral Coset Problem

We showed in the previous subsection that if we know how to solve the DHSP with the promise that the hidden subgroup is of the form $\langle (s,1)\rangle = \{(0,0),(s,1)\}$ where $s \in Z_N$ is unknown, then we can in fact solve the general DHSP in which the hidden subgroup could be any of the possible ones. We now have to take a closer look to the DHSP with a promise and how to solve it. We let $f\colon D_N \to S$ be a function that fulfills the subgroup promise with respect to H, where H is $\{(0,0),(s,1)\}$ for some unknown $s \in \mathbb{Z}_N$ (H could actually be the trivial subgroup $\{(0,0)\}$ but as it is easy to deal with it, we will leave it on the side). Finding H is equivalent to finding s, so we will in fact focus on finding the unknown value s.

A standard method to solve the DHSP with a promise thanks to a quantum algorithm is by coset sampling. The beginning is the same as the algorithm we described in Chapter 1, for the abelian case: we build a superposition over the elements of G and compute $f(G)$ in an ancillary register thanks to the hiding function f. We then measure the ancillary, which leaves us with a coset state $|(x,0)H\rangle$, a superposition over a coset of the unknown subgroup H which is promised to be $\langle(s,1)\rangle = \langle(0,0),(s,1)\rangle$. This method is shown in Algorithm 4.

Algorithm 4 Building coset states

Require: A unitary U_f to compute f.

1: Prepare a uniform superposition over $\mathbb{Z}_N \times \mathbb{Z}_2$ and compute f by using U_f

$$\frac{1}{\sqrt{2N}} \sum_{x\in\mathbb{Z}_N} \sum_{b\in\mathbb{Z}_2} |x, b, f(x,b)\rangle$$

2: Measure the ancillary register and discard it

$$\frac{1}{\sqrt{2}} \sum_{b\in\mathbb{Z}_2} |x+bs, b\rangle = \frac{1}{\sqrt{2}} \left(|x,0\rangle + |x+s,1\rangle\right) \stackrel{\text{def}}{=} |(x,0)H\rangle$$

It can be verified that the value x is picked at random from $\mathbb{Z}_N$ during the measurement. This procedure is the crucial argument used in [EH99] to conclude that solving the DHSP with a promise reduces to the Dihedral Coset Problem (DCP), which is the quantum problem of finding s from *coset states* $|(x,0)H\rangle$, defined as follows.

Problem 4.1. *Dihedral Coset Problem (DCP). Suppose we have a collection of coset states $|(x,0)H\rangle$ for random unknown $x \in [\![0,N]\!]$. The DCP asks to find $s \in [\![0,N]\!]$.*

From coset states, two strategies are available to us. The first one consists in applying a tensor product of a QFT and a Hadamard gate on $\mathbb{Z}_N \times \mathbb{Z}_2$ (and thus proceed in a way analogous to what is done for abelian groups, as seen in the previous chapter) and the second one consists in applying a QFT on $\mathbb{Z}_N$ only. Let's take a closer look at these two strategies.

1.2.1 Quantum Fourier Transform over $\mathbb{Z}_N \times \mathbb{Z}_2$

In their paper [EH99], Ettinger and Høyer applied $\mathbf{F}_N \otimes \mathbf{H}$ on $|(x,0)H\rangle$, which gives

$$\frac{1}{2\sqrt{N}} \sum_{a\in\mathbb{Z}_N} \sum_{b\in\mathbb{Z}_2} \omega_N^{ax} \left(1 + \omega_N^{as+b\frac{N}{2}}\right) |a\rangle\,|b\rangle\,. \tag{4.1}$$

They then measured the two registers to obtain a pair $(a,b) \in \mathbb{Z}_N \times \mathbb{Z}_2$ that is measured with probability given in the following lemma:

Lemma 4.4 ([EH99])**.** *The probability for (a, b) to be the outcome of the measurement is*

$$\begin{cases} \frac{1}{N}\cos^2\left(\pi\frac{as}{N}\right) & \text{if } b = 0 \\ \frac{1}{N}\sin^2\left(\pi\frac{as}{N}\right) & \text{if } b = 1 \end{cases}$$

Proof. The probability of measuring $(a, b) \in \mathbb{Z}_N \times \mathbb{Z}_2$ is

$$\frac{1}{4N}\left|1 + \omega_N^{as+b\frac{N}{2}}\right|^2 = \frac{1}{N}\cos^2\left(\frac{\pi}{2}b + \pi\frac{as}{N}\right).$$

□

Ettinger and Høyer then built an algorithm upon this method to retrieve information about s from a collection of $O(\log N)$ results of this procedure, as stated in Theorem 4.1, but it runs in exponential classical time and it appears to be difficult to design a more efficient algorithm using this strategy.

Theorem 4.1 (Theorem 3 of [EH99])**.** *There exists a quantum algorithm that outputs either "trivial" or the secret value s using at most $O(\log N)$ evaluations of the hiding function f. The output is always "trivial" if H is trivial and the algorithm outputs s with probability at least $1 - \frac{1}{2N}$ otherwise.*

We will not give the proof of this theorem here. We can simply remark from Lemma 4.4 that the probability distribution we get from measuring states of the form given by Equation 4.1 is non-uniformly distributed on $\mathbb{Z}_N$ depending on s. From such states measurements, we can thus retrieve information about s using Lemma 4.4. That is roughly speaking what Ettinger-Høyer algorithm does: it produces a linear number (in $\log N$) of couples (a, b) with the above procedure and then goes through all possible values for s and finds which one best fits the measurement results using a statistical argument. The number of queries to the hiding function is thus linear (in $\log N$) and enough to solve the DCP, however, the algorithm has the major disadvantage of requiring an exponential classical time, since it proceeds to an enumeration on a list of N values.

1.2.2 Quantum Fourier Transform over $\mathbb{Z}_N$

We now look at the other strategy, of applying the QFT over $\mathbb{Z}_N$ only, which yields

$$\frac{1}{\sqrt{2N}}\sum_{k\in\mathbb{Z}_N}\omega_N^{kx}|k\rangle\left(|0\rangle + \omega_N^{sk}|1\rangle\right)$$

Measuring the first register collapses the superposition on

$$\frac{1}{\sqrt{2}}|k\rangle\left(|0\rangle + \omega_N^{sk}|1\rangle\right)$$

for a value k picked at random from $\mathbb{Z}_N$. In the second register, we obtain a state which will be denoted by $|\psi_k\rangle$ and called a *phase vector*.

Definition 4.1 (Phase vector)**.** Let $k \in \mathbb{Z}_N$. We denote by $|\psi_k\rangle$ the phase vector defined as

$$|\psi_k\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}(|0\rangle + \omega_N^{sk} |1\rangle) \tag{4.2}$$

where $s \in \mathbb{Z}_N$ is an unknown integer we are looking for.

These states will turn out to be useful to reveal information about the secret s in a much more efficient way than with the previous strategy. Namely, we will be able to design subexponential time quantum algorithms to solve the DCP. We will consider in what follows that we have access to an oracle outputting phase vectors as defined in Definition 4.1, since finding $s \in [\![0, N]\!]$ from a collection of states $|\psi_k\rangle$ for known and uniformly distributed random $k \in [\![0, N]\!]$ solves the DCP and thus the DHSP in turn, as we saw previously.

2 Kuperberg's first algorithm

With a simple but clever way of assembling two phase vectors to obtain one with better properties, Kuperberg designed in 2003 the first algorithm to solve the DCP in subexponential time [Kup05]. For the sake of clarity, we will take N to be a power of 2, namely $N \stackrel{\text{def}}{=} 2^n$, but the algorithm described here works for any integer N.

It is indeed possible to combine two phase vectors in order to construct a new one in the following way

$$\begin{aligned} |\psi_p, \psi_q\rangle &= \frac{1}{2}\left(|0,0\rangle + \omega_N^{sp} |1,0\rangle + \omega_N^{sq} |0,1\rangle + \omega_N^{s(p+q)} |1,1\rangle\right) \\ &\xmapsto{\text{CNOT}} \frac{1}{2}\left(|0,0\rangle + \omega_N^{s(p+q)} |1,0\rangle + \omega_N^{sq} |0,1\rangle + \omega_N^{sp} |1,1\rangle\right) \\ &= \frac{1}{\sqrt{2}}\left(|\psi_{p+q}, 0\rangle + \omega_N^{sq} |\psi_{p-q}, 1\rangle\right) \end{aligned}$$

A measurement of the second qubit will leave the first one in the state $|\psi_{p+q}\rangle$ if 0 is obtained and in the state $|\psi_{p-q}\rangle$ if 1 is obtained. Both cases happen with probability $1/2$. Having this combination method on hand, it is interesting to note that if p and q had a certain number m of least significant bits in common, then with probability $1/2$ we would get $|\psi_{p-q}\rangle$, where $p - q$ would have its m least significant bits equal to 0.

In parallel, we note that the state $\left|\psi_{N/2}\right\rangle$, associated to the value $N/2$ whose binary development is $(0, \ldots, 0, 1)$ (*i.e.*, all its bits are zeroes except for the most significant

one), is

$$\begin{aligned}\left|\psi_{N/2}\right\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + \omega_N^{s\frac{N}{2}}|1\rangle\right)\\ &= \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{\mathrm{lsb}(s)}|1\rangle\right)\\ &= \mathbf{H}\,|\mathrm{lsb}(s)\rangle\end{aligned}$$

So we have a method that, if we iterate it, produces phase vectors with more and more least significant bits equal to zero, and a state with all its bits equal to zero except for the most significant one, which we would like to obtain because it would give us information about the secret, namely $\mathrm{lsb}(s)$.

Kuperberg therefore designed the sieve which is roughly represented in Figure 4.2 and described as follows. We start by generating many phase vectors, which are gathered in a list. Their most significant bit, which is represented by a darker orange column, will not be affected. The first step is to classify the states according to their value on their m least significant bits (there will typically be 2^m lists for the 2^m possible bit strings on m bits). Once this is done, we take the phase vectors in each sublist two by two, combine them by the CNOT trick, and keep the resulting state when the difference is obtained. Finally, all the states are grouped together in the same list and the process is iterated.

2.1 Algorithm

The pseudocode for Kuperberg's algorithm is given by Algorithm 5.

Algorithm 5 Kuperberg's Algorithm

Require: A parameter m and a list L of 2^ℓ coset states as in Equation 4.2, where each copy has $k \in \mathbb{Z}_N$ chosen independently and uniformly at random.
Ensure: The least significant bit of the secret s.

1: **for** j from 0 to $m-1$ **do**
2: Collect the coset states into pairs $(|\psi_p\rangle, |\psi_q\rangle)$ sharing at least their m lower bits, their jm trailing zeroes being excluded, or $n-1-jm$ if $j = m-1$. Discard the remaining states that cannot be paired.
3: Use the combination routine to create a state $|\psi_{p\pm q}\rangle$ from each pair, and discard it if the $+$ sign occurs.
4: **if** $|\psi_{2^{n-1}}\rangle \in L$ **then** Measure it in the $|\pm\rangle$ basis and **return** the result
5: **return** "The algorithm failed"

As previously described, this algorithm solves the DCP by collecting states that share many of their last significant bits into pairs $(|\psi_p\rangle, |\psi_q\rangle)$, such that $|\psi_{p-q}\rangle$ is likely to have many of its least significant bits equal to zero. It would require an exponential

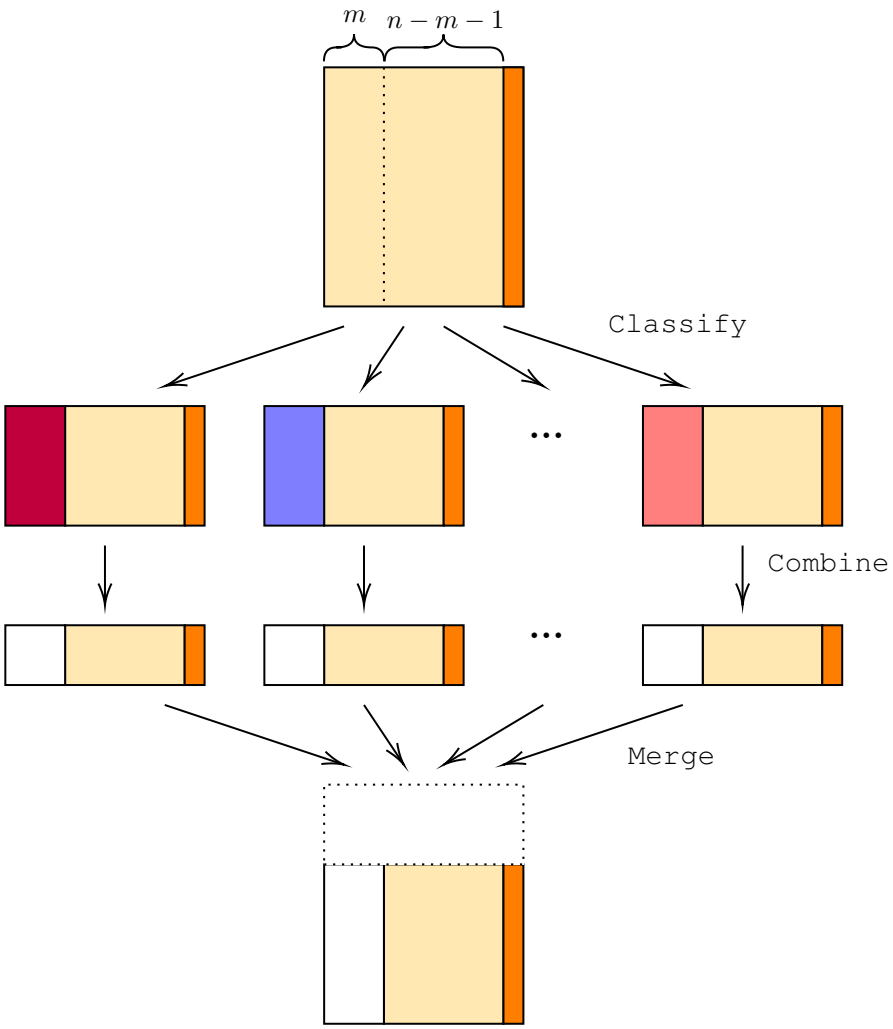

Figure 4.2: Diagram representing the process of the Kuperberg algorithm. It is inspired by the one for the BKW algorithm which can be found in the slides [The22] corresponding to the paper [LY22], the only difference being in the combination step, where instead of using a vector that we XOR to all the others, we take them two by two and combine them with a CNOT and a measure. We start with a list of phase vectors whose labels' most significant bit is shown in dark orange. The classification step separates these labels according to their value on their m least significant bits (each color corresponds to a different string of m bits). The combination step then allows us to construct phase vectors whose labels have their m least significant bits equal to 0, which is shown in white.

time to zero out all but the most significant one in one shot so instead, we proceed by zeroing out the m least significant bits in successive rounds. This parameter m has to be carefully chosen as it can bring the time complexity of the algorithm down to subexponential, as we will see in what follows.

We will also look at the number of initial phase vectors, which should be chosen such as there remains at least one element at the end of the process, *i.e.*, one state with all its $n-1$ least significant bits equal to zero. Indeed, when $\left|\psi_{N/2}\right\rangle$ is obtained, we are left with applying a Hadamard gate on it and measuring it in order to obtain $\mathrm{lsb}(s)$. From this point where we know the parity of s, we can determine to which subgroup $D_{N/2}$ of D_N it belongs and then reapply the algorithm to determine the second least significant bit of s (which is its parity bit in $D_{N/2}$), and so on, until the whole secret is discovered.

Bonnetain and Naya-Plasencia proved in [BN18] that it is actually possible to recover all the bits of s at once rather than working on successive least significant bits in dihedral groups with smaller and smaller cardinal. It turns out that their procedure allows a polynomial speed-up over Kuperberg's algorithm.

They first showed that it is possible to retrieve s from a collection of phase vectors $|\psi_{k_i}\rangle$ with k_i such that $2^i | k_i$ but $2^{i+1} \not| k_i$, for each i from 0 to $n-1$. It is equivalent to say that there exists $\alpha_i \in [\![0, 2^{n-1-i}-1]\!]$ such that

$$|\psi_{k_i}\rangle = \left|\psi_{(2\alpha_i+1)2^i}\right\rangle.$$

Indeed, up to a phase correction, we can find s_{n-1-i} from $|\psi_{k_i}\rangle$, as shown in the following computation:

$$\begin{aligned}
\left|\psi_{(2\alpha_i+1)2^i}\right\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{\mathbf{i}\pi s \frac{2\alpha_i+1}{2^{n-1-i}}}|1\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{\mathbf{i}\pi(s-(s \mod 2^{n-1-i}))\frac{2\alpha_i+1}{2^{n-1-i}}} e^{\mathbf{i}\pi(s \mod 2^{n-1-i})\frac{2\alpha_i+1}{2^{n-1-i}}}|1\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{\mathbf{i}\pi s_{n-1-i}} e^{\mathbf{i}\pi(s \mod 2^{n-1-i})\frac{2\alpha_i+1}{2^{n-1-i}}}|1\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{s_{n-1-i}} e^{\mathbf{i}\pi(s \mod 2^{n-1-i})\frac{2\alpha_i+1}{2^{n-1-i}}}|1\rangle\right)
\end{aligned}$$

Starting from $i = n-1$, we can directly determine the least significant bit of s. For $i = n-2$, we then know $(s \mod 2^{n-1-i}) = s_0$, meaning that we can apply a phase correction of angle

$$-\pi s_0 \frac{2\alpha_i+1}{2^{n-1-i}}$$

on $\left|\psi_{(2\alpha_i+1)2^{n-1-i}}\right\rangle$, giving the state $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{s_{n-1-i}}|1\rangle)$, which will give us s_1. Iterating this process, we will be able to determine the whole secret, as shown by Algorithm 6.

Algorithm 6 Algorithm to retrieve s from n coset states

Require: $\forall i \in [\![0, n-1]\!]$ a phase vector $\left|\psi_{(2\alpha_i+1)2^i}\right\rangle$, where $\alpha_i \in [\![0, 2^i - 1]\!]$.
Ensure: The secret s.
1: From $|\psi_{2^{n-1}}\rangle$, retrieve s_0 by measuring the state in the $|\pm\rangle$ basis
2: **for** i from $n-2$ to 0 **do**
3: Use the bits $s_0, \ldots, s_{i-1}$ to apply a phase correction of angle

$$-\pi(s \mod 2^{n-1-i})\frac{2\alpha+1}{2^{n-1-i}}$$

on $\left|\psi_{(2\alpha_i+1)2^i}\right\rangle$
4: Measure the corrected state in the $|\pm\rangle$ basis to retrieve s_{n-1-i}
5: **return** s

Tweaking a bit Kuperberg's algorithm to keep states $|\psi_{k_i}\rangle$ (as described above) on the way to produce $|\psi_{k_{n-1}}\rangle = \left|\psi_{N/2}\right\rangle$, Bonnetain and Naya-Plasencia then proposed Algorithm 7.

Algorithm 7 Variant of Kuperberg's algorithm to obtain s in one pass

Require: 2^ℓ phase vectors as in Equation 4.2, where each copy has $k \in \mathbb{Z}_N$ chosen independently and uniformly at random.
Ensure: The secret s.
1: Separate the coset states in pools P_i, where $P_i \stackrel{\text{def}}{=} \{|\psi_k\rangle : 2^i | k, 2^{i+1} \not| k\}$
2: **for** i from 0 to $n-2$ **do**
3: **while** $|P_i| \geq 3$ **do**
4: Pop $|\psi_a\rangle$ and $|\psi_b\rangle$ of P_i such that $a+b$ or $a-b$ has the highest possible divisibility by 2 (and is not 0)
5: Combine $|\psi_a\rangle$ and $|\psi_b\rangle$ and insert the resulting state in the appropriate pool
6: **if** $\forall i \in [\![0, n-1]\!], P_i \neq \emptyset$ **then** Pick one element in each P_i, apply Algorithm 6 and **return** s
7: **return** "The algorithm failed"

We will come back on Algorithm 7 with more details in Chapter 6, as it was the inspiration for a new algorithm for solving the DCP.

We will now dive deeper into the parameters of Kuperberg's algorithm and its complexity.

2.2 Complexity

Kuperberg's algorithm has subexponential query, time and space complexities, as stated by the following theorem

Theorem 4.2 (Theorem 3.1 of [Kup05]). *Letting $m = \lceil \sqrt{n} \rceil$, Algorithm 5 then requires $O\left(2^{3\sqrt{n}}\right)$ quantum queries and space, and $\tilde{O}\left(2^{3\sqrt{n}}\right)$ computation time.*

Sketch of proof (inspired by [Chi22]). The parameters required for the algorithm to work can be estimated as follows. The combination step takes two phase vectors and produces one. This state is of interest for our purpose with probability 1/2. It means than Step 3 of Algorithm 5 divides by 4 the size of the list of phase vectors we have in Step 2. We also notice that on average, 2^{m-1} phase vectors could be discarded at Step 2, since we are working on m bits and we will typically have an odd number of phase vectors with probability 1/2 in each sublist (*i.e.*, one unmatched state per sublist). Thus, at the end of the combination routine with $j = 0$, we have on average $2^{\ell-2} - 2^{m-1}$ remaining phase vectors. In turn, when $j = 1$, we get $2^{\ell-4} - 2^{m-3} - 2^{m-1}$ states, and so on. More generally, at the end of the combination routine for j going from 0 to $m-2$, we have

$$2^{\ell-2(j+1)} - \sum_{i=0}^{j} 2^{m-1-2i} = 2^{\ell-2(j+1)} + \frac{2^{m-1-2j} - 2^{m+1}}{3}$$

phase vectors on average. At the beginning of the last step, *i.e.*, for $j = m-1$, we have $2^{\ell-2m} + \frac{2^{m-1-2(m-1)} - 2^{m+1}}{3}$ states. It remains $n - 1 - jm$ bits to put to 0, meaning that on average, 2^{n-2-jm} phase vectors will be discarded. We end up with

$$2^{\ell-2m} + \frac{2^{-m+1} - 2^{m+1}}{3} - 2^{n-2-(m-1)m}$$

states. These states will be of the form $|\psi_{2^{n-1}}\rangle$ or $|\psi_0\rangle$. In order to simplify the calculations, we will ask to have 2^{m-1} states at the end of the algorithm, even if we could ask for less. We first give a lower bound on the number of phase vectors we previously computed and then we will check for when it is greater than 2^{m-1}.

$$2^{\ell-2m} + \frac{2^{-m+1} - 2^{m+1}}{3} - 2^{n-2-(m-1)m} \geq 2^{\ell-2m} - 2^{m-1} - 2^{n-2-(m-1)m}$$

We now look for a lower bound for the polynomial $n - 2 - (m-1)m$. We get it for $m = \frac{\sqrt{4n-7}+1}{2}$, which means that $m \approx \sqrt{n}$. We will keep this value in mind but will keep up with the notation m for the sake of simplicity. We now have

$$2^{\ell-2m} + \frac{2^{-m+1} - 2^{m+1}}{3} - 2^{n-2-(m-1)m} \geq 2^{\ell-2m} - 2^{m-1} - 1$$

Now if we ask for this expression to be greater than 2^{m-1},

$$\begin{aligned} 2^{\ell-2m} - 2^{m-1} - 1 &\geq 2^{m-1} \\ 2^{\ell-2m} &\geq 2^m + 1 \\ 2^\ell &\geq 2^{3m} + 2^{2m} \end{aligned}$$

Thus, ℓ should be greater than $3m+1$, *i.e.*, $\ell \approx 3\sqrt{n}$. □

It is in fact possible to study the complexity of Kuperberg's algorithm more precisely and show that its complexity in terms of queries, classical and quantum time and classical and quantum space is a $\tilde{O}\left(2^{\sqrt{2\log(3)n}}\right)$ (see [Kup05], Theorem 5.1). Simulations have been carried out by Bonnetain and Schrottenloher to study the practical complexity of this algorithm [BS20]. They obtained a query complexity close to $12 \times 2^{1.8\sqrt{n}}$ (note that $\sqrt{2\log 3} \simeq 1.78$).

3 Regev's algorithm

Kuperberg's first algorithm requires to store, at each time, a subexponential number of phase vectors; thus, it has subexponential quantum memory complexity. Regev [Reg04a] modified the combination routine to reduce the number of qubits to polynomial, while keeping the time complexity subexponential.

The combination routine in question takes a number of, say ℓ, phase vectors that have in common that their a least significant bits are zero, and combines them to form a new one that has its $a+r$ least significant bits set to zero (with of course $r > 0$). From this method, Regev constructs an algorithm that makes calls to the oracle until it has ℓ phase vectors that share the same number of least significant bits equal to zero. In this way, we can represent the process as a system of communicating vessels. The first vase, once full of elements that have no particular structure, builds one that will belong to the second vase, and so on. This idea for keeping the space usage of the resolution of the DCP low is described in Figure 4.3.

Now let us focus on the combination routine. It combines ℓ phase vectors for a well-chosen ℓ (to minimize the overall complexity) in the following way. Let B be some chosen, arbitrary value. We start with ℓ phase vectors $|\psi_{\kappa_1}\rangle, \ldots, |\psi_{\kappa_\ell}\rangle$ and we let $\boldsymbol{\kappa} \stackrel{\text{def}}{=} (\kappa_1, \ldots, \kappa_\ell)$. We tensor these phase vectors, giving us the superposition:

$$|\psi_{\boldsymbol{\kappa}}\rangle \stackrel{\text{def}}{=} \bigotimes_{i=1}^{\ell} |\psi_{\kappa_i}\rangle = \sum_{\mathbf{b}\in\mathbb{F}_2^\ell} \omega_N^{s\mathbf{b}\cdot\boldsymbol{\kappa}} |\mathbf{b}\rangle \tag{4.3}$$

We then compute $\mathbf{b} \cdot \boldsymbol{\kappa} \bmod B$ in an ancillary register, and measure it, which gives us a value z. This projects the superposition on the vectors $\mathbf{b}$ such that $\mathbf{b} \cdot \boldsymbol{\kappa} \bmod B = z$. We choose ℓ and the size of B such that on average two solutions $\mathbf{b}_0$ and $\mathbf{b}_1$ occur. The state becomes proportional to:

$$|\mathbf{b}_0\rangle + \omega_N^{s(\mathbf{b}_1 - \mathbf{b}_0)\cdot\boldsymbol{\kappa}} |\mathbf{b}_1\rangle$$

If there are more than two solutions, we choose two and project onto them (the exact process will be described later). Finally, we remap $\mathbf{b}_0, \mathbf{b}_1$ to $0, 1$ respectively. We have obtained a phase vector $|\psi_\kappa\rangle$ with a label $\kappa = (\mathbf{b}_1 - \mathbf{b}_0) \cdot \boldsymbol{\kappa} \leq B$, *i.e.*, by definition of $\mathbf{b}_0$ and $\mathbf{b}_1$, κ has now its last $\log(B)$ bits equal to zero. Then, step by step (actually, thanks to the process described before and which is illustrated in Figure 4.3 with this combination routine plugged in), we can build phase vectors whose labels have more and more of their least significant bits zeroed out until we obtain the label $N/2 = 2^{n-1}$.

Regev [Reg04a] and later Childs, Jao and Soukharev [CJS14] used this combination routine to get an algorithm with $\tilde{O}\left(2^{\sqrt{2n \log n}}\right)$ queries and $O(n)$ quantum memory. In the following sections, we give a more detailed description of the algorithm and an analysis of its complexity.

3.1 Algorithm

We provide in Algorithm 8 the pseudocode for Regev's combination routine.

The projective measurement in Step 5 can be done by marking the vectors of interest thanks to an unitary operator $\mathbf{U}$ defined as:

$$\mathbf{U}|\mathbf{b}\rangle|0\rangle \mapsto |\mathbf{b}\rangle|\beta\rangle \qquad \text{where}$$

$$\beta = \begin{cases} 1 & \text{if} \quad \mathbf{b} \in \{\mathbf{b}_0, \mathbf{b}_1\} \\ 0 & \text{otherwise} \end{cases}$$

and second, measuring this ancillary qubit β. If 1 is measured, we succeeded to project the superposition of t elements on the two we targeted and we can go to Step 6 of the algorithm. On the other hand, if 0 is measured, then we obtained a superposition of all the solutions except the two we wanted to project on. Thus, we pick two other solutions and restart the projection process with these two vectors. This process has some small probability to fail if t is odd that we can upperbound by $1/3$.

The final relabeling step is quite simple and can be achieved in the following way:

$$\begin{aligned} & (\omega_N^{s\mathbf{b}_0\cdot\boldsymbol{\kappa}} |\mathbf{b}_0\rangle + \omega_N^{s\mathbf{b}_1\cdot\boldsymbol{\kappa}} |\mathbf{b}_1\rangle)|0\rangle \\ \xrightarrow{\mathbf{U}} \quad & \omega_N^{s\mathbf{b}_0\cdot\boldsymbol{\kappa}} |\mathbf{b}_0\rangle|0\rangle + \omega_N^{s\mathbf{b}_1\cdot\boldsymbol{\kappa}} |\mathbf{b}_1\rangle|1\rangle \\ \xrightarrow{\mathbf{V}} \quad & |\mathbf{0}\rangle (\omega_N^{s\mathbf{b}_0\cdot\boldsymbol{\kappa}} |0\rangle + \omega_N^{s\mathbf{b}_1\cdot\boldsymbol{\kappa}} |1\rangle) \end{aligned}$$

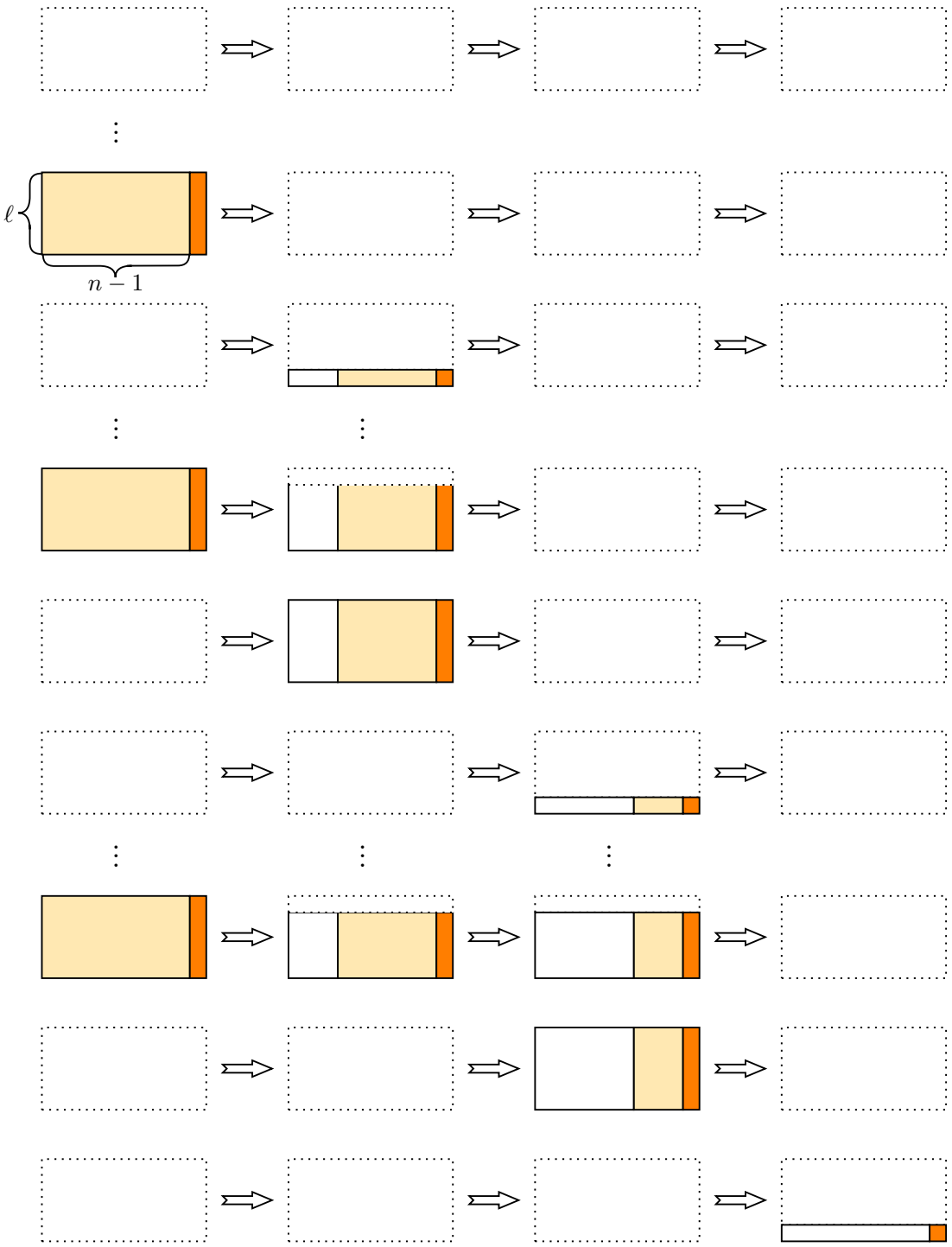

Figure 4.3: Diagram showing the system of communicating vessels that enabled Regev to design a polynomial-space algorithm to solve the DCP. The dotted boxes have a capacity of ℓ phase vectors and must be filled so that their contents can be used to construct a new phase vector that will (most likely) belong to the next box. The arrows represent the combination process in question. Calls are made to the oracle until our vase system has filled the penultimate vase, allowing us to construct a solution state.

Algorithm 8 Regev's Combination Routine

Require: a parameter r and ℓ phase vectors as in Equation 4.2 such that $\forall j \in [\![1, \ell]\!], 2^a | \kappa_j$

Ensure: a state $|\psi_\kappa\rangle$ such that $2^{a+r} | \kappa$

1: Build $|\psi_{\boldsymbol{\kappa}}\rangle$ where $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_\ell)$

$$\sum_{\mathbf{b} \in \mathbb{F}_2^\ell} \omega_N^{s\mathbf{b}\cdot\boldsymbol{\kappa}} |\mathbf{b}\rangle$$

2: Compute the function $\boldsymbol{\kappa} \mapsto \mathbf{b} \cdot \boldsymbol{\kappa} \bmod 2^{a+r}$ in an ancillary register

$$\sum_{\mathbf{b} \in \mathbb{F}_2^\ell} \omega_N^{s\mathbf{b}\cdot\boldsymbol{\kappa}} |\mathbf{b}\rangle \left| \mathbf{b} \cdot \boldsymbol{\kappa} \bmod 2^{a+r} \right\rangle$$

3: Measure the ancillary register. The state collapses to:

$$\sum_{\substack{\mathbf{b} \in \mathbb{F}_2^\ell : \\ \mathbf{b}\cdot\boldsymbol{\kappa} \bmod 2^{a+r} = z}} \omega_N^{s\mathbf{b}\cdot\boldsymbol{\kappa}} |\mathbf{b}\rangle |z\rangle$$

4: Compute $\left\{ \mathbf{b} \in \mathbb{F}_2^\ell \colon \mathbf{b} \cdot \boldsymbol{\kappa} \bmod 2^{a+r} = z \right\}$, of size t. Denote by $\mathbf{b}_i$ the elements of this set. The state can be rewritten as

$$\sum_{i=0}^{t-1} \omega_N^{s\mathbf{b}_i\cdot\boldsymbol{\kappa}} |\mathbf{b}_i\rangle |z\rangle$$

5: A projective measurement on a couple of solutions $(|\mathbf{b}_0\rangle, |\mathbf{b}_1\rangle)$ gives

$$\frac{1}{\sqrt{2}} \left(\omega_N^{s\mathbf{b}_0\cdot\boldsymbol{\kappa}} |\mathbf{b}_0\rangle + \omega_N^{s\mathbf{b}_1\cdot\boldsymbol{\kappa}} |\mathbf{b}_1\rangle \right)$$

6: **return** The relabeled state

$$\frac{1}{\sqrt{2}} \left(|0\rangle + \omega_N^{s(\mathbf{b}_1 - \mathbf{b}_0)\cdot\boldsymbol{\kappa}} |1\rangle \right)$$

where the operators $\mathbf{U}$ and $\mathbf{V}$ are defined as follows:

$$\mathbf{U} |\mathbf{b}_b\rangle |0\rangle \mapsto |\mathbf{b}_b\rangle |b\rangle \qquad \text{and} \qquad \mathbf{V} |\mathbf{b}_b\rangle |b\rangle \mapsto |0\rangle |b\rangle \qquad \text{for } b \in \{0, 1\}.$$

Recall that the implementation of the operator $\mathbf{V}$ is possible because $\mathbf{b}_0$ and $\mathbf{b}_1$ are classicaly known.

3.2 Complexity

Regev's algorithm has subexponential query and time complexities but only needs a polynomial space, as stated by the following theorem.

Theorem 4.3 (See [Reg04a]). *Letting $\ell = r + 4 = O\left(\sqrt{n \log n}\right)$, Regev's algorithm then requires $2^{O\left(\sqrt{n \log n}\right)}$ queries and time, and polynomial (in n) space.*

Sketch of proof (given by [Reg04a]). Since each routine in the process sets $O\left(\sqrt{n \log n}\right)$ new bits to zero, we will need $O\left(\sqrt{\frac{n}{\log n}}\right)$ routines in the pipeline, described in Figure 4.3. We won't prove it here but it can be shown that the combination routine is successful with constant probability (see for example [Reg04a; Bon19a]). Then, inputting a total of $\ell\sqrt{\frac{n}{\log n}}$ phase vectors suffices for the algorithm to output a solution with very high probability. This quantity is of the same order as $2^{O\left(\sqrt{n \log n}\right)}$, hence the query complexity estimate. Finally, the time complexity is of the same order since we can bound the time taken by the routine combination by $2^{O(\ell)}$. □

Regev's study of the complexity of his own algorithm, which was fairly superficial, was taken up in detail by Childs, Jao and Soukharev. They proved the following result.

Theorem 4.4 ([CJS14]). *Regev's algorithm actually requires $2^{\left(\frac{1}{\sqrt{2}}+o(1)\right)\sqrt{n \log n}}$ queries and quantum time, $2^{(\sqrt{2}+o(1))\sqrt{n \log n}}$ classical time, and polynomial (in n) space.*

There are a number of possible tradeoffs, which can be achieved by adjusting the value of ℓ. This was the subject of a study by Bonnetain, and we refer to his paper on the subject [Bon19b] as well as his PhD thesis [Bon19a].

Low Queries Variant Regev's combination method, described by Algorithm 8, is used in a communicating vessel system. The more successive rounds there are in this system, the fewer vectors are required as input to the combination method. And vice versa. When we want to optimize the algorithm in terms of time, we obtain Theorem 4.3, with what is usually called Regev's algorithm, which uses a polynomial space and $2^{O\left(\sqrt{n \log n}\right)}$ queries and time. But if we take our system of communicating vessels to have only one round, we obtain an algorithm that takes $\ell = n$ vectors and directly produces the one of interest (*i.e.*, the one with $k = N/2$) at the cost of solving a subset-sum problem on n-bit numbers. The result is an algorithm with exponential complexity in classical time, but which uses only $O(n)$ queries to find a bit of the secret, hence a quadratic number (in n) of queries to find the whole secret. This limiting case of Regev's algorithm was described and studied by Bonnetain and Schrottenloher in [BS20]. It will be our starting point to design a new algorithm which reduces the DCP to an instance of quantum subset-sum instead of a classical one in Chapter 5.

4 Kuperberg's second algorithm

Like the two previous algorithms, Kuperberg's *collimation sieve* [Kup13] is a hybrid quantum/classical procedure starting from the initial phase vectors, where we need to perform both quantum computations which create new vectors, and classical computations which give their description. It improves on the time complexity of Regev's algorithm and his own first algorithm, while retaining its use in quantum space polynomial. It is inspired by Regev's algorithm and works in the same way with tensored phase vectors. However, in order to control these new phase vectors, we need to know the list of all their labels. These lists will become of subexponential size, although the vector itself requires only a polynomial amount of qubits. This is why the algorithm combines a polynomial quantum memory with a subexponential *classical* memory, while Regev's algorithm needed polynomial classical and quantum space.

4.1 Algorithm

To introduce Kuperberg's second algorithm, we begin by rewriting Regev's algorithm in a slightly different way. Specifically, instead of indexing the phase vector with a vector $\boldsymbol{\kappa}$ of size ℓ and writing the algorithm with all the scalar products $\boldsymbol{\kappa} \cdot \mathbf{b}$ for $\mathbf{b} \in \mathbb{F}_2^\ell$, we list all the scalar products directly as a vector $\mathbf{k} = (k_1, \cdots, k_{\mathscr{L}})$ with in this case $\mathscr{L} = 2^\ell$, and then directly work with the values k_i in this version of the algorithm. Namely, we can give a more general definition of phase vectors, as the superposition:

$$|\psi_{\mathbf{k}}\rangle \stackrel{\text{def}}{=} \sum_{i \in [\![1, \mathscr{L}]\!]} \omega_N^{sk_i} |i\rangle \tag{4.4}$$

We can now present Kuperberg's second algorithm combination routine, which uses this more general definition to generalize in some sense Regev's algorithm. The first main difference is that the former does not necessarily reduce the list of labels down to 2 in the end, contrary to the latter. In other words, where Regev uses a vector of length $\mathscr{L}$ to produce one of length 2, Kuperberg uses two of length $\mathscr{L}$ and $\mathscr{L}'$ to produce one of length $\mathscr{L}$" (where in practice $\mathscr{L} \simeq \mathscr{L}' \simeq \mathscr{L}$"), as shown in Algorithm 9. The second main difference is that while Regev uses a vector of size $\mathscr{L} = 2^\ell$ for some ℓ, Kuperberg uses (and produces) vectors of size $\mathscr{L}$ which can be any number, not necessarily a power of 2.

Starting from a certain set of phase vectors, we can identify them with the classical lists of their labels. The combination step operates on these lists like a purely classical list-merging algorithm, in which new lists of labels are formed from the pairs of labels satisfying a certain condition. This algorithm can be represented as a *merging tree* in which all nodes are lists of labels (resp. phase vectors). On the classical side, Kuperberg's algorithm is thus similar to Wagner's generalized birthday algorithm [Wag02], which is a

Algorithm 9 Combination routine in the collimation sieve.

Require: a parameter r and 2 phase vectors $|\psi_{\mathbf{k}}\rangle$ and $|\psi_{\mathbf{k}'}\rangle$ of respective length $\mathscr{L}$ and $\mathscr{L}'$, such that for some a, $\forall i \in [\![1, \mathscr{L}]\!]$, $2^a | k_i$ and $\forall j \in [\![1, \mathscr{L}']\!]$, $2^a | k'_j$

Ensure: a phase vector $|\psi_{\mathbf{v}}\rangle$ of length $\mathscr{L}$", such that $\forall i \in [\![1, \mathscr{L}"]\!], 2^{a+r} | v_i$

1: Tensor $|\psi_{\mathbf{k}}\rangle$ and $|\psi_{\mathbf{k}'}\rangle$

$$|\psi_{\mathbf{k}}\rangle \otimes |\psi_{\mathbf{k}'}\rangle = \sum_{i \in [\![1,\mathscr{L}]\!]} \sum_{j \in [\![1,\mathscr{L}']\!]} \omega_N^{s(k_i + k'_j)} |i\rangle |j\rangle$$

2: Compute the function $(i, j) \mapsto k_i + k'_j \bmod 2^{a+r}$ into an ancillary register

$$\sum_{i \in [\![1,\mathscr{L}]\!]} \sum_{j \in [\![1,\mathscr{L}']\!]} \omega_N^{s(k_i + k'_j)} |i\rangle |j\rangle \left| k_i + k'_j \bmod 2^{a+r} \right\rangle$$

3: Measure the ancillary register. The state collapses to:

$$\sum_{\substack{i \in [\![1,\mathscr{L}]\!], j \in [\![1,\mathscr{L}']\!] \\ k_i + k'_j \bmod 2^{a+r} = z}} \omega_N^{s(k_i + k'_j)} |i\rangle |j\rangle |z\rangle$$

4: Compute $\left\{(i, j) \in [\![1, \mathscr{L}]\!] \times [\![1, \mathscr{L}']\!] : k_i + k'_j \bmod 2^{a+r} = z\right\}$, of size $\mathscr{L}$".

5: Apply to the state a transformation that maps the pairs (i, j) to $[\![1, \mathscr{L}"]\!]$.

6: Return the state and the vector of corresponding labels $k_i + k'_j$.

binary merging tree of depth $\sqrt{n}$. In Wagner's algorithm, the goal is to impose stronger conditions at each level which culminate in a full-zero sum. Here, the same conditions are imposed on the labels in the phase vectors. A success in the list-merging routine is equivalent to a success in the collimation routine (we obtain a phase vector with the wanted label). The query, time and memory complexities depend on the shape of the tree. Even though the conditions are actually chosen at random at the measurement step in Algorithm 9, we can consider them chosen at random *before* the combination to analyze the algorithm. The merging tree will typically have the shape pictured in Figure 4.4.

4.2 Complexity

In the following lemma, we give the complexity of Kuperberg's second algorithm.

Lemma 4.5. *Kuperberg's second algorithm requires on average* $O\left(\sqrt{2n}2^{\sqrt{2n}}\right)$ *(classical and quantum) time,* $O\left(2^{\sqrt{2n}}\right)$ *queries and classical memory, and* $\mathsf{poly}(n)$ *qubits.*

Proof. The optimal time complexity is obtained with a tree (as described in Figure 4.4)

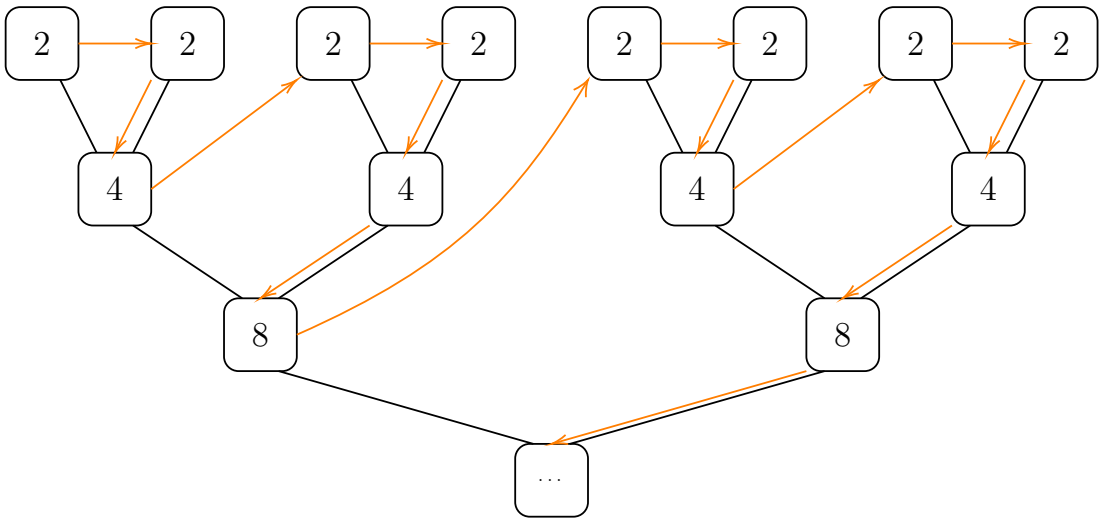

Figure 4.4: We start from the leaves of the tree, which correspond to phase vectors of length $\mathscr{L} = 2$, towards the root, where the solution vector will be found. At the i-th level of the tree, we form pairs of phase vectors of length (approximately) 2^i, which we combine two by two, to form new ones of length (approximately) 2^{i+1} that will belong to the $(i+1)$-th level. If we were to process this tree level by level, we would not only have to store 2^{d+1} classical indices of the phase vectors, where d is the depth of the tree, but also all the phase vectors themselves. A better strategy exists, and it is a kind of reverse depth-first search. This method allows us to store at most a reasonable number of phase vectors at any time, *i.e.*, use $\mathsf{poly}(n)$ qubits.

as follows. It starts with lists of size 2, *i.e.*, two-labeled phase vectors. At level i starting from the leaves, the lists have (expected) size 2^i, and they are merged pairwise into a list of size 2^{i+1}. This means that when combining phase vectors of the i-th level of the tree, we can eliminate $2i - (i+1) = i - 1$ bits. So the depth d of the tree should be such that:

$$\sum_{i=1}^{d-1}(i-1) = n$$

i.e., d should be close to $\sqrt{2n}$. We directly deduce that there are in total $2^{\sqrt{2n}}$ leaves (hence queries to do). The (classical) cost of merging, over the whole tree, is equal to the sum of all list sizes. It is also the (quantum) cost of the relabeling operations:

$$\sum_{i=1}^{d} 2^{\sqrt{2n}-i} \times 2^i = O\left(\sqrt{2n}2^{\sqrt{2n}}\right)$$

To compute the memory complexity, one must note that it is not required to store whole levels of the merging tree. Instead, we compute the lists (resp. the phase vectors)

depth-first, and store only one node of each level at most, *i.e.*, d nodes in total, as we can prove by induction. Indeed, if $d = 2$, we will have to store the two initial phase vectors to produce a final one, so we will store at most d phase vectors. Assume this is true for a tree of i levels. It is then also true for a tree of $i + 1$ levels, since we will store the root of one subtree of i levels while we have to go through the whole other subtree, *i.e.* store at most i additional nodes, until we arrive to its root and combine the result with the root of the former subtree. In the end this means that we will have to store at most around $\sqrt{2n}$ phase vectors at any time. For the same reason, the classical memory complexity is $O\left(2^{\sqrt{2n}}\right)$. □

Up to this point, the analysis was only performed on average, and in practice, there is a significant variance in the list sizes in the tree. More precise analyses were performed in [Pei20; Chá+22]. It follows from them that the list size after merging should be considered smaller than the expected one by an "adjusting factor" $\sqrt{3/(2\pi)}$. Furthermore, the combination may create lists that are too large, which must be discarded. The empirical analysis of Peikert [Pei20] gives a factor $(1 - \delta)$ of loss at each level, with $\delta = 0.02$.

The smaller factor in list sizes simply means that at level i, we will not exactly eliminate $i - 1$ bits, but $i - c$ where $c = \log_2\left(1 + \sqrt{\frac{3}{2\pi}}\right) \simeq 0.76$. (We can control the interval size in Algorithm 9 very precisely.) Thus d is in fact solution to:

$$\sum_{i=1}^{d}(i - c) = n \implies \frac{d^2}{2} - ch = n \implies d \simeq c + \sqrt{2n + 4c^2} \ .$$

Finally, the loss at each level induces a global multiplicative factor $(1 - \delta)^{-d} = 2^{-\log_2(1-\delta)d} \simeq 2^{0.029d}$ on the complexity. Therefore, accounting for the adjusting factor and discards, the query complexity of the sieve is close to:

$$2^{1.029\left(0.76+\sqrt{2n+2.30}\right)} \tag{4.5}$$

and the quantum time complexity multiplies this by a factor $0.76 + \sqrt{2n + 2.30}$. The difference with the exact $2^{\sqrt{2n}}$ is not negligible, but not large either. At $n = 4608$, the two exponents are respectively 99.6 and 96.

This analysis applies if we want to obtain a specific label, *e.g.*, the label 1. Afterwards, the algorithm can be repeated n times. For a generic N (not a power of 2), one typically produces all labels which are powers of 2 and uses a QFT to directly recover the secret. This is done for example in [BS20] but Peikert [Pei20] proposed a more advanced method to recover multiple bits of the secret with each phase vector.

Lemma 4.5 gives the complexity of finding one bit of the secret. We give an important formula for computing the complexities for finding the whole secret from this lemma.

Lemma 4.6. *Let $\alpha > 0$ and n be a positive integer. We have*

$$\sum_{i=1}^{n} 2^{\alpha\sqrt{i}} = O\left(\sqrt{n}2^{\alpha\sqrt{n}}\right).$$

Proof. When i is a perfect square, let say $i = j^2$, we have that $2^{\alpha\sqrt{i}} = 2^{\alpha j}$. Now for any i between the two perfect squares $(j-1)^2$ and j^2, we have the upper bound $2^{\alpha\sqrt{i}} < 2^{\alpha j}$. In order to use this, we rewrite the sum:

$$\begin{aligned}
\sum_{i=1}^{n} 2^{\alpha\sqrt{i}} &\leq \sum_{j=0}^{\lceil\sqrt{n}\rceil-1} \sum_{k=j^2+1}^{(j+1)^2} 2^{\alpha\sqrt{k}} \\
&\leq \sum_{j=0}^{\lceil\sqrt{n}\rceil-1} \sum_{k=j^2+1}^{(j+1)^2} 2^{\alpha(j+1)} \\
&= \sum_{j=0}^{\lceil\sqrt{n}\rceil-1} (2j+1)2^{\alpha(j+1)}
\end{aligned}$$

Using the formula for geometric series, we obtain:

$$\begin{aligned}
\sum_{i=1}^{n} 2^{\alpha\sqrt{i}} &\leq 2^{\alpha+1}\frac{(2^\alpha-1)\lceil\sqrt{n}\rceil 2^{\alpha\lceil\sqrt{n}\rceil} - 2^\alpha(2^{\alpha\lceil\sqrt{n}\rceil}-1)}{(2^\alpha-1)^2} + 2^\alpha\frac{2^{\alpha\lceil\sqrt{n}\rceil}-1}{2^\alpha-1} \\
&= \frac{2^\alpha}{2^\alpha-1}\left((2\lceil\sqrt{n}\rceil+1)2^{\alpha\lceil\sqrt{n}\rceil} - \frac{2^{\alpha+1}}{2^\alpha-1}(2^{\alpha\lceil\sqrt{n}\rceil}-1) - 1\right) \\
&\leq \frac{2^\alpha}{2^\alpha-1}(2\lceil\sqrt{n}\rceil+1)2^{\alpha\lceil\sqrt{n}\rceil} \quad .
\end{aligned}$$

which allows us to conclude the proof, α being fixed. □

In Chapter 5, we will consider the task of obtaining labels which, instead of reaching a prescribed k, match k on a certain number of bits only (we can say that the phase vectors are *partially collimated*), let say i: this complexity is of order $2^{\sqrt{2i}}$. By Lemma 4.6, we can obtain a sequence of i phase vectors collimated on $1, \ldots, i$ bits with a query complexity (by Lemma 4.6):

$$\sum_{j=1}^{i} 2^{\sqrt{2j}} = O\left(\sqrt{i}2^{\sqrt{2i}}\right)$$

Chapter 5
A Query Interpolation Algorithm

Let $n \stackrel{\text{def}}{=} \lceil \log N \rceil$. In this chapter, based on [RST23], we first propose a new algorithm for solving the DCP using a linear number of queries. It is somewhat analogous to Regev's algorithm where instead of reducing the DCP to a classical subset-sum problem, it reduces the DCP to a *quantum* subset-sum problem. In the first case, the algorithm makes $O(n)$ queries to find one bit of the secret, meaning it has to be iterated $O(n)$ times. With this new algorithm, which is inspired by [Reg02; Ste+09], we only need $O(1)$ quantum subset-sum instances, *i.e.*, $O(n)$ queries, to find the whole secret.

Second, we present a simple and natural method of interpolation between Kuperberg's second algorithm (which is the state of the art) and the new algorithm we mentioned above. It consists in using Kuperberg's algorithm to more or less preprocess the states given as input to our algorithm. The difficulty of solving the inherent quantum subset-sum problem instance will depend on the preprocessing step.

Finally, as a building block of our algorithms, we study quantum subset-sum algorithms when the problem to solve is partially in superposition. We show here that we can still improve over Grover's search even under the constraint of a polynomial quantum memory, using an exponential classical memory, with or without quantum access. Specifically, we show that the QRACM-based algorithm of [Bon+20] adapts to this case and reaches a complexity $\tilde{O}\left(2^{0.2356n}\right)$. Without QRACM, we reach a quantum time $\tilde{O}\left(2^{0.4165n}\right)$ using $O\left(2^{0.2334n}\right)$ bits of classical memory, improving over a previous algorithm by Helm and May [HM20]. In both cases, we also give non-asymptotic estimates of their complexity.

All together, we can summarize the complexity exponents of the different algorithms for solving the DCP in Table 5.1, including the new one we propose. Note that this is a corrected version of the same table appearing in the paper [RST23] (same for the table that will follow in the next paragraph where some evaluations are given). In this table and in this chapter in general, Regev's algorithm refers to the low-queries variant described in Section 3.2, and 0.283 is the best asymptotic exponent that we can obtain for classical subset-sum algorithms known at the moment [Bon+20], if there are no constraints on the memory.

Table 5.1: Complexity exponents of algorithms for finding the whole secret s.

Algorithm	Queries	Classical Time	Quantum Time	Classical Space
Kuperberg II	$\sqrt{2n} + \frac{1}{2}\log n + 3$	$\sqrt{2n} + \frac{1}{2}\log n + 3$	$\sqrt{2n} + \frac{1}{2}\log n + 3$	$\sqrt{2n}$
Regev	$2\log n + 3$	$0.283n + 3$	$2\log n + 3$	$0.283n$
Ettinger-Hoyer	$\log n + 6.5$	n	$\log n + 6.5$	$\log n$
Alg. 12 w/ QRACM	$\log n + 3$	$0.238n + 12$	$0.238n + \frac{3}{2}\log n + 12$	$0.238n$
Alg. 12 w/o QRACM	$\log n + 3$	$< 0.232n$	$0.418n + \frac{3}{2}\log n + 15.5$	$< 0.232n$

We propose two versions of our algorithm, one with QRACM and one without, both using polynomial quantum space. Note that our algorithm with QRACM outperforms other algorithms using a linear number of queries when we look at the complexity in classical time + quantum time.

Impact on CSIDH. Although Kuperberg's second algorithm is the one with the best time complexity for solving the DCP, it is still interesting to look at algorithms that only use a linear number of queries, since for example, CSIDH cryptanalysis via the resolution of the DCP involves the use of a very expensive oracle.

We give in Table 5.2 a few examples of complexity exponents for parameters of CSIDH.

Table 5.2: Complexity exponents for some parameters of CSIDH, computed from the expressions given in Table 5.1. The quantum space is polynomial in n.

	Algorithm	Queries	Classical Time	Quantum Time	Classical Space
CSIDH-512	Regev	19	76	19	73
($n = 256$)	Alg. 12 w/ QRACM	11	73	85	61
CSIDH-1024	Regev	21	148	21	145
($n = 512$)	Alg. 12 w/ QRACM	12	134	148	122
CSIDH-1792	Regev	23	257	23	254
($n = 896$)	Alg. 12 w/ QRACM	13	226	240	214
CSIDH-3072	Regev	25	438	25	435
($n = 1536$)	Alg. 12 w/ QRACM	14	378	394	366
CSIDH-4096	Regev	25	583	25	580
($n = 2048$)	Alg. 12 w/ QRACM	14	500	516	488

Organization. In Section 1, we give some preliminaries on subset-sum algorithms that we will use as black boxes afterwards. In Section 2, we recall the reduction from the DCP to the subset-sum problem, and introduce our new idea of using a *quantum* subset-sum solver. Our interpolation between the sieving and subset-sum approaches is detailed in Section 3. Finally, our contributions on quantum subset-sum algorithms and the details of the black boxes that we used in the chapter are provided in Section 4.

Contents

1 Preliminaries

Quantum Memory. We work with different types of memory:

- quantum memory (*i.e.*, qubits): some DCP algorithms (*e.g.*, Kuperberg's first algorithm [Kup05]) need to store many coset states, which creates a subexponential quantum memory requirement;

- classical memory with quantum random-access (QRACM): the QRACM (or qRAM, QROM in some papers) is a specialized hardware which stores classical data and accesses this data in quantum superposition. That is, we assume that given a classical memory of M bits $y_0, \ldots, y_{M-1}$, the following unitary operation:

$$|x\rangle |i\rangle \xmapsto{\mathsf{Access}} |x \oplus y_i\rangle |i\rangle$$

  can be implemented in time $O(1)$. QRACM is a very common assumption in quantum computing, and it appears in several works on the DCP [Kup13; Pei20] but also on collision-finding [BHT98] and subset-sum algorithms [Bon+20].

- classical memory without quantum random-access: the Access operation can be implemented in M arithmetic operations using a sequential circuit. This removes the QRACM assumption, and we fall back on the basic quantum circuit model. Some algorithms using QRACM can be re-optimized in a non-trivial way when memory access is costly, and this is the case of subset-sum [HM20].

The Subset-Sum Problem. As we saw in Chapter 4, the DCP can be reduced to the Subset-sum problem; this leads to the most query-efficient algorithms, and depending on the cost of queries, to the best optimization for some instances. We recall here the definition of this problem

Problem 5.1 (Subset-sum)**.** *A subset-sum instance is given by* $(v, \mathbf{k}), v \in \mathbb{Z}_N, \mathbf{k} \in \mathbb{Z}_N^m$ *for some modulus* N *and integer* m*. The problem is to find a vector (or all vectors)* $\mathbf{b} \in \{0,1\}^m$ *such that* $\mathbf{b} \cdot \mathbf{k} = v \mod N$*.*

When $m \simeq n = \lceil \log N \rceil$, there is one solution on average. The instance is said to be of *density one*. Heuristic classical and quantum algorithms based on the *representation technique* [HJ10; BCJ11] allow to solve it in exponential time in n. In the following, we will use these algorithms as black boxes. We first need a classical subset-sum solver.

Fact 3. *We have a classical algorithm* $\mathscr{S}^{\mathrm{C}}$ *which, on input a subset-sum instance* $(v, \mathbf{k})$ *of density one, finds all solutions. It has a time complexity in* $\tilde{O}\left(2^{c_{c\mathsf{SS}} n}\right)$ *where* $c_{c\mathsf{SS}} < 1$*.*

Here, the parameter $c_{c\mathsf{SS}}$ is the best asymptotic exponent that we can obtain for classical subset-sum algorithms. If there are no constraints on the memory, we can take $c_{c\mathsf{SS}} = 0.283$ which is the best value known at the moment [Bon+20].

In this paper, we will also need (quantum) algorithms solving a more difficult problem, in which $\mathbf{k}$ is fixed, but the target v is *in superposition*. We will call this type of algorithm a *quantum* subset-sum solver.

Fact 4. *We have a quantum algorithm* $\mathscr{S}^{\mathrm{Q}}$ *which has a complexity cost in* $\tilde{O}\left(2^{c_{q\mathsf{SS}} n}\right)$ *(where* $c_{q\mathsf{SS}} < 1$*), which, given an error bound* ε*, given a known (classical)* $\mathbf{k} \in \mathbb{Z}_N^m$ *and on input a quantum* v*, maps:*

$$|v\rangle |\mathbf{b}\rangle \mapsto |v\rangle \left|\mathbf{b} \oplus \mathscr{S}^{\mathrm{Q}}(v)\right\rangle$$

where, for a proportion at least $1 - \varepsilon$ *of all* v *admitting a solution,* $\mathscr{S}^{\mathrm{Q}}(v)$ *is selected uniformly at random from the solutions to the subset-sum problem, i.e., from the set* $\{\mathbf{b} \colon \mathbf{b} \cdot \mathbf{k} = v\}$*.*

Notice that in the way we implement the solver, we can only guarantee that it succeeds on a large proportion of inputs (there remains some probability of error). However, it depends on some precomputations that we can redo, to obtain a heuristically independent solver which allows to reduce ε and / or to ensure that we get more solutions.

Though we could implement the function $\mathscr{S}^{\mathrm{Q}}$ by running an available classical (or quantum) subset-sum algorithm, it would then require exponential amounts of qubits. Using only $\mathsf{poly}(n)$ qubits, we know for sure that $c_{q\mathsf{SS}} \leq 0.5$, because we can use Grover's algorithm to exhaustively search for a solution $\mathbf{b}$. This search uses $\mathsf{poly}(n)$ qubits

only. In Section 4, we will show that we can reach smaller values for $c_{q\mathsf{SS}}$, which differ depending on whether we allow QRACM or not.

2 Reducing the DCP to a Subset-sum Problem

Recall that we note $n = \lceil \log N \rceil$, where N is not necessarily a power of 2. We will focus in this section on two algorithms to solve the DCP: the first one (from Regev [Reg04a]) uses a classical subset-sum solver and the other (ours) uses a quantum one.

2.1 Using a Classical Subset-sum Solver

By reducing Regev's algorithm to a single level as described in Section 3.2, we can directly produce $\mathrm{lsb}(s)$ from n phase vectors. The pseudocode of the corresponding algorithm is given in Algorithm 10.

It can be proven that in Step 4, the number of solutions is quite small but generally enough for our purpose. In Step 5, the solution vectors we want to project our superposition on are marked in an ancillary register which is then measured. Either we will get what we want, or we will end up with a superposition of the solution vectors that were not marked, in which case we start the process again with two other solution vectors. For more details, we refer to the extensive study of Regev's algorithm by Childs, Jao and Soukharev [CJS14].

The following lemma gives us the complexity of Algorithm 10, derived from Regev's algorithm.

Lemma 5.1 (Subsection 3.3 [BS20])**.** *There exists an algorithm which finds* $\mathrm{lsb}(s)$ *with* $O(n)$ *queries and quantum time and space. It has the same usage in classical time and space as the subset-sum solver* $\mathscr{S}^{\mathrm{C}}$*.*

Algorithm 10 finds one bit of the secret. In order to retrieve the whole secret, we will have to repeat this procedure n times. Thus, we get an algorithm using a quadratic number of calls to the oracle, exponential classical time and space because of the subset-sum solver, linear quantum space and quadratic quantum time.

It turns out that we could solve the classical subset-sum problem on the side with a quantum computer, leading to some tradeoffs described in [Bon19b]. But we show hereafter that we can also build an algorithm which directly uses a quantum subset-sum solver instead of having to measure the ancillary register to get a classical instance of a subset-sum problem.

Algorithm 10 Finding $\mathrm{lsb}(s)$ using a classical subset-sum solver $\mathscr{S}^{\mathrm{C}}$

Require: $|\psi_{k_1}\rangle, \ldots, |\psi_{k_n}\rangle$ with $\mathbf{k} \stackrel{\text{def}}{=} (k_1 \ldots k_n) \in \mathbb{Z}_N^n$.
Ensure: $\mathrm{lsb}(s)$.

1: Tensor the phase vectors and append a register on $\mathbb{F}_2^{n-1}$

$$\bigotimes_{i=1}^{n} |\psi_{k_i}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{b}\in\mathbb{F}_2^n} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} |\mathbf{b}\rangle$$

2: Compute the inner product of $\mathbf{b}$ and $\mathbf{k}$ in the ancillary register

$$\frac{1}{\sqrt{2^n}} \sum_{\mathbf{b}\in\mathbb{F}_2^n} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} |\mathbf{b}\rangle \left|\mathbf{b}\cdot\mathbf{k} \mod 2^{n-1}\right\rangle$$

3: Measure the ancillary register ▷ Z is a normalizing constant

$$\frac{1}{\sqrt{Z}} \sum_{\substack{\mathbf{b}\in\mathbb{F}_2^n:\\ \mathbf{b}\cdot\mathbf{k}=z \mod 2^{n-1}}} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} |\mathbf{b}\rangle |z\rangle$$

4: Search for vectors $\mathbf{b_i}$ such that $\mathbf{b_i}\cdot\mathbf{k} = z \mod 2^{n-1}$ using $\mathscr{S}^{\mathrm{C}}$
5: Project the superposition onto a pair of solutions, *e.g.*, $(\mathbf{b}_1, \mathbf{b}_2)$

$$\frac{1}{\sqrt{2}} \left(\omega_N^{s\mathbf{b_1}\cdot\mathbf{k}} |\mathbf{b}_1\rangle + \omega_N^{s\mathbf{b_2}\cdot\mathbf{k}} |\mathbf{b}_2\rangle\right)$$

6: Relabel the basis states to $(|0\rangle, |1\rangle)$, resulting in

$$\frac{\omega_N^{s\mathbf{b_1}\cdot\mathbf{k}}}{\sqrt{2}} \left(|0\rangle + \omega_N^{s(\mathbf{b_2}-\mathbf{b_1})\cdot\mathbf{k}} |1\rangle\right)$$

7: Apply a Hadamard gate on the qubit, measure it and output the result.

2.2 Using a Quantum Subset-sum Solver

The main observation that led to the design of the algorithm we introduce hereafter is that on one hand, we would like to build the superposition

$$\frac{1}{\sqrt{N}} \sum_{j\in\mathbb{Z}_N} \omega_N^{sj} |j\rangle \tag{5.1}$$

since applying the inverse QFT on $\mathbb{Z}_N$ on it would directly give the secret s, and on the other hand, we know that it would be possible, thanks to a quantum subset-sum solver, to prepare the state

$$\frac{1}{\sqrt{Z(\mathbf{k})}} \sum_{\mathbf{b}\in\mathbb{F}_2^m} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} |\mathbf{b}\cdot\mathbf{k} \mod N\rangle \tag{5.2}$$

where $Z(\mathbf{k})$ is a normalizing constant depending on $\mathbf{k}$. Indeed, preparing this state is done by using Regev's trick (see [Reg02; Ste+09]), *i.e.*,

(*i*) by tensoring m phase vectors

$$\frac{1}{\sqrt{M}} \sum_{\mathbf{b}\in\mathbb{F}_2^m} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} |\mathbf{b}\rangle |\mathbf{0}_n\rangle ,$$

(*ii*) then computing the subset-sum in the second register to get the entangled state

$$\frac{1}{\sqrt{M}} \sum_{\mathbf{b}\in\mathbb{F}_2^m} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} |\mathbf{b}\rangle |\mathbf{b}\cdot\mathbf{k} \mod N\rangle ,$$

(*iii*) and finally disentangle it thanks to a quantum subset-sum algorithm which from $\mathbf{b}\cdot\mathbf{k} \mod N$ and $\mathbf{k}$ (which is classical) recovers $\mathbf{b}$ and subtracts it from the first register to get the state we want.

As one can see, if we could take $m = n$ and have an isomorphism between the vectors $\mathbf{b}$ and the knapsack sums $\mathbf{b}\cdot\mathbf{k} \mod N$, the prepared state (5.2) would be exactly the superposition (5.1).

However, there would be many cases in which multiple solutions to the subset-sum problem exist. Thus we take $m < n$ and define $M \stackrel{\text{def}}{=} 2^m < N$. This is different from Algorithm 10, where such collisions are needed. We obtain Algorithm 11, which uses Regev's trick with a quantum subset-sum solver in Step 3.

Despite M being smaller than N, some cases still yield multiple solutions, and furthermore the subset-sum solver (as given by Fact 4) fails on some instances. This is why we distinguish between Algorithm 11 in which we consider the quantum subset-sum solver to be ideal (*i.e.*, it finds back $\mathbf{b}$ from $\mathbf{b}\cdot\mathbf{k}$ and $\mathbf{k}$ with certainty), and the algorithm that we actually build in practice: Algorithm 12.

The analysis of Algorithm 12 is related to the set of $\mathbf{b}$ on which the quantum subset-sum solver succeeds: $\mathscr{S}^{\mathrm{Q}}(\mathbf{b}\cdot\mathbf{k}) = \mathbf{b}$ for a fixed $\mathbf{k}$.

Notation 1. *Let us denote by* $\mathscr{G}(\mathbf{k})$ *the set of* $\mathbf{b}$*'s that are correctly found back by* $\mathscr{S}^{\mathrm{Q}}$ *for a given* $\mathbf{k}$*:*

$$\mathscr{G}(\mathbf{k}) \stackrel{def}{=} \{\mathbf{b}\in\mathbb{F}_2^m : \mathscr{S}^{\mathrm{Q}}(\mathbf{b}\cdot\mathbf{k}) = \mathbf{b}\}$$

and let $G(\mathbf{k})$ *be the size of the set* $\mathscr{G}(\mathbf{k})$*.*

We apply in Step 4 a measurement in order to disentangle the superposition we have, so we can apply an inverse QFT in the same natural way as in the ideal algorithm. We show that the probability of success of the measurement (*i.e.*, of measuring 0) is good enough for our purpose when taking m close to n. We also prove under the same

Algorithm 11 Ideal algorithm

Require: A parameter $m < n$ and phase vectors $|\psi_{k_i}\rangle$ for $i \in [\![1, m]\!]$.
Ensure: An element $j \in \mathbb{Z}_N$.

1: Tensor the m phase vectors and append a register on $\mathbb{Z}_N$

$$\bigotimes_{i=1}^{m} |\psi_{k_i}\rangle |\mathbf{0}_n\rangle = \frac{1}{\sqrt{M}} \sum_{\mathbf{b}\in\mathbb{F}_2^m} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} |\mathbf{b}\rangle |\mathbf{0}_n\rangle$$

2: Compute the inner product of $\mathbf{b}$ and $\mathbf{k}$ in the ancillary register

$$\frac{1}{\sqrt{M}} \sum_{\mathbf{b}\in\mathbb{F}_2^m} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} |\mathbf{b}\rangle |\mathbf{b}\cdot\mathbf{k} \mod N\rangle$$

3: Uncompute $\mathbf{b}$ thanks to $\mathbf{k}$ and $|\mathbf{b}\cdot\mathbf{k} \mod N\rangle$

$$\frac{1}{\sqrt{Z(\mathbf{k})}} \sum_{\mathbf{b}\in\mathbb{F}_2^m} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} |\mathbf{0}_m\rangle |\mathbf{b}\cdot\mathbf{k} \mod N\rangle$$

4: Apply the inverse QFT on $\mathbb{Z}_N$ on the second register

$$\frac{1}{\sqrt{N}} \sum_{j\in\mathbb{Z}_N} \left(\frac{1}{\sqrt{Z(\mathbf{k})}} \sum_{\mathbf{b}\in\mathbb{F}_2^m} \omega_N^{(s-j)\mathbf{b}\cdot\mathbf{k}} \right) |\mathbf{0}_m\rangle |j\rangle$$

5: Measure the state and output the resulting j.

assumption that the algorithm outputs the secret with good probability. All in all, these two properties lead to our main result.

Theorem 5.1. *There exists an algorithm which finds s using $O(n)$ queries and the same usage in time and space as the subset-sum solver $\mathscr{S}^{\mathrm{Q}}$.*

In order to analyze Algorithm 12 and prove Theorem 5.1, we will proceed in two steps.

Step 1.

The first step is to give a lower bound on $\mathbb{E}_{\mathbf{k}}[G(\mathbf{k})]$. This lower bound is given by estimating the number of vectors which admit more than one possible solution.

To arrive here, we first take a look at the normalization constant $Z(\mathbf{k})$ and we compute $\mathbb{E}_{\mathbf{k}}[Z(\mathbf{k})]$ (the average over all choices of $\mathbf{k}$). This can be done by simply looking at the measurement step in Algorithm 11.

Lemma 5.2. *We have*

$$\mathbb{E}_{\mathbf{k}}[Z(\mathbf{k})] = M\left(1 + \frac{M-1}{N}\right).$$

Algorithm 12 Finding s using a quantum subset-sum solver $\mathscr{S}^{\mathrm{Q}}$

Require: A parameter $m < n$ and phase vectors $|\psi_{k_i}\rangle$ for $i \in [\![1, m]\!]$.
Ensure: An element $j \in \mathbb{Z}_N$.

1: Tensor the phase vectors and append a register on $\mathbb{Z}_N$

$$\bigotimes_{i=1}^{m} |\psi_{k_i}\rangle \, |\mathbf{0}_n\rangle = \frac{1}{\sqrt{M}} \sum_{\mathbf{b} \in \mathbb{F}_2^m} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} \, |\mathbf{b}\rangle \, |\mathbf{0}_n\rangle$$

2: Compute the inner product of $\mathbf{b}$ and $\mathbf{k}$ in the ancillary register

$$\frac{1}{\sqrt{M}} \sum_{\mathbf{b} \in \mathbb{F}_2^m} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} \, |\mathbf{b}\rangle \, |\mathbf{b}\cdot\mathbf{k} \mod N\rangle$$

3: Apply $\mathscr{S}^{\mathrm{Q}}$ to uncompute $\mathbf{b}$

$$\frac{1}{\sqrt{M}} \sum_{\mathbf{b} \in \mathbb{F}_2^m} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} \left| \mathbf{b} \oplus \mathscr{S}^{\mathrm{Q}}(\mathbf{b}\cdot\mathbf{k}) \right\rangle |\mathbf{b}\cdot\mathbf{k} \mod N\rangle$$

4: Measure the first register. If the result is not $\mathbf{0}_m$, abort and restart with new coset states. Otherwise, we obtain

$$\frac{1}{\sqrt{G(\mathbf{k})}} \sum_{\mathbf{b} \in \mathscr{G}} \omega_N^{s\mathbf{b}\cdot\mathbf{k}} \, |\mathbf{0}_m\rangle \, |\mathbf{b}\cdot\mathbf{k} \mod N\rangle$$

5: Apply the inverse QFT on $\mathbb{Z}_N$ on the second register

$$\frac{1}{\sqrt{N}} \sum_{j \in \mathbb{Z}_N} \left(\frac{1}{\sqrt{G(\mathbf{k})}} \sum_{\mathbf{b} \in \mathscr{G}} \omega_N^{(s-j)\mathbf{b}\cdot\mathbf{k}} \right) |\mathbf{0}_m\rangle \, |j\rangle$$

6: Measure the state and output the resulting j.

Proof. Fix $\mathbf{k} = (k_1, \cdots, k_m)$. For all $j \in \mathbb{Z}_N$, the measurement in Algorithm 11 returns j with probability:

$$\begin{aligned}
\mathbb{P}_{ideal}\left[j|\mathbf{k}\right] &= \frac{1}{NZ(\mathbf{k})} \left| \sum_{\mathbf{b} \in \mathbb{F}_2^m} \omega_N^{(s-j)\mathbf{b}\cdot\mathbf{k}} \right|^2 \\
&= \frac{1}{NZ(\mathbf{k})} \left| \prod_{i=1}^{m} \left(1 + \omega_N^{(s-j)k_i}\right) \right|^2 \qquad = \frac{1}{NZ(\mathbf{k})} \prod_{i=1}^{m} \left|1 + \omega_N^{(s-j)k_i}\right|^2 \\
&= \frac{1}{NZ(\mathbf{k})} \prod_{i=1}^{m} 4\cos^2\left(\pi k_i \frac{s-j}{N}\right) \qquad = \frac{M^2}{NZ(\mathbf{k})} \prod_{i=1}^{m} \cos^2\left(\pi k_i \frac{s-j}{N}\right) .
\end{aligned}$$

Furthermore, we have $\sum_{j\in\mathbb{Z}_N} \mathbb{P}_{ideal}[j|\mathbf{k}] = 1$, so we can write:

$$Z(\mathbf{k}) = \frac{M^2}{N} \sum_{j\in\mathbb{Z}_N} \prod_{i=1}^{m} \cos^2\left(\pi k_i \frac{s-j}{N}\right) \tag{5.3}$$

It follows that

$$\mathbb{E}_{\mathbf{k}}[Z(\mathbf{k})] = \frac{M^2}{N} \sum_{j\in\mathbb{Z}_N} \mathbb{E}\left[\prod_{i=1}^{m} \cos^2\left(\pi k_i \frac{s-j}{N}\right)\right]$$

and since the k_i are i.i.d., we have

$$\begin{aligned}\mathbb{E}_{\mathbf{k}}[Z(\mathbf{k})] &= \frac{M^2}{N}\left(1 + \sum_{j\in\mathbb{Z}_N\setminus\{s\}} \prod_{i=1}^{m} \mathbb{E}\left[\cos^2\left(\pi k_i \frac{s-j}{N}\right)\right]\right) \\ &= \frac{M^2}{N}\left(1 + (N-1)\prod_{i=1}^{m}\frac{1}{2}\right) = \frac{M}{N}(N+M-1)\ .\end{aligned}$$

□

Next, we give a relation between $G(\mathbf{k})$ and $Z(\mathbf{k})$.

Lemma 5.3. *For any* $\mathbf{k}$*:*

$$G(\mathbf{k}) \geq (1-\varepsilon)\,(2M - Z(\mathbf{k}))\,.$$

Proof. Fix $\mathbf{k}$. Let $\mathscr{B}(j)$ be the set of vectors whose knapsack sum is j:

$$\mathscr{B}(j) \stackrel{\text{def}}{=} \{\mathbf{b}\in\mathbb{F}_2^m \colon \mathbf{b}\cdot\mathbf{k} = j \bmod N\}$$

and let $\mathscr{C}_i$ be the set of vectors $\mathbf{b}$ that have i collisions:

$$\mathscr{C}_i \stackrel{\text{def}}{=} \{\mathbf{b}\in\mathbb{F}_2^m \colon \#\mathscr{B}(\mathbf{b}\cdot\mathbf{k}) = i\}\,.$$

We denote by C_i the size of the set $\mathscr{C}_i$.
If we take a closer look at $Z(\mathbf{k})$, we have that

$$\begin{aligned} Z(\mathbf{k}) &= \sum_{j\in\mathbb{Z}_N}\left|\sum_{\mathbf{b}\in B(j)} \omega_N^{s\mathbf{b}\cdot\mathbf{k}}\right|^2 & &= \sum_{j\in\mathbb{Z}_N}\left|\omega_N^{sj}\right|^2\left|\sum_{\mathbf{b}\in B(j)} 1\right|^2 \\ &= \sum_{j\in\mathbb{Z}_N}\sum_{\mathbf{b}\in B(j)}\sum_{\mathbf{b}'\in B(j)} 1 & &= \sum_{j\in\mathbb{Z}_N}\sum_{\mathbf{b}\in B(j)}\sum_{\mathbf{b}'\in B(\mathbf{b}\cdot\mathbf{k})} 1 \\ &= \sum_{j\in\mathbb{Z}_N}\sum_{\mathbf{b}\in B(j)} \#\mathscr{B}(\mathbf{b}\cdot\mathbf{k}) & &= \sum_{\mathbf{b}\in\mathbb{F}_2^m} \#\mathscr{B}(\mathbf{b}\cdot\mathbf{k}) \\ &= \sum_{i\geq 1}\sum_{\mathbf{b}\in\mathbb{F}_2^m\colon \#\mathscr{B}(\mathbf{b}\cdot\mathbf{k})=i} i & &= \sum_{i\geq 1} iC_i \end{aligned}$$

Letting $C_{>1}$ be the number of vectors $\mathbf{b}$ with at least one collision (*i.e.*, for which there exists $\mathbf{b}' \neq \mathbf{b}$ such that they have the same knapsack sum), we have $C_{>1} = \sum_{i>1} C_i$. From

$$Z(\mathbf{k}) = \sum_{i\geq 1} iC_i = C_1 + 2\sum_{i\geq 2} C_i + \sum_{i\geq 3}(i-2)C_i \ ,$$

it follows that we have the lower bound:

$$Z(\mathbf{k}) \geq C_1 + 2C_{>1} \ .$$

Injecting twice the equation $C_1 = M - C_{>1}$ in this inequality and using the trivial bound $G(\mathbf{k}) \geq (1-\varepsilon)C_1$, we conclude the proof. □

From Lemma 5.2 and Lemma 5.3, we immediately deduce:

Lemma 5.4.

$$\mathbb{E}_{\mathbf{k}}\left[G(\mathbf{k})\right] \geq (1-\varepsilon)M\left(1 - \frac{M-1}{N}\right).$$

Step 2.

The second step in our proof computes the probability of success of the "real" algorithm by relating it to $\mathbb{E}_{\mathbf{k}}\left[G(\mathbf{k})\right]$.

Lemma 5.5. *Algorithm 12 outputs the secret s with probability $\geq (1-\varepsilon)\frac{M(N-M+1)}{N^2}$.*

Proof. We compute the probability of measuring $j \in \mathbb{Z}_N$ at the end of Algorithm 12. In particular, we have for s

$$\mathbb{P}_{real}\left[s|\mathbf{k}\right] = \frac{1}{NG(\mathbf{k})}\left|\sum_{\mathbf{b}\in\mathscr{G}} \omega_N^0\right|^2 = \frac{G(\mathbf{k})}{N}$$

We have by Lemma 5.4 that $\mathbb{E}\left[G(\mathbf{k})\right] \geq (1-\varepsilon)M\left(1-\frac{M-1}{N}\right)$. We finish the proof by observing that $\mathbb{P}_{real}\left[s\right] = \mathbb{E}\left[\mathbb{P}_{real}\left[s|\mathbf{k}\right]\right] \geq (1-\varepsilon)\frac{M(N-M+1)}{N^2}$. □

Finally, we can prove Theorem 5.1.

Proof. Step 4 of Algorithm 12 succeeds with average probability $\frac{\mathbb{E}[G(\mathbf{k})]}{M}$ which is greater than $(1-\varepsilon)\frac{N-M+1}{N}$ (by Lemma 5.4). The final measurement of the algorithm outputs the secret with probability $\geq (1-\varepsilon)\frac{M(N-M+1)}{N^2}$ (by Lemma 5.5). We will thus have to repeat the algorithm an expected number smaller than $\frac{N^3}{(1-\varepsilon)^2M(N-M+1)^2}$ times. By letting m be equal to $n-1$, we obtain that the algorithm will have to be repeated less than $8/(1-\varepsilon)^2$ times. Thus, we can conclude that our algorithm needs $O\left(n\right)$ queries and has complexity costs identical to the ones of the subset-sum solver, since the subset-sum resolution is the only exponential step of the algorithm. □

3 Interpolation algorithm

If we take a look at the ideal algorithm and consider that there is no collision, we can see that we would like 2^m to be as close to N as possible in order for the sum

$$\frac{1}{\sqrt{M}} \sum_{\mathbf{b} \in \mathbb{F}_2^m} \omega_N^{(s-j)\mathbf{b}\cdot\mathbf{k}}$$

to contain as many as possible elements of the sum

$$\frac{1}{\sqrt{N}} \sum_{\mathbf{b} \in \mathbb{F}_2^n} \omega_N^{(s-j)\mathbf{b}\cdot\mathbf{k}}.$$

In the mean time, it is clear that the closest M gets to N, the more likely collisions $\mathbf{b_1} \cdot \mathbf{k} = \mathbf{b_2} \cdot \mathbf{k}$ for $\mathbf{b}_1 \neq \mathbf{b}_2$ become. We thus have to find a compromise on the value m or more interestingly play with the values k_i used in the algorithm, to avoid collisions and to simplify the resolution of the subset-sum problem.

In fact, we can reduce the size of the subset-sum problem we have to solve by pre-processing the states to get values of k_i that will allow us to solve the subset-sum problem on some bits by Gaussian elimination. Constructing these k_i's can be achieved by Kuperberg's second algorithm (or any improvement). Given a threshold parameter $t \in [\![1, m]\!]$, we can consider the following configuration for the k_i to use as inputs in Algorithm 12 (dots represent unknown bits and the i-th bit of the j-th row is the j-th bit of the binary expansion of k_i):

$$\begin{array}{c} \\ k_1 \\ k_2 \\ \vdots \\ k_{m-t} \\ k_{m-t+1} \\ \vdots \\ k_m \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & \cdots & m-t & m-t+1 & \cdots & n \end{array} \\ \begin{pmatrix} 1 & \bullet & \cdots & \bullet & \bullet & \cdots & \bullet \\ 0 & 1 & \ddots & \bullet & \bullet & \cdots & \bullet \\ \vdots & \vdots & \ddots & \ddots & \vdots & \cdots & \bullet \\ 0 & 0 & \cdots & 1 & \bullet & \cdots & \bullet \\ 0 & 0 & \cdots & 0 & \bullet & \cdots & \bullet \\ \vdots & \vdots & & \vdots & \vdots & \cdots & \bullet \\ 0 & 0 & \cdots & 0 & \bullet & \cdots & \bullet \end{pmatrix} \end{array} \tag{5.4}$$

In Algorithm 12, it turns out that we can keep a good probability of finding the secret s by letting m be equal to $n-1$ so that is what we will assume afterwards.

To build phase vectors that satisfy the configuration in Equation 5.4, we will approximately have to query the oracle

$$\sum_{i=1}^{n-t} 2^{\sqrt{c_{\mathsf{DCP}} i}} + t 2^{\sqrt{c_{\mathsf{DCP}}(n-t)}}$$

leading to a query and time complexities of $O\left((\sqrt{n-t}+t)2^{\sqrt{c_{\mathsf{DCP}}(n-t)}}\right)$ (by Lemma 4.6), where c_{DCP} is the constant of the algorithm used to construct the states ($c_{\mathsf{DCP}} = 2$ for Kuperberg's second algorithm). For the subset-sum problem, solving it on the first $n-t$ bits is easy (thanks to a Gaussian elimination), the difficulty comes from the last t bits, leading to a complexity in $O\left(2^{c_{q\mathsf{SS}}t}\right)$ time, where $c_{q\mathsf{SS}}$ is the complexity exponent of the quantum subset-sum solver. This parameter t can be used in a natural way to obtain an interpolation algorithm, since it allows to obtain a tradeoff between the preparation of the states and the resolution of the problem (which amounts to solving a quantum subset-sum problem).

We can now give an interpolation algorithm derived from Algorithm 12. We note that letting q be the query complexity exponent, it is possible to determine t from n and the value q we can afford. Using Kuperberg's second algorithm (or any improvement) to compute suitable phase vectors as described before and then giving them as inputs to Algorithm 12, we can retrieve the secret s as described by Algorithm 13 with the complexities given by Theorem 5.2.

Algorithm 13 Interpolation algorithm (using a quantum SS solver)

Require: q such that 2^q is the number of queries we are allowed to do.
Ensure: The secret s.

1: Use Kuperberg's second algorithm (or any improvement) to create states $|\psi_{k_i}\rangle$ for $i \in [\![1, m]\!]$ satisfying the configuration represented by Matrix (5.4), where $t \approx \frac{q}{c_{\mathsf{SS}}}$.
2: Apply Algorithm 12 on these m states to obtain a value $j \in \mathbb{Z}_N$.
3: Check if j is the secret. If not, return to Step 1. Otherwise, output j.

Theorem 5.2. *Let $t \in [\![1, m]\!]$. Algorithm 13 finds s with $O\left((\sqrt{n-t}+t)2^{\sqrt{c_{\mathsf{DCP}}(n-t)}}\right)$ queries in $O\left((\sqrt{n-t}+t)2^{\sqrt{c_{\mathsf{DCP}}(n-t)}} + 2^{c_{q\mathsf{SS}}t}\right)$ quantum time, classical space $O\left(2^{\sqrt{c_{\mathsf{DCP}}(n-t)}} + 2^{c_{q\mathsf{SS}}t}\right)$ and $O(\mathsf{poly}(n))$ quantum space.*

We notice that when $t = m$, the k_i are kept random and we have to solve the "full rank" subset-sum, matching with Algorithm 12. On the other side, when $t = 1$, we fall back on Kuperberg's second algorithm since we have in this case to construct a collection of states divisible by all the successive powers of 2. Finally, when $1 < t < m$, we have new algorithms working for any number of queries between $O(n)$ and $\tilde{O}\left(2^{\sqrt{c_{\mathsf{DCP}}n}}\right)$.

4 Quantum Subset-sum Algorithms

In this section, we consider quantum algorithms solving the *quantum* subset-sum problem introduced in Section 1. We give both asymptotic complexities and numerical estimates.

Recall that we consider a subset-sum instance $(v, \mathbf{k}), \mathbf{k} \in \mathbb{Z}_N^m$, where v is *in superposition*, and $\mathbf{k}$ will remain fixed. The problem is to find $\mathbf{b}$ such that $\mathbf{b} \cdot \mathbf{k} = v \mod N$ for a given (fixed) modulus N. For a given v, if there are many solutions, we want to find one selected uniformly at random (under heuristics). If we want all solutions, then we can run multiple instances of the solver (we will have to redo the pre-computations that we define below). A given solver, defined for a specific $\mathbf{k}$, is expected to work only for some (large) proportion $1 - \varepsilon$ of v. We can check whether the output is a solution or not and measure the obtained bit to collapse on the cases of success.

4.1 Algorithms Based on Representations

The best algorithms to solve the subset-sum problem with density one are list-merging algorithms using the *representation technique* [HJ10; BCJ11]. The best asymptotic complexities (both classical and quantum) are given in [Bon+20]. We detail the representation framework following the depiction given in [Bon+20]. To ease the description, we start with the case $v = 0$, *i.e.*, the *homogeneous* case, and we will show below how to extend it easily to $v \neq 0$.

Guessed Weight. We assume that the solution $\mathbf{b}$ is of weight $\lceil m/2 \rceil$. This is true only with probability: $p_m := 2^{-m}\binom{m}{\lceil m/2 \rceil} = 1/\mathsf{poly}(m)$. If not, we re-randomize the subset-sum instance by multiplying $\mathbf{b}$ by a random invertible matrix. Thus if we manage to solve an instance of weight $\lceil m/2 \rceil$, the total complexity to solve any instance will introduce a multiplicative factor $\frac{1}{p_m}$ that we will have to estimate.

Distributions. We consider *distributions* of vectors having certain relative weights: $D^m[\alpha] \subseteq \{0,1\}^m$ is the set of vectors having weight αm. The basic idea of representations is to write the solution $\mathbf{b}$ as a sum of vectors of smaller relative weights, *e.g.*, $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$ where $\mathbf{b}_1 \in D^m[\alpha_1], \mathbf{b}_2 \in D^m[\alpha_2]$ and $\alpha_1 + \alpha_2 = \frac{1}{2}$. In this paper, we consider only representations with coefficients 0 or 1. Extended representations can be considered, using more coefficients (which have to cancel out each other). However, the advantage of using extended representations becomes quickly insignificant in practice. It is also harder to compute the number of representations, or the filtering probabilities that we define below.

Merging Tree. A subset-sum algorithm is defined by a *merging tree*. A node in this tree is a *list* $L[\ell, \alpha, c]$, which represents a set of vectors drawn from $\{0,1\}^m$ under several conditions: 1. the size of the list is $2^{m\ell}$; 2. the vectors are sampled u.a.r. from a prescribed distribution $D^m[\alpha]$; 3. the vectors satisfy a *modular condition* of cm bits. With $v = 0$, the following condition can be used: $\mathbf{e} \cdot \mathbf{k} \mod N \in [-N/2^{cm}; N/2^{cm}]$ for

some number c. More generally, the modular conditions can be chosen arbitrarily, as long as they remain compatible with the target v.

Once the tree structure is chosen, its parameters are optimized under several constraints. First, the lists have a certain maximal size. A distribution $D^m[\alpha]$ has size $\binom{m}{\alpha m}$, which is asymptotically estimated as $\simeq 2^{h(\alpha)m}$. This creates the constraint $\ell \leq h(\alpha) - c$. Second, we expect the root list to contain the solution of the problem, *i.e.*, $\ell = 0$ (one element), $\alpha = \frac{1}{2}$ and $c = 1$. Finally, each non-leaf list L has its parameters determined by its two children L_1, L_2. Indeed, it is obtained via the *merging-filtering* operation which selects, among all pairs of vectors $(\mathbf{e}_1, \mathbf{e}_2) \in L_1 \times L_2$, the pairs such that: $\mathbf{e}_1 + \mathbf{e}_2$ satisfies the modular condition (merging) and satisfies the weight condition (filtering). The parameters are:

$$\begin{cases} \alpha = \alpha_1 + \alpha_2 \text{ (increasing weights)} \\ \ell = \ell_1 + \ell_2 - (c - \min(c_1, c_2)) - \mathsf{pf}(\alpha_1, \alpha_2) \end{cases} \tag{5.5}$$

Here, pf is the probability that two vectors chosen u.a.r. in their respective distributions will not have colliding 1s.

Lemma 5.6 (Lemma 1 in [Bon+20])**.** *Let $\mathbf{e}_1, \mathbf{e}_2$ be drawn u.a.r. from $D^m[\alpha_1], D^m[\alpha_2]$ with $\alpha_1 + \alpha_2 \leq 1$. The probability that $\mathbf{e}_1 + \mathbf{e}_2 \in D^m[\alpha_1 + \alpha_2]$ is equal to:*

$$\mathsf{PF}(\alpha_1, \alpha_2, m) := \binom{m - \alpha_1 m}{\alpha_2 m} \Big/ \binom{m}{\alpha_2 m} \simeq 2^{m\mathsf{pf}(\alpha_1, \alpha_2)}$$

where $\mathsf{pf}(\alpha_1, \alpha_2) := h\left(\frac{1-\alpha_2}{\alpha_1}\right)\alpha_1 - h(\alpha_1)$.

Classical Computation of the Tree. To any correctly parameterized merging tree corresponds a classical subset-sum algorithm that runs as follows: it creates the leaf lists by sampling their distributions at random. It then builds the parent lists by *merging-filtering* steps. The *merging* operation is efficient, since elements can be ordered according to the modular condition to be satisfied.

Lemma 5.7 (Lemma 2 in [Bon+20])**.** *Let L_1, L_2 be two sorted lists stored in classical memory with random access. In $\log_2$, relatively to m, the parent list L can be built in time:* $\max(\min(\ell_1, \ell_2), \ell_1 + \ell_2 - (c - \min(c_1, c_2)))$ *and in memory* $\max(\ell_1, \ell_2, \ell)$.

Quantum Computation of the Tree. While the more advanced quantum subset-sum algorithms use quantum walks [Ber+13; HM18; Bon+20], we want to focus here on algorithms using few qubits, which at the moment, rely only on quantum merging with Grover search. They replace the classical merging operation by the following.

Lemma 5.8 (Lemma 4 in [Bon+20]). *Let L_2 be a sorted list stored in QRACM. Assume given a unitary U that produces, in time t_{L_1}, a uniform superposition of elements of L_1. Then there exists a unitary U' that produces a uniform superposition of elements of L, in time $O\left(\frac{t_{L_1}}{\sqrt{\mathsf{pf}(\alpha_1,\alpha_2)}}\max(\sqrt{2^{cm}/|L_2|},1)\right)$.*

Since the goal is only to sample u.a.r. from the root list, only half of the lists in the tree need actually to be stored in QRACM. The others are sampled using the unitary operators given by Lemma 5.8. In short, the obtained subset-sum algorithm is a sequence of Grover searches which use existing lists stored in memory to sample elements in new lists with more constraints.

Heuristics. The standard subset-sum heuristic assumes that the elements of all lists in the tree (not only the leaf lists) behave as if they were uniformly sampled from the set of vectors of right weight, satisfying the modular condition. This heuristic ensures that the list sizes are very close to their average: for each L obtained by merging and filtering $L_1[\ell_1,\alpha_1,c_1]$ and $L_2[\ell_2,\alpha_2,c_2]$, we have:

$$|L| \simeq \frac{|L_1||L_2|}{2^{m(c-\min(c_1,c_2))}\mathsf{PF}(\alpha_1,\alpha_2,m)} \ ,$$

where the approximation is exact down to a factor 2. This is true with overwhelming probability for all lists of large expected size via Chernoff-Hoeffding bounds, and even if the root list is of expected size 1, the probability that it actually ends up empty is smaller than $e^{-0.5} \simeq 0.61$.

4.2 From Asymptotic to Exact Optimizations

As the time and memory complexities of a subset-sum algorithm are determined by its merging tree, we seek to select a tree which minimizes these parameters. Given a certain subset-sum problem, we first select a tree shape. As an example, the best subset-sum algorithm with low qubits (using QRACM) is the "quantum HGJ" algorithm of [Bon+20], whose structure is reproduced in Figure 5.1. At level 3, it splits the vectors into two halves, and merges without filtering. While all lists are obtained via quantum merging/filtering, the main computation is performed after obtaining L_1^3, L_1^2, L_1^1, where the main branch is explored using Grover's algorithm: we search through the lists $L_0^3, L_0^2 L_0^1$ without representing them in memory. The quadratic speedup of Grover search makes the tree unbalanced, which is reflected on the naming of its parameters in Figure 5.1.

The *asymptotic* time complexity of the algorithms has the form $\tilde{O}\left(2^{\beta m}\right)$, and is the result of summing together the costs of all merging steps. Through the approximation of binomial coefficients, the list sizes are approximated in $\log_2$ and relatively to m. The

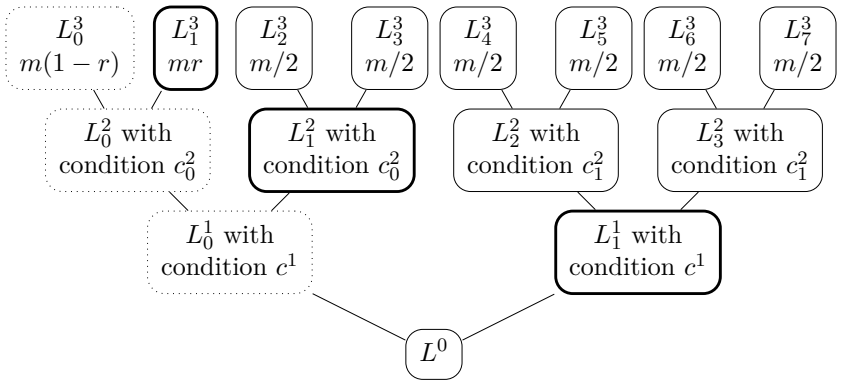

Figure 5.1: Quantum HGJ algorithm. Dotted lists are search spaces (they are not stored). Bold lists are stored in QRACM. The first level uses a left-right split of vectors, without filtering.

parameters (relative weights, modular conditions and sizes) are numerically optimized. The optimization of Figure 5.1 in [Bon+20] yields the complexity $\tilde{O}\left(2^{0.2356m}\right)$.

In this paper, we also perform non-asymptotic optimizations for a given m. Since we use only $\{0,1\}$-representations, the filtering probability is well known and has a simple expression (Lemma 5.6). Since the binomial coefficients can be extended as functions of $\mathbb{R}^2$, we can perform an exact numerical optimization of list sizes for a given m. Afterwards, the numbers obtained are rounded, in particular the weights of representations, and we take the point which gives us the best results: smallest complexity and biggest average size for L^0.

Example. Let us take $n = \log_2 N = 256$, $m = n - 1 = 255$, and the structure of Figure 5.1. We adapt the optimization code of [Bon+20] by taking the exact exponents (not relative to n) and optimize numerically under the constraint $\left|L^0\right| = 2^2$ (to ensure that there are solutions). The asymptotic formula would give $2^{0.2356n} \simeq 2^{60.31}$. Numerical optimization gives us a time $2^{63.81}$, but this admits non-integer parameters and it is only the *maximum* between all steps. By rounding the parameters well, we obtain Figure 5.2.

To compute the quantum time complexity, we consider the list sizes to be exact and use the formula of Lemma 5.8 without the O. The subtrees on the right can be computed in $2^{65.71}$ operations; the slight increase is due to the fact that we take a sum of their respective terms and not a maximum. In the left branch, we sample from L^0 in $2^{63.48}$ operations.

The actual time complexity is slightly bigger, due to the variation in list sizes, and the constant complexity overhead ($\pi/2$) of Grover search. More importantly, these operations require: • to recompute a sum, using m (controlled) additions modulo N; • to test membership in some distribution; • to sample from input distributions D^n.

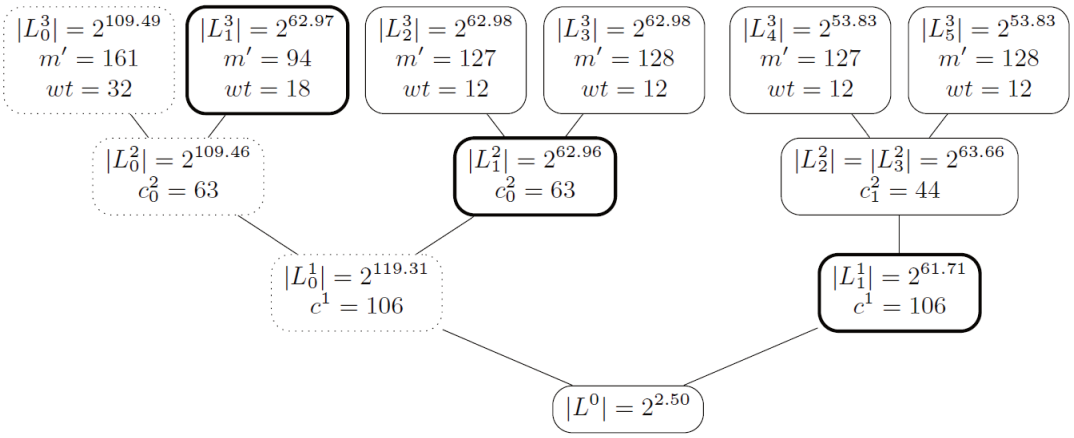

Figure 5.2: Optimization of Figure 5.1 for $m = 255$. The size of the support is indicated by m' and the weight by wt.

The latter can be done using a circuit given in [Ess+21], which for a weight k and n bits, has a gate count $\tilde{O}(nk)$ and uses $n + 2\lceil \log(k+1) \rceil$ qubits. All of this boils down to m arithmetic operations or $O(m^2)$ quantum gates.

Finally, this sampler works only for a proportion $\frac{1}{p_m} = 2^{-5.33}$ of subset-sum instances, so we need to re-randomize accordingly. After running the optimization for $128 \leq m \leq 1024$ and $n = m+1$, we found that the subset-sum solver would use approximately $2^{0.238m+9.203}$ arithmetic operations, for a final list L^0 of size 2 on average. Under the subset-sum heuristic, we assume an independence between all tuples of elements in the initial lists. Using Chernoff-Hoeffding bounds the probability that the final list is empty is smaller than $e^{-1} \simeq 0.37$. To reduce it to a smaller constant ε, we may simply run multiple independent instances of the solver. This increases the asymptotic complexity by a factor $O(-\log \varepsilon)$.

4.3 Solving Subset-Sum in Superposition

We now show that we can reuse the structure of the QRACM-based subset-sum algorithm of Figure 5.1 to solve the problem in *superposition* over the target v, while still keeping the number of qubits polynomial.

The basic idea is to reduce the problem with a given $v \neq 0$ to $v = 0$: $\mathbf{k}' \cdot \mathbf{b} = 0 \mod N$, where $\mathbf{k}'$ is a length $m+1$ vector where we append $-v$ to $\mathbf{k}$. We can then modify *any* existing tree-based subset-sum algorithm solving this instance to force all vectors in the leftmost leaf list to have a 1 in the last coordinate, and all vectors in the other leaves to have 0 in this coordinate. Then only the lists in the left branch of the tree depend on v. The complexity is unchanged.

Following the tree structure of Figure 5.1, we create the lists L_1^3, L_1^2, L_1^1 in a precomputation step. Then we define a quantum algorithm that outputs an element in L^0 (or a superposition of such elements), and we run this algorithm in superposition over v.

Subset-Sum without QRACM. Helm and May [HM20] showed that quantum subset-sum algorithms using a small classical memory (without quantum access) can have better time-memory tradeoffs than classical ones. They obtained a time $\tilde{O}\left(2^{0.428m}\right)$ for a memory $O\left(2^{0.285m}\right)$, however their algorithm does not have an unbalanced structure like ours.

We improve on this time-memory tradeoff by adapting the tree of Figure 5.1 as follows: we remove L_0^3 and L_1^3 and their parameters, and directly sample in L_0^2. Assuming that the lists L_1^2 and L_1^1 are precomputed classically, we sample from L^0 with the same algorithm, except that it replaces each QRACM access (in time 1) by a sequential memory access (in time $\left|L_1^2\right|$ and $\left|L_1^1\right|$ for L_1^2 and L_1^1 respectively), *i.e.*, a quantum circuit which encodes the elements of the lists as a sequence of standard gates. The asymptotic optimization gives a time $\tilde{O}\left(2^{0.4165m}\right)$ with a memory $O\left(2^{0.2324m}\right)$. The parameters are displayed in Figure 5.3.

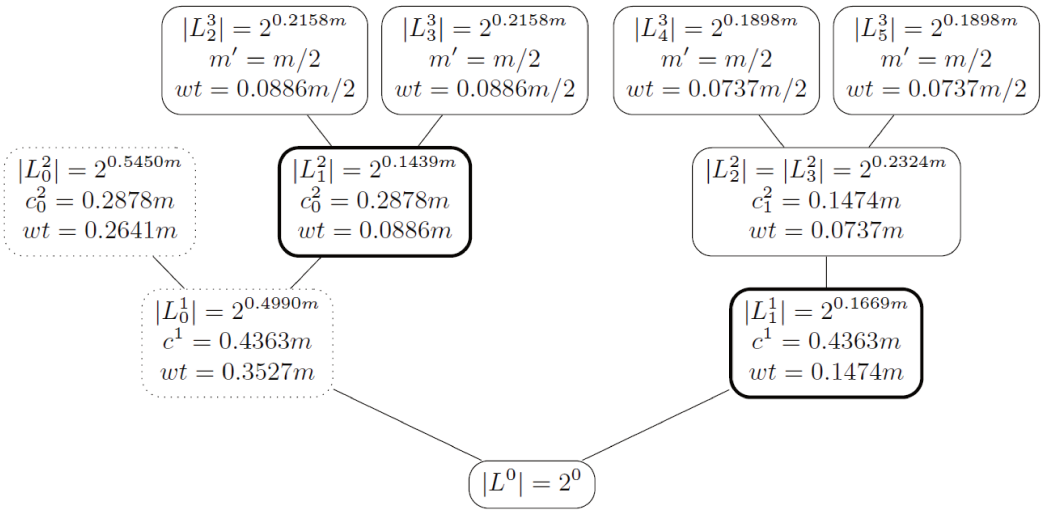

Figure 5.3: Asymptotic optimization of our quantum subset-sum algorithm without QRACM. The lists on the right of the tree are constructed with classical computations, using classical RAM. The lists L_1^2 and L_1^1 are stored in classical memory without random access.

The difference between asymptotic and non-asymptotic optimization is bigger here. For $m = n - 1 = 127$, with the constraint $\left|L^0\right| = 2^2$, we obtain a time $2^{60.01} > 2^{128\times 0.4165} =$

$2^{53.31}$ and a memory $2^{26.82} < 2^{128\times 0.2324} = 2^{29.75}$. On top of this, we must also take p_m into account.

After running optimizations for $n = 128$ to 1024, we obtained a count of about $2^{0.418m+12.851}$ blocks of m arithmetic operations (m^2 quantum gates). The point at which the algorithm starts improving over Grover search lies around $n = 157$.

Chapter 6
A Space Interpolation Algorithm

To briefly review the main historical stages in solving the DHSP, the first major milestone was Kuperberg's first algorithm, proposed in 2003. It solves the DHSP directly, using only CNOT gates and measurements, in sub-exponential time, but also with sub-exponential space. Less than a year later, Regev proposed an algorithm that is also sub-exponential in time, but with two notable differences: it reduces the DHSP to a subset-sum problem and requires only a polynomial (classical and quantum) space. Subsequent efforts have then focused on this idea of reducing the DHSP to a subset-sum problem, leading to the state-of-the-art, Kuperberg's second algorithm, and several tradeoffs. On the other hand, only minor improvements have been made on Kuperberg's first algorithm (of the order of polynomial factors, see Chapter 4), and the basic idea of the algorithm does not seem to have been exploited beyond that.

In this chapter, we look at just that and give some hints as to what could be done to obtain new algorithms for solving the DHSP using only CNOT gates and measurements, with the aim of building (both classical and quantum) space-efficient algorithms. The work presented here is still in progress, which is why conjectures about the complexity of the algorithms introduced hereafter are given and attempts to prove them are presented. In addition, experimental results are given, reinforcing in particular a conjecture about the complexity of one of the algorithms presented. This algorithm would solve the DCP in $\mathbb{Z}_N$ where $N = 2^n$ using *at most* n qubits (and likewise a classical space of the order of n^2 bits) in (classical and quantum) time $\tilde{\Theta}\left(2^{0.415n}\right)$ with an equivalent number of queries. This algorithm would therefore be the fastest among all the previously known algorithms that use a linear number of qubits to solve the DCP (namely, Ettinger-Høyer algorithm and the algorithm presented in Chapter 5). The work presented in this chapter is the object of a preprint to come soon [RT23].

Contents

1 A Space-Efficient Algorithm

We recall that phase vectors are defined as states

$$|\psi_k\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}(|0\rangle + \omega_N^{sk}|1\rangle) \tag{6.1}$$

where $s \in \mathbb{Z}_N$ is an unknown integer we are looking for. We will assume for the sake of simplicity that $N = 2^n$.

In what follows, we assume we have an oracle $\mathscr{O}$ outputting at each call an integer k drawn uniformly at random from $\mathbb{Z}_N$ and the corresponding phase vector $|\psi_k\rangle$ as defined in Equation (6.1). When such a couple is obtained, we will put it in a *pool* denoted by $\mathscr{P}_i$ if 2^{i-1} is the greatest power of 2 that divides k, more precisely:

$$\mathscr{P}_i \longleftarrow (k, |\psi_k\rangle) \quad \text{if} \quad k \in \mathcal{D}_i \quad \text{where}$$

$$\forall i \in [\![1, n]\!], \qquad \mathcal{D}_i \stackrel{\text{def}}{=} \left\{(2\alpha - 1)2^{i-1}, \alpha \in [\![1, 2^{n-i}]\!]\right\}.$$

Thanks to this definition, Bonnetain and Naya-Plasencia [BN18] proposed to write Kuperberg's first algorithm in a more opportunistic way, for which we recall in Algorithm 14 a pseudo-code implementation.

Algorithm 14 Opportunistic variant of Kuperberg's first algorithm

Require: A set of phase vectors.
Ensure: $\left|\psi_{N/2}\right\rangle$.
1: Classify the phase vectors in the pools $\mathscr{P}_i$
2: **for** $i = 1$ to $n - 1$ **do**
3: **while** $|\mathscr{P}_i| \geq 2$ **do**
4: Pick $|\psi_a\rangle$ and $|\psi_b\rangle$ from $\mathscr{P}_i$ such that $a \pm b$ has the highest possible divisibility by 2 (and is not 0)
5: Combine $|\psi_a\rangle$ and $|\psi_b\rangle$ and insert the result in the appropriate pool
6: **if** $\mathscr{P}_n \neq \emptyset$ **then return** $\left|\psi_{N/2}\right\rangle$
7: **return** "The algorithm failed".

It is claimed in [BN18] that Algorithm 14 has the same complexity as Kuperberg's first algorithm and this is the algorithm that inspired us to build a new one that uses at most n qubits and n^2 bits to solve the DCP, making it the most space-efficient algorithm to date. Its pseudocode is provided in Algorithm 15.

Algorithm 15 Our algorithm

Require: An oracle $\mathscr{O}$ outputting uniformly at random $|\psi_k\rangle$ with $k \in \mathbb{Z}_N$.
Ensure: $\left|\psi_{N/2}\right\rangle$.
1: **repeat**
2: **while** for all i, $|\mathscr{P}_i| < 2$ **do**
3: Call $\mathscr{O}$ and insert the result in the appropriate pool.
4: Let i be such that $|\mathscr{P}_i| = 2$.
5: Pick $|\psi_a\rangle$ and $|\psi_b\rangle$ from $\mathscr{P}_i$.
6: Combine $|\psi_a\rangle$ and $|\psi_b\rangle$ and insert the result in the appropriate pool
7: **until** there is a state in the pool $\mathscr{P}_n$
8: **return** $\left|\psi_{N/2}\right\rangle$.

Note that when the state $|\psi_0\rangle$ is produced, we just drop it off, since it is basically the state $|+\rangle$ and does not carry any information about the secret.

From this algorithm we cannot say much other than to study the probability distributions corresponding, on one hand, to what the oracle produces and, on the other, to what is obtained by combining two elements from the same pool.

Probability Distributions. The probability distribution of elements picked uniformly at random from the oracle is given by the following lemma.

Lemma 6.1 (Initial distribution)**.** *Let $i \in [\![1, n]\!]$. The probability that the oracle $\mathscr{O}$ yields a phase vector belonging to the pool $\mathscr{P}_i$, denoted by $\mathbb{P}_{\mathscr{O}}[i]$, is*

$$\mathbb{P}_{\mathscr{O}}[i] = 2^{-i}.$$

Proof. This is straightforward since $\mathbb{P}_{\mathscr{O}}[i] = \frac{|\mathcal{D}_i|}{|\mathbb{Z}_N|} = \frac{2^{n-i}}{2^n}$. □

The probability distribution of elements obtained from combining two phase vectors of a same pool is given by the following lemma.

Lemma 6.2 (Combination distribution)**.** *Let $i \in [\![1, n-1]\!]$ and $j \in [\![1, n]\!]$. The probability that combining two phase vectors from $\mathscr{P}_i$ yields a phase vector from $\mathscr{P}_j$, denoted by $\mathbb{P}_i[j]$, is*

$$\mathbb{P}_i[j] = \begin{cases} 2^{i-j} & \text{if } j > i \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Let $\mathbb{P}_i^+[j]$ (resp. $\mathbb{P}_i^-[j]$) be the probability for two phase vectors from $\mathscr{P}_i$ to produce one from $\mathscr{P}_j$ when combined and their sum (resp. difference) is obtained. Let $k \in \mathcal{D}_i$. There exists $\alpha \in [\![1, 2^{n-i}]\!]$ such that $k = (2\alpha - 1)2^{i-1}$. We first look for the proportion $\mathbb{P}_i^+[j]$ of $\ell \in \mathcal{D}_i$, which we can write as $\ell = (2\beta - 1)2^{i-1}$ for some $\beta \in [\![1, 2^{n-i}]\!]$, such that $k + \ell = (\alpha + \beta - 1)2^i \in \mathscr{P}_j$, *i.e.*,

$$\mathbb{P}_i^+[j] = \frac{1}{|\mathcal{D}_i|} \# \left\{ \beta \in [\![1, 2^{n-i}]\!] \;\middle|\; (\alpha + \beta - 1)2^i \in \mathscr{P}_j \right\}$$

Looking closely at the condition, it is equivalent to say that there must exist $\gamma \in [\![1, 2^{n-j}]\!]$ such that

$$(\alpha + \beta - 1)2^i = (2\gamma - 1)2^{j-1}$$

We can see that if $j \leq i$, no value for γ exists such that this equation holds. We assume from now on that $j > i$. The condition becomes

$$\alpha + \beta - 1 = 2^{j-(i+1)} \bmod 2^{j-i}$$

It follows that

$$\mathbb{P}_i^+[j] = \frac{1}{|\mathcal{D}_i|} \frac{2^{n-i}}{2^{j-i}} = 2^{i-j}$$

We use the same reasoning for the difference, $\mathbb{P}_i^-[j]$ being the proportion of $\ell \in \mathscr{P}_i$ such that $|k - \ell| = |\alpha - \beta|2^i \in \mathscr{P}_j$, *i.e.*,

$$\mathbb{P}_i^-[j] = \frac{1}{|\mathcal{D}_i|} \# \left\{ \beta \in [\![1, 2^{n-i}]\!] \;\middle|\; |\alpha - \beta|2^i \in \mathscr{P}_j \right\}$$

It is equivalent to say that there must exist $\gamma \in [\![1, 2^{n-j}]\!]$ such that

$$|\alpha - \beta| 2^i = (2\gamma - 1) 2^{j-1}$$

Once again if $j \leq i$, no value for γ exists such that this equation holds. We assume from now on that $j > i$. The condition is equivalent to

$$|\alpha - \beta| = 2^{j-(i+1)} \bmod 2^{j-i}$$

It follows that

$$\mathbb{P}_i^- [j] = \frac{1}{|\mathcal{D}_i|} \frac{2^{n-i}}{2^{j-i}} = 2^{i-j}$$

By definition, the overall probability $\mathbb{P}_i [j]$ is

$$\mathbb{P}_i [j] = \frac{1}{2} \left(\mathbb{P}_i^+ [j] + \mathbb{P}_i^- [j] \right) = 2^{i-j}.$$

□

These probability distributions would actually be useful for studying the mean hitting time of the Markov chain corresponding to the process followed in the algorithm, which would thus give us the average time complexity of the algorithm. The problem is that there is an exponential number of states in this Markov chain.

On the Markov chain of Algorithm 15. Indeed, these states are vectors of length the number of qubits we are using (namely, n). They contain in their i-th entry the number at time t of phase vectors contained in the i-th pool. The transitions between different vectors are made according to the probability distribution of the combination if we start from a vector that contains a 2 (since in this case we have two phase vectors in the same subpool that we have to combine) or according to the probability distribution given by the oracle in the other case (since if there are no elements to combine, we draw a new one from the oracle). All in all, the Markov chain has an exponential number of states, which makes it difficult to study. Indeed, we have:

- the configurations were a solution have been found, *i.e.*, there is 1 element in $\mathscr{P}_n$, coming either from the oracle (which means before calling the oracle, there was no pool with 2 elements) or either from a combination (meaning that 2 elements from a pool were used to produce the solution, leaving no pool with 2 elements) are 2^{n-1} of them,
- the configurations for which no solution has been found, corresponding either to a situation where a query must be made (*i.e.*, no pool contains 2 elements), or to a situation where a combination must be made (*i.e.*, there is a pool which contains 2 elements). There are therefore $2^{n-1} + \binom{n-1}{1} 2^{n-2} = (n+1) 2^{n-2}$ of these configurations.

In total, the Markov chain corresponding to our algorithm therefore has $(n+3)2^{n-2}$ states. Studying the mean hitting time starting from the configuration where we do not have any phase vector in any of the pools would give us the exact average-time complexity of our algorithm, but unfortunately we could not manage to deal with this exponential amount of states and their intricate structure.

Our analysis. In order to avoid this exponential number of states to consider, we looked at a different algorithm which is based on the same idea but where we assume that all the calls to the oracle have been made from the outset. All that remains is to combine the phase vectors (which are taken at random, unlike in Algorithm 14). Its pseudocode is given in Algorithm 16. In order to determine the query and time complexity of this algorithm, we barely have to find what quantity of phase vectors we should give it as input. We conjecture that the number of calls to the oracle made by this algorithm is of the same order as that of Algorithm 15, which is what we are observing experimentally (experimental results are given at the end of the section).

Algorithm 16 The algorithm we study (sequential version)

Require: A set of 2^ℓ phase vectors.
Ensure: $\left|\psi_{N/2}\right\rangle$.
1: Classify the phase vectors in the pools $\mathscr{P}_i$
2: **for** $i = 1$ to $n-1$ **do**
3: **while** $|\mathscr{P}_i| \geq 2$ **do**
4: Pick $|\psi_a\rangle$ and $|\psi_b\rangle$ from $\mathscr{P}_i$.
5: Combine $|\psi_a\rangle$ and $|\psi_b\rangle$ and insert the result in the appropriate pool
6: **if** $\mathscr{P}_n \neq \emptyset$ **then return** $\left|\psi_{N/2}\right\rangle$
7: **return** "The algorithm failed".

Algorithm 15 is essentially a version that makes calls to the oracle as it goes along and so recycles the qubits it uses, whereas Algorithm 16 stores all the states it needs as input. The latter then processes the phase vectors sequentially. It first sorts them according to the pool to which they belong, then runs through these pools in order, since combining elements from the i-th pool produces elements that belong exclusively to pools $i+1$ to n, as shown in Lemma 6.2. In this way, the algorithm gradually concentrates the phase vectors in the last pools. Indeed, on the i-th iteration of the For loop, at most one element (which could not have been matched) remains in pools $\mathscr{P}_1$ through $\mathscr{P}_i$, while the pool $\mathscr{P}_{i+1}$ will contain a quantity of phase vectors denoted $\mathcal{Q}_{i+1}$. This must be greater than or equal to 2 in order to form at least one pair of phase vectors, which can be combined to obtain a more interesting pair. Finally, for the algorithm to succeed, we must have $\mathcal{Q}_n$ equal to at least 1 at the end.

We can also write this algorithm in parallel mode, which will be even more easier to study: instead of acting sequentially by processing the pools one after the other, we can process all the pools $\mathscr{P}_i$ at once and place the results in a separate pool, then classify these new states in the pools $\mathscr{P}_i$ again and iterate the process. This is the method used in Algorithm 17.

Algorithm 17 The algorithm we study (parallel version)

Require: A pool $\mathscr{P}$ of 2^ℓ phase vectors.
Ensure: $\left|\psi_{N/2}\right\rangle$.
1: **for** $t = 1$ to $n-1$ **do**
2: Take the phase vectors from $\mathscr{P}$ and classify them in the pools $\mathscr{P}_i$
3: **for** $i = 1$ to $n-1$ **do**
4: **while** $|\mathscr{P}_i| \geq 2$ **do**
5: Pick $|\psi_a\rangle$ and $|\psi_b\rangle$ from $\mathscr{P}_i$.
6: Combine $|\psi_a\rangle$ and $|\psi_b\rangle$ and insert the result in $\mathscr{P}$
7: **if** $\mathscr{P}_n \neq \emptyset$ **then return** $\left|\psi_{N/2}\right\rangle$
8: **return** "The algorithm failed".

In order to study the average-case complexity of Algorithm 17, we construct a vector $\mathbf{b}$ corresponding to the initial distribution of phase vectors, *i.e.*, whose i-th entry is equal to the expected number of phase vectors that will belong to $\mathscr{P}_i$ at the beginning of the algorithm. We thus have from Lemma 6.1:

$$b_i \stackrel{\text{def}}{=} 2^\ell \mathbb{P}_{\mathscr{O}}[i] = 2^{\ell-i}.$$

We also define the transition matrix $\mathbf{A}$ corresponding to the action on the vector $\mathbf{b}$ of one iteration of the main For loop, from the distribution given in Lemma 6.2. Namely, combining two elements of $\mathscr{P}_i$ will produce one element of $\mathscr{P}_j$ with probability $\mathbb{P}_j[i]$, so we let:

$$\mathbf{A}_{i,j} \stackrel{\text{def}}{=} \begin{cases} \mathbb{P}_j[i] & \text{if } i = n \\ \frac{1}{2}\mathbb{P}_j[i] & \text{otherwise.} \end{cases}$$

The halving factor comes from the fact that we take two elements to produce one, except when we are already in the pool of solutions. It turns out that the matrix $\mathbf{A}$ has a particular form, specified in the following lemma.

Lemma 6.3. $\mathbf{A}$ *is a square matrix of order* n *such that*

$$\mathbf{A} = \begin{pmatrix} \mathbf{N} & \mathbf{0}^T \\ \mathbf{u} & 1 \end{pmatrix} \tag{6.2}$$

where $\mathbf{0} = (0, \dots, 0)$ *is a vector of size* $n-1$, $\mathbf{u}$ *is a positive vector of size* $n-1$ *and* $\mathbf{N}$ *is a nilpotent matrix of order and index* $n-1$ *(the smallest power of* $\mathbf{N}$ *yielding the zero matrix) whose elements under the diagonal are nonnegative and zero over the diagonal.*

Proof. This directly follows from the definition of $\mathbf{A}$ and Lemma 6.2. □

Analyzing the average-case complexity of Algorithm 17 then boils down to studying the powers of $\mathbf{A}$, since one iteration of the main For loop corresponds to applying the matrix $\mathbf{A}$ on $\mathbf{b}$, two iterations to applying $\mathbf{A}^2$, and so on. It will thus be crucial to have a closed-form expression for powers of the matrix $\mathbf{A}$, which is provided in the following lemma.

Lemma 6.4. *Let* p *be any positive integer, then*

$$\mathbf{A}^p = \begin{pmatrix} \mathbf{N}^p & \mathbf{0}^T \\ \mathbf{u}\left(\sum_{\ell=0}^{p-1} \mathbf{N}^\ell\right) & 1 \end{pmatrix}$$

Proof. This is trivially true for $p = 1$. Assume it holds for some fixed p. Then we have

$$\mathbf{A}^p\mathbf{A} = \begin{pmatrix} \mathbf{N}^p & \mathbf{0}^T \\ \mathbf{u}\left(\sum_{\ell=0}^{p-1} \mathbf{N}^\ell\right) & 1 \end{pmatrix} \begin{pmatrix} \mathbf{N} & \mathbf{0}^T \\ \mathbf{u} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{N}^{p+1} & \mathbf{0}^T \\ \mathbf{u} + \mathbf{u}\left(\sum_{\ell=1}^{p} \mathbf{N}^\ell\right) & 1 \end{pmatrix}$$

concluding the proof by induction. □

Recall that our goal is to determine the quantities $\mathcal{Q}_i$. Since solutions can accumulate in the last pool $\mathscr{P}_n$, all we have to do is look at the number of elements in $\mathscr{P}_n$ at the end of the algorithm to obtain $\mathcal{Q}_n$. However, for any other pool $\mathscr{P}_i$, it is necessary to accumulate the number of elements that go through $\mathscr{P}_i$ over the iterations of the main For loop in order to find back the value of $\mathcal{Q}_i$.

Actually, if we denote $\overline{\mathcal{Q}}$ the vector $(\overline{\mathcal{Q}}_1, \dots, \overline{\mathcal{Q}}_n)$ with $\overline{\mathcal{Q}}_i \overset{\text{def}}{=} \mathbb{E}[\mathcal{Q}_i]$, and if we let $\mathbf{Q} \overset{\text{def}}{=} \sum_{p=0}^{n-2} \mathbf{N}^p$, we have the following lemma.

Lemma 6.5. *We have*

$$\overline{\mathcal{Q}} \overset{def}{=} \tilde{\mathbf{Q}}\mathbf{b}^T \qquad \textit{where} \qquad \tilde{\mathbf{Q}} = \begin{pmatrix} \mathbf{Q} & \mathbf{0}^T \\ \mathbf{u}\mathbf{Q} & 1 \end{pmatrix}$$

Now, in order to analyze the average-case complexity of Algorithm 17, it remains to determine $\overline{\mathcal{Q}}$ and make sure to choose a value for ℓ such that $\overline{\mathcal{Q}}_i$ is greater or equal to 2 for every $i \in [\![1, n-1]\!]$ and greater or equal to 1 when $i = n$.

Thanks to Lemma 6.5, we can easily find the average-case complexity of Algorithm 17 and thus prove the following theorem.

Theorem 6.1. *Algorithm 17 runs in* $\Theta\left(2^{0.415n}\right)$ *average-time with a similar amount of queries and qubits.*

It directly follows that finding the whole secret s with Algorithm 17 would all the same require $\Theta\left(2^{0.415n}\right)$ time, space and queries.

1.1 Proof of Theorem 6.1

We start our analysis by giving an expression for the entries of a power of the nilpotent matrix $\mathbf{N}$ we are working with.

Lemma 6.6. *Let p be a positive integer. We have for all $i \in [\![1, n-1]\!]$ and $j \in [\![1, i-1]\!]$:*

$$(\mathbf{N}^p)_{i,j} = \binom{i-j-1}{p-1} 2^{j-i-p}.$$

Proof. We have for any square matrix $\mathbf{M}$ of order $n-1$ and integer p greater than 1:

$$(\mathbf{M}^p)_{i,j} = \sum_{k_1,\ldots,k_{p-1} \in [\![1,n]\!]} \left(\prod_{\ell=1}^{p} \mathbf{M}_{k_{\ell-1},k_\ell} \right)$$

with $k_0 = i$ and $k_p = j$. When taking $\mathbf{M} = \mathbf{N}$, where $\mathbf{N}$ is a nilpotent matrix whose elements under the diagonal are nonnegative and zero over the diagonal, we obtain:

$$(\mathbf{N}^p)_{i,j} = \sum_{j < k_{p-1} < \cdots < k_1 < i} \left(\prod_{\ell=1}^{p} \mathbf{N}_{k_{\ell-1},k_\ell} \right).$$

From Lemma 6.2, we have

$$\mathbf{N}_{k_{\ell-1},k_\ell} = \mathbf{A}_{k_{\ell-1},k_\ell} = 2^{k_\ell - k_{\ell-1} - 1}$$

so the product simplifies to

$$\prod_{\ell=1}^{p} \mathbf{N}_{k_{\ell-1},k_\ell} = 2^{k_p - k_0 - p} = 2^{j-i-p}.$$

Thus we have

$$(\mathbf{N}^p)_{i,j} = 2^{j-i-p} \sum_{j < k_{p-1} < \cdots < k_1 < i} 1 = \binom{i-j-1}{p-1} 2^{j-i-p} \ .$$

□

We can now give an expression for the entries of the matrix $\tilde{\mathbf{Q}}$.

Lemma 6.7. *We have for $i, j \in [\![1, n]\!]$:*

$$\tilde{\mathbf{Q}}_{i,j} = \begin{cases} 0 & \text{if} \quad j > i \\ 1 & \text{if} \quad j = i \\ \frac{1}{3}\left(\frac{3}{4}\right)^{i-j} & \text{otherwise.} \end{cases}$$

Proof. Let $i, j \in [\![1, n-1]\!]$. We have by definition and Lemma 6.6:

$$\begin{aligned} \mathbf{Q}_{i,j} &= \sum_{p=1}^{i-j} \binom{i-j-1}{p-1} 2^{j-i-p} \\ &= \frac{1}{4^{i-j}} \sum_{p=0}^{i-j-1} \binom{i-j-1}{p} 2^{i-j-1-p} \\ &= \frac{3^{i-j-1}}{4^{i-j}} \end{aligned}$$

Now, let $j \in [\![1, n-1]\!]$. We have

$$\begin{aligned} (\mathbf{uQ})_j &= \sum_{i=1}^{n-1} u_i \mathbf{Q}_{i,j} \\ &= u_j + \sum_{i=j+1}^{n-1} u_i \frac{3^{i-j-1}}{4^{i-j}} \\ &= 2^{j-(n+1)} + \frac{2^{2j-(n+1)}}{3^{j+1}} \sum_{i=j+1}^{n-1} \left(\frac{3}{2}\right)^i \\ &= 2^{j-(n+1)} + \frac{2^{2j-(n+1)}}{3^{j+1}} \left(\frac{3^n}{2^{n-1}} - \frac{3^{j+1}}{2^j}\right) \\ &= \frac{3^{n-j-1}}{4^{n-j}} \end{aligned}$$

Together with the definition of $\tilde{\mathbf{Q}}$ in Lemma 6.5, we conclude the proof. □

Now that we have a closed-form expression for the entries of $\tilde{\mathbf{Q}}$ and since we know $\mathbf{b}$, we can look at $\overline{\mathcal{Q}}$.

Lemma 6.8. *Let $i \in [\![1, n]\!]$. We have*

$$\overline{\mathcal{Q}}_i = \frac{3^{i-1}}{2^{2i-1}}.$$

Proof. For any $i \in [\![1, n]\!]$, we have by Lemma 6.5:

$$\begin{aligned}
\overline{\mathcal{Q}_i} &\overset{\text{def}}{=} \tilde{\mathbf{Q}}_i \cdot \mathbf{b} \\
&= \sum_{j=1}^{n} \tilde{\mathbf{Q}}_{i,j} b_j \\
&= \frac{1}{2^i} + \sum_{j=1}^{i-1} \frac{1}{3} \left(\frac{3}{4}\right)^{i-j} \frac{1}{2^j} \quad \text{by Lemma 6.7} \\
&= \frac{1}{2^i} + \frac{3^{i-1}}{2^{2i}} \sum_{j=1}^{i-1} \left(\frac{2}{3}\right)^j \\
&= \frac{1}{2^i} + \frac{3^{i-1}}{2^{2i}} \left(2 - \frac{2^i}{3^{i-1}}\right) \\
&= \frac{3^{i-1}}{2^{2i-1}}
\end{aligned}$$

Concluding the proof. □

Proof of Theorem 6.1. We want at least 2 phase vectors in each pool except in the last one, where we can have just one phase vector, *i.e.*, one solution. In other words, we want:

$$\begin{cases} 3^{i-1} 2^{\ell-2i+1} & \geq 2 \qquad \forall i \in [\![1, n-1]\!] \\ 3^{n-1} 2^{\ell-2n+1} & \geq 1 \end{cases}$$

i.e.,

$$\begin{cases} \ell & \geq 2i - \log(3)(i-1) = (2 - \log(3))i + \log 3 \\ \ell & \geq 2n - \log(3)(n-1) - 1 = (2 - \log(3))n + \log 3 - 1 \end{cases}$$

Taking $\ell = (2 - \log(3))n$ satisfies all these inequations, meaning that we should have an input set of approximately $2^{0.415n}$ phase vectors in order for Algorithm 17 to output the state $\left|\psi_{N/2}\right\rangle$ on expectation. □

1.2 Experimental Results

From the study of Algorithm 17 in this section, we conjecture that the complexity of our algorithm is of the same order. Specifically,

Conjecture 6.1. *Using Algorithm 15, one can solve the* DCP *in* $\tilde{\Theta}\left(2^{0.415n}\right)$ *time and queries, with at most* n *qubits.*

We implemented the classical part of Algorithm 15 in Python, picking uniformly at random numbers in $\mathbb{Z}_N$ (instead of generating them with the quantum oracle) and

Listing 6.1: Python code for Algorithm 15

```
from math import log2
from random import randint

def algo(n):
    N = 2**n
    pools = [[] for _ in range(n)]
    n_queries = 0
    pool_idx = 0

    while True:

        while len(pools[pool_idx]) == 2:
            # if a pool has size 2, combine the elements
            b = (1 - 2 * randint(0,1))
            k = (pools[pool_idx].pop() +
b * pools[pool_idx].pop()) % N
            if k != 0:
                pool_idx = int(log2(k & -k))
                pools[pool_idx].append(k)

        if N/2 in pools[n-1]:
            # if the target has been produced, return it
            break

        # else, query the oracle
        k = randint(0,N-1)
        n_queries += 1
        while k == 0:
            k = randint(0,N-1)
            n_queries += 1
        pool_idx = int(log2(k & -k))
        pools[pool_idx].append(k)

    return n_queries
```

choosing uniformly at random between the sum and the difference when combining two phase vectors. The code is given hereafter.

We then ran this implementation a thousand times on small values of n (from 5 up to 35) and computed the mean query complexity. The results are given in Table 6.1. They are indeed of the order of $2^{(2-\log 3)n}$, and more precisely they appear to be very close to $2^{(2-\log 3)n+1}$.

Note that the time complexity is necessarily of the same order as the query complexity. There cannot be more than n combination steps before calling the oracle a new time, so in the worst case the time complexity is n times the query complexity.

2 A Space-Interpolation Algorithm

We have just described an algorithm using at most n qubits, which we have matched to pools of size 2^i, for i ranging from 0 to $n-1$. A very simple observation is that if we have more qubits, we can subdivide these pools. We will do this uniformly, in order to keep the size of the subpools equal to a power of 2, and at least 2, so we can have two different elements to combine when falling in such a subpool. We introduce a parameter m such that each pool will have 2^m subpools, with the exception of those that are too small which will be divided in subpools of 2 elements.

Formally, for any $i \in [\![1, n-1]\!]$, we will divide $\mathscr{P}_i$ in $2^{\min(n-i-1,m)}$ subpools, where m will be chosen such that the total number of subpools will be as close to the number X of qubits we have at our disposal as possible, *i.e.*, m is the greatest integer such that

$$\begin{aligned} X &\geq 1 + \sum_{i=1}^{n-1} \varphi(i) \\ &= 1 + \sum_{i=1}^{n-m-2} 2^m + \sum_{i=n-m-1}^{n-1} 2^{n-i-1} \\ &= 1 + (n-m-2)2^m + 2^{m+1} - 1 \\ &= (n-m)2^m \end{aligned}$$

We will assume that $m > 0$ since the case $m = 0$ corresponds to the algorithm without subpools, *i.e.*, the algorithm described in the previous section. We will place a couple $(k, |\psi_k\rangle)$ in a subpool denoted by $\mathscr{P}_{(i,v_i)}$, where $i \in [\![1, n]\!]$ and $v_i \in [\![0, \varphi(i)-1]\!]$, in the following manner:

$$\mathscr{P}_{(i,v_i)} \longleftarrow (k, |\psi_k\rangle) \quad \text{if} \quad k \in \mathcal{D}_{(i,v_i)} \quad \text{where}$$

$$\mathcal{D}_{(i,v_i)} \stackrel{\text{def}}{=} \left\{ (2(\varphi(i)\alpha + v_i) + 1)2^i, \alpha \in \left[\!\left[0, \frac{2^{n-i}}{\varphi(i)} - 1\right]\!\right] \right\} \qquad \text{and} \qquad \varphi(i) \stackrel{\text{def}}{=} 2^{\min(n-i-1,m)}$$

n	Query exponent	$0.415n + 1$	Abs. difference
5	2.91	3.07	0.16
6	3.37	3.49	0.12
7	3.75	3.9	0.15
8	4.22	4.32	0.1
9	4.68	4.73	0.05
10	5.08	5.15	0.07
11	5.5	5.56	0.06
12	5.94	5.98	0.04
13	6.35	6.39	0.04
14	6.74	6.81	0.07
15	7.18	7.22	0.05
16	7.61	7.64	0.03
17	8.06	8.05	0.01
18	8.5	8.47	0.03
19	8.89	8.88	0.0
20	9.26	9.3	0.04
21	9.68	9.71	0.03
22	10.1	10.13	0.03
23	10.55	10.54	0.01
24	10.94	10.96	0.02
25	11.29	11.38	0.08
26	11.79	11.79	0.0
27	12.17	12.21	0.03
28	12.63	12.62	0.01
29	13.04	13.04	0.0
30	13.38	13.45	0.07
31	13.82	13.87	0.05
32	14.25	14.28	0.03
33	14.68	14.69	0.02
34	15.1	15.11	0.01
35	15.52	15.52	0.01

Table 6.1: The mean query complexity exponent of Algorithm 15 (1000 shots) is given and compared to $0.415n + 1$. The rounded absolute difference between both values (taken before rounding) is given in the last column.

Note that we do not subdivide $\mathscr{P}_n$, since $\mathcal{D}_n$ contains only one element, but we will refer to it as $\mathscr{P}_{(n,0)}$ for notation consistency. Note also that $\varphi(i)$ is defined such that subpools will correspond to sets of elements of size at least 2, because we want to combine different phase vectors, even in the smallest subpools.

We now present our generalized algorithm for finding one bit of the secret s, from phase vectors produced by an oracle $\mathscr{O}$. The pseudocode for this algorithm is given by Algorithm 18.

Algorithm 18 Our generalized algorithm

Require: An oracle $\mathscr{O}$ outputting uniformly at random $|\psi_k\rangle$ with $k \in \mathbb{Z}_N$.
Ensure: $\left|\psi_{N/2}\right\rangle$.
1: **repeat**
2: **while** for all (i, v_i), $\left|\mathscr{P}_{(i,v_i)}\right| < 2$ **do**
3: Call $\mathscr{O}$ and insert the result in the appropriate pool.
4: Let (i, v_i) be such that $\left|\mathscr{P}_{(i,v_i)}\right| = 2$.
5: Pick $|\psi_a\rangle$ and $|\psi_b\rangle$ from $\mathscr{P}_{(i,v_i)}$.
6: Combine $|\psi_a\rangle$ and $|\psi_b\rangle$ and insert the result in the appropriate pool
7: **until** there is a state in the bin $\mathscr{P}_{(n,0)}$
8: **return** $\left|\psi_{N/2}\right\rangle$.

As for the previous section, we give the probability distributions corresponding to the oracle outputs and the combination process.

Probability distributions. The probability distribution of elements picked uniformly at random from the oracle is given by the following lemma.

Lemma 6.9 (Initial distribution)**.** *Let $i \in [\![1, n]\!]$ and $v_i \in [\![0, \varphi(i) - 1]\!]$. The probability that the oracle $\mathscr{O}$ yields a phase vector belonging to the subpool $\mathscr{P}_{(i,v_i)}$, denoted by $\mathbb{P}_{\mathscr{O}}[(i, v_i)]$, is*

$$\mathbb{P}_{\mathscr{O}}[(i, v_i)] \stackrel{def}{=} \frac{1}{\varphi(i)2^i}$$

Proof. This is straightforward since $\mathbb{P}_{\mathscr{O}}[(i, v_i)] = \frac{|\mathcal{D}_{(i,v_i)}|}{|\mathbb{Z}_N|} = \frac{\frac{2^{n-i}}{\varphi(i)}}{2^n}$. □

The probability distribution of elements obtained from combining two phase vectors of a same subpool is given by the following lemma.

Lemma 6.10 (Combination distribution (subpools))**.** *Let $i \in [\![1, n-1]\!]$ and $v_i \in [\![0, \varphi(i) - 1]\!]$. Let $j \in [\![i+1, n]\!]$ and $w_j \in [\![0, \varphi(j) - 1]\!]$. The probability that combining two phase vectors from $\mathscr{P}_{(i,v_i)}$ yields a phase vector from $\mathscr{P}_{(j,w_j)}$, denoted by $\mathbb{P}_{(i,v_i)}[(j, w_j)]$, is*

- *when* $i = n - 1$*:*

$$\mathbb{P}_{(i,v_i)}\left[(n,0)\right] = \frac{1}{2}$$

- *when* $n - (m+1) \leq i < n - 1$*:*

$$\mathbb{P}_{(i,v_i)}\left[(j,w_j)\right] = \begin{cases} \frac{1}{4} & \text{if } (j,w_j) = (n,0) \\ \frac{1}{2} & \text{if } (j,w_j) = (i+1, v_i \bmod \varphi(i+1)) \end{cases}$$

- *when* $i < n - (m+1)$*:*

$$\mathbb{P}_{(i,v_i)}\left[(j,w_j)\right] = \begin{cases} \frac{1}{4} & \text{if } (j,w_j) = (i+1, v_i \bmod \varphi(i+1)) \\ \frac{1}{4} & \text{if } (j,w_j) = (i+1, (v_i + 2^{m-1}) \bmod \varphi(i+1)) \\ \frac{2^{m+i-j-1}}{\varphi(j)} & \text{for } w_j \in [\![0, \varphi(j)-1]\!] \text{ for } j \in [\![i+m+1, n]\!] \end{cases}.$$

When $j \in [\![1, i]\!]$*, the probability is always zero.*

Please note that v_i depends on i and w_j depends on j, but also on i and v_i. For the sake of clarity, we dropped the latter from the index of w.

Proof. Let $\mathbb{P}^{+}_{(i,v_i)}\left[(j,w_j)\right]$ (resp. $\mathbb{P}^{-}_{(i,v_i)}\left[(j,w_j)\right]$) be the probability for two phase vectors from $\mathscr{P}_{(i,v_i)}$ to produce one from $\mathscr{P}_{(j,w_j)}$ when combined and their sum (resp. difference) is obtained.
Let $k \in \mathcal{D}_{(i,v_i)}$. There exists $\alpha \in \left[\!\left[0, \frac{2^{n-i}}{\varphi(i)} - 1\right]\!\right]$ such that

$$k = (2(\varphi(i)\alpha + v_i) + 1)2^i.$$

We first look for the proportion $\mathbb{P}^{+}_{(i,v_i)}\left[(j,w_j)\right]$ of $\ell \in \mathcal{D}_{(i,v_i)}$, which we can write for some $\beta \in \left[\!\left[0, \frac{2^{n-i}}{\varphi(i)} - 1\right]\!\right]$ as

$$\ell = (2(\varphi(i)\beta + v_i) + 1)2^i,$$

such that

$$k + \ell = (\varphi(i)(\alpha+\beta) + 2v_i + 1)2^{i+1} \in \mathcal{D}_{(j,w_j)},$$

i.e., there must exist $\gamma \in \left[\!\left[0, \frac{2^{n-j}}{\varphi(j)} - 1\right]\!\right]$ such that

$$(\varphi(i)(\alpha+\beta) + 2v_i + 1)2^{i+1} = (2(\varphi(j)\gamma + w_j) + 1)2^j. \tag{6.3}$$

Necessarily, $j \geq i+1$. We have three cases to consider. Recall that $\varphi(i) \overset{\text{def}}{=} 2^{\min(n-(i+1),m)}$.

- When $i < n-(m+1)$, we have $\varphi(i) = 2^m$. Equation 6.3 becomes
$$2^m(\alpha+\beta) + 2v_i + 1 = (2w_j+1)2^{j-(i+1)} \bmod \varphi(j)2^{j-i}$$
Necessarily, $2^{j-(i+1)}$ has to be odd, so $j = i+1$. The condition can then be simplified to
$$2^{m-1}(\alpha+\beta) + v_i = w_j \bmod \varphi(j)2^{j-i} \ .$$
We deduce that
$$\mathbb{P}^+_{(i,v_i)}\left[(j, v_i \bmod \varphi(i+1))\right] = \frac{1}{2} \quad \text{and} \quad \mathbb{P}^+_{(i,v_i)}\left[\left(j, v_i + 2^{m-1} \bmod \varphi(i+1)\right)\right] = \frac{1}{2}.$$
- When $n-(m+1) \leq i < n-1$, we have $\varphi(i) = 2^{n-(i+1)}$ and $\varphi(j) = 2^{n-(j+1)}$. Equation 6.3 becomes
$$(\alpha+\beta)2^n + v_i 2^{i+2} + 2^{i+1} = \gamma 2^n + w_j 2^{j+1} + 2^j$$
Necessarily $j = i+1$ and it follows that
$$\mathbb{P}^+_{(i,v_i)}\left[(i+1, v_i)\right] = 1.$$
- When $i = n-1$, j can only be equal to n and we have $\varphi(i) = \varphi(j) = 1$. Equation 6.3 becomes
$$\alpha + \beta + 2v_i + 1 = 2\gamma + 2w_j + 1$$
implying that $\alpha + \beta = 0 \bmod 2$. It follows that
$$\mathbb{P}^+_{(n-1,0)}\left[(n, 0)\right] = \frac{1}{2}.$$

We use the same reasoning for the difference, $\mathbb{P}^-_{(i,v_i)}\left[(j, w_j)\right]$ being the proportion of $\ell \in \mathcal{D}_{(i,v_i)}$ such that
$$|k-\ell| = |\alpha-\beta|\varphi(i)2^{i+1} \in \mathcal{D}_{(j,w_j)},$$
i.e., there must exist $\gamma \in \left[\!\left[0, \frac{2^{n-j}}{\varphi(j)} - 1\right]\!\right]$ such that
$$|\alpha-\beta|\varphi(i)2^{i+1} = (2(\varphi(j)\gamma + w_j) + 1)2^j. \tag{6.4}$$
Necessarily, $j \geq i+1$. Now we have three cases to consider.

- When $i < n-(m+1)$, we have $\varphi(i) = 2^m$. Equation 6.4 becomes
$$|\alpha-\beta|2^m = (2w_j+1)2^{j-(i+1)} + \gamma\varphi(j)2^{j-i}$$
Necessarily, $j \geq i+m+1$. We can rewrite the condition as
$$|\alpha-\beta| = (2w_j+1)2^{j-m-(i+1)} \bmod \varphi(j)2^{j-m-i}$$
It follows that
$$\mathbb{P}^-_{(i,v_i)}\left[(j, w_j)\right] = \frac{2^{m+i-j}}{\varphi(j)}$$

- When $n-(m+1) \leq i < n-1$, we have $\varphi(i) = 2^{n-(i+1)}$ and $\varphi(j) = 2^{n-(j+1)}$. Equation 6.4 becomes
$$|\alpha - \beta| 2^n = (2w_j + 1)2^j + \gamma 2^n$$
implying that $j = n$. It follows that
$$\mathbb{P}^-_{(i,v_i)}\left[(n,0)\right] = \frac{1}{2}.$$
The case $i = n-1$ is similar and leads to the same result.

- When $i = n-1$, j can only be equal to n and we have $\varphi(i) = \varphi(j) = 1$. Equation 6.4 becomes
$$|\alpha - \beta| = 2\gamma + 2w_j + 1$$
implying that $\alpha + \beta = 1 \bmod 2$. It follows that
$$\mathbb{P}^-_{(n-1,0)}\left[(n,0)\right] = \frac{1}{2}.$$

By definition, the overall probability $\mathbb{P}_{(i,v_i)}\left[(j,w_j)\right]$ is
$$\mathbb{P}_{(i,v_i)}\left[(j,w_j)\right] = \frac{1}{2}\left(\mathbb{P}^+_{(i,v_i)}\left[(j,w_j)\right] + \mathbb{P}^-_{(i,v_i)}\left[(j,w_j)\right]\right)$$
which allows us to conclude the proof. □

On the Markov chain of Algorithm 18. As we said in the previous section, the right way to study this algorithm would also be to analyse the corresponding Markov chain. Once again, this chain has an exponential number of states, which makes it difficult to study. Indeed, we have:

- the configurations were a solution have been found, *i.e.*, there is 1 element in $\mathscr{P}_{(n,0)}$, coming either from the oracle (which means before calling the oracle, there was no subpool with 2 elements) or either from a combination (meaning that 2 elements from a subpool were used to produce the solution, leaving no subpool with 2 elements) are $2^{(n-m)2^m-1}$ of them,

- the configurations for which no solution has been found, corresponding either to a situation where a query must be made (*i.e.*, no subpool contains 2 elements), or to a situation where a combination must be made (*i.e.*, there is a subpool which contains 2 elements). There are therefore $2^{(n-m)2^m-1} + \binom{(n-m)2^m-1}{1} 2^{(n-m)2^m-2} = \left((n-m)2^m+1\right) 2^{(n-m)2^m-2}$ of these configurations.

In total, the Markov chain corresponding to our algorithm has $\left((n-m)2^m+3\right) 2^{(n-m)2^m-2}$ states. This is a prohibitive quantity, even though studying the mean hitting time of this Markov chain would give us the exact average complexity of our algorithm.

Our analysis. The algorithm we will actually study, in the same way as we did in the previous section, is Algorithm 19.

Algorithm 19 The algorithm we study (parallel version)

Require: A pool $\mathscr{P}$ of 2^ℓ phase vectors.
Ensure: $\left|\psi_{N/2}\right\rangle$.
1: **for** $t = 1$ to $n-1$ **do**
2: Take the phase vectors from $\mathscr{P}$ and classify them in the subpools $\mathscr{P}_{(i,v_i)}$
3: **for** $i = 1$ to $n-1$ **do**
4: **while** $\left|\mathscr{P}_{(i,v_i)}\right| \geq 2$ **do**
5: Pick $|\psi_a\rangle$ and $|\psi_b\rangle$ from $\mathscr{P}_{(i,v_i)}$.
6: Combine $|\psi_a\rangle$ and $|\psi_b\rangle$ and insert the result in $\mathscr{P}$
7: **if** $\mathscr{P}_{(n,0)} \neq \emptyset$ **then return** $\left|\psi_{N/2}\right\rangle$
8: **return** "The algorithm failed".

We construct a vector $\tilde{\mathbf{b}}$ corresponding to the initial distribution of phase vectors, *i.e.*, whose (i, v_i)-th entry is equal to the expected number of phase vectors that will belong to $\mathscr{P}_{(i,v_i)}$ at the beginning of the algorithm. We thus have from Lemma 6.9:

$$\tilde{b}_{(i,v)} = 2^\ell \mathbb{P}_{\mathscr{O}}\left[(i,*)\right] = \frac{2^{\ell-i}}{\varphi(i)}$$

We also define the transition matrix $\tilde{\mathbf{A}}$ corresponding to the action on average on the vector $\tilde{\mathbf{b}}$ of one iteration of the main For loop, from the distribution given in Lemma 6.10. Namely, combining two elements of $\mathscr{P}_{(i,v_i)}$ will produce one element of $\mathscr{P}_{(j,w_j)}$ with probability $\mathbb{P}_{(j,w_j)}\left[(i,v_i)\right]$, so we let:

$$\tilde{\mathbf{A}}_{(i,v_i),(j,w_j)} = \begin{cases} \mathbb{P}_{(j,w_j)}\left[(i,v_i)\right] & \text{if } i = n \\ \frac{1}{2}\mathbb{P}_{(j,w_j)}\left[(i,v_i)\right] & \text{otherwise.} \end{cases}$$

The halving factor comes from the fact that we take two elements to produce one, except when we are already in the pool of solutions.

We can actually simplify a great deal this analysis by considering the combination distribution from pool to pool rather than from subpool to subpool.

Lemma 6.11. *Let $t \in [\![0, n-1]\!]$. We have for any $i \in [\![1, n]\!]$ and $v_i, v_i' \in [\![0, \varphi(i)-1]\!]$:*

$$(\tilde{\mathbf{A}}^t \mathbf{b}^T)_{(i,v_i)} = (\tilde{\mathbf{A}}^t \mathbf{b}^T)_{(i,v_i')}.$$

Proof. This is true by definition of $\mathbf{b}$ for $t = 0$ (since $b_{(i,v_i)}$ does not depend on v_i). Now assume this is true for some $t \in [\![0, n-2]\!]$. Using Lemma 6.10 and its notation, we want

to prove that for any $j \in [\![1, n]\!]$ and $w_j, w'_j \in [\![0, \varphi(j) - 1]\!]$:

$$(\tilde{\mathbf{A}}^{t+1}\mathbf{b}^T)_{(j,w_j)} = (\tilde{\mathbf{A}}^{t+1}\mathbf{b}^T)_{(j,w'_j)}.$$

We take back the different configurations of Lemma 6.10 for i:

- If $i = n - 1$, we have only one subpool since $j = n$ (*i.e.*, $w_j = w'_j = 0$), so the property trivially holds.
- If $n - (m + 1) \leq i < n - 1$, the case $j = n$ is once again trivial. For the other case, we first note that $\varphi(i) = 2^{n-i-1}$ and $\varphi(i+1) = 2^{n-i-2}$. Thus, any subpool indexed by (j, w_j) with $j = i + 1$ and $w_j \in [\![0, \varphi(i + 1) - 1]\!]$ has exactly two preimages by $\tilde{\mathbf{A}}$: those are (i, w_j) and $(i, w_j + 2^{n-i-2})$, and the probability to go from one of the two to the considered image is constant. Therefore, the property holds.
- If $i < n - (m + 1)$, we have $\varphi(i) = \varphi(i + 1) = 2^m$. Any subpool indexed by (j, w_j) with $j = i + 1$ and $w_j \in [\![0, \varphi(i + 1) - 1]\!]$ has once again exactly two preimages by $\tilde{\mathbf{A}}$: those are (i, w_j) and $(i, w_j + 2^{m-1} \bmod 2^m)$, and the probability to go from one of the two to the considered image is once again constant. For the remaining case, we can see that every subpool (i, v_i) has for images every subpool indexed by (j, w_j) for $w_j \in [\![0, \varphi(j) - 1]\!]$, for $j \in [\![i + m + 1, n]\!]$ with a probability that does not depend on v_i nor w_j, meaning that every subpool of a same pool will be attained with the same probability, *i.e.*, the property holds.

□

It follows from this important lemma that we can in some sense merge in our analysis the subpools of each pool. This allows us to once again work with a transition matrix between the n pools, instead of working with a transition matrix between the $(n - m)2^m$ subpools. Namely, we give in the following lemma the transition probabilities between pools.

Corollary 6.1 (Combination distribution (pools))**.** *Let $i \in [\![1, n - 1]\!]$ and $j \in [\![i + 1, n]\!]$. The probability that combining two phase vectors from $\mathscr{P}_{(i,*)}$ yields a phase vector from $\mathscr{P}_{(j,*)}$ (where the asterisk means that it can be any of the subdivisions of the considered pool), denoted by $\mathbb{P}_i[j]$, is*

- *when $i = n - 1$:*
$$\mathbb{P}_i[n] = \frac{1}{2}$$
- *when $n - (m + 1) \leq i < n - 1$:*
$$\mathbb{P}_i[j] = \begin{cases} \frac{1}{4} & \text{if } j = n \\ \frac{1}{2} & \text{if } j = i + 1 \end{cases}$$

- *when* $i < n - (m+1)$*:*

$$\mathbb{P}_i[j] = \begin{cases} \frac{1}{2} & \text{if } j = i+1 \\ 2^{m+i-j-1} & \text{for } j \in [\![i+m+1, n]\!] \end{cases}.$$

When $j \in [\![1, i]\!]$*, the probability is always zero.*

Proof. We simply take back the probability distribution of Lemma 6.10 and forget about the subpools indexes. When $i < n-(m+1)$, we merge the $1/4$ probabilities of falling into $\mathscr{P}_{(i+1, v \bmod \varphi(i+1))}$ and into $\mathscr{P}_{(i+1,(v+2^{m-1}) \bmod \varphi(i+1))}$: we simply have a $1/2$ probability of falling into $\mathscr{P}_{(i,*)}$. The same applies when $j \in [\![i+m+1, n]\!]$: we sum every probability to fall into $\mathscr{P}_{(j,*)}$, *i.e.*, we sum $\varphi(j)$ times $\frac{2^{m+i-j-1}}{\varphi(j)}$, leading to a probability of $2^{m+i-j-1}$ to go from $\mathscr{P}_{(i,*)}$ to $\mathscr{P}_{(j,*)}$. □

We can then take back our method of analysis and define a new transition matrix from the combination distribution we just gave in Corollary 6.1. Namely, we let, as we did in the previous section, for $i, j \in [\![1, n]\!]$:

$$\mathbf{A}_{i,j} \stackrel{\text{def}}{=} \begin{cases} \mathbb{P}_j[i] & \text{if } i = n \\ \frac{1}{2}\mathbb{P}_j[i] & \text{otherwise.} \end{cases}$$

In the exact same way, we define from the probabilities of getting an element in $\mathscr{P}_{(i,*)}$ when calling the oracle a vector $\mathbf{b}$ as follows:

$$b_i \stackrel{\text{def}}{=} 2^{\ell}\varphi(i)\mathbb{P}_{\mathscr{O}}[(i,*)] = 2^{\ell-i}$$

It turns out that the matrix $\mathbf{A}$ has the same form as in Lemma 6.3. We can thus analyze Algorithm 19 as we did for Algorithm 17. Using the same notation, we want to determine $\overline{\mathcal{Q}}$ (as defined in Lemma 6.5 and make sure to choose a value for ℓ such that $\overline{\mathcal{Q}}_i$ is greater or equal to 2 for every $i \in [\![1, n-1]\!]$ and greater or equal to 1 when $i = n$. In the end, we will be able to give a loose estimate of the average-case complexity of Algorithm 19, which is stated in the following conjecture.

Conjecture 6.2. *Algorithm 19 runs in* $O\left(2^{\log 3 \frac{n}{m+1} + 2m}\right)$ *time with the same amount of queries and qubits.*

The $O\left(2^{\log 3 \frac{n}{m+1} + 2m}\right)$ for the time complexity seems actually to be a *very rough* upper bound on the actual complexity of the algorithm, as we will see in the last subsection, where we will give experimental results. For now on, we proceed in the same way as for the analysis of the n-qubits algorithm: we take a closer look at $\mathbf{Q}$ and $\mathbf{uQ}$.

2.1 Clues for Conjecture 6.2

We are soon faced with a problem in carrying out our study: looking at the powers of the nilpotent matrix $\mathbf{N}$ is much less straightforward than in the case where $m = 0$. However, determining $\mathbf{Q}$, the sum of the powers of $\mathbf{N}$, is essential if we want to obtain $\overline{\mathcal{Q}}$. We manage hereafter to obtain the first $\overline{\mathcal{Q}}_i$ values for i ranging from 1 to $m + 1$ and also to give a good estimate for $\overline{\mathcal{Q}}_n$. Unfortunately, all the other $\overline{\mathcal{Q}}_i$ values remain unknown. In the end, we are therefore forced to give a very large estimate for ℓ based on the values we know, but which should be much lower when our study is complete and the $\overline{\mathcal{Q}}$ vector is fully known.

Estimate of $\overline{\mathcal{Q}}_n$. We will use the fact that $(\mathbf{uQ})_j = \left(\mathbf{A}^{n-1}\right)_{n,j}$ to estimate the value of $\overline{\mathcal{Q}}_n$. We first show that the values $(\mathbf{uQ})_j$ are recursively defined and let $\sigma_j \stackrel{\text{def}}{=} (\mathbf{uQ})_j$.

Lemma 6.12. *We have:*

$$\sigma_j = \begin{cases} 1 & \text{if} \quad j = n \\ \frac{1}{4} & \text{if} \quad j = n-1 \\ \frac{1}{8} + \frac{1}{4}\sigma_{j+1} & \text{if} \quad j \in [\![n-m-1, n-2]\!] \\ \frac{1}{4}\sigma_{j+1} + \sum_{i=j+m+1}^{n} 2^{j+m-i-2}\sigma_i & \text{if} \quad j \in [\![1, n-m-2]\!]\,. \end{cases}$$

Proof. We have

$$\mathbf{A}^p = \mathbf{A}^{p-1}\mathbf{A}$$
$$\left(\mathbf{A}^p\right)_{n,j} = \sum_{i=1}^{n} \left(\mathbf{A}^{p-1}\right)_{n,i} \mathbf{A}_{i,j}$$

Taking p to be greater than n, we get $p > p - 1 \geq n - 1$. Since $n - 1$ is the index of the nilpotent matrix $\mathbf{N}$, it follows that

$$\sigma_j = \sum_{i=1}^{n} \sigma_i \mathbf{A}_{i,j} \ .$$

Now, the matrix $\mathbf{N}$ being a lower triangular matrix whose diagonal is zero, we have

$$\sigma_j = \sum_{i=j+1}^{n} \sigma_i \mathbf{A}_{i,j} \ .$$

Plugging the probability distributions given by Corollary 6.1 in this equation, we conclude the proof. □

From this lemma, it is then possible to give a closed-form expression for some of the entries $(\mathbf{uQ})_j$, namely, when $j \in [\![n-m-1, n-1]\!]$.

Corollary 6.2. *Let $j \in [\![n-m-1, n-1]\!]$. We have:*

$$\sigma_j = \frac{1}{6}\left(1 + \frac{2}{4^{n-j}}\right)$$

Proof. The case $j = n-1$ is readily verified. Now for $j \in [\![n-m-1, n-2]\!]$, we have from Lemma 6.12:

$$\begin{aligned}
\sigma_j &= \frac{1}{8} + \frac{1}{4}\sigma_{j+1} \\
&= \frac{1}{8}\sum_{\ell=0}^{k-j-1} \frac{1}{4^\ell} + \frac{1}{4^{k-j}}\sigma_k \qquad \text{for } k > j \\
&= \frac{1}{8}\sum_{\ell=0}^{n-j-2} \frac{1}{4^\ell} + \frac{1}{4^{n-j-1}}\sigma_{n-1} \qquad \text{for } k = n-1 \\
&= \frac{1}{6}\left(1 - \frac{1}{4^{n-j-1}}\right) + \frac{1}{4^{n-j}} \\
&= \frac{1}{6}\left(1 + \frac{2}{4^{n-j}}\right)
\end{aligned}$$

□

Unfortunately, giving a closed-form expression for the entries σ_j becomes increasingly difficult as j decreases. As can be seen in Lemma 6.12, when $j < n-m-2$, σ_j depends on the σ_i's with $i \geq j+m+1$. It follows that we can theoretically give a closed-form expression in packets of $m+1$ consecutive σ_j, but it is simpler to give lower and upper bounds for these. We now give a tight interval in which σ_j can be found for any j.

Lemma 6.13. *Let $\alpha \in [\![0, \lceil \frac{n}{m} \rceil]\!]$. For $j \in [\![n-\alpha(m+1), n-\alpha(m+1)+m]\!] \stackrel{def}{=} \mathscr{I}_\alpha$, we have:*

$$\frac{1}{2 \times 3^\alpha} \leq \sigma_j < \frac{1}{2 \times 3^{\alpha-1}} \ .$$

Proof. Only $j = n$ belongs to $\mathscr{I}_0$ and since $\sigma_n = 1$, the bound is verified for $\alpha = 0$.

By Corollary 6.2, we have for $j \in \mathscr{I}_1 = [\![n-(m+1), n-1]\!]$:

$$\sigma_j = \frac{1}{6}\left(1 + \frac{2}{4^{n-j}}\right) \geq \frac{1}{6}$$

so the bound is also verified for $\alpha = 1$.

From Lemma 6.12, we note that $\sigma_j < \sigma_{j+1}$. Thus, for $\alpha \in [\![2, \lceil \frac{n}{m} \rceil]\!]$, we actually have for all $j \in \mathscr{I}_\alpha$ that

$$\sigma_j \geq \sigma_{n-\alpha(m+1)} \ . \tag{6.5}$$

We now assume that

$$\sigma_{n-\alpha(m+1)} \geq \frac{1}{2 \times 3^\alpha}$$

and prove that this bound holds for $\alpha+1$. We have

$$\begin{aligned}\sigma_{n-(\alpha+1)(m+1)} &= \frac{1}{4}\sigma_{n-(\alpha+1)(m+1)+1} + \sum_{j=n-\alpha(m+1)}^{n} 2^{n-(\alpha+1)(m+1)+m-j-2}\sigma_j \\ &\geq \frac{1}{4}\sigma_{n-(\alpha+1)(m+1)} + \sum_{j=n-\alpha(m+1)}^{n} 2^{n-\alpha(m+1)-j-3}\sigma_j\end{aligned}$$

Thus

$$\begin{aligned}\frac{3}{4}\sigma_{n-(\alpha+1)(m+1)} &\geq 2^{-\alpha(m+1)-3}\sigma_n + \sum_{a=1}^{\alpha}\sum_{j\in\mathscr{I}_a} 2^{n-\alpha(m+1)-i-3}\sigma_j \\ &\geq 2^{-\alpha(m+1)-3} + \sum_{a=1}^{\alpha}\sum_{j\in\mathscr{I}_a} 2^{n-\alpha(m+1)-j-3}\sigma_{n-a(m+1)} \\ &\geq 2^{-\alpha(m+1)-3} + \sum_{a=1}^{\alpha}\sum_{j\in\mathscr{I}_a} \frac{2^{n-\alpha(m+1)-j-4}}{3^a} \\ &= 2^{-\alpha(m+1)-3} + \frac{2^{m+1}-1}{2^{m+4}}\sum_{a=1}^{\alpha}\frac{2^{(a-\alpha)(m+1)}}{3^a} \\ &= 2^{-\alpha(m+1)-3} + \frac{2^{m+1}-1}{2^{m+1}-3}\frac{2^{m+1}-3^\alpha 2^{-(\alpha-1)(m+1)}}{3^\alpha 2^{m+4}} \\ &\geq 2^{-\alpha(m+1)-3} + \frac{1}{8\times 3^\alpha} - 2^{-\alpha(m+1)-3} \\ &= \frac{1}{8\times 3^\alpha}\end{aligned}$$

We obtain

$$\sigma_{n-(\alpha+1)(m+1)} \geq \frac{1}{2\times 3^{\alpha+1}} \ .$$

We conclude the proof by induction, the upper bound being provable in the exact same way. □

From this lemma, we can now give an asymptotic expression for $\overline{\mathcal{Q}}_n$.

Lemma 6.14. *We asymptotically have*

$$\overline{\mathcal{Q}}_n = \Theta\left(3^{-\lceil\frac{n}{m+1}\rceil}\right).$$

Proof.

$$\begin{aligned}\overline{\mathcal{Q}}_n = (\mathbf{uQ}|1)\cdot\mathbf{b} &= b_n + \sum_{1\le j\le n-1}\sigma_j b_j \\ &= 2^{-n} + \sum_{1\le j\le n-1} 2^{-j}\sigma_j \\ &= 2^{-n} + \sum_{a=1}^{\lceil\frac{n}{m+1}\rceil-1}\sum_{j\in\mathscr{I}_a} 2^{-j}\sigma_j + \sum_{j=1}^{n-\lceil\frac{n}{m+1}\rceil(m+1)+m} 2^{-j}\sigma_j\end{aligned}$$

We now use the upper bound of Lemma 6.13 to obtain:

$$\begin{aligned}(\mathbf{uQ}|1)\cdot\mathbf{b} &\ge 2^{-n} + \sum_{a=1}^{\lceil\frac{n}{m+1}\rceil-1} 3^{-a}\sum_{j\in\mathscr{I}_a} 2^{-(j+1)} + \sum_{j=1}^{n-\lceil\frac{n}{m+1}\rceil(m+1)+m} 2^{-(j+1)}3^{-\lceil\frac{n}{m+1}\rceil} \\ &= 2^{-n} + \left(2^{m+1}-1\right)\sum_{a=1}^{\lceil\frac{n}{m+1}\rceil-1} 2^{(a-1)(m+1)-n}3^{-a} + \frac{1-2^{\lceil\frac{n}{m+1}\rceil(m+1)-m-n}}{2\times 3^{\lceil\frac{n}{m+1}\rceil}} \\ &= \frac{1}{2\times 3^{\lceil\frac{n}{m+1}\rceil}} + \sum_{a=0}^{\lceil\frac{n}{m+1}\rceil-1} 2^{a(m+1)-n}3^{-a} - \sum_{a=1}^{\lceil\frac{n}{m+1}\rceil} 2^{(a-1)(m+1)-n}3^{-a} \\ &= \frac{1}{2\times 3^{\lceil\frac{n}{m+1}\rceil}} + \sum_{a=0}^{\lceil\frac{n}{m+1}\rceil-1} 2^{a(m+1)-n+1}3^{-(a+1)} \\ &= \frac{1}{2\times 3^{\lceil\frac{n}{m+1}\rceil}} + \frac{1}{3\times 2^{n-1}}\sum_{a=0}^{\lceil\frac{n}{m+1}\rceil-1}\left(\frac{2^{m+1}}{3}\right)^a \\ &= \frac{1}{2\times 3^{\lceil\frac{n}{m+1}\rceil}} + \frac{1}{2^{n-1}}\frac{\left(\frac{2^{m+1}}{3}\right)^{\lceil\frac{n}{m+1}\rceil}-1}{2^{m+1}-3}\end{aligned}$$

For n and m big enough, we have

$$(\mathbf{uQ}|1)\cdot\mathbf{b} \ge \frac{1}{2\times 3^{\lceil\frac{n}{m+1}\rceil}} + \frac{1}{2^m}\frac{1}{3^{\lceil\frac{n}{m+1}\rceil}}.$$

Using the lower bound of Lemma 6.13 instead of the upper bound, we can also show that

$$(\mathbf{uQ}|1)\cdot\mathbf{b} \le \frac{1}{2\times 3^{\lceil\frac{n}{m+1}\rceil-1}} + \frac{1}{2^m}\frac{1}{3^{\lceil\frac{n}{m+1}\rceil-1}}$$

□

Now that we have an estimate for $(\mathbf{uQ}|1)\cdot\mathbf{b}$, *i.e.* for $\overline{\mathcal{Q}}_n$, we look at the case where $i < n$.

Value of $\overline{\mathcal{Q}}_i$ for $i \in [\![1, m+1]\!]$. It turns out that we can easily give an expression for $\overline{\mathcal{Q}}_i = (\mathbf{Q}_i|0) \cdot \mathbf{b}$ for $i \in [\![1, m+1]\!]$ (where $\mathbf{Q}_i$ denotes the i-th row of the matrix $\mathbf{Q}$), from the probability distribution of the subpools. In this aim, we assume that $m < \frac{n-1}{2}$ (when m is greater, the algorithm uses in any case an exponential amount of qubits, which is not really appealing), meaning that $m < n - (m+1)$. It follows that for $i \in [\![1, m+1]\!]$, $\varphi(i) = 2^m$. We then have a closed-form expression for $\overline{\mathcal{Q}}_i$.

Lemma 6.15. *Let $i \in [\![1, m+1]\!]$. We have*

$$\overline{\mathcal{Q}}_i = \frac{2^i - 1}{2^{m+2i-1}}.$$

Proof. We take back the probability distributions of Lemma 6.9 and Lemma 6.10. In any subpool of the first pool, we have a proportion of $\frac{1}{2\varphi(1)} = \frac{1}{2^{m+1}}$ phase vectors, so the equality is verified for $i = 1$. Now we assume that it holds for a given value $i \in [\![1, m+1]\!]$. Then, for the $i+1$-th pool, we initially have a proportion of $\frac{1}{2^{i+1}\varphi(i+1)}$ phase vectors and we have to add a quarter of the proportion of phase vectors we had in the i-th pool, *i.e.*,

$$\frac{1}{2^{m+i+1}} + \frac{2^i - 1}{2^{m+2i+1}} = \frac{2^{i+1} - 1}{2^{m+2i+1}}$$

concluding the proof by induction. □

The equality given in Lemma 6.15 does not hold anymore for i greater than $m+1$, since we have to take into consideration the probability to come from the $i-(m+1)$-th pool to the i-th one (see Lemma 6.10).

Conclusion. We do not currently have an expression for $\overline{\mathcal{Q}}_i$ for i from $m+2$ to $n-1$. We just use the ones we do have to conclude. So we want at least 2 phase vectors in each subpool except in the last one, where we can have just one phase vector, *i.e.*, one solution. In other words, we want from Lemma 6.14 and Lemma 6.15:

$$\begin{cases} (2^i - 1)2^{\ell - 2i - m + 1} & \geq 2 \qquad \forall i \in [\![1, m+1]\!] \\ 2^\ell 3^{-\left\lceil \frac{n}{m+1} \right\rceil} & \geq 1 \end{cases}$$

i.e.,

$$\begin{cases} \ell & \geq i + m + 1 \qquad \forall i \in [\![1, m+1]\!] \\ \ell & \geq \log(3) \left\lceil \frac{n}{m+1} \right\rceil \end{cases}$$

Taking $\ell = \log(3) \left\lceil \frac{n}{m+1} \right\rceil + 2m$ largely satisfies all these inequations. Once again, some inequations are missing, for $i \in [\![m+2, n-1]\!]$. We will see in the next subsection that a much smaller value for ℓ should be enough, but we do not have for the moment any more precise value.

2.2 Experimental Results

We now give some experimental results, starting with a plot of the mean query complexity exponent (over 1000 shots) of Algorithm 18, depending on the value of m and as a function of n. This is shown in Figure 6.1.

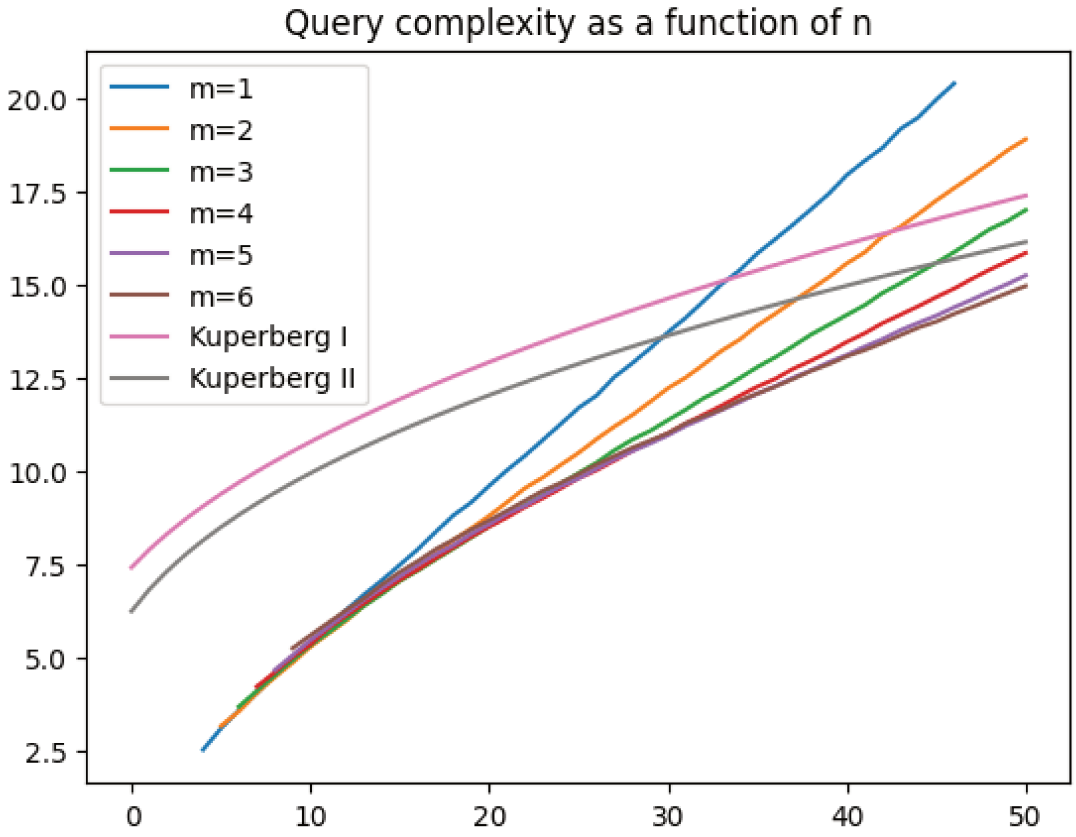

Figure 6.1: The mean query complexity exponent of Algorithm 18 (1000 shots) is given for values of m ranging from 1 to 5, as a function of n. It is compared to the query complexity of Kuperberg's algorithms.

A few remarks can be made:

- first of all, when $m = 1$, we seem to find back a complexity in $0.415n$
- second, as m grows, the asymptotic query complexity seems to get better, but we do not know to what extent
- last but not least, Algorithm 18 seems to be more efficient than both Kuperberg's algorithms for some time depending on the value m.

Now we give the other point of view: we plot the mean query complexity exponent (over 1000 shots) of Algorithm 18, depending on the value of n and as a function of m (for m between 1 and 14, as it suffices to observe what is important). This is shown in Figure 6.2.

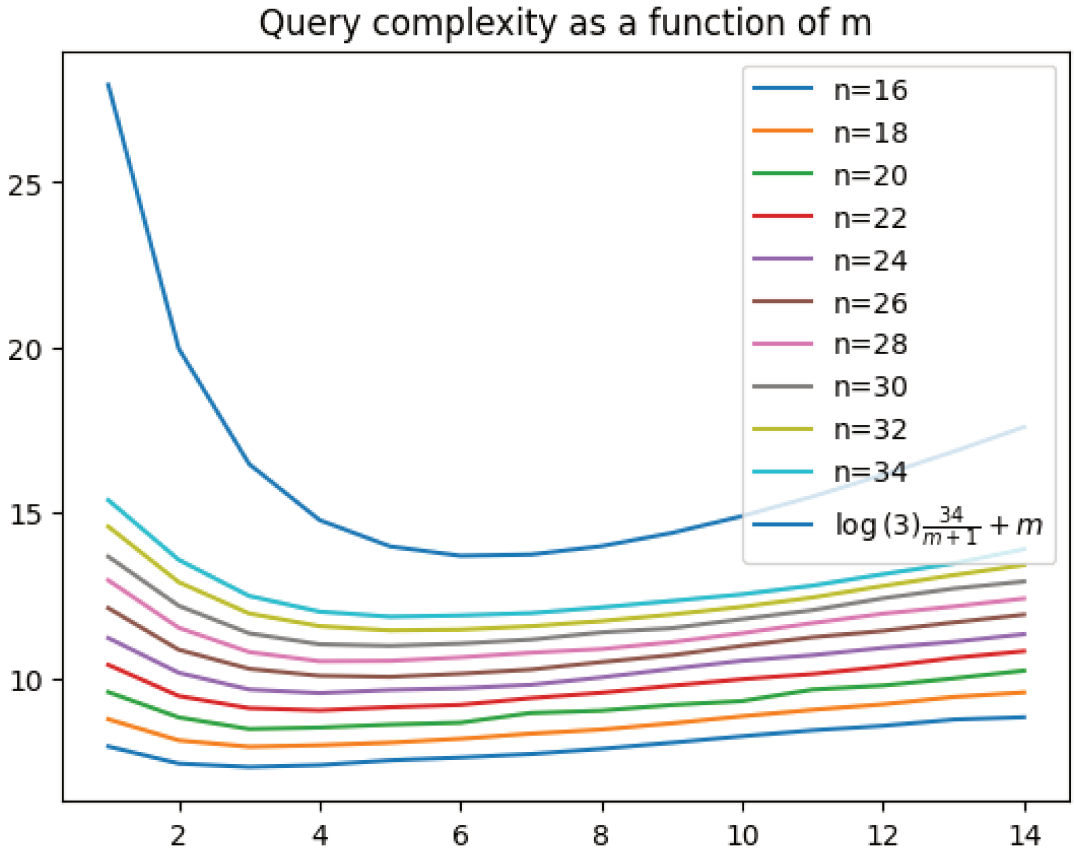

Figure 6.2: The mean query complexity exponent of Algorithm 18 (1000 shots) is given for different values of n between 8 and 34, as a function of m.

Here we can clearly see that for each value of n, there exists an optimal non-trivial value m which gives the lowest query complexity. The main question is obviously what is the value of this optimal m ? Can it be better than $\sqrt{2n}$, meaning that Algorithm 18 would beat Kuperberg's second algorithm (on the query and time complexities, not the space one) ? Or at least could it better or match than the exponent of Kuperberg's first algorithm, which is roughly $1.8\sqrt{n}$?

We also plotted $\log(3)\frac{n}{m+1} + m$, for the value $n = 34$. Please note that for any of the other values for n, the corresponding curve is all the same completely all over the associated query complexity curve. Plotting $\log(3)\frac{n}{m+1} + 2m$, the formula we arrived at in the previous subsection, is absurdly too far beyond the curves obtained in practice,

leading us to assert that this formula is a very rough upper bound of the actual query complexity of our algorithm.

Having the entire vector $\overline{\mathcal{Q}}$ would allow us to refine the value of ℓ and obtain one sufficient for Algorithm 18 to succeed in producing a solution. It seems likely that we would obtain an expression of the form $cst_1 \frac{n}{m+1} + cst_2 m$ for ℓ (where cst_1 and cst_2 are constants to determine). We could then, by equating the two terms, determine the optimal value for m leading to an algorithm with the lowest complexity in time and queries (observed as the minimum in Figure 6.2). Finally, we could of course compare our algorithm with those of Kuperberg.

3 Conclusion

As we have just seen, there is still room for precise analysis of the proposed algorithm. The method used here might not be the most suitable, and even if we were to carry out the calculations to the end, we might not obtain a result corresponding to what we observe experimentally. In fact, the right way to study our algorithm would be to analyse the corresponding Markov chain and its mean hitting time.

We also propose a variant of our algorithm that is *a priori* more efficient. The only difference is in the way the pools are defined. Although it seems more promising than the one presented above, according to the practical experiments we have carried out, it would also involve the study of the underlying Markov chain, or the same kind of process as the one we have used here, only more complicated (in particular, the involved matrix $\mathbf{A}$ does not contain a nilpotent matrix, which made things easier for us in our study). The idea is to put phase vectors whose associated values are close in $\mathbb{Z}_N$ into the same pool instead of doing it according to their least significant bits. More precisely, we define the pools as follows: a couple $(k, |\psi_k\rangle)$ generated by the oracle will be placed in the i-th pool, denoted by $\mathscr{P}_i$ if k is in between 2^i and 2^{i+1}, more precisely:

$$\mathscr{P}_i \longleftarrow (k, |\psi_k\rangle) \quad \text{if} \quad k \in \mathcal{D}_i \quad \text{where}$$

$$\forall i \in [\![0, n-1]\!], \qquad \mathcal{D}_i \stackrel{\text{def}}{=} [\ 2^i, 2^{i+1}\ ).$$

In the same way as we did previously in this chapter, these pools might actually be subdivised in smaller subpools depending on the value m derived from the number of qubits X we have at our disposition. Namely, we will place a couple $(k, |\psi_k\rangle)$ in a subpool denoted by $\mathscr{P}_{(i,v_i)}$, where $i \in [\![0, n-1]\!]$ and $v_i \in \left[\!\left[0, 2^{\min(i-1,m)} - 1\right]\!\right]$, in the following manner:

$$\mathscr{P}_{(i,v_i)} \longleftarrow (k, |\psi_k\rangle) \quad \text{if} \quad k \in \mathcal{D}_{(i,v_i)} \quad \text{where}$$

$$\mathcal{D}_{(i,v_i)} \stackrel{\text{def}}{=} [\ 2^i + v_i 2^{i-\min(i-1,m)}, 2^i + (v_i+1)2^{i-\min(i-1,m)}\ ).$$

Note that with this approach, the solution state that we wish to construct is $|\psi_1\rangle$ instead of $\left|\psi_{N/2}\right\rangle$ and that it will therefore be found in $\mathscr{P}_0$. Note also that we can save a few qubits compared with the previous approach in the following way: when the oracle generates a state with $k \in [\![N/2, N-1]\!]$, we can bring it back into $[\![0, N/2-1]\!]$ at a lower cost, by combining $|\psi_k\rangle$ with $|\psi_N\rangle$ (which is equal to $|+\rangle$). One time out of two, this will give back $|\psi_k\rangle$, in which case you will have to repeat the process until you obtain $|\psi_{N-k}\rangle$. Finally, this trick allows us to deal with any value of N very easily, not only powers of 2.

Conclusion and Perspectives

Conclusion In this thesis, we have seen two new concrete applications of the Quantum Fourier Sampling technique, firstly in complexity theory and secondly in algorithms. In both cases, the implications for cryptography are direct, reminding us of the major importance of studying this method in cryptanalysis, well beyond the famous Shor algorithm. In this respect, we recalled in Chapter 1 the application of Quantum Fourier Sampling in the more general context of solving the HSP in any abelian group.

These fundamental reminders, together with the elementary but no less necessary reminders of code-based cryptography given in Chapter 2, enabled us to give in Chapter 3 a quantum reduction of the problem of finding a low-weight codeword in a code to the problem of decoding in its dual code. This result is an important advance in complexity theory, since no reduction in this direction existed between these problems, whether classical or quantum, and it gives us a better understanding of the links between these problems.

Quantum Fourier Sampling has also been shown to be of interest in the context of dihedral groups and not just abelian groups, as we saw in Chapter 4. We reviewed there the main algorithms that have been milestones for solving the DCP, to which the security of many cryptosystems, whether asymmetric or symmetric, can be reduced, and more generally to which many other problems, whether used in lattice-based or isogeny-based cryptography, can be reduced.

By combining the Quantum Fourier Sampling method with an idea due to Regev for solving the DCP, we have then introduced in the Chapter 5 a new kind of algorithm for solving the HSP in a dihedral group. Namely, we reduced the resolution of the DCP to that of a *quantum* subset-sum problem, which offers a new alternative to the two other main methods of resolution already known (direct resolution or reduction to a classical subset-sum problem). We thus obtained the first algorithm to improve over Ettinger-Hoyer algorithm in the regime of algorithms using a linear number of queries to the oracle. Finally, we also give a very natural interpolation on the number of queries between this new algorithm and Kuperberg's second algorithm.

Finally, in Chapter 6, we give new ways of solving the HSP in a dihedral group without

reducing it to a subset-sum problem. In particular, we present a new algorithm that beats all previously known algorithms in terms of quantum space usage and whose time complexity is the lowest among all algorithms using a linear space. We also give an interpolation between this algorithm and, most probably, Kuperberg's first algorithm. Giving the complexity of this interpolation explicitly is unfortunately a difficult task and requires further work.

Perspectives We have shown that our quantum reduction of the short codeword problem to the decoding problem is interesting when we consider the Hamming metric, but that on the contrary, with the rank metric, it operates in an area where the first problem is already easy. This is because the rank metric is much coarser than the Hamming metric. That said, there is the Lee metric, which is less coarse than the Hamming metric and is making its way into post-quantum cryptography, for which it would be interesting to investigate this reduction. Finally, although this reduction was originally studied and presented with the Euclidean metric, its full potential had not been exploited. As we saw in the Hamming metric, we had to consider going beyond $d_{\mathrm{GV}}/2$ for the decoding problem, where in the lattice setting, [Ste+09] stopped before. It would therefore be interesting to investigate this further for the Euclidean metric.

As we discussed in the last chapter, there are still a number of things to be done with our space-interpolation algorithm and one major idea to be explored. First of all, we need to refine the study of our algorithm so that we can obtain its complexity more precisely, and thus be able to compare it with Kuperberg's algorithms in particular. Analysing the Markov chain corresponding to the algorithm would be optimal but appears to be a complicated task. Secondly, the idea of changing the way in which subpools are defined seems promising and worth looking into, but it also seems all the more difficult to study. It is, however, more intuitive and represents a genuinely new approach compared with Kuperberg's first algorithm.

Bibliography

[AD97] Miklós Ajtai and Cynthia Dwork. "A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence". In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*. 1997, pp. 284–293 (cit. on pp. 1, 62).

[Adj+23] Gora Adj, Stefano Barbero, Emanuele Bellini, Andre Esser, Luis Rivera-Zamarripa, Carlo Sanna, Javier Verbel, and Floyd Zweydinger. *MiRitH*. Round 1 Submission to the NIST Post-Quantum Cryptography Additional Signatures Call. May 2023 (cit. on p. 22).

[Adl79] Leonard Adleman. "A subexponential algorithm for the discrete logarithm problem with applications to cryptography". In: *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. 1979, pp. 55–60 (cit. on p. 5).

[AFS05] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. "A family of fast syndrome based cryptographic hash functions". In: *Ed Dawson, Serge Vaudenay (editors). Progress cryptology-Mycrypt First international conference on cryptology Malaysia, ISBN 978-3-540-28938-8*. Vol. 3715. LNCS. Kuala Lumpur, Malaysia: Springer, Sept. 2005, pp. 64–83 (cit. on p. 29).

[Agu+20] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Maxime Bros, Alain Couvreur, Jean-Christophe Deneuville, Philippe Gaborit, Gilles Zémor, and Adrien Hauteville. *Rank Quasi Cyclic (RQC)*. Second Round submission to NIST Post-Quantum Cryptography call. Apr. 2020 (cit. on pp. 22, 29).

[Ala+20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. "Cryptographic Group Actions and Applications". In: *ASIACRYPT (2)*. Vol. 12492. Lecture Notes in Computer Science. Springer, 2020, pp. 411–439 (cit. on p. 62).

[Ala+22] Gorjan Alagic, David Cooper, Quynh Dang, Thinh Dang, John M. Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl A. Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Daniel Apon. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. 2022 (cit. on p. 62).

[Ale03] Alekhnovich, Michael. "More on Average Case vs Approximation Complexity". In: *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*. IEEE Computer Society, 2003, pp. 298–307 (cit. on p. 19).

[App+17] Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. "Low-Complexity Cryptographic Hash Functions". In: *ITCS*. Vol. 67. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 7:1–7:31 (cit. on pp. 19, 23).

[Ara+18] Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. "A new algorithm for solving the rank syndrome decoding problem". In: *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*. IEEE. 2018, pp. 2421–2425 (cit. on p. 22).

[Ara+19] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zémor, Carlos Aguilar Melchor, Slim Bettaieb, Loïc Bidoux, Magali Bardet, and Ayoub Otmani. *ROLLO (merger of Rank-Ouroboros, LAKE and LOCKER)*. Second round submission to the NIST post-quantum cryptography call. NIST Round 2 submission for Post-Quantum Cryptography. Mar. 2019 (cit. on pp. 22, 29).

[Ara20] Nicolas Aragon. "Cryptographie à base de codes correcteurs d'erreurs en métrique rang et application". PhD thesis. Université de Limoges, Dec. 2020 (cit. on p. 12).

[Ara+23a] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. *RYDE*. Round 1 Submission to the NIST Post-Quantum Cryptography Additional Signatures Call. 2023 (cit. on p. 22).

[Ara+23b] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, and Jean-Pierre Tillich. *MIRA*. Round 1 Submission to the NIST Post-Quantum Cryptography Additional Signatures Call. 2023 (cit. on p. 22).

[Bar13] Razvan Barbulescu. "Algorithms of discrete logarithm in finite fields". PhD thesis. Université de Lorraine, 2013 (cit. on p. 5).

[Bar+14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. "A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic". In: *Advances in Cryptology - EUROCRYPT 2014*. Vol. 8441. LNCS. Copenhagen, Denmark: Springer, May 2014, pp. 1–16 (cit. on p. 5).

[Bar+20] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. "An Algebraic Attack on Rank Metric Code-Based Cryptosystems". In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 64–93 (cit. on p. 22).

[Bar+20] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. "Improvements of Algebraic Attacks for solving the Rank Decoding and MinRank problems". In: *Advances in Cryptology - ASIACRYPT 2020, International Conference on the Theory and Application of Cryptology and Information Security, 2020. Proceedings*. 2020, pp. 507–536 (cit. on p. 22).

[Bar+22] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, and Jean-Pierre Tillich. *Revisiting Algebraic Attacks on MinRank and on the Rank Decoding Problem*. 2022 (cit. on p. 22).

[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*. 2011, pp. 364–385 (cit. on pp. 86, 96).

[BCN89] Andries E. Brouwer, Arjeh M. Cohen, and Arnold Neumaier. *Distance-Regular Graphs*. Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge / A Series of Modern Surveys in Mathematics 18. Springer Verlag Berlin Heidelberg, 1989. ISBN: 978-3-642-74341-2 (cit. on p. 55).

[Bec+12] 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 (cit. on pp. 18, 19, 23).

[Bel+19] Emanuele Bellini, Florian Caullery, Philippe Gaborit, Marc Manzano, and Víctor Mateu. "Improved Veron Identification and Signature Schemes in the Rank Metric". In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2019*.

Vol. abs/1903.10212. Paris, France: IEEE, July 2019, pp. 1872–1876 (cit. on p. 29).

[Bel+20] Emanuele Bellini, Philippe Gaborit, Alexandros Hasikos, and Víctor Mateu. "Enhancing Code Based Zero-Knowledge Proofs Using Rank Metric". In: *Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings*. Ed. by Stephan Krenn, Haya Shulman, and Serge Vaudenay. Vol. 12579. Lecture Notes in Computer Science. Springer, 2020, pp. 570–592 (cit. on p. 29).

[Ber05] Daniel J. Bernstein. "The Poly1305-AES Message-Authentication Code". In: *Fast Software Encryption*. Ed. by Henri Gilbert and Helena Handschuh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 32–49 (cit. on p. 1).

[Ber+13] Daniel J. Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. "Quantum Algorithms for the Subset-Sum Problem". In: *Post-Quantum Cryptography 2011*. Vol. 7932. LNCS. Limoges, France, June 2013, pp. 16–33 (cit. on p. 97).

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. "Quantum Cryptanalysis of Hash and Claw-Free Functions". In: *LATIN*. Vol. 1380. Lecture Notes in Computer Science. Springer, 1998, pp. 163–169 (cit. on p. 85).

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. "CSI-FiSh: Efficient Isogeny Based Signatures Through Class Group Computations". In: *ASIACRYPT (1)*. Vol. 11921. Lecture Notes in Computer Science. Springer, 2019, pp. 227–247 (cit. on p. 62).

[BLP11] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. "Smaller decoding exponents: ball-collision decoding". In: *Advances in Cryptology - CRYPTO 2011*. Vol. 6841. LNCS. 2011, pp. 743–760 (cit. on p. 18).

[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*. Sept. 2017 (cit. on p. 18).

[BM18] Leif Both and Alexander May. "Decoding Linear Codes with High Error Rate and Its Impact for LPN Security". In: *Post-Quantum Cryptography 2018*. Ed. by Tanja Lange and Rainer Steinwandt. Vol. 10786. LNCS. Fort Lauderdale, FL, USA: Springer, Apr. 2018, pp. 25–46 (cit. on pp. 19, 23).

[BMT78] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. "On the inherent intractability of certain coding problems". In: *IEEE Trans. Inform. Theory* 24.3 (May 1978), pp. 384–386 (cit. on p. 19).

[BN18] Xavier Bonnetain and María Naya-Plasencia. "Hidden Shift Quantum Cryptanalysis and Implications". In: *Advances in Cryptology – ASIACRYPT 2018*. Ed. by Thomas Peyrin and Steven Galbraith. Cham: Springer International Publishing, 2018, pp. 560–592 (cit. on pp. viii, xii, 1, 70, 104, 105).

[Bon+19] Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. "Quantum Attacks Without Superposition Queries: The Offline Simon's Algorithm". In: *Advances in Cryptology – ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Cham: Springer International Publishing, 2019, pp. 552–583. ISBN: 978-3-030-34578-5 (cit. on p. 3).

[Bon19a] Xavier Bonnetain. "Hidden Structures and Quantum Cryptanalysis". Theses. Sorbonne Université, Nov. 2019 (cit. on p. 77).

[Bon19b] Xavier Bonnetain. *Improved low-qubit hidden shift algorithms*. arXiv:1901.11428. 2019 (cit. on pp. 77, 87).

[Bon+20] Xavier Bonnetain, Rémi Bricout, André Schrottenloher, and Yixin Shen. "Improved Classical and Quantum Algorithms for Subset-Sum". In: *Advances in Cryptology – ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Cham: Springer International Publishing, 2020, pp. 633–666 (cit. on pp. 83, 85, 86, 96–99).

[Bra+19] Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. "Worst-Case Hardness for LPN and Cryptographic Hashing via Code Smoothing". In: *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. LNCS. Springer, 2019, pp. 619–635 (cit. on pp. 19, 23).

[Bro22] Maxime Bros. "Algebraic cryptanalysis and contributions to post-quantum cryptography based on error-correcting codes in the rank-metric". PhD thesis. Université de Limoges, Dec. 2022 (cit. on p. 12).

[BS20] Xavier Bonnetain and André Schrottenloher. "Quantum Security Analysis of CSIDH". In: *EUROCRYPT (2)*. Vol. 12106. Lecture Notes in Computer Science. Springer, 2020, pp. 493–522 (cit. on pp. 62, 73, 77, 81, 87).

[Car+23] Kévin 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: 28th International Conference on the Theory and Application of Cryptology and Information Security,*

Taipei, Taiwan, December 5–9, 2022, Proceedings, Part IV. Taipei, Taiwan: Springer-Verlag, 2023, 477–507. ISBN: 978-3-031-22971-8 (cit. on pp. 19, 23).

[Cas+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. "CSIDH: An Efficient Post-Quantum Commutative Group Action". In: *Advances in Cryptology – ASIACRYPT 2018*. Ed. by Thomas Peyrin and Steven Galbraith. Cham: Springer International Publishing, 2018, pp. 395–427 (cit. on pp. 1, 62).

[CE03] Henry Cohn and Noam Elkies. "New Upper Bounds on Sphere Packings I". In: *Ann. of Math. (2)* 157.2 (2003), pp. 689–714. ISSN: 0003486X (cit. on pp. vii, xi).

[Chá+22] Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. "The SQALE of CSIDH: sublinear Vélu quantum-resistant isogeny action with low exponents". In: *J. Cryptogr. Eng.* 12.3 (2022), pp. 349–368 (cit. on pp. 62, 81).

[Chi22] Andrew Childs. *Lecture notes on quantum algorithms - Kuperberg's algorithm for the dihedral HSP*. 2022 (cit. on pp. 61, 72).

[CJS14] Andrew Childs, David Jao, and Vladimir Soukharev. "Constructing elliptic curve isogenies in quantum subexponential time". In: *Journal of Mathematical Cryptology* 8.1 (2014), pp. 1–29 (cit. on pp. 1, 74, 77, 87).

[CLS22] Federico Canale, Gregor Leander, and Lukas Stennes. "Simon's Algorithm and Symmetric Crypto: Generalizations and Automatized Applications". In: *Advances in Cryptology – CRYPTO 2022*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Cham: Springer Nature Switzerland, 2022, pp. 779–808. ISBN: 978-3-031-15982-4 (cit. on p. 3).

[Con23] Keith Conrad. *Expository papers, Dihedral Groups I and II*. Retrieved January 3, 2023, from https://kconrad.math.uconn.edu/blurbs/. 2023 (cit. on pp. 61, 63).

[Cou20] Alain Couvreur. *Introduction to coding theory*. Retrieved January 3, 2023, from http://www.lix.polytechnique.fr/ alain.couvreur/teaching.html. 2020 (cit. on pp. 12, 18).

[CS96] Florent Chabaud and Jacques Stern. "The Cryptographic Security of the Syndrome Decoding Problem for Rank Distance Codes". In: *Advances in Cryptology - ASIACRYPT 1996*. Vol. 1163. LNCS. Kyongju, Korea: Springer, Nov. 1996, pp. 368–381 (cit. on p. 22).

[Deb19] Thomas Debris-Alazard. "Cryptographie fondée sur les codes : nouvelles approches pour constructions et preuves ; contribution en cryptanalyse". Theses. Sorbonne Université, Dec. 2019 (cit. on p. 12).

[Deb21a] Thomas Debris-Alazard. *Code-based cryptography - An intractable problem related to codes, decoding*. Retrieved February 2, 2023, from https://tdalazard.io/. 2021 (cit. on p. 12).

[Deb21b] Thomas Debris-Alazard. *Code-based cryptography - Random codes*. Retrieved February 2, 2023, from https://tdalazard.io/. 2021 (cit. on p. 12).

[Del78] Philippe Delsarte. "Bilinear Forms over a Finite Field, with Applications to Coding Theory". In: *J. Comb. Theory, Ser. A* 25.3 (1978), pp. 226–241 (cit. on p. 17).

[DG19] Luca De Feo and Steven D. Galbraith. "SeaSign: Compact Isogeny Signatures from Class Group Actions". In: *EUROCRYPT (3)*. Vol. 11478. Lecture Notes in Computer Science. Springer, 2019, pp. 759–789 (cit. on p. 62).

[DH76] Whitfield Diffie and Martin Hellman. "New directions in cryptography". In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654 (cit. on pp. 1, 62).

[DPV19] Thomas Decru, Lorenz Panny, and Frederik Vercauteren. "Faster SeaSign Signatures Through Improved Rejection Sampling". In: *PQCrypto*. Vol. 11505. Lecture Notes in Computer Science. Springer, 2019, pp. 271–285 (cit. on p. 62).

[DR22] Thomas Debris-Alazard and Nicolas Resch. *Worst and Average case hardness of Decoding via Smoothing Bounds*. preprint. eprint. Dec. 2022 (cit. on pp. 19, 23).

[DRT23] Thomas Debris-Alazard, Maxime Remaud, and Jean-Pierre Tillich. "Quantum Reduction of Finding Short Code Vectors to the Decoding Problem". In: *IEEE transactions on Information Theory* (2023) (cit. on pp. vi, x, 23).

[Dum89] Ilya Dumer. "Two decoding algorithms for linear codes". In: *Probl. Inf. Transm.* 25.1 (1989), pp. 17–23 (cit. on pp. 19, 23).

[EH99] Mark Ettinger and Peter Høyer. "On Quantum Algorithms for Noncommutative Hidden Subgroups". In: *Lecture Notes in Computer Science* (1999), 478–487 (cit. on pp. viii, xi, 8, 64–66).

[Ess+21] Andre Esser, Sergi Ramos-Calderer, Emanuele Bellini, José I. Latorre, and Marc Manzano. "An Optimized Quantum Implementation of ISD on Scalable Quantum Resources". In: *IACR Cryptol. ePrint Arch.* (2021), p. 1608 (cit. on p. 100).

[FLP08] Jean-Charles Faugère, Françoise Levy-dit-Vehel, and Ludovic Perret. "Cryptanalysis of Minrank". In: *Advances in Cryptology - CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. LNCS. 2008, pp. 280–296 (cit. on pp. 19, 21).

[FS09] Matthieu Finiasz and Nicolas Sendrier. "Security Bounds for the Design of Code-based Cryptosystems". In: *Advances in Cryptology - ASIACRYPT 2009*. Ed. by M. Matsui. Vol. 5912. LNCS. Springer, 2009, pp. 88–105 (cit. on p. 18).

[FS96] Jean-Bernard Fischer and Jacques Stern. "An efficient pseudo-random generator provably as secure as syndrome decoding". In: *Advances in Cryptology - EUROCRYPT'96*. Ed. by Ueli Maurer. Vol. 1070. LNCS. Springer, 1996, pp. 245–255. ISBN: ISBN 978-3-540-61186-8 (cit. on p. 19).

[Gib96] Keith Gibson. "The Security of the Gabidulin Public Key Cryptosystem". In: *Advances in Cryptology - EUROCRYPT '96*. Ed. by Ueli Maurer. Vol. 1070. LNCS. Springer, 1996, pp. 212–223. ISBN: 978-3-540-61186-8 (cit. on p. 21).

[GPT91] Ernst M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. "Ideals over a non-commutative ring and their applications to cryptography". In: *Advances in Cryptology - EUROCRYPT'91*. LNCS 547. Brighton, Apr. 1991, pp. 482–489 (cit. on p. 21).

[GRS16] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. "On the Complexity of the Rank Syndrome Decoding Problem". In: *IEEE Trans. Inform. Theory* 62.2 (2016), pp. 1006–1019 (cit. on p. 22).

[GZ16] Philippe Gaborit and Gilles Zémor. "On the hardness of the decoding and the minimum distance problems for rank codes". In: *IEEE Trans. Inform. Theory* 62(12) (2016), pp. 7245–7252 (cit. on p. 21).

[Ham50] R. W. Hamming. "Error detecting and error correcting codes". In: *The Bell System Technical Journal* 29.2 (1950), pp. 147–160 (cit. on p. 17).

[Hau17] Adrien Hauteville. "Nouveaux protocoles et nouvelles attaques pour la cryptologie basée sur les codes en métrique rang. (New protocols and new attacks on rank metric code-based cryptography)". PhD thesis. University of Limoges, France, 2017 (cit. on p. 12).

[HH00] L. Hales and S. Hallgren. "An improved quantum Fourier transform algorithm and applications". In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. 2000, pp. 515–525 (cit. on p. 6).

[HJ10] Nicholas Howgrave-Graham and Antoine Joux. "New generic algorithms for hard knapsacks". In: *Advances in Cryptology - EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Sringer, 2010 (cit. on pp. 86, 96).

[HM18] Alexander Helm and Alexander May. "Subset Sum Quantumly in 1.17^n". In: *13th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2018)*. Ed. by Stacey Jeffery. Vol. 111. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 5:1–5:15 (cit. on p. 97).

[HM20] Alexander Helm and Alexander May. "The Power of Few Qubits and Collisions - Subset Sum Below Grover's Bound". In: *PQCrypto*. Vol. 12100. Lecture Notes in Computer Science. Springer, 2020, pp. 445–460 (cit. on pp. 83, 85, 101).

[HP03] W. Cary Huffman and Vera Pless. *Fundamentals of error-correcting codes*. Cambridge University Press, Cambridge, 2003, pp. xviii+646. ISBN: 0-521-78280-5 (cit. on p. 12).

[Jab01] Abdulrahman Al Jabri. "A statistical decoding algorithm for general linear block codes". In: *Cryptography and coding. Proceedings of the 8th IMA International Conference*. Ed. by Bahram Honary. Vol. 2260. LNCS. Cirencester, UK: Springer, Dec. 2001, pp. 1–8 (cit. on p. 18).

[Kit95] Alexei Y. Kitaev. "Quantum measurements and the Abelian Stabilizer Problem". In: *Electron. Colloquium Comput. Complex.* TR96 (1995) (cit. on p. 8).

[KLM06] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. Oxford University Press, Nov. 2006 (cit. on pp. 2, 3).

[Kup05] Greg Kuperberg. *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem*. arXiv:quant-ph/0302112. 2005 (cit. on pp. vii, xi, 67, 72, 73, 85).

[Kup13] Greg Kuperberg. *Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem*. arXiv:1112.3333. 2013 (cit. on pp. vii, xi, 78, 85).

[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*. Vol. 330. LNCS. Springer, 1988, pp. 275–280 (cit. on p. 18).

[Lee58] William C. Y. Lee. "Some properties of nonbinary error-correcting codes". In: *IRE Transactions on Information Theory* 4.2 (1958), pp. 77–82 (cit. on p. 17).

[Leo88] Jeffrey Leon. "A probabilistic algorithm for computing minimum weights of large error-correcting codes". In: *IEEE Trans. Inform. Theory* 34.5 (1988), pp. 1354–1359 (cit. on p. 18).

[Leq21] Matthieu Lequesne. "Analysis of code-based post-quantum cryptosystems". PhD thesis. Sorbonne Université, May 2021 (cit. on p. 12).

[Lev79] Vladimir Iossifovitch Levenshtein. "Bounds for packings in $n-$dimensional Euclidean space". In: *Soviet Math. Dokl.* 20 (1979), pp. 417–421 (cit. on pp. vii, xi).

[LM09] Vadim Lyubashevsky and Daniele Micciancio. "On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem". In: *Proceedings of CRYPTO 2009*. Vol. 5677. Lecture Notes in Computer Science. IACR. Springer, 2009, pp. 577–594 (cit. on p. 62).

[Loi06] Pierre Loidreau. *Properties of codes in rank metric*. 2006 (cit. on p. 58).

[Loi07] Pierre Loidreau. *Metrique rang et cryptographie*. Hal theses, HDR Thesis tel-00200407. 2007 (cit. on p. 12).

[LY22] Hanlin Liu and Yu Yu. "A Non-heuristic Approach to Time-Space Tradeoffs and Optimizations for BKW". In: *Advances in Cryptology – ASIACRYPT 2022*. Ed. by Shweta Agrawal and Dongdai Lin. Cham: Springer Nature Switzerland, 2022, pp. 741–770. ISBN: 978-3-031-22969-5 (cit. on p. 69).

[McE+77] R. J. McEliece, E. R. Rodemich, H. Rumsey, and L. R. Welch. "New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities". In: *IEEE Trans. Inform. Theory* 23.2 (1977), pp. 157–166 (cit. on pp. vii, xi).

[McE78] Robert J. McEliece. "A Public-Key System Based on Algebraic Coding Theory". In: DSN Progress Report 44. Jet Propulsion Lab, 1978, pp. 114–116 (cit. on p. 19).

[Mil14] Sean Miller. *The Number of Subgroups of the Dihedral Group D(n)*. Retrieved January 3, 2023, from https://www.greeleyschools.org/Page/15355. 2014 (cit. on pp. 63, 64).

[Mis+12] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. *MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes*. 2012 (cit. on p. 19).

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. "Decoding random linear codes in $O(2^{0.054n})$". In: *Advances in Cryptology - ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, 2011, pp. 107–124 (cit. on pp. 18, 19, 23).

[MO15] Alexander May and Ilya Ozerov. "On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes". In: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 203–228 (cit. on pp. 18, 19, 23).

[Mor23] Rocco Mora. "Algebraic techniques for decoding Reed-Solomon codes and cryptanalyzing McEliece-like cryptosystems". Theses. Sorbonne Université, Apr. 2023 (cit. on p. 12).

[NC16] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016. ISBN: 978-1-10-700217-3 (cit. on p. 34).

[NIS16] NIST. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016 (cit. on pp. vi, x, 62).

[NIS22] NIST. *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. 2022 (cit. on pp. vi, x).

[OJ02] Alexei V. Ourivski and Thomas Johansson. "New Technique for Decoding Codes in the Rank Metric and Its Cryptography Applications". English. In: *Problems of Information Transmission* 38.3 (2002), pp. 237–246. ISSN: 0032-9460 (cit. on p. 22).

[OS09] Raphael Overbeck and Nicolas Sendrier. "Code-based cryptography". In: *Post-quantum cryptography*. Ed. by Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. Springer, 2009, pp. 95–145. ISBN: 978-3-540-88701-0 (cit. on p. 12).

[Ove05] Raphael Overbeck. "A New Structural Attack for GPT and Variants". In: *Mycrypt*. Vol. 3715. LNCS. 2005, pp. 50–63 (cit. on p. 21).

[Pei20] Chris Peikert. "He Gives C-Sieves on the CSIDH". In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 463–492 (cit. on pp. 62, 81, 85).

[PH78] S. Pohlig and M. Hellman. "An improved algorithm for computing logarithms overGF(p)and its cryptographic significance (Corresp.)" In: *IEEE Transactions on Information Theory* 24.1 (1978), pp. 106–110 (cit. on p. 5).

[Pol78] J. M. Pollard. "Monte Carlo Methods for Index Computation (mod p)". In: *Mathematics of Computation* 32.143 (1978), pp. 918–924 (cit. on p. 5).

[Pra62] Eugene Prange. "The use of information sets in decoding cyclic codes". In: *IRE Transactions on Information Theory* 8.5 (1962), pp. 5–9 (cit. on pp. 18, 19, 23).

[Reg02] Oded Regev. "Quantum computation and lattice problems". In: (2002). arXiv:cs/0304005 (cit. on pp. 62, 83, 89).

[Reg04a] Oded Regev. *A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space*. arXiv:quant-ph/0406151. 2004 (cit. on pp. vii, viii, xi, 73, 74, 77, 87).

[Reg04b] Oded Regev. "New Lattice-Based Cryptographic Constructions". In: *J. ACM* 51.6 (2004), 899–942. ISSN: 0004-5411 (cit. on pp. 1, 62).

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. 2005, pp. 84–93 (cit. on pp. vi, x, 23, 146).

[Reg09] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". Extended version of [Reg05], dated May 2009. 2009 (cit. on pp. 19, 23, 25).

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (1978), pp. 120–126 (cit. on p. 1).

[RST23] Maxime Remaud, André Schrottenloher, and Jean-Pierre Tillich. "Time and Query Complexity Tradeoff for the Dihedral Coset Problem". In: *PQCrypto 2023*. LNCS. Springer, 2023 (cit. on pp. viii, xii, 83).

[RT23] Maxime Remaud and Jean-Pierre Tillich. "Linear Space Algorithm for the Dihedral Coset Problem". In: (2023). Preprint (cit. on p. 103).

[Sch21] Travis Schedler. *Group representation theory, Lecture Notes*. 2021 (cit. on p. 9).

[Sha73] D. Shanks. "Five number-theoretic algorithms". In: *Proceedings of the Second Manitoba Conference on Numerical Mathematics, Congressus Numerantium* VII (1973), pp. 51–70 (cit. on p. 5).

[Sho94] Peter W. Shor. "Algorithms for quantum computation: Discrete logarithms and factoring". In: *FOCS*. Ed. by S. Goldwasser. 1994, pp. 124–134 (cit. on pp. 5, 24).

[Sim94] Daniel R. Simon. "On the Power of Quantum Computation". In: *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*. IEEE Computer Society, 1994, pp. 116–123 (cit. on pp. 3, 24).

[Ste+09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. "Efficient Public Key Encryption Based on Ideal Lattices". In: *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, 2009, pp. 617–635 (cit. on pp. vi, x, 33, 49, 62, 83, 89, 134).

[Ste88] Jacques Stern. "A method for finding codewords of small weight". In: *Coding Theory and Applications*. Ed. by G. D. Cohen and J. Wolfmann. Vol. 388. LNCS. Springer, 1988, pp. 106–113 (cit. on pp. 18, 19, 23).

[Ste93] Jacques Stern. "A New Identification Scheme Based on Syndrome Decoding". In: *Advances in Cryptology - CRYPTO'93*. Ed. by D.R. Stinson. Vol. 773. LNCS. Springer, 1993, pp. 13–21 (cit. on pp. 19, 29).

[The22] TheIACR. *A Non-heuristic Approach to Time-space Tradeoffs and Optimizations for BKW*. 2022 (cit. on p. 69).

[Var97] Alexander Vardy. "The Intractability of Computing the Minimum Distance of a Code". In: *IEEE Trans. Inform. Theory* 43.6 (Nov. 1997), pp. 1757–1766 (cit. on p. 19).

[VBE96] Vlatko Vedral, Adriano Barenco, and Artur Ekert. "Quantum networks for elementary arithmetic operations". In: *Phys. Rev. A* 54 (1 1996), pp. 147–153 (cit. on p. 6).

[vW99] Paul C. van Oorschot and Michael J. Wiener. "Parallel Collision Search with Cryptanalytic Applications". In: *Journal of Cryptology* 12 (1999), pp. 1–28 (cit. on p. 5).

[Wag02] David Wagner. "A generalized birthday problem". In: *Advances in Cryptology - CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. LNCS. Springer, 2002, pp. 288–303. ISBN: 978-3-540-44050-5 (cit. on p. 78).

[Wol19] Ronald de Wolf. *Quantum Computing: Lecture Notes*. 2019 (cit. on p. 9).

[Yu+19] Yu Yu, Jiang Zhang, Jian Weng, Chun Guo, and Xiangxue Li. "Collision Resistant Hashing from Sub-exponential Learning Parity with Noise". In: *ASIACRYPT (2)*. Vol. 11922. Lecture Notes in Computer Science. Springer, 2019, pp. 3–24 (cit. on pp. 19, 23).