Classical and quantum cryptanalysis of lattices and codes

Lynn Engelberts

Classical and quantum cryptanalysis of lattices and codes

ILLC Dissertation Series DS-2026-08

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31 (0)20 525 6051
e-mail: illc@uva.nl
homepage: https://www.illc.uva.nl/

The research for and publication of this doctoral thesis received financial assistance from the Dutch National Growth Fund (NGF), as part of the QDNL program.

Printed and bound by Proefschriftspecialist

ISBN: 978-94-93539-34-1

Classical and Quantum Cryptanalysis of Lattices and Codes

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek
ten overstaan van een door het College voor Promoties ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit
op woensdag 17 juni 2026, te 11.00 uur

door Lynn Engelberts
geboren te Haarlem

Promotiecommissie

Promotores:	prof. dr. R.M. de Wolf	Universiteit van Amsterdam
	prof. dr. L. Ducas	Universiteit Leiden
Overige leden:	prof. dr. S.M. Jeffery	Universiteit van Amsterdam
	prof. dr. C. Schaffner	Universiteit van Amsterdam
	dr. N.A. Resch	Universiteit van Amsterdam
	prof. dr. P. Nguyen	École Normale Supérieure, Paris
	dr. D. Stehlé	Cryptolab

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

voor Alec en Sam

Contents

Part II: Quantum cryptanalysis

List of publications

This dissertation is based on the following papers, listed in chronological order. Contributions were shared equally among the authors of the first three papers. For the fourth paper, the author of this dissertation was the main contributor.

[EEL25]: *Quantum Sieving for Code-Based Cryptanalysis and its Limitations for ISD.*

Lynn Engelberts, Simona Etinski, Johanna Loyer.

Designs, Codes and Cryptography, 93:1611–1644, 2025. Presented as a contributed talk at CFAIL 2024 (affiliated workshop of CRYPTO 2024).

(Chapter 7 is based on this paper.)

[DEL25]: *Wagner's Algorithm Provably Runs in Subexponential Time for SIS$^\infty$.*

Léo Ducas, Lynn Engelberts, Johanna Loyer.

Proceedings of CRYPTO 2025.

(Chapter 3 is based on this paper.)

[DEP25]: *Predicting Module-Lattice Reduction.*

Léo Ducas, Lynn Engelberts, Paola de Perthuis.

Proceedings of ASIACRYPT 2025.

(Chapter 4 is based on this paper.)

[ECGH+25]: *An Improved Quantum Algorithm for 3-Tuple Lattice Sieving.*

Lynn Engelberts, Yanlin Chen, Amin S. Gilani, Maya-Iggy van Hoof, Stacey Jeffery, Ronald de Wolf.

Proceedings of CRYPTO 2026 (to appear). Presented as a contributed talk at QIP 2026.

(Chapter 6 is based on this paper.)

Acknowledgements

I am incredibly grateful to the many people who supported me throughout my PhD and who made these years so meaningful.

The biggest thanks go to my promotors Ronald de Wolf and Léo Ducas. I am very grateful to have had both of you as supervisors: your guidance, knowledge, and insights have played a major role in my development as a researcher. Doing research with both of you has been such a valuable and enjoyable experience, and I truly hope we will continue collaborating in the future. *Ronald*: thank you for trusting me as your PhD student and for guiding me through the transition from student to researcher: my progress benefited greatly from your knowledge and way of thinking. I am grateful you always made time when needed and, above all, that you continued to believe in me even when progress was not immediately visible. *Léo*: there was never a moment I regretted asking you to become my second promotor. Thank you for introducing me to the fascinating world of lattice-based cryptography and to the beautiful mathematics surrounding it. I continue to be impressed by your geometric intuition, which inspired several results in this thesis.

Another huge thank you to my co-authors and collaborators, including on projects that did not lead to a paper: Amin, Arjan, Johanna, Léo, Maya, Mehrdad, Paola, Rachel, Romy, Ronald, Ryan, Sam, Simon, Simona, Stacey, Yanlin, and Yixin. An extra thanks to Romy for inviting me to Bristol twice, which facilitated some of these collaborations and sparked new friendships along the way. I also greatly enjoyed working together with Sarah Arpin to set up a post-quantum cryptography reading group between CWI and Leiden, and am grateful to all who participated. All these collaborations have made my research more productive, rewarding, and above all more enjoyable!

For bringing this thesis to its final form, I am very grateful to my supervisors and Chris for their thorough feedback on various parts of this thesis, and to Johanna and Simona for their help in creating some of the figures and for their feedback on Chapter 7. A warm thank you to Eline, Judith, and my mother for helping improve

the readability and accessibility of the Dutch summary, and to Jelle and his team from Proefschriftspecialist for their efforts in printing this thesis.

I thank the members of my doctorate committee for taking the time to read and evaluate my thesis: Stacey Jeffery, Christian Schaffner, Nicolas Resch, Phong Nguyen, and Damien Stehlé. I look forward to discussing my research with you.

I am also grateful to QDNL, CWI, QuSoft, the ILLC, and the UvA for supporting my PhD research through funding and other ways.

I could not have wished for a better environment in which to do my PhD. I therefore want to thank my friends and colleagues at CWI and QuSoft, as well as those from other institutions I met at conferences and other academic events.

Special thanks go to *Simona* and *Yanlin*: throughout my PhD you both felt like my older academic siblings that I could rely on for literally anything. Thank you so much for looking after me at CWI and conferences, for your commitment during collaborations, and for always making me laugh.

It was a real pleasure to do my PhD surrounded by the researchers of my group: thank you for everything you taught me about research and academia, for the restorative breaks from work, and for the fun times at conferences. You are all genuinely good company, and I feel very lucky to call several of you my friends. A few people I want to explicitly thank. My office mates *Dyon*, *Jordi*, *Marten*, *Niels*, and *Sarah*: you all made the office a place I enjoyed coming to. *Amira*: your laugh is contagious, and I am sure one day we will win a foosball tournament. *Arjan*: our India adventures still make me laugh. *Fran*: thanks for your active role in the group and for thinking of the well-being of others. *Freek*: thanks for generously sharing your wisdom with me. *Harold*: we made a great team, fueled by kruidnoten and mandarins. *Galina*: your coffee and museum recommendations never disappoint. *Ido* and the rest of the Fermioniq team: thanks for the many good chats and lunches. *Jana*: thank you for inviting me to your reading group; in some sense, this kick-started my PhD. *Llorenç*: you made sure there are many photos to remember the past years. *Nikhil*: foosball hasn't been the same since you left. *Sebas*: I am truly grateful that you and Chelsea are inseparable. *Subha*: you are definitely one of the strongest women I know.

An extra thank you to Fran, Harold, and Maris for the rewarding experience co-organizing various seminars and other sessions, and to Ailsa, Garazi, Gina, Mani, and Sarah for taking over these roles. Thanks also to Florian and Yvonne for the great time brainstorming about the scientific program of QuSoft's 10-year-anniversary conference. I am grateful to Aldo, Carla, Christian, Doutzen, Harry, Jop, Kareljan, Koen, Ronald, Susanne, Tony, and the support staff for making the

A&C group and QuSoft such a successful and pleasant place to work. I also want to thank Doutzen for making me feel welcome from day one, and Carla for ensuring there are always plant-based options when meals are provided.

My warmest thanks to those from my group I did not yet mention: Adam, Ake, Akshay, Alicja, Anna, Artemis, Carli, Daan P, Daan S, Davi, Dmitry, Emiel, Filippo, Hemant, Jana, Jelena, Jeroen, Jiri, John, Jonas, Joppe, Joran, Krystal, Léo, Lisa, Liam, Lorenzo, Ludovico, Marc, Maxim, Mehrdad, Michael, Peter, Philip, Philippe, Poojith, Priscilla, Quinten, Randy, Rene, Salvatore, Sebastian V, Seenivasan, Stacey, Yaroslav, Zongbo.

I also spent a lot of time with my wonderful colleagues from the Cryptology group: thank you all for always welcoming me to seminars and for letting me join your lunch breaks; I am excited to be joining your group soon! I particularly want to thank my academic half-brothers *Ludo*, *Shane*, and *Wessel*: it is great to share so many (research) interests and to have spent so much time together at CWI, conferences, and elsewhere. *Eamonn*: thank you for the research chats when you were still at CWI, and for welcoming me at your research group in London. *Johanna*: I think you are one of the people that my research interests overlap with the most; it has been a joy collaborating on two projects with you.

The above hopefully already demonstrates that CWI is not only a stimulating place to do research, but that working there is also a lot of fun. This is partly because of the various social activities, including the foosball breaks, the Activity Committee events, the lunches organized for colleagues learning Dutch, and the running events we participated in together. These activities introduced me to great people from other research groups, such as Aditya, Hema, Max, Rik, and Vlad. I also want to thank Erik, Paul, and Rob for the many cheerful interactions in the hallways and canteen. Thanks to everyone at CWI for the great time!

As much as I loved my work, these years were made complete by spending time with people outside of it: my partner, our families, and our friends. Thank you all for making me feel appreciated, supporting my career choices, reminding me that life is full of great things beyond research, and for inspiring me through being the wonderful people you are.

To the dear friends I made in the Netherlands, UK, and Australia: I appreciate you and our friendships very much, and cannot wait to make more memories together! I would like to explicitly thank some of you. *Selma*: dankjewel dat je me al ruim vijftien jaar steunt in álles wat ik doe en voor de allerbeste lachbuien zorgt (tegenwoordig samen met *Amir* en *Samer*). *Anna*: dankjewel dat ik elk succes met jou kan vieren en dat je samen met Chris mijn paranimf wilt zijn; dat

we tegelijkertijd ons promotieonderzoek aan de UvA deden heeft de afgelopen jaren extra fijn en bijzonder gemaakt. *Eva*: je weet altijd wat er speelt in mijn leven en wanneer ik een hart onder de riem nodig heb; muchas gracias voor onze vriendschap en alle geweldige herinneringen (ook samen met *Bart*). *Sophie vBdK*: dankjewel voor je aanstekelijke energie en ambitie; met jou staan er altijd leuke én ontspannende activiteiten gepland. *David* en *Eireamhan*: ik ben blij dat jullie via Anna en Sophie in mijn leven zijn gekomen en waardeer onze diepe gesprekken en avonturen; bedankt dat ik volledig mezelf kan zijn bij jullie. *Eline*, *Judith* en *Laura*: dankzij jullie blijf ik mijn wereldbeeld telkens verruimen en is er altijd een concert om naar uit te kijken. *Jay*: thank you for showing me the beauty of mathematics and for the adventures in Las Vegas and the surrounding national parks.

To my own and my partner's family: thank you all so much for your support over the past years! Ik heb het ontzettend getroffen met zo'n hechte, warme én gezellige familie. Mijn enthousiasme voor de wetenschap heb ik grotendeels aan mijn grootouders *Corry* en *Gert* te danken. Dankzij jullie heb ik van jongs af aan een open en nieuwsgierige kijk op de medemens en wereld ontwikkeld. Oma: van u heb ik geleerd dat ook als vrouw alles mogelijk is. Opa: het is een enorme eer om mijn proefschrift te mogen verdedigen op precies dezelfde plek als u dat in 1971 deed. Mijn ouders *Lidy* en *Lambert*: bedankt voor jullie onvoorwaardelijke steun en liefde. Jullie hebben mij nooit een richting opgelegd, maar juist de deuren opengezet zodat ik zelf mijn eigen weg kon vinden. Mam: je inspireert me iedere dag in hoe liefdevol, positief en krachtig je in het leven staat. Pap: jouw discipline en loyaliteit zijn goud waard en je motiveert me als geen ander om mijn grenzen te blijven verleggen. Mijn zus *Robin*: ik ben dol op jouw unieke, nieuwsgierige en zorgzame karakter en het maakt me trots om jou te mogen zien uitgroeien tot de prachtige persoon die je bent. Dankjewel dat je altijd in mij gelooft en zo geïnteresseerd bent in alles wat ik doe.

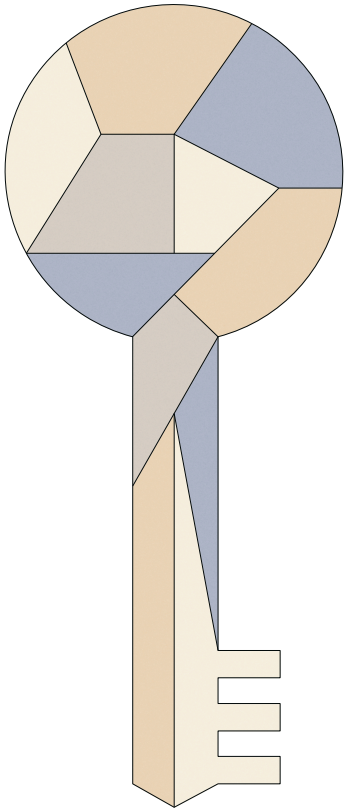
Last but not least, *Chris*: with you, every day is more fun than I could ever have wished for. You bring the humor, structure, and stability I need, and you show me how to be more dedicated to other people, animals, and the world around us. Thank you for your endless support and for inspiring me to become a better and happier person.

Dit proefschrift draag ik op aan mijn neven Sam en Alec. Jullie herinneren me er dagelijks aan om uitdagingen als kansen te zien, door te zetten wanneer het moeilijk wordt en in mijn ambities te blijven geloven. Sam en Alec, het is gelukt!

Lynn Engelberts *Amsterdam, 2026*

Chapter 1

General introduction



Contents of this chapter

1.1 Cryptography and the role of cryptanalysis

Cryptography studies techniques for protecting communication and information from untrusted parties. Despite tracing its roots back to ancient civilizations such as Ancient Egypt, cryptography is arguably more indispensable than ever in a modern digital society, where information is ubiquitous and privacy is highly valued. Indeed, cryptography plays a central role in securing internet connections (HTTPS, TLS), digital communication (Signal, protected e-mail services), financial transactions (credit cards, online banking), cloud storage (Google Drive, iCloud), and the list goes on. Privacy itself is also recognized as a universal human right [Uni48].

A fundamental cryptographic task is to hide a message sent over a public channel in a way that allows only the intended recipient to recover it, known as *encryption*. One of the earliest encryption methods used is the class of substitution ciphers, where each symbol in an alphabet is systematically replaced by other symbols. For instance, the *Caesar cipher* shifts each letter in an alphabet by k positions to the right, wrapping the end of the alphabet around to the beginning, and where k is kept secret. For example, when the alphabet is $\Sigma = \{a, b, \ldots, z\}$ of size 26 and $k = 3$, the word "zoo" becomes "crr". This method is named after the Roman general Julius Caesar, who reportedly used it with $k = 3$ [Kah96]. More complex substitution ciphers can be found in rotor machines, such as the Enigma machine used by the Nazi regime during the Second World War. Nowadays, these ciphers have generally been substituted with more secure digital methods.

When designing cryptographic systems, one can consider several notions of security, four of which are particularly important:

- Confidentiality: the information is kept secret from unauthorized parties
- Integrity: the information is not altered by an unauthorized party
- Authentication: the sender's identity is verified
- Non-repudiation: a party cannot later deny having performed a specific action

These notions are realized via different cryptographic primitives; for instance, the primitive of encryption aims to ensure the confidentiality of information. It is often difficult to prove strong security guarantees for practical cryptosystems, so confidence in their security typically results from their resistance to extensive attack efforts. The study of techniques for compromising the security of cryptographic systems is known as *cryptanalysis*. Cryptography and cryptanalysis are in continual interplay, and together they form the field of *cryptology*.

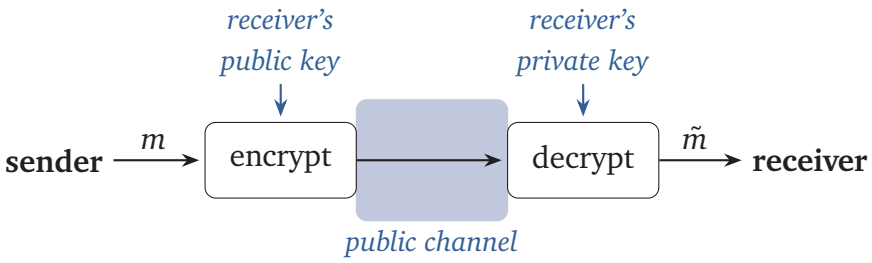
To achieve a certain security notion such as confidentiality, there should be some element of the cryptosystem that is not publicly known: otherwise, a malicious party would know how to decrypt any encrypted message just as well as the intended recipient. While one may be tempted to achieve this by keeping the overall cryptosystem secret, this is generally not recommended. For instance, deploying different cryptosystems for each use case is impractical, keeping all system details secret is difficult, and such secrecy prevents the public scrutiny that is needed to establish trust in a system's security. Instead, most cryptographic primitives are associated with a large *key space*, so that the primitive can be instantiated with different private keys but the algorithm itself can be made public (thereby allowing for cryptanalysis). This key space should be large enough to ensure that an exhaustive search over all the keys is computationally infeasible.

In modern cryptography, keys are either *symmetric* or *asymmetric*. Once again taking encryption as an example, in *symmetric-key cryptography* the sender and the receiver of a message share the same key, allowing either of them to encrypt and decrypt. This shared private key allows for sufficiently efficient cryptographic primitives, such as the widely-used Advanced Encryption Standard (AES). One logistic hurdle of symmetric-key cryptography is that the private key must be shared in advance between the sender and the receiver. But how? For a long time, key exchange seemed to require both parties to meet in advance or to make use of a secure private channel, such as a trusted courier.

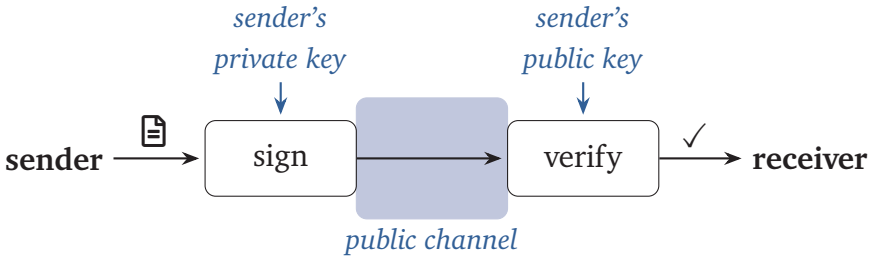
Fortunately, in 1976, Diffie and Hellman [DH76] came up with a more practical solution: let a user generate their own pair (k_1, k_2) of keys, so that one key k_1 can be made public while the other key k_2 remains private. Based on this concept, Diffie and Hellman proposed a practical key-exchange scheme and explained how asymmetric key pairs can be used for encryption and for another primitive called *digital signatures* (see Figure 1.1 for an illustration). The first practical schemes for asymmetric-key encryption and digital signatures followed shortly after [RSA78]. Diffie and Hellman's seminal work thereby laid the foundations for the field of *asymmetric-key cryptography*, also known as public-key cryptography.

Yet, asymmetric-key cryptography also has its downsides: all known schemes are far less efficient than their symmetric-key counterparts. For this reason, modern cryptography makes use of both, depending on the context. For example, to communicate over a public channel in a way that preserves confidentiality, an asymmetric-key scheme is generally used to exchange a (relatively short) key, which can then be used for the more efficient symmetric-key encryption of a much longer message. Similarly, the asymmetric-key primitive of digital signatures can

be used to achieve authentication in an untrusted setup (e.g., to verify the source of a public key before beginning encryption), whereas faster symmetric-key schemes like Message Authentication Codes (MACs) are used when the parties already share a secret key.



(a) Asymmetric-key encryption aims to ensure confidentiality. A message is encrypted using the receiver's public key, and the receiver can recover it ($\tilde{m} = m$) using the corresponding private key. Meanwhile, an adversary may observe the encrypted message but should not be able to obtain any meaningful information about the message itself.



(b) Digital signatures aim to ensure authenticity. A document is signed using the sender's private key, and the receiver can verify the signature (✓) using the corresponding public key. Meanwhile, an adversary may observe the signed document but should not be able to produce a valid signature themselves without the sender's private key.

Figure 1.1: Two important asymmetric-key primitives between two parties in the presence of adversaries.

By design, asymmetric-key cryptography relies on the existence of "hard" computational problems: it should be computationally infeasible to recover the private key given only the public key and other public information. Proving that such hard computational problems exist would be a breakthrough result in theoretical computer

science (as it would resolve the famous unproven conjecture that P ≠ NP). Instead, confidence in the security of asymmetric-key cryptography is typically based on the *assumption* that a certain well-studied computational problem is hard to solve. For instance, Diffie-Hellman's original scheme was based on the assumed hardness of the *discrete logarithm problem*, which also underlies the security of elliptic-curve cryptography (ECC) [Mil86; Kob87]. Another well known mathematical problem used to build asymmetric-key cryptography is the problem of finding the prime factors of a large composite integer, known as *integer factorization*. This problem underlies the famous RSA schemes [RSA78] for encryption and digital signatures, named after their inventors Rivest, Shamir, and Adleman. Both ECC and RSA have been widely used for decades, and confidence in their security has been based on the assumption that no efficient algorithm exists for solving the discrete logarithm and integer factorization problems — a belief that remains widely held, though only when the attacker is restricted to *classical* computation.

1.2 The quantum threat to cryptography

The situation changed in 1994, when Peter Shor proved that both the discrete-logarithm problem and integer factorization could be efficiently solved by a quantum computer [Sho97]. Quantum computation follows the laws of quantum mechanics, which is generally understood as the fundamental description of our physical universe. The idea of building a quantum computer was proposed about a decade earlier by Benioff [Ben82], Manin [Man80; Man99], and Feynman [Fey82; Fey85], and further developed by Deutsch [Deu85]. One of the original motivations was to apply it to simulate the dynamics of quantum systems, but soon after, quantum computation was shown to provide speedups over classical computation for a broader range of applications. Initially, the only provable speedups compared to classical computation were in the so-called black-box model, but Shor's seminal work demonstrated a quantum speedup for a problem of significant practical importance: breaking asymmetric-key cryptography.

The fundamental difference between quantum and classical computers lies in their basic units of computation and in the rules governing how these units interact and can be manipulated. Whereas the unit of (deterministic) classical computation is a single bit $b \in \{0, 1\}$, the quantum analog is called a quantum bit, or *qubit*, which can be in any linear combination of 0 and 1. That is, a quantum state formed from n qubits can be in a *superposition* of 2^n different configurations.

Quantum mechanics tells us that when we extract classical information from a quantum state (by "measuring" it), we obtain a single classical outcome.

Moreover, by applying linear operations to superpositions we can manipulate quantum states in order to put more weight onto favorable outcomes via *interference* effects (similar to how noise-cancelling headphones suppress unwanted sound waves). Another important feature of quantum computation is that of *entanglement*, which is a non-classical kind of correlation between quantum systems. By combining these features inherent to quantum mechanics, one can attempt to design a quantum algorithm that, hopefully, outperforms all known classical algorithms for certain computational tasks.

Indeed, that is precisely what Shor's quantum algorithms achieve: by exploiting the periodic structure behind the discrete-logarithm problem and integer factorization, a useful interference pattern can be created that allows us to extract the desired outcome efficiently. This shows that the mathematical problems behind ECC and RSA are easy to solve for quantum computers, despite being widely believed to be hard for classical computers. Consequently, once large-scale quantum computers are built, the cost of attacking these schemes is essentially of the same order as that of executing them, whereas from a security perspective we would like a very large separation between these costs.

Another quantum algorithm that impacts cryptography is Grover's search algorithm [Gro96], which provides a speedup for certain search tasks that naturally appear in both asymmetric-key and symmetric-key cryptography, such as the search for a private key. Grover's algorithm gives a quadratic speedup for the task of searching over the set of all possible keys, which can be counteracted by doubling key sizes. Fortunately for cryptography, this quantum speedup is not as significant as Shor's, and often only relevant when there is no additional structure to exploit. Although superquadratic quantum speedups over classical attacks appear feasible for symmetric-key cryptography (e.g., see [BSS22]), the quantum threat is considered to primarily affect asymmetric-key cryptography.

Quantum computing calls for a global transition to "quantum-safe" cryptography. At the time of writing, practical quantum computers are not powerful enough to break currently deployed cryptosystems, but rapid progress is being made (such as Google's most recent quantum chip [AAAA+25]). While expected timelines[1] can vary widely, there is a clear and pressing need to transition sooner rather than later: sensitive encrypted information could be stored already today,

[1] See [MP24] for an overview of experts' responses to when they expect a quantum computer to be capable of factoring a 2048-bit number in under 24 hours.

and decrypted once it becomes feasible (this is known as the *harvest now, decrypt later* issue). In addition, due to the multi-layered and interdependent nature of cryptography, migrating to different cryptosystems takes time and effort.[2]

Given the need to transition to quantum-safe cryptography, it is natural to ask what form this transition should take. We can separate two distinct proposals: *quantum cryptography* and *post-quantum cryptography*. In quantum cryptography, the idea is to build cryptographic protocols by leveraging quantum effects directly; that is, not only the attacker, but also the interacting parties are allowed to make use of quantum-mechanical principles. Interestingly, quantum-cryptographic protocols were already developed by Brassard and Bennett [BB14] and by Ekert [Eke91] several years *before* Shor's work motivated a change to quantum-safe cryptography. Both [BB14] and [Eke91] considered the task of quantum-key distribution (QKD), which allows two parties to generate a shared secret key. However, despite its name, a QKD protocol does not fully resolve the key-exchange issue of symmetric-key cryptography by itself: it assumes that the two parties can exchange authenticated messages, and therefore still requires a mechanism for authentication. At least for the near future, QKD and other quantum-cryptographic protocols are widely regarded as incomplete solutions for quantum-safe cryptography that face technical limitations, and several security authorities discourage their deployment [Nat20; Nat21; ABNA+23].[3]

In this thesis, we focus on the other approach, which currently appears more viable: post-quantum cryptography. It follows the conventional ("classical") paradigm of cryptography, but is based on mathematical problems believed to be hard for both classical *and* quantum algorithms. Asymmetric-key schemes based on problems other than the discrete-logarithm problem and integer factorization were already proposed in the late 1970s (such as McEliece's encryption scheme [McE78] and Lamport's digital-signature scheme [Lam79]), but the search for alternative proposals has become more urgent since the publication of Shor's algorithm.

Especially in the last decade, research on the design and cryptanalysis of post-quantum cryptographic schemes has seen a large increase. In 2016, the US National Institute of Standards and Technology (NIST), one of the primary orga-

[2]The quantum threat also highlights the importance of cryptographic agility, that is, designing cryptographic ecosystems that enable rapid replacement of (and upgrades to) primitives in response to new attacks.

[3]Quantum cryptography does allow for primitives that cannot be achieved using classical techniques alone. An example is quantum copy-protection [Aar09], which leverages the no-cloning theorem of quantum mechanics to prevent quantum software from being duplicated without authorization (whereas classical software can simply be copied).

nizations responsible for developing cryptographic standards, initiated a public call for post-quantum schemes [NIS16], which received many submissions from academia and industry. In 2024, several schemes were standardized by NIST (as announced in [NIS22b]), but at the time of writing an additional call for more digital-signature schemes is still ongoing [NIS22a]. Other standardization organizations are also involved in developing post-quantum standards, some of which follow the recommendations of NIST.

Candidates for post-quantum cryptography

There are various candidates for computational problems from which asymmetric-key cryptography can be built in a way that is believed (or at least hoped) to withstand classical *and* quantum attacks. The most common proposals belong to one of the following categories:

- Lattice-based cryptography
- Code-based cryptography
- Isogeny-based cryptography
- Multivariate-based cryptography
- Hash-based cryptography

Schemes for encryption and digital signatures have been proposed for each of these categories, except for hash-based cryptography, for which only digital-signature schemes are known.

Efficiency through structure. It is not only important that post-quantum cryptographic schemes resist classical and quantum attacks, but also that they are *practical*: sufficiently fast and compact to replace ECC, RSA, and other widely deployed schemes. Unfortunately, currently-proposed post-quantum schemes are either significantly slower than these traditional schemes or require longer key sizes (or both). Post-quantum schemes therefore often incorporate additional structure to enhance practicality, though this structure may also introduce avenues for attack.

NIST standards. In 2022, NIST announced [NIS22b] that it would standardize several schemes for two important asymmetric-key primitives: key-encapsulation mechanisms (KEMs) and digital signatures. KEMs are closely related to encryp-

tion, and are used to securely establish a shared symmetric key between two parties. See Table 1.1 for an overview of the schemes that NIST has announced for standardization at the time of writing. (The second KEM, HQC, was selected for standardization in a later round, after the 2022 selections [ABCC+25].)

Primitive	Scheme	NIST standard	Approach
KEM	Kyber [BDKL+18]	ML-KEM [NIS24a]	Lattice-based
	HQC [MABB+22]	—	Code-based
Digital signature	Dilithium [DKLL+18]	ML-DSA [NIS24b]	Lattice-based
	Falcon [PFHK+18]	FN-DSA	Lattice-based
	Sphincs$^+$ [BDEF+17]	SLH-DSA [NIS24c]	Hash-based

Table 1.1: Schemes selected for standardization by NIST. The table lists the original scheme names and the corresponding standard names and specifications where available.

As shown in the table, the schemes selected for standardization are lattice-based, code-based, and hash-based. An additional call for digital signatures is still ongoing [ABCC+24], and NIST has expressed particular interest in schemes that are not based on (structured) lattices, in order to ensure a wider range of approaches. Standardizing schemes based on diverse security assumptions may reduce the risk of systemic failure should new attacks emerge in the future.

1.3 Cryptanalytic model and methods

Cryptanalysis can be carried out at multiple levels, such as by studying the physical implementation of a cryptographic scheme, by analyzing the design of the scheme itself, or by indirectly addressing its security via its underlying hardness assumptions. In this thesis, we take the latter perspective. We contribute to the cryptanalysis of post-quantum cryptography by designing and analyzing algorithms for several mathematical problems underlying the security of lattice-based and code-based schemes. We now briefly outline the various cryptanalytic methods that we will use later on.

Classical and quantum adversary model. Since post-quantum cryptography is intended to remain secure in an era with large-scale quantum computers, its cryptanalysis should take into account an extra family of tools: quantum algorithms. We therefore distinguish between two algorithmic approaches:

- Classical cryptanalysis: classical techniques for attacking post-quantum cryptography
- Quantum cryptanalysis: quantum techniques for attacking post-quantum cryptography, focusing on which additional speedups are feasible compared to classical attacks

Both classical and quantum cryptanalysis are needed to gain confidence in the security of post-quantum schemes, as these schemes are newer and have received less scrutiny than ECC and RSA, including from a classical perspective. In particular, as these post-quantum schemes are based on different mathematical foundations, cryptanalysts are required to develop new mathematical tools and intuition.

Investigating whether quantum speedups are possible requires yet another mindset: quantum computers operate according to different, and arguably less intuitive, principles. The resulting quantum-cryptanalytic tools are also less developed. In order to understand the potential of quantum speedups, it can sometimes be helpful to view the problem as a more abstract one for which quantum speedups are known, such as unstructured search, collision finding, or hidden-subgroup problems. For example, the discrete-logarithm problem and integer factorization can both be formulated as special instances of the (Abelian) hidden-subgroup problem.

Asymptotic versus concrete analysis. To analyze the cryptanalytic impact of a classical or quantum algorithm, we want to estimate the amount of *resources* (such as time or memory) that it needs as a function of the parameters of the cryptographic scheme at hand.

There are two important approaches for estimating this attack cost: asymptotic and concrete. Asymptotic analysis considers the scaling of the resources as the scheme's parameters grow very large, allowing us to classify the attack cost into more and less efficient categories. This enables us to assess the extent to which parameters should be altered in response to a new attack (to achieve the same level of security as prior to the attack): a *quadratic* speedup can typically be circumvented by doubling relevant parameters, whereas an *exponential* speedup would force a

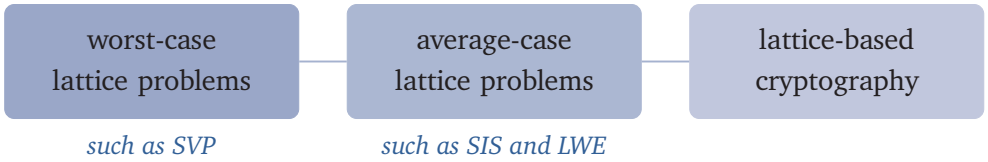
more significant change, often rendering a scheme unviable. Asymptotic estimates of the cost of an attack also allow comparison between different algorithms.

However, in order to understand the *practical* impact of an attack, an asymptotic analysis alone does not suffice. It only tells us how the used resources grow as the parameters get very large, which may not apply to parameter regimes that are relevant to cryptography. In particular, the cost of an attack may include some factors that are overruled by others when the parameters grow large (and hence hidden inside an asymptotic formula), but which make a significant difference for concrete instantiations. To determine appropriate parameters for a given level of security, it is therefore crucial to also analyze the *concrete* cost of an algorithm for a fixed set of parameters.

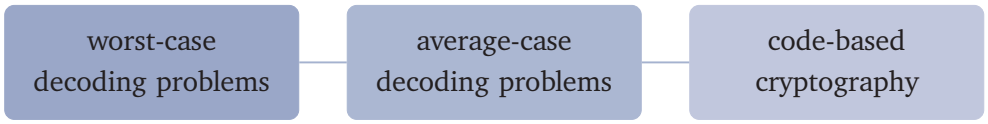
Use of experiments and heuristics. Theory often provides only *upper bounds* on the cost of an attack. Consequently, theoretical cryptanalysis is typically complemented by experiments that assess the practical performance of an attack, which may be substantially faster than a proven upper bound. When it is unclear how to prove performance guarantees that closely resemble empirical observations, cryptanalysts sometimes introduce heuristic assumptions in their analyses. These heuristics can, in turn, be tested experimentally. However, such experiments are often limited to smaller parameter sizes than those used in practice, because running attacks at cryptographic scale is generally infeasible (indeed, if it were feasible, this would indicate that the scheme is insecure).

1.4 Main contributions of this thesis

Post-quantum cryptographic candidates have not received as much scrutiny as the schemes (such as ECC and RSA) that they replace. While the NIST standardization process has increased cryptanalytic research, there are many important open questions regarding the security of proposed schemes against both classical and quantum attacks. In this thesis, we address several of these questions, focusing on two key candidates: lattice-based cryptography and code-based cryptography. We study their security by designing and analyzing algorithms for solving the underlying computational problems and for attacking the cryptographic constructions. Figure 1.2 illustrates the different levels at which one can attempt to cryptanalyze lattice-based and code-based cryptography (abbreviations are defined below).



(a) Lattice-based cryptography relies on the hardness of lattice problems.



(b) Code-based cryptography relies on the hardness of decoding problems.

Figure 1.2: Three levels of security assumptions in lattice-based and code-based cryptography: worst-case problems (hard for *some* input), average-case problems (hard for a *random* input), and the security of cryptographic constructions.

We briefly outline the contributions of this thesis.

Part I: Classical cryptanalysis

The first part of this thesis presents contributions to the classical cryptanalysis of lattice-based cryptography [DEL25; DEP25]. We consider two complementary ways of studying the security of lattice-based cryptography. First, we analyze the hardness of one of the central problems underlying many lattice-based schemes: the *Short Integer Solution* problem (SIS). Second, we consider a common tool for attacks on lattice-based cryptography, namely *lattice basis reduction*, and analyze its performance when leveraging the additional structure of the lattices used in many lattice-based schemes.

Wagner-style discrete Gaussian sampler and its application to SIS. The security of many lattice-based schemes depends on the assumed hardness of SIS and the closely related *Learning with Errors* problem (LWE). In 2015, Kirchner and Fouque [KF15] presented a subexponential-time algorithm for solving special

instances of LWE in certain parameter regimes. Their algorithm can be formulated as a two-step procedure:

1. Find many solutions to a related SIS problem
2. Use these solutions to solve LWE (via a so-called dual distinguisher)

However, an issue in their original proof was later discovered by [HKM18] and only partially resolved, leaving open whether the original claim of [KF15] can be recovered. This issue already arises in the first step, which aims to solve SIS, a problem worthy of study in its own right.

In Chapter 3, we build on Kirchner and Fouque's work and take a closer look at the first step of their algorithm. This step is reminiscent of Wagner's algorithm for the generalized birthday problem [Wag02], combined with a trick known as generalized lazy-modulus switching to obtain subexponential complexity. To prove that this Wagner-style algorithm actually outputs valid SIS solutions, we reinterpret it in a way that allows us to use a powerful tool in the analysis of lattice algorithms: the discrete Gaussian distribution.

Altogether, we obtain an algorithm that we refer to as a *Wagner-style discrete Gaussian sampler* for SIS, as it samples SIS solutions from a distribution close to a discrete Gaussian. In fact, this reformulation shows that the underlying algorithmic ideas are not specific to SIS, suggesting that they may find applications elsewhere. We partially address the question of recovering the original claim of [KF15] by proving that the Wagner-style discrete Gaussian sampler solves SIS in subexponential time in parameter regimes where previously known provable algorithms required exponential time. While these results for SIS do not immediately translate into conclusions for the overall LWE algorithm of [KF15], we suggest concrete next steps toward recovering their original claim.

Finally, we discuss the cryptanalytic implications of our results, as our results apply to a special case of SIS (namely, in the infinity norm) that underlies the security of the NIST standard ML-DSA (Table 1.1). Fortunately for cryptography, this Wagner-style algorithm does not appear to threaten the concrete security of ML-DSA, as state-of-the-art practical attacks remain significantly faster, despite having exponential asymptotic complexity.

Predicting the performance of module-lattice basis reduction. Our second contribution to lattice-based cryptanalysis considers a family of lattices known as *module lattices*. These lattices are equipped with additional algebraic structure that allows for more efficient cryptography, and they appear in many practical

schemes, including the lattice-based NIST standards (Table 1.1). It is therefore crucial to understand whether this additional structure introduces vulnerabilities exploitable by attacks. One of the key tools in practical lattice attacks is lattice basis reduction, particularly the *block Korkine-Zolotarev (BKZ) algorithm* [Sch87; SE94]. Although BKZ seems oblivious to the algebraic structure of module lattices, there is a natural module-lattice analog that exploits this structure [LPSW19; MS20], which we call *module-BKZ*.

To better understand the security of module-lattice-based cryptography, we study the practical performance of module-BKZ in Chapter 4 and compare it to unstructured BKZ. The performance of BKZ is commonly studied using heuristics known as the *Gaussian Heuristic* and the *Geometric Series Assumption*. The resulting heuristic-based analysis allows predicting the behavior of BKZ, and is backed by extensive experiments. We therefore adopt a similar approach for module-BKZ, and provide a predictive model for analyzing its practical behavior based on module-lattice analogs of the aforementioned heuristics, guided and supported by experiments with publicly available code.

These predictions allow us to compare the performance of module-BKZ and BKZ on module lattices. We show that module-BKZ does not outperform unstructured BKZ in all cases. Instead, its advantage depends on the algebraic number field over which the module lattice is defined, especially on the field's so-called *discriminant*. The advantage of module-BKZ applies to some schemes that have been proposed, but the lattice-based NIST standards do not appear to be affected so far. However, we highlight several important questions that must be addressed to properly assess the practical impact on these NIST standards and other module-lattice-based schemes.

Part II: Quantum cryptanalysis

The second part of this thesis presents contributions to the *quantum* cryptanalysis of lattice-based [ECGH+25] and code-based cryptography [EEL25]. In this part, the goal is to understand to what extent an attacker equipped with a quantum computer can improve over classical attacks. To this end, we explore the potential of several quantum algorithmic tools that are useful for speeding up search problems naturally arising in cryptanalysis: Grover's search algorithm [Gro96], amplitude amplification [BHMT02], and quantum-walk algorithms such as [MNRS11].

Quantum algorithm for 3-tuple lattice sieving and its application to SVP. Lattice-based cryptography fundamentally relies on the hardness of the *Shortest Vector Problem* (SVP). The fastest known classical [BDGL16] and quantum [BCSS23] algorithms for SVP are lattice sieving methods. These algorithms require both exponential time and exponential memory, but variants known as *k-tuple lattice sieving* allow for time-memory trade-offs. The fastest attacks correspond to $k = 2$, and increasing k reduces the memory requirement at the expense of time. The time complexity of k-tuple sieving can be somewhat improved using quantum techniques, such as Grover's search algorithm and quantum walks.

In Chapter 6, we present a faster quantum algorithm for 3-tuple lattice sieving. Under standard heuristics, this results in a faster quantum algorithm for SVP in the (limited) memory regime of 3-tuple sieving. Our result is obtained via a *nested* application of amplitude amplification aided by a classical preprocessing step. This preprocessing step is based on a common technique in lattice sieving known as *locality-sensitive filtering* (LSF) [BDGL16]. Using this technique, we construct a classical data structure that can be queried during the quantum algorithm. Prior quantum algorithms for lattice sieving considered a similar data structure for LSF, but either did not leverage it in the quantum part of the algorithm or prepared the data structure in superposition. Our work therefore shows how to exploit a classically prepared data structure for LSF within a quantum algorithm. Combined with nested amplitude amplification, this approach allows us to improve over the prior best time complexity for 3-tuple sieving from [CL23].

While quadratic (or even better) quantum speedups for SVP remain out of reach, our results suggest that the power of quantum speedups for sieving approaches to SVP is not yet fully understood. Our approach appears fairly general, and might lead to further quantum speedups beyond 3-tuple sieving. Natural next steps are therefore to investigate whether similar ideas yield faster quantum algorithms for 4-tuple sieving, or even for 2-tuple sieving, which currently corresponds to the fastest quantum algorithm for SVP when memory is unlimited. It also remains open whether our techniques can be combined with other approaches, such as quantum walks, to obtain additional improvements for k-tuple lattice sieving.

Quantum code sieving and its limitations in ISD algorithms. *Code-based* cryptography arises from a different mathematical setting than lattice-based cryptography, but similarities between the two sometimes allow for transferring cryptanalytic techniques. It was recently shown that techniques reminiscent of (2-tuple) lat-

tice sieving can be used to obtain sieving-based attacks on code-based cryptography. Specifically, the authors of [GJN23] presented a *code-sieving algorithm* that can be used as a subroutine in *information-set decoding* (ISD), an important class of algorithms for solving the computational problems underlying code-based cryptography. The resulting sieving-based ISD algorithm was further improved by [DEEK24] by incorporating more advanced techniques from lattice sieving, achieving improved time-memory trade-offs and an asymptotic runtime close to the state of the art among classical ISD algorithms.

Since lattice sieving can be sped up using quantum algorithms, we investigate in Chapter 7 whether code sieving admits quantum speedups as well, and whether such speedups lead to an improved *quantum ISD algorithm*. We present Grover-based and quantum-walk-based algorithms for the core search procedure of the code-sieving algorithm from [DEEK24], and show that these techniques yield quantum speedups for code sieving. Our algorithms are inspired by quantum algorithms for lattice sieving [Laa15; CL21], and we show how similar techniques can be adapted to the setting of codes. We accompany our asymptotic analysis with publicly available code that allows us to numerically optimize and compare the runtimes of the classical, Grover-based, and quantum-walk-based approaches. The relative speedups between these approaches are comparable to what is observed in lattice sieving.

Finally, we show that these quantum speedups for the code-sieving subroutine by themselves do not translate into an improved quantum ISD algorithm. We show that this *no-go result* also applies to natural modifications of the resulting sieving-based quantum ISD algorithm, and explain that the main bottleneck appears inherent to the current code-sieving framework. Our work therefore illustrates that the framework would need to be adapted to improve over the state of the art for quantum attacks on code-based cryptography.

Chapter 2

Preliminaries

This chapter introduces the notation and preliminary material used throughout the thesis. We also define the computational model used in our analysis. Since several of the following chapters focus on lattice-based cryptanalysis, we briefly review the necessary lattice background.

Contents of this chapter

2.1 Notation and basic concepts

General notation. The binary logarithm is denoted by $\log := \log_2$, and the natural logarithm by $\ln := \log_e$. We define $\exp(x) := e^x$. The complex conjugate of a number x is denoted by $\overline{x}$.

For a real number x, we let $\lfloor x \rfloor$ denote the largest integer n satisfying $n \leq x$, and $\lceil x \rceil$ the smallest integer satisfying $n \geq x$. We let $\lfloor x \rceil := \lfloor x + 1/2 \rfloor$ denote the nearest integer to x, with ties rounded upward.

We use standard notation for the open, half-open, and closed intervals between real numbers $x \leq y$, namely (x, y), $(x, y]$, $[x, y)$, and $[x, y]$. For non-negative integers $x \leq y$, we define $[x : y] := \{x, x+1, \ldots, y\}$ and $[x] := [1 : x]$. We write $\binom{y}{x} := \frac{y!}{x!(y-x)!}$ for the binomial coefficient.

Number systems. The set of (rational) integers is denoted by $\mathbb{Z}$, and its subset of positive integers by $\mathbb{N} := \{1, 2, 3, \ldots\}$. We write $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ for the sets of rational, real, and complex numbers, respectively. We have $\mathbb{N} \subsetneq \mathbb{Z} \subsetneq \mathbb{Q} \subsetneq \mathbb{R} \subsetneq \mathbb{C}$. The positive real numbers are denoted by $\mathbb{R}_{>0}$, and we define $\mathbb{R}_{\geq 0} := \mathbb{R}_{>0} \cup \{0\}$.

The set of integers modulo q is denoted by $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$, and we will sometimes identify it with $\{0, \ldots, q-1\}$. It is a commutative ring, and it is also a field if and only if q is prime. For $q = 2$, this gives the binary finite field, denoted $\mathbb{F}_2$.

Modules. The notion of a vector space can be generalized by replacing the field of scalars with a ring, giving what is known as a module. Let R be a commutative ring with multiplicative identity 1. A set M is an *R-module* if it forms an Abelian group under addition and there exists a multiplication operation $\cdot\colon R \times M \to M$ such that:

- $1 \cdot x = x$ for all $x \in M$
- $(rr') \cdot x = r \cdot (r' \cdot x)$ for all $r, r' \in R$ and $x \in M$
- $(r + r') \cdot x = r \cdot x + r' \cdot x$ for all $r, r' \in R$ and $x \in M$
- $r \cdot (x + x') = r \cdot x + r \cdot x'$ for all $r \in R$ and $x, x' \in M$

When R is a field, this corresponds to the usual notion of a vector space over R.

M is *finitely generated over R* if there is a finite set of elements $b_1, \ldots, b_n \in M$ such that

$$M = \left\{ \sum_{i=1}^{n} r_i b_i \colon r_i \in R \right\}.$$

If the b_i are also R-linearly independent (that is, for all $r_1, \ldots, r_n \in R$, $\sum_{i=1}^n r_i b_i = 0$ implies $r_1 = \cdots = r_n = 0$), then $\{b_1, \ldots, b_n\}$ is an *R-basis* of M, and we call M a *free R-module*. Since the module structure and the notion of a basis depend on the ring of scalars, we will specify the base ring whenever it is not clear from context.

While finitely-generated R-modules over a field R always admit an R-basis, this is not necessarily true for R-modules over arbitrary rings R. An example of a free $\mathbb{Z}$-module is $\mathbb{Z}^n$, which has a $\mathbb{Z}$-basis given by the standard basis of $\mathbb{R}^n$. For $q \geq 1$, $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ is an example of a finitely generated $\mathbb{Z}$-module that is not free over $\mathbb{Z}$.

Scalars, vectors, matrices. We usually denote a scalar by a non-bold lowercase letter (e.g., x), a vector by a bold lowercase letter (e.g., $\mathbf{x}$ with coefficients x_i), and a matrix by a bold uppercase letter (e.g., $\mathbf{X}$). We write $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ to indicate that $\mathbf{X}$ has columns $\mathbf{x}_i$. The transpose of a matrix $\mathbf{X}$ is denoted by $\mathbf{X}^\top$, and the determinant of $\mathbf{X}$ by $\det(\mathbf{X})$.

We denote the $n \times n$ identity matrix by $\mathbf{I}_n$. Moreover, we write $\mathbf{X} = [\mathbf{A} \mid \mathbf{B}]$ for the horizontal concatenation of matrices $\mathbf{A}$ and $\mathbf{B}$, and $\mathbf{x} = (\mathbf{a}\,;\,\mathbf{b})$ for the vertical concatenation of column vectors $\mathbf{a}$ and $\mathbf{b}$.

Norms and corresponding unit balls. Two common norms on the vector space $\mathbb{R}^n$ encountered in later chapters are the Euclidean norm (or ℓ_2-norm) and the ℓ_∞-norm. For $\mathbf{x} \in \mathbb{R}^n$, these are defined by

$$\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^n x_i^2} \qquad \text{and} \qquad \|\mathbf{x}\|_\infty := \max\{|x_i| : i = 1, \ldots, n\}$$

respectively. We have $\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n}\|\mathbf{x}\|_\infty$. We will sometimes write $\|\cdot\|$ as shorthand for $\|\cdot\|_2$.

The n-dimensional Euclidean unit ball is defined as $\mathcal{B}_n := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 \leq 1\}$, and we analogously define $\mathcal{B}_n^\infty := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_\infty \leq 1\}$.

In Chapter 4, we work with finite-degree field extensions $K \supseteq \mathbb{Q}$ (i.e., number fields) and introduce a non-geometric norm for vectors over K, known as the *algebraic norm* $\mathrm{N}(\cdot)$.

Probability notation. A random variable will usually be denoted by a non-bold uppercase letter (e.g., X). We write $X \sim D$ to denote that X is sampled from the distribution D. For a function f on the support of D, we write $\mathbb{E}_{X\sim D}[f(X)]$, or simply $\mathbb{E}_X[f(X)]$, to denote the expectation of $f(X)$ when $X \sim D$.

For a set $\mathcal{X}$ that admits a uniform distribution (such as a finite set), we write $U(\mathcal{X})$ for this uniform distribution on $\mathcal{X}$. For $N \in \mathbb{N}$, we let $U(\mathcal{X}, N)$ denote the distribution that samples a multiset $X \subseteq \mathcal{X}$ by selecting N elements from $\mathcal{X}$ independently and uniformly with replacement. We will refer to the resulting multiset X simply as a set, so its cardinality $|X|$ equals the total number of sampled elements, counting repetitions. When a proof requires explicit use of independence, we view X as an ordered sequence $X = (X_1, \ldots, X_N)$ of random variables.

2.1.1 Asymptotic terminology

We use standard Landau notation for classifying asymptotic formulas. That is, for functions $f\colon \mathbb{R} \to \mathbb{R}$ and $g\colon \mathbb{R} \to \mathbb{R}$, we write:

- $f(x) = O(g(x))$ if there exist constants $C, x_0 > 0$ such that $|f(x)| \le C\cdot|g(x)|$ for every $x \ge x_0$
- $f(x) = \Omega(g(x))$ if there exist constants $C, x_0 > 0$ such that $|f(x)| \ge C\cdot|g(x)|$ for every $x \ge x_0$
- $f(x) = \Theta(g(x))$ if both $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$
- $f(x) = o(g(x))$ if, for all constants $C > 0$, there exists $x_0 > 0$ such that $|f(x)| < C \cdot |g(x)|$ for every $x \ge x_0$ (equivalently, if $\lim_{x\to\infty} \frac{|f(x)|}{|g(x)|} \to 0$)
- $f(x) = \omega(g(x))$ if, for all constants $C > 0$, there exists $x_0 > 0$ such that $|f(x)| > C \cdot |g(x)|$ for every $x \ge x_0$ (equivalently, if $\lim_{x\to\infty} \frac{|f(x)|}{|g(x)|} \to \infty$)

Moreover, we use the following shorthand notation:

$$\begin{aligned}
\mathrm{poly}(f(x)) &:= f(x)^{O(1)} \\
\mathrm{polylog}(f(x)) &:= \log(f(x))^{O(1)} \\
\tilde{O}(f(x)) &:= O(f(x) \cdot \mathrm{polylog}(f(x))) \\
\tilde{\Omega}(f(x)) &:= \Omega(f(x)/\mathrm{polylog}(f(x)))
\end{aligned}$$

Categories of asymptotic growth. Common categories of the order of growth of functions f include:

- $f(x)$ is (at most) *polynomial* in x if $f(x) = x^{O(1)}$
- $f(x)$ is *subexponential* in x if $f(x) = 2^{o(x)}$
- $f(x)$ is (at most) *exponential* in x if $f(x) = 2^{O(x)}$
- $f(x)$ is *superexponential* in x if $f(x) = 2^{\omega(x)}$

Usually, a function f is called "negligible" if $f(x) = o(1/x^c)$ for all constants $c > 0$. However, as our notion of negligible depends on the context, we will be explicit with the intended meaning.

2.1.2 Tail bounds

Algorithmic analysis is often concerned with the tail behavior of random variables, namely probabilities of the form $\Pr[X \geq x]$ (the *upper tail*) and $\Pr[X \leq x]$ (the *lower tail*). Three common tools for studying such tail probabilities are Markov's inequality, Chebyshev's inequality, and the Chernoff bound.

Lemma 2.1.1 (Markov's inequality)**.** *Let X be a non-negative random variable with finite expectation $\mu := \mathbb{E}[X]$. Then $\Pr[X \geq (1+\delta)\mu] \leq 1/(1+\delta)$ for all $\delta \geq 0$.*

Lemma 2.1.2 (Chebyshev's inequality)**.** *Let X be a random variable with finite expectation $\mu := \mathbb{E}[X]$ and finite variance $\sigma^2 := \mathbb{E}[X^2] - \mu^2$. Then $\Pr[|X - \mu| \geq \delta\mu] \leq \sigma^2/(\delta\mu)^2$ for all $\delta > 0$.*

Lemma 2.1.3 (Chernoff bound [Che52])**.** *Let $X = \sum_{i=1}^{m} X_i$ be a sum of independent random variables $X_i \in \{0,1\}$ with expectation $\mu := \mathbb{E}[X]$. Then*

(i) $\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2}{2+\delta}\mu}$ *for all* $\delta \geq 0$.

(ii) $\Pr[X \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2}{2}\mu}$ *for all* $\delta \in (0,1)$.

(iii) $\Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\frac{\delta^2}{3}\mu}$ *for all* $\delta \in (0,1)$.

Some of our proofs in Part II rely on the following corollary of Lemma 2.1.3.

Corollary 2.1.4 (Simple application of the Chernoff bound)**.** *Let* $m\colon \mathbb{N} \to \mathbb{N}$. *For* $n \in \mathbb{N}$, *let* $X(n) = \sum_{i=1}^{m(n)} X_i(n)$ *be a sum of independent random variables* $X_i(n) \in \{0,1\}$. *Then* $X(n) \leq_n \max\{1, \mathbb{E}[X(n)]\}$, *except with probability* $2^{-\omega(n)}$. *Moreover, if* $\mathbb{E}[X(n)] = \omega(n)$, *then* $X(n) = \Theta(\mathbb{E}[X(n)])$, *except with probability* $2^{-\omega(n)}$.

Proof. Let $\mu := \mathbb{E}[X]$, where $X = X(n)$. Applying part (i) of Lemma 2.1.3 with $\delta = n^2 \max(1, 1/\mu)$ yields $\Pr[X \geq (1+\delta)\mu] \leq e^{-\delta\mu/2} \leq e^{-n^2/2} = 2^{-\omega(n)}$, where the last inequality uses that $\delta \geq 2$ for $n \geq 2$. Note that $(1+\delta)\mu \leq_n \max\{1, \mu\}$ by definition of δ. (This can be seen by separating the case $\mu \leq 1$ and $\mu > 1$.) To prove the second part, assume that $\mu = \omega(n)$. Applying part (iii) of Lemma 2.1.3 with arbitrary constant $\delta \in (0,1)$ yields $\Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\delta^2\mu/3} = 2^{-\omega(n)}$ by assumption on μ, from which the claim follows. □

2.2 Computational model

In this thesis, we model a classical computer as a random-access machine, which is a mathematical abstraction where basic operations (such as arithmetic operations, logical operations, and comparisons) take unit time and each memory location can be accessed in constant time, similar to physical computers with random-access memory (RAM). A quantum computer extends this model of computation: it can also perform quantum operations (elementary quantum gates and measurements) and access data via quantum analogs of RAM. In Part I of this thesis, we consider only classical algorithms; in Part II, we focus primarily on quantum algorithms.

In Section 2.2.1, we provide a brief overview of the basics of quantum computation. In Section 2.2.2, we describe two common models of quantum RAM. In Section 2.2.3, we explain how we measure computational cost for classical and quantum algorithms.

2.2.1 Quantum background

Quantum systems can be described by unit vectors in a complex *Hilbert space* $\mathcal{H}$, that is, a complex inner-product space that is complete with respect to the metric induced by its inner product. We consider only finite-dimensional Hilbert spaces in this thesis, in which case any inner-product space is complete.

Quantum states. A unit vector in $\mathcal{H}$ is called a *quantum state*. We denote quantum states using Dirac notation: a quantum state is denoted by $|\psi\rangle$ and its conjugate transpose by $\langle\psi|$. Two basic examples of quantum states are

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{and} \qquad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Every 2-dimensional quantum state $|\psi\rangle$ is in a *superposition* of $|0\rangle$ and $|1\rangle$, meaning that $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ for some $\alpha_0, \alpha_1 \in \mathbb{C}$ satisfying $|\alpha_0|^2 + |\alpha_1|^2 = 1$. We call a 2-dimensional quantum state a *qubit* or *1-qubit state*. Qubits form the fundamental units of quantum computation.

Taking the tensor product of two qubits yields a 4-dimensional quantum state such as $|0\rangle \otimes |1\rangle$, and repeating this process yields 2^n-dimensional quantum states for arbitrary $n \in \mathbb{N}$. When the tensor-product structure is clear, we denote the tensor product of quantum states $|\psi\rangle$ and $|\phi\rangle$ by the shorthand $|\psi\rangle |\phi\rangle$, and sometimes even write $|\psi, \phi\rangle$.

More concretely, a 2^n-dimensional quantum state $|\psi\rangle$ is called an *n-qubit state* and can be written as a superposition of the states $|i\rangle$ with $i \in \{0,1\}^n$, namely

$$|\psi\rangle = \sum_{i \in \{0,1\}^n} \alpha_i |i\rangle \tag{2.1}$$

for some $\alpha_i \in \mathbb{C}$ satisfying $\sum_{i \in \{0,1\}^n} |\alpha_i|^2 = 1$. The values α_i are called *amplitudes*, and the set $\{|i\rangle : i \in \{0,1\}^n\}$ is called the *computational basis* or standard basis of the Hilbert space $\mathcal{H}$ to which $|\psi\rangle$ belongs. We can represent $|\psi\rangle$ as a superposition of any other orthonormal basis states of $\mathcal{H}$.

Measurements. Classical information can be extracted from a quantum state via *(projective) measurements*. If we measure the n-qubit state $|\psi\rangle$ from Equation (2.1) in the computational basis, then we obtain outcome $i \in \{0,1\}^n$ with probability $|\alpha_i|^2$ (this axiom of quantum mechanics is known as "Born's rule"), and the quantum system has "collapsed" to $|i\rangle$. In general, this is an irreversible procedure, because the original amplitudes are gone after observing $|\psi\rangle$.

Mathematically, a measurement on an n-qubit state $|\psi\rangle \in \mathcal{H}$ corresponds to a collection $(P_j)_{j=1}^m$ of projectors on $\mathcal{H}$ such that $\sum_{j=1}^m P_j = \mathbf{I}_{2^n}$. A measurement of $|\psi\rangle$ yields outcome $j \in [m]$ with probability $\|P_j |\psi\rangle\|^2$, in which case the system

has collapsed to the quantum state

$$\frac{P_j |\psi\rangle}{\|P_j |\psi\rangle\|}.$$

A measurement in the computational basis corresponds to the collection of projectors $|i\rangle\langle i|$ for $i \in \{0,1\}^n$.

Unitary operations. Besides performing measurements, there is another way to manipulate quantum states: via unitary operations. Specifically, we can map an n-qubit state $|\psi\rangle$ to another one (in the same Hilbert space) by applying a *unitary* U to $|\psi\rangle$, defined as an inner-product-preserving matrix $U \in \mathbb{C}^{2^n \times 2^n}$. Unitary operations are therefore linear and reversible (each unitary U has an inverse U^{-1}).

Two examples of 1-qubit unitaries are the bitflip matrix X and the Hadamard matrix H defined as

$$X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \text{and} \qquad H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

We have $X|0\rangle = |1\rangle$ and $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Note that X and H are self-inverse. Applying H to each qubit of $|0\rangle^{\otimes n}$ yields the uniform superposition

$$\frac{1}{\sqrt{2^n}} \sum_{i \in \{0,1\}^n} |i\rangle .$$

In other words, a uniform superposition over all 2^n computational basis states can be created using only n applications of H. In Part II of this thesis, we will often make use of this fact to efficiently create a uniform superposition over all elements in a finite set.

Universal gate sets. Similar to how classical operations can be written as a sequence of elementary gates such as AND, OR, and NOT, we can decompose a unitary into a sequence of elementary *quantum gates*. All n-qubit unitaries can be written as a finite sequence of arbitrary 1-qubit unitaries and the 2-qubit unitary known as the CNOT gate [NC00, Sec. 4.5.2]. Such a set of quantum gates is called a *universal gate set*. From a practical point of view, the aforementioned gate set is not realistic: implementing all 1-qubit unitaries would require infinite precision. Instead, it is common to use a slightly weaker notion of universality, and consider a finite gate set from which all other unitaries can be efficiently approximated up to

sufficiently small precision. This is justified by the Solovay-Kitaev theorem [NC00, Appendix 3].

Remark 2.2.1 (Errors add linearly). Approximation errors in a sequence of unitaries accumulate at most linearly in operator norm. More precisely, let

$$U = U_t U_{t-1} \dots U_1 \qquad \text{and} \qquad \tilde{U} = \tilde{U}_t \tilde{U}_{t-1} \dots \tilde{U}_1$$

be products of unitaries on $\mathcal{H}$. If $\|U_i |\psi\rangle - \tilde{U}_i |\psi\rangle\| \leq \epsilon_i$ for all quantum states $|\psi\rangle \in \mathcal{H}$ and $i \in [t]$, then $\|U |\psi\rangle - \tilde{U} |\psi\rangle\| \leq \sum_{i\in[t]} \epsilon_i$ for all quantum states $|\psi\rangle \in \mathcal{H}$. To see that this holds for $t = 2$ (the general case follows for similar reasons): by the triangle inequality, for all quantum states $|\psi\rangle \in \mathcal{H}$, we have

$$\begin{aligned} \|U_2 U_1 |\psi\rangle - \tilde{U}_2 \tilde{U}_1 |\psi\rangle\| &= \|U_2(U_1 - \tilde{U}_1) |\psi\rangle + (U_2 - \tilde{U}_2)\tilde{U}_1 |\psi\rangle\| \\ &\leq \|U_2(U_1 - \tilde{U}_1) |\psi\rangle\| + \|(U_2 - \tilde{U}_2)\tilde{U}_1 |\psi\rangle\| \\ &= \|(U_1 - \tilde{U}_1) |\psi\rangle\| + \|(U_2 - \tilde{U}_2)\tilde{U}_1 |\psi\rangle\| \end{aligned}$$

which is $\leq \epsilon_1 + \epsilon_2$ if $\|U_1 |\psi'\rangle - \tilde{U}_1 |\psi'\rangle\| \leq \epsilon_1$ and $\|U_2 |\psi'\rangle - \tilde{U}_2 |\psi'\rangle\| \leq \epsilon_2$ for all quantum states $|\psi'\rangle \in \mathcal{H}$.

This general fact for $U, \tilde{U}$ allows one to relate the probabilities of measurement outcomes for the states $U |\psi\rangle$ and $\tilde{U} |\psi\rangle$. See [NC00, Sec. 4.5.3] for more details. These facts are for instance useful when dealing with imperfect quantum subroutines.

Quantum algorithms. A classical algorithm takes as input a bit string, performs a sequence of classical operations, and then outputs a bit string. Analogously, a *quantum algorithm* takes as input a quantum state, performs a sequence of unitary operations and measurements, and then outputs a quantum state, a bit string, or both. Note that a quantum algorithm $\mathcal{A}$ that applies only a sequence of unitaries (and no measurements) is reversible, and we denote its inverse by $\mathcal{A}^{-1}$.

The input of a quantum algorithm may be classical, since bit strings can be viewed as computational basis states. Besides the qubits needed to describe the input, a quantum algorithm may use *auxiliary qubits* (also known as ancilla qubits) that serve as additional workspace during the computation; these auxiliary qubits are typically initialized to $|0\rangle$.

Quantum algorithms generalize the notion of classical computation, because all classical operations can be implemented in a reversible manner using a 3-qubit unitary known as the Toffoli gate, which is a reversible NAND gate. This

implementation comes at some cost, but the overhead will be negligible for our purposes.

We refer the reader to [Wol23] or [NC00, Chapter 4] for more details on quantum computing. In Chapter 5, we will provide an overview of the main quantum-algorithmic tools that we will use in Part II.

2.2.2 Quantum RAM

Classical algorithms have access to classical data, stored in memory cells. However, many quantum algorithms require access to classical or quantum data in *quantum superposition*, including several algorithms used in quantum cryptanalysis. Such access can be modelled by introducing the notion of quantum RAM; see Figure 2.1 for an illustration. We distinguish between two types:

- Quantum-readable classical-writable *classical* memory (QCRAM)
- Quantum-readable quantum-writable *quantum* memory (QQRAM)

These types give rise to three models of quantum algorithms: the plain *quantum circuit model* (in which memory is not accessed in superposition, or the resources used to build quantum RAM are counted in the same way as other operations), the *QCRAM model*, and the *QQRAM model*. We describe the latter two models below.

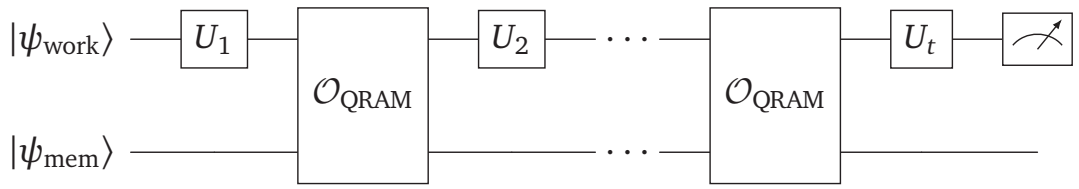


Figure 2.1: Illustration of a quantum algorithm and its usage of quantum RAM, inspired by [Pat23, Fig. 2.1]. The algorithm acts on two quantum registers: a *work register* that corresponds to the input and auxiliary qubits used for the computation, and a *memory register* that stores the classical or quantum memory accessed during the computation. It applies a sequence of unitaries $U_1, U_2, \ldots, U_t$ to the work register, alternated with queries $\mathcal{O}_{\text{QRAM}}$ to the memory register, and followed by a measurement. If the algorithm uses only QCRAM queries, then $|\psi_{\text{mem}}\rangle$ is a classical state and remains classical throughout.

Remark 2.2.2 (Terminology). There is no consensus on the right terminology in the literature for these models. QCRAM and QQRAM are sometimes called QRACM and QRAQM (respectively), and the term QRAM is sometimes used to refer to either model or as an umbrella term for both.

QCRAM. The QCRAM model considers classical memory stored as an n-bit string $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, and supports *QCRAM queries* to x, defined as applications of the unitary

$$\mathcal{O}_x \colon |i, b\rangle \mapsto |i, b \oplus x_i\rangle$$

where $i \in [n]$ and $b \in \{0,1\}$. By linearity, this operation allows us to read multiple bits of the classical data in superposition. In this model, write operations (that is, changes to the stored string x) can only be performed classically.

The unitary $\mathcal{O}_x$ corresponds to Figure 2.1 via the unitary

$$\mathcal{O}_{\text{QRAM}} \colon \overbrace{|w\rangle\,|i,b\rangle}^{\text{work register}}\underbrace{|x\rangle}_{\text{memory register}} \mapsto |w\rangle\,\mathcal{O}_x\,|i,b\rangle\,|x\rangle$$

where $|w\rangle$ represents the qubits in the work register on which $\mathcal{O}_x$ does not act.

In classical algorithms, allowing RAM queries with unit or logarithmic cost is standard practice. The intuition is that the n bits of memory can be, for example, arranged on the leaves of a binary tree with depth $\lceil \log_2 n \rceil$, and querying the i-th bit corresponds to traversing a $\lceil \log_2 n \rceil$-length path from the root to the i-th leaf of this binary tree. For similar reasons, QCRAM queries are often assumed "cheap" to execute (that is, in time $O(\log n)$) once the classical memory is stored in QCRAM.

QQRAM. While QCRAM operations act on classical memory, with a unitary $\mathcal{O}_x$ corresponding to a single classical string $x \in \{0,1\}^n$, the QQRAM model allows interaction with *quantum* memory. Specifically, this model additionally includes *QQRAM queries*, defined as applications of a unitary of the form

$$\mathcal{O} \colon |i\rangle\,|b\rangle\,|x\rangle \mapsto |i\rangle\,|x_i\rangle\,|x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n\rangle$$

where $x = (x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \in \{0,1\}^n$, $i \in [n]$, and $b \in \{0,1\}$. This operation swaps the contents of the $|b\rangle$ register and the i-th qubit in the memory

register $|x\rangle$, thereby allowing us to both read and write to the quantum memory in superposition. That is, using $\mathcal{O}$, we can implement the unitary maps

$$|i\rangle\,|b\rangle\,|x\rangle \mapsto |i\rangle\,|b \oplus x_i\rangle\,|x\rangle$$

and

$$|i\rangle\,|b\rangle\,|x\rangle \mapsto |i\rangle\,|b\rangle\,|x_1, \ldots, x_{i-1}, x_i \oplus b, x_{i+1}, \ldots, x_n\rangle\,.$$

In particular, the QQRAM model includes the QCRAM model as a special case.

The unitary $\mathcal{O}$ corresponds to Figure 2.1 via the unitary

$$\mathcal{O}_{\mathrm{QRAM}} \colon \overbrace{|w\rangle\,|i,b\rangle}^{\text{work register}}\ \underbrace{|x\rangle}_{\text{memory register}} \mapsto |w\rangle\,\mathcal{O}\,|i,b\rangle\,|x\rangle$$

where $|w\rangle$ again represents the qubits on which $\mathcal{O}$ does not act (note that $\mathcal{O}$ now also acts on the memory register).

Physical implementation. The physical implementation of QCRAM or QQRAM is nontrivial, and the assumption that they can be realized at low cost is controversial. This is often attributed to the challenge and costs of performing quantum error-correction on the whole device, whose size will have to be proportional to the number of stored bits or qubits rather than proportional to their logarithm (though the circuit depth may still be logarithmic). Yet, assuming access to QCRAM or QQRAM remains conceptually acceptable for theoretical purposes, as it combines two fairly uncontroversial concepts: classical RAM and quantum superposition. Similar to how classical RAM is assumed to be fast and error-free without a concern for error correction, one may hope that future quantum hardware can implement such memory with comparable reliability and efficiency as well. Moreover, determining the runtime of the best quantum algorithms for cryptographically relevant problems under fairly optimistic assumptions on the hardware is especially important, as such algorithms may jeopardize the security of existing cryptosystems.

2.2.3 Cost measures

The cost of an algorithm is typically expressed as a function $f(n)$ of the *input size* n, that is, the number of bits needed to specify the input. However, it is sometimes more natural to consider a different parameter. For instance, several algorithms studied in this thesis take vectors in $\mathbb{R}^m$ as input, and their cost is more naturally expressed in terms of the ambient dimension m or a closely related parameter. In such cases, we follow the common convention of representing real numbers with finite precision: sufficiently high so that we can ignore rounding errors, while ensuring that the encountered vectors in $\mathbb{R}^m$ can be represented using $\mathrm{poly}(m)$ bits. This overhead is negligible for the asymptotic analysis of the superpolynomial-time algorithms we consider.

When analyzing the cost of classical algorithms, we count each elementary operation as one *computational step*. For quantum algorithms, we similarly count a single elementary quantum gate (for a fixed universal gate set), QCRAM query, or QQRAM query as one computational step. The *time complexity* or *runtime* of an algorithm is the total number of steps it takes on a worst-case input. The *expected runtime* of an algorithm is the expectation of its runtime over the relevant randomness (which may be the algorithm's internal randomness, its input, or both). An algorithm is said to be *(time-)efficient* if its time complexity is polynomial in the input size.

The *memory complexity* of an algorithm is the number of classical bits (for classical memory) and/or qubits (for quantum memory) used by the algorithm.

Throughout this thesis, the use of quantum operations or quantum memory is explicitly indicated. When relevant, we count QCRAM or QQRAM queries separately to distinguish them from other quantum gates and clarify our assumptions about their cost.

2.3 Lattice background

In this section, we present the necessary background on lattices and briefly explain their relation to cryptography.

2.3.1 Lattices

Lattices and their bases. A basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ of a linear subspace of $\mathbb{R}^m$ allows writing each element of the subspace as an $\mathbb{R}$-linear combination of the $\mathbf{b}_i$. When instead restricting those combinations to the $\mathbb{Z}$-span of $\mathbf{B}$, we obtain what is commonly known as a *Euclidean lattice*, or *lattice* in short. In this thesis, we will typically denote lattices by symbols like $\mathcal{L}$ or Λ.

Definition 2.3.1 (Euclidean lattice)**.** A subset $\mathcal{L}$ of the Euclidean vector space $\mathbb{R}^m$ is a *lattice* if the following conditions are satisfied:

(*i*) $\mathcal{L}$ is an additive subgroup of $\mathbb{R}^m$: $\mathcal{L} \neq \emptyset$ and $-\mathbf{x}, \mathbf{x} + \mathbf{y} \in \mathcal{L}$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{L}$.

(*ii*) $\mathcal{L}$ is uniformly discrete: there exists $r > 0$ such that, for all $\mathbf{x} \in \mathcal{L}$, the set $\{\mathbf{y} \in \mathcal{L} \setminus \{\mathbf{x}\} : \|\mathbf{x} - \mathbf{y}\|_2 < r\}$ is empty.

We can equivalently define a lattice as a set of the form

$$\mathcal{L}(\mathbf{B}) := \mathbf{B}\mathbb{Z}^n = \left\{ \sum_{i=1}^{n} z_i \mathbf{b}_i : z_i \in \mathbb{Z} \right\} \subseteq \mathbb{R}^m$$

where $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{R}^{m \times n}$ is a matrix with linearly independent columns. We refer to n as the *dimension* of $\mathcal{L}(\mathbf{B})$, that is, the dimension is the $\mathbb{R}$-rank of

$$\operatorname{span}_{\mathbb{R}}(\mathcal{L}(\mathbf{B})) := \operatorname{span}_{\mathbb{R}}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} : \mathbf{x} \in \mathbb{R}^n\}.$$

The lattice is said to be of *full rank* if $n = m$. In the remainder of this thesis, a lattice is always assumed to have nonzero dimension (note that Definition 2.3.1 technically allows $\mathcal{L} = \{\mathbf{0}\}$).

Example 2.3.2. The integers $\mathbb{Z}^n$ form an n-dimensional lattice. For a lattice $\mathcal{L}$ and nonzero $\alpha \in \mathbb{R}$, the scaling $\alpha\mathcal{L}$ is also a lattice.

Let $\mathcal{L} \subseteq \mathbb{R}^m$ be an n-dimensional lattice. We call $\mathbf{B} \in \mathbb{R}^{m \times n}$ a *basis* of $\mathcal{L}$ if $\mathcal{L} = \mathcal{L}(\mathbf{B})$. When $n \geq 2$, this basis is not unique: by combining elements we can form infinitely

many different bases of $\mathcal{L}$. More concretely, given an arbitrary basis $\mathbf{B}$ of $\mathcal{L}$, the bases of $\mathcal{L}$ are exactly the matrices $\mathbf{B}' \in \mathbb{R}^{m\times n}$ that satisfy $\mathbf{B}' = \mathbf{BU}$ for $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$, where $\mathrm{GL}_n(\mathbb{Z}) := \{\mathbf{U} \in \mathbb{Z}^{n\times n} : \det(\mathbf{U}) = \pm 1\}$ is the (infinite) set of unimodular matrices.

Remark 2.3.3 ($\mathbb{Z}$-basis). A lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$ is in particular a free $\mathbb{Z}$-module (though the converse is not necessarily true), and $\mathbf{B}$ is a *$\mathbb{Z}$-basis* of $\mathcal{L}$.

Determinant or covolume of a lattice. An important invariant of a lattice $\mathcal{L}$ is its *determinant* $\det(\mathcal{L})$, also sometimes called its *covolume*, which is defined as the volume of the quotient group $\mathrm{span}_{\mathbb{R}}(\mathcal{L})/\mathcal{L}$. Equivalently (which is not entirely trivial), it is defined as

$$\det(\mathcal{L}) := \sqrt{|\det(\mathbf{B}^\top \mathbf{B})|}$$

for an arbitrary basis $\mathbf{B}$ of $\mathcal{L}$, where $\mathbf{B}^\top\mathbf{B}$ is called the *Gram matrix* of $\mathbf{B}$. The fact that $\det(\mathcal{L})$ is independent of the choice of lattice basis follows from the fact that two bases $\mathbf{B}$ and $\mathbf{B}'$ of $\mathcal{L}$ can be written as $\mathbf{B}' = \mathbf{BU}$ for $\mathbf{U} \in \mathrm{GL}_n(\mathbb{Z})$, so $|\det((\mathbf{B}')^\top\mathbf{B}')| = |\det(\mathbf{U}^\top\mathbf{B}^\top\mathbf{BU})| = |\det(\mathbf{U}^\top)\det(\mathbf{B}^\top\mathbf{B})\det(\mathbf{U})| = |\det(\mathbf{B}^\top\mathbf{B})|$.

Given a lattice $\mathcal{L}$ of dimension n, we define the lattice $\mathcal{L}^{(1)} = \mathcal{L}/\det(\mathcal{L})^{1/n}$, and call it the *normalized* lattice of $\mathcal{L}$, as it has determinant 1.

Remark 2.3.4 (Density of a lattice). The quantity $1/\det(\mathcal{L})$ is often viewed as a measure of how "dense" a lattice $\mathcal{L}$ is, that is, how close the lattice vectors are packed together. (Equivalently, $\det(\mathcal{L})$ measures how "sparse" $\mathcal{L}$ is.) This is because the span of an n-dimensional lattice can be written as

$$\mathrm{span}_{\mathbb{R}}(\mathcal{L}) = \sum_{\mathbf{x}\in\mathcal{L}} \mathbf{x} + \mathcal{P}(\mathbf{B}) \qquad \text{for} \qquad \mathcal{P}(\mathbf{B}) := \mathbf{B}\cdot[-1/2, 1/2)^n$$

where each $\mathbf{x} + \mathcal{P}(\mathbf{B})$ contains exactly one lattice element and has volume $\det(\mathcal{L})$.

First minimum. Another important geometric property of a lattice $\mathcal{L} \subseteq \mathbb{R}^m$ is the norm ("length") of its shortest nonzero vectors. For the ℓ_2-norm on $\mathbb{R}^m$, we define

$$\lambda_1(\mathcal{L}) := \inf\{\|\mathbf{x}\|_2 : \mathbf{x} \in \mathcal{L} \setminus \{\mathbf{0}\}\}.$$

Because $\mathcal{L}$ is a (nontrivial) uniformly discrete subgroup of $\mathbb{R}^m$, we have $\lambda_1(\mathcal{L}) > 0$, and the infimum is in fact a minimum. Accordingly, $\lambda_1(\mathcal{L})$ is called the *first minimum* or *minimum distance* of $\mathcal{L}$.

We define $\lambda_1^\infty(\mathcal{L})$ similarly for the ℓ_∞-norm.

Minkowski's bound. Minkowski's convex body theorem (also known as Minkowski's first theorem) provides an upper bound on $\lambda_1(\mathcal{L})$ and $\lambda_1^\infty(\mathcal{L})$ for an n-dimensional lattice $\mathcal{L}$:

$$\lambda_1^\infty(\mathcal{L}) \le \det(\mathcal{L})^{1/n} \qquad \text{and} \qquad \lambda_1(\mathcal{L}) \le 2\left(\frac{\det(\mathcal{L})}{\mathrm{vol}(\mathcal{B}_n)}\right)^{1/n} \le \sqrt{n}\det(\mathcal{L})^{1/n}$$

where $\mathcal{B}_n$ is the n-dimensional Euclidean unit ball.

Dual lattice. The *dual* of a lattice $\mathcal{L}$ is defined as

$$\mathcal{L}^* := \{\mathbf{y} \in \mathrm{span}_{\mathbb{R}}(\mathcal{L}) : \forall \mathbf{x} \in \mathcal{L}, \langle \mathbf{x}, \mathbf{y}\rangle \in \mathbb{Z}\}$$

which is a lattice as well, and satisfies $(\mathcal{L}^*)^* = \mathcal{L}$. A basis $\mathbf{B}$ of $\mathcal{L}$ yields a basis $\mathbf{D} := \mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1}$ of $\mathcal{L}^*$. If $\mathcal{L}$ is of full rank, then this dual basis equals $(\mathbf{B}^\top)^{-1}$. It can be shown that $\mathrm{span}_{\mathbb{R}}(\mathbf{B}) = \mathrm{span}_{\mathbb{R}}(\mathbf{D})$, so $\mathcal{L}$ and $\mathcal{L}^*$ have the same dimension. Moreover, $\det(\mathcal{L}^*) = 1/\det(\mathcal{L})$.

Example 2.3.5. We have $(\mathbb{Z}^n)^* = \mathbb{Z}^n$ (a lattice $\mathcal{L}$ with $\mathcal{L}^* = \mathcal{L}$ is called *self-dual*). For a lattice $\mathcal{L}$ and nonzero $\alpha \in \mathbb{R}$, the dual of $\alpha\mathcal{L}$ is $(1/\alpha)\mathcal{L}^*$.

Family of q-ary lattices. A lattice $\mathcal{L} \subseteq \mathbb{R}^m$ is called *q-ary* if $q\mathbb{Z}^m \subseteq \mathcal{L} \subseteq \mathbb{Z}^m$. Most q-ary lattices appearing in cryptography are constructed to have the following form (in fact, a q-ary lattice can always be written in either of these forms):

$$\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = 0 \bmod q\} \subseteq \mathbb{Z}^m,$$
$$\Lambda_q(\mathbf{A}) := \{\mathbf{y} \in \mathbb{Z}^m : \exists \mathbf{s} \in \mathbb{Z}_q^n, \mathbf{y} = \mathbf{A}^\top\mathbf{s} \bmod q\} = \mathbf{A}^\top\mathbb{Z}^n + q\mathbb{Z}^m \subseteq \mathbb{Z}^m$$

defined by a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. Both are of full rank, and they are duals up to appropriate scaling: namely, $\Lambda_q(\mathbf{A}) = q \cdot (\Lambda_q^\perp(\mathbf{A}))^*$.

If $m \ge n$ and $\mathbf{A}$ is of full rank, then we can assume without loss of generality that $\mathbf{A} = [\mathbf{A}' \mid \mathbf{I}_n]$ for some $\mathbf{A}' \in \mathbb{Z}_q^{n\times(m-n)}$. Then a basis of $\Lambda_q^\perp(\mathbf{A})$ is given by

$$\begin{pmatrix} 0 & \mathbf{I}_{m-n} \\ q\mathbf{I}_n & -\mathbf{A}' \end{pmatrix}.$$

In the literature one typically finds a slightly different basis, where the first $m-n$ rows are swapped with the last n rows. This given basis will, however, be more convenient for us in Chapter 3.

Projections and primitive sublattices. For $S \subseteq \mathbb{R}^m$, we write π_S for the projection onto $\mathrm{span}_{\mathbb{R}}(S)$ and $\pi_S^{\perp}$ for the projection orthogonal to $\mathrm{span}_{\mathbb{R}}(S)$. For example, if $S = \{\mathbf{x}\}$ for a nonzero vector $\mathbf{x} \in \mathbb{R}^m$, then we obtain (writing $\mathbf{x}$ instead of $\{\mathbf{x}\}$)

$$\pi_{\mathbf{x}}(\mathbf{y}) = \frac{\langle \mathbf{y}, \mathbf{x} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle}\mathbf{x} \qquad \text{and} \qquad \pi_{\mathbf{x}}^{\perp}(\mathbf{y}) = \mathbf{y} - \frac{\langle \mathbf{y}, \mathbf{x} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle}\mathbf{x}.$$

For an n-dimensional lattice $\mathcal{L}$ and nonzero $\mathbf{x} \in \mathcal{L}$, this yields a lattice $\pi_{\mathbf{x}}^{\perp}(\mathcal{L})$ of dimension $n - 1$. More generally, a k-dimensional sublattice $S \subseteq \mathcal{L}$ induces a lattice $\pi_S^{\perp}(\mathcal{L})$ of dimension $n - k$. (The case $k = 1$ is recovered by taking $S = \mathbf{x}\mathbb{Z}$.) Lattices of this form are called *projected* sublattices of $\mathcal{L}$. The dual $\mathcal{P}^*$ of $\mathcal{P} = \pi_S^{\perp}(\mathcal{L})$ is, moreover, a sublattice of $\mathcal{L}^*$. Bases of $\mathcal{P}$ and $\mathcal{P}^*$ can be efficiently obtained from bases of $\mathcal{L}$ and S.

In this thesis, we will consider projected sublattices obtained from (projecting) so-called primitive sublattices of $\mathcal{L}$.

> **Definition 2.3.6** (Primitive vector and primitive sublattice)**.** Let $\mathcal{L}$ be a lattice. A vector $\mathbf{x} \in \mathcal{L}$ is *primitive* (with respect to $\mathcal{L}$) if $\mathbf{x} \notin z\mathcal{L}$ for all integers $z > 1$. More generally, a sublattice $S \subseteq \mathcal{L}$ is *primitive* if $S = \mathrm{span}_{\mathbb{R}}(S) \cap \mathcal{L}$.

It can be shown that a k-dimensional sublattice $S \subseteq \mathcal{L}$ is primitive if and only if, for each basis $(\mathbf{b}_1, \ldots, \mathbf{b}_k)$ of S, there exists a basis of $\mathcal{L}$ of the form $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$. Given a primitive sublattice S of $\mathcal{L}$, there exists a sublattice $\mathcal{C} \subseteq \mathcal{L}$ that satisfies $S \oplus \mathcal{C} = \mathcal{L}$ (that is, $S + \mathcal{C} = \mathcal{L}$ and $S \cap \mathcal{C} = \{\mathbf{0}\}$), called a *complement* to S. While such a complement is not unique in general, a fixed $\mathcal{C}$ ensures that each $\mathbf{x} \in \pi_S^{\perp}(\mathcal{L})$ has a unique preimage ("lift") $\mathbf{y} \in \mathcal{C}$ such that $\pi_S^{\perp}(\mathbf{y}) = \mathbf{x}$. More precisely, $\pi_S^{\perp}$ induces a bijection from $\mathcal{C}$ to $\pi_S^{\perp}(\mathcal{C}) = \pi_S^{\perp}(\mathcal{L})$, which is efficiently computable in both directions.

For more details, we refer to [Mar03, Chapter 1].

GSO of a lattice basis. Given a basis $\mathbf{B} \in \mathbb{R}^{m \times n}$, we write $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ for the set of vectors obtained by Gram-Schmidt orthogonalization, defined as follows.[1]

Definition 2.3.7 (GSO)**.** Given $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for $\mathbb{R}$-linearly independent $\mathbf{b}_i$, we define its *GSO* as $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$, where $\mathbf{b}_1^* = \mathbf{b}_1$ and, for all $1 < i \leq n$,

$$\mathbf{b}_i^* := \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^* \quad \text{with } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \text{ for all } 1 \leq j < i.$$

In other words, $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$ for all $i \in [n]$, where $\pi_i := \pi^{\perp}_{\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}}$.

Note that the $\mathbf{b}_i^*$ are orthogonal, but not normalized.

While $\mathrm{span}_{\mathbb{R}}(\mathbf{B}^*) = \mathrm{span}_{\mathbb{R}}(\mathbf{B})$, the lattice $\mathcal{L}(\mathbf{B}^*)$ is typically not equal to $\mathcal{L}(\mathbf{B})$. For all $1 \leq i \leq j \leq n$, we obtain a projected sublattice $\mathcal{L}(\mathbf{B}_{[i:j]})$ of $\mathcal{L}$ with basis

$$\mathbf{B}_{[i:j]} := (\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j))$$

and of dimension $j - i + 1$. Note that each $\mathcal{L}(\mathbf{B}_{[i:j]})$ is a sublattice of $\pi_i(\mathcal{L}) = \mathcal{L}(\mathbf{B}_{[i:n]})$.

The determinant of $\mathcal{L}$ and of the projected sublattices $\mathcal{L}(\mathbf{B}_{[i:j]})$ can be expressed in terms of the GSO $\mathbf{B}^*$. Namely, for all $1 \leq i \leq j \leq n$, we have

$$\det(\mathcal{L}) = \prod_{i=1}^{n} \|\mathbf{b}_i^*\|_2 \qquad \text{and} \qquad \det(\mathcal{L}(\mathbf{B}_{[i:j]})) = \prod_{k=i}^{j} \|\mathbf{b}_k^*\|_2.$$

As we will see in Chapter 4, these projected sublattices play an important role in *lattice (basis) reduction* algorithms, which aim to transform a given lattice basis into one of better quality, in a suitable well-defined sense.

[1] Note that we also use the notation $\mathcal{L}^*$ for the dual of a lattice $\mathcal{L}$.

2.3.2 Computational problems based on lattices

As mentioned earlier, lattices form a useful building block for cryptography. So what then are the "hard" computational problems underlying the security of lattice-based cryptography?

Three fundamental lattice problems are the Shortest Vector Problem (SVP), the Closest Vector Problem (CVP), and Bounded Distance Decoding (BDD). As we will discuss below, these problems are considered hard in the worst case under standard complexity-theoretic assumptions. Through so-called *worst-case to average-case reductions* their worst-case hardness translates into the average-case hardness of two central problems in lattice-based cryptography: the Short Integer Solution problem (SIS) and the Learning with Errors problem (LWE).

For these problems, hardness is quantified in terms of the dimension of the input lattice. To support the security of lattice-based cryptography, the best known algorithms for the underlying problems should take exponential time in the dimension. Such asymptotic guarantees are not sufficient, as we also need concrete security: all practical algorithms should be too slow for cryptographically relevant dimensions (rather than as the dimension tends to ∞). In most proposed cryptosystems, lattice dimensions range from a few hundred to a few thousand. However, the state-of-the-art lattice attacks also apply algorithms to lower-dimensional lattices, so it is crucial to understand hardness in these smaller dimensions as well.

We will now formally define the aforementioned lattice problems. As a lattice consists of infinitely many vectors, the problem instance is typically specified by a basis. For a lattice $\mathcal{L} \subseteq \mathbb{R}^m$, a basis yields a finite representation of $\mathcal{L}$ using at most m^2 coefficients. While a lattice admits infinitely many bases, the computational hardness of lattice problems may depend heavily on the particular basis provided. We will return to this point later (see Chapter 4).

Worst-case lattice problems

Shortest Vector Problem. Every lattice $\mathcal{L}$ has at least one shortest nonzero element with respect to the ℓ_2-norm, whose length is given by $\lambda_1(\mathcal{L}) > 0$. This gives rise to the Shortest Vector Problem (SVP), which asks to find a shortest nonzero vector in a given lattice, possibly allowing for a shortest length up to a certain approximation factor $\gamma = \gamma(n)$ that may implicitly depend on the lattice dimension n. For $\gamma = 1$, this problem is known as *exact* SVP (or just SVP), and for

$\gamma > 1$ as *approximate* SVP. The following definition considers SVP in the ℓ_2-norm, but naturally generalizes to other norms.

> **Problem 2.3.8** (γ-SVP)**.** Given a basis of a lattice $\mathcal{L} \subseteq \mathbb{R}^m$ and a real $\gamma \geq 1$, find a nonzero lattice vector $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{x}\|_2 \leq \gamma \cdot \lambda_1(\mathcal{L})$.

Ajtai [Ajt96] proved that exact SVP is NP-hard under randomized reductions, thereby essentially resolving a fifteen-year-old conjecture by van Emde Boas [Emd81]. It is, however, unclear how to build cryptography from exact SVP, so lattice-based cryptosystems rely instead on the hardness of γ-SVP for polynomial approximation factor $\gamma = \text{poly}(n)$ with $\gamma \geq \sqrt{n}$, for which such hardness results are not known. In fact, for $\gamma \geq \sqrt{n}$, γ-SVP lies in the complexity class coNP [AR05], so under widely believed complexity assumptions those cryptographically relevant instances of γ-SVP are not NP-hard.

SVP with $\gamma = \text{poly}(n)$ is still believed hard to solve, even for quantum computers, with the best known algorithms taking at least exponential time. However, we cannot take γ too large: for instance, there exist classical polynomial-time algorithms for γ-SVP with approximation factor that is exponential $\gamma = 2^{O(n)}$ [LLL82] or even slightly subexponential $2^{O(n \log\log n/\log n)}$ [Sch87; GN08a; MW16].

Remark 2.3.9 (GapSVP). Complexity classes are defined for decision problems rather than search problems, so the hardness results for lattice problems like SVP are in fact derived via a suitable decision problem. The (promise) decision problem of SVP is known as GapSVP.

Closest Vector Problem. The task of SVP is to find a nonzero lattice vector within minimal distance from the origin $\mathbf{0} \in \mathbb{R}^m$. One may generalize this task to finding a closest lattice vector to an arbitrary target vector $\mathbf{t} \in \mathbb{R}^m$, yielding the Closest Vector Problem (CVP). *Exact CVP* (γ-CVP with $\gamma = 1$) asks to find a lattice vector within minimal distance from $\mathbf{t}$, whereas approximate variants ($\gamma > 1$) allow for some slack. Similar to SVP, the hardness of the problem relies on the approximation factor γ, with cryptographically relevant choices again corresponding to $\gamma = \text{poly}(n)$.

In the following, we write $\text{dist}(\mathbf{t}, \mathcal{L}) := \min\{\|\mathbf{t} - \mathbf{x}\|_2 : \mathbf{x} \in \mathcal{L}\}$ for the minimal distance between a vector $\mathbf{t} \in \mathbb{R}^m$ and a lattice $\mathcal{L} \subseteq \mathbb{R}^m$.

> **Problem 2.3.10** (γ-CVP)**.** Given a basis of a lattice $\mathcal{L} \subseteq \mathbb{R}^m$, a real $\gamma > 0$, and a vector $\mathbf{t} \in \text{span}_{\mathbb{R}}(\mathcal{L})$, find a lattice vector $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{t} - \mathbf{x}\|_2 \leq \gamma \cdot \text{dist}(\mathbf{t}, \mathcal{L})$.

Note that γ-CVP does not require the output vector to be nonzero, so a γ-CVP oracle will likely output $\mathbf{x} = \mathbf{0}$ when given $\mathbf{t} = \mathbf{0}$, and does therefore not directly yield an algorithm for γ-SVP. It was nevertheless shown in [GMSS99] that an algorithm for γ-CVP can be efficiently turned into an algorithm for γ-SVP (in the same dimension). The NP-hardness of exact CVP was already known since 1981 [Emd81], and does not rely on a randomized reduction.

Bounded Distance Decoding. One can also consider a variant of CVP where the target vector is guaranteed to be not too far from the lattice, quantified by a function $\alpha = \alpha(n)$ that may depend on the dimension n of the input lattice. This promise search problem is known as Bounded Distance Decoding (BDD).

> **Problem 2.3.11** (α-BDD)**.** Given a basis of a lattice $\mathcal{L} \subseteq \mathbb{R}^m$, a real $0 < \alpha < 1/2$, and a vector $\mathbf{t} \in \mathrm{span}_{\mathbb{R}}(\mathcal{L})$ that is promised to satisfy $\mathrm{dist}(\mathbf{t}, \mathcal{L}) \leq \alpha \cdot \lambda_1(\mathcal{L})$, find the unique lattice vector $\mathbf{x} \in \mathcal{L}$ such that $\|\mathbf{t} - \mathbf{x}\|_2 = \mathrm{dist}(\mathbf{t}, \mathcal{L})$.

By assumption, the target vector is within distance $< \lambda_1(\mathcal{L})/2$ of the lattice, which implies that there is a *unique* closest lattice vector $\mathbf{x}$ to $\mathbf{t}$. Note that BDD gets harder as α (and thus the promised distance) increases.

Average-case lattice problems

Worst-case to average-case reductions. SVP, CVP, and BDD are lattice problems whose hardness assumptions are for *worst-case* instances, and therefore do not naturally lend themselves for cryptography. Instead, one would like to build cryptography from problems that are hard *on average*, i.e., for random instances of the problem. It is, however, typically harder to gain confidence in the average-case hardness of computational problems, as most complexity-theoretic tools are for worst-case proofs.

For lattices, this changed in the 1990s, when Ajtai [Ajt96] proved that an algorithm for a certain average-case lattice problem would yield an efficient algorithm for a worst-case lattice problem. Such results are referred to as *worst-case to average-case reductions*: solving one problem in the worst case "reduces to" solving another problem in the average case. Since Ajtai's result, more of these kind of reductions have been established (e.g., [Reg09; Pei09]), which have been central to the development of lattice-based cryptography.

Short Integer Solution problem. The original average-case problem considered by Ajtai [Ajt96] is the Short Integer Solution problem (SIS), which Ajtai used to analyze the security of a cryptographic hash function.

> **Problem 2.3.12** ($\mathrm{SIS}_{n,m,q,\beta}$)**.** Let $n, m, q \in \mathbb{N}$ with $m \geq n$, and let $\beta \in \mathbb{R}_{>0}$. Given a matrix $\mathbf{A} \sim U(\mathbb{Z}_q^{n\times m})$, find a nonzero vector $\mathbf{x} \in \mathbb{Z}^m$ that satisfies $\mathbf{Ax} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\|_2 \leq \beta$.

While there is no explicit lattice in the above definition, SIS *can* be formulated as an average-case short vector problem. Indeed, Problem 2.3.12 is equivalent to finding a nonzero vector of norm at most β in the q-ary lattice $\Lambda_q^\perp(\mathbf{A}) \subseteq \mathbb{Z}^m$.

The hardness of SIS is quantified by the parameter β. Note that the problem becomes easy when we remove the norm condition (use Gaussian elimination) or when $\beta \geq q$ (take $\mathbf{x} = (q, 0, \ldots, 0)$). On the other hand, the problem becomes vacuously hard (i.e., typically no solution exists) when β is too small.

It makes sense to consider SIS with other norms than the ℓ_2-norm, and these variants are of great interest for cryptography. In Chapter 3, we will consider SIS in the ℓ_∞-norm (Problem 3.1.1), which underlies the security of Dilithium [DKLL+18], now standardized by NIST under the name ML-DSA.

Learning with Errors problem. A problem that is closely related to SIS is the Learning with Errors problem (LWE). This problem was proposed by Regev in his seminal paper [Reg09], along with an LWE-based asymmetric-key encryption scheme. Regev proved that solving LWE implies an efficient *quantum* algorithm for γ-SVP for some $\gamma \in \mathrm{poly}(n)$, which was subsequently turned into a classical reduction by Peikert [Pei09].

> **Problem 2.3.13** ($\mathrm{LWE}_{n,m,q,\beta}$)**.** Let $n, m, q \in \mathbb{N}$ with $m \geq n$, and let χ be a distribution on $\mathbb{Z}_q$. Given $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, where $\mathbf{A} \sim U(\mathbb{Z}_q^{n\times m})$ and $\mathbf{b} = \mathbf{A}^\top\mathbf{s} + \mathbf{e} \bmod q$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \sim \chi^m$, find $\mathbf{s}$.

LWE can be viewed as a BDD problem (Problem 2.3.11) on lattices of the form $\Lambda_q(\mathbf{A})$, where the target is $\mathbf{b}$ and the promised distance from the lattice is $\|\mathbf{e}\|_2$ (which can be upper bounded with high probability, depending on χ). Note that the lattices $\Lambda_q(\mathbf{A})$ are related to the SIS lattices $\Lambda_q^\perp(\mathbf{A})$ by duality.

Part I: Classical cryptanalysis

Chapter 3

Wagner-style discrete Gaussian sampling and its impact on SIS

In this chapter, we study an approach to solving the Short Integer Solution problem (SIS). SIS is one of the foundational average-case lattice problems that underpins, for example, the NIST standard ML-DSA, also known as Dilithium. We present an algorithm that provably solves nontrivial instances of SIS in subexponential time, using a strategy that is conceptually inspired by an algorithm due to Wagner. This result is also relevant to an open question about the asymptotic complexity of the Learning with Errors problem, and the analysis in this chapter suggests that the algorithmic strategy may apply to other problems. Finally, we address the impact on the concrete security of ML-DSA and illustrate that a superpolynomial asymptotic speedup does not necessarily translate into an improved attack in practice.

This chapter is based on [DEL25].

Contents of this chapter

3.1 Overview

3.1.1 Problem setting

The Short Integer Solution problem (SIS) asks to find an integer vector of norm at most β that satisfies a set of n equations in m variables modulo q, as formally defined in Problem 2.3.12. It was introduced by Ajtai [Ajt96] along with his worst-case to average-case reduction: the problem of finding a short basis in an arbitrary lattice (in the worst case) reduces to solving SIS (in the average case). Since then, a plethora of cryptographic schemes have been based on the presumed average-case hardness of SIS.

One can instantiate the above-mentioned norm in multiple ways. The worst-case to average-case reductions [Ajt96; MR07] consider SIS in the ℓ_2-norm, with norm bound β significantly smaller than q. However, when it comes to practical cryptographic schemes, designers are often tempted to consider SIS instances outside of the asymptotic coverage of the reduction, or even to consider variants of the problem. This is the case for the new NIST standard ML-DSA (formerly known as Dilithium [DKLL+18]), which considers SIS in the ℓ_∞-norm with norm bound β rather close to the modulus q.

The dual of the latter SIS variant is commonly known as the Learning with Errors problem (LWE) [Reg09] with "narrow" error distribution, say ternary errors. This version of LWE was proven to be solvable in slightly subexponential time by Kirchner and Fouque [KF15] using a variant of the Blum-Kalai-Wasserman algorithm (BKW) [BKW03], at least when the number of samples m is large enough. More precisely, it was claimed in [KF15] that m linear in n suffices, but a subsequent work of Herold, Kirshanova, and May [HKM18] found an issue in the proof. Although [HKM18] resolved the issue for $m \geq Cn \log n$ (for some constant $C > 0$), it was left open whether fewer samples would suffice. Whether this limitation on the number of samples is fundamental or an artifact of the proof was left unresolved.

The work in this chapter was motivated by the folklore reinterpretation of BKW as a combination of Wagner's algorithm [Wag02], applied to the dual lattice (thereby solving an SIS instance), and the Aharonov-Regev distinguisher [AR05]. This raises a natural question:

Using the tricks of Kirchner and Fouque, but applied only to the Wagner step, can one provably solve interesting instances of SIS in subexponential time?

Given the importance of understanding the hardness of SIS for lattice-based cryptography, this question is already worth studying on its own. Moreover, new insights into the Wagner step may ultimately help resolve the open question for narrow-error LWE.

3.1.2 Contributions

We answer the above question positively, namely we prove that a variant of Wagner's algorithm solves SIS with n equations modulo $q = n^{\Theta(1)}$ in subexponential time $\exp(O(n/\log\log n))$ for ℓ_∞-norm bound $\beta = q/\mathrm{polylog}(n)$, while only requiring $m = n + \omega(n/\log\log n)$ many SIS variables. We also achieve subexponential complexity for ISIS and SIS$^\times$, which are variants of SIS in the ℓ_2-norm, for ℓ_2-norm bound $\beta = q\sqrt{n}/\mathrm{polylog}(n)$. In fact, in each of these applications, the $\mathrm{polylog}(n)$ factor in the norm bound can be replaced by something slightly larger, such as $\beta = 2^{\log(n)^c}$ for an arbitrary constant $c < 1$. All prior algorithms (including heuristic ones) for these problems had exponential asymptotic complexity $\exp(\Omega(n))$.

Besides these applications, we provide a significant revision of the ideas and techniques at hand. First, we propose a more abstract interpretation of Wagner for lattices using a zigzag through a chain of projected lattices and superlattices, depicted in Figure 3.1. This emphasizes that the key principle of the algorithm is not tied to SIS, and its subexponential complexity really stems from the ease of constructing the appropriate chain of lattices in the case of SIS lattices.

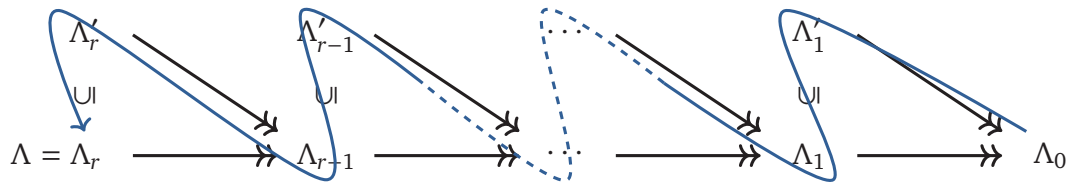


Figure 3.1: Wagner's algorithm interpreted over a chain of projected lattices Λ_i and auxiliary superlattices Λ'_i, where the goal is to sample from a lattice Λ. Thick black double-arrows denote surjective orthogonal projections between lattices. The blue curvy arrow denotes the path taken by the algorithm. Starting with many (short) vectors in the lattice Λ_0, it follows the zigzag path until it generates (short) vectors in the lattice Λ.

Furthermore, we circumvent the sample-amplification technique [Lyu05], and instead follow in the footsteps of [ADRS15; ADS15; AS18; ALS21; ACKS21] by relying on discrete Gaussian techniques to obtain provable results. Specifically, we use discrete Gaussian sampling and rounding to maintain precise control of the distribution of vectors throughout the chain of lattices.

Finally, we consider whether this approach threatens the concrete security of the NIST standard ML-DSA. It turns out that, despite its subexponential complexity, this algorithm is far less efficient against the concrete parameters of ML-DSA than standard lattice-reduction attacks.

3.1.3 Technical overview

The analog of SIS (Problem 2.3.12) in the ℓ_∞-norm is defined as follows.

> **Problem 3.1.1** ($\mathrm{SIS}^{\infty}_{n,m,q,\beta}$)**.** Let $n, m, q \in \mathbb{N}$ with $m \geq n$, and let $\beta \in \mathbb{R}_{>0}$. Given a matrix $\mathbf{A} \sim U(\mathbb{Z}_q^{n\times m})$, find a nonzero vector $\mathbf{x} \in \mathbb{Z}^m$ that satisfies $\mathbf{Ax} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\|_\infty \leq \beta$.

Note that this problem is easy to solve for $\beta \geq q/2$. Indeed, a nontrivial solution $\mathbf{x} \in \mathbb{Z}_q^m$ to $\mathbf{Ax} = \mathbf{0} \bmod q$ (efficiently obtained by Gaussian elimination) can be lifted to an integer vector by choosing representatives in $[-q/2, q/2)$ for each entry.

When an SIS instance $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is of full rank[1] we can, without loss of generality, assume that $\mathbf{A}$ is written in systematic form, i.e.,

$$\mathbf{A} = [\mathbf{A}' \mid \mathbf{I}_n] \quad \text{with} \quad \mathbf{A}' \in \mathbb{Z}_q^{n\times(m-n)}.$$

Using this writing of $\mathbf{A}$, it becomes clear that the problem of finding $\mathbf{x} \in \mathbb{Z}^m$ satisfying $\mathbf{Ax} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\|_\infty \leq \beta$ is equivalent to finding $\mathbf{z} \in \mathbb{Z}^{m-n}$ satisfying $\|\mathbf{x}(\mathbf{z})\|_\infty \leq \beta$, where $\mathbf{x}(\mathbf{z}) := (\mathbf{z} \,;\, -\mathbf{A}'\mathbf{z})$ is the vertical concatenation of $\mathbf{z}$ and $-\mathbf{A}'\mathbf{z}$. This reformulation of the problem naturally suggests a strategy inspired by the work of Wagner [Wag02].

[1] This happens with overwhelming probability when $m - n = \omega(1)$: a uniformly random matrix in $\mathbb{Z}_q^{n\times m}$ (with $m \geq n$) is of full rank with probability at least $1 - 1/q^{m-n}$.

Wagner-style algorithm for SIS

Wagner's generalized birthday algorithm [Wag02] addresses the problem of finding elements in a list $L_0 \subseteq \mathbb{Z}_q^n$ whose sum is $\mathbf{0}$ mod q, by progressively enforcing more coordinates to sum up to zero.[2] Similar ideas were independently used in [BKW03] to solve a dual problem.

Wagner's strategy yields an algorithm for Problem 3.1.1 as follows. Given an SIS instance $\mathbf{A} = [\mathbf{A}' \mid \mathbf{I}_n] \in \mathbb{Z}_q^{n \times m}$, divide the rows of the parity-check matrix $\mathbf{A}$ into r blocks of equal size n/r, as illustrated in Figure 3.2 (for $b_j := n/r$). The algorithm iteratively solves smaller SIS instances given by the parity-check matrices $\mathbf{A}_i = [\mathbf{A}'_i \mid \mathbf{I}_{in/r}]$ for $i = 1$ up to r, where each $\mathbf{A}'_i$ is defined by the first i blocks of rows of $\mathbf{A}'$.

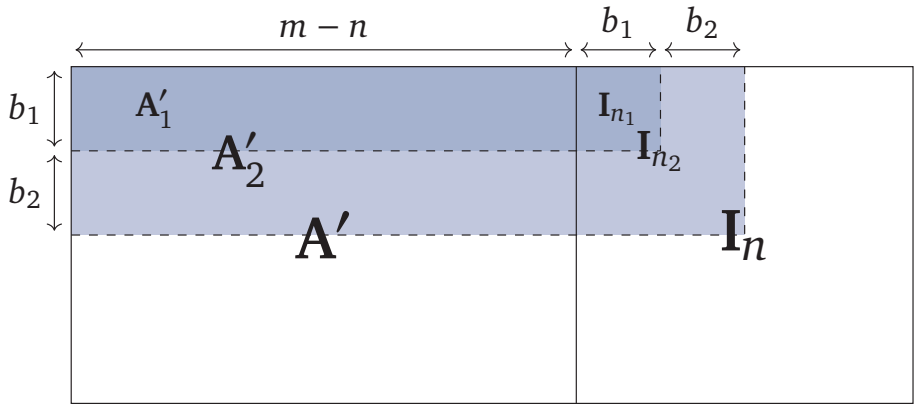


Figure 3.2: Illustration of the parity-check matrices $\mathbf{A}_i := [\mathbf{A}'_i \mid \mathbf{I}_{n_i}]$, where $\mathbf{A}'_i$ is the matrix defined by the first $n_i := \sum_{j=1}^{i} b_j$ rows of $\mathbf{A}'$. (Without the generalized lazy-modulus switching technique, the b_j are set equal to n/r.) Each iteration i of the algorithm solves an SIS instance given by parity-check matrix $\mathbf{A}_i$, until ultimately the whole matrix $\mathbf{A} = \mathbf{A}_r$ is covered.

Specifically, the algorithm begins by filling an initial list L_0 with short vectors in $\mathbb{Z}^{m-n}$. Then, in iteration i, it computes, for each $\mathbf{x} \in L_{i-1} \subseteq \mathbb{Z}^{m-n+(i-1)n/r}$, the unique (modulo q) vector $\mathbf{y} \in \mathbb{Z}^{n/r}$ such that $\mathbf{A}_i(\mathbf{x} \,;\, \mathbf{y}) = \mathbf{0}$ mod q. This vector $\mathbf{y}$ is likely to be of high norm since it is uniformly distributed modulo q. The algorithm then searches among those for a pair of vectors $\mathbf{x}'_1 = (\mathbf{x}_1 \,;\, \mathbf{y}_1)$, $\mathbf{x}'_2 = (\mathbf{x}_2 \,;\, \mathbf{y}_2)$

[2] Wagner originally considered $q = 2$, but the algorithm follows the same principle for $q > 2$.

that satisfy $\mathbf{y}_1 = \mathbf{y}_2 \bmod q$, to then add $\mathbf{x}'_1 - \mathbf{x}'_2 = (\mathbf{x}_1 - \mathbf{x}_2 \ ; \ \mathbf{0})$ to the list L_i. This ensures that the difference vector $\mathbf{x}'_1 - \mathbf{x}'_2$ remains short, as $\mathbf{x}_1, \mathbf{x}_2$ are short and the rest has norm zero. The final list L_r contains vectors $\mathbf{x} \in \mathbb{Z}^m$ that satisfy $\mathbf{A}\mathbf{x} = \mathbf{0} \bmod q$, providing potential nonzero and short solutions to the original SIS problem.

Lazy-modulus switching. The aforementioned approach constructs the lists L_i by combining vectors that lie in the same *bucket*, where the bucket of the vector $(\mathbf{x} \ ; \ \mathbf{y})$ corresponding to $\mathbf{x} \in L_{i-1}$ is labeled by the value of $\mathbf{y}$ modulo q. The vectors in L_i are then guaranteed to be of the form $(\mathbf{x}' \ ; \ \mathbf{0})$, but this is not necessary for the algorithm to succeed: a vector of the form $(\mathbf{x}' \ ; \ \mathbf{y}')$ with both $\mathbf{x}'$ and $\mathbf{y}'$ being short suffices. Based on this observation, the authors of [AFFP14] consider using a smaller modulus $p < q$, and combine $(\mathbf{x}_1 \ ; \ \mathbf{y}_1)$ and $(\mathbf{x}_2 \ ; \ \mathbf{y}_2)$ if $\lfloor \frac{p}{q}\mathbf{y}_1 \rceil = \lfloor \frac{p}{q}\mathbf{y}_2 \rceil \bmod p$, where the operation $\lfloor \cdot \rceil \bmod p$ applied to a vector is defined as rounding each coordinate to its nearest integer modulo p. This modified condition for combining vectors may result in a nonzero "rounding error" in the $\mathbf{y}'$-part that is traded for a reduced number of buckets: $p^{n/r}$ instead of $q^{n/r}$.

Two concurrent works [KF15; GJMS17] generalize this *lazy-modulus switching* technique by considering different moduli $p_1, \ldots, p_r$ (not necessarily prime). In this approach, the matrix $\mathbf{A}$ is divided into r blocks of respective size $b_1, \ldots, b_r$. The rounding errors induced by iteration i are at most q/p_i, and may double in each subsequent iteration when the vectors are combined. Hence, it makes sense to use decreasing moduli p_i to balance the accumulation of rounding error. Each step requires (more than) $p_i^{b_i}$ vectors to find collisions in $\mathbb{Z}_{p_i}^{b_i}$, leading to increasing block sizes b_i. This increasing choice of b_i's is central to the subexponential complexity claim of [KF15], as is shown below.

A naive analysis

Algorithm 3.1 is a rephrased version of the algorithm of [KF15] without the LWE dual-distinguishing step. Let us analyze it to highlight the core difficulty in proving its correctness.

List construction. Since each vector in L_i is either paired or left alone in a bucket, it follows that $|L_i| \leq 2|L_{i+1}| + p_i^{b_i}$. Thus, by initializing a list L_0 of size $3^r N$, we obtain at least N vectors in L_r, assuming $N \geq \max_i p_i^{b_i}$.

Algorithm 3.1: Wagner-style algorithm for SIS

Input: Integers n, m, q
Full-rank matrix $\mathbf{A} = [\mathbf{A}' \mid \mathbf{I}_n] \in \mathbb{Z}_q^{n \times m}$
Integer parameters $N, r, (p_i)_{i=1}^r, (b_i)_{i=1}^r$ with $\sum_{i=1}^r b_i = n$

Output: List of vectors $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{Ax} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\|_\infty \leq 2^r$

1: Initialize a list L_0 of $3^r N$ independent samples from $U(\{-1, 0, 1\}^{m-n})$
2: **for** $i = 1, \ldots, r$ **do**
3: Initialize empty lists $B(\mathbf{c})$ for each $\mathbf{c} \in \mathbb{Z}_{p_i}^{b_i}$
4: **for** $\mathbf{x} \in L_{i-1}$ **do** *// Bucketing*
5: Compute the unique $\mathbf{y} \in \mathbb{Z}_q^{b_i}$ satisfying $\mathbf{A}_i(\mathbf{x} \,;\, \mathbf{y}) = \mathbf{0} \bmod q$
6: Compute $\mathbf{c} = \left\lfloor \frac{p_i}{q} \mathbf{y} \right\rceil \bmod p_i$
7: Append $\mathbf{x}' := (\mathbf{x} \,;\, \mathbf{y})$ to $B(\mathbf{c})$
8: Initialize an empty list L_i
9: **for** $\mathbf{c} \in \mathbb{Z}_{p_i}^{b_i}$ **do** *// Combining*
10: **for** each two vectors $\mathbf{x}'_1, \mathbf{x}'_2$ in $B(\mathbf{c})$ **do**
11: Append $\mathbf{x}'_1 - \mathbf{x}'_2$ to L_i
12: Remove $\mathbf{x}'_1$ and $\mathbf{x}'_2$ from $B(\mathbf{c})$
13: **return** L_r

Bucketing and combining. At the beginning of each iteration $i \in \{1, \ldots, r\}$, the algorithm initializes some empty lists $B(\mathbf{c})$, the *buckets*, each labeled by a coset $\mathbf{c} \in \mathbb{Z}_{p_i}^{b_i}$. Every iteration works in two phases. First, the for-loop over the $\mathbf{x} \in L_{i-1}$ performs the *bucketing* phase: each "lift" $(\mathbf{x} \,;\, \mathbf{y})$ is added to a bucket depending on their $\mathbf{y}$-value. In the *combining* phase, the for-loop over each bucket representative $\mathbf{c} \in \mathbb{Z}_{p_i}^{b_i}$ takes differences $\mathbf{x}'_1 - \mathbf{x}'_2$ of many pairs of vectors $\mathbf{x}'_1, \mathbf{x}'_2$ in that bucket. Since $\mathbf{A}_i \mathbf{x}'_1 = \mathbf{0} \bmod q$ and $\mathbf{A}_i \mathbf{x}'_2 = \mathbf{0} \bmod q$, the difference vector $\mathbf{x}'_1 - \mathbf{x}'_2$ also satisfies $\mathbf{A}_i(\mathbf{x}'_1 - \mathbf{x}'_2) = \mathbf{0} \bmod q$ by linearity. Moreover, if two vectors

$\mathbf{x}'_1 = (\mathbf{x}_1 \ ; \ \mathbf{y}_1)$, $\mathbf{x}'_2 = (\mathbf{x}_2 \ ; \ \mathbf{y}_2)$ belong to the same bucket $B(\mathbf{c})$, then they satisfy $\mathbf{c} = \lfloor \frac{p_i}{q}\mathbf{y}_1 \rceil = \lfloor \frac{p_i}{q}\mathbf{y}_2 \rceil \bmod p_i$. In particular, $\mathbf{y}_1$ and $\mathbf{y}_2$ are "close" to each other, ensuring their difference is short: we have $\|\mathbf{y}_1 - \mathbf{y}_2\|_\infty \leq q/p_i$, and thus $\|\mathbf{x}'_1 - \mathbf{x}'_2\|_\infty \leq \max\{2\|\mathbf{x}_1\|_\infty, 2\|\mathbf{x}_2\|_\infty, q/p_i\}$. By induction over the r iterations, it follows that the algorithm outputs vectors $\mathbf{x} \in L_r$ satisfying

$$\|\mathbf{x}\|_\infty \leq \max_{0 \leq i \leq r} 2^{r-i} \frac{q}{p_i}, \tag{3.1}$$

where we define $p_0 := q$.

Time complexity of Algorithm 3.1

Equation (3.1) shows that the choice of the number of iterations r and the moduli p_i influences the maximal norm of the vectors. These parameters also affect the algorithm's time complexity, along with the other parameters that need to be chosen. Although the algorithm parameters N, r, p_i, and b_i should be integers to make sense, we consider them as real numbers for simplicity in this introductory section.

Parameter selection. Let the target norm for the SIS problem be $\beta = \frac{q}{f}$ for some factor $f > 1$. Note that the problem is easiest when $f = 1$ (norm q) and harder when f increases (as the norm q/f becomes shorter). The bound on the output norms given in Equation (3.1) is minimized when both terms are balanced: $2^r = 2^{r-i}q/p_i$. We therefore set $p_i := q/2^i$. There are $p_i^{b_i}$ distinct vectors modulo p_i with b_i coordinates, so to keep the number of samples comparable at each step i, we set $N = p_i^{b_i}$. This implies $b_i = \frac{\log N}{\log p_i} = \frac{\log_2 N}{\log_2(q)-i}$. By the choice of the p_i, the final vectors have a norm of at most 2^r. To ensure that it is at most $\beta = q/f$, it suffices to choose r such that $2^r \leq \frac{q}{f}$, so we set $r := \log_2(q/f) - 1$. We must also ensure that $n = n_r = \sum_{i=1}^r b_i$. Since the b_i increase with i, we can bound their sum by $n = \sum_{i=1}^r b_i \leq \int_1^{r+1} \frac{\log_2 N}{\log_2(q)-x}\mathrm{d}x = \log_2(N) \cdot \ln\left(\frac{\log_2(q)-1}{\log_2(q)-(r+1)}\right)$.[3] Thus,

$$n \leq \log_2(N) \cdot \ln\left(\frac{\log_2 q}{\log_2 f}\right).$$

[3] For f an increasing function, $\int_0^r f(x)\,\mathrm{d}x \leq \sum_{i=1}^r f(i) \leq \int_1^{r+1} f(x)\,\mathrm{d}x$. Also, for $A, B, a, b > 0$, $\int_a^b \frac{A}{B-x}\mathrm{d}x = A\ln(\frac{B-a}{B-b})$.

Rewriting, we conclude that taking $\log_2(N) = \frac{n}{\ln\ln(q) - \ln\ln(f)}$ suffices. Up to rounding of the parameters (as they need to be integers), we would get the following statement.

Tentative theorem. *Let $n, m, q \in \mathbb{N}$ and $f > 1$. There exists an algorithm that, given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, returns a (possibly zero) vector $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{Ax} = \mathbf{0} \bmod q$ and $\|\mathbf{x}\|_\infty \leq \frac{q}{f}$ in time*

$$T = 2^{\frac{n}{\ln\ln(q) - \ln\ln(f)}} \cdot \mathrm{poly}(n, \log q).$$

There is a catch! The output list of the algorithm contains a SIS solution only if at least one of the output vectors is *nonzero*. So it remains to be proven that the vectors do not all cancel out to zero. At the first iteration, the set $\{-1, 0, 1\}^{m-n}$ is significantly larger than the number of buckets, hence differences of vectors from the same bucket are unlikely to be zero vectors. However, from the second iteration onward, specifying the vector distributions is far from straightforward.

This issue does not arise in the original works of [Wag02; BKW03], where the algorithms operate in a regime that assumes m to be very large (whereas SIS and LWE are typically considered for $m = \mathrm{poly}(n)$). In the case of solving ternary LWE with [BKW03], this situation can be emulated using a sample-amplification technique from [Lyu05]. However, according to [HKM18], this requires at least $m = \Theta(n \log n)$ samples to start with; the argument of [KF15] that this should work for $m = \Theta(n)$ has been shown to be flawed in [HKM18].

Yet, it is intuitively not clear why this approach should fail when $m = \Theta(n)$. There is enough entropy to avoid collisions at the first step, and the entropy should intuitively increase at later stages, since the vectors are getting larger in ℓ_∞-norm. To formalize this intuition, we shift our perspective on the algorithm.

From parity-check perspective to lattices

As mentioned in Section 2.3.2, the SIS problem is equivalent to the short-vector problem that asks to find a nonzero vector of norm at most β in the lattice

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\}.$$

We consider the sequence of lattices $\Lambda_0 := \mathbb{Z}^{m-n}, \ldots, \Lambda_r := \Lambda_q^\perp(\mathbf{A})$ with

$$\Lambda_i := \Lambda_q^\perp(\mathbf{A}_i) = \{\mathbf{x} \in \mathbb{Z}^{m-n+n_i} : \mathbf{A}_i\mathbf{x} = \mathbf{0} \bmod q\} \tag{3.2}$$

for $i \geq 1$, where we refer to Figure 3.2 for the definitions of the $\mathbf{A}_i$. Each parity-check matrix $\mathbf{A}_i$ adds a block of b_i new coordinates, making Λ_{i-1} a projection of Λ_i. It is easy to sample bounded vectors in $\Lambda_0 := \mathbb{Z}^{m-n}$; for example, one can sample in $\{-1, 0, 1\}^{m-n}$ as was done in Algorithm 3.1. The goal is to use these initial samples to go back through the projections toward $\Lambda_q^\perp(\mathbf{A})$, as illustrated in Figure 3.1 (with $\Lambda_0 = \mathbb{Z}^{m-n}$ and $\Lambda = \Lambda_q^\perp(\mathbf{A})$), while keeping the norms bounded.

Although one can lift a vector from Λ_{i-1} to Λ_i, such a lifted vector would not be short because the lattice Λ_i we are lifting over is too sparse. Instead, what happens in Algorithm 3.1 is that we implicitly lift to a denser lattice $\Lambda_i' \supseteq \Lambda_i$, and then take the difference of two vectors in the same coset of the quotient Λ_i'/Λ_i to fall again in Λ_i. More explicitly, setting $\mathbf{c} = \lfloor \frac{p_i}{q}\mathbf{y} \rceil$ as in Algorithm 3.1, note that the vector $(\mathbf{x}, \mathbf{y} - \frac{q}{p_i}\mathbf{c})$ lives in the lattice $\Lambda_i' = \Lambda_i + (\{0\}^{m-n+n_{i-1}} \times \frac{q}{p_i}\mathbb{Z}^{b_i})$ which satisfies $|\Lambda_i'/\Lambda_i| = p_i^{b_i}$. Furthermore, the coset of that vector in the quotient Λ_i'/Λ_i is exactly determined by $\mathbf{c}$ mod p_i.

This provides a clean and pleasant interpretation of the algorithm as walking in a commutative diagram shown in Figure 3.1. It further provides the framework to deploy the full machinery of discrete Gaussian distributions over lattices: the initial samples can easily be made Gaussian over $\mathbb{Z}^{m-n}$, the lifting step as well using the randomized Babai algorithm [GPV08; EWY23a], and the combination step preserves Gaussians using convolution lemmas [Pei10; MP13]. (Those tools have already been used in [ADRS15; ADS15; ACKS21; AS18; ALS21] to solve short-vector problems.) There are further technicalities to control the independence between the samples, which we handle using the (conditional) similarity notion introduced in [ALS21]. This permits to control all the distributions throughout the algorithm, leading to provable conclusions. In particular, a careful choice of the algorithm's parameters guarantees, with high probability, that the final distribution is not concentrated at zero [PR06, Lemma 2.11].

3.1.4 Organization

Section 3.2 provides chapter-specific background on the tools used for analyzing the algorithms and lattices that we will encounter. In Section 3.3, we present our Gaussian sampler and analyze its asymptotic time complexity. In Section 3.4, we use the Gaussian sampler to asymptotically solve several variants of SIS, carefully avoiding the "canceling out to zero" issue. In Section 3.5, we discuss the impact of the attack on the concrete security of ML-DSA. We conclude the chapter in Section 3.6 by discussing some open questions, and explain why our result does not directly lead to a provable subexponential-time algorithm for LWE with narrow error distribution.

3.2 Preliminaries

Notation. For $x, y \in \mathbb{R}$ and $\delta \in \mathbb{R}_{\geq 0}$, the notation $x = e^{\pm\delta} y$ is used as shorthand for $x \in [e^{-\delta} y, e^{\delta} y]$. For a tuple $x = (x_1, \ldots, x_N)$ and $i \in [N]$, we define $x_{-i} := (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N)$, and we define $x_{-\{i,j\}}$ analogously.

3.2.1 Similarity of distributions

A common way to measure the distance between discrete random variables X and Y is the *statistical distance*, defined as $\Delta(X, Y) := \frac{1}{2} \sum_{x \in \mathcal{X}} |\Pr[X = x] - \Pr[Y = x]|$. Inspired by [ALS21], we consider instead the notions of *similarity* and *conditional similarity*, which are defined below and allow us to measure pointwise distances. These concepts are slightly stronger than statistical distance (see Remark 3.2.2), and are particularly useful for handling small dependencies arising from "bucket-and-combine" type of algorithms (like each iteration of Wagner-style algorithms).

Definition 3.2.1 (Similar)**.** Let D be a probability distribution over a set $\mathcal{X}$, and let $\delta \geq 0$ be a real. A random variable $X \in \mathcal{X}$ is δ-similar to D if, for all $x \in \mathcal{X}$, it holds that

$$\Pr_X[X = x] = e^{\pm\delta} \cdot \Pr_{Y \sim D}[Y = x].$$

Remark 3.2.2 (Similarity implies closeness in statistical distance). The notion of similarity is stronger than statistical distance: for all $\delta \in [0,1]$, being δ-similar implies being within statistical distance δ, since for any such δ we have $[e^{-\delta}, e^{\delta}] \subseteq [1-2\delta, 1+2\delta]$.

The next definition is closely related to [ALS21, Definition 4.1], but uses different terminology and is more general. Following [ALS21], we use the convention that $\Pr[E_0 \mid E_1] = 0$ if $\Pr[E_1] = 0$.

Definition 3.2.3 (Conditionally similar)**.** Let D be a probability distribution over a set $\mathcal{X}$, and let $\delta \geq 0$ be a real. Discrete random variables $X_1, \ldots, X_N \in \mathcal{X}$ are *conditionally δ-similar to independent samples from D* if, for all $i \in [N]$ and $x \in \mathcal{X}^N$, it holds that

$$\Pr_{X_1,\ldots,X_N}[X_i = x_i \mid X_{-i} = x_{-i}] = e^{\pm\delta} \cdot \Pr_{Y\sim D}[Y = x_i].$$

In particular, being conditionally 0-similar (i.e., $\delta = 0$) is equivalent to being independently distributed according to D, up to events of measure zero.

Remark 3.2.4 (Conditional similarity implies marginal similarity). For $N = 1$, conditional similarity coincides with Definition 3.2.1. More generally, for all $N \geq 1$ and $\delta \geq 0$, conditional δ-similarity implies marginal δ-similarity. Indeed, if $X_1, \ldots, X_N \in \mathcal{X}$ are conditionally δ-similar to independent samples from a distribution D on $\mathcal{X}$, then, for all $i \in [N]$ and $x \in \mathcal{X}$, we have $\Pr[X_i = x] = \sum_{y\in\mathcal{X}^{N-1}} \Pr[X_i = x | X_{-i} = y] \Pr[X_{-i} = y] = e^{\pm\delta} \cdot \Pr_{Y\sim D}[Y = x]$.

A useful property of similarity and conditional similarity is that these notions are closed under convex combinations (as already observed in [ALS21]).

Lemma 3.2.5. *Let D be a probability distribution over a set $\mathcal{X}$, and let $\delta \geq 0$ be a real. Let $\mathcal{E}$ be a countable set of mutually exclusive and collectively exhaustive events. If discrete random variables $X_1, \ldots, X_N \in \mathcal{X}$ satisfy*

$$\Pr_{X_1,\ldots,X_N}[X_i = x_i \mid X_{-i} = x_{-i} \text{ and } E] = e^{\pm\delta} \cdot \Pr_{Y\sim D}[Y = x_i]$$

for all $i \in [N]$, $x \in \mathcal{X}^N$, and $E \in \mathcal{E}$, then $X_1, \ldots, X_N$ are conditionally δ-similar to independent samples from D.

Proof. Suppose that the premise is true for random variables $X_1, \ldots, X_N$. Then, for all $i \in [N]$ and $x \in \mathcal{X}^N$, we have

$$\begin{aligned}
&\Pr_{X_1,\ldots,X_N}[X_i = x_i \mid X_{-i} = x_{-i}] \\
&= \sum_{E\in\mathcal{E}} \Pr_{X_1,\ldots,X_N}[X_i = x_i \text{ and } E \mid X_{-i} = x_{-i}] \\
&= \sum_{E\in\mathcal{E}} \Pr_{X_1,\ldots,X_N}[X_i = x_i \mid E \text{ and } X_{-i} = x_{-i}] \cdot \Pr_{X_1,\ldots,X_N}[E \mid X_{-i} = x_{-i}] \\
&\le \sum_{E\in\mathcal{E}} e^{\delta} \cdot \Pr_{Y\sim D}[Y = x_i] \cdot \Pr_{X_1,\ldots,X_N}[E \mid X_{-i} = x_{-i}] \qquad \text{(by assumption)} \\
&= e^{\delta} \cdot \Pr_{Y\sim D}[Y = x_i] \qquad \left(\text{since } \textstyle\sum_{E\in\mathcal{E}} \Pr[E \mid X_{-i} = x_{-i}] = \frac{\sum_{E\in\mathcal{E}} \Pr[E \text{ and } X_{-i}=x_{-i}]}{\Pr[X_{-i}=x_{-i}]} = 1\right)
\end{aligned}$$

and the lower bound follows for similar reasons. □

The following property can be viewed as the data-processing inequality for conditional similarity.

Lemma 3.2.6. *Let D be a probability distribution over a set $\mathcal{X}$, and let $\delta \ge 0$ be a real. If discrete random variables $X_1, \ldots, X_N \in \mathcal{X}$ are conditionally δ-similar to independent samples from D, then*

$$\Pr_{X_1,\ldots,X_N}[f(X_i) = 1] = e^{\pm\delta} \cdot \Pr_{Y\sim D}[f(Y) = 1]$$

for all $i \in [N]$ and all functions $f\colon \mathcal{X} \to \{0,1\}$.

Proof. Suppose that $X_1, \ldots, X_N$ are conditionally δ-similar to independent samples from D. Let $i \in [N]$ and let $f\colon \mathcal{X} \to \{0,1\}$ be an arbitrary function. Define $f^{-1}(1) := \{x \in \mathcal{X}\colon f(x) = 1\}$. Then

$$\begin{aligned}
\Pr_{X_1,\ldots,X_N}[f(X_i) = 1] &= \sum_{x\in\mathcal{X}} \Pr_{X_1,\ldots,X_N}[f(X_i) = 1 \text{ and } X_i = x] \\
&= \sum_{x\in f^{-1}(1)} \Pr_{X_1,\ldots,X_N}[X_i = x] \\
&\le \sum_{x\in f^{-1}(1)} e^{\delta} \cdot \Pr_{Y\sim D}[Y = x] \qquad \text{(by assumption)}
\end{aligned}$$

$$= e^{\delta} \cdot \sum_{x \in f^{-1}(1)} \Pr_{Y \sim D}[Y = x]$$
$$= e^{\delta} \cdot \Pr_{Y \sim D}[f(Y) = 1]$$

and the lower bound follows for similar reasons. □

3.2.2 Discrete Gaussian distribution and smoothness

In the following, when the subscripts s and $\mathbf{c}$ are omitted, they are respectively taken to be 1 and $\mathbf{0}$.

For any real $s > 0$ and $\mathbf{c} \in \mathbb{R}^n$, we define the Gaussian function on $\mathbb{R}^n$ centered at $\mathbf{c}$ with parameter s by

$$\forall \mathbf{x} \in \mathbb{R}^n, \quad \rho_{s,\mathbf{c}}(\mathbf{x}) := \exp(-\pi \|(\mathbf{x} - \mathbf{c})/s\|_2^2).$$

For any countable set A, we define $\rho_{s,\mathbf{c}}(A) = \sum_{\mathbf{x} \in A} \rho_{s,\mathbf{c}}(\mathbf{x})$. Note that $\rho_{s,\mathbf{c}}(\mathbf{x}) = \rho_s(\mathbf{x} - \mathbf{c})$, and thus $\rho_{s,\mathbf{c}}(A) = \rho_s(A - \mathbf{c})$.

For any real $s > 0$, $\mathbf{c} \in \mathbb{R}^n$, and full-rank lattice $\mathcal{L} \subseteq \mathbb{R}^n$, we define the discrete Gaussian distribution over $\mathcal{L}$ centered at $\mathbf{c}$ with parameter s by

$$\forall \mathbf{x} \in \mathcal{L}, \quad D_{\mathcal{L},s,\mathbf{c}}(\mathbf{x}) := \frac{\rho_{s,\mathbf{c}}(\mathbf{x})}{\rho_{s,\mathbf{c}}(\mathcal{L})} = \frac{\rho_s(\mathbf{x} - \mathbf{c})}{\rho_s(\mathcal{L} - \mathbf{c})}$$

and it is 0 for $\mathbf{x} \notin \mathcal{L}$.

Similarly, for any $\mathbf{t} \in \mathbb{R}^n$, we define $D_{\mathcal{L}-\mathbf{t},s,\mathbf{c}}(\mathbf{y}) := \frac{\rho_{s,\mathbf{c}}(\mathbf{y})}{\rho_{s,\mathbf{c}}(\mathcal{L}-\mathbf{t})}$ for $\mathbf{y} \in \mathcal{L} - \mathbf{t}$. (Note that $D_{\mathcal{L},s,\mathbf{c}} \equiv \mathbf{c} + D_{\mathcal{L}-\mathbf{c},s}$.)

Bounds on the norm of discrete Gaussian samples. We can tail-bound the ℓ_2-norm and ℓ_∞-norm of a discrete Gaussian sample using the following lemmas.

Lemma 3.2.7 ([Ban93, Lemma 1.5])**.** *For any full-rank lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and real $R \geq \sqrt{n/2\pi}$,*

$$\frac{\rho(\mathcal{L} \setminus R \cdot \mathcal{B}_n)}{\rho(\mathcal{L})} < (R\sqrt{2\pi e/n})^n \cdot e^{-\pi R^2}$$

where $R \cdot \mathcal{B}_n := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 \leq R\}$.

Lemma 3.2.8 ([Ban95, Lemma 2.10]). *For any full-rank lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and real $R > 0$,*

$$\frac{\rho(\mathcal{L} \setminus R \cdot \mathcal{B}_n^\infty)}{\rho(\mathcal{L})} < 2n \cdot e^{-\pi R^2}$$

where $R \cdot \mathcal{B}_n^\infty := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_\infty \leq R\}$.

Smoothness. The work of [MR07] introduced a lattice quantity known as the smoothing parameter. More precisely, for any full-rank lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and real $\varepsilon > 0$, we define the *smoothing parameter* $\eta_\varepsilon(\mathcal{L})$ as the smallest real $s > 0$ such that $\rho_{1/s}(\mathcal{L}^* \setminus \{\mathbf{0}\}) \leq \varepsilon$.

Intuitively, it gives a lower bound on s such that $D_{\mathcal{L},s}$ "behaves like" a continuous Gaussian distribution, in a specific mathematical sense. The following lemma justifies the name of the smoothing parameter.

Lemma 3.2.9 (Implicit in [MR07, Lemma 4.4]). *For any full-rank lattice $\mathcal{L} \subseteq \mathbb{R}^n$, real $\varepsilon \in (0, 1)$, real $s \geq \eta_\varepsilon(\mathcal{L})$, and $\mathbf{c} \in \mathbb{R}^n$,*

$$\frac{\rho_{s,\mathbf{c}}(\mathcal{L})}{\rho_s(\mathcal{L})} \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right].$$

For s slightly above smoothing, we can upper bound the probability of the most likely outcome of the discrete Gaussian distribution.

Lemma 3.2.10 (Min-entropy [PR06, Lemma 2.11]). *For any full-rank lattice $\mathcal{L} \subseteq \mathbb{R}^n$, reals $\varepsilon > 0$ and $s \geq 2\eta_\varepsilon(\mathcal{L})$, center $\mathbf{c} \in \mathbb{R}^n$, and vector $\mathbf{x} \in \mathbb{R}^n$,*

$$\Pr_{X \sim D_{\mathcal{L},s,\mathbf{c}}}[X = \mathbf{x}] \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot 2^{-n}.$$

In particular, for $\mathbf{x} = \mathbf{0}$, Lemma 3.2.10 upper bounds the probability that a discrete Gaussian sample is zero (if the standard deviation s is large enough).

Sampling and combining. In our algorithms, we sample from (scalings of) $\mathbb{Z}^n$ using the exact Gaussian sampler from [BLPR+13].

Lemma 3.2.11 (Implicit in [BLPR+13, Lemma 2.3])**.** *There is a randomized algorithm that, given a real $s \geq \sqrt{\ln(2n+4)/\pi}$ and $\mathbf{c} \in \mathbb{R}^n$, returns a sample from $D_{\mathbb{Z}^n,s,\mathbf{c}}$ in expected time* $\mathrm{poly}(n, \log s, \log\|\mathbf{c}\|_\infty)$.

Our analysis uses a variant of the convolution lemma [MP13, Theorem 3.3] (see also [Pei10, Theorem 3.1]) that bounds how similar the difference of two discrete Gaussians is to a discrete Gaussian. It is a slightly tighter result than [ALS21, Lemma 2.14].

Lemma 3.2.12 (Explicit variant of convolution lemma)**.** *Let $\mathcal{L} \subseteq \mathbb{R}^n$ be a full-rank lattice and let $s \geq \sqrt{2}\eta_\varepsilon(\mathcal{L})$ for some real $\varepsilon > 0$. For $i = 1, 2$, let $\mathcal{L}+\mathbf{c}_i$ be an arbitrary coset of $\mathcal{L}$ and Y_i an independent sample from $D_{\mathcal{L}+\mathbf{c}_i,s}$. Then the distribution of $Y_1 - Y_2$ satisfies*

$$\forall \mathbf{y} \in \mathcal{L} + \mathbf{c}_1 - \mathbf{c}_2, \quad \Pr[Y_1 - Y_2 = \mathbf{y}] \in \left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot D_{\mathcal{L}+\mathbf{c}_1-\mathbf{c}_2,\sqrt{2}s}(\mathbf{y}).$$

It follows, for instance, that $Y_1 - Y_2$ is 3ε-similar to $D_{\mathcal{L}+\mathbf{c}_1-\mathbf{c}_2,\sqrt{2}s}$ when $\varepsilon \leq \frac{1}{2}$.

Proof. Let $D_1 := D_{\mathcal{L}+\mathbf{c}_1,s}$ and $D_2 := D_{\mathcal{L}+\mathbf{c}_2,s}$. For $Y_1 \sim D_1$ and $Y_2 \sim D_2$, the distribution of $Y_1 - Y_2$ has support $\mathcal{L} + \mathbf{c}_1 - \mathbf{c}_2$. For all $\mathbf{x} \in \mathcal{L} + \mathbf{c}_1 - \mathbf{c}_2$, we have

$$\begin{aligned}
\Pr_{\substack{Y_1\sim D_1\\ Y_2\sim D_2}}[Y_1 - Y_2 = \mathbf{x}] &= \sum_{\mathbf{y}_1 \in \mathcal{L}+\mathbf{c}_1} \Pr_{\substack{Y_1\sim D_1\\ Y_2\sim D_2}}[Y_1 = \mathbf{y}_1 \text{ and } Y_1 - Y_2 = \mathbf{x}] \\
&= \sum_{\mathbf{y}_1 \in \mathcal{L}+\mathbf{c}_1} \Pr_{Y_1\sim D_1}[Y_1 = \mathbf{y}_1] \cdot \Pr_{\substack{Y_1\sim D_1\\ Y_2\sim D_2}}[Y_1 - Y_2 = \mathbf{x} \mid Y_1 = \mathbf{y}_1] \\
&= \sum_{\mathbf{y}_1 \in \mathcal{L}+\mathbf{c}_1} \frac{\rho_s(\mathbf{y}_1)}{\rho_s(\mathcal{L}+\mathbf{c}_1)} \cdot \frac{\rho_s(\mathbf{y}_1 - \mathbf{x})}{\rho_s(\mathcal{L}+\mathbf{c}_2)} \qquad \text{(def. of } D_1, D_2\text{)} \\
&= \rho_{\sqrt{2}s}(\mathbf{x}) \cdot \sum_{\mathbf{y}_1 \in \mathcal{L}+\mathbf{c}_1} \frac{\rho_{s/\sqrt{2}}(\mathbf{y}_1 - \mathbf{x}/2)}{\rho_s(\mathcal{L}+\mathbf{c}_1) \cdot \rho_s(\mathcal{L}+\mathbf{c}_2)} \qquad (3.3) \\
&= \rho_{\sqrt{2}s}(\mathbf{x}) \cdot \frac{\rho_{s/\sqrt{2}}(\mathcal{L} + \mathbf{c}_1 - \mathbf{x}/2)}{\rho_s(\mathcal{L}+\mathbf{c}_1) \cdot \rho_s(\mathcal{L}+\mathbf{c}_2)}
\end{aligned}$$

where Equation (3.3) holds since $\rho_s(\mathbf{v}_1) \cdot \rho_s(\mathbf{v}_1 - \mathbf{v}_2) = \rho_s(\mathbf{v}_2/\sqrt{2}) \cdot \rho_s(\sqrt{2}\mathbf{v}_1 - \mathbf{v}_2/\sqrt{2}) = \rho_{\sqrt{2}s}(\mathbf{v}_2) \cdot \rho_{s/\sqrt{2}}(\mathbf{v}_1 - \mathbf{v}_2/2)$ for all $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^n$. Since $s \geq \sqrt{2}\eta_\varepsilon(\mathcal{L})$, $\rho_{s/\sqrt{2}}(\mathcal{L} + \mathbf{c}_1 - \mathbf{x}/2) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \rho_{s/\sqrt{2}}(\mathcal{L})$ by Lemma 3.2.9. Hence,

$$\Pr_{\substack{Y_1 \sim D_1 \\ Y_2 \sim D_2}}[Y_1 - Y_2 = \mathbf{x}] \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \rho_{\sqrt{2}s}(\mathbf{x}) \cdot \frac{\rho_{s/\sqrt{2}}(\mathcal{L})}{\rho_s(\mathcal{L} + \mathbf{c}_1) \cdot \rho_s(\mathcal{L} + \mathbf{c}_2)}.$$

Summing both sides implies $1 \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \rho_{\sqrt{2}s}(\mathcal{L} + \mathbf{c}_1 - \mathbf{c}_2) \cdot \frac{\rho_{s/\sqrt{2}}(\mathcal{L})}{\rho_s(\mathcal{L}+\mathbf{c}_1) \cdot \rho_s(\mathcal{L}+\mathbf{c}_2)}$, i.e., $\frac{\rho_{s/\sqrt{2}}(\mathcal{L})}{\rho_s(\mathcal{L}+\mathbf{c}_1) \cdot \rho_s(\mathcal{L}+\mathbf{c}_2)} \in \left[1, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \frac{1}{\rho_{\sqrt{2}s}(\mathcal{L}+\mathbf{c}_1-\mathbf{c}_2)}$. It follows that

$$\Pr_{\substack{Y_1 \sim D_1 \\ Y_2 \sim D_2}}[Y_1 - Y_2 = \mathbf{x}] \in \left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \frac{\rho_{\sqrt{2}s}(\mathbf{x})}{\rho_{\sqrt{2}s}(\mathcal{L} + \mathbf{c}_1 - \mathbf{c}_2)}$$

as we wanted to show. □

3.2.3 Bounds on smoothing parameters of relevant lattices

The following lemma upper bounds the smoothing parameter of $\mathbb{Z}^n$. (Note that it can also be viewed as a special case of Lemma 3.2.14 below.)

Lemma 3.2.13 (Special case of [MR07, Lemma 3.3])**.** *For any real $\varepsilon > 0$,*

$$\eta_\varepsilon(\mathbb{Z}^n) \leq \sqrt{\frac{\ln(2n(1 + 1/\varepsilon))}{\pi}}.$$

The next lemma gives an upper bound on $\eta_\varepsilon(\mathcal{L})$ in terms of $\lambda_1^\infty(\mathcal{L}^*)$. It will be used to obtain an upper bound on the smoothing parameter of the lattices Λ_i that we consider.

Lemma 3.2.14 (Part of [Pei08, Lemma 3.5])**.** *For any full-rank lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and real $\varepsilon > 0$,*

$$\lambda_1^\infty(\mathcal{L}^*) \cdot \eta_\varepsilon(\mathcal{L}) \leq \sqrt{\frac{\ln(2n(1 + 1/\varepsilon))}{\pi}}.$$

The lattices Λ_i considered in this chapter (Equation (3.2)) are q-ary lattices of the form $\Lambda_i = \Lambda_q^\perp(\mathbf{A}_i)$ for some matrices $\mathbf{A}_i$, and we recall from Section 2.3.1 that their dual lattice is of the form $\frac{1}{q}\Lambda_q(\mathbf{A}_i)$. We obtain the following lower bound on $\lambda_1^\infty(\Lambda_q(\mathbf{A}))$ for random matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$.

Lemma 3.2.15 (Variant of [GPV08, Lemma 5.3])**.** *Let n, m, q be positive integers with q prime and $q^{1-n/m} \geq 6$. For $\mathbf{A} \sim U(\mathbb{Z}_q^{n\times m})$,*

$$\lambda_1^\infty(\Lambda_q(\mathbf{A})) > \frac{q^{1-n/m} \cdot 2^{-n/m}}{3}$$

except with probability $< 2^{-n}$. In particular, if $m \geq n$, then the right-hand side is lower bounded by $\frac{q^{1-n/m}}{6}$.

Proof. For some positive real B to be determined, let $S := \{\mathbf{y} \in \mathbb{Z}^m : \|\mathbf{y}\|_\infty \leq B\}$, and note that $|S| = (2B+1)^m$. For all $\mathbf{s} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\}$, $\Pr[\mathbf{A}^\top \mathbf{s} \bmod q \in S] = |S| \cdot q^{-m} = (2B+1)^m \cdot q^{-m}$. Taking the union bound over all $\mathbf{s} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\}$ gives

$$\begin{aligned}\Pr[\lambda_1^\infty(\Lambda_q(\mathbf{A})) \leq B] &= \Pr[\exists \mathbf{s} \in \mathbb{Z}_q^n \setminus \{\mathbf{0}\} \text{ such that } \mathbf{A}^\top \mathbf{s} \bmod q \in S] \\ &\leq |\mathbb{Z}_q^n \setminus \{\mathbf{0}\}| \cdot (2B+1)^m \cdot q^{-m} \\ &< (2B+1)^m \cdot q^{n-m}.\end{aligned}$$

Let $B := \frac{1}{3} q^{1-n/m} \cdot 2^{-n/m}$, and observe that $q^{1-n/m} \geq 6$ implies $q^{1-n/m} \cdot 2^{-n/m} \geq 3$ for all $m \geq n$. It follows that $B \geq 1$ and thus $\Pr[\lambda_1^\infty(\Lambda_q(\mathbf{A})) \leq B] < (3B)^m \cdot q^{n-m} = 2^{-n}$ as desired. The last part is immediate. □

Lemma 3.2.16 (Smoothing parameter of $\Lambda_q^\perp(\mathbf{A})$)**.** *Let n, m, q be positive integers with q prime, $m \geq n$, and $q^{1-n/m} \geq 6$. Let $\varepsilon \leq \frac{1}{4m}$ be a positive real. For $\mathbf{A} \sim U(\mathbb{Z}_q^{n\times m})$,*

$$\eta_\varepsilon(\Lambda_q^\perp(\mathbf{A})) < \sqrt{\frac{72 \ln(1/\varepsilon)}{\pi}} \cdot q^{n/m}$$

except with probability $< 2^{-n}$.

Proof. Let $\mathbf{A} \sim U(\mathbb{Z}_q^{n\times m})$. Since $m \geq n$, Lemma 3.2.15 implies $\lambda_1^\infty(\Lambda_q(\mathbf{A})) > \frac{1}{6}q^{1-n/m}$, except with probability $< 2^{-n}$. Since the dual of $\mathcal{L} := \Lambda_q^\perp(\mathbf{A})$ is $\mathcal{L}^* = \frac{1}{q}\Lambda_q(\mathbf{A})$, we obtain $\lambda_1^\infty(\mathcal{L}^*) > \frac{1}{6}q^{-n/m}$. Furthermore, by Lemma 3.2.14 (recall that $\mathcal{L}$ is full-rank and has dimension m), we have

$$\lambda_1^\infty(\mathcal{L}^*) \cdot \eta_\varepsilon(\mathcal{L}) \leq \sqrt{\frac{\ln(2m(1+1/\varepsilon))}{\pi}} \leq \sqrt{\frac{2\ln(1/\varepsilon)}{\pi}}$$

where the last inequality uses $\varepsilon \leq \frac{1}{4m}$. The statement follows. □

3.3 Wagner-style Gaussian sampler

In Section 3.1.3, we presented a warm-up version of the Wagner-style algorithm, which returns N short vectors in $\Lambda_q^\perp(\mathbf{A})$. However, these vectors are possibly all equal to $\mathbf{0}$. We now show that a variant of that algorithm, using discrete Gaussians, allows us to avoid that issue. In particular, we present a Wagner-style algorithm for sampling N vectors from a distribution that is essentially $D_{\Lambda_q^\perp(\mathbf{A}),s}$ when s is sufficiently large. Such samples can be shown to be short and nonzero with high probability. Specifically, we present an algorithm for sampling from $D_{\Lambda_q^\perp(\mathbf{A}),s}$ in expected subexponential time for $m = n + \omega(n/\log\log n)$, $q = n^{\Theta(1)}$, and $s = q/f$ for some $f = \omega(1)$.

Recall that, for some $r \in \mathbb{N}$ and $b_1, \ldots, b_r$ such that $n = \sum_{i=1}^r b_i$, we define the q-ary lattices

$$\Lambda_0 = \mathbb{Z}^{m-n} \quad \text{and} \quad \Lambda_i = \Lambda_q^\perp(\mathbf{A}_i) = \{\mathbf{x} \in \mathbb{Z}^{m-n+n_i} : \mathbf{A}_i\mathbf{x} = \mathbf{0} \bmod q\} \tag{3.4}$$

for $i = 1, \ldots, r$, where $\mathbf{A}_i \in \mathbb{Z}_q^{n_i\times(m-n+n_i)}$ is the matrix corresponding to the first $n_i := \sum_{j=1}^i b_j$ SIS equations (recall Figure 3.2).

Our approach to sample N vectors from $D_{\Lambda_q^\perp(\mathbf{A}),s}$ for a given parameter s is to start from many vectors sampled from D_{Λ_0,s_0}, where s_0 is such that $s = \sqrt{2^r}s_0$. Then, we iteratively (for $i \in \{1, \ldots, r\}$) transform a list of vectors that are conditionally similar to independent samples from $D_{\Lambda_{i-1},\sqrt{2^{i-1}}s_0}$ into a list of samples that are conditionally similar to independent samples from $D_{\Lambda_i,\sqrt{2^i}s_0}$. Then, after the last iteration, the list contains samples that are conditionally similar to independent samples from $D_{\Lambda_q^\perp(\mathbf{A}),s}$, as desired. (Using Lemma 3.2.10, we can then lower bound the probability that one such sample is nonzero.)

As explained in Section 3.1.3, mapping vectors in Λ_{i-1} to vectors in Λ_i will be done by first lifting the vectors in Λ_{i-1} to vectors in a suitable superlattice $\Lambda'_i \supseteq \Lambda_i$, and then combining them into vectors in Λ_i. Specifically, for some $p_i \in \mathbb{N}$, the lattices Λ'_i are defined by $\Lambda'_i = \mathcal{L}(\mathbf{B}'_i)$ for

$$\mathbf{B}'_i := \begin{pmatrix} \mathbf{0} & \mathbf{I}_{m-n} \\ \mathbf{D}_i & -\mathbf{A}'_i \end{pmatrix} \text{ with } \mathbf{D}_i := \begin{pmatrix} \mathbf{0} & q\mathbf{I}_{n_{i-1}} \\ \frac{q}{p_i}\mathbf{I}_{b_i} & \mathbf{0} \end{pmatrix} \tag{3.5}$$

The first b_i columns of $\mathbf{B}'_i$ generate (an embedding into $\mathbb{R}^{m-n+n_i}$ of) $\frac{q}{p_i}\mathbb{Z}^{b_i}$, which is a primitive sublattice of Λ'_i that we denote by $\mathcal{S}$. Consider the projected lattice $\mathcal{P} = \pi_{\mathcal{S}}^{\perp}(\Lambda'_i)$, and note that it is (an embedding of) Λ_{i-1}. Thus, we can consider ways to lift from Λ_{i-1} to Λ'_i; in Section 3.3.1, we consider a randomized way of lifting that preserves discrete Gaussian distributions.

From there, we would like to produce samples in Λ_i, rather than Λ'_i. A natural approach is to bucket our samples according to their cosets in the quotient Λ'_i/Λ_i, and then take differences within those buckets. Using standard analysis of convolutions of discrete Gaussians, we show in Section 3.3.2 that these differences are still essentially discrete Gaussian, yet with a width parameter increased by a factor of $\sqrt{2}$.

In Section 3.3.3, we then lay out the resulting Gaussian variant of Wagner's algorithm, and demonstrate its correctness and time complexity, under certain smoothing constraints on the parameters. Finally, in Section 3.3.4 we provide a choice of parameters that satisfy these constraints and allow for a subexponential-time algorithm for sampling from $D_{\Lambda_q^{\perp}(\mathbf{A}),s}$.

3.3.1 Discrete Gaussian lifting

Algorithm 3.2 below lifts vectors from Λ_{i-1} to vectors in Λ'_i. It revisits the GPV sampling algorithm [GPV08] with a reinterpretation of the induction: rather than reducing the problem in dimension n to two instances in dimensions 1 and $n-1$ (respectively), we consider arbitrary splits in dimensions n' and $n-n'$. Algorithm 3.2 can be viewed as a special case of [EWY23a, Alg. 2], and our Lemma 3.3.1 is a variant of [EWY23a, Theorem 1], where we analyze the conditional similarity of the output with respect to the discrete Gaussian (instead of merely looking at the statistical distance).

In particular, Lemma 3.3.1 (with $\delta = 0$ and $N = 1$) shows that Algorithm 3.2 turns a sample from $D_{\mathcal{P},s}$ into a sample that is 3ε-similar to (and thus within statistical distance 3ε from) $D_{\mathcal{L},s}$, whenever $s \geq \eta_\varepsilon(\mathcal{S})$ for $0 < \varepsilon \leq \frac{1}{2}$.

Algorithm 3.2: $\texttt{DGLift}(\mathcal{P}, \mathcal{L}, s, \mathbf{x})$

Input: Lattice $\mathcal{P}$
Lattice $\mathcal{L}$ such that $\mathcal{P} = \pi_{\mathcal{S}}^{\perp}(\mathcal{L})$ for a primitive sublattice $\mathcal{S} \subseteq \mathcal{L}$
Real $s > 0$
Vector $\mathbf{x} \in \mathcal{P}$

Output: Vector $\mathbf{x}' \in \mathcal{L}$ such that $\pi_{\mathcal{S}}^{\perp}(\mathbf{x}') = \mathbf{x}$

1: Let $\mathcal{C}$ be a complement to $\mathcal{S}$
2: Compute the unique $\mathbf{y} \in \mathcal{C}$ such that $\pi_{\mathcal{S}}^{\perp}(\mathbf{y}) = \mathbf{x}$ *// Lifting*
3: Sample $\mathbf{z} \sim D_{\mathcal{S},s,\mathbf{x}-\mathbf{y}}$ *// Sampling*
4: **return** $\mathbf{x}' := \mathbf{z} + \mathbf{y}$

Lemma 3.3.1 (Complexity and distribution of $\texttt{DGLift}$)**.** *Let $\mathcal{L} \subseteq \mathbb{R}^n$ be a lattice, $\mathcal{S} \subseteq \mathcal{L}$ a primitive sublattice, and define $\mathcal{P} := \pi_{\mathcal{S}}^{\perp}(\mathcal{L})$. For a real $s > 0$, let $\mathcal{A}$ be a randomized algorithm that, given $\mathbf{c} \in \mathrm{span}(\mathcal{S})$, returns a sample from $D_{\mathcal{S},s,\mathbf{c}}$. Then $\texttt{DGLift}(\mathcal{P}, \mathcal{L}, s, \cdot)$ (Algorithm 3.2) is a randomized algorithm that, given a vector $\mathbf{x} \in \mathcal{P}$, outputs a vector $\mathbf{x}'$ in $\mathcal{L}$. It uses one query to $\mathcal{A}$, and all other operations run in polynomial time.*

Moreover, for any reals $\delta \geq 0$ and $0 < \varepsilon \leq \frac{1}{2}$ satisfying $s \geq \eta_\varepsilon(\mathcal{S})$, if $X_1, \ldots, X_N \in \mathcal{P}$ are conditionally δ-similar to independent samples from $D_{\mathcal{P},s}$, then the distribution of $X'_1, \ldots, X'_N$ for $X'_i := \texttt{DGLift}(\mathcal{P}, \mathcal{L}, s, X_i)$ is conditionally $(\delta + 3\varepsilon)$-similar to independent samples from $D_{\mathcal{L},s}$.

We will invoke the above lemma with $\mathcal{S} = \frac{q}{p_i}\mathbb{Z}^{b_i}$, hence an exact polynomial-time sampler is available whenever $s \geq \frac{q}{p_i}\sqrt{\ln(2b_i + 4)/\pi}$ by Lemma 3.2.11.

Proof. Consider Algorithm 3.2, where we use algorithm $\mathcal{A}$ for the sampling step, and let $\mathcal{C}$ be the complement of $\mathcal{S}$ that it considers. Note that $\pi_{\mathcal{S}}^{\perp}$ induces a bijection from $\mathcal{C}$ to $\mathcal{P}$, and that both directions can be computed in polynomial time. The claim on time and query complexity of Algorithm 3.2 is thus immediate.

For correctness, we remark that any output vector $\mathbf{x}' = \mathbf{z} + \mathbf{y}$ belongs to $\mathcal{L}$, since $\mathbf{z} \in \mathcal{S} \subseteq \mathcal{L}$ and $\mathbf{y} \in \mathcal{C} \subseteq \mathcal{L}$. Furthermore, $\mathbf{x}' = \pi_{\mathcal{S}}(\mathbf{x}') + \pi_{\mathcal{S}}^{\perp}(\mathbf{x}')$ with $\pi_{\mathcal{S}}(\mathbf{x}') = \mathbf{z} + \pi_{\mathcal{S}}(\mathbf{y})$ and $\pi_{\mathcal{S}}^{\perp}(\mathbf{x}') = \pi_{\mathcal{S}}^{\perp}(\mathbf{y}) = \mathbf{x}$, so the output is as desired.

For the remainder of the proof, let $f(\mathbf{x}) \coloneqq \texttt{DGLift}(\mathcal{P}, \mathcal{L}, s, \mathbf{x})$. We remark that $\mathcal{S} \oplus \mathcal{C} = \mathcal{L}$ implies that any $\mathbf{v} \in \mathcal{L}$ can be uniquely written as $\mathbf{v} = \mathbf{v}_{\mathcal{S}} + \mathbf{v}_{\mathcal{C}}$ for $\mathbf{v}_{\mathcal{S}} \in \mathcal{S}$ and $\mathbf{v}_{\mathcal{C}} \in \mathcal{C}$.

We first observe that for all $\mathbf{x} \in \mathcal{P}$ and $\mathbf{x}' \in \mathcal{L}$, the probability that DGLift on input $\mathbf{x}$ outputs $\mathbf{x}'$ is

$$\begin{aligned}\Pr[f(\mathbf{x}) = \mathbf{x}'] &= \Pr_{Z \sim D_{\mathcal{S},s,\mathbf{x}-\mathbf{y}(\mathbf{x})}}[Z = \mathbf{x}' - \mathbf{y}(\mathbf{x})] \\ &= \begin{cases} \frac{\rho_s(\pi_{\mathcal{S}}(\mathbf{x}'))}{\rho_s(\mathcal{S}+\pi_{\mathcal{S}}(\mathbf{x}'_{\mathcal{C}}))} & \text{if } \mathbf{x} = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}') \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{3.6}$$

where $\mathbf{y}(\mathbf{x})$ denotes the unique $\mathbf{y} \in \mathcal{C}$ such that $\pi_{\mathcal{S}}^{\perp}(\mathbf{y}) = \mathbf{x}$. Indeed, note that $\mathbf{x}' - \mathbf{y}(\mathbf{x}) \in \mathcal{S}$ if and only if $\mathbf{x}'_{\mathcal{C}} = \mathbf{y}(\mathbf{x})$ if and only if $\mathbf{x} = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}'_{\mathcal{C}})$ if and only if $\mathbf{x} = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}')$. Therefore, $\Pr_{Z \sim D_{\mathcal{S},s,\mathbf{x}-\mathbf{y}(\mathbf{x})}}[Z = \mathbf{x}' - \mathbf{y}(\mathbf{x})] = \frac{\rho_s(\mathbf{x}'-\mathbf{x})}{\rho_s(\mathcal{S}-\mathbf{x}+\mathbf{y}(\mathbf{x}))} = \frac{\rho_s(\pi_{\mathcal{S}}(\mathbf{x}'))}{\rho_s(\mathcal{S}-\mathbf{x}+\mathbf{y}(\mathbf{x}))}$ if $\mathbf{x} = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}')$ and 0 otherwise. Since $\mathbf{x} = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}')$ implies $\mathbf{y}(\mathbf{x}) = \mathbf{x}'_{\mathcal{C}}$, and thus $\mathbf{y}(\mathbf{x}) - \mathbf{x} = \pi_{\mathcal{S}}(\mathbf{y}(\mathbf{x})) = \pi_{\mathcal{S}}(\mathbf{x}'_{\mathcal{C}})$, Equation (3.6) follows.

Next, we prove the following intermediate claim.

Claim 3.3.2. *For $s \geq \eta_\varepsilon(\mathcal{S})$, we have $\rho_s(\mathcal{P}) \cdot \rho_s(\mathcal{S}) \in \left[1, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \rho_s(\mathcal{L})$.*

Proof of the claim. By Equation (3.6), if the input is a random variable $X \sim D$ for some distribution D on $\mathcal{P}$, then for all $\mathbf{x}' \in \mathcal{L}$, we have

$$\begin{aligned}\Pr_{X,Z}[f(X) = \mathbf{x}'] &= \Pr_{X,Z}[X = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}')] \cdot \Pr_{X,Z}[f(X) = \mathbf{x}' \mid X = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}')] \\ &= \Pr_{X \sim D}[X = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}')] \cdot \frac{\rho_s(\pi_{\mathcal{S}}(\mathbf{x}'))}{\rho_s(\mathcal{S} + \pi_{\mathcal{S}}(\mathbf{x}'_{\mathcal{C}}))} \\ &\in \left[1, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \Pr_{X \sim D}[X = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}')] \cdot \frac{\rho_s(\pi_{\mathcal{S}}(\mathbf{x}'))}{\rho_s(\mathcal{S})}. \qquad \text{(Lemma 3.2.9)}\end{aligned}$$

In particular, if D is $D_{\mathcal{P},s}$, then we have (for all $\mathbf{x}' \in \mathcal{L}$) that $\Pr[f(X) = \mathbf{x}'] \in \left[1, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \frac{\rho_s(\mathbf{x}')}{\rho_s(\mathcal{P})\rho_s(\mathcal{S})}$ since $\pi_{\mathcal{S}}^{\perp}(\mathbf{x}') + \pi_{\mathcal{S}}(\mathbf{x}') = \mathbf{x}'$. Finally, summing both sides over all $\mathbf{x}' \in \mathcal{L}$ yields $1 \in \left[1, \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \frac{\rho_s(\mathcal{L})}{\rho_s(\mathcal{P})\rho_s(\mathcal{S})}$, which proves the claim. □

We now proceed with the main proof. Suppose that the input consists of N random variables conditionally δ-similar to independent samples from $D_{\mathcal{P},s}$. By Equation (3.6), we know that for all $\mathbf{x}' \in \mathcal{L}^N$ and any $I \subseteq [N]$,

$$\begin{aligned}
&\Pr_{(X_1,Z_1),\ldots,(X_N,Z_N)}[\forall j \in I, f(X_j) = \mathbf{x}'_j] \\
&= \Pr_{X_1,\ldots,X_N}[\forall j \in I, X_j = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}'_j)] \cdot \Pr_{Z_1,\ldots,Z_N}[\forall j \in I, f(\pi_{\mathcal{S}}^{\perp}(\mathbf{x}'_j)) = \mathbf{x}'_j] \\
&= \Pr_{X_1,\ldots,X_N}[\forall j \in I, X_j = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}'_j)] \cdot \prod_{j\in I} \frac{\rho_s(\pi_{\mathcal{S}}(\mathbf{x}'_j))}{\rho_s(\mathcal{S} + \pi_{\mathcal{S}}(\mathbf{x}'_{j,\mathcal{C}}))} \qquad (3.7)
\end{aligned}$$

since the Z_j are independent when the values of the X_j are fixed. (Here, we write $\mathbf{x}'_{j,\mathcal{C}}$ for the unique $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{x}'_j = \mathbf{s} + \mathbf{c}$ for $(\mathbf{s}, \mathbf{c}) \in \mathcal{S} \times \mathcal{C}$.) To conclude the proof, take any $i \in [N]$ and $\mathbf{x}' \in \mathcal{L}^N$. Then

$$\begin{aligned}
&\Pr_{X_1,\ldots,X_N}[f(X_i) = \mathbf{x}'_i \mid \forall j \in [N] \setminus \{i\}, f(X_j) = \mathbf{x}'_j] \\
&= \frac{\Pr_{X_1,\ldots,X_N}[\forall j \in [N], f(X_j) = \mathbf{x}'_j]}{\Pr_{X_1,\ldots,X_N}[\forall j \in [N] \setminus \{i\}, f(X_j) = \mathbf{x}'_j]} && \text{(def. conditional probability)} \\
&= \frac{\Pr_{X_1,\ldots,X_N}[\forall j \in [N], X_j = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}'_j)]}{\Pr_{X_1,\ldots,X_N}[\forall j \in [N] \setminus \{i\}, X_j = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}'_j)]} \cdot \frac{\rho_s(\pi_{\mathcal{S}}(\mathbf{x}'_i))}{\rho_s(\mathcal{S} + \pi_{\mathcal{S}}(\mathbf{x}'_{i,\mathcal{C}}))} && \text{(Equation (3.7))} \\
&= \Pr_{X_1,\ldots,X_N}[X_i = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}'_i) \mid \forall j \in [N] \setminus \{i\}, X_j = \pi_{\mathcal{S}}^{\perp}(\mathbf{x}'_j)] \cdot \frac{\rho_s(\pi_{\mathcal{S}}(\mathbf{x}'_i))}{\rho_s(\mathcal{S} + \pi_{\mathcal{S}}(\mathbf{x}'_{i,\mathcal{C}}))} \\
&= e^{\pm\delta} \cdot \frac{\rho_s(\pi_{\mathcal{S}}^{\perp}(\mathbf{x}'_i))}{\rho_s(\mathcal{P})} \cdot \frac{\rho_s(\pi_{\mathcal{S}}(\mathbf{x}'_i))}{\rho_s(\mathcal{S} + \pi_{\mathcal{S}}(\mathbf{x}'_{i,\mathcal{C}}))} && \text{(by assumption)} \\
&= e^{\pm\delta} \cdot \frac{\rho_s(\mathbf{x}'_i)}{\rho_s(\mathcal{P})\rho_s(\mathcal{S} + \pi_{\mathcal{S}}(\mathbf{x}'_{i,\mathcal{C}}))} \\
&\in \left[e^{-\delta}, e^{\delta} \cdot \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \frac{\rho_s(\mathbf{x}'_i)}{\rho_s(\mathcal{P})\rho_s(\mathcal{S})} && \text{(by Lemma 3.2.9)} \\
&\subseteq \left[e^{-\delta} \cdot \frac{1-\varepsilon}{1+\varepsilon}, e^{\delta} \cdot \frac{1+\varepsilon}{1-\varepsilon}\right] \cdot \frac{\rho_s(\mathbf{x}'_i)}{\rho_s(\mathcal{L})}. && \text{(by the claim)}
\end{aligned}$$

Since $\left[\frac{1-\varepsilon}{1+\varepsilon}, \frac{1+\varepsilon}{1-\varepsilon}\right] \subseteq [e^{-3\varepsilon}, e^{3\varepsilon}]$ for all $0 < \varepsilon \leq \frac{1}{2}$, the lemma follows. □

3.3.2 Combining to a sublattice

We now show that, given many independent discrete Gaussian samples from a lattice $\mathcal{L}'$, we can construct many vectors in a *full-rank* sublattice $\mathcal{L} \subseteq \mathcal{L}'$ whose distributions are (conditionally) similar to a discrete Gaussian over $\mathcal{L}$.

By the convolution lemma [Pei08; MP13] (more precisely, by Lemma 3.2.12), the difference of two independent samples from $D_{\mathcal{L}',s}$ follows a distribution similar to $D_{\mathcal{L}',\sqrt{2}s}$. If we condition on the result being in the sublattice $\mathcal{L}$, then this distribution can in fact be shown to be similar to $D_{\mathcal{L},\sqrt{2}s}$.

Motivated by this fact, we consider an algorithm (Algorithm 3.3) that first buckets its input vectors in $\mathcal{L}'$ with respect to their cosets modulo the sublattice $\mathcal{L}$, and then (carefully) combines pairs of vectors in the same cosets to obtain vectors in $\mathcal{L}$.[4] If we start with at least $3|\mathcal{L}'/\mathcal{L}|$ vectors from $\mathcal{L}'$, then the number of output vectors is only a constant factor smaller, as shown by Lemma 3.3.3. Furthermore, if the input vectors are conditionally similar to independent samples from $D_{\mathcal{L}',s}$, then the output vectors are conditionally similar to independent samples from $D_{\mathcal{L},\sqrt{2}s}$, as shown by Lemma 3.3.4.

Lemma 3.3.3 (Correctness of Algorithm 3.3)**.** *Algorithm 3.3 is correct, that is, given two full-rank lattices $\mathcal{L} \subseteq \mathcal{L}'$ in $\mathbb{R}^d$ and a list of $N \geq 3|\mathcal{L}'/\mathcal{L}|$ vectors in $\mathcal{L}'$, it returns a list of $\lfloor N/3 \rfloor$ vectors in $\mathcal{L}$.*

Proof. By construction, each element of L_{out} is of the form $\mathbf{y} = \mathbf{x} - \mathbf{x}'$ for $\mathbf{x} = \mathbf{x}' \bmod \mathcal{L}$, so $\mathbf{y} \in \mathcal{L}$. It thus remains to show that the output list L_{out} always consists of $\lfloor N/3 \rfloor$ elements. Suppose, for contradiction, that the algorithm returns a list of size ℓ for some $\ell \in \{0, \ldots, \lfloor N/3 \rfloor - 1\}$. Then 2ℓ elements in L are used as part of the output, so there must be $N - 2\ell$ unused list elements. (Here, we talk about list elements instead of vectors, since two list elements may correspond to the same vector.) Their corresponding cosets must be distinct (otherwise the algorithm would have been able to find more than ℓ output elements), so $N - 2\ell \leq |\mathcal{L}'/\mathcal{L}| \leq N/3$. It follows that $\lfloor N/3 \rfloor \leq N/3 \leq \ell$, which is a contradiction. Hence, the algorithm always succeeds to construct $\lfloor N/3 \rfloor$ output vectors. □

[4] Algorithm 3.3 is just a reformulation of [ALS21, Algorithm 2] with a different number of output vectors (and output vectors of the form $\mathbf{x} - \mathbf{x}'$ instead of $\mathbf{x} + \mathbf{x}'$).

Algorithm 3.3: `BucketAndCombine`$(\mathcal{L}', \mathcal{L}, L)$

Input: Full-rank lattices $\mathcal{L} \subseteq \mathcal{L}'$ in $\mathbb{R}^d$
List L of N vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{L}'$, where $N \geq 3|\mathcal{L}'/\mathcal{L}|$

Output: List L_{out} of $\lfloor N/3 \rfloor$ vectors in $\mathcal{L}$

Initialize empty lists $B(\mathbf{c})$ for each coset $\mathbf{c} \in \mathcal{L}'/\mathcal{L}$
for $i = 1, \ldots, N$ **do** *// Bucketing*
    Let $\mathbf{c}_i := \mathbf{x}_i \bmod \mathcal{L}$
    Append $\mathbf{x}_i$ to $B(\mathbf{c}_i)$
Initialize an empty list L_{out}
for $i = 1, \ldots, N$ **do** *// Combining*
    if $B(\mathbf{c}_i)$ contains at least two elements and $|L_{\text{out}}| < \lfloor N/3 \rfloor$ **then**
        Let $\mathbf{x}, \mathbf{x}'$ be the first two elements in $B(\mathbf{c}_i)$
        Append $\mathbf{y} := \mathbf{x} - \mathbf{x}'$ to L_{out}
        Remove $\mathbf{x}$ and $\mathbf{x}'$ from $B(\mathbf{c}_i)$
return L_{out}

The following is a variant of [ALS21, Lemma 4.5], suitable for our purposes.

Lemma 3.3.4 (Output distribution)**.** *Let $\mathcal{L} \subseteq \mathcal{L}'$ be full-rank lattices in $\mathbb{R}^d$. Let $\delta \geq 0$, $0 < \varepsilon \leq \frac{1}{2}$, and $s \geq \sqrt{2}\eta_\varepsilon(\mathcal{L}')$ be reals. If the input list consists of $N \geq 3|\mathcal{L}'/\mathcal{L}|$ random variables on $\mathcal{L}'$ that are conditionally δ-similar to independent samples from $D_{\mathcal{L}',s}$, then Algorithm 3.3 returns a list of $\lfloor N/3 \rfloor$ vectors in $\mathcal{L}$ that are conditionally $(4\delta + 3\varepsilon)$-similar to independent samples from $D_{\mathcal{L},\sqrt{2}s}$.*

Our proof makes use of the following fact: given a sample X from a distribution similar to $D_{\mathcal{L}',s}$, if we condition on $X = \mathbf{c} \bmod \mathcal{L}$ for some $\mathcal{L} \subseteq \mathcal{L}'$ and $\mathbf{c} \in \mathcal{L}'$, then this distribution is similar to $D_{\mathcal{L}+\mathbf{c},s}$. More generally, conditioning on cosets preserves conditional similarity.

Lemma 3.3.5 (Conditioning on cosets)**.** *Let $\mathcal{L} \subseteq \mathcal{L}'$ be full-rank lattices in $\mathbb{R}^d$. Let N be a positive integer, and let $\delta \geq 0$ and $s > 0$ be reals. Suppose that $X_1, \ldots, X_N \in \mathcal{L}'$ are discrete random variables that are conditionally δ-similar to independent samples from $D_{\mathcal{L}',s}$. Then, for all $i \in [N]$, all $\mathbf{c} \in (\mathcal{L}'/\mathcal{L})^N$, and all $\mathbf{x} \in (\mathcal{L}')^N$ satisfying $\mathbf{x}_j = \mathbf{c}_j \bmod \mathcal{L}$ for all $j \in [N]$,*

$$\Pr[X_i = \mathbf{x}_i \mid X_{-i} = \mathbf{x}_{-i} \text{ and } \forall j \in [N], X_j = \mathbf{c}_j \bmod \mathcal{L}] = e^{\pm 2\delta} \cdot \frac{\rho_s(\mathbf{x}_i)}{\rho_s(\mathcal{L} + \mathbf{c}_i)}.$$

Proof. Suppose that $X = (X_1, \ldots, X_N)$ consists of N random variables on $\mathcal{L}'$ that are conditionally δ-similar to independent samples from $D_{\mathcal{L}',s}$. Then, by definition, we have for all $i \in [N]$ and all $\mathbf{x} \in (\mathcal{L}')^N$ that

$$\Pr[X_i = \mathbf{x}_i \mid X_{-i} = \mathbf{x}_{-i}] = e^{\pm\delta} \cdot \frac{\rho_s(\mathbf{x}_i)}{\rho_s(\mathcal{L}')}. \tag{3.8}$$

Consider arbitrary $i \in [N]$, $\mathbf{c} \in (\mathcal{L}'/\mathcal{L})^N$, and $\mathbf{x} \in (\mathcal{L}')^N$ satisfying $\mathbf{x}_j = \mathbf{c}_j \bmod \mathcal{L}$ for all $j \in [N]$. Then, by definition of conditional probability,

$$\begin{aligned}
&\Pr[X_i = \mathbf{x}_i \mid X_{-i} = \mathbf{x}_{-i} \text{ and } \forall j \in [N], X_j = \mathbf{c}_j \bmod \mathcal{L}] \\
&= \frac{\Pr[X_i = \mathbf{x}_i \text{ and } X_{-i} = \mathbf{x}_{-i} \text{ and } \forall j \in [N], X_j = \mathbf{c}_j \bmod \mathcal{L}]}{\Pr[X_{-i} = \mathbf{x}_{-i} \text{ and } \forall j \in [N], X_j = \mathbf{c}_j \bmod \mathcal{L}]} \\
&= \frac{\Pr[X_i = \mathbf{x}_i \text{ and } X_{-i} = \mathbf{x}_{-i}]}{\Pr[X_{-i} = \mathbf{x}_{-i} \text{ and } X_i = \mathbf{c}_i \bmod \mathcal{L}]} && (\text{as } \mathbf{x}_j = \mathbf{c}_j \bmod \mathcal{L}\ \forall j \in [N]) \\
&= \frac{\Pr[X_i = \mathbf{x}_i \mid X_{-i} = \mathbf{x}_{-i}]}{\sum_{\mathbf{v} \in \mathcal{L} + \mathbf{c}_i} \Pr[X_i = \mathbf{v} \mid X_{-i} = \mathbf{x}_{-i}]} \\
&= e^{\pm 2\delta} \frac{\rho_s(\mathbf{x}_i)/\rho_s(\mathcal{L}')}{\rho_s(\mathcal{L} + \mathbf{c}_i)/\rho_s(\mathcal{L}')}
\end{aligned}$$

where we apply Equation (3.8) twice (to both the numerator and denominator) to obtain the last line. The conclusion then immediately follows. □

Proof of Lemma 3.3.4. Consider Algorithm 3.3 on input a list of $N \geq 3|\mathcal{L}'/\mathcal{L}|$ random variables on $\mathcal{L}'$ that are conditionally δ-similar to independent samples from $D_{\mathcal{L}',s}$. By Lemma 3.3.3, it returns a list of $\lfloor N/3 \rfloor$ vectors in $\mathcal{L}$. Let $Y_1, \ldots, Y_M$ be the random variables corresponding to the vectors in the output list (in order), where $M := \lfloor N/3 \rfloor$. We want to show that, for all $j \in [M]$ and $\mathbf{y} \in \mathcal{L}^M$,

$$\Pr_{X_1,\ldots,X_N}[Y_j = \mathbf{y}_j \mid Y_{-j} = \mathbf{y}_{-j}] = e^{\pm(4\delta+3\varepsilon)} D_{\mathcal{L},\sqrt{2}s}(\mathbf{y}_j).$$

By Lemma 3.2.5, it suffices to show that for all $\mathbf{c}_1, \ldots, \mathbf{c}_N \in \mathcal{L}'/\mathcal{L}$ such that $\Pr_{X_1,\ldots,X_N}[\forall i \in [N], X_i = \mathbf{c}_i \bmod \mathcal{L}] > 0$, we have, for all $j \in [M]$, $\mathbf{y} \in \mathcal{L}^M$,

$$\begin{aligned}&\Pr_{X_1,\ldots,X_N}[Y_j = \mathbf{y}_j \mid Y_{-j} = \mathbf{y}_{-j} \text{ and } \forall i \in [N], X_i = \mathbf{c}_i \bmod \mathcal{L}] \\ &= e^{\pm(4\delta+3\varepsilon)} D_{\mathcal{L},\sqrt{2}s}(\mathbf{y}_j). \end{aligned} \tag{3.9}$$

Consider any $\mathbf{c}_1, \ldots, \mathbf{c}_N \in \mathcal{L}'/\mathcal{L}$ such that $\Pr_{X_1,\ldots,X_N}[\forall i \in [N], X_i = \mathbf{c}_i \bmod \mathcal{L}] > 0$. Note that the output (in particular, the way the vectors are paired) is entirely determined by the cosets $(\mathbf{c}_1, \ldots, \mathbf{c}_N)$ for $\mathbf{c}_i := \mathbf{x}_i \bmod \mathcal{L}$. In particular, for any permutation $\pi\colon [N] \to [N]$ such that $(\mathbf{x}'_1, \ldots, \mathbf{x}'_N) := (\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(N)})$ satisfies $\mathbf{y}_j = \mathbf{x}'_{2j-1} - \mathbf{x}'_{2j}$ for all $j \in [M]$, we have that $\mathbf{x}'_1, \ldots, \mathbf{x}'_{2M}$ is entirely determined by the cosets $(\mathbf{c}_1, \ldots, \mathbf{c}_N)$. (The order of the vectors $\mathbf{x}'_{2M+1}, \ldots, \mathbf{x}'_N$ does not affect the algorithm's output.) Without loss of generality, we therefore redefine $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ as $(\mathbf{x}'_1, \ldots, \mathbf{x}'_N)$ for such a permutation (allowing us to write $\mathbf{y}_j = \mathbf{x}_{2j-1} - \mathbf{x}_{2j}$ for all $j \in [M]$).

Let D be the distribution of the input variables $X = (X_1, \ldots, X_N)$ *conditional on* $X_i = \mathbf{c}_i \bmod \mathcal{L}$ for all $i \in [N]$. By the conditional similarity assumption and by Lemma 3.3.5, for all $i \in [N]$ and all $\mathbf{x} \in (\mathcal{L}')^N$ satisfying $\mathbf{x}_j = \mathbf{c}_j \bmod \mathcal{L}$ for all $j \in [N]$, we have

$$\Pr_{X \sim D}[X_i = \mathbf{x}_i \mid X_{-i} = \mathbf{x}_{-i}] = e^{\pm 2\delta} D_{\mathcal{L}+\mathbf{c}_i,s}(\mathbf{x}_i) \tag{3.10}$$

which we repeatedly use below.

To show that Equation (3.9) holds for all $j \in [M]$, $\mathbf{y} \in \mathcal{L}^M$, and thus to finish the proof, it suffices (by another application of Lemma 3.2.5) to show that for all $j \in [M]$, $\mathbf{y} \in \mathcal{L}^M$, and all $\mathbf{v} \in (\mathcal{L}')^{N-2}$ satisfying $\Pr[X_{-\{2j-1,2j\}} = \mathbf{v} | Y_{-j} = \mathbf{y}_j] > 0$, we have

$$\Pr_{X \sim D}[Y_j = \mathbf{y}_j \mid Y_{-j} = \mathbf{y}_{-j} \text{ and } X_{-\{2j-1,2j\}} = \mathbf{v}] = e^{\pm(4\delta+3\varepsilon)} D_{\mathcal{L},\sqrt{2}s}(\mathbf{y}_j).$$

So take any $j \in [M]$ and $\mathbf{y} \in \mathcal{L}^M$, and any such $\mathbf{v} \in (\mathcal{L}')^{N-2}$. Since $X_{-\{2j-1,-2j\}}$ determines all the entries of Y_{-j}, $\Pr[X_{-\{2j-1,2j\}} = \mathbf{v} | Y_{-j} = \mathbf{y}_j] > 0$ implies $\Pr[Y_{-j} = \mathbf{y}_{-j} \mid X_{-\{2j-1,2j\}} = \mathbf{v}] = 1$. Hence,

$$\Pr_{X\sim D}[Y_j = \mathbf{y}_j \mid Y_{-j} = \mathbf{y}_{-j} \text{ and } X_{-\{2j-1,2j\}} = \mathbf{v}] = \Pr_{X\sim D}[Y_j = \mathbf{y}_j \mid X_{-\{2j-1,2j\}} = \mathbf{v}].$$

Writing $\mathbf{c} \coloneqq X_{2j-1} \bmod \mathcal{L} = X_{2j} \bmod \mathcal{L}$, we have

$$\begin{aligned}
&\Pr_{X\sim D}[Y_j = \mathbf{y}_j \mid X_{-\{2j-1,2j\}} = \mathbf{v}] \\
&= \Pr_{X\sim D}[X_{2j-1} - X_{2j} = \mathbf{y}_j \mid X_{-\{2j-1,2j\}} = \mathbf{v}] \\
&= \sum_{\mathbf{x}\in\mathcal{L}+\mathbf{c}} \Pr_{X\sim D}[X_{2j-1} = \mathbf{x} \text{ and } X_{2j} = \mathbf{x} - \mathbf{y}_j \mid X_{-\{2j-1,2j\}} = \mathbf{v}] \\
&= e^{\pm 2\delta} \cdot \sum_{\mathbf{x}\in\mathcal{L}+\mathbf{c}} D_{\mathcal{L}+\mathbf{c},s}(\mathbf{x}) \cdot \Pr_{X\sim D}[X_{2j} = \mathbf{x} - \mathbf{y}_j \mid X_{-\{2j-1,2j\}} = \mathbf{v}]
\end{aligned}$$

by definition of conditional probability and Equation (3.10) (for $i \coloneqq 2j-1$). Then,

$$\begin{aligned}
&\Pr_{X\sim D}[X_{2j} = \mathbf{x} - \mathbf{y}_j \mid X_{-\{2j-1,2j\}} = \mathbf{v}] \\
&= \sum_{\mathbf{z}\in\mathcal{L}+\mathbf{c}} \Pr_{X\sim D}[X_{2j-1} = \mathbf{z} \text{ and } X_{2j} = \mathbf{x} - \mathbf{y}_j \mid X_{-\{2j-1,2j\}} = \mathbf{v}] \\
&= e^{\pm 2\delta} \cdot D_{\mathcal{L}+\mathbf{c},s}(\mathbf{x} - \mathbf{y}_j) \cdot \sum_{\mathbf{z}\in\mathcal{L}+\mathbf{c}} \Pr_{X\sim D}[X_{2j-1} = \mathbf{z} \mid X_{-\{2j-1,2j\}} = \mathbf{v}] \\
&= e^{\pm 2\delta} \cdot D_{\mathcal{L}+\mathbf{c},s}(\mathbf{x} - \mathbf{y}_j)
\end{aligned}$$

using the definition of conditional probability and Equation (3.10). We complete the proof by noting that Lemma 3.2.12 and $\frac{1+\varepsilon}{1-\varepsilon} \le e^{3\varepsilon}$ for $\varepsilon \le \frac{1}{2}$ imply

$$\begin{aligned}
\Pr_{X\sim D}[Y_j = \mathbf{y}_j \mid X_{-\{2j-1,2j\}} = \mathbf{v}] &= e^{\pm 4\delta} \cdot \sum_{\mathbf{x}\in\mathcal{L}+\mathbf{c}} D_{\mathcal{L}+\mathbf{c},s}(\mathbf{x}) \cdot D_{\mathcal{L}+\mathbf{c},s}(\mathbf{x} - \mathbf{y}_j) \\
&= e^{\pm 4\delta} \cdot \Pr_{(X_1,X_2)\sim(D_{\mathcal{L}+\mathbf{c},s})^2}[X_1 - X_2 = \mathbf{y}_j] \\
&= e^{\pm(4\delta+3\varepsilon)} \cdot D_{\mathcal{L},\sqrt{2}s}(\mathbf{y}_j).
\end{aligned}$$

□

3.3.3 Full algorithm: Wagner as a Gaussian sampler

We are now ready to present our Gaussian sampler, laid out as Algorithm 3.4.

Algorithm 3.4: Wagner-style Gaussian sampler

Input: Integers n, m, q
Full-rank matrix $\mathbf{A} = [\mathbf{A}' \mid \mathbf{I}_n] \in \mathbb{Z}_q^{n \times m}$
Integer parameters $N, r, (p_i)_{i=1}^r, (b_i)_{i=1}^r$ with $\sum_{i=1}^r b_i = n$
Real parameter $s_0 > 0$

Output: List of vectors in $\Lambda_q^\perp(\mathbf{A})$

Let $\Lambda_0 := \mathbb{Z}^{m-n}$
Initialize a list L_0 of $3^r N$ independent samples from D_{Λ_0, s_0}
for $i = 1, \ldots, r$ **do**
    Let Λ_i be as defined in Equation (3.4)
    Let $\Lambda_i' = \mathcal{L}(\mathbf{B}_i')$ for $\mathbf{B}_i'$ as defined in Equation (3.5)
    $L_{i-1}' := \emptyset$
    for $\mathbf{x} \in L_{i-1}$ **do** *// Lift*
        Sample $\mathbf{x}' \sim \texttt{DGLift}(\Lambda_{i-1}, \Lambda_i', s_{i-1}, \mathbf{x})$ ▷ Algorithm 3.2
        Append $\mathbf{x}'$ to L_{i-1}'
    $L_i = \texttt{BucketAndCombine}(\Lambda_i', \Lambda_i, L_{i-1}')$ ▷ Algorithm 3.3 *// Combine*
    $s_i := \sqrt{2} s_{i-1}$
return L_r

Remark 3.3.6. In our applications of Algorithm 3.4, we consider input parameter s_0 satisfying $s_0 \geq \sqrt{\ln(2(m-n)+4)/\pi}$ and $\sqrt{2^{i-1}} s_0 \geq \frac{q}{p_i}\sqrt{\ln(2b_i+4)/\pi}$ for all $i \in [r]$. This allows us to use the exact sampler from Lemma 3.2.11 to sample from D_{Λ_0, s_0} (in the initialization) and from $D_{\frac{q}{p_i}\mathbb{Z}^{b_i}, \sqrt{2^{i-1}} s_0}$ (in iterations $i = 1, \ldots, r$).

Theorem 3.3.7 (Correctness of one iteration). *Let $\delta \geq 0$ and $0 < \varepsilon \leq \frac{1}{2}$ be reals. For $i \in [r]$, consider iteration i of Algorithm 3.4 with well-defined parameters. If*

$$s_{i-1} \geq \max\left\{\eta_\varepsilon\left(\tfrac{q}{p_i}\mathbb{Z}^{b_i}\right), \sqrt{2}\eta_\varepsilon(\Lambda_i'), \tfrac{q}{p_i}\sqrt{\ln(2b_i+4)/\pi}\right\}$$

and L_{i-1} consists of $|L_{i-1}| \geq 3p_i^{b_i}$ vectors in Λ_{i-1} that are conditionally δ-similar to independent samples from $D_{\Lambda_{i-1},s_{i-1}}$, then L_i consists of $\lfloor |L_{i-1}|/3 \rfloor$ vectors in Λ_i that are conditionally $(4\delta+15\varepsilon)$-similar to independent samples from $D_{\Lambda_i,\sqrt{2}s_{i-1}}$.

Proof. Let $\mathcal{S}$ be the embedding of $\frac{q}{p_i}\mathbb{Z}^{b_i}$ in $\mathbb{R}^{m-n+n_i}$ and $\mathcal{P}$ the embedding of Λ_{i-1} in $\mathbb{R}^{m-n+n_i}$. Note that $\mathcal{S}$ is a primitive sublattice of Λ_i' and that there is a complement $\mathcal{C}$ to $\mathcal{S}$ such that $\mathcal{P} = \pi_{\mathcal{S}}^{\perp}(\mathcal{C})$. Since $s_{i-1} \geq \eta_\varepsilon(\frac{q}{p_i}\mathbb{Z}^{b_i}) = \eta_\varepsilon(\mathcal{S})$, the application of the algorithm from Lemma 3.3.1 turns the $|L_{i-1}|$ random variables on Λ_{i-1} into $|L_{i-1}|$ random variables on Λ_i' that are conditionally $(\delta + 3\varepsilon)$-similar to $D_{\Lambda_i',s_{i-1}}$. Here, as oracle to sample from $D_{\frac{q}{p_i}\mathbb{Z}^{b_i},s_{i-1},\mathbf{c}}$ (exactly), we use Lemma 3.2.11 to sample from $D_{\mathbb{Z}^{b_i},\frac{p_i}{q}s_{i-1},\frac{p_i}{q}\mathbf{c}}$, and multiply the resulting vector by $\frac{q}{p_i}$. This is justified since we assume $\frac{p_i}{q}s_{i-1} \geq \sqrt{\ln(2b_i+4)/\pi}$.

By Lemma 3.3.4 (where we use that $s_{i-1} \geq \sqrt{2}\eta_\varepsilon(\Lambda_i'))$, the output list L_i of `BucketAndCombine` consists of $\lfloor |L_{i-1}|/3 \rfloor$ vectors in Λ_i that are conditionally δ'-similar to independent samples from $D_{\Lambda_i,\sqrt{2}s_{i-1}}$, where $\delta' = 4(\delta + 3\varepsilon) + 3\varepsilon = 4\delta + 15\varepsilon$. □

Theorem 3.3.8 (Correctness of Algorithm 3.4). *Let $0 < \varepsilon \leq \frac{1}{2}$ be a real, and r an integer. If the input parameters satisfy $N \geq p_i^{b_i}$,*

$$s_0 \geq \sqrt{\ln(2(m-n)+4)/\pi}$$

and

$$\sqrt{2^{i-1}}s_0 \geq \max\left\{\eta_\varepsilon\left(\tfrac{q}{p_i}\mathbb{Z}^{b_i}\right), \sqrt{2}\eta_\varepsilon(\Lambda_i'), \tfrac{q}{p_i}\sqrt{\ln(2b_i+4)/\pi}\right\}$$

for all $i \in [r]$, then Algorithm 3.4 returns a list of size N consisting of vectors that are conditionally $4^r 5\varepsilon$-similar to independent samples from $D_{\Lambda_q^{\perp}(\mathbf{A}),\sqrt{2^r}s_0}$.

Proof. To obtain the list L_0, we use the exact $D_{\mathbb{Z}^{m-n},s_0}$-sampler from Lemma 3.2.11. This is justified, since $s_0 \geq \sqrt{\ln(2(m-n)+4)/\pi}$ by assumption. We show by induction that, for each iteration $i \in \{1,\dots,r\}$, the output list L_i consists of $3^{r-i}N$ vectors in Λ_i that are conditionally δ_i-similar to independent samples from D_{Λ_i,s_i} for $\delta_i := (4^i-1)5\varepsilon$. The theorem then immediately follows, since $(4^r-1)5\varepsilon \leq 4^r 5\varepsilon$.

Base case ($i = 1$). By assumption, $|L_0| = 3^r N \geq 3^r p_1^{b_1} \geq 3p_1^{b_1}$ and $s_0 \geq \max\{\eta_\varepsilon(\frac{q}{p_1}\mathbb{Z}^{b_1}), \sqrt{2}\eta_\varepsilon(\Lambda_1'), \frac{q}{p_1}\sqrt{\ln(2b_1+4)/\pi}\}$. Therefore, Theorem 3.3.7 implies that the output list L_1 of iteration $i = 1$ consists of $3^{r-1}N$ vectors in Λ_1 that are conditionally 15ε-similar to independent samples from D_{Λ_1,s_1}. Since $\delta_1 = 15\varepsilon$, this proves the base case.

Inductive step. Consider any $i \in \{2,\dots,r\}$, and suppose the claim holds for all $1 \leq j \leq i-1$. By the induction hypothesis and assumption, $|L_{i-1}| = 3^{r-(i-1)}N \geq 3^{r-(i-1)}p_i^{b_i} \geq 3p_i^{b_i}$ and $s_{i-1} = \sqrt{2^{i-1}}s_0 \geq \max\{\eta_\varepsilon(\frac{q}{p_i}\mathbb{Z}^{b_i}), \sqrt{2}\eta_\varepsilon(\Lambda_i'), \frac{q}{p_i}\sqrt{\ln(2b_i+4)/\pi}\}$. Therefore, by Theorem 3.3.7, the output list L_i of iteration i consists of $3^{r-i}N$ vectors in Λ_i. By the induction hypothesis the vectors in L_{i-1} are conditionally δ_{i-1}-similar to independent samples from $D_{\Lambda_{i-1},s_{i-1}}$, so by Theorem 3.3.7 the vectors in L_i are conditionally $(4\delta_{i-1}+15\varepsilon)$-similar to independent samples from D_{Λ_i,s_i}. Since $4\delta_{i-1}+15\varepsilon = 4(4^{i-1}-1)5\varepsilon + 15\varepsilon = (4^i-1)5\varepsilon = \delta_i$, the claim follows. □

Remark 3.3.9 (Expected runtime). The use of the exact sampler from Lemma 3.2.11 ensures that all vectors processed by Algorithm 3.4 have expected bitsize at most $\mathrm{poly}(m, r, \log s_0, \log q)$ if $p_i \leq q$ for every $i \in [r]$. This yields the upper bound $(3^r N + \sum_{i=1}^r |L_{i-1}|) \cdot \mathrm{poly}(m, r, \log s_0, \log q) = 3^r N \cdot \mathrm{poly}(m, r, \log s_0, \log q)$ on the expected runtime of Algorithm 3.4.

3.3.4 Putting it all together

We now have all the ingredients to prove the existence of a subexponential-time algorithm for sampling from a distribution conditionally similar to $D_{\Lambda_q^\perp(\mathbf{A}),s}$ for random (full-rank) matrices $\mathbf{A}$. We write the width s of the desired discrete Gaussian distribution as $s = q/f$ for some $f > 1$, and remark that the difficulty of sampling with width q/f increases with f. Below, we demonstrate that subexponential complexity is feasible when q/f and m are above a certain threshold.

We note that $q^{1-n/m} \geq 2^{\Theta(\log n/\log\log n)}$ for the parameters of interest, which tends to infinity as n grows (so the assumption $q^{1-n/m} \geq 6$ is not too restrictive).

Theorem 3.3.10. *For $n \in \mathbb{N}$, let $m \geq n$ be an integer and $q = \mathrm{poly}(n)$ be a prime such that $q^{1-n/m} \geq 6$. Let $f > 1$ and $\varepsilon \leq \frac{1}{m}$ be positive reals such that $\frac{q}{f} \geq \sqrt{\ln(1/\varepsilon)}$. There exists a value of N satisfying*

$$\log_2(N) = \frac{n/2}{\ln(\ln(q)) - \ln(\ln(f) + \frac{1}{2}\ln\ln(1/\varepsilon)) - O(1)} \tag{3.11}$$

such that, if $m \geq n + 2\log_2 N$, then there is a randomized algorithm that, given a uniformly random full-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, with probability at least $1 - 2^{-\tilde{\Omega}(n)}$ returns N vectors in $\Lambda_q^\perp(\mathbf{A})$. The output vectors are conditionally $q^4\varepsilon$-similar to independent samples from $D_{\Lambda_q^\perp(\mathbf{A}),\frac{q}{f}}$. This algorithm has time and memory complexity $N \cdot \mathrm{poly}(m)$.

Proof. We first provide a choice of parameters for Algorithm 3.4 (instantiated with the exact sampler from Lemma 3.2.11) and show that the conditions of Theorem 3.3.8 are satisfied.

Choice of parameters. For $\varepsilon' := \varepsilon/5$, let N be the smallest integer such that

$$\log_2(N/q) \geq \frac{n/2}{\ln(\ln(q)) - \ln(\ln(f) + \frac{1}{2}\ln(\frac{144}{\pi}\ln(3/\varepsilon')) + \frac{1}{2})}.$$

Then N satisfies Equation (3.11) for sufficiently large n. Let $s_0 := \frac{q/f}{\sqrt{2^r}}$, where[5]

$$r := \lfloor 2\log_2(q/f) - \log_2(\tfrac{144}{\pi}\ln(3/\varepsilon'))\rfloor.$$

For $i \in [r]$, define $p_i := \lfloor q/\sqrt{2^i}\rfloor$. For $i \in [r-1]$, define $b_i := \lceil \frac{\log_2(N)}{\log_2(q)-i/2} - 1\rceil$, and define $b_r := n - \sum_{i=1}^{r-1} b_i$ so that $\sum_{i=1}^{r} b_i = n$. Note that p_i, b_i are integers.

To show that these parameters satisfy the conditions of Theorem 3.3.8, we consider sufficiently large n so that $\varepsilon' \leq \frac{1}{6}$ (i.e., $\varepsilon \leq \frac{5}{6}$) and assume $2\log_2(N) \leq m - n$.

[5]For all interesting parameters, we have $r \geq 1$. For the rare setting of parameters for which $r < 1$, we replace r by $r + 10$. (This suffices since, for any valid parameters, the assumption $q/f \geq \sqrt{\ln(1/\varepsilon)}$ ensures that $2\log_2(q/f) - \log_2(144\ln(3/\varepsilon')/\pi) + 10 \geq 1$ when $n \geq 2$.) Note that adding a constant to r does not affect the proof; in particular, Equation (3.12) would still hold as it decreases with r.

Verifying that N is large enough. We claim that $N \geq p_i^{b_i}$ for all $i \in [r]$. For all $i \in [r-1]$, this claim follows immediately from the definition of b_i: $b_i \leq \frac{\log_2(N)}{\log_2(q)-i/2} \leq \frac{\log_2(N)}{\log_2(p_i)}$. So it remains to show that $b_r \leq \frac{\log_2(N)}{\log_2(p_r)}$. By definition,

$$\begin{aligned}\log_2(N/q) &\geq \frac{n/2}{\ln(\ln(q)) - \ln\big(\ln(f) + \frac{1}{2}\ln(\frac{144}{\pi}\ln(3/\varepsilon')) + \frac{1}{2}\big)} \\ &\geq \frac{n/2}{\ln(\log_2(q)) - \ln(\log_2(f) + \frac{1}{2}\log_2(\frac{144}{\pi}\ln(3/\varepsilon')) + \frac{1}{2})} \\ &\geq \frac{n/2}{\ln\left(\frac{2\log_2(q)}{2\log_2(q)-r}\right)} \qquad (3.12)\end{aligned}$$

since $r \geq 2\log_2(q/f) - \log_2(\frac{144}{\pi}\ln(3/\varepsilon')) - 1$. In particular, using known facts of integrals (recall Footnote 3 on page 53) we obtain

$$n \leq 2\log_2(N/q) \cdot \ln\left(\frac{2\log_2(q)}{2\log_2(q)-r}\right) = \int_0^r \frac{2\log_2(N/q)}{2\log_2(q)-x}dx \leq \sum_{i=1}^{r} \frac{2\log_2(N/q)}{2\log_2(q)-i}.$$

Since $\frac{2\log_2(N/q)}{2\log_2(q)-i} \leq \frac{2\log_2(N)}{2\log_2(q)-i} - 1 \leq b_i$, we obtain $n \leq \sum_{i=1}^{r-1} b_i + \frac{2\log_2(N/q)}{2\log_2(q)-r}$, and thus $b_r = n - \sum_{i=1}^{r-1} b_i \leq \frac{2\log_2(N/q)}{2\log_2(q)-r} \leq \frac{2\log_2(N)}{2\log_2(q)-r} \leq \frac{\log_2(N)}{\log_2(p_r)}$, as desired.

Verifying the smoothing conditions. To show that the smoothing conditions in Theorem 3.3.8 are satisfied with probability $1 - 2^{-\tilde{\Omega}(n)}$, it suffices to show that the following holds (for large enough n):

(*I*) $s_0 \geq \sqrt{\ln(2(m-n)+4)/\pi}$ and $\sqrt{2^{i-1}}s_0 \geq \frac{q}{p_i}\sqrt{\ln(2b_i+4)/\pi}$ for all $i \in [r]$.

(*II*) With probability $1 - 2^{-\tilde{\Omega}(n)}$, $\sqrt{2^{i-1}}s_0 \geq \sqrt{2}\max\{\eta_{\varepsilon'/3}(\frac{q}{p_i}\mathbb{Z}^{b_i}), \eta_{\varepsilon'/3}(\Lambda_{i-1})\}$ for all $i \in [r]$.

Indeed, if $\sqrt{2^{i-1}}s_0 \geq \sqrt{2}\max\{\eta_{\varepsilon'/3}(\frac{q}{p_i}\mathbb{Z}^{b_i}), \eta_{\varepsilon'/3}(\Lambda_{i-1})\}$ for all $i \in [r]$, then $\sqrt{2^{i-1}}s_0 \geq \max\{\eta_{\varepsilon'}(\frac{q}{p_i}\mathbb{Z}^{b_i}), \sqrt{2}\eta_{\varepsilon'}(\Lambda_i')\}$ for all $i \in [r]$, because of [EWY23a, Proposition 2] (with the embedding of $\frac{q}{p_i}\mathbb{Z}^{b_i}$ in $\mathbb{R}^{m-n+n_i}$ as sublattice) and because $\eta_{\varepsilon'/3}(\frac{q}{p_i}\mathbb{Z}^{b_i}) \geq \eta_{\varepsilon'}(\frac{q}{p_i}\mathbb{Z}^{b_i})$.

To prove (*I*) and (*II*), we use that our choice of r implies $s_0 \geq \sqrt{\frac{144\ln(3/\varepsilon')}{\pi}}$. Furthermore, we emphasize that $\varepsilon \leq \frac{1}{m}$ implies $\varepsilon' \leq \frac{3}{4m}$, so $\varepsilon' \leq \min\{\frac{3}{4(m-n)}, \frac{3}{4b_1}\}$ and $\varepsilon' \leq \min\{\frac{3}{4(m-n+n_{i-1})}, \frac{3}{4b_i}\}$ for all $i \in \{2,\ldots,r\}$. (Hence, we can apply the results from Section 3.2.3 to upper bound $\eta_{\varepsilon'}(\frac{q}{p_i}\mathbb{Z}^{b_i})$ and $\eta_{\varepsilon'}(\Lambda_i')$.)

Consider any $i \in [r]$. By Lemma 3.2.13 (with dimension b_i and using that $\varepsilon' \le \frac{3}{4b_i}$), we obtain $\sqrt{2}\eta_{\varepsilon'/3}(\frac{q}{p_i}\mathbb{Z}^{b_i}) \le \frac{q}{p_i}\sqrt{\frac{4\ln(3/\varepsilon')}{\pi}} \le \sqrt{2^{i-1}}\sqrt{\frac{32\ln(3/\varepsilon')}{\pi}} \le \sqrt{2^{i-1}}s_0$ (since $p_i = \lfloor q/\sqrt{2^i} \rfloor \ge (q/\sqrt{2^i})/2$ whenever $p_i \ge 2$, and thus $\frac{q}{p_i} \le 2\sqrt{2^i}$). Furthermore, for $i = 1$, we have by Lemma 3.2.13 that $\sqrt{2}\eta_{\varepsilon'/3}(\Lambda_0) = \sqrt{2}\eta_{\varepsilon'/3}(\mathbb{Z}^{m-n}) < \sqrt{\frac{4\ln(3/\varepsilon')}{\pi}} \le s_0$. Since $\varepsilon' \le \frac{3}{4m}$ we have $\frac{3}{\varepsilon'} \ge 2m \ge 2(m-n)+4$ and $\frac{3}{\varepsilon'} \ge 4m \ge 4n \ge 2n+4 \ge 2b_i + 4$ for all $i \in [r]$. Hence, $s_0 \ge \sqrt{\frac{4\ln(3/\varepsilon')}{\pi}}$ implies $s_0 \ge \sqrt{\frac{\ln(3/\varepsilon')}{\pi}} \ge \sqrt{\frac{\ln(2(m-n)+4)}{\pi}}$, and $\sqrt{2^{i-1}}s_0 \ge \frac{q}{p_i}\sqrt{\frac{4\ln(3/\varepsilon')}{\pi}}$ implies $\sqrt{2^{i-1}}s_0 \ge \frac{q}{p_i}\sqrt{\frac{\ln(3/\varepsilon')}{\pi}} \ge \frac{q}{p_i}\sqrt{\frac{\ln(2b_i+4)}{\pi}}$ for all $i \in [r]$. Thus, (I) holds for $n \ge 2$.

Next, we observe that $q^{\frac{n_j}{m-n+n_j}} \le \sqrt{2^j}$ for all $j \in [r]$. Indeed, for all $j \in [r]$, we have $n_j \le jb_j$, so $\frac{n_j}{j}(2\log_2(q) - j) \le b_j(2\log_2(q) - j) \le 2\log_2(N) \le m - n$. Since $\frac{n_j}{j}(2\log_2(q) - j) \le m - n$ if and only if $q^{\frac{n_j}{m-n+n_j}} \le \sqrt{2^j}$, the claim follows. Thus, for each $i \in \{2, \dots, r\}$, Lemma 3.2.16 and the previous claim imply that $\sqrt{2}\eta_{\varepsilon'/3}(\Lambda_{i-1}) < \sqrt{\frac{144\ln(3/\varepsilon')}{\pi}}q^{\frac{n_{i-1}}{m-n+n_{i-1}}} \le \sqrt{\frac{144\ln(3/\varepsilon')}{\pi}}\sqrt{2^{i-1}} \le \sqrt{2^{i-1}}s_0$, except with probability $< 2^{-n_{i-1}}$.

Therefore, the union bound implies that, with probability at least $1 - \sum_{i=1}^{r-1} 2^{-n_i}$, we have $\sqrt{2^{i-1}}s_0 \ge \sqrt{2}\max\{\eta_{\varepsilon'/3}(\frac{q}{p_i}\mathbb{Z}^{b_i}), \eta_{\varepsilon'/3}(\Lambda_{i-1})\}$ for all $i \in [r]$. Note that $1 - \sum_{i=1}^{r-1} 2^{-n_i} \ge 1 - r2^{-b_1} = 1 - 2^{-\tilde{\Omega}(n)}$ since $\log_2(r) = \log_2\log_2(n) + O(1)$, $b_1 = \Omega(\frac{\log_2 N}{\log_2 n})$, and $\log_2(N) = \Omega(\frac{n}{\log_2\log_2(n)})$, so (II) holds as well.[6]

Conclusion of the proof. Theorem 3.3.8 then implies that the output of this algorithm consists of at least N vectors in $\Lambda_q^\perp(\mathbf{A})$ that are conditionally $4^r 5\varepsilon'$-similar to independent samples from $D_{\Lambda_q^\perp(\mathbf{A}),\sqrt{2^r}s_0}$. Since $4^r \le q^4$, we obtain that they are conditionally δ-similar for $\delta = q^4 5\varepsilon' = q^4\varepsilon$.

Finally, the expected runtime of Algorithm 3.4 is $3^r N \cdot \mathrm{poly}(m, r, \log s_0, \log q)$ by Remark 3.3.9. Since $r = O(\log q)$, $s_0 \le q$, and $q = \mathrm{poly}(m)$, it follows that the time and memory complexity are both upper bounded by $N \cdot \mathrm{poly}(m)$. □

[6] It also follows that $\eta_{\varepsilon/4}(\Lambda_q^\perp(\mathbf{A})) < \frac{q}{f}$. Indeed, another application of Lemma 3.2.16 yields $\eta_{\varepsilon/4}(\Lambda_q^\perp(\mathbf{A})) < \sqrt{72\ln(4/\varepsilon)/\pi}q^{n/m}$, except with probability $< 2^{-n}$, but this does not affect the lower bound on the success probability. By the aforementioned fact, we have $q^{n/m} \le \sqrt{2^r}$, so by definition of r we obtain $\eta_{\varepsilon/4}(\Lambda_q^\perp(\mathbf{A})) < \frac{q}{f}$.

3.4 Asymptotic application to SIS variants

We now apply our previous result on Gaussian sampling for SIS lattices to solving various variants of SIS in subexponential time.

3.4.1 Implications for SIS^∞

The following theorem instantiated with $m = n + \omega(n/\log\log n)$ such that[7] $m = \mathrm{poly}(n)$, $q = n^{\Theta(1)}$, $\beta = q/\mathrm{polylog}(n)$, and a sufficiently small $\varepsilon = 1/\mathrm{poly}(n)$ provides a subexponential-time algorithm for $\mathrm{SIS}^\infty_{n,m,q,\beta}$. In fact, our subexponential complexity holds for even smaller norm bounds: for instance, we can achieve $\beta = q/2^{\log(n)^c}$ for any constant $c < 1$.

Theorem 3.4.1. *For $n \in \mathbb{N}$, let $m = n + \omega(n/\ln\ln n)$ be integer and $q = n^{\Theta(1)}$ be prime such that $q^{1-n/m} \geq 6$. Let $f > 1$ and $\varepsilon \leq \frac{1}{mq^4}$ be positive reals such that $\ln(f\sqrt{\ln(1/\varepsilon)}) = O(\ln(n)^c)$ for some $c < 1$. There exists an algorithm that solves $\mathrm{SIS}^\infty_{n,m,q,\beta}$ for $\beta := \frac{q}{f}\sqrt{\ln m}$ in expected time*

$$T = 2^{\frac{n/2}{\ln(\ln(q)) - \ln\left(\ln(f) + \frac{1}{2}\ln\ln(1/\varepsilon)\right) - O(1)}} \cdot \mathrm{poly}(m)$$

with success probability $1 - \frac{1}{\Omega(n)}$.

Proof. Apply Theorem 3.3.10 with input n, m, q, f, ε to the $\mathrm{SIS}^\infty_{n,m,q,\beta}$ instance $\mathbf{A}$. With probability at least $1 - 2^{-\tilde{\Omega}(n)}$, it returns a list of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in $\Lambda_q^\perp(\mathbf{A})$ (so they are solutions to $\mathbf{A}\mathbf{x} = \mathbf{0} \bmod q$) that are conditionally $q^4\varepsilon$-similar to independent samples from $D_{\Lambda_q^\perp(\mathbf{A}),s}$ for $s := \frac{q}{f}$. In particular, it follows that the first vector $\mathbf{x}_1$ follows a distribution D that is $q^4\varepsilon$-similar to $D_{\Lambda_q^\perp(\mathbf{A}),s}$ (recall Remark 3.2.4). By Footnote 6 we have $\frac{q}{f} > \eta_{\varepsilon/4}(\Lambda_q^\perp(\mathbf{A}))$. We note that we may, without loss of generality, assume $\frac{q}{f} > 2\eta_{\varepsilon/4}(\Lambda_q^\perp(\mathbf{A}))$ by replacing the role of the constant 144 in the proof of Theorem 3.3.10 by a larger constant.

[7] Note that one may always decrease m by ignoring SIS variables, hence the condition $m = \mathrm{poly}(n)$ is without loss of generality.

Since D is $q^4\varepsilon$-similar to $D_{\Lambda_q^\perp(\mathbf{A}),s}$, Lemma 3.2.6 (together with the fact that $e^{-x} \geq 1 - x$ for all $x \in \mathbb{R}$) implies

$$\begin{aligned}\Pr[\|\mathbf{x}_1\|_\infty \leq \beta \wedge \mathbf{x}_1 \neq \mathbf{0}] &\geq e^{-q^4\varepsilon} \Pr_{X \sim D_{\Lambda_q^\perp(\mathbf{A}),s}}[\|X\|_\infty \leq \beta \wedge X \neq \mathbf{0}] \\ &\geq \Pr_{X \sim D_{\Lambda_q^\perp(\mathbf{A}),s}}[\|X\|_\infty \leq \beta \wedge X \neq \mathbf{0}] - q^4\varepsilon \\ &\geq \Pr_{X \sim D_{\Lambda_q^\perp(\mathbf{A}),s}}[\|X\|_\infty \leq \beta \wedge X \neq \mathbf{0}] - \frac{1}{m}.\end{aligned}$$

We now show that $\Pr_{X \sim D_{\Lambda_q^\perp(\mathbf{A}),s}}[\|X\|_\infty \leq \beta \wedge X \neq \mathbf{0}] \geq 1 - \frac{2}{m^2} - \frac{1}{2^{m-1}}$, from which it follows that $\mathbf{x}_1$ is a solution to SIS^∞ with probability at least $1 - \frac{2}{m^2} - \frac{1}{2^{m-1}} - \frac{1}{m} \geq 1 - \frac{3}{m} \geq 1 - \frac{3}{n}$ when $n \geq 2$ (where we use that $m \geq n$), thereby proving the theorem.

It remains to prove our claim. We have

$$\Pr_{X \sim D_{\Lambda_q^\perp(\mathbf{A}),s}}[\|X\|_\infty \leq \beta \wedge X \neq \mathbf{0}] \geq \Pr_{X \sim D_{\Lambda_q^\perp(\mathbf{A}),s}}[\|X\|_\infty \leq \beta] - \Pr_{X \sim D_{\Lambda_q^\perp(\mathbf{A}),s}}[X = \mathbf{0}].$$

Since $s = \frac{q}{f} > 2\eta_{\varepsilon/4}(\Lambda_q^\perp(\mathbf{A}))$, Lemma 3.2.10 implies $\Pr_{X \sim D_{\Lambda_q^\perp(\mathbf{A}),s}}[X = \mathbf{0}] \leq \frac{1+\varepsilon/4}{1-\varepsilon/4} \cdot 2^{-m} \leq \frac{1}{2^{m-1}}$ (as $\frac{1+x/4}{1-x/4} \leq 2$ for all $x \in [0,1]$). Also, Lemma 3.2.8 yields

$$\Pr_{X \sim D_{\Lambda_q^\perp(\mathbf{A}),s}}[\|X\|_\infty \leq \beta] > 1 - 2me^{-\pi(\frac{\beta f}{q})^2} \geq 1 - \frac{2}{m^2}$$

where we use that $\beta = \frac{q}{f}\sqrt{\ln m}$. Our claim follows. □

One may remark that our application of Lemma 3.2.8 makes us lose a $\sqrt{\ln m}$ factor on the norm bound to reach a constant success probability per sample. It would be tempting to only aim for a success probability barely greater than $1/N$ instead, however, the proof would then require $\varepsilon \approx 1/N$, which would make the algorithm exponential. This is admittedly a counterintuitive situation, and plausibly a proof artifact.

3.4.2 Implications for ISIS and SIS$^\times$ in ℓ_2-norm

For the same parameters as above, but instead considering the ℓ_2-norm, our Gaussian sampler outputs vectors of norm at most $\beta = q\sqrt{m}/f$ in subexponential time for any $f = \mathrm{polylog}(n)$. However, SIS in the ℓ_2-norm is trivial for such a bound; for example, $(q, 0, \ldots, 0)$ is a valid solution.

Yet, some schemes [ETWY22] have used the inhomogeneous version of SIS (ISIS) for bounds $\beta > q$, which was shown [DEP23] to be equivalent to solving SIS$^\times$, a variant of SIS where the solution must be nonzero modulo q. The work of [DEP23] notes that the problem becomes trivial when $\beta \geq q\sqrt{n/12}$ and proposes a heuristic attack that is better than pure lattice reduction when $\beta > q$; however, it appears to run in exponential time in n for $\beta = q\sqrt{n}/\mathrm{polylog}(n)$. Our Gaussian sampler directly yields a provably subexponential-time algorithm in that regime.

Theorem 3.4.2. *For $n \in \mathbb{N}$, let $m = n + \omega(n/\ln\ln n)$ be integer and $q = n^{\Theta(1)}$ be prime such that $q^{1-n/m} \geq 6$. Let $f > 1$ and $\varepsilon \leq \frac{1}{mq^4}$ be positive reals such that $\ln(f\sqrt{\ln(1/\varepsilon)}) = O(\ln(n)^c)$ for some $c < 1$. There exists an algorithm that solves $\mathrm{ISIS}_{n,m,q,\beta}$ and $\mathrm{SIS}^\times_{n,m,q,\beta}$ for $\beta := \frac{q}{f}\sqrt{m}$ in expected time*

$$T = 2^{\frac{n/2}{\ln(\ln(q)) - \ln\left(\ln(f) + \frac{1}{2}\ln\ln(1/\varepsilon)\right) - O(1)}} \cdot \mathrm{poly}(m)$$

with success probability $1 - \frac{1}{\Omega(n)}$.

The proof is essentially equivalent to that of Theorem 3.4.1, using Lemma 3.2.7 instead of Lemma 3.2.8 to tail-bound the ℓ_2-norm of a discrete Gaussian.

Note again here that one may always choose $m = n(1 + o(1))$ without loss of generality even when the given m is much larger, simply by ignoring some SIS variables. Hence, reaching the norm bound $\beta = q\sqrt{m}/\mathrm{polylog}(n)$ also permits to reach $\beta = q\sqrt{n}/\mathrm{polylog}(n)$.

3.5 Concrete and heuristic application to ML-DSA

Our asymptotic result for SIS^∞ immediately raises the question of its impact on the concrete security of ML-DSA, formerly known as Dilithium. However, the algorithm described above introduces inefficiencies for the sake of provability. There are various details that one would approach differently when aiming to break the problem in practice, some of which we describe below. The following analysis might be too aggressive: it is possible that the resulting algorithm will fail, and our conclusions for ML-DSA should only be read as a rough *underestimation* of the cost of the algorithm as currently presented. That said, it is possible that further tricks and refinements (such as those proposed in [BGJM+20]) could lead to a more practical variant of the attack.

Relaxing independence. If N is the number of buckets for the colliding phase, then our provable algorithm used $3^r N$ many initial samples, losing a factor of 3 on the list size at each iteration, but never re-using a sample twice. If we allow ourselves to re-use a sample in several combinations, then $3N$ samples will be enough to maintain the list size throughout the algorithm. In this case, there will be an average of 3 samples per bucket $(\mathbf{x}, \mathbf{y}, \mathbf{z})$, from which we can build 3 different pairs to be subtracted: $\mathbf{x}-\mathbf{y}, \mathbf{x}-\mathbf{z}, \mathbf{z}-\mathbf{y}$. Note that if the buckets have unequal sizes, then the total number of available pairs only increases due to the convexity of the function $x \mapsto \binom{x}{2} = \frac{x(x-1)}{2}$. In the context of BKW, such a heuristic improvement can be traced back to at least Levieil and Fouque [LF06].

Initializing sparse ternary vectors. Having chosen some N, one may set the initial list L_0 of vectors from $\mathbb{Z}^{m-n}$ to be as small as possible without duplicates. We therefore choose ternary vectors of ℓ_∞-norm w, where w is the smallest integer such that $2^w \binom{m-n}{w} \geq N$. This gives an initial variance of $\sigma_0^2 = w/(m-n)$.[8]

Rounding and quantization. The introduction of Gaussians is also motivated by provability, and one would rather use regular rounding in practice. Following the analysis of [KF15; GJMS17], the rounding introduced an error of deviation $\sigma_i = q/(p_i\sqrt{12})$. Both works also mention that lattice quantization could replace

[8] To relate the σ_i with the standard deviations of the discrete Gaussian distributions in the rest of the chapter, note that a discrete Gaussian distribution with parameter s_i has standard deviation $s_i/\sqrt{2\pi}$.

this rounding, which would improve the deviation to $q/(p_i\sqrt{2\pi e})$. This allows to choose smaller p_i while maintaining $\sigma_i = 2^{i/2} \cdot \sigma_0$, and therefore increases b_i.

Fractional parameters. The parameters p_i and b_i would need to be integers, which might force the attacker not to match exactly the optimal parametrization. However, the exercise of globally rounding those parameters optimally appears painful. We ignore these constraints for the attacker, leading to further underestimation of the attack cost.

Central Gaussian heuristic. At step r, we have obtained $3N$ many samples of standard deviation σ_r and we wish to know whether one of them is likely to have an ℓ_∞-norm bound less than β. We proceed using the error function as if the distribution at hand was Gaussian; there are many coordinates where this is reasonable, as they result from summing 2^{j+1} coordinates for $j \leq r$. This is questionable for the very last coordinates when using rounding; it seems less of an issue when using quantization which should result in a distribution close to uniform over a ball.

Having computed the success probability p for one sample, we consider the attack successful if $Np > 1/2$.

Early abort. It could be that after $r' < r$ steps, the remaining dimensions to lift over are nonzero yet small enough that one of the N samples could have small enough coordinates, while running one further step would double the variance of the rest of the vector. Such a trick has already been considered for lattice reduction attacks [DKLL+18; DEP23] on some instances of ISIS and SIS$^\times$, where, for different reasons, one would also naturally have many candidate solutions at hand. Hence, we consider such an option when exploring the parameter space. The number of left-over dimensions will be denoted by $\ell = n - \sum_{i=1}^{r'} b_i$.

3.5.1 Concrete analysis against ML-DSA

The SIS$^\infty$ parameters corresponding to the different security levels of the NIST standard ML-DSA [DKLL+18] are given in Table 3.1.

In order to calculate the smallest N such that the attack is successful according to the above analysis, we use the script from `https://github.com/lducas/Provable-Wagner-SIS`. This script offers both the straightforward rounding and

quantization version. The quantization version gave slightly better results, which we present in Table 3.2.

NIST level	n	m	q	β	q/β
2	$256 \cdot 4$	$256 \cdot 9$	8380417	350209	23.9
3	$256 \cdot 6$	$256 \cdot 12$	8380417	724481	11.6
5	$256 \cdot 8$	$256 \cdot 16$	8380417	769537	10.9

Table 3.1: SIS^{∞} parameters underlying ML-DSA.

NIST level	$\log_2 N$	w	σ_0	r'	$\sigma_{r'}$	ℓ
2	269.9	37	0.1700	40	178277.2	28.9
3	343.0	47	0.1749	42	366845.5	50.5
5	450.2	61	0.1726	42	361934.4	104.0

Table 3.2: Best attack parameters against ML-DSA according to the heuristic analysis above, using the quantization trick.

The results in the table underestimate the cost of the attack, as the attack cost (in terms of binary gates) is clearly larger than N. Hence, we conclude that Wagner's algorithm alone does not threaten the concrete security of ML-DSA: for example at NIST security level 2 [NIS16] (classically, approximately 2^{128} hash evaluations), it already requires operating and storing more than 2^{256} vectors.

3.6 Discussion

We conclude this chapter by discussing a few natural open questions about the practicality of Wagner's algorithm for SIS, its potential for LWE, and other applications.

3.6.1 Potential for practical attacks

When would this Wagner-style algorithm for SIS result in a practical attack? Given our results on the concrete application to ML-DSA, it remains open whether there exist (interesting) concrete parameters where Wagner has a practical chance to actually outperform lattice reduction for solving $\mathrm{SIS}^{\infty}_{n,m,q,\beta}$ with (say) $\beta = q/4$.

Looking at the asymptotic complexity $2^{O(n/\log\log q)}$ of the Wagner approach, it seems natural to choose a large q. However, lattice reduction attacks have runtime $2^{O(n\log(n)/\log(q))}$ when $q = n^{\Theta(1)}$ and m is large enough. Thus, to potentially outperform lattice reduction, it seems more favorable to choose q small and potentially take n very large. Fortunately, there is another possibility: a small number of variables m. Indeed, the results in this chapter established that Wagner could work for m as small as $n(1 + o(1))$, a regime that is very unfavorable to lattice attacks as it makes the determinant of the lattice huge.

A starting point to figure out whether this regime does indeed allow Wagner to practically outperform lattice reduction, would be to compare estimates of their costs for concrete parameters. The security estimation script from `https://github.com/pq-crystals/security-estimates/blob/master/MSIS_security.py` can be used to estimate the cost of lattice reduction attacks according to the core-SVP methodology, which could be compared with the underestimation of the concrete cost of Wagner from Section 3.5. For example, considering SIS^{∞} with $n = 500, m = 600, q = 1000, \beta = q/4$, this gives an estimated cost of roughly 2^{107} for lattice reduction, compared to a (heuristic and optimistic) estimate of $N = 2^{54}$ for Wagner. It is important to emphasize here that both numbers are merely illustrative, and may not be accurate estimates. To get more accurate estimates of the cost of Wagner for SIS, one could for instance adapt the software of [WBLW25].

3.6.2 Potential implications for LWE

The Wagner-style Gaussian sampler presented in this chapter was inspired by Kirchner and Fouque's algorithm for narrow-error LWE [KF15], which first samples many SIS solutions ("dual" vectors) in a Wagner-style manner and then applies a dual distinguisher to solve a decisional LWE instance. It is therefore natural to ask whether the discrete-Gaussian approach yields a provable subexponential-time algorithm for narrow-error LWE in the parameter regime where $m \ll n \log n$ (ideally $m = O(n)$), to essentially reprove the original claim of [KF15].

Having obtained a discrete Gaussian sampler for SIS lattices allowing for subexponential complexity in that parameter regime, one may be tempted to think that simply applying the dual distinguisher of [AR05] directly yields such an LWE algorithm. The problem, however, is that such a distinguisher needs to consider subexponentially many samples simultaneously. An argument based on the data-processing inequality shows that this forces one to instantiate the Gaussian sampler with $\varepsilon = 2^{-\tilde{\Omega}(n)}$, instead of $\varepsilon = n^{-\Theta(1)}$. Unfortunately, the complexity of the Gaussian sampler becomes exponential for such small ε.

This issue resonates with the one raised in [HKM18] regarding the proof of [KF15] when the parameter m is linear in n; namely, their issue is about reaching exponentially small statistical distance. However, it is technically different: for the Gaussian sampler, it cannot be fixed by increasing m to $\Theta(n \log n)$. Instead, the smoothing parameters of $\frac{q}{p_i}\mathbb{Z}^{b_i}$ are the limiting factor.

Thanks to the generality of the Wagner-style Gaussian sampler, it is plausible that this superlattice $\frac{q}{p_i}\mathbb{Z}^{b_i}$ of $q\mathbb{Z}^{b_i}$ can be replaced by a different one that has a similar index but a much smaller smoothing parameter. This gives hope that future work may finally provably fix the claim of [KF15] for m linear in n.

3.6.3 Other asymptotic applications

While we only applied it to SIS (and discussed its potential implications for LWE), our lattice re-interpretation of Wagner's algorithm shows that its potential may extend beyond SIS lattices. The underlying stategy, including the two main components `DGLift` (Algorithm 3.2) and `BucketAndCombine` (Algorithm 3.3), can be applied to any chain of projected lattices Λ_i and auxiliary superlattices Λ'_i, where we recall the high-level picture in Figure 3.1. Moreover, most of our analysis applies more generally as well, including the two key ingredients we used to prove that the output list contains a short nonzero lattice vector: Banaszczyk's lemmas (Lemmas 3.2.7 and 3.2.8) on the ℓ_2- and ℓ_∞-norm of discrete Gaussian samples, and Peikert and Rosen's min-entropy result (Lemma 3.2.10).

A natural open question is therefore whether Wagner's algorithm admits other interesting applications. In particular, are there other problems that allow a similar incremental, sieving-style strategy to this one (where SIS constraints are gradually added)?

Chapter 4

Predicting module-lattice basis reduction

In this chapter, we focus on one of the key tools in lattice-based cryptanalysis: lattice basis reduction, and particularly the BKZ algorithm. While the performance of BKZ has been analyzed extensively in the literature, many practical lattice-based schemes are based on module lattices, which are equipped with extra algebraic structure from a number field that is ignored by BKZ. In this chapter, we therefore study a module-lattice analog of BKZ, module-BKZ, and analyze its practical performance on random module lattices by predicting the quality of the output basis. The results identify the discriminant Δ_K of the underlying number field K as the main contributor to the output quality. As in prior work on unstructured BKZ, the analysis is heuristic-based but thoroughly validated through experiments. Altogether, this allows identifying number fields for which module-BKZ outperforms its unstructured counterpart, and provides concrete directions to further assess the implications for module-lattice-based cryptography.

This chapter is based on [DEP25].

The numerical results presented in this chapter were generated using code available at the repository `https://github.com/lducas/mBKZ/`. References of the form [`.py`] link to the relevant code in the online version of this thesis.

Contents of this chapter

4.1 Overview

4.1.1 Problem setting

Module lattices are lattices with some additional algebraic structure inherited from an underlying number field, that is, a finite-degree field extension of $\mathbb{Q}$. These structured lattices were introduced in cryptography in 1996 with the NTRU cryptosystem [HPS98], and have since seen increasing interest driven by foundational results on their average-case hardness [SSTX09; LPR13; SS11; LS15]. In particular, they now underlie the security of three NIST post-quantum standards, ML-KEM, ML-DSA, and FN-DSA [NIS22b], as well as various variants.

The central cryptanalytic tool for attacking those cryptosystems is *block lattice reduction*, a term covering a variety of algorithms (including BKZ [Sch87; GN08b], slide reduction [GN08a], and DBKZ [MW16]) that generalize the LLL algorithm [LLL82] in a way that offers a time-quality trade-off. Roughly speaking, block lattice reduction finds a shortest vector in an n-dimensional lattice up to an approximation factor of $\Theta(\sqrt{\beta})^{n/\beta}$ in time $2^{\Theta(\beta)}$, where $\beta \geq 2$ is a parameter known as the *blocksize*. It operates by solving a Shortest Vector Problem (SVP) in blocks of dimension β, corresponding to projected sublattices of the n-dimensional lattice. These blockwise SVP computations govern the overall runtime.

Until recently, lattice-reduction attacks were mostly oblivious to the module structure of the lattice. Yet, the potential of module variants of such algorithms was already addressed in the documentation [ABDK+21, Section 5.3, Q8] of Kyber (now standardized as ML-KEM) as part of its submission to the NIST standardization process. Specifically, question Q8 arises from the concern that the SVP subroutine could benefit from a d to d^2 speedup factor when applied to a module lattice over a cyclotomic field of degree d [BNP17]. Cyclotomic fields are obtained by adjoining a c-th primitive root of unity to $\mathbb{Q}$ (giving the c-th cyclotomic field), and are widely used in module-lattice-based schemes such as Kyber.

The discussion of Q8 in [ABDK+21] remarks that one could work over any subfield of the 512-th cyclotomic field used in Kyber, allowing to adjust the degree d to any power of 2 between 1 and 256. While the choice of a large d appears to maximize the potential gain in the SVP subroutine, it is highlighted to be at odds with various other speedups, such as progressive BKZ [AWHT16], progressive sieving [LM18], and the dimensions-for-free technique [Duc18]. More fundamentally, Q8 notes that such a *module-BKZ* algorithm, even when using the same blocksize as BKZ, may lead to a basis of slightly worse quality when applied

to module lattices over *power-of-two* cyclotomic fields, which are defined by a 2^k-th primitive root of unity ($k \geq 2$). Especially since module-lattice analogs of block lattice reduction have recently been developed [LPSW19; MS20], this calls for addressing the following question (relevant to all module-lattice-based schemes and not just Kyber):

Given access to the same SVP oracle for lattices in dimension β, is the output quality of module-BKZ better or worse than that of unstructured BKZ, and by how much?

This question has not been fully answered in [LPSW19; MS20]: they focus on the worst-case behavior of module-lattice analogs of LLL [LLL82] and slide reduction [GN08a], and are more concerned with the consequences of the existence of a fast short-vector oracle for module lattices of small rank over large number fields. Namely, both works present a worst-case to worst-case reduction from finding an approximately short vector in a module lattice to finding approximately short vectors in module lattices of smaller rank, where the trade-off in approximation factor depends on the module rank and on properties of the number field. However, there is often a significant gap between the theoretical, worst-case understanding of block lattice reduction and its *practical* performance. While [KK24] experimented with module-LLL (which can be viewed as a special case of module-BKZ) on the NTRU problem over power-of-two cyclotomic fields, they reported a negative result without a predictive analysis. In this chapter, we instead analyze the average-case performance of module-BKZ, develop predictions for its behavior, and compare them with those for unstructured BKZ.

4.1.2 Contributions

We present a quantitative study of the practical performance of module-lattice analogs of block lattice reduction. More precisely, we address the preceding question via a heuristic analysis supported by extensive experiments, yielding predictions of the *slope* of the basis profile obtained by module-BKZ. For a number field K of degree d, we denote the *module-BKZ $\mathbb{Q}$-slope* by

$$\mathrm{slope}_{\mathbb{Q}}(\mathrm{mBKZ}_K^{\beta/d})$$

where β corresponds to the dimension of the lattice given to the SVP oracle (and β/d to the K-rank of the corresponding module). The $\mathbb{Q}$-slope measures the quality of the output basis as a function of this *blocksize* β. Similar to unstructured BKZ, the

module-BKZ $\mathbb{Q}$-slope tends to increase toward zero as β increases, corresponding to better bases.

Our slope predictions allow us to compare the performance of module-BKZ with that of unstructured BKZ across different number fields. As shown in Figure 4.1, our predictions agree reasonably well with experimental results for module-BKZ over cyclotomic fields K.[1] A small gap remains, which may partly be explained by head and tail phenomena [YD17; BSW18] not captured by the slope model (the Geometric Series Assumption).

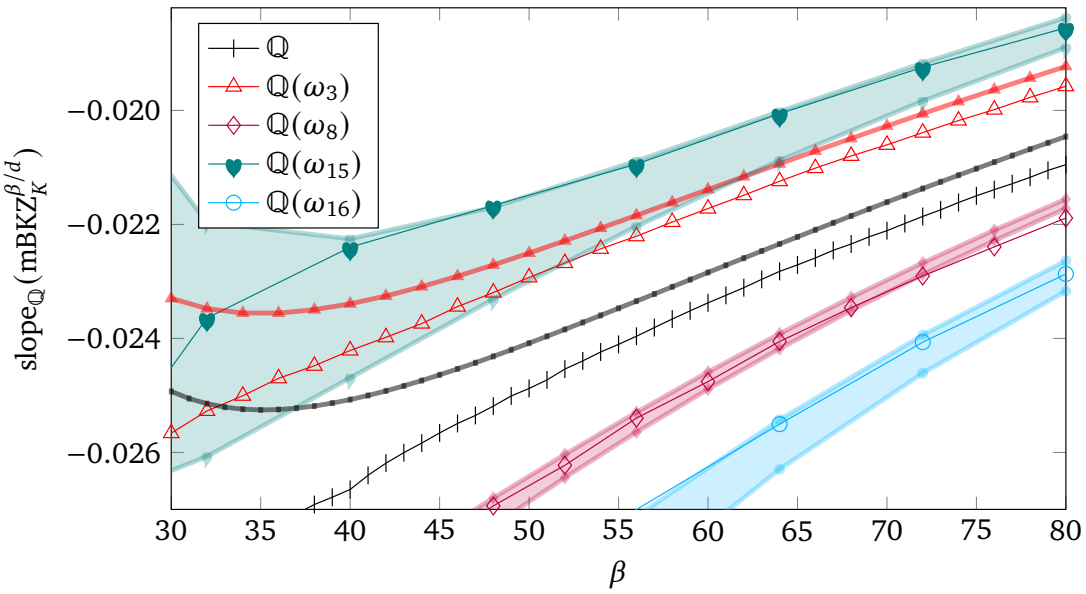


Figure 4.1: Module-BKZ $\mathbb{Q}$-slope for several cyclotomic fields K, with blocksize β ranging over multiples of the degree d of K. The case $K = \mathbb{Q}$ serves as a baseline for unstructured BKZ on embedded module lattices, under the general belief that BKZ is oblivious to their algebraic structure. Experimentally measured slopes [.py] are represented by thin lines with large marks, and were averaged over 5 random module lattices of dimension $rd = 240$. We progressively ran $5d$ tours for each multiple-of-d blocksize β to be close to convergence. Prediction intervals [.py] are represented by thick lines with small marks and a filled region in between. For $\mathbb{Q}$ and $\mathbb{Q}(\omega_3)$, both ends of the interval are too close to be distinguished.

[1] We conducted additional experiments [.py] that support the belief that BKZ is essentially oblivious to the module-lattice structure, justifying the use of $K = \mathbb{Q}$ as the unstructured baseline.

Based on the slope predictions, we also predict the "equivalent" blocksize β_{eq} for module-BKZ that results in the same slope as BKZ with blocksize β. A smaller blocksize $\beta_{\text{eq}} < \beta$ typically corresponds to a cheaper SVP oracle and suggests that module-BKZ benefits from leveraging the module structure. Besides providing concrete predictions for β_{eq} using explicit formulas and prediction scripts in Python, we show that it asymptotically satisfies:

$$\beta_{\text{eq}} = \beta + \ln\left(\frac{|\Delta_K|}{d^d}\right)\frac{\beta}{d\ln\beta}(1+o(1)) + d - 1 + o(1) \tag{4.1}$$

where d is the degree of the underlying number field K and Δ_K its discriminant. (See Heuristic Claim 4.5.3 for a more refined expression.) The $d-1+o(1)$ term in the asymptotic formula plays a role in the case $|\Delta_K| = d^d$, which for a cyclotomic field K holds if and only if K is a power-of-two cyclotomic field. For every other cyclotomic field K, $|\Delta_K| < d^d$, thereby yielding a blocksize gain in this case.

Equation (4.1) shows that, for a fixed number field K, the additive gain or loss in the blocksize is barely sublinear: $\beta_{\text{eq}} - \beta = \Theta(\beta/\ln\beta)$ with a constant depending on the discriminant Δ_K. As illustrated in Figure 4.2, our concrete estimates show that is quite substantial in practice for cyclotomic fields. This figure also shows that the asymptotic summary is not precise enough to be used as an approximation for concrete security estimates.

As a by-product, we provide an implementation of module-BKZ for various cyclotomic fields (including the c-th cyclotomic field for all $c \leq 16$), built on top of `fplll` [The23] and `G6K` [ADHK+19]. While the implementation is in principle applicable beyond cyclotomic fields, we encountered some technical limitations that we discuss later on. Although far from optimized, our implementation already enables experimentation with module-BKZ beyond the strict focus of this chapter, and should be useful for addressing several open questions posed in Section 4.6.

The implementation is available at `https://github.com/lducas/mBKZ/`, and we refer to the relevant parts of the code using the symbol [`.py`].

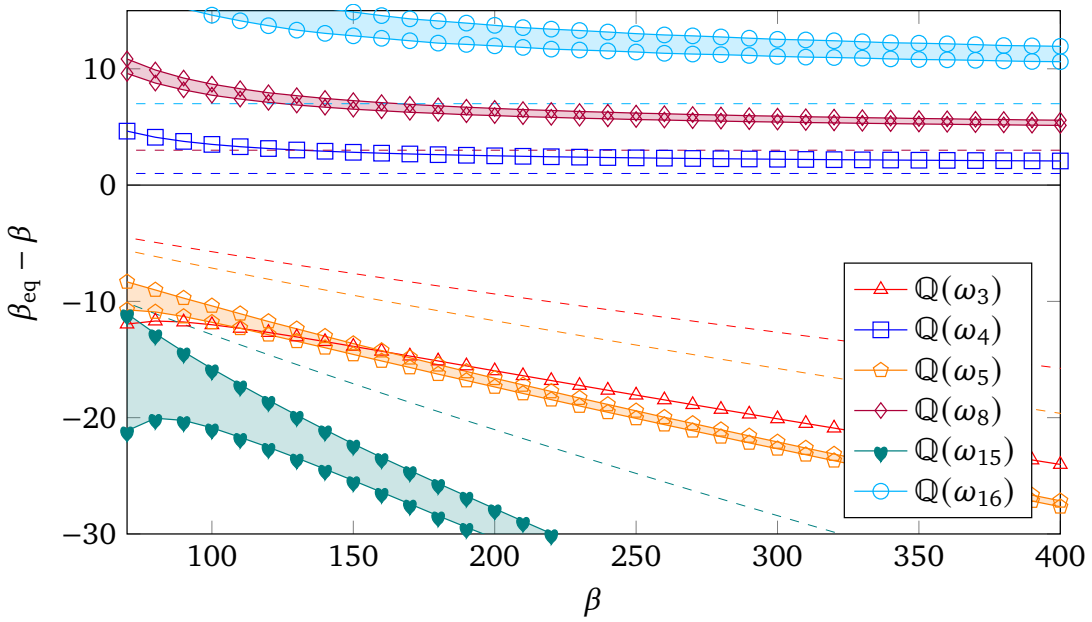


Figure 4.2: Predictions for the difference $\beta_{\mathrm{eq}} - \beta$ of blocksizes for $\mathrm{mBKZ}_K^{\beta_{\mathrm{eq}}/d}$ to reach the same slope as unstructured BKZ^{β} for several cyclotomic fields K. Concrete predictions [.py] are represented by lines with large marks and a filled region in between. Asymptotic predictions [.py] based on Heuristic Claim 4.5.3 (ignoring some lower-order terms) are represented by dashed lines.

4.1.3 Technical overview

The goal of lattice reduction is to turn a given lattice basis into one of better quality. As is standard in lattice reduction, we measure the quality of a $\mathbb{Z}$-basis $(\mathbf{z}_1, \ldots, \mathbf{z}_n)$ of an n-dimensional Euclidean lattice by its *profile*: the sequence

$$\ln\|\mathbf{z}_1^*\|, \ldots, \ln\|\mathbf{z}_n^*\|$$

of logarithmic Euclidean norms of the vectors obtained by Gram-Schmidt orthogonalization (without normalization). As we will see later, the aforementioned $\mathbb{Q}$-*slope* can be interpreted as the slope of a linear fit to the points $(i, \ln\|\mathbf{z}_i^*\|)$.

The sum of those $\ln\|\mathbf{z}_i^*\|$ is an invariant of the lattice, namely the logarithm of the lattice's determinant. The profile of a (reduced) lattice basis is typically decreasing in i, and the flatter it is, the better. In particular, since $\mathbf{z}_1^* = \mathbf{z}_1$, a flat profile provides a short nonzero lattice vector. We are therefore interested in how lattice-reduction algorithms modify the global trend of the basis profile.

In this chapter, we focus on the behavior of lattice reduction on the subfamily of *module* lattices. Module lattices are defined over a number field K such as the c-th cyclotomic field $K = \mathbb{Q}(\omega_c)$, which is the smallest field containing both $\mathbb{Q}$ and a primitive c-th root of unity ω_c. The degree of K is defined by $d = [K : \mathbb{Q}]$. K is associated with a ring of integers denoted $\mathcal{O}_K$ and a canonical embedding $\boldsymbol{\sigma}$ that embeds elements of K into $\mathbb{R}^d$. This embedding naturally extends to vectors $\mathbf{x} \in K^r$, yielding vectors of the form $\boldsymbol{\sigma}(\mathbf{x}) \in \mathbb{R}^{rd}$. In particular, the canonical embedding transforms a finitely-generated $\mathcal{O}_K$-module $\mathcal{M} \subseteq K^r$ of rank r into a Euclidean lattice $\boldsymbol{\sigma}(\mathcal{M}) \subseteq \mathbb{R}^{rd}$ of dimension rd.

Given an $\mathcal{O}_K$-module $\mathcal{M}$, each nonzero $\mathbf{b} \in \mathcal{M}$ generates a rank-1 module $\mathbf{b}\mathcal{O}_K$, which embeds into a d-dimensional Euclidean lattice. Such a vector $\mathbf{b}$ therefore gives rise to d vectors in the embedded lattice $\boldsymbol{\sigma}(\mathcal{M})$ that are $\mathbb{R}$-linearly independent and implicitly carry K's algebraic structure. Specifically, we obtain d such vectors from the embeddings of

$$x_1\mathbf{b}, \ldots, x_d\mathbf{b}$$

where $(x_1, \ldots, x_d)$ can be any $\mathbb{Z}$-basis of $\mathcal{O}_K$. For example, if $K = \mathbb{Q}(\omega_c)$ is a cyclotomic field, where ω_c denotes a primitive c-th root of unity, then such a basis is given by $(1, \omega_c, \ldots, \omega_c^{d-1})$.

While unstructured BKZ simply treats a module lattice as a Euclidean lattice, module-BKZ takes the above observations into account: each time a nonzero vector $\mathbf{b} \in \mathcal{M}$ of short length $\|\mathbf{b}\| := \|\boldsymbol{\sigma}(\mathbf{b})\|$ is found (using an SVP oracle in a smaller dimension β), the algorithm inserts the corresponding vectors

$$\boldsymbol{\sigma}(x_1\mathbf{b}), \ldots, \boldsymbol{\sigma}(x_d\mathbf{b})$$

into the lattice basis for a suitable choice of $(x_1, \ldots, x_d)$. (Since the size of the basis is fixed, some existing basis vectors must be removed.) Using the properties of K, we can then attempt to predict how these d inserted vectors alter the overall basis profile of the rd-dimensional lattice; recall that a local change affects the profile globally due to the determinant invariant.

Two enlightening cases. Consider an $\mathcal{O}_K$-module over the fourth cyclotomic field $K = \mathbb{Q}(\imath)$, which has degree $d = 2$. Having found a somewhat short vector $\mathbf{b}_1$, one would naturally set $\mathbf{b}_2 := \imath\mathbf{b}_1$, since $(1, \imath)$ is a $\mathbb{Z}$-basis of $\mathcal{O}_K = \mathbb{Z}[\imath]$. The embedded vectors $\boldsymbol{\sigma}(\mathbf{b}_1)$ and $\boldsymbol{\sigma}(\mathbf{b}_2) = \boldsymbol{\sigma}(\imath\mathbf{b}_1)$ are always orthogonal with respect to the Euclidean inner product, so $\|\boldsymbol{\sigma}(\mathbf{b}_1)^*\| = \|\boldsymbol{\sigma}(\mathbf{b}_2)^*\|$. (Other valid choices are

$\mathbf{b}_2 \coloneqq (k + \imath)\mathbf{b}_1$ for $k \in \mathbb{Z}$, but these will not change $\|\boldsymbol{\sigma}(\mathbf{b}_2)^*\|$.) The profile of the embedded basis may look perfectly flat locally (for these $d = 2$ vectors), but this constraint actually makes the global profile less flat: $\|\boldsymbol{\sigma}(\mathbf{b}_2)^*\|$ is a bit larger than it would have been for an unstructured reduction, which implies that more of the determinant has been consumed, lowering the rest of the profile.

If we instead consider the third cyclotomic field $\mathbb{Q}(\omega_3)$, which also has degree $d = 2$, then the situation is rather different. In this case, $\mathbf{b}_1$ and $\mathbf{b}_2 \coloneqq \omega_3\mathbf{b}_1$ always form an angle of $\pi/3$, so $\|\boldsymbol{\sigma}(\mathbf{b}_1)^*\| = \sqrt{4/3}\ \|\boldsymbol{\sigma}(\mathbf{b}_2)^*\|$. In particular, $\boldsymbol{\sigma}(\mathbf{b}_2)^*$ is significantly shorter than what it would have been in an unstructured reduction, making the profile locally more declined, but globally flatter across the full basis.

The general case. Consider an $\mathcal{O}_K$-module $\mathcal{M}$ over an arbitrary number field K of degree d, and let $\mathbf{b}_1 \in \mathcal{M}$ correspond to the first vector in a reduced basis of $\boldsymbol{\sigma}(\mathcal{M})$. The effect of the first d basis vectors on the overall basis profile then depends on $\mathbf{b}_1\mathcal{O}_K$, and therefore on $\|\mathbf{b}_1\|$. In case of unstructured BKZ, this length is predicted by the Gaussian Heuristic, which is backed by non-heuristic theorems [Rog56; Söd11; Che13; LN24] if one assumes a careful definition of random lattices. In case of module-BKZ, we should however take into account the module structure. Fortunately, an adaptation of these theorems has recently been proven for random module lattices over cyclotomic fields [GSVV24], allowing us to formulate a module-lattice analog of the Gaussian Heuristic to predict $\|\mathbf{b}_1\|$.

If K is an imaginary quadratic number field (including the previous two examples $\mathbb{Q}(\imath)$ and $\mathbb{Q}(\omega_3)$), then the rank-1 module lattice $\mathbf{b}_1\mathcal{O}_K$ is always a scaled rotation of $\mathcal{O}_K$ itself, and the quality of the first d vectors in the reduced basis is therefore directly related to the density of $\mathcal{O}_K$ as a lattice, or, in algebraic terms, to the absolute discriminant $|\Delta_K|$ of K.

The situation is a bit more complex beyond imaginary quadratics (including all other cyclotomic fields $K \supsetneq \mathbb{Q}$), where $\mathbf{b}_1\mathcal{O}_K$ need not be a scaled rotation of $\mathcal{O}_K$: it gets *skewed*. Yet, using a heuristic analysis (adapted from [DW21, Lemma 4.4]) we show that the skewness quickly decreases with the rank of $\mathcal{M}$ if we model the direction of $\mathbf{b}_1$ as a uniform unit vector, suggesting that the skewness is controlled and asymptotically irrelevant. Our experiments demonstrate less average skewness than predicted by our model, so this model is not perfectly accurate. As we do not have a better model to offer, we translate this into an interval, with one end corresponding to no skewness and the other to the spherical-model estimate. Luckily, the consequence of that uncertainty is quantitatively mild, fading away as the blocksize grows.

Another complication can happen: it is possible that $\mathbf{b}_1\mathcal{O}_K$ does not contain all module-lattice points in $\mathbf{b}_1K$. In such cases, instead of taking $\mathbf{b}_1\mathcal{O}_K$ as the first rank-1 module (corresponding to the first d basis vectors), we want to work with the larger module $\mathbf{b}_1K \cap \mathcal{M}$, which may affect the basis profile. Here again, a heuristic analysis involving the Dedekind zeta function (in a fashion similar to [ABD16; DPPW22; DW21]) allows modeling the distribution of the *index* of $\mathbf{b}_1\mathcal{O}_K$ in $\mathbf{b}_1K \cap \mathcal{M}$, in order to measure how much larger $\mathbf{b}_1K \cap \mathcal{M}$ is, and consequently the effect on the basis profile. Similar to our model for skewness, this model appears to be an overestimate compared to the experimentally observed index, for an explainable reason, yet without an obvious fix. Luckily again, the uncertainty it leaves is inconsequential.

This gives us four terms driving the output quality of module-BKZ: the module-lattice Gaussian Heuristic, the discriminant, the skewness, and the index. Putting it all together, we conclude with a concrete prediction of the (slope of the) basis profile produced by module-BKZ, and an asymptotic analysis of the gain or loss in terms of the blocksize.

Remark 4.1.1 (Dense sublattices). The discussion above suggests that what gives module-BKZ the potential to outperform its unstructured counterpart is its ability to find not just short vectors, but *dense* sublattices (which happen to be rank-1 modules). Interestingly, this advantage is not only about the ability to find them, but also about their mere existence in case of module lattices. Namely, a recent work [DL25] predicted the output quality of a variant of BKZ that uses a densest-sublattice oracle for unstructured lattices, and concluded that the densest sublattices of random *unstructured* lattices are not dense enough to significantly improve over vanilla BKZ.

4.1.4 Organization

Section 4.2 provides chapter-specific background on module lattices. In Section 4.3, we formally define module-BKZ and describe the details of our implementation. In Section 4.4, we present our slope prediction for module-BKZ and dive into the four main terms driving this slope, where we motivate the underlying heuristics using experimental results. In Section 4.5, we compare the asymptotic blocksizes for module-BKZ and BKZ that yield the same slope. We conclude the chapter in Section 4.6 by addressing the cryptanalytic impact of our predictions, and by suggesting directions for future work.

4.2 Preliminaries

Throughout this chapter, we use the symbol [.py] for external (clickable) links to the corresponding part of the code where available.

Remark 4.2.1 (On random lattices). In this chapter, we study the performance of module-BKZ on *random lattices*, which involves analyzing various random lattices encountered by the algorithm. It is therefore worth mentioning what we mean with "random" here. It is well known that using the Haar measure, one can formally define a uniform distribution over the set of n-dimensional lattices of unit determinant [Sie45]. However, as is the case for unstructured BKZ, the distribution of the random lattices encountered in our analysis of module-BKZ is not well understood. We will therefore not always explicitly define the underlying distribution, and (implicitly) refer to its actual distribution induced by module-BKZ. In our analysis, we circumvent this lack of understanding similarly to how unstructured BKZ is analyzed: we approximate these distributions using heuristics that we verify experimentally.

Whenever we write $\mathbb{E}[X]$ without specifying the distribution of the random variable X, we implicitly refer to the (unknown) distributions encountered during the BKZ or module-BKZ algorithms.

4.2.1 Algebraic background

An (algebraic) number field K is a finite-degree field extension of $\mathbb{Q}$. Its *degree* is $[K : \mathbb{Q}]$. We denote the number of roots of unity in K by μ_K.

For $\alpha \in \mathbb{C}$, we let $\mathbb{Q}(\alpha)$ denote the smallest field containing $\mathbb{Q}$ and α. By the so-called primitive-element theorem (e.g., see [Eve]), a number field K is of the form $K = \mathbb{Q}(\alpha)$ for some $\alpha \in \mathbb{C}$ (and thus $\alpha \in K$).

Embeddings. When K is a number field of degree d, it admits d distinct *embeddings* into $\mathbb{C}$: there are exactly d injective field homomorphisms $\sigma\colon K \to \mathbb{C}$ (each embedding therefore leaves $\mathbb{Q}$ unchanged). An embedding is called *real* if $\sigma(K) \subseteq \mathbb{R}$, and called *complex* otherwise. Complex embeddings come in conjugate pairs: for each complex embedding σ, the map $\overline{\sigma}\colon x \mapsto \overline{\sigma(x)}$ (for $x \in K$) is a complex embedding as well. We can therefore write $d = d_{\mathbb{R}} + 2d_{\mathbb{C}}$, where $d_{\mathbb{R}}$ denotes the number of real embeddings and $d_{\mathbb{C}}$ the number of conjugate pairs of complex embeddings.

We denote the set of real embeddings by $\mathcal{E}_{\mathbb{R}}$, the $d_{\mathbb{C}}$ embeddings corresponding to roots with strictly positive imaginary part by $\mathcal{E}_{\mathbb{C}}^{+}$, and their conjugates by $\overline{\mathcal{E}_{\mathbb{C}}^{+}}$. Altogether, the set $\mathcal{E}$ of all d embeddings of K decomposes as $\mathcal{E} = \mathcal{E}_{\mathbb{R}} \bigsqcup \mathcal{E}_{\mathbb{C}}^{+} \bigsqcup \overline{\mathcal{E}_{\mathbb{C}}^{+}}$.

Canonical embedding. The *canonical embedding* of K is the map $\boldsymbol{\sigma} : K \to \mathbb{C}^d$ defined by $\boldsymbol{\sigma}(x) = (\sigma(x))_{\sigma \in \mathcal{E}}$. Since the complex embeddings come in conjugate pairs, we may equivalently view $\boldsymbol{\sigma}$ as an embedding $K \to \mathbb{R}^d$ by choosing one embedding from each conjugate pair and identifying $\mathbb{C} \cong \mathbb{R}^2$ (while preserving the Euclidean norm of $\boldsymbol{\sigma}(x)$). We will often use this "real" representation.

Ring of integers. We write $\mathcal{O}_K$ for the *ring of integers* of K, which consists of the elements $x \in K$ such that $Q(x) = 0$ for some monic polynomial $Q \in \mathbb{Z}[X]$. The ring $\mathcal{O}_K$ is a free $\mathbb{Z}$-module of rank d, and thus

$$\mathcal{O}_K = \left\{ \sum_{i=1}^{d} z_i x_i : z_i \in \mathbb{Z} \right\}$$

for some $\mathbb{Z}$-basis $(x_1, \ldots, x_d)$ of $\mathcal{O}_K$. For instance, $(1, \alpha, \ldots, \alpha^{d-1})$ is such a basis if $\mathcal{O}_K = \mathbb{Z}[\alpha]$ for some $\alpha \in K$.

Discriminant. The ring of integers $\mathcal{O}_K$ and embeddings of K allow us to define its *discriminant* Δ_K. Specifically, writing $\sigma_1, \ldots, \sigma_d$ for the embeddings of K and $(x_1, \ldots, x_d)$ for an arbitrary $\mathbb{Z}$-basis of $\mathcal{O}_K$, we define $\Delta_K := \det((\sigma_i(x_j))_{i,j \in [d]})^2$.

The ring $K_{\mathbb{R}}$. We will typically work over the ring $K_{\mathbb{R}} := K \otimes_{\mathbb{Q}} \mathbb{R}$. The embeddings of K extend naturally to $K_{\mathbb{R}}$ and yield a ring isomorphism $K_{\mathbb{R}} \cong \mathbb{R}^{d_{\mathbb{R}}} \times \mathbb{C}^{d_{\mathbb{C}}}$. The ring $K_{\mathbb{R}}$ has a well-defined notion of complex conjugation: for all $x \in K_{\mathbb{R}}$, we define its complex conjugate as the element $\overline{x} \in K_{\mathbb{R}}$ satisfying $\sigma(\overline{x}) = \overline{\sigma(x)}$ for all $\sigma \in \mathcal{E}$.

Each embedding is extended to $K_{\mathbb{R}}^m$ for $m \geq 1$ by defining $\sigma(\mathbf{x}) = (\sigma(x_i))_{i=1}^m$ for all $\mathbf{x} \in K_{\mathbb{R}}^m$. This also allows us to extend the definition of complex conjugation to $\mathbf{x} \in K_{\mathbb{R}}^m$ by applying it coordinatewise.

Trace norm. We define the field *trace* $\mathrm{Tr}\colon K_{\mathbb{R}} \to \mathbb{R}$ by $x \mapsto \sum_{\sigma\in\mathcal{E}} \sigma(x)$, which induces an inner product on $K_{\mathbb{R}}$ given by $\langle x, y\rangle := \mathrm{Tr}(x\overline{y})$ for $x, y \in K_{\mathbb{R}}$. This trace inner product is extended to $K_{\mathbb{R}}^m$ by defining $\langle \mathbf{x}, \mathbf{y}\rangle := \sum_{i=1}^m \langle x_i, y_i\rangle$ for all $\mathbf{x} = (x_1, \ldots, x_m), \mathbf{y} = (y_1, \ldots, y_m) \in K_{\mathbb{R}}^m$, and thus

$$\langle \mathbf{x}, \mathbf{y}\rangle := \sum_{i=1}^m \langle x_i, y_i\rangle = \sum_{i=1}^m \mathrm{Tr}(x_i\overline{y_i}) = \sum_{i=1}^m \sum_{\sigma\in\mathcal{E}} \sigma(x_i)\overline{\sigma(y_i)}.$$

In particular, this yields a geometric norm $\|\cdot\|\colon \mathbf{x} \mapsto \sqrt{\langle \mathbf{x}, \mathbf{x}\rangle}$, called the *trace norm*. Note that $\|\mathbf{x}\| = \|\boldsymbol{\sigma}(\mathbf{x})\|$.

We also define $\langle \mathbf{x}, \mathbf{y}\rangle_K := \sum_{i=1}^m x_i\overline{y_i} \in K_{\mathbb{R}}$, which satisfies $\mathrm{Tr}(\langle \mathbf{x}, \mathbf{y}\rangle_K) = \langle \mathbf{x}, \mathbf{y}\rangle$.

Algebraic norm. Besides the trace norm $\|x\|$, we consider the *algebraic norm* $\mathrm{N}(x) := \prod_{\sigma\in\mathcal{E}} \sigma(x)$ of elements $x \in K_{\mathbb{R}}$. By the inequality of arithmetic and geometric means, we have $\sqrt{d}\,\mathrm{N}(x)^{1/d} \le \|x\|$. For $\mathbf{x} \in K_{\mathbb{R}}^m$, we write $\mathrm{N}(\mathbf{x})$ as shorthand for $\mathrm{N}(\langle \mathbf{x}, \mathbf{x}\rangle_K)^{1/2}$. (With slight abuse of notation, we take the square root outside of the norm, so we do not have to define the square root over $K_{\mathbb{R}}$.)

Cyclotomic fields

The main class of number fields we consider in this work is the class of *cyclotomic fields*, which are number fields of the form

$$K = \mathbb{Q}(\omega_c)$$

for $c \in \mathbb{N}$, where ω_c is a primitive c-th root of unity (i.e., $\omega_c \in \mathbb{C}$ and satisfies $\omega_c^c = 1$ and $\omega_c^j \ne 1$ for all $1 \le j < c$). The degree of $K = \mathbb{Q}(\omega_c)$ equals $d = \phi(c)$, where ϕ denotes Euler's totient function (i.e., $\phi(x)$ equals the number of integers in $[x]$ coprime to $x \in \mathbb{N}$). The ring of integers of K is $\mathcal{O}_K = \mathbb{Z}[\omega_c]$. Moreover, when $c > 2$, all of K's embeddings are complex: $d_{\mathbb{R}} = 0$ and $d_{\mathbb{C}} = d/2$.

The *conductor* of a cyclotomic field K is the minimal $c \in \mathbb{N}$ such that $K = \mathbb{Q}(\omega_c)$. If $c \in \mathbb{N}$ is odd, then $\mathbb{Q}(\omega_c) \cong \mathbb{Q}(\omega_{2c})$, and this criterion captures all isomorphic cyclotomic fields: the conductor c of a cyclotomic field K is either odd (in which case $K \cong \mathbb{Q}(\omega_{2c})$) or a multiple of 4. The number of roots of unity in a cyclotomic field K of conductor c equals $\mu_K = c$ when c is even, and $\mu_K = 2c$ when c is odd.

Ideals, modules, and module lattices

A *fractional ideal* of $\mathcal{O}_K$ is an $\mathcal{O}_K$-submodule $\mathfrak{I} \subseteq K$ (i.e., it is closed under addition and under multiplication by elements of $\mathcal{O}_K$) for which there exists $x \in \mathcal{O}_K \setminus \{0\}$ satisfying $x\mathfrak{I} \subseteq \mathcal{O}_K$. The algebraic norm $\mathrm{N}(\mathfrak{I})$ of a fractional ideal $\mathfrak{I}$ is defined as $\mathrm{N}(\mathfrak{I}) := [\mathcal{O}_K : x\mathfrak{I}]/|\mathrm{N}(x)|$ for any $x \in K \setminus \{0\}$ satisfying $x\mathfrak{I} \subseteq \mathcal{O}_K$.

Definition 4.2.2 ($\mathcal{O}_K$-module and pseudobasis)**.** A *$\mathcal{O}_K$-module* or *module* in $K_{\mathbb{R}}^m$ of *rank* r is a set of the form

$$\mathcal{M} = \sum_{i=1}^{r} \mathbf{b}_i \mathfrak{I}_i \subseteq K_{\mathbb{R}}^m$$

where $\mathfrak{I}_1, \ldots, \mathfrak{I}_r$ are nonzero fractional $\mathcal{O}_K$-ideals, and $\mathbf{b}_1, \ldots, \mathbf{b}_r \in K_{\mathbb{R}}^m$ are $K_{\mathbb{R}}$-linearly independent. The set $(\mathbf{b}_i, \mathfrak{I}_i)_{i=1}^r$ is called a *pseudobasis* for $\mathcal{M}$.[a]

$\mathcal{M}$ is a *free* module if the $\mathfrak{I}_i$'s are all equal to $\mathcal{O}_K$, that is, if $\mathcal{M} = \mathbf{B} \cdot \mathcal{O}_K^r$ for $\mathbf{B} := (\mathbf{b}_1, \ldots, \mathbf{b}_r) \in K_{\mathbb{R}}^{m \times r}$.

[a] Technically, $\mathcal{M}$ is a *finitely-generated* $\mathcal{O}_K$-module, but the word "finitely generated" is often omitted in the literature, so we stick to that convention. Another way to represent such modules, besides pseudobases, is via module filtrations (see [MS20]).

The rank r of a module $\mathcal{M} \subseteq K_{\mathbb{R}}^m$ satisfies $r = \dim_{K_{\mathbb{R}}} \mathrm{span}_{K_{\mathbb{R}}}(\mathcal{M})$. For simplicity, we will often consider $m = r$ in the remainder of this work. Denoting by $\mathbf{B}$ the matrix with columns $\mathbf{b}_1, \ldots, \mathbf{b}_r$, the determinant of $\mathcal{M}$ in $K_{\mathbb{R}}$ is defined as

$$\det_{K_{\mathbb{R}}}(\mathcal{M}) = \det(\overline{\mathbf{B}}^\top \mathbf{B})^{1/2} \prod_{i=1}^{r} \mathfrak{I}_i.$$

Unital pseudobasis. A pseudobasis $(\mathbf{b}_i, \mathfrak{I}_i)_{i=1}^r$ is said to be *unital* if $1 \in \mathfrak{I}_i$ for all i (equivalently, if $\mathcal{O}_K \subseteq \mathfrak{I}_i$ for all i). We remark that $\mathcal{O}_K \subseteq \mathfrak{I}_i$ implies that $\mathrm{N}(\mathfrak{I}_i) \leq 1$. Any pseudobasis can be turned into a unital pseudobasis using [LPSW19, Alg. 3.2].

Module lattice. Through the embeddings of K, we can view modules as Euclidean lattices. More precisely, the canonical embedding $\boldsymbol{\sigma}$ maps a rank-r module $\mathcal{M} \subseteq K_{\mathbb{R}}^r$ to an rd-dimensional Euclidean lattice $\boldsymbol{\sigma}(\mathcal{M})$ in $\mathbb{R}^{rd}$. For example, taking $K = \mathbb{Q}$ (so $d = 1$ and $\mathcal{O}_K = \mathbb{Z}$), the canonical embedding is trivial, and we recover the usual notion of Euclidean lattices in $\mathbb{Q}^r$. Note that the geometry is preserved: the trace inner product of a module element in $K_{\mathbb{R}}^r$ equals the Euclidean inner product of the corresponding embedded vector in $\mathbb{R}^{rd}$. It can be shown that the determinant of the embedded lattice $\boldsymbol{\sigma}(\mathcal{M})$ equals $\det(\boldsymbol{\sigma}(\mathcal{M})) = |\Delta_K|^{r/2} \mathrm{N}(\det_{K_{\mathbb{R}}}(\mathcal{M}))$.

We summarize the relations between properties of $\mathcal{M}$ and $\boldsymbol{\sigma}(\mathcal{M})$ in Table 4.1. In the remainder of this chapter, we will typically write $\mathcal{M}$ to denote both the module itself and the associated Euclidean lattice $\boldsymbol{\sigma}(\mathcal{M})$.

Object	$\mathcal{O}_K$-module $\mathcal{M}$	Euclidean lattice $\boldsymbol{\sigma}(\mathcal{M})$
Space	$K_{\mathbb{R}}^r$	$\mathbb{R}^{rd}$
Dimension	Dimension r of its $K_{\mathbb{R}}$-span (the *rank* of $\mathcal{M}$)	Dimension rd of its $\mathbb{R}$-span (the *dimension* of $\boldsymbol{\sigma}(\mathcal{M})$)
Representation	Pseudobasis $((\mathbf{b}_i, \mathfrak{I}_i))_{i=1}^r$, i.e., the $\mathbf{b}_i \in K_{\mathbb{R}}^r$ are $K_{\mathbb{R}}$-linearly independent and the $\mathfrak{I}_i$ are nonzero fractional $\mathcal{O}_K$-ideals	$\mathbb{Z}$-basis $(\mathbf{z}_1, \ldots, \mathbf{z}_{rd})$, i.e., the $\mathbf{z}_i \in \mathbb{R}^{rd}$ are $\mathbb{R}$-linearly independent
Determinant	$\det_{K_{\mathbb{R}}}(\mathcal{M})$	$\det(\boldsymbol{\sigma}(\mathcal{M}))$, equals $\lvert\Delta_K\rvert^{r/2} \mathrm{N}(\det_{K_{\mathbb{R}}}(\mathcal{M}))$
Geometric norm	Trace norm $\lVert\mathbf{b}\rVert$	Euclidean norm $\lVert\boldsymbol{\sigma}(\mathbf{b})\rVert$, equals $\lVert\mathbf{b}\rVert$

Table 4.1: Important quantities associated with a module lattice over a degree-d number field K, where $\boldsymbol{\sigma}$ denotes the canonical embedding of K.

4.2.2 GSO over $K_{\mathbb{R}}$

Two vectors are said to be $K_{\mathbb{R}}$-linearly independent if and only if the zero vector cannot be expressed as a non-trivial $K_{\mathbb{R}}$-linear combination of them. This notion allows us to extend Gram-Schmidt orthogonalization (Definition 2.3.7) to matrices $\mathbf{B} \in K_{\mathbb{R}}^{m \times r}$ with $K_{\mathbb{R}}$-linearly independent columns. (For instance, see [FS10; LPSW19].)

Definition 4.2.3 (GSO over $K_{\mathbb{R}}$)**.** Given $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_r)$ such that the $\mathbf{b}_i$ are $K_{\mathbb{R}}$-linearly independent, we define its GSO as $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_r^*)$, where $\mathbf{b}_1^* = \mathbf{b}_1$ and, for all $1 < i \leq r$,

$$\mathbf{b}_i^* := \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^* \quad \text{with } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle_K}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle_K} \text{ for all } 1 \leq j < i.$$

Note that $\langle \mathbf{b}_i^*, \mathbf{b}_j^* \rangle_K = 0$ for all $i \neq j$.

More generally, we define the projection $\pi_i : \mathbf{x} \mapsto \mathbf{x} - \sum_{j=1}^{i-1} \frac{\langle \mathbf{x}, \mathbf{b}_j^* \rangle_K}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle_K} \mathbf{b}_j^*$ for all i, so $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$. Given a rank-$r$ module lattice $\mathcal{M}$ with pseudobasis $\mathfrak{B} = ((\mathbf{b}_i, \mathfrak{I}_i))_{i=1}^r$, we define the *projected module lattices*

$$\mathcal{M}(\mathfrak{B}_{[i:j]}) = \pi_i(\mathbf{b}_i)\mathfrak{I}_i + \cdots + \pi_i(\mathbf{b}_j)\mathfrak{I}_j$$

for $1 \leq i \leq j \leq r$. Note that $\mathcal{M}(\mathfrak{B}_{[i:j]})$ has rank $j - i + 1$, and depends on $\mathfrak{B}$.

The determinant of $\mathcal{M}$ and of its projected module lattices can be expressed in terms of the GSO of the pseudobasis. Namely, for all $1 \leq i \leq j \leq r$, we have

$$\det(\mathcal{M}(\mathfrak{B}_{[i:j]})) = |\Delta_K|^{(j-i+1)/2} \cdot \prod_{k=i}^{j} \mathrm{N}(\mathbf{b}_k^*) \, \mathrm{N}(\mathfrak{I}_k).$$

4.3 Module-BKZ

We first review how the BKZ slope is often predicted for arbitrary lattices. Next, we describe the modified BKZ algorithm for module lattices, *module-BKZ*, and present the specifics of our implementation.

4.3.1 Prediction of the BKZ slope

Given a basis of an n-dimensional (Euclidean) lattice with GSO $(\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$, we define its $\mathbb{Q}$-*profile* as the sequence $(\ell_1^{\mathbb{Q}}, \ldots, \ell_n^{\mathbb{Q}})$, where $\ell_i^{\mathbb{Q}} := \ln\|\mathbf{b}_i^*\|$.[2] As mentioned earlier, the more balanced the profile is, the better.

Turning a basis into a basis of better quality is the task of lattice reduction algorithms, such as the BKZ algorithm [Sch87; SE94]. In short, for blocksize β, the BKZ algorithm aims to return a lattice basis $\mathbf{B}$ such that its GSO $(\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)$ is BKZ$^\beta$*-reduced*, i.e., it satisfies

$$\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{L}(\mathbf{B}_{[i:\min(i+\beta-1,n)]})) \qquad \forall 1 \le i \le n$$

where $\mathcal{L}(\mathbf{B}_{[i:j]})$ is the lattice generated by $(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_j))$ and π_i is the linear projection orthogonal to the span of $\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}$. Approximation variants of BKZ exist as well, but are beyond the scope of this thesis.

Slope prediction under the Geometric Series Assumption

One can predict the slope of the $\mathbb{Q}$-profile of a BKZ$^\beta$-reduced basis. Such predictions are often based on the empirical observation that the $\mathbb{Q}$-profile of a BKZ$^\beta$-reduced basis tends to resemble a descending straight line for sufficiently large $\beta \ll n$ (say $\beta \ge 50$). This is formalized by the Geometric Series Assumption (GSA), as first proposed by Schnorr [Sch03].

> **Heuristic 4.3.1** (Euclidean-lattice GSA)**.** *Let $\beta \ll n$ be sufficiently large. There is a constant $\alpha_{\mathbb{Q}} > 1$ (depending only on β) such that the $\mathbb{Q}$-profile of a* BKZ$^\beta$*-reduced basis of a random n-dimensional Euclidean lattice of fixed determinant satisfies*
>
> $$\mathbb{E}[\ell_i^{\mathbb{Q}}] = \mathbb{E}[\ell_1^{\mathbb{Q}}] - (i-1)\ln\alpha_{\mathbb{Q}} \qquad \forall 1 \le i \le n. \qquad \text{(GSA)}$$

[2] This is usually simply called the (log-)profile, but as we will generalize this notion to other number fields than $\mathbb{Q}$ we explicit the number field for unambiguous discussions.

Due to the lattice invariant $\ln\det(\mathcal{L}) = \sum_{i=1}^{n} \ell_i^{\mathbb{Q}}$ and the definition of BKZ reduction, the GSA is equivalent to the following prediction of the *expected* $\mathbb{Q}$*-slope* (e.g., see [AD21; BW25]), defined as

$$\mathrm{slope}_{\mathbb{Q}}(\mathrm{BKZ}^{\beta}) := -\ln \alpha_{\mathbb{Q}}.$$

Heuristic Claim 4.3.2 ($\mathbb{Q}$-profile of BKZ^{β} under GSA)**.** *Let $\mathcal{L}$ be a random n-dimensional Euclidean lattice of fixed determinant, and let $\mathbf{B}$ be a BKZ^{β}-reduced basis of $\mathcal{L}$ for some sufficiently large $\beta \ll n$. Then the GSA predicts*

$$\mathbb{E}[\ell_i^{\mathbb{Q}}] = \frac{n+1-2i}{2} \ln \alpha_{\mathbb{Q}} + \frac{1}{n} \ln\det(\mathcal{L})$$

for all $1 \le i \le n$, where:

$$\ln \alpha_{\mathbb{Q}} = \frac{2}{\beta - 1}\, \mathbb{E}_{\mathbf{s}}[\ln\|\mathbf{s}\|]$$

where $\mathbf{s}$ is a shortest vector in a random normalized projected lattice $\mathcal{L}(\mathbf{B}_{[j:j+\beta-1]})^{(1)}$ for any $1 \le j < n - \beta$.

We recall that, for an n-dimensional lattice $\mathcal{L}$, the normalized lattice $\mathcal{L}^{(1)}$ is defined as $\mathcal{L}^{(1)} = \mathcal{L}/\det(\mathcal{L})^{1/n}$, which ensures that $\mathcal{L}^{(1)}$ has determinant 1.

Further slope prediction under the Gaussian Heuristic

In the analysis of BKZ algorithms, it is standard to assume that the normalized projected lattices $\mathcal{L}(\mathbf{B}_{[j:j+\beta-1]})^{(1)}$ behave as random β-dimensional lattices of unit determinant, allowing to invoke the Gaussian Heuristic to estimate $\mathbb{E}_{\mathbf{s}}[\ln\|\mathbf{s}\|]$.

More precisely, the Gaussian Heuristic states that, for an n-dimensional Euclidean lattice $\mathcal{L}$ and a measurable set $\mathcal{B} \subseteq \mathrm{span}(\mathcal{L})$, the number of nonzero lattice vectors in $\mathcal{L} \cap \mathcal{B}$ approximately equals $\mathrm{vol}(\mathcal{B})/\det(\mathcal{L})$. Applying this heuristic to the n-dimensional Euclidean unit ball $\mathcal{B}_n$ gives a prediction of $\lambda_1(\mathcal{L}) \approx (\det(\mathcal{L})/\mathrm{vol}(\mathcal{B}_n))^{1/n}$ for the first minimum of $\mathcal{L}$.

For a well-defined notion of *random* lattices of unit determinant, there exists a more precise and non-heuristic result regarding the expectation of $\lambda_1(\mathcal{L})$, namely $\lambda_1(\mathcal{L})^n \cdot \mathrm{vol}(\mathcal{B}_n)$ converges in distribution to the exponential distribution of parameter $1/2$ [BSW18, Theorem 1] (attributed to [Söd11; Che13]). As we study the (log-)profile, we are more interested in $\mathbb{E}[\ln \lambda_1(\mathcal{L})]$, but this value can also be

predicted by the aforementioned theorem. Indeed, the logarithm of an exponential distribution can be expressed in terms of a Gumbel distribution, whose mean is known. Note that this non-heuristic result still downgrades to the following *heuristic* when applied to the lattices appearing in BKZ, as their distributions are not yet well understood.

Heuristic 4.3.3 ($\ln \lambda_1$ under the Euclidean-lattice Gaussian Heuristic [.py])**.** *Let $\mathcal{L}$ be a random n-dimensional lattice of unit determinant. Then the Gaussian Heuristic predicts its expected logarithmic first minimum is given by*

$$\mathbb{E}[\ln \lambda_1(\mathcal{L})] = \mathrm{lgh}_{\mathbb{Q}}(n) := \frac{1}{n}\left(\ln \frac{2}{\mathrm{vol}(\mathcal{B}_n)} - \gamma\right) \tag{GH}$$

where $\gamma \approx 0.57721$ denotes the Euler-Mascheroni constant.

This heuristic extends to lattices of arbitrary determinant after appropriate scaling. Namely, given a random n-dimensional lattice $\mathcal{L}$ of fixed determinant D, the Gaussian Heuristic predicts $\mathbb{E}[\ln \lambda_1(\mathcal{L})] = \mathrm{lgh}_{\mathbb{Q}}(n) + \frac{1}{n}\ln D$.

By the Gaussian Heuristic, it is common to heuristically predict the slope of the $\mathbb{Q}$-profile as

$$\mathrm{slope}_{\mathbb{Q}}(\mathrm{BKZ}^{\beta}) = -\frac{2}{\beta - 1}\mathrm{lgh}_{\mathbb{Q}}(\beta).$$

4.3.2 Module-BKZ

The BKZ algorithm merely treats a rank-r module $\mathcal{M} \subseteq K_{\mathbb{R}}^r$ as an rd-dimensional Euclidean lattice, where $d = \deg(K)$, thereby ignoring the underlying module structure. Instead, we consider a generalization of BKZ, which we call $\mathrm{mBKZ}_K^{\beta_K}$ for a parameter $2 \le \beta_K \le r$. This module-BKZ algorithm is presented in Algorithm 4.1 and finds short vectors in the rank-r module using an SVP oracle on rank-β_K projected modules, thereby generalizing module-LLL [LPSW19] to $\beta_K \ge 2$. These projected modules correspond to Euclidean lattices of dimension $\beta_K d$, which makes it natural to compare $\mathrm{mBKZ}_K^{\beta_K}$ to $\mathrm{BKZ}^{\beta_K d}$. The output of Algorithm 4.1 is an $\mathrm{mBKZ}_K^{\beta_K}$-reduced pseudobasis of $\mathcal{M}$, defined as follows.

Definition 4.3.4 ($\mathrm{mBKZ}_K^{\beta_K}$-reduced)**.** For $2 \le \beta_K \le r$, we say that a pseudobasis $\mathfrak{B}$ of a rank-r module $\mathcal{M}$ is $\mathrm{mBKZ}_K^{\beta_K}$-reduced if

$$\|\mathbf{b}_i^*\| = \lambda_1(\mathcal{M}(\mathfrak{B}_{[i:\min(i+\beta_K-1,r)]})) \qquad \forall 1 \le i \le r.$$

Algorithm 4.1: Module-BKZ algorithm (high-level)

Input: Unital pseudobasis $\mathfrak{B}$ of a rank-r module $\mathcal{M}$
Parameter $\beta_K \leq r$

Output: Unital $\mathrm{mBKZ}_K^{\beta_K}$-reduced pseudobasis

1: **while** $\mathfrak{B}$ is not $\mathrm{mBKZ}_K^{\beta_K}$-reduced **do**
2: **for** $i = 1, \ldots, r$ **do**
3: Let $\mathbf{v} = \mathrm{SVP}(\mathcal{M}(\mathfrak{B}_{[i:\min(i+\beta_K-1,r)]}))$
4: Let $\mathfrak{I}$ be such that $\mathbf{v}\mathfrak{I} = \mathbf{v}K \cap \mathcal{M}(\mathfrak{B}_{[i:\min(i+\beta_K-1,r)]})$
5: Lift $\mathbf{v}\mathfrak{I}$ to $\mathbf{w}\mathfrak{I} \subseteq \mathcal{M}$ satisfying $\pi_i(\mathbf{w}) = \mathbf{v}$
6: Insert $(\mathbf{w}, \mathfrak{I})$ into $\mathfrak{B}$ at position i and remove linear dependencies
7: **return** $\mathfrak{B}$

See, e.g., [BW25, Sec. 2.8] for a similar pseudocode description of the unstructured BKZ algorithm.

We refer to a single execution of the while-loop as an *mBKZ tour*. Line 5 is essentially Babai lifting [Bab86], and Line 6 can be achieved (for instance) using the algorithm from [FS10, Theorem 4]. While Algorithm 4.1 may encounter non-unital pseudobases during an mBKZ tour, the final output is always unital by construction. We remark that formal variants of LLL and BKZ typically consider an *approximate* SVP oracle to enforce significant improvement at each insertion step. As we are not concerned with guaranteed termination, our model instead uses an exact SVP oracle in Line 3. This approach assumes approximate convergence after sufficiently many mBKZ tours, which is at least heuristically supported by [HPS11].

Remark 4.3.5. Note that the fractional ideal $\mathfrak{I}$ considered in Line 4 contains $\mathcal{O}_K$ and is the ideal of minimal norm that satisfies $\mathbf{v}\mathfrak{I} \subseteq \mathcal{M}(\mathfrak{B}_{[i:\min(i+\beta_K-1,n)]})$. Indeed, consider a vector $\mathbf{v} \in \mathcal{M}'$ for some module $\mathcal{M}'$, and let $\mathfrak{I}$ be the fractional ideal such that $\mathbf{v}\mathfrak{I} = \mathbf{v}K \cap \mathcal{M}'$. Then $\mathfrak{I} = \{x \in K : x\mathbf{v} \in \mathcal{M}'\}$. Since $x\mathbf{v} \in \mathcal{M}'$ for all $x \in \mathcal{O}_K$, we have that $\mathcal{O}_K \subseteq \mathfrak{I}$, so $\mathrm{N}(\mathfrak{I}) \leq 1$. Moreover, for all fractional ideals $\mathfrak{I}'$ such that $\mathbf{v}\mathfrak{I}' \subseteq \mathcal{M}'$, we have $\mathfrak{I}' \subseteq \mathfrak{I}$, and thus $\mathrm{N}(\mathfrak{I}) \leq \mathrm{N}(\mathfrak{I}')$.

4.3.3 Implementation

For our experiments, we developed a publicly available Python implementation of module-BKZ for cyclotomic fields, together with the experimentation and prediction scripts, and the generated data [.py]. Instead of redeveloping the entire lattice-reduction stack to obtain one specialized to module lattices, we rely on existing libraries `fplll/fpylll` [The23] and `G6K` [ADHK+19]. When implementing the SVP oracle from G6K [.py], we choose less aggressive strategies than typically done, allowing us to be almost certain that we have really solved SVP, as our purpose is more to understand and predict than to fine-tune for speed.

The module lattices considered in our experiments are natural $\mathcal{O}_K$-analogs of q-ary lattices, namely lattices defined by random linear equations modulo $q\mathcal{O}_K$ [.py], though our implementation is not restricted to those.

The choice of using existing lattice-reduction libraries raises two technical difficulties: protecting the module structure from the whims of libraries that are designed to only consider $\mathbb{Z}$-bases, and having a geometrically faithful embedding of the cyclotomic rings that remains integral. We discuss our solutions to both issues.

Restructured $\mathbb{Z}$-bases. Our objective is to represent a module lattice $\mathcal{M}$ by a $\mathbb{Z}$-basis, while maintaining enough of its structure. Let $(\mathbf{b}_1, \ldots, \mathbf{b}_{rd}) \in K_{\mathbb{R}}^{r \times rd}$ be a $\mathbb{Z}$-basis of $\mathcal{M}$ at some stage of the algorithm. Naively, one would be tempted to enforce that, for all blocks i, the corresponding vectors $\mathbf{b}_{(i-1)d+1}, \ldots, \mathbf{b}_{(i-1)d+d}$ $\mathbb{Z}$-generate a rank-1 module. This constraint may be broken when applying size reduction, the most frequent basis maintenance task of the lattice-reduction library. Size reduction can hardly be avoided for reasons of numerical stability, and replacing it by a module-lattice-analog of size reduction would require significant modifications in `fplll`.

Instead, it suffices to impose the constraint that $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is indeed the $\mathbb{Z}$-basis of a rank-1 module [.py], and that this property holds recursively for the rest of the basis (i.e., $\mathbf{b}_{d+1}, \ldots, \mathbf{b}_{rd}$) projected orthogonally to $\mathbf{b}_1, \ldots, \mathbf{b}_d$. This property is not affected by size reduction, and is sufficient to implement module-BKZ.

This structure is going to be broken whenever we solve SVP on a block by `G6K`. We now want a procedure to repair the module structure; more precisely, given a certain first vector $\mathbf{b}_1$, we would like to enforce that $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is a $\mathbb{Z}$-basis of $\mathbf{b}_1 K \cap \mathcal{M}$ [.py], which is a rank-1 module containing $\mathbf{b}_1\mathcal{O}_K$. Hence, we first insert a $\mathbb{Z}$-basis of $\mathbf{b}_1\mathcal{O}_K$ into the current $\mathbb{Z}$-basis of $\mathcal{M}$, and would then like to run LLL to eliminate linear dependencies [.py].

Yet, LLL may find shorter vectors for this block and thus re-break the module structure of the first block: the first d vectors may not $\mathbb{Z}$-generate $\mathbf{b}_1 K \cap \mathcal{M}$. In principle, one can prevent LLL from doing just that while still eliminating linear dependencies, simply by running LLL (in Line 6) with parameter δ very close to 0, so that vectors are less likely to be swapped.

Unfortunately, the `fplll` API forbids setting $\delta \leq 1/4$; the reason might be that there is no absolute guarantee in terms of basis quality for such small δ. Yet, for such parameters LLL still guarantees that the potential does not increase and that linear dependencies are eliminated. We argue that it would be desirable to relax the API.

Fortunately, we do not have to resort to patching `fplll` to get our prototype running, at least for some cyclotomic rings of interest (say with conductor ≤ 16); we found out experimentally that repeated attempts to restructure, with decreasing δ from 0.99 would eventually succeed before reaching $1/4$ [.py]. Running BKZ progressively with increasing blocksize also seems to help in avoiding the issue.

Cyclic embedding. A second technical difficulty with our approach comes from the definition of inner products. Our implementation considers module lattices defined over a cyclotomic field $K = \mathbb{Q}(\omega_c)$, so we are working in the cyclotomic ring $\mathcal{O}_K = \mathbb{Z}[\omega_c]$ with the trace inner product. However, the `fplll` library requires the input lattice to be represented with integer coefficients, and considers the standard inner product over $\mathbb{R}^n$.

The cyclotomic field $K = \mathbb{Q}(\omega_c)$ satisfies $K \cong \mathbb{Q}[X]/\Phi_c(X)$ for the c-th cyclotomic polynomial $\Phi_c(X) = \prod_j (X - \omega_c^j)$, with j ranging over all $j \in [c]$ with $\gcd(j, c) = 1$. If we represent elements of $\mathbb{Z}[\omega_c] \cong \mathbb{Z}[X]/\Phi_c(X)$ using the coefficients of a polynomial $\sum_{i=0}^{c-1} a_i X^i$, then the inner product does not correspond to the desired trace inner product. We could represent a vector by its embeddings to have the correct inner product, but those values are defined over $\mathbb{C}$ and thus may be irrational. The following *cyclic embedding* solves this conundrum; an idea that already underpins several existing works [DD12; BDF18; DP16].

Consider the cyclic ring $\mathcal{C}_c = \mathbb{R}[X]/(X^c - 1)$, and view it as a Euclidean vector space in the standard way. That is, elements of $\mathcal{C}_c$ are polynomials $\sum_{i=0}^{c-1} a_i X^i$ represented by their coefficients a_i, naturally leading to the *coefficient inner product* $\langle \sum_{i=0}^{c-1} a_i X^i, \sum_{i=0}^{c-1} b_i X^i \rangle_{\text{coef}} := \sum_{i=0}^{c-1} a_i b_i$. The discrete Fourier transform F_c sends such a polynomial $P(X) = \sum_{i=0}^{c-1} a_i X^i$ (more precisely, its coefficients) to the vector $(P(\omega_c^j))_{j=0}^{c-1} \in \mathbb{C}^c$. It is well known that its scaling $\frac{1}{\sqrt{c}} F_c$ is an isometry.

Similarly, the trace inner product of $\mathbb{Q}(\omega_c)$ is isometric to the standard inner product of $\mathbb{C}^{\phi(n)}$ via the canonical embedding. The latter Hermitian space is trivially embedded into $\mathbb{C}^c$ by padding $c - \phi(c)$ many zero coordinates at positions corresponding to the non-primitive c-th roots of unity. This allows us to define the cyclic embedding by completing a commutative diagram, as depicted in Figure 4.3 [.py].

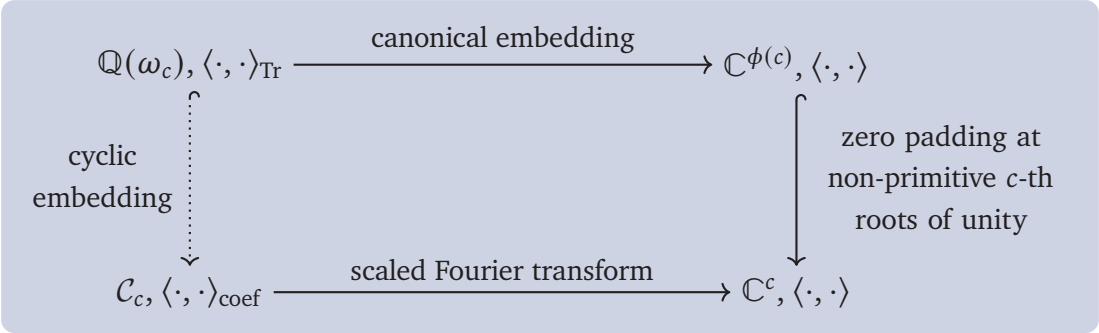


Figure 4.3: Commutative diagram defining the cyclic embedding of the cyclotomic field $\mathbb{Q}(\omega_c)$ into the cyclic ring $\mathcal{C}_c := \mathbb{R}[X]/(X^c - 1)$ via an isometric map.

One further notices that the ring of integers $\mathbb{Z}[\omega_c] \subseteq \mathbb{Q}(\omega_c)$ is represented by elements of $\frac{1}{\sqrt{c}}\mathbb{Z}[X]/(X^c - 1) \subseteq \mathcal{C}_c$ under this isometric map. Indeed, the cyclic embedding of $x \in \mathbb{Z}[\omega_c]$ is the polynomial $\sum_{i=0}^{c-1} a_i X^i \in \mathcal{C}_c$ with coefficients

$$a_i = \frac{1}{\sqrt{c}} \sum_{\sigma \in \mathcal{E}} \sigma(x)\sigma(\omega_c^{-i}) = \frac{1}{\sqrt{c}} \sum_{\sigma \in \mathcal{E}} \sigma(x\omega_c^{-i}) = \frac{1}{\sqrt{c}} \mathrm{Tr}(x\omega_c^{-i}).$$

As both x and ω_c lie in $\mathbb{Z}[\omega_c]$, we have $x\omega_c^{-i} \in \mathbb{Z}[\omega_c]$, so its trace lies in $\mathbb{Z}$. Hence, multiplying the cyclic embeddings of elements of $\mathbb{Z}[\omega_c]$ by a factor of $\sqrt{c}$ results in a scaled isometric embedding of $\mathbb{Z}[\omega_c]$ into $\mathbb{Z}^c$, a space that is compatible with libraries such as `fplll`. Note that the embedding dimension c is strictly larger than the degree $\phi(c)$ of the number field, but this is not an issue as the aforementioned lattice-reduction library is perfectly capable of working with lattices embedded in a vector space of larger dimension.

4.4 Prediction of the module-BKZ slope

In this section, we present a slope prediction for (structured) module-BKZ, motivated by numerical observations and supported by theoretical analysis, which makes use of a module-lattice analog of the Geometric Series Assumption (GSA).

Let K be a number field of degree d and discriminant Δ_K. Consider a random rank-r module $\mathcal{M}$ over K of fixed determinant, and a sufficiently large β_K satisfying $2 \leq \beta_K \ll r$. Viewing $\mathcal{M}$ as an rd-dimensional Euclidean lattice $\mathcal{L}$, the analysis from Section 4.3.1 (using the GSA) implies that the unstructured BKZ$^{\beta_K d}$ algorithm results in a slope prediction of

$$\mathrm{slope}_{\mathbb{Q}}(\mathrm{BKZ}^{\beta_K d}) = -\frac{2}{\beta_K d - 1}\mathbb{E}_{\mathbf{s}}[\ln\|\mathbf{s}\|] \tag{4.2}$$

where the random variable $\mathbf{s} \in \mathbb{R}^{\beta_K d}$ reflects the behavior of BKZ. Specifically, $\mathbf{s}$ is a shortest vector in a random $\beta_K d$-dimensional lattice of unit determinant. We recall from Section 4.3.1 that by invoking the Gaussian Heuristic one can predict $\mathbb{E}_s[\ln\|\mathbf{s}\|]$ as $\mathrm{lgh}_{\mathbb{Q}}(\beta_K d)$, as is common in the slope analysis of unstructured BKZ.

In Section 4.4.1, we use a module-lattice analog of the GSA to derive the following slope prediction [.py] for the mBKZ$_K^{\beta_K}$ algorithm:

$$\begin{aligned}\mathrm{slope}_{\mathbb{Q}}(\mathrm{mBKZ}_K^{\beta_K}) &= -\frac{2}{\beta_K d^2 - d^2}\,\mathbb{E}_{\mathbf{s},\mathfrak{I}}[\ln\det(\mathbf{s}\mathfrak{I})] \\ &= -\frac{2}{\beta_K d - d}\Bigg(\underbrace{\mathbb{E}_{\mathbf{s}}[\ln\|\mathbf{s}\|]}_{t_1} + \underbrace{\frac{1}{2d}\ln\frac{|\Delta_K|}{d^d}}_{t_2} \\ &\qquad + \underbrace{\mathbb{E}_{\mathbf{s}}\left[\ln\frac{\sqrt{d}\,\mathrm{N}(\mathbf{s})^{1/d}}{\|\mathbf{s}\|}\right]}_{t_3} + \underbrace{\frac{1}{d}\mathbb{E}_{\mathfrak{I}}[\ln\mathrm{N}(\mathfrak{I})]}_{t_4}\Bigg)\end{aligned} \tag{4.3}$$

where the random variables $\mathbf{s} \in K_{\mathbb{R}}^{\beta_K}$ and $\mathfrak{I} \supseteq \mathcal{O}_K$ reflect the behavior of module-BKZ. Specifically, $\mathbf{s}$ is a shortest vector in a random rank-β_K module lattice of unit determinant, and $\mathfrak{I}$ is a fractional ideal in the random mBKZ-reduced pseudo-basis.[3]

[3] As is standard in the analysis of BKZ algorithms, these distributions are not formally defined, which is why most of our analysis remains heuristic (recall Remark 4.2.1).

Apart from the terms t_1, t_2, t_3, t_4, the slope predictions of unstructured BKZ in Equation (4.2) and structured module-BKZ in Equation (4.3) also differ in the denominator, changing from $\beta_K d - 1$ to $\beta_K d - d$.

After deriving this slope prediction, we dive into the four main terms t_1, t_2, t_3, t_4, enabling us to compare with the usual BKZ slope prediction from Equation (4.2). We refer to these four terms as:

t_1: the module-lattice analog of $\mathrm{lgh}_{\mathbb{Q}}(\beta_K d)$

t_2: the discriminant gap

t_3: the skewness gap

t_4: the index gap

The discriminant gap t_2 is fully determined by K, and for cyclotomic fields we have $t_2 \leq 0$. While the other terms also depend on the pseudobasis at hand, they can be upper bounded independent of it: $t_1 \leq \mathrm{lgh}_{\mathbb{Q}}(\beta_K d) + \ln(2)$ by Minkowski's bound, $t_3 \leq 0$ by the inequality of arithmetic and geometric means, and $t_4 \leq 0$ because we consider only unital pseudobases. Sections 4.4.2 to 4.4.5 are aimed at providing an estimate for those terms, rather than just a generic bound.

In Section 4.4.6, we use our analysis of the four terms in order to conclude with an upper and lower bound on the module-BKZ $\mathbb{Q}$-slope prediction (Equation (4.3)), yielding the prediction interval shown in Figure 4.1.

Remark 4.4.1 (Explanation of terminology). Since the term $\sqrt{d}\,\mathrm{N}(\mathbf{s})^{1/d}/\|\mathbf{s}\|$ equals 1 if and only if the $|\sigma(\mathbf{s})|^2$ are identical for all embeddings $\sigma \in \mathcal{E}$, the quantity t_3 measures, in some sense, how *skewed* these values are (with $t_3 < 0$ corresponding to skewness). The term t_4 is defined by the norm of ideals $\mathfrak{I} \supseteq \mathcal{O}_K$, which satisfy $\mathrm{N}(\mathfrak{I}) = 1/\mathrm{N}(\mathfrak{I}^{-1})$ where $\mathrm{N}(\mathfrak{I}^{-1})$ equals the *index* of $\mathfrak{I}^{-1}$ as a subgroup of $\mathcal{O}_K$.

4.4.1 Module-lattice Geometric Series Assumption (GSA)

Consider a pseudobasis $((\mathbf{b}_i, \mathfrak{I}_i))_{i=1}^r$ of a rank-r module $\mathcal{M}$ over a number field K of degree d. For each component $(\mathbf{b}_i, \mathfrak{I}_i)$ of the pseudobasis, the canonical embedding of K yields a $\mathbb{Z}$-basis

$$\mathbf{z}_{(i-1)d+1}, \mathbf{z}_{(i-1)d+2}, \ldots, \mathbf{z}_{(i-1)d+d} \in \mathbb{R}^{rd}$$

of the d-dimensional Euclidean lattice associated with $\mathbf{b}_i\mathfrak{I}_i$, see Figure 4.4.

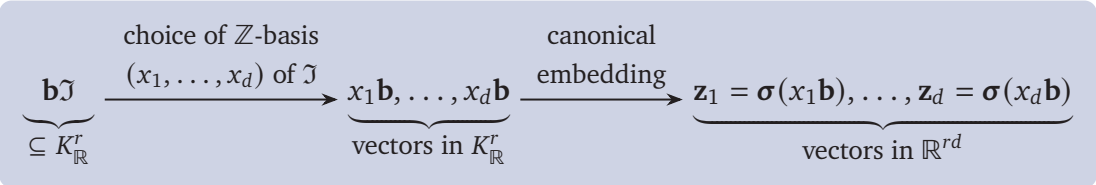


Figure 4.4: Given $\mathbf{b} \in K_{\mathbb{R}}^r$ and a $\mathbb{Z}$-basis $(x_1, \ldots, x_d)$ of a fractional $\mathcal{O}_K$-ideal $\mathfrak{I}$, the canonical embedding $\boldsymbol{\sigma}$ yields a $\mathbb{Z}$-basis of the d-dimensional Euclidean lattice associated with the rank-1 module $\mathbf{b}\mathfrak{I}$. The determinant of this lattice is fixed and independent of the choice of $(x_1, \ldots, x_d)$.

This results in a $\mathbb{Z}$-basis $\mathbf{z}_1, \ldots, \mathbf{z}_{rd}$ of the rd-dimensional Euclidean lattice associated with $\mathcal{M}$. The $\mathbb{Q}$-*profile* corresponding to this basis (recall Section 4.3.1) is defined as the sequence $(\ell_i^{\mathbb{Q}})_{i=1}^{rd}$ where:

$$\ell_i^{\mathbb{Q}} := \ln\|\mathbf{z}_i^*\| \qquad \forall 1 \le i \le rd.$$

In addition, we define the corresponding *K-profile* as the sequence $(\ell_i^K)_{i=1}^r$ where:

$$\ell_i^K := \ln\det(\mathbf{b}_i^*\mathfrak{I}_i) = \sum_{j=1}^{d} \ell_{(i-1)d+j}^{\mathbb{Q}} \qquad \forall 1 \le i \le r.$$

In particular, $\sum_{i=1}^r \ell_i^K = \ln\det(\mathcal{M})$, and each ℓ_i^K/d denotes the average log-norm of the GSO vectors $\mathbf{z}_{(i-1)d+1}^*, \mathbf{z}_{(i-1)d+2}^*, \ldots, \mathbf{z}_{(i-1)d+d}^*$ corresponding to $\mathbf{b}_i\mathfrak{I}_i$. The value ℓ_i^K is independent of the choice of $\mathbb{Z}$-basis for $\mathfrak{I}_i$.

Module-lattice analog of the GSA

Figure 4.5a illustrates the K-profiles of mBKZ-reduced bases of module lattices over the cyclotomic fields $\mathbb{Q}(\omega_3)$, $\mathbb{Q}(\omega_4)$, $\mathbb{Q}(\omega_{15})$, and $\mathbb{Q}(\omega_{16})$, compared with $\mathbb{Q}$, the unstructured case. A first observation is that the K-profile of a mBKZ-reduced basis appears to resemble a descending straight line, i.e., the GSA (Heuristic 4.3.1) seems to generalize to these K-profiles, up to a tail phenomenon that is well known for unstructured BKZ [AD21, Def. 9]. However, we notice that the module-BKZ slope varies: it is better (flatter) for $\mathbb{Q}(\omega_3)$ and $\mathbb{Q}(\omega_{15})$, and worse for $\mathbb{Q}(\omega_4)$ and $\mathbb{Q}(\omega_{16})$.[4]

Moreover, we observe various periodic patterns in the corresponding $\mathbb{Q}$-profiles shown in Figure 4.5b. These repeated patterns correspond to the profile of a (reduced) $\mathbb{Z}$-basis of $\mathcal{O}_K$, or a small distortion thereof. In the case of $\mathbb{Q}(\omega_4)$, $\mathcal{O}_K \cong \mathbb{Z}^2$ (up to scaling), and we have $\|\mathbf{b}_1\| = \|\mathbf{b}_2^*\| = 1$: the pattern is perfectly flat on each period. For $\mathbb{Q}(\omega_3)$, $\mathcal{O}_K$ is a scaled hexagonal lattice, with $\|\mathbf{b}_1\| = 1$, $\|\mathbf{b}_2^*\| = \frac{\sqrt{3}}{2}$. For $\mathbb{Q}(\omega_{16})$, we have $\mathcal{O}_K \cong \mathbb{Z}^8$, but the pattern is not exactly flat nor perfectly periodic: $\mathcal{O}_K$ seems to be slightly and randomly *skewed*.

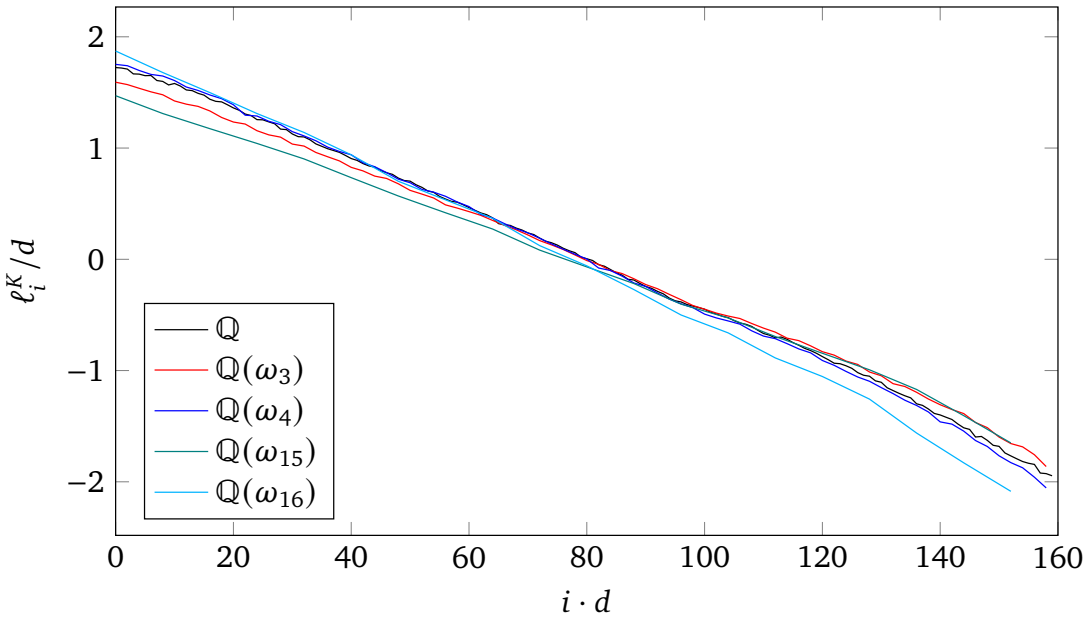
These observations regarding Figure 4.5a motivate the following module-lattice analog of the GSA, as already proposed for module-LLL in [KK24, Heuristic 3].[5]

> **Heuristic 4.4.2** (Module-lattice GSA)**.** *Let K be a number field and let $\beta_K \ll r$ be sufficiently large. There is a constant $\alpha_K > 1$ (depending on β_K) such that the K-profile of an $\mathrm{mBKZ}_K^{\beta_K}$-reduced pseudobasis of a random rank-r module $\mathcal{M}$ over K of fixed determinant satisfies*
>
> $$\mathbb{E}[\ell_i^K] = \mathbb{E}[\ell_1^K] - (i-1)\ln \alpha_K \qquad \forall 1 \le i \le r. \tag{mGSA}$$

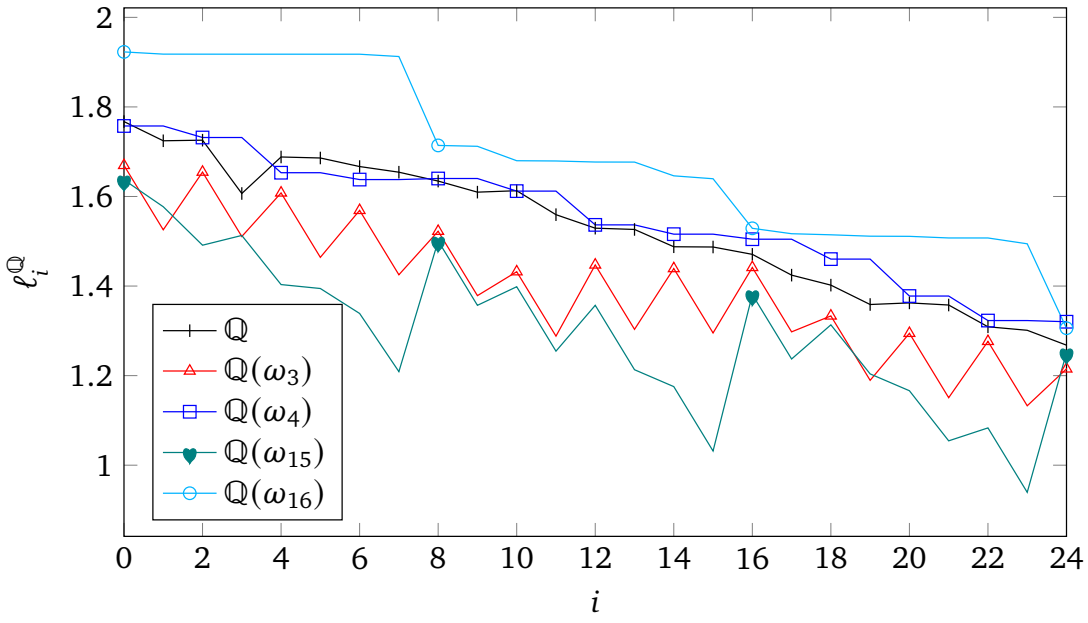
We recall that $\ell_i^K = \ln \det(\mathbf{b}_i^* \mathfrak{I}_i) = \frac{1}{2}\ln|\Delta_K| + \ln \mathrm{N}(\mathbf{b}_i^*) + \ln \mathrm{N}(\mathfrak{I}_i)$ for $1 \le i \le r$.

[4] This discrepancy can be explained by the difference in discriminant gap: cyclotomic fields K with a power-of-two conductor c have $|\Delta_K| = \phi(c)^{\phi(c)}$, whereas a strict inequality holds for conductors $c = 3$ and $c = 15$, contributing to a flatter profile. (See Section 4.4.3.)

[5] Technically, the heuristic in [KK24] considers module-LLL using an algebraic SVP oracle, minimizing the *algebraic* norm of the basis vectors, whereas our definition of module-BKZ deals with minimizing the Euclidean norm. See also Open Question 6.



(a) Plot of the normalized K-profile $(\ell_i^K)_{i=1}^r$, averaged over 5 bases.



(b) Plot of $\mathbb{Q}$-profiles $(\ell_i^{\mathbb{Q}})_{i=1}^{rd}$, zoomed in.

Figure 4.5: The profiles resulting from $\text{mBKZ}_K^{\beta_K}$ in dimension $rd = 160$ with block-size $\beta_K d = 64$ after 30 tours, where $d = \deg(K)$ and $K = \mathbb{Q}(\omega_c)$ for $c = 1, 3, 4, 15, 16$ [.py]. The respective degrees are $d = 1, 2, 2, 8, 8$, and their integer multiples are marked in part (b).

Slope prediction under the module-lattice GSA

Similar to the unstructured case (Section 4.3.1), the module invariant $\det(\mathcal{M})$ and the definition of mBKZ reduction allow us to translate Heuristic 4.4.2 into the following prediction of the *expected K-slope*, $\mathrm{slope}_K(\mathrm{mBKZ}_K^{\beta_K}) := -\ln \alpha_K$ [.py].

Heuristic Claim 4.4.3 (K-profile of $\mathrm{mBKZ}_K^{\beta_K}$ under mGSA)**.** *Let K be a number field of degree d. Let $\mathcal{M}$ be a random rank-r module over K of fixed determinant, and let $\mathfrak{B} = ((\mathbf{b}_i, \mathfrak{I}_i))_{i=1}^r$ be an $\mathrm{mBKZ}_K^{\beta_K}$-reduced pseudobasis of $\mathcal{M}$ for some sufficiently large $\beta_K \ll r$. Then the module-lattice GSA predicts*

$$\mathbb{E}[\ell_i^K] = \frac{r+1-2i}{2} \ln \alpha_K + \frac{1}{r} \ln \det(\mathcal{M})$$

for all $1 \le i \le r$, where:

$$\ln \alpha_K = \frac{2}{\beta_K - 1} \mathbb{E}_{\mathbf{s},\mathfrak{I}}[\ln \det(\mathbf{s}\mathfrak{I})]$$

where $\mathbf{s}$ is a shortest vector in a random normalized projected module $\mathcal{M}(\mathfrak{B}_{[j:j+\beta_K-1]})^{(1)}$ and $\mathfrak{I}$ the corresponding fractional ideal $\mathfrak{I}_j \supseteq \mathcal{O}_K$, for any $1 \le j < r - \beta_K$.

Remark 4.4.4 (Prediction of the $\mathbb{Q}$-slope). While the Euclidean-lattice GSA (Heuristic 4.3.1) may not hold locally, it is globally plausible and allows us to translate the slope of the K-profile back to that of the corresponding $\mathbb{Q}$-profile. Namely, we predict:

$$\mathrm{slope}_{\mathbb{Q}}(\mathrm{mBKZ}_K^{\beta_K}) = \frac{1}{d^2} \mathrm{slope}_K(\mathrm{mBKZ}_K^{\beta_K}) = -\frac{1}{d^2} \ln \alpha_K$$

resulting in the slope prediction given in Equation (4.3). Note that the first equality holds under Heuristic 4.3.1: $\ln \alpha_K = \ell_1^K - \ell_2^K = \sum_{j=1}^d \ell_j^{\mathbb{Q}} - \sum_{j=1}^d \ell_{d+j}^{\mathbb{Q}} = d^2 \ln \alpha_{\mathbb{Q}}$, where we use Equation (mGSA), the definition of the ℓ_i^K's, and Equation (GSA).

Justification of Heuristic Claim 4.4.3. Let $\mathfrak{B} := ((\mathbf{b}_i, \mathfrak{I}_i))_{i=1}^{r}$ be the random pseudobasis output by module-BKZ. We first show that the formula for $\ln \alpha_K$ follows from Heuristic 4.4.2. Fix arbitrary $1 \le j < r - \beta_K$, and define

$$D_j := \det(\mathcal{M}(\mathfrak{B}_{[j:j+\beta_K-1]})).$$

By definition of the K-profile, applying Equation (mGSA) to all $i \in [\beta_K]$ gives

$$\frac{1}{\beta_K}\mathbb{E}[\ln(D_j)] = \frac{1}{\beta_K}\sum_{i=0}^{\beta_K-1}\mathbb{E}[\ell_{j+i}^K] = \mathbb{E}[\ell_j^K] - \frac{\beta_K - 1}{2}\ln \alpha_K.$$

Since $\ell_j^K = \ln\det(\mathbf{b}_j^*\mathfrak{I}_j) = \ln\det(\mathbf{s}\mathfrak{I}_j) + \frac{1}{\beta_K}\ln(D_j)$ for $\mathbf{s} := \mathbf{b}_j^*/D_j^{1/\beta_K d}$ (where we use $\mathrm{N}(D_j^{1/\beta_K d}) = D_j^{1/\beta_K}$), we obtain $\ln \alpha_K = \frac{2}{\beta_K - 1}\mathbb{E}[\ln\det(\mathbf{s}\mathfrak{I}_j)]$. Note that, by definition of mBKZ reduction, $\mathbf{s}$ is a shortest vector in the normalized projected module lattice $\mathcal{M}(\mathfrak{B}_{[j:j+\beta_K-1]})^{(1)}$, and $\mathfrak{I}_j \supseteq \mathcal{O}_K$ a fractional ideal.

To conclude, we apply Equation (mGSA) to all $i \in [r]$, giving

$$\ln\det(\mathcal{M}) = \sum_{i=1}^{r}\mathbb{E}[\ell_i^K] = r\mathbb{E}[\ell_1^K] - \frac{r(r-1)}{2}\ln \alpha_K.$$

Hence, $\mathbb{E}[\ell_1^K] = \frac{r-1}{2}\ln \alpha_K + \frac{1}{r}\ln\det(\mathcal{M})$, so Equation (mGSA) implies

$$\mathbb{E}[\ell_i^K] = \frac{r+1-2i}{2}\ln \alpha_K + \frac{1}{r}\ln\det(\mathcal{M})$$

for all $i \in [r]$, as desired. □

4.4.2 Module-lattice analog of the Gaussian Heuristic

We approximate the first term $t_1 = \mathbb{E}_{\mathbf{s}}[\ln\|\mathbf{s}\|]$ in Equation (4.3) by extending the Gaussian Heuristic to module lattices. This is justified if the first projected module lattice of an $\mathrm{mBKZ}_K^{\beta_K}$-reduced pseudobasis behaves like a random rank-β_K module lattice (of the same volume).

Specifically, let K be a number field of degree d. For a positive integer r, we write $\mathrm{lgh}_K(r)$ for the expected logarithmic first minimum of a random rank-r module lattice over K of unit determinant. In the general case, Minkowski's bound allows us to prove $\mathrm{lgh}_K(r) \le \mathrm{lgh}_{\mathbb{Q}}(rd) + \ln(2)$ for any number field K. In the case of a cyclotomic field $K = \mathbb{Q}(\omega_c)$ of conductor c and degree $d = \phi(c)$, a recent study [GSVV24, Theorem 38] proved that the first minimum of a (formally

well-defined) random rank-r module lattice of unit determinant is asymptotically concentrated around $(\mu_K/\mathrm{vol}(\mathcal{B}_{rd}))^{1/rd}$, where we recall that μ_K denotes the number of roots of unity in K, and $\mathcal{B}_{rd}$ the rd-dimensional Euclidean unit ball.[6]

While this does not predict the exact average, we can attempt to make such a prediction by heuristically "merging" [GSVV24, Theorem 38] and Heuristic 4.3.3. Namely, we use the latter as a baseline for $\mathbb{Q}$, and the former to estimate the gap between $\mathbb{Q}$ and K. Because $\mu_{\mathbb{Q}} = 2$, one reaches the following heuristic.

Heuristic 4.4.5 ($\ln \lambda_1$ under the module-lattice Gaussian Heuristic [.py])**.** *Let K be a number field of degree d. Let $\mathcal{M}$ be a random rank-r module lattice of unit determinant over K. Then the module-lattice Gaussian Heuristic predicts its expected logarithmic first minimum is given by*

$$\mathbb{E}\left[\ln \lambda_1(\mathcal{M})\right] = \mathrm{lgh}_K(r) \coloneqq \mathrm{lgh}_{\mathbb{Q}}(rd) + \frac{1}{rd}\ln\frac{\mu_K}{2}. \tag{mGH}$$

Remark 4.4.6. Similar to Heuristic 4.3.3, the above heuristic extends to random module lattices of arbitrary fixed determinant by appropriate scaling.

This results in the following heuristic estimate:

$$t_1 = \mathrm{lgh}_K(\beta_K) = \mathrm{lgh}_{\mathbb{Q}}(\beta_K d) + \frac{1}{\beta_K d}\ln\frac{\mu_K}{2} = \frac{1}{\beta_K d}\left(\ln\frac{\mu_K}{\mathrm{vol}(\mathcal{B}_{\beta_K d})} - \gamma\right)$$

where γ denotes the Euler-Mascheroni constant.

We verify this prediction experimentally in Figure 4.6, and note it to be rather accurate even for relatively small module rank r, and increasingly accurate as r grows. We note that even for $K = \mathbb{Q}$ the average is slightly underestimated, but the theorem on which we base the module-lattice Gaussian Heuristic only provides a concentration bound, not an expectation. For $K = \mathbb{Q}$, there also exists a refined estimate for the average [Che13; BSW18], and it would be interesting to extend it to other number fields.

[6] We also considered the simpler, specialized Theorem 3 of [GSVV24], but found that it yields poor predictions for even conductors c. Recall $\mu_K = 2c$ for odd c and $\mu_K = c$ otherwise.

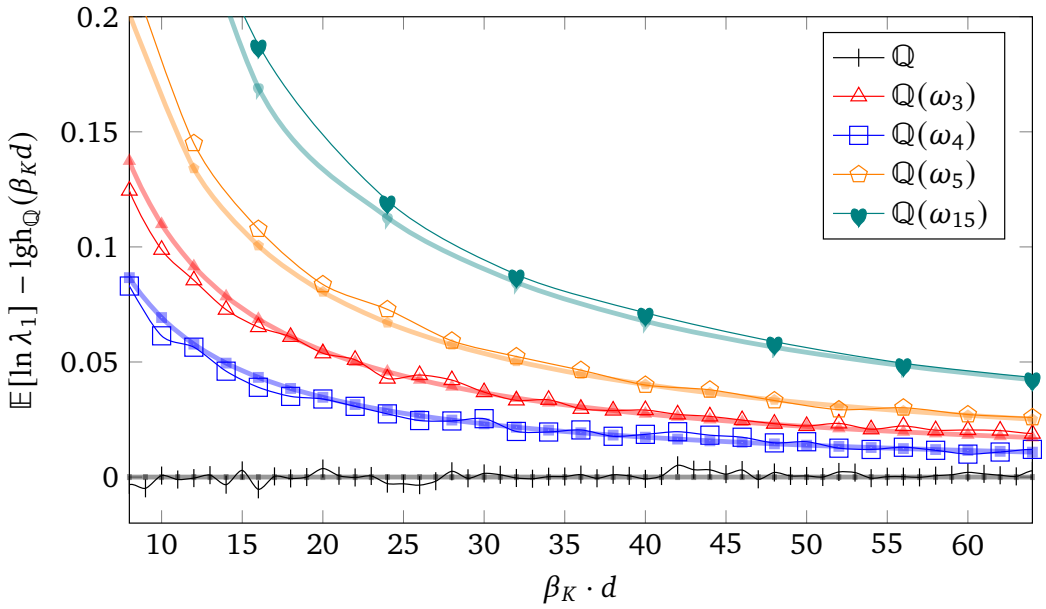


Figure 4.6: Logarithmic gap to the Gaussian Heuristic over $\mathbb{Q}$ ($\mathrm{lgh}_{\mathbb{Q}}(\beta_K d)$). Thick lines with small marks are predictions ($\mathrm{lgh}_K(\beta_K)$), thin lines with large marks are experimental average, taken over 1000 samples [.py].

4.4.3 Discriminant gap

The next term in the formula for $\mathrm{slope}_{\mathbb{Q}}(\mathrm{mBKZ}_K^{\beta})$ is the discriminant gap $t_2 = \frac{1}{2d}\ln\left(\frac{|\Delta_K|}{d^d}\right)$. Ignoring other terms, a smaller discriminant brings the predicted slope closer to 0, thereby contributing to a flatter module-BKZ profile. In the case that K is a cyclotomic field, we have the following explicit formula for its discriminant Δ_K, allowing us to compute the discriminant gap. For $c \in \mathbb{N}$, we write $\mathcal{P}_c$ for the set of distinct prime factors dividing c.

Lemma 4.4.7 (Discriminant of cyclotomic fields [Was82, Prop. 2.7] [.py])**.** *For $c \in \mathbb{N}$, let ω_c be a primitive c-th root of unity. The discriminant Δ_K of $K = \mathbb{Q}(\omega_c)$ equals $\Delta_K = (-1)^{\frac{\phi(c)}{2}} c^{\phi(c)} \prod_{p\in\mathcal{P}_c} p^{-\frac{\phi(c)}{p-1}}$.*

We obtain the following formula for the discriminant gap t_2 of cyclotomic fields, revealing that t_2 merely depends on the set $\mathcal{P}_c$ for the field's conductor c.

Lemma 4.4.8 (Discriminant gap formula). *For $c \in \mathbb{N}$, let ω_c be a primitive c-th root of unity. Then the discriminant gap t_2 of $K = \mathbb{Q}(\omega_c)$ is independent of the exponents in the prime decomposition of c, and equals:*

$$t_2 = \frac{1}{2} \sum_{p \in \mathcal{P}_c} \left(\frac{p-2}{p-1} \ln(p) - \ln(p-1) \right).$$

Proof. For $c \in \mathbb{N}$, let ω_c be a primitive c-th root of unity, and define $K := \mathbb{Q}(\omega_c)$ and $d := \phi(c)$. By Lemma 4.4.7, $\ln(|\Delta_K|) = d(\ln(c) - \sum_{p \in \mathcal{P}_c} \frac{1}{p-1} \ln(p))$, so by definition the discriminant gap t_2 of K equals $t_2 = \frac{1}{2d} \ln(|\Delta_K|) - \frac{1}{2} \ln(d) = \frac{1}{2}(\ln(c) - \ln(d) - \sum_{p \in \mathcal{P}_c} \frac{1}{p-1} \ln(p))$. By definition of ϕ, $d = \phi(c) = c \prod_{p \in \mathcal{P}_c} (1 - \frac{1}{p})$, so $\ln(c) - \ln(d) = \sum_{p \in \mathcal{P}_c} (\ln(p) - \ln(p-1))$. The result immediately follows. □

In other words, for cyclotomic fields of conductor c, we have $t_2 = 0$ if c is a power of 2, and $t_2 < 0$ if c has an odd prime factor. In particular, for a composite conductor c, each distinct odd prime factor contributes to decreasing the discriminant gap t_2, thereby flattening the predicted module-BKZ profile. However, we remark that the quantity $\frac{p-2}{p-1} \ln(p) - \ln(p-1)$ is minimal for $p = 5$ and is increasing for $p \geq 5$. This is illustrated in Figure 4.7, where we provide the respective values of t_2 for prime conductors $c = 5, 7, 11, 3, 13, 17, 19$, listed in increasing order with respect to t_2. Asymptotically, for $p \to \infty$, the contribution of a prime factor p is $\frac{1-\ln(p)}{2p} + O\left(\frac{1}{p^2}\right)$.

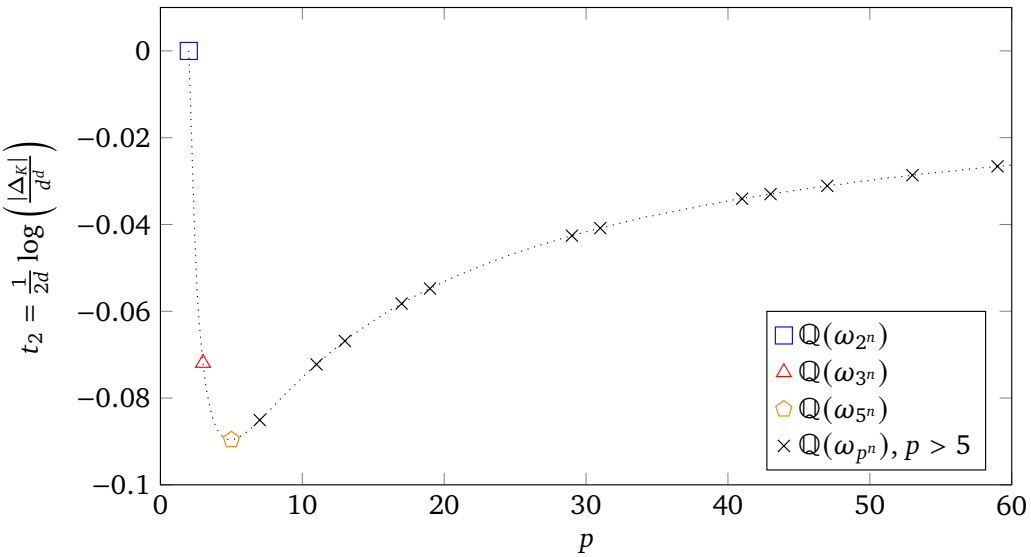


Figure 4.7: Discriminant gap t_2 for cyclotomic fields of conductor $c = p^n$, where p is a prime and $n \in \mathbb{N}$ [.py].

We also numerically checked in sage whether subfields of cyclotomic fields could give good values of t_2 up to conductor 105 [.py]. They turned out to always be equal to or larger than for the original cyclotomic.

4.4.4 Skewness gap

Next, we consider the skewness gap $t_3 = \mathbb{E}\left[\ln\left(\frac{\sqrt{d}\,\mathrm{N}(\mathbf{s})^{1/d}}{\|\mathbf{s}\|}\right)\right]$, where $\mathbf{s}$ is a shortest vector in one of the random projected lattices encountered by module-BKZ. We recall that $\sqrt{d}\,\mathrm{N}(\mathbf{s})^{1/d}/\|\mathbf{s}\|$ measures how *skewed* the values $(|\sigma(\mathbf{s})|^2)_{\sigma\in\mathcal{E}}$ are. Since this quantity is invariant by scaling the value $\|\mathbf{s}\|$, we only need to model its direction. We are therefore naturally tempted to model the direction of $\mathbf{s}$ as being uniform over the $(\beta_K - 1)$-dimensional unit sphere $\mathcal{S}(K_{\mathbb{R}}^{\beta_K})$. Although this omits the fact that $\mathbf{s}$ should be a shortest vector of a module lattice (which certainly constraints its direction [CDPR16]), we can compare this modeled skewness gap, denoted $\widetilde{t_3}$, with the experimentally observed skewness. Our analysis of $\widetilde{t_3}$ is similar to that of [DW21, Lemma 4.4], generalized to arbitrary number fields and module ranks.

Let us first express the independent real Gaussian variables more explicitly using the definition of the trace norm. For $\mathbf{s} = (s_1, \ldots, s_{\beta_K})$, we have

$$\|\mathbf{s}\|^2 = \mathrm{Tr}(\langle \mathbf{s}, \mathbf{s}\rangle_K) = \sum_{\sigma\in\mathcal{E}} \sum_{j=1}^{\beta_K} \sigma(s_j)\sigma(\overline{s_j}).$$

For each $j \in [\beta_K]$, $\sigma_{\mathbb{R}} \in \mathcal{E}_{\mathbb{R}}$, and $\sigma_{\mathbb{C}} \in \mathcal{E}_{\mathbb{C}}^+$, let $A_{\sigma_{\mathbb{R}},j} := \sigma_{\mathbb{R}}(s_j)$, $B_{\sigma_{\mathbb{C}},j} := \sqrt{2}\cdot \mathrm{Re}(\sigma_{\mathbb{C}}(s_j))$, and $C_{\sigma_{\mathbb{C}},j} := \sqrt{2}\cdot \mathrm{Im}(\sigma_{\mathbb{C}}(s_j))$, which gives

$$\|\mathbf{s}\|^2 = \sum_{j=1}^{\beta_K} \left(\sum_{\sigma\in\mathcal{E}_{\mathbb{R}}} A_{\sigma,j}^2 + \sum_{\sigma\in\mathcal{E}_{\mathbb{C}}^+} (B_{\sigma,j}^2 + C_{\sigma,j}^2) \right). \tag{4.4}$$

On the other hand, the algebraic norm of $\mathbf{s}$ satisfies

$$\mathrm{N}(\mathbf{s})^2 = \prod_{\sigma\in\mathcal{E}} \sum_{j=1}^{\beta_K} \sigma(s_j)\sigma(\overline{s_j}) = \left(\prod_{\sigma\in\mathcal{E}_{\mathbb{R}}} \sum_{j=1}^{\beta_K} A_{\sigma,j}^2 \right) \left(\prod_{\sigma\in\mathcal{E}_{\mathbb{C}}^+} \frac{1}{2} \sum_{j=1}^{\beta_K} (B_{\sigma,j}^2 + C_{\sigma,j}^2) \right)^2.$$

It is therefore $A_{\sigma,j}$, $B_{\sigma,j}$, and $C_{\sigma,j}$ in Equation (4.4) that we model as independent Gaussian variables, say of mean 0 and variance 1. The terms $\sum_{j=1}^{\beta_K} A_{\sigma,j}^2$, $\sum_{j=1}^{\beta_K} B_{\sigma,j}^2$, and $\sum_{j=1}^{\beta_K} C_{\sigma,j}^2$ then follow a χ^2 distribution, and so does $\|\mathbf{s}\|^2$. The expected logarithm of a χ^2 distribution relates to the digamma function, denoted ψ, via

$\mathbb{E}[\ln \chi_a^2] = \psi(a/2) + \ln 2$, where a is the number of degrees of freedom. We proceed with the calculation [.py], where we use $d = d_{\mathbb{R}} + 2d_{\mathbb{C}}$ to cancel some $\ln 2$ terms:

$$\begin{aligned}\widetilde{t_3} &:= \mathbb{E}_{\mathbf{s}\sim U(\mathcal{S}(K_{\mathbb{R}}^{\beta_K}))}\left[\ln \frac{\sqrt{d}\,\mathrm{N}(\mathbf{s})^{1/d}}{\|\mathbf{s}\|}\right] \\ &= \frac{\ln d}{2} + \frac{d_{\mathbb{R}}\mathbb{E}[\ln \chi^2_{\beta_K}] + 2d_{\mathbb{C}}(\mathbb{E}[\ln \chi^2_{2\beta_K}] - \ln 2)}{2d} - \frac{\mathbb{E}[\ln \chi^2_{\beta_K d}]}{2} \\ &= \frac{\ln d}{2} + \frac{d_{\mathbb{R}}\psi(\beta_K/2) + 2d_{\mathbb{C}}(\psi(\beta_K) - \ln 2)}{2d} - \frac{\psi(\beta_K d/2)}{2}. \end{aligned} \tag{4.5}$$

When K is a totally imaginary field, such as a cyclotomic field of conductor $c > 2$, we have $d_{\mathbb{R}} = 0$ and $2d_{\mathbb{C}} = d$. In fact, Equation (4.5) equals 0 when K is an imaginary *quadratic* field ($d = 2$). Moreover, for $K = \mathbb{Q}$, our model adequately predicts no skewness as well.

Figure 4.8 suggests that the model and experiments converge for large values of β_K, but $\widetilde{t_3}$ significantly underestimates the skewness term t_3 for smaller β_K. This means that our model is not exactly accurate. In particular, it does not account for the fact that $\mathbf{s}$ must be a shortest vector of a module lattice, and therefore of $\mathbf{s}\mathcal{O}_K$.

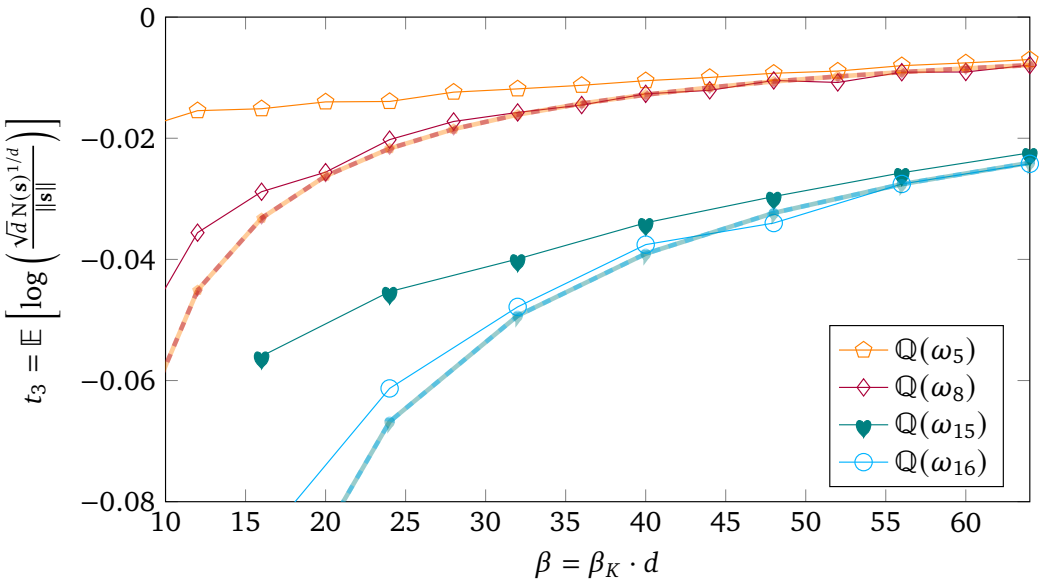


Figure 4.8: Skewness gap (term t_3). Thick translucent lines with small marks correspond to the prediction $\widetilde{t_3}$ of our model in Equation (4.5), thin lines with large marks are the experimental average, taken over 1000 samples [.py]. We remark that as the prediction depends only on $d_{\mathbb{R}}$ and $d_{\mathbb{C}}$, predictions for different rings can overlap.

Let us fix $\|\mathbf{s}\| = 1$ by rescaling. In our model, $\mathrm{N}(\mathbf{s})^{1/d}$, has a small probability of being arbitrarily small, for example if the $A_{i,\sigma}$ are small enough for a fixed σ and all i (think of points on a sphere of dimension $\beta_K d$ close to a hyperplane of dimension β_K). However, such a situation is forbidden by Minkowski's bound: if $\mathrm{N}(\mathbf{s})^{1/d}$ gets arbitrarily small, then $\det(\mathbf{s}\mathcal{O}_K)$ does as well, putting an arbitrary small upper bound on the first minimum of the lattice.

4.4.5 Index gap

It remains to analyze the index gap $t_4 = \frac{1}{d}\mathbb{E}[\ln(\mathrm{N}(\mathfrak{I}))]$. By construction of module-BKZ, the output pseudobasis is unital, so $\mathcal{O}_K \subseteq \mathfrak{I}$ and thus $\ln(\mathrm{N}(\mathfrak{I})) \leq 0$. Let us first clarify that $\ln(\mathrm{N}(\mathfrak{I})) < 0$ does not require $\mathcal{O}_K$ to have non-principal ideals: the shortest vector of a principal ideal may not be a generator. In fact, there are cases where the shortest generator is subexponentially larger than the shortest vector [CDPR16, Lemma 6.2].

We provide a lower bound $\widetilde{t_4}$ on t_4 via a density-based argument using the Dedekind zeta function ζ_K, in a similar way as [ABD16, Sec 2.2], [DPPW22, App. A], and [DW21, Claim 4.2]. We assume here that the random rank-β_K modules admit a K-basis $\mathbf{B}$ rather than a pseudobasis, and let $\mathbf{v} = \mathbf{B}\mathbf{x}$ be a shortest vector. The ideal $\mathfrak{I}$ constructed in the module-BKZ algorithm is then equal to $\gcd(\{x_i\mathcal{O}_K\}_i)^{-1}$.

Following standard analysis, and assuming that the x_i's are random large elements of $\mathcal{O}_K$, the probability that this gcd is a multiple of an ideal $\mathfrak{a} \subseteq \mathcal{O}_K$ is $\mathrm{N}(\mathfrak{a})^{-\beta_K}$. Writing D for the distribution of the $\mathfrak{I}$ under our model, we obtain the following by decomposing $\mathfrak{I}$ over the prime ideals of $\mathcal{O}_K$ [.py]:

$$\begin{aligned}
\widetilde{t_4} &:= \frac{1}{d}\mathbb{E}_{\mathfrak{I}\sim D}[\ln \mathrm{N}(\mathfrak{I})] \\
&= -\frac{1}{d}\sum_{\mathfrak{p}}\sum_{i\in\mathbb{N}} i\cdot(\Pr[\mathfrak{I}^{-1}\subseteq\mathfrak{p}^i] - \Pr[\mathfrak{I}^{-1}\subseteq\mathfrak{p}^{i+1}])\cdot\ln\mathrm{N}(\mathfrak{p}) \\
&= -\frac{1}{d}\sum_{\mathfrak{p}}\sum_{i\in\mathbb{N}} i\cdot(\mathrm{N}(\mathfrak{p})^{-i\beta_K} - \mathrm{N}(\mathfrak{p})^{-(i+1)\beta_K})\cdot\ln\mathrm{N}(\mathfrak{p}) \\
&= -\frac{1}{d}\sum_{\mathfrak{p}}\sum_{i\in\mathbb{N}} \mathrm{N}(\mathfrak{p})^{-i\beta_K}\ln\mathrm{N}(\mathfrak{p}) \\
&= -\frac{1}{d}\sum_{\mathfrak{p}}\frac{\ln\mathrm{N}(\mathfrak{p})}{\mathrm{N}(\mathfrak{p})^{\beta_K}-1} = \frac{1}{d}\frac{\zeta_K'(\beta_K)}{\zeta_K(\beta_K)}
\end{aligned} \tag{4.6}$$

where ζ_K denotes the Dedekind zeta function of K, and ζ_K'/ζ_K is its logarithmic derivative [.py]. Here, we used a telescoping identity $\sum_{i=1}^{\infty} i \cdot (x^i - x^{i+1}) = \sum_{i=1}^{\infty} x^i$.

However, over $\mathbb{Q}$, $\mathbb{Q}(\omega_3)$, and $\mathbb{Q}(\omega_4)$, the algebraic norm is an increasing function of the Euclidean norm, so shortest vectors and generators of rank-1 ideals always coincide: it can never be that $\mathrm{N}(\mathfrak{I}) < 1$ despite the above analysis saying it happens with nonzero probability. The previous analysis fails to capture that $\mathbf{v}$ is short, which lowers the probability of finding divisors in this gcd.

The failure of the modeled index gap $\widetilde{t_4}$ is evident in Figure 4.9: for small rank β_K, the model predicts a significant index gap, while for a cyclotomic field of conductor $c \in \{1, 3, 4, 5, 8, 15\}$ the index gap was always trivial over 1000 samples. Although we did encounter $\mathrm{N}(\mathfrak{I}) < 1$ for conductor 16, this still occurred much less often than predicted. Overall, it seems better to treat this model as a lower bound on t_4.

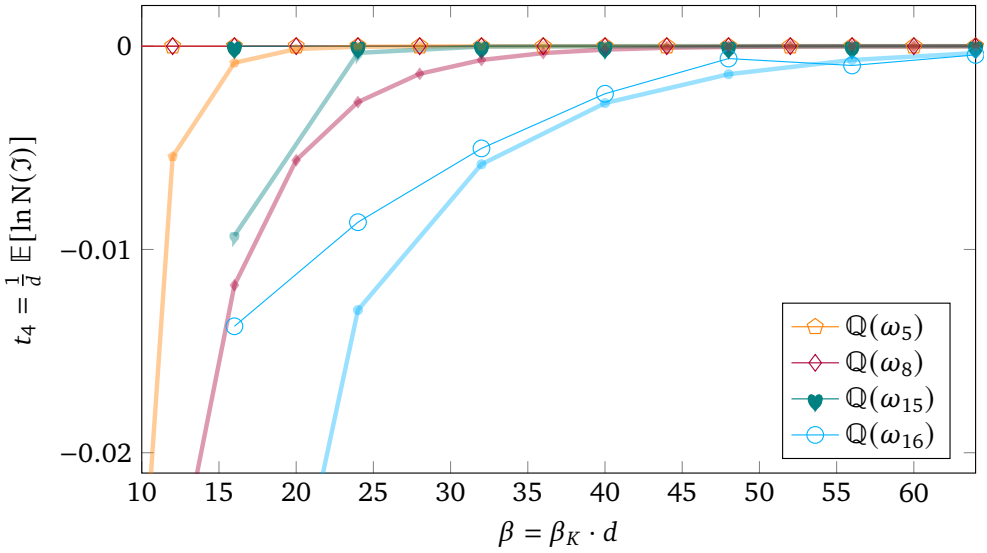


Figure 4.9: Index gap (term t_4). Thick translucent lines with small marks correspond to the prediction $\widetilde{t_4}$ of our model in Equation (4.6), thin lines with large marks are the experimental average, taken over 1000 samples [.py].

Yet, as we discuss in Section 4.5.1, the modeled index gap $\widetilde{t_4}$ converges exponentially fast to 0 as a function of β_K anyway, so this term is actually rather well controlled for large blocksizes.

4.4.6 Conclusion on the module-BKZ slope

We now bring together our analysis of the terms t_1, t_2, t_3, t_4 and conclude with prediction bounds for the $\mathbb{Q}$-slope of module-BKZ, as illustrated in Figure 4.1. We have shown that for a degree-d number field K the module-BKZ slope satisfies

$$\mathrm{slope}_{\mathbb{Q}}(\mathrm{mBKZ}_K^{\beta_K}) = -\frac{2}{\beta_K d - d}(t_1 + t_2 + t_3 + t_4)$$

where:

- $t_1 = \mathrm{lgh}_{\mathbb{Q}}(\beta_K d) + \frac{1}{\beta_K d} \ln \frac{\mu_K}{2}$
- $t_2 = \frac{1}{2d} \ln \frac{|\Delta_K|}{d^d}$
- $\frac{\ln d}{2} + \frac{d_{\mathbb{R}}\psi(\beta_K/2) + 2d_{\mathbb{C}}(\psi(\beta_K) - \ln 2)}{2d} - \frac{\psi(\beta_K d/2)}{2} \leq t_3 \leq 0$
- $\frac{1}{d}\frac{\zeta_K'(\beta_K)}{\zeta_K(\beta_K)} \leq t_4 \leq 0.$

Here, we rely on the module-lattice Gaussian Heuristic for t_1, while the lower bounds for t_3 and t_4 are based on the spherical model and the density-based argument (respectively). In contrast, the formula for t_2 does not rely on any heuristic and depends only on K.

We implemented these formulas for some cyclotomic fields to predict the module-BKZ $\mathbb{Q}$-slope interval, and compared it against the results from our module-BKZ implementation. This prediction interval and its comparison with practice are shown in Figure 4.1 in Section 4.1. Looking at the plots, our prediction interval seems accurate for a rather large range of blocksizes, showing a significant gain for $\mathbb{Q}(\omega_3)$ and $\mathbb{Q}(\omega_{15})$, and a significant loss for $\mathbb{Q}(\omega_8)$, compared to $\mathbb{Q}$. We observe a minor quantitative misfit for $\mathbb{Q}$ and $\mathbb{Q}(\omega_3)$, which might be due to head and tail phenomena that the module-lattice GSA does not account for, but at least qualitatively the gain for $\mathbb{Q}(\omega_3)$ is experimentally confirmed. Such a misfit also appears in unstructured BKZ [YD17; BSW18], and can possibly be overcome by a tail-adapted refinement and simulation (see Open Questions 1, 2 below).

4.5 Asymptotic analysis of the blocksize gain

Consider a random rank-r module lattice over a degree-d number field K. We have heuristically established the following slope predictions for BKZ and module-BKZ, given an SVP oracle for (unstructured) lattices of dimension β:

$$\mathrm{slope}_{\mathbb{Q}}(\mathrm{BKZ}^{\beta}) = -\frac{2}{\beta-1}\,\mathrm{lgh}_{\mathbb{Q}}(\beta),$$

$$\mathrm{slope}_{\mathbb{Q}}(\mathrm{mBKZ}_K^{\beta/d}) = -\frac{2}{\beta-d}(t_1+t_2+t_3+t_4),$$

where the terms $t_i = t_i(\beta/d)$ implicitly depend on K and the K-rank $\beta_K := \beta/d$.

In order to compare the predicted BKZ and module-BKZ slopes, we consider, for a given blocksize β, the "equivalent" blocksize $\beta_{\mathrm{eq}} = \beta_{\mathrm{eq}}(\beta)$ such that

$$\mathrm{slope}_{\mathbb{Q}}(\mathrm{BKZ}^{\beta}) = \mathrm{slope}_{\mathbb{Q}}(\mathrm{mBKZ}_K^{\beta_{\mathrm{eq}}/d}).$$

The blocksize gain or loss of module-BKZ is then defined as the difference $\beta_{\mathrm{eq}} - \beta$ (corresponding to the difference in dimensions for the used SVP oracle). We can estimate $\beta_{\mathrm{eq}} - \beta$ using our slope predictions [.py], as illustrated in Figure 4.2 in Section 4.1. In the remainder of this section, we will instead analyze how $\beta_{\mathrm{eq}} - \beta$ behaves as $\beta \to \infty$.

4.5.1 Asymptotic behavior of each term

In order to identify the leading asymptotic terms in the module-BKZ slope, we first detail the asymptotic contribution of each term t_i. We will denote the module-BKZ blocksize by β.

Module-lattice Gaussian Heuristic t_1. To analyze the first term t_1, which is given by $t_1 = \mathrm{lgh}_{\mathbb{Q}}(\beta) + \frac{1}{\beta}\ln\frac{\mu_K}{2}$ under Heuristic 4.4.5, we recall that $\mathrm{lgh}_{\mathbb{Q}}(\beta) = \frac{1}{\beta}\big(\ln(2) - \gamma - \ln(\mathrm{vol}(\mathcal{B}_\beta))\big)$ where γ denotes the Euler-Mascheroni constant, and $\mathcal{B}_\beta$ the β-dimensional Euclidean unit ball. Denoting the Gamma function by Γ, we have $\ln(\mathrm{vol}(\mathcal{B}_\beta)) = \frac{\beta}{2}\ln(\pi) - \ln\Gamma(1+\frac{\beta}{2})$, which implies $\mathrm{lgh}_{\mathbb{Q}}(\beta) = \frac{1}{\beta}\big(\ln(2) - \gamma + \ln\Gamma(1+\frac{\beta}{2})\big) - \frac{1}{2}\ln(\pi)$. Since $\ln\Gamma(z) = z\ln(z) - z - \frac{\ln z}{2} + \frac{\ln(2\pi)}{2} + o(1)$ as z grows, we have $\mathrm{lgh}_{\mathbb{Q}}(\beta) = \frac{\ln(\beta)}{2} - \frac{\ln(2\pi e)}{2} + \frac{\ln(\beta)}{2\beta} + \frac{\ln(4\pi)-2\gamma}{2\beta} + o\left(\frac{1}{\beta}\right)$, and obtain:

$$t_1 = \frac{\ln(\beta)}{2} - \frac{\ln(2\pi e)}{2} + \frac{\ln\beta}{2\beta} + \frac{\ln(\pi\mu_K^2) - 2\gamma}{2\beta} + o\left(\frac{1}{\beta}\right).$$

Discriminant gap t_2. The second term, $t_2 = \frac{1}{2d} \ln \frac{|\Delta_K|}{d^d}$, depends only on K, and is therefore constant in β.

Skewness gap t_3. Following our analysis in Section 4.4.4, we assume that t_3 converges to $\widetilde{t_3}$ as $\beta \to \infty$, where $\widetilde{t_3} = \frac{\ln d}{2} + \frac{d_{\mathbb{R}}\psi(\beta/2d)+2d_{\mathbb{C}}(\psi(\beta/d)-\ln 2)}{2d} - \frac{\psi(\beta/2)}{2}$ for the digamma function ψ. Since the digamma function behaves asymptotically as $\psi(z) = \ln(z) - \frac{1}{2z} + o\left(\frac{1}{z}\right)$, we obtain:

$$t_3 = \frac{1 - d_{\mathbb{R}} - d_{\mathbb{C}}}{2\beta} + o\left(\frac{1}{\beta}\right) = o\left(\frac{\ln \beta}{\beta}\right).$$

Index gap t_4. Following our analysis in Section 4.4.5, we assume that t_4 converges to $\widetilde{t_4}$ as $\beta \to \infty$, where $\widetilde{t_4} = -\frac{1}{d} \sum_{\mathfrak{p}} \frac{\ln \mathrm{N}(\mathfrak{p})}{\mathrm{N}(\mathfrak{p})^{\beta_K}-1}$ with $\mathfrak{p}$ ranging over the prime ideals of $\mathcal{O}_K$ and $\beta_K \coloneqq \beta/d$. By Lemmas 4.5.1 and 4.5.2 below, we obtain $t_4 \geq -\sum_p \frac{\ln(p)}{p^{\beta_K}-1}$, where p ranges over the prime numbers $p \in \mathbb{N}$ such that $\mathfrak{p} \mid p\mathcal{O}_K$ for some prime ideal $\mathfrak{p} \subseteq \mathcal{O}_K$. Thus, writing $\mathcal{P}$ for the set of prime numbers in $\mathbb{N}$, we obtain $t_4 \geq -\sum_{p\in\mathcal{P}} \frac{\ln(p)}{p^{\beta_K}-1} \geq -\sum_{p\in\mathcal{P}} \frac{2\ln(p)}{p^{\beta_K}} \geq -\sum_{p\in\mathcal{P}} \frac{1}{p^{\beta_K-1}} \geq -\frac{1}{2^{\beta_K-1}} - \int_2^\infty \frac{du}{u^{\beta_K-1}}$. In particular, for $\beta_K \geq 3$, we have $-\frac{1}{2^{\beta_K-1}} - \frac{1}{(\beta_K-2)2^{\beta_K-2}} \leq t_4 \leq 0$, so we conclude:

$$t_4 = -\frac{1}{2^{\beta_K-1}} + o\left(\frac{1}{2^{\beta_K}}\right) = o\left(\frac{1}{\beta}\right).$$

The following lemmas are presented in different form in [Neu92; Sam13].

Lemma 4.5.1. *Let K be a number field of degree d. For every prime number $p \in \mathbb{N}$, $p\mathcal{O}_K$ can be decomposed into a product of at most d prime ideals $\mathfrak{p}$ such that, for each $\mathfrak{p}$, there exists $k \in [d]$ for which $\mathrm{N}(\mathfrak{p}) = p^k$.*

Proof. Let $p\mathcal{O}_K = \prod_{i\in I} \mathfrak{p}_i^{\nu_i}$ be the unique decomposition of $p\mathcal{O}_K$ into a product of distinct prime ideals. It follows that $\mathrm{N}(p\mathcal{O}_K) = \prod_{i\in I} \mathrm{N}(\mathfrak{p}_i)^{\nu_i}$. Since $\mathrm{N}(p\mathcal{O}_K) = p^d$, we obtain $\prod_{i\in I} \mathrm{N}(\mathfrak{p}_i)^{\nu_i} = p^d$. Hence, for each $i \in I$, there exists $k_i \in [d]$ such that $\mathrm{N}(\mathfrak{p}_i) = p^{k_i}$, and thus $\sum_{i\in I} k_i\nu_i = d$ implies $|I| \leq d$. □

Lemma 4.5.2. *Let K be a number field of degree d. For every prime ideal $\mathfrak{p}$ of $\mathcal{O}_K$, there exists a unique prime number $p \in \mathbb{N}$ such that $\mathfrak{p} \mid p\mathcal{O}_K$, and moreover $\mathrm{N}(\mathfrak{p}) = p^k$ for some $k \in [d]$.*

Proof. Let $\mathfrak{p}$ be a prime ideal of $\mathcal{O}_K$. Then $\mathfrak{p} \cap \mathbb{Z}$ is a prime ideal (p) in $\mathbb{Z}$ (see the proof of [Neu92, Theorem 3.1]). Hence, $p\mathcal{O}_K \subseteq \mathfrak{p}$ and $\mathfrak{p} \mid p\mathcal{O}_K$, which implies $\mathrm{N}(\mathfrak{p}) = p^k$ for some $k \in [d]$. Moreover, there is no other prime $q \in \mathbb{N}$, $q \neq p$, such that $\mathfrak{p} \mid q\mathcal{O}_K$, for otherwise there exists $k' \in [d]$ such that $\mathrm{N}(\mathfrak{p}) = q^{k'}$, which would contradict $\mathrm{N}(\mathfrak{p}) = p^k$. □

4.5.2 Asymptotic gain compared to unstructured BKZ

Finally, we use the previous analysis to show the asymptotic relation between β and β_{eq}, when β_{eq} is chosen such that $\mathrm{slope}_{\mathbb{Q}}(\mathrm{BKZ}^{\beta}) = \mathrm{slope}_{\mathbb{Q}}(\mathrm{mBKZ}_K^{\beta_{\mathrm{eq}}/d})$.

Heuristic Claim 4.5.3. *Let K be a number field of degree d, and $\beta_{\mathrm{eq}} = \beta_{\mathrm{eq}}(\beta)$ be such that $\mathrm{slope}_{\mathbb{Q}}(\mathrm{mBKZ}_K^{\beta_{\mathrm{eq}}/d}) = \mathrm{slope}_{\mathbb{Q}}(\mathrm{BKZ}^{\beta})$. Then, as $\beta \to \infty$, our heuristic analysis predicts*

$$\beta_{\mathrm{eq}} = \beta + \ln\left(\frac{|\Delta_K|}{d^d}\right)\frac{\beta}{d\ln\beta}\left(1 + \frac{\ln(2\pi e^2)}{\ln\beta} + o\left(\frac{1}{\ln\beta}\right)\right) + d - 1 + o(1).$$

In particular, we asymptotically predict

$$\beta_{\mathrm{eq}} - \beta = \begin{cases} \ln\left(\frac{|\Delta_K|}{d^d}\right)\frac{\beta}{d\ln\beta} + o\left(\frac{\beta}{\ln\beta}\right) & \text{for all } K \\ d - 1 + o(1) & \text{if } |\Delta_K| = d^d. \end{cases}$$

This prediction implies that, for power-of-two cyclotomic fields (where $|\Delta_K| = d^d$), module-BKZ requires a blocksize that is larger by at most an additive constant in order to reach the same slope as unstructured BKZ. In contrast, for all other cyclotomic fields (where $|\Delta_K| < d^d$), module-BKZ provides a sublinear additive gain of $\Theta(\beta/\ln\beta)$ in the required blocksize to reach the same slope as BKZ with blocksize β; this corresponds to a speedup by a factor $\exp(\Theta(\beta/\ln\beta))$ compared to BKZ.

Justification of Heuristic Claim 4.5.3. By definition of β_{eq}, we have

$$\frac{\mathrm{lgh}_{\mathbb{Q}}(\beta)}{\beta-1} = \frac{t_1+t_2+t_3+t_4}{\beta_{\mathrm{eq}}-d} \tag{4.7}$$

where the t_i are functions of β_{eq}. Our asymptotic analysis from Section 4.5.1 implies that $t_1 = \mathrm{lgh}_{\mathbb{Q}}(\beta_{\mathrm{eq}}) + o\left(\frac{\ln\beta_{\mathrm{eq}}}{\beta_{\mathrm{eq}}}\right)$, that $t_2 = \frac{1}{2d}\ln\frac{|\Delta_K|}{d^d}$ is constant, and that t_3 and t_4 are both $o\left(\frac{\ln\beta_{\mathrm{eq}}}{\beta_{\mathrm{eq}}}\right)$. Section 4.5.1 also yields $2\,\mathrm{lgh}_{\mathbb{Q}}(\beta) = \ln(\beta) - \ln(2\pi e) + \frac{\ln\beta}{\beta} + o\left(\frac{\ln\beta}{\beta}\right)$. Considering the dominant terms of Equation (4.7), we obtain $\frac{\ln\beta_{\mathrm{eq}}}{\beta_{\mathrm{eq}}} \sim \frac{\ln\beta}{\beta}$, and in particular $o\left(\frac{\ln\beta_{\mathrm{eq}}}{\beta_{\mathrm{eq}}}\right) = o\left(\frac{\ln\beta}{\beta}\right)$. Therefore,

$$\beta_{\mathrm{eq}} = d + (\beta-1)\cdot A(\beta) \tag{4.8}$$

where

$$A(\beta) = \frac{2\,\mathrm{lgh}_{\mathbb{Q}}(\beta_{\mathrm{eq}}) + 2t_2 + o\left(\frac{\ln\beta}{\beta}\right)}{2\,\mathrm{lgh}_{\mathbb{Q}}(\beta)} = 1 + \frac{\ln\left(\frac{\beta_{\mathrm{eq}}}{\beta}\right) + 2t_2 + o\left(\frac{\ln\beta}{\beta}\right)}{\ln(\beta) - \ln(2\pi e) + \frac{\ln\beta}{\beta} + o\left(\frac{\ln\beta}{\beta}\right)}.$$

As $-\frac{\ln\beta_{\mathrm{eq}}}{\beta_{\mathrm{eq}}} = -\frac{\ln\beta}{\beta} + o\left(\frac{\ln\beta}{\beta}\right)$, applying the W_{-1} branch of the Lambert W function (for $\beta_{\mathrm{eq}} \geq 3$) gives $-\ln\beta_{\mathrm{eq}} = W_{-1}\left(-\frac{\ln\beta}{\beta} + o\left(\frac{\ln\beta}{\beta}\right)\right) = -\ln\beta + o(\ln\beta)$. It follows that $\ln\frac{\beta_{\mathrm{eq}}}{\beta} = o(\ln\beta)$, so

$$A(\beta) = 1 + \frac{\ln\left(\frac{\beta_{\mathrm{eq}}}{\beta}\right) + 2t_2}{\delta} + o\left(\frac{1}{\beta}\right) = 1 + o(1)$$

for $\delta = \delta(\beta) := \ln(\beta) - \ln(2\pi e) + \frac{\ln\beta}{\beta}$. Replacing this into Equation (4.8) yields

$$\frac{\beta_{\mathrm{eq}}}{\beta} = 1 + \frac{\ln\left(\frac{\beta_{\mathrm{eq}}}{\beta}\right) + 2t_2}{\delta} + \frac{d-1}{\beta} + o\left(\frac{1}{\beta}\right)$$

which, for $\varepsilon = \varepsilon(\beta) := 2t_2 + (d-1)\,\frac{\ln\beta}{\beta} + o\left(\frac{\ln\beta}{\beta}\right)$, implies

$$-\delta\frac{\beta_{\mathrm{eq}}}{\beta}e^{-\delta\frac{\beta_{\mathrm{eq}}}{\beta}} = -\delta\, e^{-\delta-\varepsilon}. \tag{4.9}$$

Let $E = E(\beta)$ be such that $-\delta\frac{\beta_{\mathrm{eq}}}{\beta} = -\delta - E$. Since $W_{-1}\left(-\delta\frac{\beta_{\mathrm{eq}}}{\beta}e^{-\delta\frac{\beta_{\mathrm{eq}}}{\beta}}\right) = -\delta\frac{\beta_{\mathrm{eq}}}{\beta} =$

$-\delta - E$, Equation (4.9) yields $-\delta - E = W_{-1}\left(-\delta e^{-\delta-\varepsilon}\right)$, and with a first order expansion of W_{-1} in 0^- we obtain $-\delta - E = -\delta - \varepsilon + \ln\delta + o(\ln\beta)$. It follows that $\frac{E}{\delta} = o(1)$. Taking logarithms of Equation (4.9), and considering β large enough for $\delta\frac{\beta_{\text{eq}}}{\beta}$ and δ to be positive, the definition of E implies $-\delta - E + \ln(\delta + E) = -\delta - \varepsilon + \ln\delta$, so expansion (using $\frac{E}{\delta} = o(1)$ and the definition of δ) yields

$$\varepsilon = E - \ln\left(1 + \frac{E}{\delta}\right) = E - \frac{E}{\ln\beta} + o\left(\frac{E}{\ln\beta}\right).$$

Therefore,

$$\begin{aligned} E &= \frac{\varepsilon}{1 - \frac{1}{\ln\beta} + o\left(\frac{1}{\ln\beta}\right)} = \varepsilon\left(1 + \frac{1}{\ln\beta} + o\left(\frac{1}{\ln\beta}\right)\right) \\ &= 2t_2\left(1 + \frac{1}{\ln\beta} + o\left(\frac{1}{\ln\beta}\right)\right) + (d-1)\,\frac{\ln\beta}{\beta} + o\left(\frac{\ln\beta}{\beta}\right). \end{aligned}$$

Finally, by substituting this into the original definition of E, we obtain

$$\frac{\beta_{\text{eq}}}{\beta} = 1 + \frac{2t_2\left(1 + \frac{1}{\ln\beta} + o\left(\frac{1}{\ln\beta}\right)\right) + (d-1)\,\frac{\ln\beta}{\beta} + o\left(\frac{\ln\beta}{\beta}\right)}{\delta}$$

and expanding $\frac{1}{1 - \frac{\ln(2\pi e) + o(1)}{\ln\beta}} = 1 + \frac{\ln(2\pi e)}{\ln\beta} + o\left(\frac{1}{\ln\beta}\right)$ gives the desired formula

$$\beta_{\text{eq}} = \beta + 2t_2\,\frac{\beta}{\ln\beta}\left(1 + \frac{\ln(2\pi e^2)}{\ln\beta} + o\left(\frac{1}{\ln\beta}\right)\right) + d - 1 + o(1).$$

□

4.6 Discussion

In this chapter, we provided a concrete and asymptotic analysis of module-BKZ's practical performance using experimentally verified heuristics. Our slope prediction allows us to conclude that module-BKZ is *not* universally better than its unstructured counterpart: it depends on the underlying number field K and especially its discriminant Δ_K. Specifically, writing d for the degree of K, we predict that module-BKZ needs an SVP oracle of dimension

$$\beta + \ln\left(\frac{|\Delta_K|}{d^d}\right)\frac{\beta}{d\ln\beta}(1+o(1)) + d - 1 + o(1)$$

to reach the same slope as BKZ with blocksize β, and our analysis suggests that module-BKZ yields an improved attack when $|\Delta_K| < d^d$. In addition, our experiments and concrete predictions indicate that this asymptotic summary underestimates the actual performance of module-BKZ.

In the remainder of this section, we outline the potential implications for module-lattice-based cryptography and list several open questions for further assessment.

4.6.1 Cryptanalytic impact

First of all, it is important to emphasize that our observations are not bound to the specific number field K underlying a cryptosystem: module-BKZ can be applied with respect to an arbitrary *subfield* of K. In particular, for a scheme over a cyclotomic field $\mathbb{Q}(\omega_s)$ of conductor s, an attacker can work over any $\mathbb{Q}(\omega_c)$ for c dividing s. For example, for Kyber, the underlying cyclotomic field has conductor $s = 512$ and degree 256, so an attacker has the freedom to work over cyclotomic fields of degree $d = 2^k$ for $k \leq 8$. A larger d should allow them to obtain speedups in the SVP oracle, but also restricts them to use a blocksize multiple of this larger value of d.

In the case of power-of-two cyclotomic fields, which are relevant to the new NIST standards ML-KEM, ML-DSA, and FN-DSA [NIS22b], the main asymptotic term $\ln(|\Delta_K|/d^d)\beta/(d\ln\beta)$ disappears as the discriminant of such a field K satisfies $|\Delta_K| = d^d$. The remaining term is $d - 1 + o(1)$, and Figure 4.2 shows a slow[7] convergence from above. This explains the disappointing performance of

[7]For example, for $\mathbb{Q}(\omega_{16})$, we have $d - 1 = 7$, but $\beta_{\text{eq}} - \beta \in [11.2, 12.6]$ at $\beta = 400$.

Karenin and Kirshanova's module-LLL implementation for overstretched NTRU parameters [KK24]. More importantly, this confirms and quantifies the discussion of Kyber's Q8 [ABDK+21, Sec 5.3, Q8], which already suggested module-BKZ might need an increased blocksize in order to match the output quality of unstructured BKZ. Nevertheless, quite a bit of work remains to settle the question of whether module-BKZ can slightly outperform BKZ in this context, see the following Open Questions 1, 2, and 3.

On the contrary, for other cyclotomic fields, especially those whose conductor has one or more *small* odd prime factors, we predict a rather substantial gain in the slope, as was illustrated in Figure 4.2. In fact, motivated by Lemma 4.4.8, we predict that each additional odd prime factor leads to a further slope gain: for instance, compare $\mathbb{Q}(\omega_{15})$ to $\mathbb{Q}(\omega_3)$ and $\mathbb{Q}(\omega_5)$ in Figure 4.2. The slope gain induced by odd prime factors is relevant as popular alternatives to power-of-two conductors are conductors of the form $2^i \cdot 3^j$ [BDF18; EFGR+22; EWY23b]. In that case, module-BKZ over the third cyclotomic field is predicted to gain 20 more dimensions on the blocksize at NIST security level 1 ($\beta \approx 380$).

There are more schemes using conductors of the form $2^i \cdot 3^j$ [PFHK+17; LS19; HWKK+25; Kpq25]. These include an intermediate parameter set of Falcon when it was submitted to the first round of the NIST PQC standardization process, and one of the two selected Korean post-quantum PKE/KEM standards NTRU+. The Homomorphic Encryption library `HElib` also supports general conductors, and a set of parameters with a multiple-of-5 conductor was highlighted in [HS14]. However, these schemes use the Euclidean norm defined by the coefficient embedding rather than the canonical embedding, and the impact of our observations would require special consideration (see Open Question 5).

Lastly, our analysis might narrow down avenues to construct more efficient schemes based on module-LIP [DPPW22]. The LIP framework [DW22] was designed to harness the decoding capabilities of dense lattices in cryptography, but competitive proposals likely require a module structure. One would naturally turn to algebraic constructions such as the construction [Bay00] of the Leech lattice as an ideal in a cyclotomic field of conductor $35, 39, 52, 56$, or 84. More problematically, one could be tempted to use Martinet's construction [Mar78] of asymptotically dense lattices based on towers of number fields of bounded discriminant (corresponding to a large negative value of the discriminant gap t_2).

4.6.2 Open questions

Although the prediction and experimental analysis of its expected slope highlight the potential advantages of module-BKZ, this study alone does not suffice to precisely quantify the cost of attacks based on module-BKZ. We list several future directions, including continued investigation of Q8 [ABDK+21, Sec 5.3, Q8]. While our implementation may help answering some of those questions, a proper API for module-lattice reduction would be beneficial.

1. **HKZ profile, tails, and dimensions for free.** Just as the module structure affects the BKZ slope, we expect it to affect the profile of module-HKZ reduction as well. Predicting HKZ shapes in a similar manner would allow replacing the Geometric Series Assumption by its tail-adapted refinement [AD21]. Perhaps more critically, this change of shape should also affect the number of dimensions for free in the SVP subroutine [Duc18], thereby slowing down or accelerating the SVP oracle, in addition to altering the slope.

2. **Profile simulation.** This study is limited to the slope of BKZ after convergence, i.e., after many tours. In practice, cryptanalysts are more aggressive and run only a single or a few tours, progressively increasing the blocksize [AWHT16]. The fine-tuning of attacks and security estimates then resorts to BKZ simulators [CN11; BSW18; XWWG+24], which should be adapted to module-BKZ. In particular, one may question how fast module-BKZ converges compared to unstructured BKZ.

3. **Advanced lattice sieving with cyclotomic symmetries.** Although our results suggest that module-BKZ performs worse than BKZ for power-of-two cyclotomic fields, the part that we model as SVP oracle calls relies in practice on subroutines that may benefit from a cyclotomic structure. Indeed, speedups and memory savings have been demonstrated for sieving over cyclotomic ideal lattices [BNP17]. However, these results were obtained for a rather naive sieving algorithm [MV10], and it is far from clear whether the same methods would combine well in practice with improved sieving techniques based on locality-sensitive hashing [Laa15; BDGL16] and other practical tricks used by the fastest known sieving implementations, such as progressive sieving and the dimensions-for-free technique [LM18; Duc18; ADHK+19; DSW21].

4. **Solving cryptographic module-lattice problems.** Another important open question is whether the observed slope gain translates into faster algorithms for solving module-SIS, module-LWE, module-LIP, and NTRU. A preliminary proof of concept (see [DEP25, Appendix C]) answers this question positively, but precise predictions would require profile simulation (Open Question 2) and a probabilistic analysis of secret recovery [DDGR20; PV21].

5. **Coefficient embedding.** Our predicted and observed slope gains are obtained using the canonical embedding to define the geometry of cyclotomic fields. While it is algebraically more natural to use the canonical embedding, some schemes [HS14; PFHK+17; LS19; HWKK+25] use the coefficient embedding instead. We note that the distortion to go from one embedding to the other depends only on the number field (in fact, no distortion occurs for power-of-two cyclotomic fields), and is constant with growing module rank r. On the contrary, our slope gain leads to a gain in the first basis vector's length that grows exponentially with r for a fixed blocksize. Hence, if r is large enough, then it will eventually be beneficial to apply module-BKZ using the canonical embedding even when the targeted scheme uses a different embedding. Nevertheless, some study is required to determine the exact break-even point and make concrete predictions for this setup.

6. **Shortest-vector versus densest-ideal oracle.** From a theoretic perspective [MS20; KK24], it would be more natural for module-BKZ to use an oracle for finding a *densest ideal* rather than a shortest vector. This raises two questions: to what extent the slope would improve, and how such an oracle could be realized reasonably efficiently compared to the best known SVP oracles [BDGL16; Duc18]. The former question boils down to establishing a Gaussian Heuristic for the algebraic norm.

In fact, some of these questions (especially Open Questions 2 and 4) have recently been addressed by [PT25]. By providing a simulator for module-BKZ, they estimate the cost of a variant of the primal attack on module-LWE that uses module-BKZ instead of BKZ. In particular, their results provide evidence that the use of module-BKZ can lead to faster secret recovery *even* for power-of-two cyclotomic fields, despite the slight blocksize loss that we have shown in this chapter.

Part II: Quantum cryptanalysis

Chapter 5

Quantum algorithmic tools for cryptanalysis

To gain confidence in the security of post-quantum cryptography, it is important to study whether the underlying computational problems can be solved faster on a quantum computer. Such a quantum speedup for a given problem can be achieved by using an approach that is fully quantum, or by using quantum subroutines inside a classical algorithm. This chapter provides a concise overview of the main quantum algorithmic tools used in the next two chapters: Grover's search algorithm, amplitude amplification, and quantum-walk algorithms. Moreover, we briefly explain how carefully nesting such algorithms can lead to further speedups.

Contents of this chapter

5.1 Grover's search algorithm

Grover's search algorithm is defined for the following task: given query access to a Boolean function $f\colon [N] \to \{0,1\}$, find a preimage of 1, or correctly determine that none exists. Query access to f is modelled by a unitary map of the form

$$\mathcal{O}_f\colon |i,b\rangle \mapsto |i, b \oplus f(i)\rangle$$

for all $i \in [N]$ and $b \in \{0,1\}$. Grover's algorithm makes use of the closely related unitary map

$$\mathcal{O}_{f,\pm}\colon |i\rangle \mapsto (-1)^{f(i)} |i\rangle \tag{5.1}$$

for all $i \in [N]$. We call $\mathcal{O}_f$ a *standard oracle* for f, and $\mathcal{O}_{f,\pm}$ a *phase oracle* for f. The latter can be implemented using one query to $\mathcal{O}_f$, a few applications of the elementary gates X and H defined in Section 2.2.1, and one auxiliary qubit:

$$\begin{aligned} |i\rangle|0\rangle &\xmapsto{I\otimes HX} |i\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &\xmapsto{\mathcal{O}_f} (-1)^{f(i)} |i\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &\xmapsto{I\otimes XH} (-1)^{f(i)} |i\rangle|0\rangle\,. \end{aligned}$$

Theorem 5.1.1 (Grover's algorithm [Gro96]). *Let $N \in \mathbb{N}$. There is a quantum algorithm that, given a phase oracle $\mathcal{O}_{f,\pm}$ for $f\colon [N] \to \{0,1\}$, with probability at least $2/3$ returns i such that $f(i) = 1$, or correctly outputs that none exists, using $O(\sqrt{N})$ applications of $\mathcal{O}_{f,\pm}$ and $\tilde{O}(\sqrt{N})$ elementary gates. It uses $\lceil \log_2(N) \rceil$ qubits, plus any auxiliary qubits required to implement $\mathcal{O}_{f,\pm}$.*

Grover [Gro96] originally analyzed the worst-case setting with a unique solution. This was later generalized to the case of multiple solutions in [BBHT98]. When the number t of solutions is known in advance, the algorithm can be further refined to achieve success probability 1 [BHMT02, Theorem 4]. These results can be summarized as follows.

Theorem 5.1.2 (Grover's algorithm, general case [Gro96; BBHT98; BHMT02]). *Let $N \in \mathbb{N}$. In each case below, there exists a quantum algorithm that, given a phase oracle $\mathcal{O}_{f,\pm}$ for $f\colon [N] \to \{0,1\}$, has the stated properties and uses $\lceil \log_2(N) \rceil$ qubits, plus any auxiliary qubits required to implement $\mathcal{O}_{f,\pm}$. Let $t := |\{i \in [N] : f(i) = 1\}|$.*

- *If t is known and nonzero, then with certainty the algorithm returns $i \in [N]$ such that $f(i) = 1$, using $O(\sqrt{N/t})$ applications of $\mathcal{O}_{f,\pm}$ and $\tilde{O}(\sqrt{N/t})$ elementary gates.*
- *If t is unknown, then the algorithm returns $i \in [N]$ such that $f(i) = 1$, if it exists, using an expected number of $O(\sqrt{N/t})$ applications of $\mathcal{O}_{f,\pm}$ and $\tilde{O}(\sqrt{N/t})$ elementary gates.*
- *If t is unknown, but it is promised that either $t \geq t_0 \geq 1$ or $t = 0$, then with probability at least $2/3$ the algorithm returns $i \in [N]$ such that $f(i) = 1$, or correctly outputs that none exists, using $O(\sqrt{N/t_0})$ applications of $\mathcal{O}_{f,\pm}$ and $\tilde{O}(\sqrt{N/t_0})$ elementary gates.*[a]

[a] Note that this statement follows from the previous bullet point and Markov's inequality.

By design, Grover's search algorithm returns a uniformly random element from the set $\{i \in [N] : f(i) = 1\}$ when its size t is nonzero. Therefore, a coupon-collector argument implies that $\tilde{O}(t)$ repetitions suffice to find all solutions with high probability, also when t is not known in advance (this can be shown using a similar argument as in Lemma 6.2.1 in the next chapter).

Remark 5.1.3 (Boosting the success probability). The error probability $\leq 1/3$ of a classical or quantum algorithm $\mathcal{A}$, such as Grover's search algorithm, can be reduced to arbitrarily small error probability $\delta > 0$ using standard methods: run $\mathcal{A}$ independently for $O(\log(1/\delta))$ times and take the majority of the answers. (Note that you cannot verify correctness of Grover when $t = 0$.) In the case of Grover's search algorithm, it suffices to run $\mathcal{A}$ for $O(\sqrt{\log(1/\delta)})$ times [BCWZ99, Theorem 3].

5.2 Amplitude amplification

Grover's algorithm can be viewed as a special case of amplitude amplification. The task of amplitude amplification is to amplify the success probability ε of a given classical or quantum algorithm $\mathcal{A}$, assuming that its correctness can be recognized. (The notation ε is used for easier comparison with Section 5.3.) In the classical setting, boosting the success probability to, say, $2/3$ requires $\Theta(1/\varepsilon)$ applications of $\mathcal{A}$.[1] When $\mathcal{A}$ can be represented as a unitary (that is, a quantum algorithm without measurements), there is a quantum algorithm [BHMT02] that boosts its success probability to $2/3$ using $O(1/\sqrt{\varepsilon})$ applications of $\mathcal{A}$ and its inverse.

If $\mathcal{A}$ can be implemented using an n-qubit unitary, then it has at most $N = 2^n$ possible outputs. We can therefore model correctness by query access to a Boolean function $f\colon [N] \to \{0,1\}$, where $f(i) = 1$ if and only if i is a correct output. The result from [BHMT02] can then be formulated as follows.

Theorem 5.2.1 (Amplitude amplification [BHMT02])**.** *Let $n, N \in \mathbb{N}$ be such that $n \geq \lceil \log_2 N \rceil$. In each case below, there exists a quantum algorithm that, given an n-qubit unitary $\mathcal{A}$ and a phase oracle $\mathcal{O}_{f,\pm}$ for $f\colon [N] \to \{0,1\}$, has the stated properties and uses n qubits, plus any auxiliary qubits required to implement $\mathcal{O}_{f,\pm}$. Let $\varepsilon > 0$ be the probability that $\mathcal{A}$ returns $i \in [N]$ such that $f(i) = 1$.*

- *If ε is known, then with certainty the algorithm returns $i \in [N]$ such that $f(i) = 1$, using $O(1/\sqrt{\varepsilon})$ applications of $\mathcal{A}$, $\mathcal{A}^{-1}$, and $\mathcal{O}_{f,\pm}$, and using $\tilde{O}(1/\sqrt{\varepsilon})$ elementary gates.*
- *If ε is promised to satisfy $\varepsilon \geq \varepsilon_0$, then with probability at least $2/3$ the algorithm returns $i \in [N]$ such that $f(i) = 1$, using $O(1/\sqrt{\varepsilon_0})$ applications of $\mathcal{A}$, $\mathcal{A}^{-1}$, and $\mathcal{O}_{f,\pm}$, and using $\tilde{O}(1/\sqrt{\varepsilon_0})$ elementary gates.*

Note that this recovers the complexity of Grover's search algorithm (Theorem 5.1.2) when there are $t \geq 1$ solutions: namely, let $\mathcal{A}$ be a unitary that creates a uniform superposition over $[N]$ and note that $\varepsilon = t/N$.

[1] Indeed, this uses the same argument as in Remark 5.1.3.

Reformulation. It can sometimes be helpful to rephrase the goal of amplitude amplification as aiming to project a state $|\psi\rangle$, which is relatively easy to prepare, onto a "marked" subspace of the underlying Hilbert space. The reformulated goal is then to obtain the state

$$|\Psi\rangle := \frac{\Pi|\psi\rangle}{\|\Pi|\psi\rangle\|} \tag{5.2}$$

where Π is the projector onto this marked subspace (and assuming $\Pi|\psi\rangle \neq 0$). This marked subspace may encode the solutions to a search problem so that they can be recovered by measuring the final state $|\Psi\rangle$, as in the formulation of Theorem 5.2.1. However, the final state can also be used as input to another quantum algorithm (without measuring it).

For instance, the Grover setting would correspond to letting $|\psi\rangle$ be a uniform superposition over $[N]$, with Π projecting onto the solutions. More generally, the previous formulation of amplitude amplification, which considers boosting the success probability ε of a (say) quantum algorithm $\mathcal{A}$, is recovered by taking $|\psi\rangle = \mathcal{A}|0\rangle$ and letting Π project onto the correct output. Note that we can rewrite $|\psi\rangle$ as

$$|\psi\rangle = \sqrt{\varepsilon}\,|\Psi\rangle + \sqrt{1-\varepsilon}\,|\Psi^{\perp}\rangle$$

where $|\Psi\rangle$ is the normalization of $\Pi|\psi\rangle$ and $|\Psi^{\perp}\rangle$ is some state orthogonal to $|\Psi\rangle$.

We will use this reformulation in the next section (where `Samp` can be thought of as $\mathcal{A}$ and `Check` has the same purpose as $\mathcal{O}_{f,\pm}$).

5.2.1 Fixed-point amplitude amplification

In Chapter 6, we will consider a variant of amplitude amplification known as *fixed-point* amplitude amplification [YLC14; GSLW19]. It ensures that the output state can be made arbitrarily close to the desired state $|\Psi\rangle$ from Equation (5.2) (assuming $\Pi|\psi\rangle \neq 0$), even if we run for more steps than necessary. (This avoids a phenomenon known as the "soufflé problem".) The version that we give below additionally ensures that the algorithm always stops after a fixed number of steps, even if $|\psi\rangle$ does not overlap the marked space at all.

Lemma 5.2.2 (Fixed-point AA (implicit in [YLC14; GSLW19]))**.** *Let* Samp *be a unitary implementable in time* S*, and let* $|\psi\rangle := \mathsf{Samp}\,|0\rangle$*. Let* Π *be a projector on the Hilbert space* $\mathcal{H}$ *of* $|\psi\rangle$*, and let* Check *be a unitary implementable in time* C *such that, for all* $|\phi\rangle \in \mathcal{H}$*,* $\mathsf{Check}\,|\phi\rangle\,|0\rangle = \Pi\,|\phi\rangle\,|1\rangle + (I-\Pi)\,|\phi\rangle\,|0\rangle$*. For all* $r \geq 1$ *and* $\delta \in (0, 1/2]$*, there exists a quantum algorithm* $\mathsf{AA}_r(\mathsf{Samp}, \mathsf{Check})$ *that satisfies the following, except with probability* δ*:*

(*i*) *If* $\|\Pi\,|\psi\rangle\| \geq 1/r$*, then it returns the quantum state* $|\psi_{\text{out}}\rangle = \frac{\Pi|\psi\rangle}{\|\Pi|\psi\rangle\|}\,|1\rangle$ *in time*

$$O(\log(1/\delta)\|\Pi\,|\psi\rangle\|^{-1}(\mathsf{S}+\mathsf{C})).$$

(*ii*) *If* $\|\Pi\,|\psi\rangle\| = 0$*, then it returns the quantum state* $|\psi_{\text{out}}\rangle = |\psi\rangle\,|0\rangle$ *in time*

$$O(\log(1/\delta)\, r\,(\mathsf{S}+\mathsf{C})).$$

We call the auxiliary qubit in the output of amplitude amplification its *flag* qubit or *flag* register. When we apply Lemma 5.2.2, we will often not state the choice of δ explicitly, but implicitly take it small enough to ensure its contribution to the overall cost is a negligible factor for the application at hand.

Proof. By [YLC14; GSLW19], there exists a universal constant $\eta > 0$ and a quantum algorithm $\mathcal{A}_{r'} = \mathcal{A}_{r'}(\mathsf{Samp}, \mathsf{Check})$ that generates a state $|\psi_{\text{out}}\rangle$ in time $O(r'(\mathsf{S}+\mathsf{C}))$, satisfying:

(1) If $\|\Pi\,|\psi\rangle\| \neq 0$ and $r' \geq \eta \log(\frac{1}{\delta})\|\Pi\,|\psi\rangle\|^{-1}$, then $\||\psi_{\text{out}}\rangle - \frac{\Pi|\psi\rangle}{\|\Pi|\psi\rangle\|}\,|1\rangle\| \leq \delta$.
(2) If $\|\Pi\,|\psi\rangle\| = 0$ and $r' \geq 1$, then $|\psi_{\text{out}}\rangle = |\psi\rangle\,|0\rangle$.

Our lemma extends this result slightly: in part (*i*), we require the runtime to be much smaller than in part (*ii*) whenever the actual (possibly unknown) value of $\|\Pi\,|\psi\rangle\|$ is significantly larger than the lower bound $1/r$ that our algorithm has for it.

Specifically, we define the quantum algorithm $\mathsf{AA}_r(\mathsf{Samp}, \mathsf{Check})$ as follows. It first runs $\mathcal{A}_{r'}$ with $r' = 1$ and measures the auxiliary qubit. If the measurement outcome is 1, then it stops and outputs the resulting state. Otherwise, it doubles r' (i.e., replaces r' by $2r'$) and repeats. This continues until $r' \geq 2\eta \log(2/\delta)r$, at which point it outputs the current state.

By construction, $\mathsf{AA}_r(\mathsf{Samp}, \mathsf{Check})$ always terminates, and outputs a state $|\psi_{\text{out}}\rangle$. Note that if $\mathsf{AA}_r(\mathsf{Samp}, \mathsf{Check})$ terminates after ℓ repetitions, then the aforementioned result implies that the total runtime is $O(2^\ell(\mathsf{S}+\mathsf{C}))$, because $\sum_{i=1}^{\ell} 2^i = O(2^\ell)$.

Suppose $\|\Pi|\psi\rangle\| \geq 1/r$. Let ℓ be the smallest integer such that

$$2^\ell \geq \eta \log(2/\delta)\|\Pi|\psi\rangle\|^{-1}.$$

By statement (1), the inner product α between $\frac{\Pi|\psi\rangle}{\|\Pi|\psi\rangle\|}|1\rangle$ and the output state $|\psi_{\text{out}}\rangle$ of $\mathcal{A}_{2^\ell}$ satisfies $|\alpha| \geq 1-\delta/2$, so the probability that measuring the auxiliary qubit yields outcome 1 is at least $(1-\delta/2)^2 \geq 1-\delta$. In other words, with probability at least $1-\delta$, $\mathsf{AA}_r(\mathsf{Samp}, \mathsf{Check})$ generates the state $|\psi_{\text{out}}\rangle = \frac{\Pi|\psi\rangle}{\|\Pi|\psi\rangle\|}|1\rangle$ after at most ℓ repetitions, from which part (i) follows.

On the other hand, if $\|\Pi|\psi\rangle\| = 0$, then statement (2) implies that $\mathsf{AA}_r(\mathsf{Samp}, \mathsf{Check})$ always generates $|\psi_{\text{out}}\rangle = |\psi\rangle|0\rangle$. The complexity claim follows because, by construction of the algorithm, it terminates after ℓ repetitions, for some ℓ satisfying $2^\ell = O(\log(1/\delta)r)$. □

5.3 Quantum-walk algorithms

Another class of quantum algorithms that has seen use in cryptanalysis are quantum-walk algorithms, also shortened as *quantum walks*. Quantum walks can be seen as a quantum analog of classical random walks. In some sense, they are a generalization of Grover's search algorithm (see Example 5.3.1).

Although various algorithmic frameworks have been proposed for quantum walks, we will focus on the so-called MNRS framework for solving search tasks.

5.3.1 MNRS framework

The MNRS quantum-walk framework is named after its inventors: Magniez, Nayak, Roland, and Santha [MNRS11]. This framework builds on Ambainis's breakthrough quantum-walk algorithm for element distinctness [Amb04]. As with classical random walks, the quantum-walk algorithm is designed to find a "marked" element in a finite set, but in contrast to unstructured search, this set will be the state space of a Markov chain, so there may be some structure to exploit. The algorithm can therefore be used to solve search problems by encoding them as a Markov chain. While the state space can be chosen to be the search space (with marked states corresponding to actual solutions), the framework is much more general, allowing for other ways to design the state space and marked elements in order to guide the search for solutions.

Slightly simplified for our purposes, the setup of the MNRS framework is as follows.[2] Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, connected, regular graph with finite vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. The goal is to find one or more "marked" vertices from a certain subset $\mathcal{M} \subseteq \mathcal{V}$, or to conclude that $\mathcal{M}$ is empty. To achieve this goal, we will assume that we can check membership in $\mathcal{M}$. In some cases, it is helpful to store some additional data $D(S)$ with each vertex $S \in \mathcal{V}$ (using some data structure that is maintained throughout the algorithm).

Let δ denote the spectral gap[3] of $\mathcal{G}$ (note that δ had a different meaning in the previous section), and let $\varepsilon > 0$ be guaranteed to satisfy $\varepsilon \leq |\mathcal{M}|/|\mathcal{V}|$ if $\mathcal{M}$ is nonempty. That is, ε is a lower bound on the probability that a uniformly random vertex is marked. Consider the following classical random-walk strategy:

1. Sample $S \sim U(\mathcal{V})$ and compute $D(S)$
2. Repeat $O(1/\varepsilon)$ times:
   (a) If $S \in \mathcal{M}$ (possibly checked using $D(S)$), return S
   (b) Repeat $O(1/\delta)$ times:
      i. Sample $S' \sim U(\Gamma(S))$, where $\Gamma(S) := \{S' \in \mathcal{V} : \{S, S'\} \in \mathcal{E}\}$
      ii. Set $S := S'$, and update $D(S)$ accordingly
3. Return "no marked element exists"

Repeating step 2(b) for $O(1/\delta)$ times has the effect of sampling a vertex uniformly at random (related to the *mixing time* of the random walk). After one repetition of step 2, the probability that the current vertex is marked is then $|\mathcal{M}|/|\mathcal{V}|$. To find a marked vertex with probability at least $2/3$, it therefore suffices to repeat step 2 for $O(|\mathcal{V}|/|\mathcal{M}|)$ times (related to the *hitting time* of the random walk), which we replace by $O(1/\varepsilon)$ if we only know a lower bound on $|\mathcal{M}|$.

Let S denote the cost of step 1, C the cost of the check in step 2(a), and U the cost of step $2(b)$. Then this classical strategy finds a marked vertex with probability at least $2/3$, roughly in total cost

$$\mathsf{S} + \frac{1}{\varepsilon}\left(\mathsf{C} + \frac{1}{\delta}\mathsf{U}\right).$$

[2] The results in [MNRS11] apply more generally to reversible, ergodic Markov chains.

[3] The spectral gap of $\mathcal{G}$ is defined as $\delta = 1 - |\lambda|$, where 1 and λ are the largest and second largest eigenvalues (in magnitude) of the normalized adjacency matrix of $\mathcal{G}$ (with 1 corresponding to the uniform eigenvector).

The total cost generally depends on parameters determined by graph $\mathcal{G}$, the definition of $\mathcal{M}$, and the choice of auxiliary data structure. When using this classical strategy to solve a certain search problem, one can then attempt to minimize the total cost by carefully choosing the encoding ($\mathcal{G}$ and $\mathcal{M}$) and by constructing a suitable data structure.

Using a quantum variant of this strategy, the MNRS framework [MNRS11] essentially achieves a quadratic speedup in the terms $1/\varepsilon$ and $1/\delta$ (for fixed $\mathcal{G}$ and $\mathcal{M}$). This allows for a different trade-off between the cost components of the algorithm. These speedups are, however, not simply obtained by starting with a uniform superposition over all vertices and adding amplitude amplification on top of steps 2 and 2(b), as more changes are needed to obtain a valid quantum algorithm. For example, at each step of the classical strategy, the algorithm can simply forget where it came from, so the classical approach "walks" on the vertices of the graph $\mathcal{G}$ (and their data). In the quantum setting, each step must instead be reversible, which is essentially achieved by keeping track of two quantum registers, with one corresponding to the current vertex and the other one to the previous vertex. The quantum walk can therefore be viewed as walking on the *edges* of $\mathcal{G}$ (and the corresponding data).

Rephrased as a walk on a graph $\mathcal{G}$ (rather than as a Markov chain), the main result of [MNRS11] is Theorem 5.3.2. The costs S, C, and U are respectively known as the *setup cost*, *checking cost*, and *update cost* of the quantum walk.

It is well known that Theorem 5.3.2 can be used to recover Grover's complexity (for example, see [San08]).

Example 5.3.1 (Recovering Grover's complexity). Let $f\colon [N] \to \{0,1\}$ be a function. Consider the task of finding a preimage of 1 or determining that none exists, where we are promised that $t := |\{i \in [N]\colon f(i) = 1\}|$ satisfies either $t \geq t_0 \geq 1$ or $t = 0$. This problem can be naturally encoded as the task of finding a marked vertex in the complete graph $\mathcal{G}$ on N vertices with $\mathcal{M} := \{i \in [N]\colon f(i) = 1\}$. ($\mathcal{G}$ has vertex set $\mathcal{V} = [N]$ and each pair of distinct vertices forms an edge.) The spectral gap of this graph is known to be $\delta = 1 - 1/(N-1) = \Omega(1)$, and we can take $\varepsilon = t_0/N$. Moreover, $\mathsf{S} = \mathsf{U} = \tilde{O}(1)$, and C is the cost of $O(1)$ applications of the unitary $\mathcal{O}_{f,\pm}$ defined in Equation (5.1). Theorem 5.3.2 then indeed recovers Theorem 5.1.2.

Theorem 5.3.2 (Quantum-walk search [MNRS11]). *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, connected, regular graph with finite vertex set $\mathcal{V}$, edge set $\mathcal{E}$, and spectral gap at least δ. Let $\mathcal{M} \subseteq \mathcal{V}$ and $\varepsilon > 0$ be such that either $\mathcal{M} = \emptyset$ or $\varepsilon \leq |\mathcal{M}|/|\mathcal{V}|$. For all $S \in \mathcal{V}$, let $|D(S)\rangle$ be a (possibly empty) classical or quantum data structure, and define the following unitary maps:*

$$\texttt{Setup}\colon |0\rangle \mapsto \frac{1}{\sqrt{|\mathcal{V}|}} \sum_{S \in \mathcal{V}} |S, D(S)\rangle$$

$$\texttt{Check}\colon |S, D(S)\rangle \mapsto \begin{cases} -|S, D(S)\rangle & \text{if } S \in \mathcal{M} \\ |S, D(S)\rangle & \text{otherwise} \end{cases}$$

$$\texttt{Update}\colon |S, D(S)\rangle\,|0\rangle \mapsto |S, D(S)\rangle \frac{1}{\sqrt{|\Gamma(S)|}} \sum_{S' \in \Gamma(S)} |S', D(S')\rangle\,.$$

There exists a quantum algorithm that, given δ, ε, and implementations of the above unitary maps, with probability at least $2/3$ returns a marked vertex $S \in \mathcal{M}$, or correctly outputs that $\mathcal{M}$ is empty, in time

$$\tilde{O}\left(\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}\left(\mathsf{C} + \frac{1}{\sqrt{\delta}}\mathsf{U}\right)\right)$$

where S, C, U *denote the time required to implement* `Setup`, `Check`, *and* `Update`, *respectively. It uses $O(n + n_{\text{aux}})$ qubits, where n is the number of qubits required to represent the states $|S, D(S)\rangle$ for $S \in \mathcal{V}$, and n_{aux} the number of auxiliary qubits required to implement the above unitary maps.*
Moreover, if $\mathcal{M} \neq \emptyset$, then for all $S \in \mathcal{M}$, the probability that the algorithm returns S is $\Omega(1/|\mathcal{M}|)$.

Johnson graph. One graph that has been used repeatedly to obtain quantum speedups via quantum walks is the Johnson graph.

> **Definition 5.3.3** (Johnson graph)**.** Given positive integers $N \geq s$, the *Johnson graph*, denoted $J(N, s)$, is the graph with vertex set $\mathcal{V} = \{S \subseteq [N] : |S| = s\}$ and edge set $\mathcal{E} = \{\{S, S'\} : S, S' \in \mathcal{V}, |S \cap S'| = s - 1\}$.

Each vertex in a Johnson graph therefore has $s(N-s)$ neighbors. Note that $J(N, 1)$ is the complete graph on N vertices. For $s > 2$, the Johnson graph $J(N, s)$ has spectral gap $\delta = \frac{N}{s(N-s)}$ [BH12, Chapter 12], and thus $\delta \geq \frac{1}{s}$.

The Johnson graph is particularly useful in algorithms for collision problems, which arise naturally in cryptanalysis. For instance, the best known (heuristic) quantum algorithm for solving SVP [BCSS23] (building on [CL21]) makes use of a quantum-walk in the Johnson graph, and we will consider a related quantum-walk algorithm for code-based cryptanalysis in Chapter 7.

5.4 Nesting quantum algorithms

In the next chapter, we will consider a quantum algorithm that consists of several layers of amplitude amplification, constructed in a way that resembles "two-oracle search" as defined below.

5.4.1 Two-oracle search

The problem solved by Grover's algorithm [Gro96] could be called *oracle search*. We recall the setup. Consider a set M_0 of elements that are easy to "sample", meaning that we can generate some superposition over the elements of M_0, where we think of the squared amplitudes as the sampling probabilities. For a marked subset $M_1 \subseteq M_0$, consider the problem of searching over M_0 for an element of M_1. If we can sample an element of M_0 in cost S and we can check membership in M_1 in cost C, then amplitude amplification (Section 5.2) finds an element of M_1 in cost (neglecting constants)

$$\frac{1}{\sqrt{\varepsilon}}(\mathsf{S} + \mathsf{C})$$

where ε is the probability that an element sampled from M_0 is in M_1. In our applications, the sampling cost S is negligible, resulting in cost $\frac{1}{\sqrt{\varepsilon}}\mathsf{C}$.

Oracle search is often called *unstructured search*, since the oracle abstracts away any potential structure of the problem that an algorithm might be able to take advantage of. In some cases, we can take advantage of a small amount of structure, such as in the setting of *two-oracle search*, first studied in [KLL15] for the case of a unique marked element. In this problem, one wants to find an element of $M_2 \subseteq M_0$, but in addition to having access to an oracle for checking membership in M_2 (at cost C_2), one also has access to an oracle for checking membership in a set M_1 such that $M_2 \subseteq M_1 \subseteq M_0$ (at cost C_1). See Figure 5.1 for a depiction. Assuming $\mathsf{C}_1 \ll \mathsf{C}_2$, this additional structure gives an advantage, intuitively because one can always first cheaply check membership in M_1, and for nonmembers, not waste time on a more expensive membership check in M_2. This significantly reduces the number of times we check membership in M_2. By variable-time search [Amb10], if ε_1 is the probability that an element sampled from M_0 is in M_1, and $\varepsilon_2 \leq \varepsilon_1$ is the probability that an element sampled from M_0 is in M_2, then a quantum algorithm can find an element of M_2 in cost

$$\sqrt{\frac{\varepsilon_1}{\varepsilon_2}}\left(\frac{1}{\sqrt{\varepsilon_1}}\mathsf{C}_1 + \mathsf{C}_2\right) = \frac{1}{\sqrt{\varepsilon_2}}\mathsf{C}_1 + \sqrt{\frac{\varepsilon_1}{\varepsilon_2}}\mathsf{C}_2$$

where we neglect logarithmic factors and assume the cost S of sampling from M_0 is negligible. When $\mathsf{S}, \mathsf{C}_1 < \mathsf{C}_2$ and $\varepsilon_1 < 1$, this improves over the cost $\frac{1}{\sqrt{\varepsilon_2}}\mathsf{C}_2$ of amplitude amplification.

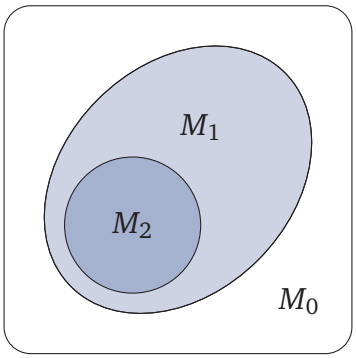


Figure 5.1: In two-oracle search, the task is to search for elements in a search space M_0 that belong to a marked subset $M_2 \subseteq M_0$, given the ability to check membership in both M_2 and some subset M_1 satisfying $M_2 \subseteq M_1 \subseteq M_0$.

Chapter 6

Quantum lattice sieving for solving SVP

The motivating problem of this chapter is the Shortest Vector Problem (SVP), whose assumed hardness is one of the cornerstones of post-quantum cryptography. We will look at its complexity through the lens of a quantum attacker, focusing on the fastest known family of attacks: heuristic lattice sieving. Specifically, we present a quantum algorithm that — through a combination of nested amplitude amplification and structured preprocessing — speeds up the main computational algorithm underlying 3-tuple sieving. Under a common heuristic, this results in the fastest known quantum algorithm for SVP in dimension n when total memory is limited to $2^{0.1887n}$. One of the main messages of this chapter is that the toolbox of quantum techniques used for sieving is not yet exhausted, suggesting that there may be room to further improve the asymptotic quantum complexity of SVP.

This chapter is based on [ECGH+25].

Contents of this chapter

6.1 Overview

6.1.1 Problem setting

The security of lattice-based cryptosystems is closely tied to the hardness of the Shortest Vector Problem (SVP), defined in Problem 2.3.8. Given linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k \in \mathbb{R}^n$, exact SVP asks for a shortest nonzero vector in the lattice

$$\Lambda = \left\{ \sum_{i=1}^{k} z_i \mathbf{b}_i : z_1, \ldots, z_k \in \mathbb{Z} \right\}$$

obtained by taking all integer linear combinations of these vectors. We denote the norm of such a vector by $\lambda_1(\Lambda)$.

Lattice-based cryptography is premised on the assumption that no efficient algorithm exists even for finding a nonzero vector of norm at most $\gamma\lambda_1(\Lambda)$ for sufficiently small γ (say, some polynomial in n), a variant known as approximate SVP. Indeed, because of the worst-case to average-case reductions mentioned in Section 2.3.2, the common belief that approximate SVP is hard for reasonably small γ is one of the reasons why SIS and LWE are used as a foundation for cryptography. Moreover, many lattice attacks, such as those relying on the BKZ algorithm [Sch87; SE94], require a subroutine for solving approximate SVP.

The fastest provable classical algorithm for SVP has runtime essentially 2^n on worst-case instances [ADRS15], which was improved to runtime roughly $2^{0.95n}$ on a quantum computer, and even to $2^{0.835n}$ on a quantum computer with a large QCRAM [ACKS25]. For meaningfully breaking cryptosystems, worst-case algorithms are more than needed and it suffices to have a fast *heuristic* algorithm, one that works for most (practical) instances under some plausible, though not quite rigorous, assumptions. After all, no one would use a cryptographic system that is known to be breakable with a significant probability. The fastest heuristic methods known for approximate SVP still take exponential time 2^{cn}, but with a constant $c < 1$ that is much smaller than for the best known worst-case algorithms. These heuristic methods are based on sieving ideas, first introduced in the context of SVP by Ajtai, Kumar, and Sivakumar [AKS01] and made more practical by Nguyen and Vidick [NV08].[1]

[1] In fact, the best known non-heuristic, worst-case algorithm for SVP [ADRS15] can also in some sense be viewed as a "sieving" algorithm.

The most basic type of sieving is 2-*tuple sieving*. To find a short vector in a lattice Λ, the idea is to begin by sampling a large list L of m random lattice vectors of roughly the same norm R, where R is chosen large enough to allow efficient sampling (in fact, we may assume $R = 1$ by scaling the lattice). We then try to find slightly shorter vectors among the sums or differences of pairs of vectors $\mathbf{x}, \mathbf{y} \in L$ (note that $\mathbf{x}+\mathbf{y}$ and $\mathbf{x}-\mathbf{y}$ are still in the lattice Λ), which yields a list L' of shorter vectors that will form the starting point of the next sieving iteration. We can then form a new list L'' of lattice vectors of even shorter norm by taking sums or differences of pairs of vectors from L', and so on.

In order to understand how large the initial list size m should be to succeed, let us forget for a moment that we are working in a lattice. Given two uniformly random unit vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, the probability that $\mathbf{x}+\mathbf{y}$ or $\mathbf{x}-\mathbf{y}$ has norm < 1 is $p \approx 2^{-0.2075n}$, which follows from basic estimates of the area of a cap on an n-dimensional sphere. Thus, given a list of m i.i.d. uniformly random unit vectors in $\mathbb{R}^n$, there are $\binom{m}{2}$ pairs of vectors, so the expected number of "good" pairs inducing a shorter vector is $p\binom{m}{2}$. Hence, choosing $m \approx 2/p \approx 2^{0.2075n}$ ensures there exist roughly m shorter vectors that can be used for the next sieving iteration.

The catch is that the list L that we start with actually consists of random *lattice* vectors, which is essentially what makes the approach heuristic. Specifically, the analysis of 2-tuple sieving assumes that, in each iteration, the normalized vectors in the current list behave like i.i.d. uniform vectors on the unit sphere $\mathcal{S}^{n-1} \subseteq \mathbb{R}^n$. While this is not an issue for the first iteration, it becomes a heuristic assumption in later iterations. This heuristic assumption is backed by extensive experiments [NV08; BLS16; ADHK+19], and has become the standard for analyzing lattice sieving.

As a polynomial (in n) number of sieving iterations suffices to find a rather short lattice vector (see Section 6.4), the overall cost of this approach for solving approximate SVP is dominated by the cost of one iteration, that is, the cost of finding m good pairs among $\binom{m}{2}$ pairs. How much time does this take? Nguyen and Vidick [NV08] used brute-force search over all $\binom{m}{2}$ pairs, which takes time $O(m^2) = O(2^{0.415n})$. This runtime has since been improved by using *nearest-neighbor* data structures (in particular, locality-sensitive filtering techniques) that allow us to quickly find, for a given $\mathbf{x} \in L$, a small number of $\mathbf{y} \in L$ that are somewhat close to $\mathbf{x}$.

Various *quantum* subroutines have been used to achieve additional speedups in the search for good pairs: Grover's search algorithm [LMP15; Laa15], quantum walks [CL21], and reusable quantum walks [BCSS23]. Currently, the best

known classical and quantum runtimes are $2^{0.2925n+o(n)}$ [BDGL16] and $2^{0.2563n+o(n)}$ [BCSS23], respectively, which are the fastest known heuristic attacks on SVP. These attacks, however, do require at least $2^{0.2075n}$ bits of memory.

The 2-tuple-sieving approach can be generalized to k-tuple sieving for some larger constant $k \geq 3$ in the natural way [BLS16]: by looking for k-tuples $\mathbf{x}_1, \ldots, \mathbf{x}_k$ from the current list L that satisfy $\|\mathbf{x}_1 \pm \mathbf{x}_2 \pm \cdots \pm \mathbf{x}_k\| < 1$ for some choice of the coefficients ± 1 (note that the number 2^{k-1} of possible sign patterns disappears in the big-O because k is a constant). The advantage of going to $k > 2$ is that the algorithm requires less memory, as the initial list size m can be smaller (see [HK17]) while still giving roughly m good k-tuples that yield shorter lattice vectors for the next iteration. The disadvantage, on the other hand, is that the time to find m good k-tuples increases with k, because the set of k-tuples has $\binom{m}{k}$ elements, which grows with k, even when we take into account that the minimal required list size m itself goes down with k. For example, for 2-tuple sieving and 3-tuple sieving, the required list sizes are $m_2 \approx 2^{0.2075n}$ and $m_3 \approx 2^{0.1887n}$, respectively, yet $\binom{m_3}{3} \gg \binom{m_2}{2}$ despite the fact that $m_3 \ll m_2$.

As in 2-tuple sieving, a relatively small (polynomial in n) number of iterations suffices for finding short lattice vectors. Hence, the overall cost of k-tuple sieving is dominated by the cost of finding m good k-tuples among the set of all $\binom{m}{k}$ k-tuples from the given list of m vectors. Studying the complexity of the latter problem is therefore crucial for understanding the hardness of SVP in limited (exponential) memory regimes. While quantum speedups are known for k-tuple sieving with $k > 2$ (e.g., [KMPM19; CL23]), these variants appear to have been explored less than $k = 2$. Hence, this chapter is concerned with the following question:

Can we improve the best known quantum runtime for 3-tuple lattice sieving, in order to obtain a faster quantum algorithm for SVP in dimension n when total memory is limited to $2^{0.1887n}$?

6.1.2 Contributions

Our main result is an improvement of the quantum time complexity of 3-tuple sieving, from $2^{0.3098n+o(n)}$ to $2^{0.2846n+o(n)}$, yielding a positive answer to the previous question. We summarize the resulting best known classical and quantum runtimes for k-tuple sieving with $k = 2, 3, 4$ in Table 6.1.

k	Memory	Classical time	Quantum time
2	0.2075	0.2925 [BDGL16]	0.2563 [BCSS23]
3	0.1887	0.3383 [CL23]	0.2846 [ECGH+25] *(this chapter)*
4	0.1724	0.3766 [HKL18]	0.3178 [KMPM19, App. B.2]

Table 6.1: Exponents of the best known classical and quantum time complexities for $\{2, 3, 4\}$-tuple sieving with minimal memory usage, suppressing $o(1)$ terms. The main result in this chapter improves the quantum time exponent for $k = 3$ from 0.3098 [CL23] to 0.2846, while keeping the memory exponent at 0.1887.

Interestingly, this means quantum 3-tuple sieving is now faster than classical 2-tuple sieving, which is still the fastest heuristic classical algorithm for SVP. (This is true besides the fact that 3-tuple sieving uses less memory than 2-tuple sieving, which is the main advantage of taking $k > 2$.) Moreover, this asymptotic improvement is significantly larger than other recent progress in heuristic quantum algorithms for SVP: for comparison, the most recent improvement for quantum 2-tuple sieving was from exponent 0.2570 [CL21] to 0.2563 [BCSS23].

Our speedup is achieved using a strategy that differs from prior quantum algorithms for k-tuple sieving and that may extend to speedups for $k \neq 3$. We carefully nest amplitude amplification and balance the different terms in the time complexity via a preprocessing step, leveraging the fact that we can afford relatively expensive preprocessing in terms of time and memory. This use of preprocessing and nested quantum subroutines is not limited to lattice sieving, and may find other quantum cryptanalytic applications.

6.1.3 Technical overview

The core computational problem that we would like to solve, as it dominates the cost of 3-tuple sieving, is the following:

> **Problem 6.1.1** (Finding many solutions)**.** Given a list L of m i.i.d. uniform samples from the unit sphere in $\mathbb{R}^n$, find m 3-tuples of distinct $\mathbf{x}, \mathbf{y}, \mathbf{z} \in L$ such that $\|\mathbf{x} - \mathbf{y} - \mathbf{z}\| \leq 1$. We will refer to such a tuple $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ as a 3-*tuple solution*.

As argued before, we could also allow all sign patterns $\|\mathbf{x} \pm \mathbf{y} \pm \mathbf{z}\|$, but asymptotically this does not matter because there are only 4 such patterns. We also assumed here for simplicity that the m initial vectors all have norm exactly 1.[2] To ensure that m such tuples exist with high probability over the choice of L, the list size m has to be at least roughly $(27/16)^{n/4+o(n)} \approx 2^{0.1887n}$ [HK17], providing a memory lower bound for 3-tuple sieving.

Expanding $\|\mathbf{x} - \mathbf{y} - \mathbf{z}\|^2$ shows that unit vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ satisfy $\|\mathbf{x} - \mathbf{y} - \mathbf{z}\| \leq 1$ if and only if

$$\langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x} - \mathbf{y}, \mathbf{z} \rangle \geq 1.$$

This fact lets us replace the condition $\|\mathbf{x} - \mathbf{y} - \mathbf{z}\| \leq 1$ by two conditions on the inner products $\langle \mathbf{x}, \mathbf{y} \rangle$ and $\langle \mathbf{x} - \mathbf{y}, \mathbf{z} \rangle$, motivating us to search for 3-tuple solutions $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ by first searching for a pair $(\mathbf{x}, \mathbf{y})$ with sufficiently large inner product. Our main result is indeed a faster quantum algorithm for a mildly relaxed version of Problem 6.1.1, which can be slightly simplified (by omitting approximations in the inner products) as follows:

> **Problem 6.1.2** (Finding many solutions with a stronger property)**.** Given a list L of m i.i.d. uniform samples from the unit sphere in $\mathbb{R}^n$, find m 3-tuples of distinct $\mathbf{x}, \mathbf{y}, \mathbf{z} \in L$ such that $\langle \mathbf{x}, \mathbf{y} \rangle \geq 1/3$ and $\langle \mathbf{x} - \mathbf{y}, \mathbf{z} \rangle \geq 2/3$ (implying $\|\mathbf{x} - \mathbf{y} - \mathbf{z}\| \leq 1$).

The latter problem demands something stronger than $\|\mathbf{x} - \mathbf{y} - \mathbf{z}\| \leq 1$, thereby slightly increasing the list size m required to ensure that m triples satisfying this stronger property exist with high probability over the input. Fortunately, the above choice of inner-product conditions guarantees that the minimal list size m is affected only by a factor $2^{o(n)}$ (e.g., see [HK17] or Lemma 6.3.17 below). This factor is negligible for our asymptotic purposes, and allows us to solve Problem 6.1.1 with list size $m \approx 2^{0.1887n}$ using an algorithm for Problem 6.1.2.

[2] It may seem odd to start with vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ of norm 1 and then to find new vectors $\mathbf{x} - \mathbf{y} - \mathbf{z}$ whose norm is still (at most) 1. In practice, one aims at finding vectors with norm $\leq 1 - \mu$ for some $\mu > 0$ that is inverse-polynomially small in n: big enough to ensure that a polynomially-small number of sieving iterations results in quite short vectors, and small enough to only affect the runtime up to subexponential factors (i.e., $o(1)$ in the exponent). Aiming at norm 1 rather than $1 - \mu$ is just to simplify notation.

A two-step procedure to find many 3-tuple solutions. We will find those m triples in Problem 6.1.2 one by one. To find one good triple, consider the following approach:

1. Create a uniform superposition over all $\mathbf{x} \in L$, and conditioned on $\mathbf{x}$ use amplitude amplification to create (in a second register) a superposition over all $\mathbf{y} \in L$ such that $\langle \mathbf{x}, \mathbf{y} \rangle \geq 1/3$.
2. Starting from the state of the previous step, conditioned on $\mathbf{x}, \mathbf{y}$ use amplitude amplification to create (in a third register) a uniform superposition over all $\mathbf{z} \in L$ such that $\langle \mathbf{x} - \mathbf{y}, \mathbf{z} \rangle \geq 2/3$, and set a "flag" qubit to 1 if such a $\mathbf{z}$ exists.

Then use amplitude amplification on top of this two-step procedure to amplify the part of the state where the flag is 1. Measuring the resulting state gives us one of the triples $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ that we want. Repeating $\tilde{O}(m)$ times, we will obtain (except with negligibly small error probability) m triples satisfying the stronger property, and hence m new vectors with norm at most 1.

Because of the somewhat surprising properties of "two-oracle search" [KLL15] (recall Section 5.4.1), this approach is already better than a basic amplitude amplification on all m^3 triples $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. To see this, ignoring polylogarithmic factors, define $p := \Pr[\langle \mathbf{x}, \mathbf{y} \rangle \geq 1/3]$ and $p' := [\langle \mathbf{x} - \mathbf{y}, \mathbf{z} \rangle \geq 2/3 \mid \langle \mathbf{x}, \mathbf{y} \rangle \geq 1/3]$ for i.i.d. uniform unit vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$. When the expected number of triples satisfying both inner-product constraints is m, we have $m^3 pp' \approx m$, so $mp' \approx 1/mp$. Step 1 takes time $1/\sqrt{p}$, and step 2 takes time $\sqrt{m/\max\{1, mp'\}} \approx \sqrt{\min\{m, m^2 p\}}$ (as it searches L for an expected number of mp' valid $\mathbf{z}$). One iteration of these steps yields a good triple with probability $\min\{mp', 1\} \approx \min\{1/mp, 1\}$, so the overall procedure finds one good triple in time

$$\underbrace{\max\{\sqrt{mp}, 1\}}_{\text{outer amplification}} \left(\underbrace{\frac{1}{\sqrt{p}}}_{\text{step 1}} + \underbrace{\sqrt{\min\{m, m^2 p\}}}_{\text{step 2}} \right) = \max\left\{ \sqrt{m}, \frac{1}{\sqrt{p}} \right\} + m\sqrt{p}.$$

In contrast, amplitude amplification over L^3 takes time $\sqrt{m^3/m} = m$ to find a good triple. (Note that $1/\sqrt{p} < m$ as $m^2 pp' \approx 1$ and p' exponentially small.)

Yet, when executed as stated, this two-step procedure can be shown to yield an overall runtime of roughly $2^{0.3350n}$ to find $m \approx 2^{0.1887n}$ good triples. While this certainly improves over just amplitude amplification (time $m^2 \approx 2^{0.3774n}$) as well as the best known classical runtime ($2^{0.3383n}$ by [CL23]), it does not yet

outperform the state-of-the-art quantum runtime of $2^{0.3098n}$ by [CL23]. Moreover, the costs of steps 1 and 2 can be seen to be unbalanced, suggesting there may be room for improvement.[3]

Improvement using locality-sensitive filtering and preprocessing. To get a faster algorithm, we improve the two-step procedure by locality-sensitive filtering for good triples using random product codes (RPCs, following [BDGL16]), implemented via preprocessing. Roughly speaking, the idea here is to choose a certain number of sufficiently random "center points", and do a preprocessing step that, for each of the m vectors in L, writes down their closest center points. The properties of RPCs allow us to do this efficiently. Then for a given $\mathbf{x}$ in step 1, we can improve the search for a $\mathbf{y} \in L$ satisfying $\langle \mathbf{x}, \mathbf{y} \rangle \geq 1/3$ by restricting the search to those $\mathbf{y} \in L$ that share a close center point with $\mathbf{x}$. Those $\mathbf{y}$ are relatively close to $\mathbf{x}$, and thus more likely to satisfy $\langle \mathbf{x}, \mathbf{y} \rangle \geq 1/3$ (meaning that we can amplify the desired ones more efficiently). Similarly, for step 2, it is easier to find a $\mathbf{z} \in L$ close to $\mathbf{x} - \mathbf{y}$ if we focus only on those $\mathbf{z} \in L$ that share a close center point with $\mathbf{x} - \mathbf{y}$. This approach may overlook some close $(\mathbf{x}, \mathbf{y})$ pairs (if $\mathbf{x}$ and $\mathbf{y}$ do not share a close center point) or some $\mathbf{z} \in L$ close to $\mathbf{x} - \mathbf{y}$ (if they do not share a close center point). However, we can show that a careful application of these ideas enables us to balance the costs of steps 1 and 2, and that this modified method (repeated $\tilde{O}(m)$ times and choosing a fresh RPC once in a while to ensure no good triple is overlooked all the time) will find m good triples. This results in a quantum algorithm that solves Problem 6.1.1 with list size $m \approx 2^{0.1887n}$ in time $2^{0.2846n}$, giving a speedup over the previous best known time complexity in this minimal memory regime.

6.1.4 Organization

Section 6.2 provides chapter-specific background, including on the model for data structures we use and on the properties of the unit sphere and RPCs that are relevant to our analysis. In Section 6.3, we present our new quantum algorithm for finding many 3-tuple solutions. In Section 6.4, we apply our quantum algorithm to SVP under a common heuristic. Finally, in Section 6.5, we discuss some directions for future work.

[3] As $m = (27/16)^{n/4+o(n)} \approx 2^{0.1887n}$ and $p \approx (8/9)^{n/2}$ (see Lemma 6.2.4), we obtain $mp \gg 1$. Step 2 is therefore more expensive than step 1, and repeating the overall procedure about m times solves Problem 6.1.2 in time roughly $m^2\sqrt{p} \approx 2^{0.3350n}$.

6.2 Preliminaries

Notation. Throughout this chapter, we use n to denote the dimension of the ambient space $\mathbb{R}^n$. The set of rotation matrices over $\mathbb{R}^n$ is denoted by $\mathrm{SO}(n)$. The set of unit vectors in $\mathbb{R}^n$ (i.e., the sphere of the n-dimensional Euclidean unit ball $\mathcal{B}_n$) is denoted by $\mathcal{S}^{n-1} := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| = 1\}$.

As motivated by Section 6.2.3, we will write p_θ as shorthand for $(1-\cos^2(\theta))^{n/2}$. For $x, y \in \mathbb{R}$ and $\epsilon > 0$, we write $x \approx_\epsilon y$ if $|x - y| \leq \epsilon$. In particular, there will be some dependency on a parameter ϵ in our analysis, which we will fix as $\epsilon := 1/(\log n)^2$ (which suffices to absorb the dependency into the $2^{o(n)}$ notation). We also write $x =_n y$ as shorthand for $2^{-o(n)}y \leq x \leq 2^{o(n)}y$, i.e., x and y are equal up to subexponential factors in n. We let $x \geq_n y$ denote $x \geq 2^{-o(n)}y$, and $x \leq_n y$ denote $x \leq 2^{o(n)}y$.

Whenever we write $|0\rangle$, we mean the all-zero computational basis state over the number of qubits implied by the context, not necessarily a single-qubit state. When considering unitaries U on quantum states in some Hilbert space H, we will often only be interested in its application on a subset $X \subseteq H$ of states, and we will sometimes (with slight abuse of notation) write "$U\colon |\psi\rangle \mapsto |\perp\rangle$ if $|\psi\rangle \in H \setminus X$", where $|\perp\rangle$ is a substitute for a quantum state that is irrelevant for our analysis.

6.2.1 Finding all elements from a set with unknown size

A natural computational task, and one we will encounter later, is the task of finding all elements of a finite set X, given an algorithm Samp that samples (say) uniformly from X. By the coupon collector's argument, $|X|^{1+o(n)}$ calls to Samp suffice to succeed with overwhelming probability. While this might seem to require knowing $|X|$ to decide when to stop, it actually suffices to keep sampling until no new element has been observed for some time — provided a suitable (possibly loose) upper bound on $|X|$ is known.

Lemma 6.2.1 (Folklore)**.** *Let X be a nonempty finite set and let Samp be a (classical, resp. quantum) algorithm that samples from X in time S. Suppose there exists $0 < \varepsilon \leq 1$ such that, for all $x \in X$, $\Pr[\mathsf{Samp}\text{ outputs } x] \geq \frac{\varepsilon}{|X|}$ where the probability is over the internal randomness of Samp. For all $\delta \in (0,1)$, there exists a (classical, resp. quantum) algorithm that, given a positive integer $t \geq \frac{1}{\varepsilon}\ln(|X|/\delta)$, outputs all elements of X in time $O(t|X|\ln(|X|)\mathsf{S})$ using $O(t|X|\ln(|X|))$ applications of Samp, except with probability δ. The algorithm uses $O(|X|)$ classical bits beyond the memory used by Samp. (The algorithm does not require prior knowledge of $|X|$.)*

Several arguments for this lemma are possible; the one presented below is inspired by [Iir20].

Proof. Let $\mathcal{A}(\mathsf{Samp}, t)$ be the algorithm that repeatedly calls Samp and maintains the set $S \subseteq X$ of distinct elements seen so far, and that outputs S once no new element has been observed in the last $(|S|+1)t$ calls. (This stopping rule is inspired by [Iir20].) The runtime of $\mathcal{A}(\mathsf{Samp}, t)$ is determined by the number of calls to Samp, and its additional memory usage beyond that of Samp is $O(|X|)$.

We claim that, except with probability δ, the following holds for all $s \in \{0, \dots, |X| - 1\}$: immediately after the set S constructed during $\mathcal{A}(\mathsf{Samp}, t)$ is of size s, a new element of X is found within the next $\lceil |X|t/(|X| - s) \rceil$ calls to Samp. Since $\lceil |X|t/(|X| - s) \rceil \leq (s + 1)t$, this implies that the stopping rule is satisfied if and only if all elements of X have been found, meaning that $\mathcal{A}(\mathsf{Samp}, t)$ outputs all elements of X. Moreover, it implies that the total number of calls to Samp is upper bounded by

$$\sum_{s=0}^{|X|-1} \left\lceil \frac{|X|t}{|X| - s} \right\rceil + (|X| + 1)t \leq |X|t \sum_{j=1}^{|X|} \frac{1}{j} + O(|X|t) = O(t|X| \ln(|X|))$$

since $\sum_{j=1}^{|X|} \frac{1}{j} \leq \ln(|X|) + 1$.

It remains to prove the claim. Fix $s \in \{0, \dots, |X| - 1\}$ and consider the call of Samp (during $\mathcal{A}(\mathsf{Samp}, t)$) that returns the s-th new element of X. By assumption on ε, the probability that a single run of Samp returns a new element of X is at least $(|X| - s)\varepsilon/|X|$, so the probability that the next $\lceil |X|t/(|X| - s) \rceil$ (independent) calls to Samp yield no new element is at most $(1 - (|X| - s)\varepsilon/|X|)^{\lceil |X|t/(|X|-s) \rceil} \leq e^{-\varepsilon t}$. The claim now follows from a union bound over all $s \in \{0, \dots, |X| - 1\}$, because $|X|e^{-\varepsilon t} \leq \delta$ by assumption on t. □

6.2.2 Relations and associated data structures

Our algorithms make use of data structures storing relations.

Definition 6.2.2. A *relation* R on $X \times Y$ is a subset $R \subseteq X \times Y$, equivalently viewed as a function $R \colon X \times Y \to \{0, 1\}$. For every $x \in X$, we let $R(x) = \{y \in Y : (x, y) \in R\}$. We let $R^\top \subseteq Y \times X$ be the relation defined by $(y, x) \in R^\top$ if and only if $(x, y) \in R$. A pair $x, x' \in X$ such that $R(x) \cap R(x') \neq \emptyset$ is called an *R-collision*.

Note that the latter is a natural generalization (from functions to relations) of the concept of a collision.

The standard way of accessing a function $f\colon X \to Y$ is through queries, meaning an algorithm is given a description of f that allows the ability to efficiently compute, for any $x \in X$, the value $f(x)$. If f is a 1-to-1 function, then a more powerful type of access allows the efficient computation of $f^{-1}(y)$ for any $y \in Y$. This is not always possible from a simple description of f, but is, for example, possible if all function values of f, $(x, f(x))$, are stored in a data structure, perhaps constructed during some preprocessing step.

For relations, we can distinguish three types of (quantum) access. First, the simplest type, *query access*, means that for any $(x, y) \in X \times Y$, it is possible to efficiently check if $(x, y) \in R$. A second type that is more analogous to evaluation of a function f is *forward superposition query access* to R, meaning we can query an oracle $\mathcal{O}_R$ that acts, for all $x \in X$, as:

$$|x\rangle|0\rangle \mapsto \begin{cases} |x\rangle \sum_{y\in R(x)} \frac{1}{\sqrt{|R(x)|}} |y\rangle & \text{if } R(x) \neq \emptyset \\ |x\rangle|\perp\rangle & \text{otherwise.} \end{cases}$$

The third type of access to R, analogous to having standard and inverse query access to f, is to have query access to both $\mathcal{O}_R$ and $\mathcal{O}_{R^\top}$. That is, one can not only implement a forward superposition query with respect to R, but also with respect to $R^\top$:

$$|y\rangle|0\rangle \mapsto \begin{cases} |y\rangle \sum_{x\in R^\top(y)} \frac{1}{\sqrt{|R^\top(y)|}} |x\rangle & \text{if } R^\top(y) \neq \emptyset \\ |y\rangle|\perp\rangle & \text{otherwise.} \end{cases}$$

We will always implement this third type of access by working with a data structure $D(R)$ that stores R in QCRAM.

Data structures for relations. Specifically, we will store a relation $R \subseteq X \times Y$ in a classical data structure, denoted $D(R)$, in a way such that the following operations can be performed using $O(\log|X| + \log|Y|)$ time and classical memory:

- **Insert:** For any $(x, y) \in (X \times Y) \setminus R$, add (x, y) to $D(R)$.
- **Lookup by x:** For any $x \in X$, return a pointer to an array containing all y such that $(x, y) \in R$, and its size $|R(x)|$.
- **Lookup by y:** For any $y \in Y$, return a pointer to an array containing all x such that $(x, y) \in R$, and its size $|R^\top(y)|$.

When constructing $D(R)$, we store the elements $(x, y) \in R$ in two different ways: once in a keyed data structure with x as the "key" and y as an associated "value", and once in another keyed data structure, with y as the key, and x as the value.

By storing $D(R)$ in QCRAM, we can use the ability to access the QCRAM in superposition to perform lookups in superposition. We then call $D(R)$ a *QCRAM data structure*.

Lemma 6.2.3. *Let $D(R)$ be a QCRAM data structure for a relation $R \subseteq X \times Y$. Then the following operations can be performed using $O(\log|X| + \log|Y|)$ time and QCRAM queries:*

- **Insert:** *For any $(x, y) \in (X \times Y) \setminus R$, add (x, y) to $D(R)$.*
- **Lookup by x in superposition:** *For any $x \in X$, map*

$$|x\rangle \mapsto \begin{cases} |x\rangle \sum_{i=1}^{|R(x)|} \frac{1}{\sqrt{|R(x)|}} |i\rangle |y_i\rangle & \text{if } R(x) \neq \emptyset \\ |x\rangle |\perp\rangle & \text{otherwise} \end{cases}$$

  where $\{y_1, \ldots, y_{|R(x)|}\} = R(x)$.
- **Lookup by y in superposition:** *For any $y \in Y$, map*

$$|y\rangle \mapsto \begin{cases} |y\rangle \sum_{i=1}^{|R^\top(y)|} \frac{1}{\sqrt{|R^\top(y)|}} |i\rangle |x_i\rangle & \text{if } R^\top(y) \neq \emptyset \\ |y\rangle |\perp\rangle & \text{otherwise} \end{cases}$$

  where $\{x_1, \ldots, x_{|R^\top(y)|}\} = R^\top(y)$.

In later sections, we will abuse notation by letting $|x, y\rangle$ denote $|x, i, y\rangle$, where i is the index of y in the array storing $R(x)$, and sometimes we will even let $|x, y, x'\rangle$ denote $|x, i, y, j, x'\rangle$ where i is as above, and j is the index of x' in the array storing $R^\top(y)$. This is not a problem, as all we require from the specific encoding of $|x, y\rangle$ is that we can do computations on both x and y — it does not matter if there is superfluous information in the encoding, as long as it is not too large.

Proof. The insertion is directly inherited from the data structure. We describe a superposition lookup of x (the case for y is virtually identical). The classical lookup by x returns a number $N = |R(x)|$, and a pointer to an array storing $R(x) = \{y_1, \ldots, y_N\}$. If $N \neq 0$, then generate $\sum_{i=1}^{N} \frac{1}{\sqrt{N}} |i\rangle |0\rangle$. Use a QCRAM query to access the entries of the array, to map $|i\rangle |0\rangle \mapsto |i\rangle |y_i\rangle$. The time and memory complexities follow immediately from the properties of $D(R)$. □

6.2.3 Geometric properties of the unit sphere

This section presents relevant background on the n-dimensional unit sphere $\mathcal{S}^{n-1}$, as unit vectors are the main mathematical objects we will work with.

Given a unit vector $\mathbf{x} \in \mathcal{S}^{n-1}$ and an angle $\alpha \in [0, \pi/2]$, we define the subset $\mathcal{C}_{\mathbf{x},\alpha} := \{\mathbf{c} \in \mathcal{S}^{n-1} : \langle \mathbf{x}, \mathbf{c} \rangle \geq \cos(\alpha)\}$ of the unit sphere, called the *spherical cap* of center $\mathbf{x}$ and angle α. Note that $\mathcal{C}_{\mathbf{x},0} = \{\mathbf{x}\}$ and that $\mathcal{C}_{\mathbf{x},\pi/2}$ is a hemisphere. The intersection of two spherical caps forms a *spherical wedge*, and is denoted by $\mathcal{W}_{\mathbf{x},\alpha,\mathbf{y},\beta} := \mathcal{C}_{\mathbf{x},\alpha} \cap \mathcal{C}_{\mathbf{y},\beta}$ for $\mathbf{x}, \mathbf{y} \in \mathcal{S}^{n-1}$ and $\alpha, \beta \in [0, \pi/2]$. If $\langle \mathbf{x}, \mathbf{y} \rangle = \cos(\theta)$ for some $\theta \in [0, \pi/2]$, we obtain

$$\mathcal{W}_{\mathbf{x},\alpha,\mathbf{y},\beta} = \begin{cases} \mathcal{C}_{\mathbf{x},\alpha} & \text{if } \theta \leq \beta - \alpha \\ \mathcal{C}_{\mathbf{y},\beta} & \text{if } \theta \leq \alpha - \beta \\ \emptyset & \text{if } \theta > \alpha + \beta \end{cases} \tag{6.1}$$

and in the remaining case ($|\alpha - \beta| < \theta \leq \alpha + \beta$) the wedge is nonempty and properly contained in both $\mathcal{C}_{\mathbf{x},\alpha}$ and $\mathcal{C}_{\mathbf{y},\beta}$.

The volume (or hyperarea) of a spherical cap or wedge allows us to quantify the probability that a random unit vector is close to one or two given vectors, respectively. Given $\mathbf{x}, \mathbf{y} \in \mathcal{S}^{n-1}$ with $\langle \mathbf{x}, \mathbf{y} \rangle = \cos(\theta)$, we denote the ratio of the volume of $\mathcal{W}_{\mathbf{x},\alpha,\mathbf{y},\beta}$ to that of $\mathcal{S}^{n-1}$ by

$$\mathcal{W}(\alpha, \beta \mid \theta) := \Pr_{\mathbf{c} \sim U(\mathcal{S}^{n-1})} \left[\langle \mathbf{x}, \mathbf{c} \rangle \geq \cos(\alpha) \text{ and } \langle \mathbf{y}, \mathbf{c} \rangle \geq \cos(\beta) \right].$$

This normalized volume depends on $\mathbf{x}, \mathbf{y}$ only through their mutual angle θ.

The normalized volume of a spherical cap is thus of the form $\mathcal{W}(\alpha, \alpha \mid 0)$, and can be estimated using the following lemma from [BDGL16].

Lemma 6.2.4 (Spherical cap volume [BDGL16, Lemma 2.1])**.** *For all $\alpha \in (0, \pi/2)$,*

$$\mathcal{W}(\alpha, \alpha \mid 0) = n^{\Theta(1)} \underbrace{\left(1 - \cos^2(\alpha)\right)^{n/2}}_{=: p_\alpha}.$$

In [BDGL16], an estimate for the normalized volume of a spherical wedge was also given, but it was missing the condition $|\alpha - \beta| \leq \theta \leq \alpha + \beta$, which is necessary for their formula to hold. The other cases follow directly from Equation (6.1) and Lemma 6.2.4: $\mathcal{W}(\alpha, \beta \mid \theta) = 0$ if $\theta > \alpha + \beta$, and $\mathcal{W}(\alpha, \beta \mid \theta) =_n \min\{p_\alpha, p_\beta\}$

if $\theta \geq |\alpha - \beta|$. (When $\theta = |\alpha - \beta|$, Lemma 6.2.4 and Lemma 6.2.5 coincide.) More detailed formulas for the wedge volume can be found in [LK14].

Lemma 6.2.5 (Spherical wedge volume (correction of [BDGL16, Lemma 2.2]))**.** *For all $\alpha, \beta, \theta \in (0, \pi/2)$ with $|\alpha - \beta| \leq \theta \leq \alpha + \beta$,*

$$\mathcal{W}(\alpha, \beta \mid \theta) = n^{\Theta(1)} \left(1 - \frac{\cos^2(\alpha) + \cos^2(\beta) - 2\cos(\alpha)\cos(\beta)\cos(\theta)}{\sin^2(\theta)}\right)^{n/2}.$$

In particular, for all $\alpha, \theta \in (0, \pi/2)$ with $\theta \leq 2\alpha$,

$$\mathcal{W}(\alpha, \alpha \mid \theta) = n^{\Theta(1)} \left(1 - \frac{2\cos^2(\alpha)}{1 + \cos(\theta)}\right)^{n/2}.$$

The previous two lemmas allow us to estimate the probability that a random unit vector $\mathbf{c}$ satisfies both $\langle \mathbf{x}, \mathbf{c} \rangle \geq \cos(\alpha)$ and $\langle \mathbf{y}, \mathbf{c} \rangle \geq \cos(\beta)$, for fixed $\mathbf{x}, \mathbf{y} \in \mathcal{S}^{n-1}$. In our analysis, we instead deal with events where those inner products are equal to $\cos(\alpha)$ and $\cos(\beta)$, respectively, up to an additive $\epsilon > 0$. Fortunately, the following (folklore) lemmas follow from Lemma 6.2.4 and Lemma 6.2.5, provided that ϵ is chosen sufficiently small; see Section 6.A for the proofs. Throughout this chapter, we therefore globally fix $\epsilon = 1/(\log n)^2$.

Lemma 6.2.6 (Spherical cap volume, approximate version)**.** *Let $\alpha \in (0, \pi/2)$ satisfy $\alpha = \Omega(1)$. For all $\mathbf{x} \in \mathcal{S}^{n-1}$,*

$$\Pr_{\mathbf{c} \sim U(\mathcal{S}^{n-1})} [\langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)] =_n p_\alpha.$$

Lemma 6.2.7 (Spherical wedge volume, approximate version)**.** *Let $\alpha, \beta, \theta \in (0, \pi/2)$ satisfy $|\alpha - \beta| + \Omega(1) \leq \theta \leq \alpha + \beta - \Omega(1)$. For all $\mathbf{x}, \mathbf{y} \in \mathcal{S}^{n-1}$ with $\langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)$,*

$$\Pr_{\mathbf{c} \sim U(\mathcal{S}^{n-1})} [\langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha) \text{ and } \langle \mathbf{y}, \mathbf{c} \rangle \approx_\epsilon \cos(\beta)] =_n \mathcal{W}(\alpha, \beta \mid \theta).$$

The $\Omega(1)$ terms in these lemmas hide a (small) absolute constant $c > 0$ that is independent of n. They ensure that the non-approximate lemmas remain applicable as $n \to \infty$, which we use in our proofs. This chapter mainly deals with normalized wedge volumes of the form $\mathcal{W}(\alpha, \alpha \mid \theta)$ or $\mathcal{W}(\theta, \alpha \mid \alpha)$, in which case the condition in Lemma 6.2.7 simplifies to $\min\{\theta, 2\alpha - \theta\} = \Omega(1)$.

6.2.4 Random product codes in $\mathbb{R}^n$ and induced relations

The main tasks in our quantum algorithm can be formulated as searching for pairs $(\mathbf{x}, \mathbf{y})$ of unit vectors that are somewhat "close" to each other, in the sense that we can bound their inner product. We simplify these tasks by only searching for pairs that form an R-collision under carefully constructed relations of the form $R = R_{(C,\alpha)}$, where

$$R_{(C,\alpha)} := \{(\mathbf{x}, \mathbf{c}) \in \mathcal{S}^{n-1} \times C : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)\}$$

for a subset $C \subseteq \mathcal{S}^{n-1}$ and $\alpha \in (0, \pi/2)$. Consequently, if $(\mathbf{x}, \mathbf{y})$ form an $R_{(C,\alpha)}$-collision (i.e., $R_{(C,\alpha)}(\mathbf{x}) \cap R_{(C,\alpha)}(\mathbf{y}) \neq \emptyset$), then there is a point $\mathbf{c} \in C$ that is close to both $\mathbf{x}$ and $\mathbf{y}$, implying that $\mathbf{x}$ and $\mathbf{y}$ are also close to each other, where closeness is quantified by the parameter α. When the dependencies on C and α are clear from context, we often just write R. Note that the relation $R_{(C,\alpha)}$ is an infinite object, but we obtain a finite relation by restricting it to a finite subset in the first coordinate. We will often consider $R_L := R|_{L \times C}$ for a finite set $L \subseteq \mathcal{S}^{n-1}$.

The family of subsets C that we will work with are based on random product codes [BDGL16].

Definition 6.2.8 (Random product code in $\mathbb{R}^n$)**.** We define $\mathcal{R}(n, b, M)$ as the family of sets $C \subseteq \mathcal{S}^{n-1}$ that can be written as

$$C = \mathbf{Q}(C^{(1)} \times \cdots \times C^{(b)})$$

where $\mathbf{Q} \in \mathrm{SO}(n)$, and $C^{(i)} \subseteq \frac{1}{\sqrt{b}}\mathcal{S}^{n/b-1}$ with $|C^{(i)}| = M^{1/b}$ for each $i \in \{1, \ldots, b\}$. Such a tuple $(\mathbf{Q}, C^{(1)}, \ldots, C^{(b)})$ is called a *description* of C.

A *random product code (RPC)* in $\mathbb{R}^n$ is a random set $C \in \mathcal{R}(n, b, M)$ obtained by sampling a uniformly random description $(\mathbf{Q}, C^{(1)}, \ldots, C^{(b)})$ from the set of all valid descriptions. We write $\mathrm{RPC}(n, b, M)$ for the resulting distribution over $\mathcal{R}(n, b, M)$.

Random product codes C have two very useful properties. For a parameter α and induced relation $R = R_{(C,\alpha)}$, we have:

(1) **Efficient decodability:** In certain parameter regimes (in particular, if b is not too small), there is an algorithm that, on input $\mathbf{x} \in \mathcal{S}^{n-1}$, computes the set $R(\mathbf{x})$ in time roughly equal to its size $|R(\mathbf{x})|$. See Lemma 6.2.9. Note that this algorithm gives forward superposition query access to R (see Section 6.2.2).

(2) **Random behavior:** In certain parameter regimes (in particular, if b is not too large), a random product code C behaves like a uniformly random subset of $\mathcal{S}^{n-1}$ in the following sense: for all $\mathbf{x}, \mathbf{y} \in \mathcal{S}^{n-1}$ satisfying $\langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)$, the probability that there exists $\mathbf{c} \in C$ satisfying $\langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)$ and $\langle \mathbf{y}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)$ (meaning that $(\mathbf{x}, \mathbf{y})$ is an R-collision) is the same, up to subexponential factors, as in the case that each element of C was independently sampled from $U(\mathcal{S}^{n-1})$. See Lemma 6.2.10.

In other words, RPCs give us sufficiently good random behavior, while still allowing for efficient decodability, which is the main reason we work with RPCs instead of uniformly random subsets.

The next two lemmas[4] show that it suffices to take $b = \log n$ for both properties to be satisfied, so we fix this choice of b in the remainder of this chapter.

Lemma 6.2.9 (Efficient decoder for $R_{(C,\alpha)}(\mathbf{x})$ (implicit in [BDGL16, Lemma 5.1]))**.** *Let $b = \omega(1)$ and $M = 2^{O(n)}$. There exists a classical algorithm that, given a description $(\mathbf{Q}, C^{(1)}, \dots, C^{(b)})$ of $C \in \mathcal{R}(n, b, M)$ and a target vector $\mathbf{x} \in \mathcal{S}^{n-1}$, returns the set $R_{(C,\alpha)}(\mathbf{x}) := \{\mathbf{c} \in C : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)\}$ in time $(|R_{(C,\alpha)}(\mathbf{x})| + 1)2^{o(n)}$.*

Next, Lemma 6.2.10 provides sufficiently tight bounds on the probability that two unit vectors form a collision under the relation $R_{(C,\alpha)}$ induced by a sample $C \sim \mathrm{RPC}(n, b, M)$. Specifically, it identifies parameter regimes where RPCs behave similarly to uniformly random subsets with respect to collision probabilities, as $\Pr_{C \sim U(\mathcal{S}^{n-1}, M)}[R_{(C,\alpha)}(\mathbf{x}) \cap R_{(C,\beta)}(\mathbf{y}) \neq \emptyset] = \Theta(\min\{1, M\mathcal{W}(\alpha, \beta \mid \theta)\})$.

Lemma 6.2.10 (Random behavior [BDGL16, Theorem 5.1])**.** *Let $\alpha, \beta \in (0, \pi/2)$ and $\theta \in [0, \pi/2)$ satisfy $|\alpha - \beta| + \Omega(1) \leq \theta \leq \alpha + \beta - \Omega(1)$ if $\theta \neq 0$, and $\alpha = \beta = \Omega(1)$ otherwise. Let $b = O(\log n)$ and let M be such that $M \cdot \mathcal{W}(\alpha, \beta \mid \theta) = 2^{-O(n)}$. For all $\mathbf{x}, \mathbf{y} \in \mathcal{S}^{n-1}$ with $\langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)$,*

$$\Pr_{C \sim \mathrm{RPC}(n,b,M)}\left[R_{(C,\alpha)}(\mathbf{x}) \cap R_{(C,\beta)}(\mathbf{y}) \neq \emptyset\right] =_n \min\{1, M \cdot \mathcal{W}(\alpha, \beta \mid \theta)\}.$$

Note that the case $\theta = 0$ deals with the probability that $R_{(C,\alpha)}(\mathbf{x})$ is nonempty (meaning there exists $\mathbf{c} \in C$ that is "close" to $\mathbf{x} = \mathbf{y}$).

[4]The proofs in [BDGL16] consider $R_{(C,\alpha)}(\mathbf{x})$ defined as $\{\mathbf{c} \in C : \langle \mathbf{x}, \mathbf{c} \rangle \geq \cos(\alpha)\}$, but can easily be seen to work for our definition (with $\approx_\epsilon$ instead of $\geq$).

6.3 Main quantum algorithm

In this section, we present a quantum algorithm for Problem 6.1.1 from the introduction: given a list L of m i.i.d. uniform samples from $\mathcal{S}^{n-1}$, this algorithm returns m triples $(\mathbf{x},\mathbf{y},\mathbf{z}) \in L^3$ satisfying $\|\mathbf{x}-\mathbf{y}-\mathbf{z}\| \leq 1$. We are specifically interested in instances with $m = (27/16)^{n/4+o(n)}$. For sufficiently large $o(n)$, this is the minimal list size to ensure that with high probability over the choice of L there exist m 3-tuple solutions [HK17, Theorem 3], and hence corresponds to the minimal memory regime of Problem 6.1.1. Our work is motivated by the observation that, for a list size m that is slightly larger, but still asymptotically $m = (27/16)^{n/4+o(n)}$, there exist in fact m 3-tuple solutions $(\mathbf{x},\mathbf{y},\mathbf{z}) \in L^3$ for which $\langle \mathbf{x},\mathbf{y}\rangle$ is essentially $1/3$ and $\langle \mathbf{x}-\mathbf{y},\mathbf{z}\rangle$ is essentially $2/3$. This allows us to reduce our search problem to (a less simplified version of) Problem 6.1.2 from the introduction. More precisely, as proven in Lemma 6.3.17, there exist $\theta,\theta' \in (0,\pi/2)$ such that

$$\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta') := \left\{(\mathbf{x},\mathbf{y},\mathbf{z}) \in L^3 : \langle \mathbf{x},\mathbf{y}\rangle \approx_\epsilon \cos(\theta), \langle \tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|},\mathbf{z}\rangle \approx_\epsilon \cos(\theta')\right\} \tag{6.2}$$

consists only of 3-tuple solutions and (with high probability over the choice of L) has size at least m. We therefore design a quantum algorithm that finds m elements of $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$, thereby solving Problem 6.1.1. As we present the algorithm for a fixed choice[5] of (θ,θ'), we usually write $\mathcal{T}_{\mathrm{sol}}$ as shorthand for $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$.

Remark 6.3.1. There is a slight subtlety in the definition of $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$: it does not explicitly require each tuple $(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ to consist of distinct vectors, which is required by Problem 6.1.1. (Note that the latter problem is trivial otherwise: for instance, consider the $|L|$ 3-tuples $(\mathbf{x},\mathbf{x},\mathbf{x})$ for $\mathbf{x} \in L$.) While we could explicitly impose that $\mathbf{x},\mathbf{y},\mathbf{z}$ be distinct in Equation (6.2), this property is already implied by Equation (6.2) in our setting of interest. Namely, if $\theta = \Omega(1)$ and $\cos(\theta') = \epsilon + \sqrt{(1-\cos(\theta)+\epsilon)/2}$ (as motivated by Lemma 6.3.17), then each $(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ that is not composed of distinct vectors must have $\langle \mathbf{x},\mathbf{y}\rangle = \cos(\theta)-\epsilon$. As the latter event has measure zero for $\mathbf{y} \sim U(\mathcal{S}^{n-1})$, we may, without loss of generality, assume that in the setting of interest each element of $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ consists of distinct vectors.

[5] For Problem 6.1.1, choosing $\cos(\theta) = 1/3$ and $\cos(\theta') = \epsilon + \sqrt{1/3+\epsilon/2}$ is optimal in the sense that it yields the smallest possible list size $|L|$ for which $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ consists of $|L|$ 3-tuple solutions. However, our quantum algorithm may also be of interest in settings where other choices are more suitable, such as the generalization of Problem 6.1.1 that searches for triples satisfying $\|\mathbf{x}-\mathbf{y}-\mathbf{z}\| \leq t$ for $t \neq 1$ (e.g., see [HK17]). We therefore present most results for a larger range of $\theta,\theta' \in (0,\pi/2)$.

6.3.1 High-level overview of our quantum algorithm

Our quantum algorithm consists of several steps of amplitude amplification (AA, Lemma 5.2.2), carefully nested together, and preceded by preprocessing the list L into a useful data structure. We describe the high-level ideas here, and defer the details to the following subsections. As outlined in the introduction, a natural strategy for finding an element of $\mathcal{T}_{\mathrm{sol}}$ is to search for a pair $(\mathbf{x}, \mathbf{y}) \in L^2$ satisfying $\langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)$, and then attempt to find $\mathbf{z} \in L$ such that $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}$. This approach can be viewed as a two-oracle search (recall Section 5.4.1) with sets $\mathcal{T}_2 \subseteq \mathcal{T}_1 \subseteq \mathcal{T}_0$ defined by

$$\begin{aligned}
\mathcal{T}_0 &:= L^2, \\
\mathcal{T}_1 &:= \left\{(\mathbf{x}, \mathbf{y}) \in \mathcal{T}_0 \colon \langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)\right\}, \\
\mathcal{T}_2 &:= \left\{(\mathbf{x}, \mathbf{y}) \in \mathcal{T}_1 \colon \exists \mathbf{z} \in L, \langle \tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z} \rangle \approx_\epsilon \cos(\theta')\right\}.
\end{aligned}$$

Whereas checking membership in $\mathcal{T}_1$ has negligible time complexity $\mathsf{C}_1 = 2^{o(n)}$, checking membership in $\mathcal{T}_2$ is a more involved search problem with nontrivial time complexity C_2. For instance, using amplitude amplification we obtain $\mathsf{C}_2 = \sqrt{|L|}\, 2^{o(n)}$ when $|L| = 2^{O(n)}$. Assuming that, given $(\mathbf{x}, \mathbf{y}) \in \mathcal{T}_2$, we can also *find* $\mathbf{z}$ such that $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}$ in time C_2 (which is true for amplitude amplification), this results in a quantum algorithm for finding an element of $\mathcal{T}_{\mathrm{sol}}$ in time

$$\sqrt{\frac{\varepsilon_1}{\varepsilon_2}} \left(\frac{1}{\sqrt{\varepsilon_1}} \mathsf{C}_1 + \mathsf{C}_2 \right) 2^{o(n)}$$

where ε_1 and ε_2 are the probabilities that an element sampled uniformly at random from $\mathcal{T}_0$ lies in $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Repeating the algorithm about $|L|$ times then hopefully solves Problem 6.1.1. Unfortunately, this naive strategy is too expensive (recall Footnote 3).

Inspired by the locality-sensitive filtering technique, which has been used in state-of-the-art lattice sieving algorithms since [BDGL16], our key idea to improve this naive strategy is to search more *locally*. The sets $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2$ will be replaced by random subsets of them that only include those pairs of vectors that lie in the same "local" region of $\mathcal{S}^{n-1}$ (formally defined as forming a collision under some suitable relation), meaning that these vectors are not too far apart. While those subsets may not cover all of $\mathcal{T}_{\mathrm{sol}}$, their randomness will ensure that repeating this approach sufficiently many times for different random subsets allows us to find all elements of $\mathcal{T}_{\mathrm{sol}}$. Altogether, this modified strategy results in a better trade-off

between the different cost components of two-oracle search.

Specifically, we restrict $\mathcal{T}_0 = L^2$ to those pairs of vectors in L that are both "close" to some vector in a fixed subset $C \subseteq \mathcal{S}^{n-1}$, where closeness is measured by a parameter α. Letting R be the relation on $\mathcal{S}^{n-1} \times C$ including exactly those pairs $(\mathbf{x}, \mathbf{c})$ that satisfy $\langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)$, we define

$$\mathcal{T}_0(R) := \left\{ (\mathbf{x}, \mathbf{y}) \in \mathcal{T}_0 \colon R(\mathbf{x}) \cap R(\mathbf{y}) \neq \emptyset \right\}$$

as the set of R-collisions in $\mathcal{T}_0 = L^2$. For each $(\mathbf{x}, \mathbf{y}) \in \mathcal{T}_0(R)$, we are now guaranteed that both $\mathbf{x}$ and $\mathbf{y}$ are close to some $\mathbf{c} \in C$, so they are also somewhat close to each other, and have a higher chance of satisfying $\langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)$. We will therefore replace the role of $\mathcal{T}_1$ by a suitable subset

$$\mathcal{T}_1(R) \subseteq \left\{ (\mathbf{x}, \mathbf{y}) \in \mathcal{T}_0(R) \colon \langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta) \right\}$$

consisting of R-collisions in $\mathcal{T}_1$. The probability ε_1' that an element sampled from $\mathcal{T}_0(R)$ is in $\mathcal{T}_1(R)$ could now be larger than ε_1, while the cost C_1' of checking membership in $\mathcal{T}_1(R)$ remains negligible, so we seem to have reduced one component of the two-oracle search cost.

However, there is a caveat: in order to project onto $\mathcal{T}_1(R)$ using amplitude amplification, we need a unitary that creates a superposition over this restricted subset $\mathcal{T}_0(R)$ of L^2, and it is not immediately clear that finding R-collisions in L^2 is easy. This will be resolved by adding a *preprocessing* phase during which the algorithm prepares a data structure in QCRAM that stores the finite relation $R_L := R|_{L \times C}$, allowing us to efficiently construct a superposition over the elements of $\mathcal{T}_0(R)$ at any later stage of the algorithm. By taking C to be a random product code (RPC, Definition 6.2.8), we can prepare such a data structure at reasonable cost.

Next, given a superposition over $\mathcal{T}_1(R)$, the goal is to check whether there exists $\mathbf{z} \in L$ satisfying $\langle \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z} \rangle \approx_\epsilon \cos(\theta')$ for a given pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{T}_1(R)$. Again, we will restrict our search to achieve this more efficiently than performing amplitude amplification over L: given $(\mathbf{x}, \mathbf{y}) \in \mathcal{T}_1(R)$, we only consider those $\mathbf{z} \in L$ that collide with the normalization of $\mathbf{x} - \mathbf{y}$ under the relation $R' := \{(\mathbf{x}, \mathbf{c}) \in \mathcal{S}^{n-1} \times C' \colon \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha')\}$ defined by another subset $C' \subseteq \mathcal{S}^{n-1}$ and parameter α'. That is, the search is restricted to a subset

$$\mathcal{T}_2(R, R') \subseteq \left\{ (\mathbf{x}, \mathbf{y}) \in \mathcal{T}_1(R) \colon \exists \mathbf{z} \in L, \langle \tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z} \rangle \approx_\epsilon \cos(\theta'), \right. \\ \left. R'(\tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}) \cap R'(\mathbf{z}) \neq \emptyset \right\}.$$

This "local" search is again facilitated by letting C' be a random product code and by preparing a data structure for $R'_L := R'|_{L\times C'}$. If restricting to R'-collisions reduces the search for $\mathbf{z}$ to a much smaller subset of L, then the cost C'_2 of checking membership in $\mathcal{T}_2(R,R')$ could be significantly less than the cost C_2 in the naive approach, possibly resulting in an improved overall time complexity.

Indeed, this *local* two-oracle search algorithm finds elements of $\mathcal{T}_2(R,R') \subseteq \mathcal{T}_2$ more efficiently than the naive strategy. By sampling the subsets C, C' randomly, using the RPC distribution from Definition 6.2.8, we can ensure that this approach finds a sufficiently random subset of $\mathcal{T}_{\text{sol}}$, so repeating the local two-oracle search algorithm for sufficiently many random pairs (C,C') allows us to find $|L|$ elements of $\mathcal{T}_{\text{sol}}$, thereby solving Problem 6.1.1.

We summarize the resulting quantum algorithm, called `3List`, in Algorithm 6.1.[6] In the following sections, we explain and analyze the individual phases of `3List` in more detail, allowing us to then prove our main result, Theorem 6.3.2, in Section 6.3.6. We remark that we cannot use the result of [KLL15] or [Amb10] directly: as our goal is to find all elements of $\mathcal{T}_2(R,R')$, we want to sample a single element with sufficient randomness so that many samples are likely to correspond to many distinct elements. We therefore give our own algorithm and analysis, for our particular setting.

Theorem 6.3.2. *There exists a quantum algorithm that, with probability* $1 - 2^{-\Omega(n)}$*, solves Problem 6.1.1 with list size* $m = (27/16)^{n/4+o(n)}$ *in time* $2^{0.284551n+o(n)}$*. This algorithm uses* $m2^{o(n)}$ *classical memory and QCRAM bits, and* $2^{o(n)}$ *qubits.*

As explained in Section 6.4, Theorem 6.3.2 implies the existence of a quantum algorithm that *heuristically* solves the Shortest Vector Problem with the stated time and memory complexity.

[6] The algorithm `SolutionSearch` in the **Search** phase is essentially the aforementioned local two-oracle search algorithm, and samples from $\mathcal{T}_2(R,R')$ for fixed (R,R'). By choosing the parameter t carefully, the **Search** phase finds all elements of $\mathcal{T}_2(R,R')$ using $|\mathcal{T}_2(R,R')|2^{o(n)}$ calls to `SolutionSearch`, even if $|\mathcal{T}_2(R,R')|$ is not known a priori.

Algorithm 6.1: `3List`

Input: List $L \subseteq \mathcal{S}^{n-1}$
Angles $\theta, \theta' \in (0, \pi/2)$

Parameters: Angles $\alpha, \alpha' \in (0, \pi/2)$
Number of repetitions ℓ
Search threshold t

Output: List L' consisting of 3-tuple solutions in $\mathcal{T}_{\text{sol}}(L, \theta, \theta')$

1. $L' \leftarrow \emptyset$
2. Repeat ℓ times:
   (a) **Sample:** Sample
   $$C \sim \text{RPC}(n, \log n, 1/p_\alpha) \quad \text{and} \quad C' \sim \text{RPC}(n, \log n, 1/p_{\alpha'})$$
   (b) **Preprocess:** $(D, D') \leftarrow \texttt{Preprocess}(L, C, C')$ (Algorithm 6.2)
   (c) **Search:** Use (D, D') to find 3-tuple solutions:
      i. $S \leftarrow \emptyset$, $\textsc{count} \leftarrow 0$
      ii. While $\textsc{count} < (|S| + 1)t$:
         A. $\mathbf{t} \leftarrow \texttt{SolutionSearch}(D, D')$ (Algorithm 6.5)
         B. If $\mathbf{t} \in L^3 \setminus S$, then add $\mathbf{t}$ to S and set $\textsc{count} \leftarrow 0$
         C. Else, set $\textsc{count} \leftarrow \textsc{count} + 1$
      iii. Add S to L'
3. Return L'

6.3.2 Sampling phase

The **Sampling** phase of Algorithm 6.1 samples two RPCs (Definition 6.2.8) C and C', and stores their descriptions. This phase can be completed using $2^{o(n)}$ time and classical memory. The sizes of C and C' depend on the parameters $\alpha, \alpha' \in (0, \pi/2)$, which affect the complexity of `3List` and will be chosen to optimize performance. The sampled RPCs C, C' and angles α, α' induce the relations

$$R := \{(\mathbf{x}, \mathbf{c}) \in \mathcal{S}^{n-1} \times C : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)\}$$

and

$$R' := \{(\mathbf{x}, \mathbf{c}) \in \mathcal{S}^{n-1} \times C' : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha')\}.$$

Remark 6.3.3 (On the RPC parameter setting). The parameter setting of the distribution of $C \sim \mathrm{RPC}(n, \log n, 1/p_\alpha)$ ensures that, for a given $\mathbf{x} \in \mathcal{S}^{n-1}$, the set $R(\mathbf{x}) = \{\mathbf{c} \in C : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)\}$ has expected size $2^{o(n)}$ (taken over the choice of C) by Lemma 6.2.6. The set $R(\mathbf{x})$ can therefore be computed in expected time $2^{o(n)}$ by Lemma 6.2.9. The same is true for C', and those properties are utilized during the **Preprocessing** and **Search** phases of `3List`.

To simplify our analysis of `3List`, we will impose the following conditions on the relations (R, R'), which will be satisfied with overwhelming probability in our applications.

Definition 6.3.4 (Well-balanced (R, R'))**.** For $L, C, C' \subseteq \mathcal{S}^{n-1}$, let $R \subseteq \mathcal{S}^{n-1} \times C$ and $R' \subseteq \mathcal{S}^{n-1} \times C'$ be relations. We say that (R, R') is *well-balanced on* L if the following conditions hold:

(*i*) $|R_L| =_n |L|$ and $|R_L^\top(\mathbf{c})| =_n \frac{|R_L|}{|C|}$ for all $\mathbf{c} \in C$, where $R_L := R|_{L \times C}$.

(*ii*) $|R'_L| =_n |L|$ and $|(R'_L)^\top(\mathbf{c}')| =_n \frac{|R'_L|}{|C'|}$ for all $\mathbf{c}' \in C'$, where $R'_L := R'|_{L \times C'}$.

(*iii*) $|\{\mathbf{z}' \in L : (\mathbf{x}, \mathbf{y}, \mathbf{z}') \in \mathcal{T}_{\mathrm{sol}}(R, R')\}| =_n 1$ for all $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(R, R')$ (as defined in Equation (6.3)).

6.3.3 Preprocessing phase

After having sampled two subsets $C, C' \subseteq \mathcal{S}^{n-1}$, we proceed to the **Preprocessing** phase. Using Algorithm 6.2, it constructs QCRAM data structures D and D' (as defined in Section 6.2.2) for the relations induced by these subsets and the parameters α and α'.

Algorithm 6.2: `Preprocess`(L, C, C')

Input: List $L \subseteq \mathcal{S}^{n-1}$
Descriptions of $C \in \mathcal{R}(n, \log n, 1/p_\alpha)$ and $C' \in \mathcal{R}(n, \log n, 1/p_{\alpha'})$

Output: QCRAM data structures $D = D(R_L)$ and $D' = D(R'_L)$, where

$$R_L := \{(\mathbf{x}, \mathbf{c}) \in L \times C : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)\}$$
$$R'_L := \{(\mathbf{x}, \mathbf{c}) \in L \times C' : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha')\}$$

1. Initialize a pair of empty data structures D and D'
2. For each $\mathbf{x} \in L$:
   (a) Compute $R_L(\mathbf{x})$ using Lemma 6.2.9
   (b) For each $\mathbf{c} \in R_L(\mathbf{x})$:
      i. Insert $(\mathbf{x}, \mathbf{c})$ into D
   (c) Compute $R'_L(\mathbf{x})$ using Lemma 6.2.9
   (d) For each $\mathbf{c} \in R'_L(\mathbf{x})$:
      i. Insert $(\mathbf{x}, \mathbf{c})$ into D'
3. Return D and D'

By the end of this classical algorithm, D stores the relation $R_L \subseteq L \times C$, which relates each $\mathbf{x} \in L$ to all "close" vectors in C, quantified by the parameter α. Similarly, D' stores the relation $R'_L \subseteq L \times C'$, which relates each $\mathbf{x} \in L$ to all close vectors in C', this time quantified by α'.

By construction of C and C', the **Preprocessing** phase can be completed in time that is optimal up to subexponential factors.

Lemma 6.3.5 (Preprocessing cost)**.** *Given as input a list $L \subseteq \mathcal{S}^{n-1}$ and the descriptions of sets $C \in \mathcal{R}(n, \log n, 1/p_\alpha)$ and $C' \in \mathcal{R}(n, \log n, 1/p_{\alpha'})$,* `Preprocess`$(L, C, C')$ *(Algorithm 6.2) returns the data structures $D(R_L)$ and $D(R'_L)$ using*

$$(|R_L| + |R'_L| + 1)2^{o(n)}$$

time and classical memory. In particular, if (R, R') is well-balanced on L (Definition 6.3.4), it uses $|L|2^{o(n)}$ time and classical memory.

Proof. By Lemma 6.2.9, for each $\mathbf{x} \in L$, step 2(a) and step 2(c) take time $(|R_L(\mathbf{x})| + 1)2^{o(n)}$ and $(|R'_L(\mathbf{x})| + 1)2^{o(n)}$, respectively, so the cost of step 2 is $\sum_{\mathbf{x}\in L}(|R_L(\mathbf{x})| + |R'_L(\mathbf{x})| + 1)2^{o(n)} = (|R_L| + |R'_L| + 1)2^{o(n)}$. As the complexity of `Preprocess` is determined by step 2, this proves the first part of the lemma. The second part follows from conditions (i) and (ii) in Definition 6.3.4. □

While most steps in the **Search** phase of Algorithm 6.1 can be achieved using the data structures $D(R_L)$ and $D(R'_L)$ that are prepared during the **Preprocessing** phase, we actually need one more tool. Namely, we would like to be able to efficiently create a uniform superposition over

$$R'(\mathbf{x}) = \{\mathbf{c}' \in C' : \langle \mathbf{x}, \mathbf{c}' \rangle \approx_\epsilon \cos(\alpha')\}$$

for a large number of vectors $\mathbf{x} \in \mathcal{S}^{n-1}$ that the algorithm will encounter (in superposition). That is, we want forward superposition query access to R' (recall Section 6.2.2) for those vectors. These $\mathbf{x}$ are not vectors from our list L itself, but rather (normalized) differences of two vectors from L. As the number of those $\mathbf{x}$ will be rather large ($\gg m$), it is too expensive for us to store each $R'(\mathbf{x})$ in a data structure. To ensure that we can create the superposition over $R'(\mathbf{x})$ in subexponential time (i.e., at negligible cost for us), we therefore consider a "decoding" subroutine `Dec`(C') that is guaranteed to run in time $2^{o(n)}$, and achieves the desired mapping for all $\mathbf{x} \in \mathcal{S}^{n-1}$ satisfying $|R'(\mathbf{x})| \leq 2^{n/\log n}$, which suffices for our purposes.

Lemma 6.3.6. *For $\alpha' \in (0, \pi/2)$ and $C' \in \mathcal{R}(n, \log n, 1/p_{\alpha'})$, define the relation*

$$R' := \{(\mathbf{x}, \mathbf{c}') \in \mathcal{S}^{n-1} \times C' : \langle \mathbf{x}, \mathbf{c}' \rangle \approx_\epsilon \cos(\alpha')\}.$$

There is a quantum algorithm $\texttt{Dec}(C')$ *that, given a description of C', implements the map*

$$\widetilde{\mathcal{O}}_{R'} : |\mathbf{x}\rangle |0\rangle \mapsto \begin{cases} |\mathbf{x}\rangle \sum_{\mathbf{c}' \in R'(\mathbf{x})} \frac{1}{\sqrt{|R'(\mathbf{x})|}} |\mathbf{c}'\rangle & \text{if } |R'(\mathbf{x})| \in [1, 2^{n/\log n}] \\ |\mathbf{x}\rangle |\perp\rangle & \text{otherwise} \end{cases}$$

in time $2^{o(n)}$ using an auxiliary register of $2^{o(n)}$ qubits. In particular, $\texttt{Dec}(C')$ *correctly implements $\mathcal{O}_{R'}$ for all $\mathbf{x} \in \mathcal{S}^{n-1}$ satisfying $|R'(\mathbf{x})| \leq 2^{n/\log n}$.*

Proof. The subroutine first enumerates the elements of $R'(\mathbf{x})$ using Lemma 6.2.9 in an auxiliary register, but stops as soon as it has found $2^{n/\log n} + 1$ elements, unless it has already finished earlier. (Note that the algorithm from Lemma 6.2.9 finds elements of $R'(\mathbf{x})$ one by one.) If it finds either zero or $2^{n/\log n} + 1$ elements, which means $|R'(\mathbf{x})| \notin [1, 2^{n/\log n}]$, then it undoes the computation and maps the second register to $|\perp\rangle$. Otherwise, when $|R'(\mathbf{x})| \in [1, 2^{n/\log n}]$, the subroutine prepares a uniform superposition over the found elements in the second register and then uncomputes the auxiliary register. Altogether this takes time $2^{o(n)}$ and uses $2^{o(n)}$ auxiliary qubits. (Note that, just as in Lemma 6.2.3, $|\mathbf{x}, \mathbf{c}'\rangle$ will be encoded by $|\mathbf{x}, i, \mathbf{c}'\rangle$ for some index i that depends on $\mathbf{x}$, $\mathbf{c}'$, and C', but this is not an issue.) □

6.3.4 Search phase

The **Search** phase of Algorithm 6.1 repeatedly invokes a quantum algorithm called `SolutionSearch`, which will be presented in Algorithm 6.5. As mentioned before, this algorithm searches for elements of $\mathcal{T}_{\text{sol}}$ by carefully nesting two layers of amplitude amplification (Lemma 5.2.2) and by searching for collisions under the relations R, R' defined by the sets $C, C' \subseteq \mathcal{S}^{n-1}$ obtained during the **Sampling** phase (and by the angles α, α'). This is achieved by leveraging the data structures $D(R_L)$ and $D(R'_L)$ that were prepared during the **Preprocessing** phase for the finite relations $R_L := R|_{L \times C}$ and $R'_L := R'|_{L \times C'}$.

More precisely, SolutionSearch samples 3-tuple solutions from the set

$$\mathcal{T}_{\mathrm{sol}}(R,R') := \Big\{(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}} \colon\ R(\mathbf{x}) \cap R(\mathbf{y}) \neq \emptyset,\ R'(\tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}) \cap R'(\mathbf{z}) \neq \emptyset, \\ |R(\mathbf{x})| \leq 2^{n/\log n},\ |R'(\tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})| \leq 2^{n/\log n}\Big\} \tag{6.3}$$

by gradually refining the search through the sets $\mathcal{T}_0(R) \supseteq \mathcal{T}_1(R) \supseteq \mathcal{T}_2(R,R')$ defined as

$$\begin{aligned} \mathcal{T}_0(R) &:= \big\{(\mathbf{x},\mathbf{y}) \in L^2 \colon R(\mathbf{x}) \cap R(\mathbf{y}) \neq \emptyset\big\}, \\ \mathcal{T}_1(R) &:= \big\{(\mathbf{x},\mathbf{y}) \in \mathcal{T}_0(R) \colon \langle \mathbf{x},\mathbf{y}\rangle \approx_\epsilon \cos(\theta),\ |R(\mathbf{x})| \leq 2^{n/\log n}\big\}, \\ \mathcal{T}_2(R,R') &:= \big\{(\mathbf{x},\mathbf{y}) \in \mathcal{T}_1(R) \colon \exists \mathbf{z} \in L, (\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(R,R')\big\}. \end{aligned} \tag{6.4}$$

The constraints that $R(\mathbf{x})$ and $R'(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})$ are of size $2^{o(n)}$ are imposed for technical reasons.[7]

The core of SolutionSearch is a subroutine, TupleSamp (presented in Algorithm 6.4), which generates a superposition over a subset of $\mathcal{T}_1(R) \times L$ and "flags" those tuples that belong to $\mathcal{T}_{\mathrm{sol}}(R,R')$. SolutionSearch applies amplitude amplification on top of TupleSamp to keep only those flagged tuples, and then measures the resulting state, yielding an element of $\mathcal{T}_{\mathrm{sol}}(R,R') \subseteq \mathcal{T}_{\mathrm{sol}}$.

The sampling subroutine TupleSamp first generates a superposition over $(\mathbf{x},\mathbf{y}) \in \mathcal{T}_1(R)$ by applying amplitude amplification to a subroutine RCollisionSamp (presented in Algorithm 6.3) that uses $D(R_L)$ to efficiently generate a superposition over $\mathcal{T}_0(R)$. TupleSamp then proceeds by searching for $\mathbf{z}$ such that $(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}$, setting a flag if such a $\mathbf{z}$ is found. Using $D(R'_L)$, we save computation effort by restricting this search to those $\mathbf{z}$ that form an R'-collision with $\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}$.

The structure of the main algorithm 3List and its subroutines, including the **Search** phase described in this section, is illustrated in Figure 6.1. We now give precise definitions of the algorithms forming the **Search** phase, and analyze them in full detail.

[7] In our applications, these sets will be of subexponential size for most $(\mathbf{x},\mathbf{y})$. Excluding the rare cases where this is not the case ensures that the sampling probabilities over $\mathcal{T}_{\mathrm{sol}}(R,R')$ remain approximately uniform.

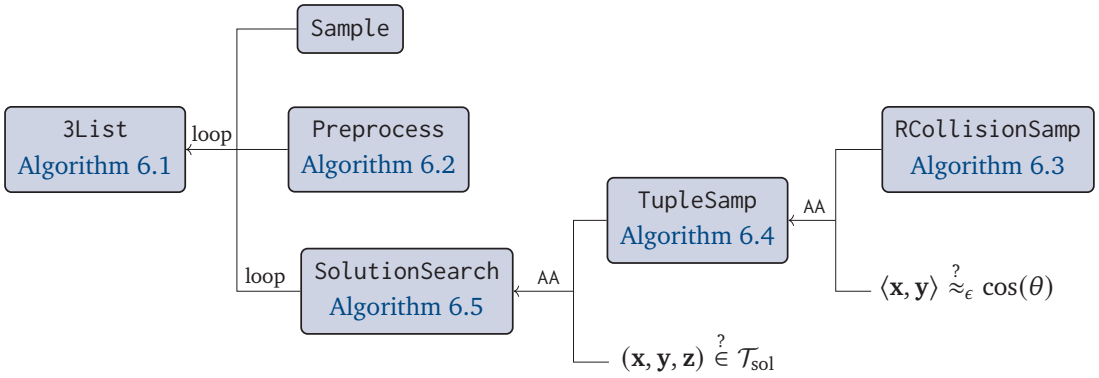


Figure 6.1: Structure of the algorithm `3List`, which repeats the following. First, the **Sampling** phase produces a pair (R, R') of random relations that are stored in a data structure during the **Preprocessing** phase. The algorithm then repeatedly calls `SolutionSearch`, which is instructed to find an element of $\mathcal{T}_{\text{sol}}$ using a nested amplitude amplification (`AA`). Given a subroutine `RCollisionSamp` that creates a superposition over R-collisions $(\mathbf{x}, \mathbf{y}) \in L^2$, the first `AA` amplifies those that satisfy $\langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)$. Next, `TupleSamp` extends them to triples $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ such that $(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z})$ forms an R'-collision (if such a $\mathbf{z}$ exists), and the final `AA` amplifies those triples that belong to $\mathcal{T}_{\text{sol}}$.

Sampling an R-collision

We start with the bottom layer, `RCollisionSamp` (Algorithm 6.3). For arbitrary sets L and C, this algorithm takes as input a QCRAM data structure $D(R)$ storing a relation $R \subseteq L \times C$ (see Section 6.2.2), and outputs a superposition over R-collisions, specifically over $(\mathbf{x}, \mathbf{c}, \mathbf{y}) \in L \times C \times L$ such that $\mathbf{c} \in R(\mathbf{x}) \cap R(\mathbf{y})$, as visualized in part (a) of Figure 6.2. While `RCollisionSamp` works for any relation, we will apply it to $L, C \subseteq \mathcal{S}^{n-1}$ (where L is an instance of Problem 6.1.1 and C an RPC), so we use vector notation such as $\mathbf{x}$ and $\mathbf{c}$ for elements of these sets.

`RCollisionSamp` starts by taking a uniform superposition over all $\mathbf{x} \in L$. Then, for all $\mathbf{x}$ such that $R(\mathbf{x}) \neq \emptyset$, it creates a superposition over all $\mathbf{c} \in R(\mathbf{x})$ in the second register, and subsequently over all $\mathbf{y} \in R^\top(\mathbf{c})$ in the third register. These steps are easy using the data structure $D(R)$. In case $R(\mathbf{x}) = \emptyset$, the second and third register are mapped to $|\perp\rangle\,|\perp\rangle$, but in our applications this typically accounts for a small fraction of the state, so it is easily suppressed using amplitude amplification when Algorithm 6.3 is called by `TupleSamp` (Algorithm 6.4).

Algorithm 6.3: `RCollisionSamp`$(D(R))$

Input: $|0,0,0\rangle$
QCRAM data structure $D(R)$, where $R \subseteq L \times C$ with L, C finite

Output: Superposition $|\psi\rangle$ over $(\mathbf{x}, \mathbf{c}, \mathbf{y}) \in L \times (C \cup \perp) \times (L \cup \perp)$

1. Generate a uniform superposition over L in the first register:

$$|0,0,0\rangle \mapsto \frac{1}{\sqrt{|L|}} \sum_{\mathbf{x} \in L} |\mathbf{x}\rangle |0\rangle |0\rangle$$

2. Controlled on $\mathbf{x}$ in the first register, generate a uniform superposition over the set $R(\mathbf{x})$ in the second register, or map the second register to $|\perp\rangle$ if $R(\mathbf{x})$ is empty:

$$|\mathbf{x}\rangle |0\rangle |0\rangle \mapsto \begin{cases} |\mathbf{x}\rangle \left(\frac{1}{\sqrt{|R(\mathbf{x})|}} \sum_{\mathbf{c} \in R(\mathbf{x})} |\mathbf{c}\rangle \right) |0\rangle & \text{if } R(\mathbf{x}) \neq \emptyset \\ |\mathbf{x}\rangle |\perp\rangle |0\rangle & \text{if } R(\mathbf{x}) = \emptyset \end{cases}$$

3. Controlled on $\mathbf{c} \neq \perp$ in the second register, generate a superposition over the set $R^{\top}(\mathbf{c})$ in the third register, and map $|\perp\rangle |0\rangle$ to $|\perp\rangle |\perp\rangle$:

$$|\mathbf{x}\rangle |\mathbf{c}\rangle |0\rangle \mapsto \begin{cases} |\mathbf{x}\rangle |\mathbf{c}\rangle \left(\frac{1}{\sqrt{|R^{\top}(\mathbf{c})|}} \sum_{\mathbf{y} \in R^{\top}(\mathbf{c})} |\mathbf{y}\rangle \right) & \text{if } \mathbf{c} \neq \perp \\ |\mathbf{x}\rangle |\mathbf{c}\rangle |\perp\rangle & \text{if } \mathbf{c} = \perp \end{cases}$$

4. Return the final quantum state

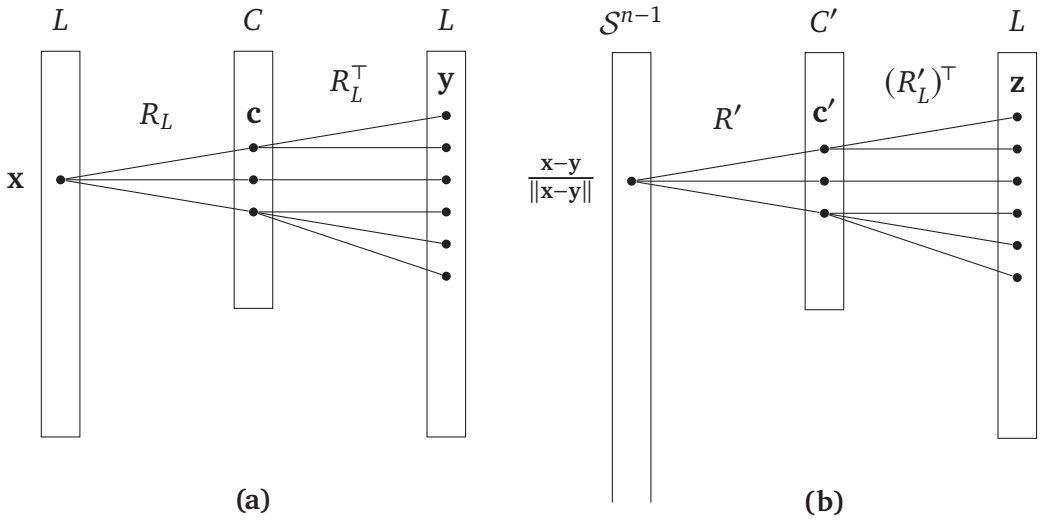


Figure 6.2: Summary of the relations between vectors encountered during the **Search** phase. Part (a) visualizes the relations during the subroutine `RCollisionSamp`, which first creates a superposition over all $\mathbf{x} \in L$, followed by taking, for each such $\mathbf{x}$, a superposition over all $\mathbf{c}$ such that $(\mathbf{x}, \mathbf{c}) \in R_L$, and then, for each such $\mathbf{c}$, over all $\mathbf{y}$ such that $(\mathbf{y}, \mathbf{c}) \in R_L$. This results in a superposition over all R_L-collisions (that is, all R-collisions in L^2). Part (b) visualizes what happens after the first step of `TupleSamp`, which amplifies those R_L-collisions $(\mathbf{x}, \mathbf{y})$ that satisfy $\langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)$. Namely, the second step creates, for any such $(\mathbf{x}, \mathbf{y})$, a superposition over $\mathbf{c}' \in C'$ satisfying $(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{c}') \in R'$, and, for each such $\mathbf{c}'$, the third step creates a superposition over all $\mathbf{z}$ such that $(\mathbf{z}, \mathbf{c}') \in R'_L$, as visualized in the figure, and amplifies those $\mathbf{z}$ satisfying $\langle \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z} \rangle \approx_\epsilon \cos(\theta')$. As for most $(\mathbf{x}, \mathbf{y})$ no such $\mathbf{z}$ exists, `SolutionSearch` applies amplitude amplification on top of `TupleSamp` to amplify exactly those $(\mathbf{x}, \mathbf{y})$ where such a $\mathbf{z}$ does exist.

Lemma 6.3.7 (Analysis of `RCollisionSamp`)**.** *For finite sets L and C, let $R \subseteq L \times C$. Let $D(R)$ be a QCRAM data structure for R.* `RCollisionSamp`$(D(R))$ *(Algorithm 6.3) outputs a state $|\psi\rangle$ such that, for all $(\mathbf{x}, \mathbf{c}, \mathbf{y}) \in L \times C \times L$ satisfying $\mathbf{c} \in R(\mathbf{x}) \cap R(\mathbf{y})$, we have*

$$\langle \mathbf{x}, \mathbf{c}, \mathbf{y} | \psi \rangle = \frac{1}{\sqrt{|L|\,|R(\mathbf{x})|\,|R^\top(\mathbf{c})|}}.$$

The algorithm uses $O(\log|L| + \log|C|)$ time and QCRAM queries, and $O(\log|L| + \log|C|)$ qubits.

Proof. The operations used by the subroutine (taking a uniform superposition over L, taking a uniform superposition over $R(\mathbf{x})$ for any $\mathbf{x} \in L$, and taking a uniform superposition over $R^\top(\mathbf{c})$ for any $\mathbf{c} \in C$) can all be done using $O(\log|L| + \log|C|)$ time and QCRAM queries, by Lemma 6.2.3. Since Algorithm 6.3 uses $O(\log|L| + \log|C|)$ qubits, the claim on the time and memory complexities follows. The output state of Algorithm 6.3 is

$$|\psi\rangle := \frac{1}{\sqrt{|L|}} \sum_{\mathbf{x}\in L:\, R(\mathbf{x})\neq\emptyset} \frac{1}{\sqrt{|R(\mathbf{x})|}} \sum_{\mathbf{c}\in R(\mathbf{x})} \frac{1}{\sqrt{|R^\top(\mathbf{c})|}} \sum_{\mathbf{y}\in R^\top(\mathbf{c})} |\mathbf{x}\rangle\,|\mathbf{c}\rangle\,|\mathbf{y}\rangle + \frac{1}{\sqrt{|L|}} \sum_{\mathbf{x}\in L:\, R(\mathbf{x})=\emptyset} |\mathbf{x}\rangle\,|\perp\rangle\,|\perp\rangle\,.$$

Therefore, $\langle \mathbf{x}, \mathbf{c}, \mathbf{y}|\psi\rangle = 1/\sqrt{|L|\,|R(\mathbf{x})|\,|R^\top(\mathbf{c})|}$ if both $\mathbf{c} \in R(\mathbf{x})$ and $\mathbf{y} \in R^\top(\mathbf{c})$. □

TupleSamp

The next layer is the subroutine `TupleSamp`, presented in Algorithm 6.4, which considers relations on $\mathcal{S}^{n-1}$. Its goal is to construct a quantum state $|\psi'\rangle$ that has sufficiently large overlap with $\mathcal{T}_{\mathrm{sol}}(R, R')$ (Equation (6.3)) so that putting amplitude amplification on top (as will be done by `SolutionSearch`) allows to sample from $\mathcal{T}_{\mathrm{sol}}(R, R')$ at not too high cost. In fact, for our applications, we want that *each* element in $\mathcal{T}_{\mathrm{sol}}(R, R')$ has rather large overlap with $|\psi'\rangle$, ensuring that repeatedly calling `SolutionSearch` allows for finding *all* elements of $\mathcal{T}_{\mathrm{sol}}(R, R')$, and not just the same element over and over again.

`TupleSamp` takes as input two relations $R \subseteq \mathcal{S}^{n-1} \times C$ and $R' \subseteq \mathcal{S}^{n-1} \times C'$, or rather their restrictions to L, given as data structures $D(R_L)$ and $D(R'_L)$. It also assumes the existence of an oracle that creates a uniform superposition over $R'(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})$ for vectors $(\mathbf{x}, \mathbf{y}) \in L^2$, which we will implement using Lemma 6.3.6. `TupleSamp` first creates a superposition over R-collisions $(\mathbf{x}, \mathbf{y}) \in L^2$ that belong to $\mathcal{T}_1(R)$ (implying that $\mathbf{x}$ and $\mathbf{y}$ are relatively "close"), and then searches for a "close" $\mathbf{z} \in L$ among those that form an R'-collision with $\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}$. Such a $\mathbf{z}$ might not exist for all $(\mathbf{x}, \mathbf{y})$, so we use a flag qubit that is set to $|1\rangle$ whenever such a $\mathbf{z}$ was found.

In particular, step 1 of `TupleSamp` creates a superposition over $\mathcal{T}_1(R)$ by applying amplitude amplification to `RCollisionSamp` (Algorithm 6.3), amplifying those R-collisions $(\mathbf{x}, \mathbf{y}) \in L^2$ that satisfy $\langle \mathbf{x}, \mathbf{y}\rangle \approx_\epsilon \cos(\theta)$ and $|R(\mathbf{x})| \leq 2^{n/\log n}$.[8]

[8] Note that the amplification step also checks whether $\mathbf{y} \neq \perp$, because the superposition constructed by `RCollisionSamp` may have nonzero amplitude on pairs $(\mathbf{x}, \mathbf{y})$ with $\mathbf{y} = \perp$, namely if $R(\mathbf{x}) = \emptyset$.

Algorithm 6.4: $\texttt{TupleSamp}(D(R_L), D(R'_L))$

Input: $|0,0,0,0,0\rangle\,|0,0\rangle$, where the last two qubits form the flag register
QCRAM data structures $D(R_L)$ and $D(R'_L)$, where

$$R_L := R|_{L\times C}$$
$$R'_L := R'|_{L\times C'}$$

with $L, C, C' \subseteq \mathcal{S}^{n-1}$ finite, $R \subseteq \mathcal{S}^{n-1}\times C$, and $R' \subseteq \mathcal{S}^{n-1}\times C'$

Output: Superposition $|\psi'\rangle$ over $(\mathbf{x},\mathbf{c},\mathbf{y},\mathbf{c}',\mathbf{z},b_F,b'_F)$ such that all basis states in its support with $b'_F = 1$ satisfy $(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\text{sol}}(R,R')$

1. Apply $\texttt{AA}_r(\texttt{RCollisionSamp}(D(R_L)), \texttt{RCollisionCheck})$ to the first three registers and first flag qubit, where:
   - $r = \max\{1, \sqrt{|\mathcal{T}_0(R)|}\}\,2^{o(n)}$ (computed using $D(R_L)$)
   - `RCollisionSamp` (Algorithm 6.3) generates a superposition $|\psi\rangle$ over $(\mathbf{x},\mathbf{c},\mathbf{y})$ such that either $\mathbf{c} \in R(\mathbf{x}) \cap R(\mathbf{y})$ or $\mathbf{y} = \perp$
   - `RCollisionCheck` checks if $\mathbf{y} \neq \perp$, $\langle \mathbf{x},\mathbf{y}\rangle \approx_\epsilon \cos(\theta)$, and $|R(\mathbf{x})| \leq 2^{n/\log n}$

   This step sets the first flag qubit to $|1\rangle$ if and only if $\mathcal{T}_1(R) \neq \emptyset$.

2. Controlled on the first three registers being in state $|\mathbf{x},\mathbf{c},\mathbf{y}\rangle$ and the first flag qubit in $|1\rangle$, apply the following unitary map to the fourth register:

$$|0\rangle \mapsto \begin{cases} \dfrac{1}{\sqrt{|R'(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})|}} \displaystyle\sum_{\mathbf{c}'\in R'(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})} |\mathbf{c}'\rangle & \text{if } |R'(\tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})| \in [1, 2^{n/\log n}] \\ |\perp\rangle & \text{otherwise} \end{cases}$$

3. Controlled on the first four registers being in state $|\mathbf{x}, \mathbf{c}, \mathbf{y}, \mathbf{c}'\rangle$ and the first flag qubit in $|1\rangle$, apply $\mathsf{AA}_{r'}(\mathsf{zSamp}(\mathbf{c}'), \mathsf{zCheck}(\mathbf{x}, \mathbf{y}))$ to the fifth register and second flag qubit, where:
   - $r' := \sqrt{|(R'_L)^\top(\mathbf{c}')|}$ (computed using $D(R'_L)$)
   - $\mathsf{zSamp}(\mathbf{c}')$ uses $D(R'_L)$ to apply the following map to the fifth register:

$$|0\rangle \mapsto \begin{cases} \dfrac{1}{\sqrt{|(R'_L)^\top(\mathbf{c}')|}} \displaystyle\sum_{\mathbf{z} \in (R'_L)^\top(\mathbf{c}')} |\mathbf{z}\rangle & \text{if } \mathbf{c}' \neq \perp \text{ and } (R'_L)^\top(\mathbf{c}') \neq \emptyset \\ |\perp\rangle & \text{otherwise} \end{cases}$$

   - $\mathsf{zCheck}(\mathbf{x}, \mathbf{y})$ checks if $\mathbf{z} \neq \perp$ and $\langle \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z} \rangle \approx_\epsilon \cos(\theta')$

   This step ensures the second flag qubit has support on $|1\rangle$ if and only if $\mathcal{T}_{\text{sol}}(R, R') \neq \emptyset$.

4. Return the final quantum state

Lemma 6.3.8 (Analysis of TupleSamp). *For $L, C, C' \subseteq \mathcal{S}^{n-1}$ of size $2^{O(n)}$, let $R \subseteq \mathcal{S}^{n-1} \times C$ and $R' \subseteq \mathcal{S}^{n-1} \times C'$ be such that (R, R') is well-balanced on L (Definition 6.3.4). Let $D(R_L)$ and $D(R'_L)$ be QCRAM data structures for $R_L := R|_{L\times C}$ and $R'_L := R'|_{L\times C'}$. With probability at least $1 - 2^{-\omega(n)}$, $\mathtt{TupleSamp}(D(R_L), D(R'_L))$ (Algorithm 6.4) generates a quantum state $|\psi'\rangle$ and satisfies:*

(1) *Complexity: If $\mathcal{T}_1(R) \neq \emptyset$, then it uses*

$$\left(\sqrt{\frac{|\mathcal{T}_0(R)|}{|\mathcal{T}_1(R)|}} + \sqrt{\frac{|L|}{|C'|}}\right) 2^{o(n)}$$

time and QCRAM queries, and $\max\{1, \sqrt{|\mathcal{T}_0(R)|}\}\, 2^{o(n)}$ otherwise. It uses $2^{o(n)}$ qubits.

(2) *Near-uniform sampling: The probability that measuring $|\psi'\rangle$ yields outcome $(\mathbf{x}, \mathbf{y}, \mathbf{z}, b'_F)$ with $b'_F = 1$ satisfies*

$$\Pr[(\mathbf{x}, \mathbf{y}, \mathbf{z}, 1)] =_n \begin{cases} \dfrac{1}{|\mathcal{T}_1(R)|} & \text{if } (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(R, R') \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Suppose (R, R') is well-balanced on L. We will show that there exists an implementation of $\mathtt{TupleSamp}(D(R_L), D(R'_L))$ (Algorithm 6.4) with the desired properties.

Let Π be the orthogonal projector onto the subspace where $b_F = 1$. Lemma 6.3.7 implies that $\mathtt{RCollisionSamp}(D(R_L))$ has complexity $2^{o(n)}$ and generates a state $|\psi\rangle$ such that

$$\|\Pi\,|\psi\rangle\|^2 = \sum_{\substack{(\mathbf{x},\mathbf{y})\in\mathcal{T}_1(R),\\ \mathbf{c}\in R(\mathbf{x})\cap R(\mathbf{y})}} \frac{1}{|L|\,|R(\mathbf{x})|\,|R_L^\top(\mathbf{c})|} =_n \frac{|\mathcal{T}_1(R)|}{|\mathcal{T}_0(R)|}$$

because $(\mathbf{x}, \mathbf{y}) \in \mathcal{T}_1(R)$ implies $|R(\mathbf{x})| \in [1, 2^{o(n)}]$ and because condition (i) of being well-balanced (Definition 6.3.4) implies $|L|\,|R_L^\top(\mathbf{c})| =_n |L|\,|R_L|/|C| =_n |\mathcal{T}_0(R)|$ for all $\mathbf{c} \in C$. Moreover, all $r =_n \max\{1, |L|/\sqrt{|C|}\}$ satisfy $r =_n \max\{1, \sqrt{|\mathcal{T}_0(R)|}\}$. By taking a sufficiently large $r = \max\{1, |L|/\sqrt{|C|}\}\, 2^{o(n)}$ (note that $|L|$ and $|C|$ are known), we ensure $\|\Pi\,|\psi\rangle\| \geq 1/r$ whenever $\|\Pi\,|\psi\rangle\| \neq 0$ (the latter holds if and only if $\mathcal{T}_1(R) \neq \emptyset$). Next, note that RCollisionCheck can be implemented in time $2^{o(n)}$, using one query to $D(R_L)$ to compute $|R(\mathbf{x})|$. Therefore, Lemma 5.2.2

implies that, with probability $1-2^{-\omega(n)}$, step 1 of $\texttt{TupleSamp}$ can be implemented in time $t_1 =_n \sqrt{\max\{1,|\mathcal{T}_0(R)|\}/\max\{1,|\mathcal{T}_1(R)|\}}$, uses one auxiliary qubit, and maps

$$|0,0,0\rangle|0\rangle \mapsto \frac{\Pi|\psi\rangle}{\|\Pi|\psi\rangle\|}|1\rangle$$

if $\mathcal{T}_1(R)\neq\emptyset$, and $|0,0,0\rangle|0\rangle\mapsto|\psi\rangle|0\rangle$ otherwise. In the remainder of this proof, we assume step 1 was successful, so the first flag qubit is either fully supported on $|0\rangle$ or fully supported on $|1\rangle$. Since steps 2 and 3 are only applied in the latter case, in the remainder of the proof we omit writing this first flag qubit.

Step 2 can be implemented using the quantum algorithm $\texttt{Dec}(C')$ from Lemma 6.3.6 in time $t_2 = 2^{o(n)}$ using $2^{o(n)}$ auxiliary qubits.[9] For step 3, we use the data structure $D(R'_L)$ to implement $\texttt{zSamp}(\mathbf{c}')$ using $2^{o(n)}$ time and QCRAM queries, as established by Lemma 6.2.3. Note that $D(R'_L)$ stores the size $|(R'_L)^\top(\mathbf{c}')|$, so step 3 (controlled on $(\mathbf{x},\mathbf{c},\mathbf{y},\mathbf{c}')$) can be implemented using the algorithm $\texttt{AA}_{r'}$ from Lemma 5.2.2 with $r' := \sqrt{|(R'_L)^\top(\mathbf{c}')|}$. In addition, $\texttt{zCheck}(\mathbf{x},\mathbf{y})$ is a straightforward check that takes time at most $2^{o(n)}$. By Lemma 5.2.2, this implements step 3 with probability $1-2^{-\omega(n)}$ in time $t_3 = \sqrt{\max_{\mathbf{c}'\in C'}|(R'_L)^\top(\mathbf{c}')|}\,2^{o(n)}$, and maps

$$|\mathbf{x},\mathbf{c},\mathbf{y},\mathbf{c}'\rangle|0\rangle|1,0\rangle \mapsto \begin{cases} |\mathbf{x},\mathbf{c},\mathbf{y},\mathbf{c}'\rangle \dfrac{1}{\sqrt{|L(\mathbf{x},\mathbf{y},\mathbf{c}')|}} \displaystyle\sum_{\mathbf{z}\in L(\mathbf{x},\mathbf{y},\mathbf{c}')} |\mathbf{z}\rangle|1,1\rangle & \text{if } (*) \\ |\mathbf{x},\mathbf{c},\mathbf{y},\mathbf{c}'\rangle|\perp\rangle|1,0\rangle & \text{otherwise} \end{cases}$$

where $(*)$ is the condition that both $\mathbf{c}'\neq\perp$ and $L(\mathbf{x},\mathbf{y},\mathbf{c}')\neq\emptyset$, and $L(\mathbf{x},\mathbf{y},\mathbf{c}') := \{\mathbf{z}\in L\colon \langle\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|},\mathbf{z}\rangle \approx_\epsilon \cos(\theta'), \mathbf{c}'\in R'(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})\cap R'(\mathbf{z})\}$.

Since $t_3 =_n |L|/|C'|$ by condition (ii) of being well-balanced, $\texttt{TupleSamp}$ uses

$$t_1+t_2+t_3 =_n \sqrt{\frac{|\mathcal{T}_0(R)|}{|\mathcal{T}_1(R)|}} + \sqrt{\frac{|L|}{|C'|}}$$

time and QCRAM queries if $\mathcal{T}_1(R)\neq\emptyset$, and $t_1+t_2+t_3 =_n \max\{1,\sqrt{|\mathcal{T}_0(R)|}\}$ otherwise. The claim on the memory complexity follows immediately from Lemma 6.3.6, Lemma 6.3.7, and the construction of the algorithm. This proves statement (1).

By construction of $\texttt{TupleSamp}$, its output state $|\psi'\rangle$ has nonzero overlap with $|\mathbf{x},\mathbf{y},\mathbf{z}\rangle|1\rangle$ (the latter register corresponding to the second flag qubit) if and only

[9] We remark that $\texttt{Dec}(C')$ takes as input a description of C'. We assume without loss of generality that such a description (which is of size $2^{o(n)}$) is stored in $D(R'_L)$ during the **Preprocessing** phase.

if $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(R, R')$, so fix an arbitrary $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(R, R')$. Note that this implies $\mathcal{T}_1(R) \neq \emptyset$, so the first flag qubit is fully supported on $|1\rangle$. By the above, $|\psi'\rangle$ satisfies

$$\langle \mathbf{x}, \mathbf{c}, \mathbf{y}, \mathbf{c}', \mathbf{z}, 1, 1|\psi'\rangle = \frac{1}{\sqrt{\|\Pi\,|\psi\rangle\|^2 |L|\,|R(\mathbf{x})|\,|R_L^\top(\mathbf{c})|\,|R'(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})|\,|L(\mathbf{x}, \mathbf{y}, \mathbf{c}')|}}$$

if $\mathbf{c} \in R(\mathbf{x}) \cap R(\mathbf{y})$ and $\mathbf{c}' \in R'(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}) \cap R'(\mathbf{z})$; and $\langle \mathbf{x}, \mathbf{c}, \mathbf{y}, \mathbf{c}', \mathbf{z}, 1, 1|\psi'\rangle = 0$ otherwise. Using condition (i) and (iii) of being well-balanced and the fact that $|R(\mathbf{x})| =_n |R'(\frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})| =_n 1$ for $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(R, R')$, we conclude that a measurement of $|\psi'\rangle$ yields outcome $(\mathbf{x}, \mathbf{y}, \mathbf{z}, 1)$ with probability $1/|\mathcal{T}_1(R)|$, up to subexponential factors in n. □

In fact, using step 1 of `TupleSamp` (Algorithm 6.4) we can also decide whether $\mathcal{T}_1(R)$ is nonempty, as the following lemma shows.

Lemma 6.3.9 (Deciding whether $\mathcal{T}_1(R) \neq \emptyset$)**.** *For $L, C \subseteq \mathcal{S}^{n-1}$ of size $2^{O(n)}$, let $R \subseteq \mathcal{S}^{n-1} \times C$ satisfy condition (i) in Definition 6.3.4. Let $D(R_L)$ be a QCRAM data structure for $R_L := R|_{L\times C}$. There exists a quantum algorithm that, with probability $1 - 2^{-\omega(n)}$, correctly decides whether $\mathcal{T}_1(R) \neq \emptyset$ using $\sqrt{|\mathcal{T}_0(R)|/|\mathcal{T}_1(R)|}\,2^{o(n)}$ time and QCRAM queries if $\mathcal{T}_1(R) \neq \emptyset$, and $\max\{1, \sqrt{|\mathcal{T}_0(R)|}\}\,2^{o(n)}$ otherwise. It uses $2^{o(n)}$ qubits.*

Proof. The quantum algorithm initializes $|0, 0, 0\rangle\,|0\rangle$, runs step 1 of `TupleSamp` (Algorithm 6.4), and then measures the last (flag) qubit. It outputs the measurement outcome. By the proof of Lemma 6.3.8, the runtime is as stated and the measurement outcome is 1 if and only if $\mathcal{T}_1(R) \neq \emptyset$, except with probability $2^{-\omega(n)}$. □

SolutionSearch

Finally, the outermost layer of the **Search** phase is Algorithm 6.5, which uses amplitude amplification to project the output state $|\psi'\rangle$ of `TupleSamp` (Algorithm 6.4) onto $\mathcal{T}_{\text{sol}}(R, R')$. If $\mathcal{T}_{\text{sol}}(R, R')$ is nonempty, then this state $|\psi'\rangle$ is solely supported on $(\mathbf{x}, \mathbf{y}, \mathbf{z}, b'_F)$ with $(\mathbf{x}, \mathbf{y}) \in \mathcal{T}_1(R)$, and with the flag bit set to $b'_F = 1$ only if $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\text{sol}}(R, R')$. Thus, as long as we perform sufficiently many rounds of amplitude amplification, Algorithm 6.5 successfully outputs an element of $\mathcal{T}_{\text{sol}}(R, R')$.

Algorithm 6.5: `SolutionSearch`$(D(R_L), D(R'_L))$

Input: QCRAM data structures $D(R_L)$ and $D(R'_L)$, where

$$R_L := R|_{L \times C}$$
$$R'_L := R'|_{L \times C'}$$

with $L, C, C' \subseteq \mathcal{S}^{n-1}$ finite, $R \subseteq \mathcal{S}^{n-1} \times C$, and $R' \subseteq \mathcal{S}^{n-1} \times C'$

Output: $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\text{sol}}(R, R')$ if $\mathcal{T}_{\text{sol}}(R, R') \neq \emptyset$, and "no solution" otherwise

1. Initialize $|0, 0, 0, 0, 0\rangle\,|0, 0\rangle$
2. If $\mathcal{T}_1(R) \neq \emptyset$ (decided using Lemma 6.3.9), then apply $\mathsf{AA}_r(\mathtt{TupleSamp}(D(R_L), D(R'_L)), \mathtt{TupleCheck})$, where:
   - $r = \sqrt{|\mathcal{T}_1(R)|}\, 2^{o(n)}$ (computed using $D(R_L)$)
   - `TupleSamp` (Algorithm 6.4) generates a superposition over $|\mathbf{x}, \mathbf{c}, \mathbf{y}, \mathbf{c}', \mathbf{z}\rangle\,|b_F, b'_F\rangle$ such that either $b'_F = 0$ or $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\text{sol}}(R, R')$
   - `TupleCheck` checks if $b'_F = 1$
3. Measure. From outcome $(\mathbf{x}, \mathbf{y}, \mathbf{z}, b'_F)$, output $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ if $b'_F = 1$, and "no solution" otherwise.

The relations between the vectors encountered during `SolutionSearch` are visualized in Figure 6.2.

Lemma 6.3.10 (Analysis of `SolutionSearch`). *For $L, C, C' \subseteq \mathcal{S}^{n-1}$ of size $2^{O(n)}$, let $R \subseteq \mathcal{S}^{n-1} \times C$ and $R' \subseteq \mathcal{S}^{n-1} \times C'$ be such that (R, R') is well-balanced on L (Definition 6.3.4). Let $D(R_L)$ and $D(R'_L)$ be QCRAM data structures for $R_L := R|_{L\times C}$ and $R'_L := R'|_{L\times C'}$. With probability $1 - 2^{-\omega(n)}$, $\texttt{SolutionSearch}(D(R_L), D(R'_L))$ (Algorithm 6.5) satisfies the following:*

(1) *Complexity: If $\mathcal{T}_{\mathrm{sol}}(R, R') \neq \emptyset$, then it outputs $\mathbf{t} \in \mathcal{T}_{\mathrm{sol}}(R, R')$ using*

$$\sqrt{\frac{|\mathcal{T}_1(R)|}{|\mathcal{T}_{\mathrm{sol}}(R, R')|}}\left(\sqrt{\frac{|\mathcal{T}_0(R)|}{|\mathcal{T}_1(R)|}} + \sqrt{\frac{|L|}{|C'|}}\right) 2^{o(n)}$$

time and QCRAM queries; otherwise, then it outputs "no solution" using

$$\left(\max\left\{1, \sqrt{|\mathcal{T}_0(R)|}\right\} + \sqrt{|\mathcal{T}_1(R)|\frac{|L|}{|C'|}}\right) 2^{o(n)}$$

time and QCRAM queries. It uses $2^{o(n)}$ qubits.

(2) *Near-uniform sampling: For all $\mathbf{t} \in \mathcal{T}_{\mathrm{sol}}(R, R')$,*

$$\Pr\left[\texttt{SolutionSearch}(D(R_L), D(R'_L)) \text{ outputs } \mathbf{t}\right] =_n \frac{1}{|\mathcal{T}_{\mathrm{sol}}(R, R')|}$$

where the probability is over the internal randomness of `SolutionSearch`.

Proof. For step 1, we decide whether $\mathcal{T}_1(R) \neq \emptyset$ using Lemma 6.3.9, which uses $2^{o(n)}$ auxiliary qubits. In case $\mathcal{T}_1(R) = \emptyset$, the lemma statement follows immediately, so suppose $\mathcal{T}_1(R) \neq \emptyset$. By Lemma 6.3.8, with probability at least $1 - 2^{-\omega(n)}$, the output state $|\psi'\rangle$ of `TupleSamp` (Algorithm 6.4) satisfies

$$\|\Pi' |\psi'\rangle\|^2 =_n \begin{cases} \dfrac{|\mathcal{T}_{\mathrm{sol}}(R, R')|}{|\mathcal{T}_1(R)|} & \text{if } \mathcal{T}_{\mathrm{sol}}(R, R') \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

where Π' denotes the orthogonal projector onto the subspace where $b'_F = 1$. If $\mathcal{T}_{\mathrm{sol}}(R, R') \neq \emptyset$, then $\|\Pi' |\psi'\rangle\| \geq_n 1/\sqrt{|\mathcal{T}_1(R)|}$. Taking r sufficiently large thus ensures $\|\Pi' |\psi'\rangle\| \geq 1/r$. In particular, this choice of r ensures that we can invoke the algorithm AA_r from Lemma 5.2.2 to implement step 2. Since `TupleCheck` can be implemented in time $2^{o(n)}$, statements (1) and (2) now follow from Lemma 6.3.8. □

Overall analysis of the Search phase

We now complete our analysis of the **Search** phase of `3List` (Algorithm 6.1). By applying a coupon-collector argument (see Lemma 6.2.1), it follows that for a sufficiently large search threshold $t = 2^{o(n)}$, the **Search** phase finds, with overwhelming probability, all elements of $\mathcal{T}_{\mathrm{sol}}(R, R')$ using at most $|\mathcal{T}_{\mathrm{sol}}(R, R')|2^{o(n)}$ calls to `SolutionSearch`, even if $|\mathcal{T}_{\mathrm{sol}}(R, R')|$ is not known to the algorithm a priori. This establishes the overall complexity of the **Search** phase.

Lemma 6.3.11 (Analysis of the **Search** phase of `3List`)**.** *For sufficiently large $t = 2^{o(n)}$, consider a single repetition of the **Search** phase of `3List` (Algorithm 6.1) for $L, C, C' \subseteq \mathcal{S}^{n-1}$ of size $2^{O(n)}$, given the QCRAM data structures $D(R_L)$ and $D(R'_L)$ constructed in the **Preprocessing** phase. Suppose (R, R') is well-balanced on L (Definition 6.3.4). With probability at least $1 - 2^{-\omega(n)}$, the **Search** phase constructs a list consisting of all elements of $\mathcal{T}_{\mathrm{sol}}(R, R')$ and uses*

$$\sqrt{\max\{1, |\mathcal{T}_{\mathrm{sol}}(R, R')|\}} \left(\max\{1, \sqrt{|\mathcal{T}_0(R)|}\} + \sqrt{|\mathcal{T}_1(R)| \frac{|L|}{|C'|}} \right) 2^{o(n)}$$

time and QCRAM queries, $2^{o(n)}$ qubits, and $\max\{1, |\mathcal{T}_{\mathrm{sol}}(R, R')|\}\, 2^{o(n)}$ additional classical memory.

Proof. The **Search** phase of `3List` is exactly the algorithm $\mathcal{A}(\mathtt{Samp}, t)$ in the proof of Lemma 6.2.1 with $X = \mathcal{T}_{\mathrm{sol}}(R, R')$ and $\mathtt{Samp} = \mathtt{SolutionSearch}$. If $\mathcal{T}_{\mathrm{sol}}(R, R') = \emptyset$, then the **Search** phase will finish after at most t repetitions of step *ii*, correctly yielding an empty set S. Otherwise, by Lemma 6.3.10, with probability $1 - 2^{-\omega(n)}$, there exists $\varepsilon \geq 2^{-o(n)}$ such that $\Pr[\mathtt{SolutionSearch}\text{ outputs }\mathbf{t}] \geq \varepsilon / |\mathcal{T}_{\mathrm{sol}}(R, R')|$ for all $\mathbf{t} \in \mathcal{T}_{\mathrm{sol}}(R, R')$. Therefore, for all $t \geq \frac{1}{\varepsilon} \ln(|\mathcal{T}_{\mathrm{sol}}(R, R')|) + \omega(n)$, Lemma 6.2.1 implies that the **Search** phase outputs all elements of $\mathcal{T}_{\mathrm{sol}}(R, R')$ in time $t\,\mathsf{S}\,|\mathcal{T}_{\mathrm{sol}}(R, R')|^{1+o(n)}$, where S denotes the time complexity of `SolutionSearch`. Since $|\mathcal{T}_{\mathrm{sol}}(R, R')| \leq |L|^3 = 2^{O(n)}$, it suffices to take sufficiently large $t = 2^{o(n)}$. The complexity of the **Search** phase then follows from Lemma 6.3.10. □

6.3.5 Final ingredients

In this section, we present a few lemmas that will be used to prove our main theorem, Theorem 6.3.2. Most of their proofs rely on the Chernoff bound, specifically on Corollary 2.1.4.

Lemma 6.3.12. *Let m be a positive integer and let $\alpha \in (0, \pi/2)$ satisfy $\alpha = \Omega(1)$ and $mp_\alpha = \omega(n)$. Let $C \subseteq \mathcal{S}^{n-1}$ be of size $2^{O(n)}$. With probability $1 - 2^{-\omega(n)}$ over $L \sim U(\mathcal{S}^{n-1}, m)$, $|R_L^\top(\mathbf{c})| =_n mp_\alpha$ for all $\mathbf{c} \in C$, and thus $|R_L| =_n |C| mp_\alpha$, where $R_L := \{(\mathbf{x}, \mathbf{c}) \in L \times C : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)\}$.*

Proof. Fix $\mathbf{c} \in C$. View $L \sim U(\mathcal{S}^{n-1}, m)$ as an ordered sequence $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ of i.i.d. uniformly random unit vectors, and consider the sum $X = \sum_{i \in [m]} X_i$ of i.i.d. random variables $X_i \in \{0, 1\}$ defined by $X_i = 1$ if and only if $\langle \mathbf{x}_i, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)$. Note that $X = |R_L^\top(\mathbf{c})|$. By Lemma 6.2.6 and by assumption, $\mathbb{E}_L[X] =_n mp_\alpha = \omega(n)$, so Corollary 2.1.4 implies $|R_L^\top(\mathbf{c})| =_n mp_\alpha$ except with probability at most $\exp(-\omega(n))$. Hence, applying a union bound over all $2^{O(n)}$ vectors $\mathbf{c} \in C$ yields $|R_L^\top(\mathbf{c})| =_n mp_\alpha$ for all $\mathbf{c} \in C$, except with probability $|C| \exp(-\omega(n)) = \exp(-\omega(n))$. □

Lemma 6.3.13. *Let $m = 2^{O(n)}$ and let $\theta' \in (0, \pi/2)$ satisfy $\theta' = \Omega(1)$. With probability $1 - 2^{-\omega(n)}$ over $L \sim U(\mathcal{S}^{n-1}, m)$, $|\{\mathbf{z} \in L : \langle \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z} \rangle \approx_\epsilon \cos(\theta')\}| \leq_n \max\{1, mp_{\theta'}\}$ for all $(\mathbf{x}, \mathbf{y}) \in L^2$.*

Proof. View L as an ordered sequence $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ and let $(i, j) \in [m]^2$. For all $k \in [m-2]$, define the random variable $X_k \in \{0, 1\}$ by $X_k = 1$ if and only if $\langle (\mathbf{x}_i - \mathbf{x}_j)/\|\mathbf{x}_i - \mathbf{x}_j\|, \mathbf{x}_k \rangle \approx_\epsilon \cos(\theta')$. If $L \sim U(\mathcal{S}^{n-1}, m)$, then the X_k are i.i.d. and $\mathbb{E}_L[\sum_{k \in [m-2]} X_k] =_n mp_{\theta'}$ by Lemma 6.2.6. Therefore, Corollary 2.1.4 implies $\sum_{k \in [m-2]} X_k \leq_n \max\{1, mp_{\theta'}\}$ except with probability at most $\exp(-\omega(n))$. The lemma statement now follows by applying a union bound over all $(i, j) \in [m]^2$, using that $m = 2^{O(n)}$ and that including $\mathbf{x}_i$ or $\mathbf{x}_j$ in the count does not affect the asymptotic upper bound. □

Lemma 6.3.14 (Size of $\mathcal{T}_1(R)$)**.** *Let $m = 2^{O(n)}$ and let $\theta, \alpha \in (0, \pi/2)$ satisfy $\min\{\theta, 2\alpha - \theta\} = \Omega(1)$ and $m\mathcal{W}(\theta, \alpha \mid \alpha) = \omega(n)$. Let $C \subseteq \mathcal{S}^{n-1}$ be of size $2^{O(n)}$, $R := \{(\mathbf{x}, \mathbf{c}) \in \mathcal{S}^{n-1} \times C : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)\}$, and $R_L := R|_{L \times C}$. With probability $1 - 2^{-\omega(n)}$ over $L \sim U(\mathcal{S}^{n-1}, m)$, $|\mathcal{T}_1(R)| \leq_n |R_L| m\mathcal{W}(\theta, \alpha \mid \alpha)$, where $\mathcal{T}_1(R)$ is defined in Equation (6.4).*

Proof. Viewing L as an ordered sequence $(\mathbf{x}_1, \dots, \mathbf{x}_m)$ of unit vectors, let $i \in [m]$ and $\mathbf{c} \in R(\mathbf{x}_i)$. Note that $L \sim U(\mathcal{S}^{n-1}, m)$ implies $L_{-i} \sim U(\mathcal{S}^{n-1}, m-1)$, where $L_{-i} := (\mathbf{x}_1, \dots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \dots, \mathbf{x}_m)$. Therefore, $\Pr_{\mathbf{y} \sim U(L_{-i})}[\langle \mathbf{x}_i, \mathbf{y}\rangle \approx_\epsilon \cos(\theta), \langle \mathbf{y}, \mathbf{c}\rangle \approx_\epsilon \cos(\alpha)] =_n \mathcal{W}(\theta, \alpha \mid \alpha)$ by Lemma 6.2.7, giving

$$\begin{aligned}\mathbb{E}_{L_{-i}}[|\{\mathbf{y} \in L_{-i} \colon \langle \mathbf{x}_i, \mathbf{y}\rangle \approx_\epsilon \cos(\theta), \langle \mathbf{y}, \mathbf{c}\rangle \approx_\epsilon \cos(\alpha)\}|] &=_n (m-1)\mathcal{W}(\theta, \alpha \mid \alpha) \\ &=_n m\mathcal{W}(\theta, \alpha \mid \alpha)\end{aligned}$$

which is $\omega(n)$. Thus, $|\{\mathbf{y} \in L_{-i} \colon \langle \mathbf{x}_i, \mathbf{y}\rangle \approx_\epsilon \cos(\theta), \langle \mathbf{y}, \mathbf{c}\rangle \approx_\epsilon \cos(\alpha)\}| =_n m\mathcal{W}(\theta, \alpha \mid \alpha)$, except with probability $\exp(-\omega(n))$ by Corollary 2.1.4.

Using a union bound over all $i \in [m]$ and $\mathbf{c} \in R(\mathbf{x}_i)$ yields $\sum_{i \in [m]} \sum_{\mathbf{c} \in R(\mathbf{x}_i)} |\{\mathbf{y} \in L_{-i} \colon \langle \mathbf{x}_i, \mathbf{y}\rangle \approx_\epsilon \cos(\theta), \langle \mathbf{y}, \mathbf{c}\rangle \approx_\epsilon \cos(\alpha)\}| \le_n |R_L| m\mathcal{W}(\theta, \alpha \mid \alpha)$, except with probability $m|C| \exp(-\omega(n)) = \exp(-\omega(n))$. The statement now follows by definition of $\mathcal{T}_1(R)$. □

Lemma 6.3.15 (Size of $\mathcal{T}_{\mathrm{sol}}(L, \theta, \theta')$)**.** *Let $m = 2^{\Omega(n)}$ and let $\theta, \theta' \in (0, \pi/2)$ satisfy $\min\{\theta, \theta'\} = \Omega(1)$ and $\min\{m^2 p_\theta, m^2 p_{\theta'}, m^3 p_\theta p_{\theta'}\} = 2^{\Omega(n)}$. With probability $1 - 2^{-\Omega(n)}$ over $L \sim U(\mathcal{S}^{n-1}, m)$, we have*

$$|\{(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(L, \theta, \theta') \colon \mathbf{x}, \mathbf{y}, \mathbf{z} \text{ are distinct}\}| =_n m^3 p_\theta p_{\theta'}$$

where $\mathcal{T}_{\mathrm{sol}}(L, \theta, \theta')$ is defined in Equation (6.2).

Proof. We think of L as an ordered sequence $(\mathbf{x}_1, \dots, \mathbf{x}_m)$ of i.i.d. uniformly random unit vectors. Let I denote the set of all $m(m-1)(m-2)$ triples (i, j, k) of distinct indices from $[m]$. For each $(i, j, k) \in I$, let Z_{ijk} be the indicator random variable defined by $Z_{ijk} = 1$ if and only if both $\langle \mathbf{x}_i, \mathbf{x}_j\rangle \approx_\epsilon \cos(\theta)$ and $\langle \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \mathbf{x}_k\rangle \approx_\epsilon \cos(\theta')$. Then $Z := \sum_{(i,j,k) \in I} Z_{ijk}$ counts the number of elements in $\mathcal{T}_{\mathrm{sol}}(L, \theta, \theta')$ that consist of distinct vectors. By linearity of expectation, $\mu := \mathbb{E}[Z]$ satisfies $\mu = m(m-1)(m-2)pp'$, where

$$p := \Pr[\langle \mathbf{x}, \mathbf{y}\rangle \approx_\epsilon \cos(\theta)] \quad \text{and} \quad p' := \Pr[\langle \tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z}\rangle \approx_\epsilon \cos(\theta')]$$

for independent samples $\mathbf{x}, \mathbf{y}, \mathbf{z} \sim U(\mathcal{S}^{n-1})$. By Lemma 6.2.6, $p =_n p_\theta$ and $p' =_n p_{\theta'}$, so it suffices to show $Z =_n \mu$. We will now argue that the variance of Z is $\sigma^2 \le \mu^2 2^{-\Omega(n)}$. Chebyshev's inequality (Lemma 2.1.2) then implies tight concentration: the probability that $|Z - \mu| \ge \mu/2$ is at most $4\sigma^2/\mu^2$, so $Z =_n \mu$ except with probability $2^{-\Omega(n)}$.

Note that $\sigma^2 = \mathbb{E}[Z^2] - \mu^2$, where $\mathbb{E}[Z^2] = \sum_{(i,j,k) \in I} \sum_{(i',j',k') \in I} \mathbb{E}[Z_{ijk} Z_{i'j'k'}]$.

Fix $(i,j,k) \in I$ and partition I into four disjoint sets I_0, I_1, I_2, I_3, where $I_\ell := \{(i',j',k') \in I : |\{i,j,k\} \cap \{i',j',k'\}| = \ell\}$. By independence and by definition of the sets I_ℓ, we obtain

$$\sum_{(i',j',k')\in I_\ell} \Pr[Z_{i'j'k'} = 1 \mid Z_{ijk} = 1] \leq \begin{cases} (m-3)(m-4)(m-5)pp' & \text{if } \ell = 0 \\ 3(m-3)(m-4)pp' & \text{if } \ell = 1 \\ 3(m-3)\max\{p,p'\} & \text{if } \ell = 2 \\ 1 & \text{if } \ell = 3. \end{cases}$$

Since $\mu = m(m-1)(m-2)pp'$, we obtain

$$\mathbb{E}[Z^2] \leq \mu^2 + O(\mu^2/\min\{m, m^2p, m^2p', \mu\}).$$

By assumption, $\min\{m, m^2p, m^2p', \mu\} =_n \min\{m, m^2p_\theta, m^2p_{\theta'}, m^3p_\theta p_{\theta'}\} = 2^{\Omega(n)}$, which implies $\sigma^2 \leq \mu^2 2^{-\Omega(n)}$. As explained before, the lemma statement now follows from Chebyshev's inequality. □

Lemma 6.3.16. *Let $\theta, \alpha \in (0, \pi/2)$ satisfy $\min\{\theta, 2\alpha - \theta\} = \Omega(1)$. For all $\mathbf{x}, \mathbf{y} \in \mathcal{S}^{n-1}$ that satisfy $\langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)$,*

$$\Pr_{C \sim \mathrm{RPC}(n, \log n, 1/p_\alpha)} \left[R(\mathbf{x}) \cap R(\mathbf{y}) \neq \emptyset \text{ and } |R(\mathbf{x})| \leq 2^{n/\log n} \right] =_n \frac{\mathcal{W}(\alpha, \alpha \mid \theta)}{p_\alpha}$$

where $R := \{(\mathbf{x}, \mathbf{c}) \in \mathcal{S}^{n-1} \times C : \langle \mathbf{x}, \mathbf{c} \rangle \approx_\epsilon \cos(\alpha)\}$.

Proof. Define $b := \log n$. The upper bound follows from Lemma 6.2.7, because $\langle \mathbf{x}, \mathbf{y} \rangle \approx_\epsilon \cos(\theta)$. Since $\Pr_C \left[|R(\mathbf{x})| \in [1, 2^{n/\log n}] \right] \geq_n 1$ by Lemma 6.2.10 (applied with $\theta = 0$) and a careful application of Markov's inequality, we obtain

$$\begin{aligned} &\Pr_C \left[R(\mathbf{x}) \cap R(\mathbf{y}) \neq \emptyset \text{ and } |R(\mathbf{x})| \leq 2^{n/\log n} \right] \\ &= \Pr_C \left[R(\mathbf{x}) \cap R(\mathbf{y}) \neq \emptyset \text{ and } |R(\mathbf{x})| \in [1, 2^{n/\log n}] \right] \\ &\geq_n \Pr_C \left[R(\mathbf{x}) \cap R(\mathbf{y}) \neq \emptyset \,\middle|\, |R(\mathbf{x})| \in [1, 2^{n/\log n}] \right]. \end{aligned}$$

Therefore, it remains to lower bound the latter, i.e., the probability of the event E that there is a center point $\mathbf{c} \in C$ that lands in the region

$$\widetilde{\mathcal{W}}_{\mathbf{x},\alpha,\mathbf{y},\alpha} = \{\mathbf{s} \in \mathcal{S}^{n-1} : \langle \mathbf{x}, \mathbf{s} \rangle \approx_\epsilon \cos(\alpha), \langle \mathbf{y}, \mathbf{s} \rangle \approx_\epsilon \cos(\alpha)\},$$

conditioned on the event E' that the number of center points $\mathbf{c} \in C$ that lie in

the "approximate" spherical cap $\widetilde{\mathcal{C}}_{\mathbf{x},\alpha} = \{\mathbf{s} \in \mathcal{S}^{n-1} : \langle \mathbf{x}, \mathbf{s}\rangle \approx_\epsilon \cos(\alpha)\} \supseteq \widetilde{\mathcal{W}}_{\mathbf{x},\alpha,\mathbf{y},\alpha}$ is between 1 and $2^{n/\log n}$. Thus, suppose event E' holds. Then event E holds if one of those center points $\mathbf{c}^*$ that lie in $\widetilde{\mathcal{C}}_{\mathbf{x},\alpha}$ (let's say $\mathbf{c}^*$ is the first such center point, assuming without loss of generality some ordering on C) also satisfies $\langle \mathbf{y}, \mathbf{c}^*\rangle \approx_\epsilon \cos(\alpha)$, i.e., $\mathbf{c}^* \in \widetilde{\mathcal{W}}_{\mathbf{x},\alpha,\mathbf{y},\alpha}$.

We recall that the distribution $C \sim \mathrm{RPC}(n, b, 1/p_\alpha)$ is defined as taking $C = \mathbf{Q}(C^{(1)} \times \cdots \times C^{(b)})$, where $\mathbf{Q} \sim U(\mathrm{SO}(n))$ and $C^{(i)} \sim U(\frac{1}{\sqrt{b}}\mathcal{S}^{n/b-1}, (1/p_\alpha)^{1/b})$ for $i \in \{1, \ldots, b\}$. Therefore, sampling $C \sim \mathrm{RPC}(n, b, 1/p_\alpha)$ conditional on $|R(\mathbf{x})| \in [1, 2^{n/\log n}]$ is equivalent to sampling C as follows:

1. Sample $\mathbf{c}^* := \mathbf{Q}(\mathbf{v}_1^*, \ldots, \mathbf{v}_b^*)$, by first sampling $\mathbf{Q} \sim U(\mathrm{SO}(n))$ and then sampling the tuple $(\mathbf{v}_1^*, \ldots, \mathbf{v}_b^*) \sim U(\frac{1}{\sqrt{b}}\mathcal{S}^{n/b-1}) \times \cdots \times U(\frac{1}{\sqrt{b}}\mathcal{S}^{n/b-1})$ conditional on $\langle \mathbf{x}, \mathbf{c}^*\rangle \approx_\epsilon \cos(\alpha)$.
2. Sample $\widetilde{C}^{(i)} \sim U(\frac{1}{\sqrt{b}}\mathcal{S}^{n/b-1}, (1/p_\alpha)^{1/b} - 1)$ for all $i \in \{1, \ldots, b\}$, conditional on $|C \cap \widetilde{\mathcal{C}}_{\mathbf{x},\alpha}| \in [1, 2^{n/\log n}]$ where $C := \mathbf{Q}(C^{(1)} \times \cdots \times C^{(b)})$ for $C^{(i)} := \{\mathbf{v}_i^*\} \cup \widetilde{C}^{(i)}$.

The second step does not influence the probability that the vector $\mathbf{c}^*$ sampled in the first step lands in $\widetilde{\mathcal{W}}_{\mathbf{x},\alpha,\mathbf{y},\alpha}$. Moreover, the distribution of the vector $\mathbf{c}^*$ sampled in the first step is the same as sampling $\mathbf{c}^* \sim U(\mathcal{S}^{n-1})$ conditional on $\langle \mathbf{x}, \mathbf{c}^*\rangle \approx_\epsilon \cos(\alpha)$. In other words, we obtain

$$\begin{aligned}
&\Pr_C\left[R(\mathbf{x}) \cap R(\mathbf{y}) \neq \emptyset \mid |R(\mathbf{x})| \in [1, 2^{n/\log n}]\right] \\
&\geq \Pr_C\left[\langle \mathbf{y}, \mathbf{c}^*\rangle \approx_\epsilon \cos(\alpha) \,\middle|\, |R(\mathbf{x})| \in [1, 2^{n/\log n}]\right] \\
&= \Pr_{\mathbf{c}^* \sim U(\mathcal{S}^{n-1})}\left[\langle \mathbf{y}, \mathbf{c}^*\rangle \approx_\epsilon \cos(\alpha) \,\middle|\, \langle \mathbf{x}, \mathbf{c}^*\rangle \approx_\epsilon \cos(\alpha)\right] \\
&=_n \frac{\mathcal{W}(\alpha, \alpha \mid \theta)}{p_\alpha}
\end{aligned}$$

by Lemma 6.2.6 and Lemma 6.2.7 (again, using $\langle \mathbf{x}, \mathbf{y}\rangle \approx_\epsilon \cos(\theta)$). □

6.3.6 Final analysis of our quantum algorithm

We now show that `3List` (Algorithm 6.1) solves Problem 6.1.1 for a random list L of size $m = (27/16)^{n/4+o(n)}$ in complexity $2^{0.284551n+o(n)}$, thereby proving Theorem 6.3.2. First, we formalize the claim we made in the introduction of Section 6.3 regarding $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$.

Lemma 6.3.17. *Let $L \subseteq \mathcal{S}^{n-1}$ and let $\theta,\theta' \in (0,\pi/2)$ satisfy $\cos(\theta) = 1/3$ and $\cos(\theta') = \epsilon + \sqrt{1/3+\epsilon/2}$. Then $\|\mathbf{x}-\mathbf{y}-\mathbf{z}\| \le 1$ for all $(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$, where $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ is defined in Equation (6.2).*
Moreover, there exists $m' = (27/16)^{n/4+o(n)}$ such that for all $m \ge m'$:

(i) $\mathbb{E}_{L\sim U(\mathcal{S}^{n-1},m)}[|\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')|] \ge m$.

(ii) With probability $1-2^{-\Omega(n)}$ over $L \sim U(\mathcal{S}^{n-1},m)$,

$$m \le |\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')| \le_n m^3 p_\theta p_{\theta'}$$

and every $(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ satisfies $\mathbf{x}\neq\mathbf{y}$, $\mathbf{y}\neq\mathbf{z}$, and $\mathbf{z}\neq\mathbf{x}$.

Proof. For all $\mathbf{x},\mathbf{y},\mathbf{z} \in \mathcal{S}^{n-1}$, we have $\|\mathbf{x}-\mathbf{y}-\mathbf{z}\|^2 \le 1$ if and only if $\langle \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z}\rangle \ge \sqrt{(1-\langle \mathbf{x},\mathbf{y}\rangle)/2}$. Consider arbitrary $\theta \in (0,\pi/2)$, and define θ' by $\cos(\theta') = \epsilon + \sqrt{(1-\cos(\theta)+\epsilon)/2}$. Then all $(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ satisfy

$$\sqrt{\frac{1-\langle \mathbf{x},\mathbf{y}\rangle}{2}} \le \sqrt{\frac{1-\cos(\theta)+\epsilon}{2}} = \cos(\theta') - \epsilon \le \langle \frac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z}\rangle$$

and thus $\|\mathbf{x}-\mathbf{y}-\mathbf{z}\| \le 1$. We will now show that the lemma statement holds for $\cos(\theta) = 1/3$ (and thus $\cos(\theta') = \epsilon + \sqrt{1/3+\epsilon/2}$).

Note that $p_{\theta'} = (1-\cos^2(\theta'))^{n/2} =_n (1-(\frac{1-\cos(\theta)}{2}))^{n/2} = (\frac{1+\cos(\theta)}{2})^{n/2} = \cos(\frac{\theta}{2})^n$ by the fixed choice of $\epsilon = 1/(\log n)^2$. Therefore, if $\theta = \Omega(1)$ and $\theta' = \Omega(1)$, we obtain (for $L \sim U(\mathcal{S}^{n-1},m)$)

$$\begin{aligned}
\mathbb{E}_L\left[|\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')|\right] &= \sum_{(\mathbf{x},\mathbf{y},\mathbf{z})\in L^3} \Pr_L\left[\langle \mathbf{x},\mathbf{y}\rangle \approx_\epsilon \cos(\theta)\right] \\
&\qquad \cdot \Pr_L\left[\langle \tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}, \mathbf{z}\rangle \approx_\epsilon \cos(\theta') \,\middle|\, \langle \mathbf{x},\mathbf{y}\rangle \approx_\epsilon \cos(\theta)\right] \\
&=_n m^3 p_\theta p_{\theta'} \\
&=_n m^3 (\sin(\theta)\cos(\tfrac{\theta}{2}))^n
\end{aligned}$$

by Lemma 6.2.6. In particular, there exists $m' = (\sin(\theta)\cos(\frac{\theta}{2}))^{-n/2}2^{o(n)}$ such that, whenever the list size $|L| = m$ is $\geq m'$, the expected size of $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ is at least m. As $(\sin(\theta)\cos(\frac{\theta}{2}))^{-n/2}$ is minimized for $\cos(\theta) = 1/3$, we choose $\cos(\theta) = 1/3$ and choose θ' accordingly. The corresponding m' then satisfies $m' = (27/16)^{n/4+o(n)}$.

It remains to prove part (ii). For all sufficiently large n, with probability 1 over $L \sim U(\mathcal{S}^{n-1}, m)$, every $(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ satisfies $\mathbf{x} \neq \mathbf{y}$, $\mathbf{y} \neq \mathbf{z}$, and $\mathbf{z} \neq \mathbf{x}$ (recall Remark 6.3.1). Indeed, consider $(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$, which implies $\|\mathbf{x}-\mathbf{y}-\mathbf{z}\| \leq 1$. By definition, $\langle \mathbf{x},\mathbf{y}\rangle \approx_\epsilon \cos(\theta)$, so $\mathbf{x} \neq \mathbf{y}$ for sufficiently large n. Next, $\mathbf{y} \neq \mathbf{z}$, because otherwise $\|\mathbf{x}-2\mathbf{y}\| > 1$, a contradiction. Finally, if $\mathbf{x} \neq \mathbf{y}$ while $\mathbf{x} = \mathbf{z}$, then $\langle \mathbf{x},\mathbf{y}\rangle = \cos(\theta) - \epsilon$ (by definition of $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$ and θ'), which occurs with measure 0 for independent $\mathbf{x},\mathbf{y} \sim U(\mathcal{S}^{n-1})$. Therefore, by choosing $m' = (27/16)^{n/4+o(n)}$ sufficiently large, part (ii) follows from Lemma 6.3.15 (note that θ and θ' satisfy the constraints). □

The following lemma implies that if $|\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')| \geq m$ and the parameters (α,α',ℓ,t) of `3List` are chosen appropriately, then, with high probability, `3List` outputs m distinct elements of $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$. To prove Theorem 6.3.2, it then remains to show that, for $\theta,\theta' \in (0,\pi/2)$ as given in Lemma 6.3.17, there exists a suitable choice of (α,α') such that the complexity is as desired.

Lemma 6.3.18. *Let $m = 2^{\Theta(n)}$ and let $\theta,\theta',\alpha,\alpha' \in (0,\pi/2)$ satisfy $\min\{\theta,\theta',2\alpha-\theta,2\alpha'-\theta'\} = \Omega(1)$, $mp_{\theta'} = 2^{o(n)}$, and $\min\{mp_\alpha, mp_{\alpha'}, m\,\mathcal{W}(\theta,\alpha \mid \alpha), m^2\,\mathcal{W}(\theta',\alpha' \mid \alpha')\} = \omega(n)$. Consider sufficiently large*

$$\ell = \frac{p_\alpha}{\mathcal{W}(\alpha,\alpha \mid \theta)}\frac{p_{\alpha'}}{\mathcal{W}(\alpha',\alpha' \mid \theta')}2^{o(n)} \qquad \text{and} \qquad t = 2^{o(n)}.$$

With probability $1 - 2^{-\Omega(n)}$ over $L \sim U(\mathcal{S}^{n-1},m)$ and the internal randomness of the algorithm, `3List`(L,θ,θ') (Algorithm 6.1) with parameters (α,α',ℓ,t) outputs a list containing m elements of $\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')$, if m elements exist, in time

$$\ell\left(m + \sqrt{\frac{|\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')|}{\ell}}\left(\sqrt{m^2 p_\alpha} + \sqrt{m^3 \mathcal{W}(\theta,\alpha \mid \alpha)p_{\alpha'}}\right)\right)2^{o(n)} \tag{6.5}$$

using $\max\{m,|\mathcal{T}_{\mathrm{sol}}(L,\theta,\theta')|\}\,2^{o(n)}$ classical memory and QCRAM bits, and $2^{o(n)}$ qubits.

Proof. On input (L, θ, θ') and with parameters $(\alpha, \alpha', \ell, t)$, `3List` (Algorithm 6.1) samples ℓ RPC pairs, independently. For each sampled RPC pair (C, C'), it obtains corresponding data structures (D, D') from $\texttt{Preprocess}(L, C, C')$ (Algorithm 6.2), and repeatedly applies $\texttt{SolutionSearch}(D, D')$ (Algorithm 6.5). In the remainder of this proof, we write $\mathcal{T}_{\text{sol}}$ as shorthand for $\mathcal{T}_{\text{sol}}(L, \theta, \theta')$ and, for each $j \in [\ell]$, we write (R_j, R'_j) for the relations corresponding to the j-th sampled RPC pair.

We claim that, for all sufficiently large $\ell = \frac{p_\alpha}{\mathcal{W}(\alpha,\alpha|\theta)} \frac{p_{\alpha'}}{\mathcal{W}(\alpha',\alpha'|\theta')} 2^{o(n)}$, the following statements simultaneously hold, except with probability $2^{-\omega(n)}$ over L and the ℓ sampled RPC pairs:

(I) For all $j \in [\ell]$, (R_j, R'_j) is well-balanced on L (Definition 6.3.4).

(II) For all $j \in [\ell]$, $|\mathcal{T}_1(R_j)| \le_n m^2 \mathcal{W}(\theta, \alpha \mid \alpha)$.

(III) For all $\mathbf{t} \in \mathcal{T}_{\text{sol}}$, $|\{j \in [\ell] : \mathbf{t} \in \mathcal{T}_{\text{sol}}(R_j, R'_j)\}| \in [1, 2^{o(n)}]$.

By Lemma 6.3.11, (I) implies that for sufficiently large $t = 2^{o(n)}$, for all $j \in [\ell]$, the j-th repetition of the outer loop of `3List` finds all elements of the corresponding set $\mathcal{T}_{\text{sol}}(R_j, R'_j)$. Since (III) implies $\bigcup_{j=1}^{\ell} \mathcal{T}_{\text{sol}}(R_j, R'_j) = \mathcal{T}_{\text{sol}}$, the output of `3List` contains all elements of $\mathcal{T}_{\text{sol}}$.

Next, an upper bound on the time complexity of `3List` is given by

$$T_{\texttt{3List}} = \sum_{j=1}^{\ell} (T_{\texttt{Sample}}(j) + T_{\texttt{Preprocess}}(j) + T_{\texttt{Search}}(j))$$

where $T_{\texttt{Sample}}(j), T_{\texttt{Preprocess}}(j), T_{\texttt{Search}}(j)$ are upper bounds on the time complexity of the respective phases in the j-th repetition of the outer loop of `3List`. It is immediate that $T_{\texttt{Sample}}(j) = 2^{o(n)}$ for all $j \in [\ell]$, and the **Sampling** phases together use at most $2^{o(n)}$ classical memory (as the memory can be reinitialized in each repetition). By Lemma 6.3.5 and (I), $T_{\texttt{Preprocess}}(j) = m2^{o(n)}$ for all $j \in [\ell]$, and the **Preprocessing** phases together use at most $m2^{o(n)}$ classical memory and QCRAM bits (as the memory can be reinitialized in each repetition). By Lemma 6.3.11, (I) and (II) imply that, with probability at least $1 - 2^{-\omega(n)}$,

$$T_{\texttt{Search}}(j) = \sqrt{\max\{1, |\mathcal{T}_{\text{sol}}(R_j, R'_j)|\}} \left(\sqrt{m^2 p_\alpha} + \sqrt{m^3 \mathcal{W}(\theta, \alpha \mid \alpha) p_{\alpha'}}\right) 2^{o(n)}$$

time and QCRAM queries, $2^{o(n)}$ qubits, and $|\mathcal{T}_{\text{sol}}(R_j, R'_j)|2^{o(n)}$ additional classical memory. Here, we use that (I) implies $|\mathcal{T}_0(R_j)| =_n |L|^2/|C| =_n m^2 p_\alpha$, which is $\omega(n)$ by assumption. By the Cauchy-Schwarz inequality and (III), we obtain

$$\sum_{j=1}^{\ell} \sqrt{\max\{1, |\mathcal{T}_{\text{sol}}(R_j, R'_j)|\}} \le \sqrt{\ell \sum_{j=1}^{\ell} \max\{1, |\mathcal{T}_{\text{sol}}(R_j, R'_j)|\}} \le_n \sqrt{\ell \max\{\ell, |\mathcal{T}_{\text{sol}}|\}}.$$

By assumption, $\min\{m\mathcal{W}(\theta,\alpha \mid \alpha), m^2\mathcal{W}(\theta',\alpha' \mid \alpha')\} = \omega(n)$, so $m^3\mathcal{W}(\theta,\alpha \mid \alpha)\mathcal{W}(\theta',\alpha' \mid \alpha') = \omega(n)$. Thus, $\ell \leq_n m^3 p_\theta p_{\theta'}$ and $\min\{m^2 p_\theta, m^2 p_{\theta'}, m^3 p_\theta p_{\theta'}\} = 2^{\Omega(n)}$, because $p_\theta = \mathcal{W}(\theta,\alpha \mid \alpha)\, 2^{\Omega(n)}$ and $p_{\theta'} = \mathcal{W}(\theta',\alpha' \mid \alpha')\, 2^{\Omega(n)}$. Therefore, with probability at least $1 - 2^{-\Omega(n)}$, $|\mathcal{T}_{\mathrm{sol}}| \geq_n m^3 p_\theta p_{\theta'}$ by Lemma 6.3.15, which implies $\sqrt{\ell \max\{\ell, |\mathcal{T}_{\mathrm{sol}}|\}} =_n \sqrt{\ell |\mathcal{T}_{\mathrm{sol}}|}$. In particular, we obtain

$$T_{\texttt{3List}} \leq_n \ell m + \sqrt{\ell|\mathcal{T}_{\mathrm{sol}}|}\left(\sqrt{m^2 p_\alpha} + \sqrt{m^3\mathcal{W}(\theta,\alpha \mid \alpha)p_{\alpha'}}\right)$$

from which the lemma follows (as the total amount of classical memory used is $\max\{m, |\mathcal{T}_{\mathrm{sol}}|\}\, 2^{o(n)}$).

It remains to prove the conditions (I), (II), and (III) claimed above. By Lemma 6.3.12, Lemma 6.3.13, and a union bound over all $j \in [\ell]$, (I) holds except with probability $2^{-\omega(n)}$ over L. Here, we use that the RPCs are of size $1/p_\alpha$ and $1/p_{\alpha'}$ (resp.), that $m \min\{p_\alpha, p_{\alpha'}\} = \omega(n)$, and that $m p_{\theta'} \leq_n 1$. Moreover, by applying Lemma 6.3.14 (using $m\mathcal{W}(\theta,\alpha \mid \alpha) = \omega(n)$) together with a union bound over the $\ell = 2^{O(n)}$ sampled RPC pairs, we obtain that (II) follows from (I), except with probability $2^{-\omega(n)}$. Finally, to prove (III), let $\mathbf{t} \in \mathcal{T}_{\mathrm{sol}}$ and define $p_{\mathbf{t}} := \Pr_{(C,C')}[\mathbf{t} \in \mathcal{T}_{\mathrm{sol}}(R,R')]$, where the distribution is over $C \sim \mathrm{RPC}(n, \log n, 1/p_\alpha)$ and $C' \sim \mathrm{RPC}(n, \log n, 1/p_{\alpha'})$. By Lemma 6.3.16,

$$\begin{aligned} p_{\mathbf{t}} &= \Pr_{C}\left[|R(\mathbf{x})| \leq 2^{n/\log n}, R(\mathbf{x}) \cap R(\mathbf{y}) \neq \emptyset\right] \\ &\quad \cdot \Pr_{C'}\left[|R'(\tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|})| \leq 2^{n/\log n}, R'(\tfrac{\mathbf{x}-\mathbf{y}}{\|\mathbf{x}-\mathbf{y}\|}) \cap R'(\mathbf{z}) \neq \emptyset\right] \\ &=_n \frac{\mathcal{W}(\alpha,\alpha \mid \theta)}{p_\alpha}\frac{\mathcal{W}(\alpha',\alpha' \mid \theta')}{p_{\alpha'}} \end{aligned}$$

where the first equality uses that C and C' are independent. Consequently, for sufficiently large $\ell = \frac{p_\alpha}{\mathcal{W}(\alpha,\alpha|\theta)}\frac{p_{\alpha'}}{\mathcal{W}(\alpha',\alpha'|\theta')}2^{o(n)}$, we obtain $\mu := \mathbb{E}[|\{j \in [\ell] : \mathbf{t} \in \mathcal{T}_{\mathrm{sol}}(R_j, R'_j)\}|] = \omega(n)$ and $\mu = 2^{o(n)}$, where the expectation is taken over the ℓ RPC pairs that are (independently) sampled during `3List`. Corollary 2.1.4 then implies $|\{j \in [\ell] : \mathbf{t} \in \mathcal{T}_{\mathrm{sol}}(R_j, R'_j)\}| \in [1, 2^{o(n)}]$, except with probability $2^{-\omega(n)}$. By applying a union bound over all (at most $2^{O(n)}$) $\mathbf{t} \in \mathcal{T}_{\mathrm{sol}}$, we conclude that (III) holds with probability $1 - 2^{-\omega(n)}$. □

We complete our analysis by proving Theorem 6.3.2.

Proof of Theorem 6.3.2. Let θ, θ' be according to Lemma 6.3.17. Consider an instance $L \sim U(\mathcal{S}^{n-1}, m)$ of Problem 6.1.1 for sufficiently large $m = (27/16)^{n/4+o(n)}$. By Lemma 6.3.17, with probability $1 - 2^{-\Omega(n)}$, $\mathcal{T}_{\text{sol}}(L, \theta, \theta')$ consists only of 3-tuple solutions and satisfies $m \leq |\mathcal{T}_{\text{sol}}(L, \theta, \theta')| \leq_n m^3 p_\theta p_{\theta'} =_n m$. Therefore, it suffices to show there exists a quantum algorithm that solves Problem 6.1.1 with the claimed time and memory complexity if $m \leq |\mathcal{T}_{\text{sol}}(L, \theta, \theta')| \leq_n m$, except with probability $2^{-\Omega(n)}$.

We remark that $\cos(\theta')$ is so close to $1/\sqrt{3}$ (when $n \to \infty$) that we will without loss of generality assume they are equal (as it only affects $p_{\theta'}$ and $\mathcal{W}(\alpha', \alpha' \mid \theta')$ by subexponential factors for similar reasons as in the proofs of Lemma 6.2.6 and Lemma 6.2.7). Suppose $\alpha, \alpha' \in (0, \pi/2)$ are constants that satisfy:

(I) $\min\{\theta, \theta', 2\alpha - \theta, 2\alpha' - \theta'\} = \Omega(1)$;

(II) $\min\{mp_\alpha, mp_{\alpha'}, m\mathcal{W}(\theta, \alpha \mid \alpha), m^2\mathcal{W}(\theta', \alpha' \mid \alpha')\} = \omega(n)$.

Then, by Lemma 6.3.18 (using that $mp_{\theta'} = 2^{o(n)}$, $m^3 p_\theta p_{\theta'} =_n m$, and that $\theta \geq \theta' = \Omega(1)$), there is a quantum algorithm that, with probability $1 - 2^{-\Omega(n)}$ over the randomness of L and the internal randomness of the algorithm, finds m elements of $\mathcal{T}_{\text{sol}}(L, \theta, \theta')$ (if they exist) with time complexity given by Equation (6.5) for some $\ell = \frac{p_\alpha}{\mathcal{W}(\alpha, \alpha|\theta)} \frac{p_{\alpha'}}{\mathcal{W}(\alpha', \alpha'|\theta')} 2^{o(n)}$, using $\max\{m, |\mathcal{T}_{\text{sol}}(L, \theta, \theta')|\}\, 2^{o(n)}$ classical memory and QCRAM bits, and $2^{o(n)}$ qubits.

To conclude the proof (with $\cos(\theta) = 1/3$ and $\cos(\theta') = 1/\sqrt{3}$), we show that setting $\alpha, \alpha' \in (0, \pi/2)$ such that $\cos(\alpha) = 0.347606$ and $\cos(\alpha') = 0.427124$ implies that (I) and (II) are satisfied, and results in the desired time complexity.

It is easy to verify that condition (I) follows from the choice of $\theta, \theta', \alpha, \alpha'$. Moreover, this particular choice of (α, α') also yields $p_\alpha \geq 2^{-0.092893n+o(n)}$, $p_{\alpha'} \geq 2^{-0.145298n+o(n)}$, $\mathcal{W}(\theta, \alpha \mid \alpha) \geq 2^{-0.136318n+o(n)}$, and $m\mathcal{W}(\theta', \alpha' \mid \alpha') \geq 2^{-0.1482329n+o(n)}$. Because $2^{0.188721d} \leq m \leq 2^{0.188722n}$, condition (II) is satisfied as well. Using similar arguments, Equation (6.5) can be shown to be at most $2^{0.284551n+o(n)}$, completing the proof. □

6.4 Application to SVP

In this section, we detail how our quantum algorithm for Problem 6.1.1 aids in finding short vectors in a lattice, and can be turned into a quantum algorithm for solving approximate SVP with small approximation factor (Problem 2.3.8).

6.4.1 Sieving algorithms for SVP and the uniform heuristic

As mentioned at the start of this chapter, the fastest known heuristic attacks on SVP are based on 2-tuple sieving, which require not only exponential time but also exponential memory. This memory complexity can be somewhat reduced by considering k-tuple sieving for some $k > 2$, at the cost of increasing the time complexity. At a high-level, k-tuple sieving aims to find a short vector in a lattice $\Lambda \in \mathbb{R}^n$ using the following strategy:

1. Sample a list L of many random vectors in Λ of norm roughly R (for large R).
2. Find many k-tuples of vectors in L whose sum or difference yields a vector of norm $< R$, and add these shorter vectors to a new list L'.
3. Repeat step 2 with $L := L'$, until a sufficiently short lattice vector is found.

One repetition of step 2 is called a *sieving iteration*, and (for $k = 3$) its task is closely related to the core computational problem we have studied: Problem 6.1.1.

How many sieving iterations are needed to solve approximate SVP? By first applying LLL-reduction [LLL82] to the lattice basis, Klein's algorithm [Kle00] (see also [GPV08]) allows to sample a list L of sufficiently random lattice vectors of norm at most $2^{O(n)}\lambda_1(\Lambda)$, using time $|L|^{1+o(1)}$. If we instruct the algorithm so that, for fixed $\tau > 0$, each iteration turns a list of vectors of norm $\leq R$ into a list of vectors of norm $\leq (1 - 1/\tau)R$, then there is a choice of $\tau = \mathrm{poly}(n)$ such that after $\mathrm{poly}(n)$ sieving iterations we obtain a list that, *if nonempty*, contains a lattice vector of norm $O(\lambda_1(\Lambda))$. This would solve γ-SVP with constant approximation factor $\gamma = O(1)$ in time $\mathrm{poly}(n) \cdot T$, where T is an upper bound on the time of one sieving iteration.

In order to estimate how many vectors the initial list must contain to actually solve approximate SVP using 2-tuple sieving (i.e., to end up with a nonempty list), Nguyen and Vidick [NV08] introduced a heuristic assumption. It naturally generalizes to k-tuple sieving [BLS16], allowing us to also estimate the memory complexity for $k > 2$ [HK17].

Roughly speaking, the heuristic assumption is that, after each iteration of a sieving algorithm, the obtained lattice points are uniformly and independently distributed within some thin annulus.

> **Heuristic 6.4.1** (Uniform heuristic). *Let $L \subseteq \mathbb{R}^n$ be a list of lattice vectors obtained after some iteration of a heuristic sieving algorithm. After appropriate rescaling, every element of L behaves as if it is an i.i.d. uniform sample from* $\mathrm{ann}_\rho = \{\mathbf{x} \in \mathbb{R}^n : \rho \leq \|\mathbf{x}\| \leq 1\}$ *for some fixed* $\rho < 1$.

This heuristic is typically considered for $\rho \longrightarrow 1$ (that is, the list vectors are assumed to be essentially i.i.d. uniform on $\mathcal{S}^{n-1}$). Of course, this heuristic is technically incorrect, since the output list could include vectors of the form $\mathbf{x} + \mathbf{y}$ and $\mathbf{x} + \mathbf{z}$, which are correlated to each other (and hence not independent). However, it does appear to capture the empirical behavior of sieving, which has been verified by several experimental studies [NV08; MV10; BLS16; ADHK+19]. The uniform heuristic yields classical and quantum algorithms for approximate SVP that are more efficient than all known non-heuristic methods in the respective classical and quantum settings.

Table 6.2 presents the minimal memory complexity that is necessary and sufficient for k-tuple sieving (under the uniform heuristic). Note that the asymptotic regime we considered for Theorem 6.3.2 corresponds exactly to the minimal list size of 3-tuple sieving proven in [HK17].

k	Minimal memory regime	
2	$(4/3)^{n/2+o(n)}$	$\approx 2^{0.2075n}$
3	$(27/16)^{n/4+o(n)}$	$\approx 2^{0.1887n}$
4	$(256/125)^{n/6+o(n)}$	$\approx 2^{0.1724n}$
$k = O(1)$	$(k^{k/(k-1)}/(k+1))^{n/2+o(n)}$	

Table 6.2: Tight memory complexity bounds for k-tuple lattice sieving with constant k under Heuristic 6.4.1, as shown in [NV08] for $k = 2$ and in [HK17] for $k > 2$.

6.4.2 An improved quantum algorithm for SVP

Combining Heuristic 6.4.1 and Theorem 6.3.2 yields a quantum algorithm that heuristically solves SVP in time $2^{0.284551n+o(n)}$ using $(27/16)^{n/4+o(n)} \approx 2^{0.188722n+o(n)}$ classical bits and QCRAM bits, and subexponentially many qubits. This is the fastest known (heuristic) quantum algorithm for SVP when the total memory is limited to $2^{0.188722n+o(n)}$.

Heuristic Claim 6.4.2. *Under Heuristic 6.4.1, there exists a quantum algorithm that solves γ-SVP in dimension n with $\gamma = O(1)$ in time $2^{0.284551n+o(n)}$ with probability $1 - 2^{-\Omega(n)}$. This algorithm uses $(27/16)^{n/4+o(n)}$ classical memory and QCRAM bits, and $2^{o(n)}$ qubits.*

As mentioned in Section 6.1.2 and shown in Table 6.1, under the same heuristic and the same memory complexity (optimal for 3-tuple sieving), our quantum algorithm improves over the prior best quantum algorithm of [CL23].

Compared to the fastest known heuristic quantum algorithm for SVP [BCSS23, Proposition 4], which uses 2-tuple sieving, the gain in memory complexity that we obtain by using 3-tuple sieving still does not fully compensate for the increase in time complexity (despite our quantum speedup): time-memory product $2^{(0.1887+0.2846)n} \approx 2^{0.4733n}$ versus $2^{(0.2075+0.2563)n} \approx 2^{0.4638n}$. (Note that a time-memory product may not be the best measure for comparing these algorithms, as time and memory are not directly interchangeable.) A benefit of our quantum algorithm is that it uses only $2^{o(n)}$ qubits, whereas [BCSS23] uses an exponential number of qubits and QQRAM (i.e., quantum-readable quantum-writable *quantum* memory, which is technologically more demanding than the QCRAM required in our approach) for implementing quantum walks. Considering the class of heuristic quantum algorithms for SVP that use at most $2^{o(n)}$ qubits and no QQRAM, the best known time complexity is $2^{0.2571n}$ [Hei21], achieved using $2^{0.2075n}$ classical memory (and also QCRAM of exponential size).

6.5 Discussion

An important takeaway of this chapter is that an appropriate use of quantum techniques combined with a well-chosen preprocessing step can lead to speedups for a cryptographically relevant problem. This is illustrated by the main quantum algorithm of this chapter, `3List`, which yields a speedup for SVP in the minimal memory regime of 3-tuple lattice sieving. This speedup was obtained without the use of quantum walks, and in particular without QQRAM.

Various ingredients of `3List`, including the use of RPCs and AA, have been used before in quantum k-tuple sieving algorithms. What distinguishes it, and enables the speedup, is first the nested use of AA, and second the way we leverage the preprocessing we do in advance. The core problem solved by `3List` (Problem 6.1.1) naturally lends itself to do two-oracle search in the vein of [KLL15]: the criterion for being a "good" 3-tuple can be split into two criteria, one of which is cheaper to check, so it makes sense to do that first to rule out some 3-tuples more cheaply. Adding locality-sensitive filtering on top (using RPCs) then allows for balancing the costs of different parts of the algorithm by introducing additional optimization parameters.

Similar to prior work, we implement locality-sensitive filtering by preprocessing a classical data structure that stores, for each vector in the input list L, its set of close center points from the RPC. However, we leverage this data structure rather differently: we use it to create superpositions over "close" pairs of vectors in L. More precisely, using the data structure, we can quickly prepare, for a given $\mathbf{x}$, a superposition over all center points $\mathbf{c}$ that are close to $\mathbf{x}$, but also quickly prepare, for each such $\mathbf{c}$, a superposition over all $\mathbf{y} \in L$ that are close to $\mathbf{c}$, and hence relatively close to $\mathbf{x}$ (recall Figure 6.2). Altogether, this enables the more sophisticated nested AA that results in our speedup.

Practical implications. While the speedup from exponent 0.3098 to 0.2846 (an improvement in the second decimal) is relatively large compared to other recent speedups in quantum lattice sieving, it is small from a practical perspective. For cryptographic constructions based on the assumed hardness of SVP, the choice of an appropriately large key size to guarantee a certain level of security usually already accounts for a quadratic quantum speedup over the best known classical attack. Such a quadratic speedup is the best speedup one can hope for just using Grover's algorithm or amplitude amplification (without exploiting further specific

structure of the problem), and is already better than all quantum speedups that we actually know how to achieve.[10] Moreover, our results include factors of the form $2^{o(n)}$ in the runtime, which are insignificant for asymptotic analysis but may be quite large for the dimensions used in practice. Even quadratic speedups may not be noticeable in practice, due to factors hidden in the asymptotics and other reasons such as the overhead of quantum error correction (e.g., see [BMNG+21]).

Further asymptotic improvements and time-memory trade-offs. Can the quantum time complexity of SVP be improved even further using 3-tuple sieving? Surprisingly, `3List` only used basic `AA`, and not the more sophisticated quantum-walk approaches that are good at finding collisions (and which have been used for sieving [CL21; BCSS23]). In fact, the work presented in this chapter started from a quantum-walk variant of `3List`, but optimizing the parameters suggested that the current version (which can be viewed as a quantum walk with vertex size 1) is optimal. This might be due to the fact that there are *many* 3-tuple solutions that need to be found, a setting in which quantum walks may be less helpful (consider that for *element distinctness*, the optimal algorithm uses quantum walks [Amb04], whereas for its many-solution variant, *collision finding*, there is an optimal quantum algorithm that uses only `AA` [BHT98]). It remains open whether there are other ways to further improve 3-tuple sieving by using quantum walks. Note, however, that such an approach would most probably use QQRAM.

In addition, the algorithm and our analysis can readily be adapted to variants of Problem 6.1.1 or Problem 6.1.2, where the number of found tuples differs from the list size m or where different choices for the angles θ, θ' (equivalently, inner products $1/3, 1/\sqrt{3}$) are considered. This could, for instance, be useful for studying time-memory trade-offs, because by allowing more memory than in the minimal memory regime of 3-tuple sieving, we may obtain better runtimes by varying the angles θ, θ' (instead of using the choice corresponding to the minimal memory regime).

[10] In fact, a quadratic quantum speedup for $\{2, 3\}$-tuple sieving is not even feasible: in both settings, a quadratic speedup over the best known classical attack (see Table 6.1) would go below the minimal memory complexity. Note, however, that this comment applies to the approach of tuple sieving where each sieving iteration is required to classically output the full list, and may not apply to other, more inherently quantum approaches (e.g., one could consider keeping the list in a quantum state).

Applications beyond 3-tuple sieving. Another natural question is whether similar techniques can be used for k-tuple sieving with $k \neq 3$. The main strategy and techniques behind `3List` can be naturally extended to $k > 3$, so it seems worthwhile to figure out if this would lead to further speedups for SVP when memory is limited to $o(2^{0.1887n})$. For instance, for $k = 4$, the core problem (i.e., the variant of Problem 6.1.1) would essentially be the task of finding, given a large list L of i.i.d. uniform unit vectors, many $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w} \in L$ such that $\|\mathbf{x} \pm \mathbf{y} \pm \mathbf{z} \pm \mathbf{w}\| \leq 1$ for some choice of sign patterns. By first searching for pairs $\mathbf{x}, \mathbf{y}$ that are somewhat close, followed by a search for a close $\mathbf{z}$, and then for a close $\mathbf{w}$, we naturally obtain a "many-oracle search" setup, where locality-sensitive filtering can be added to in the same manner as done in `3List`.

For $k = 2$, any runtime improvement would replace the fastest known quantum algorithm for SVP, making progress on this front particularly significant. It is not immediately clear how to apply the two-oracle search strategy of `3List` to $k = 2$, as a similar setup would seem to require splitting the main criterion for a "good" 2-tuple into multiple criteria. A potentially more promising direction is to explore additional ways of leveraging preprocessing, for instance by building on the approach presented in this chapter.

6.A Appendix

For completeness, we provide the proofs of Lemma 6.2.6 and Lemma 6.2.7.

Proof of Lemma 6.2.6. Let $\mathbf{x} \in \mathcal{S}^{n-1}$, and define $p := \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x}, \mathbf{c}\rangle \approx_\epsilon \cos(\alpha)]$. By Lemma 6.2.4,

$$\begin{aligned} p &\le \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x}, \mathbf{c}\rangle \ge \cos(\alpha) - \epsilon] \\ &=_n (1 - (\cos(\alpha) - \epsilon)^2)^{n/2} \\ &= (1 - \cos^2(\alpha))^{n/2} \left(1 + \frac{2\epsilon\cos(\alpha) - \epsilon^2}{1 - \cos^2(\alpha)}\right)^{n/2} \\ &\le (1 - \cos^2(\alpha))^{n/2} \exp(O(\epsilon d)) \end{aligned}$$

since $\alpha = \Omega(1)$ implies $1 - \cos^2(\alpha) = \Omega(1)$. As $\epsilon = 1/(\log n)^2$, we obtain $p \le_n p_\alpha$.

Moreover, by the union bound and Lemma 6.2.4, we obtain that for some constant $k > 0$,

$$\begin{aligned} p &\ge \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x}, \mathbf{c}\rangle \ge \cos(\alpha)] - \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x}, \mathbf{c}\rangle \ge \cos(\alpha) + \epsilon] \\ &\ge \frac{1}{n^k}(1 - \cos^2(\alpha))^{n/2} - n^k(1 - (\cos(\alpha) + \epsilon)^2)^{n/2} \\ &= (1 - \cos^2(\alpha))^{n/2} \left(\frac{1}{n^k} - n^k \left(1 - \frac{2\epsilon\cos(\alpha) + \epsilon^2}{1 - \cos^2(\alpha)}\right)^{n/2}\right). \end{aligned}$$

Since $(1 - \frac{2\epsilon\cos(\alpha)+\epsilon^2}{1-\cos^2(\alpha)})^{n/2} \le (1 - \epsilon^2)^{n/2} \le \exp(-\epsilon^2 n/2)$, the choice $\epsilon = 1/(\log n)^2$ yields $p \ge_n p_\alpha$. □

Proof of Lemma 6.2.7. Let $\mathbf{x}, \mathbf{y} \in \mathcal{S}^{n-1}$ be such that $\langle \mathbf{x}, \mathbf{y}\rangle \approx_\epsilon \cos(\theta)$, and define

$$p := \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x}, \mathbf{c}\rangle \approx_\epsilon \cos(\alpha), \langle \mathbf{y}, \mathbf{c}\rangle \approx_\epsilon \cos(\beta)].$$

By Lemma 6.2.5, it suffices to show that $p =_n (1 - \gamma^2)^{n/2}$, where

$$\gamma^2 := \frac{\cos^2(\alpha) + \cos^2(\beta) - 2\cos(\alpha)\cos(\beta)\cos(\theta)}{\sin^2(\theta)}.$$

We proceed by showing that $(1-\widetilde{\gamma}^2)^{n/2} \le_n p \le_n (1-\overline{\gamma}^2)^{n/2}$ for some $\widetilde{\gamma}^2 \le \gamma^2 + O(\epsilon)$ and $\overline{\gamma}^2 \ge \gamma^2 - O(\epsilon)$ that are defined in terms of $(\alpha, \beta, \theta, \epsilon)$, allowing us to prove $p =_n (1 - \gamma^2)^{n/2}$.

More precisely, define $\cos(\phi) := \langle \mathbf{x}, \mathbf{y} \rangle$. Then $\cos(\theta) - \epsilon \leq \cos(\phi) \leq \cos(\theta) + \epsilon$, which implies $-3\epsilon \leq \sin^2(\phi) - \sin^2(\theta) \leq 2\epsilon$. In addition, the assumption $|\alpha - \beta| + \Omega(1) \leq \theta \leq \alpha + \beta - \Omega(1)$ implies $\theta = \Omega(1)$. Using $\varepsilon = o(1)$, it follows that there exists a constant $\kappa > 0$ such that (for sufficiently large n) $\sin^2(\phi) \geq \kappa$. Using standard trigonometric identities, we also obtain

$$\begin{aligned}\sin^2(\theta)(1-\gamma^2) &= \sin^2(\theta) - \cos^2(\alpha) - \cos^2(\beta) + 2\cos(\alpha)\cos(\beta)\cos(\theta) \\ &= 4\sin\left(\tfrac{(\beta+\theta)+\alpha}{2}\right)\sin\left(\tfrac{(\beta+\theta)-\alpha}{2}\right)\sin\left(\tfrac{\alpha+(\beta-\theta)}{2}\right)\sin\left(\tfrac{\alpha-(\beta-\theta)}{2}\right)\end{aligned}$$

and thus the assumption on α, β, θ implies $1 - \gamma^2 \geq \sin^2(\theta)(1-\gamma^2) = \Omega(1)$. In other words, there also exists a constant $\kappa' > 0$ such that $1 - \gamma^2 \geq \kappa'$ (for sufficiently large n).
Now, for the upper bound, note that Lemma 6.2.5 implies

$$p \leq \Pr_{\mathbf{c} \sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x}, \mathbf{c} \rangle \geq \cos(\alpha) - \epsilon, \langle \mathbf{y}, \mathbf{c} \rangle \geq \cos(\beta) - \epsilon] =_n (1 - \overline{\gamma}^2)^{n/2}$$

where

$$\begin{aligned}\overline{\gamma}^2 &:= \frac{(\cos(\alpha)-\epsilon)^2 + (\cos(\beta)-\epsilon)^2 - 2(\cos(\alpha)-\epsilon)(\cos(\beta)-\epsilon)\cos(\phi)}{\sin^2(\phi)} \\ &\geq \frac{\cos^2(\alpha) + \cos^2(\beta) - 2\cos(\alpha)\cos(\beta)\cos(\theta)}{\sin^2(\phi)} \\ &\quad - \frac{2\epsilon(\cos(\alpha) + \cos(\beta) + \cos(\alpha)\cos(\beta))}{\sin^2(\phi)} \\ &\geq \frac{\cos^2(\alpha) + \cos^2(\beta) - 2\cos(\alpha)\cos(\beta)\cos(\theta)}{\sin^2(\phi)} - \frac{6\epsilon}{\sin^2(\phi)} \\ &= \gamma^2 \frac{\sin^2(\theta)}{\sin^2(\phi)} - \frac{6\epsilon}{\sin^2(\phi)} \\ &= \gamma^2 \left(1 - \frac{\sin^2(\phi) - \sin^2(\theta)}{\sin^2(\phi)}\right) - \frac{6\epsilon}{\sin^2(\phi)}.\end{aligned}$$

Since $\gamma^2 \leq 1$, $\sin^2(\phi) - \sin^2(\theta) \leq 2\epsilon$, and $\sin^2(\phi) \geq \kappa$, we obtain $\overline{\gamma}^2 \geq \gamma^2 - \frac{8\epsilon}{\kappa}$. The desired upper bound now follows, because (using $1 - \gamma^2 \geq \kappa'$ and $\epsilon = 1/(\log n)^2$)

$$\begin{aligned}(1-\overline{\gamma}^2)^{n/2} &= (1-\gamma^2)^{n/2}\left(1 + \frac{\gamma^2 - \overline{\gamma}^2}{1-\gamma^2}\right)^{n/2} \\ &\leq (1-\gamma^2)^{n/2}\left(1 + \frac{8\epsilon}{\kappa\kappa'}\right)^{n/2} \\ &\leq_n (1-\gamma^2)^{n/2}.\end{aligned}$$

For the lower bound, note that the union bound implies

$$\begin{aligned}
p \geq & \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x},\mathbf{c}\rangle \geq \cos(\alpha)-\epsilon, \langle \mathbf{y},\mathbf{c}\rangle \geq \cos(\beta)-\epsilon] \\
& - \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x},\mathbf{c}\rangle > \cos(\alpha)+\epsilon, \langle \mathbf{y},\mathbf{c}\rangle \geq \cos(\beta)-\epsilon] \\
& - \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x},\mathbf{c}\rangle \geq \cos(\alpha)-\epsilon, \langle \mathbf{y},\mathbf{c}\rangle > \cos(\beta)+\epsilon] \\
\geq_n & \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x},\mathbf{c}\rangle \geq \cos(\alpha)-\epsilon, \langle \mathbf{y},\mathbf{c}\rangle \geq \cos(\beta)-\epsilon] \\
\geq & \Pr_{\mathbf{c}\sim U(\mathcal{S}^{n-1})}[\langle \mathbf{x},\mathbf{c}\rangle \geq \cos(\alpha), \langle \mathbf{y},\mathbf{c}\rangle \geq \cos(\beta)]
\end{aligned}$$

where the second inequality can be shown using similar methods as how we have shown the upper bound, using that $\epsilon = 1/(\log n)^2$. By Lemma 6.2.5, we thus obtain

$$p \geq_n (1-\widetilde{\gamma}^2)^{n/2} = (1-\gamma^2)^{n/2}\left(1-\frac{\widetilde{\gamma}^2-\gamma^2}{1-\gamma^2}\right)^{n/2}$$

where

$$\begin{aligned}
\widetilde{\gamma}^2 &:= \frac{\cos^2(\alpha)+\cos^2(\beta)-2\cos(\alpha)\cos(\beta)\cos(\phi)}{\sin^2(\phi)} \\
&= \frac{\cos^2(\alpha)+\cos^2(\beta)-2\cos(\alpha)\cos(\beta)(\cos(\theta)+\cos(\phi)-\cos(\theta))}{\sin^2(\theta)}\frac{\sin^2(\theta)}{\sin^2(\phi)} \\
&= \left(\gamma^2 - \frac{2\cos(\alpha)\cos(\beta)(\cos(\phi)-\cos(\theta))}{\sin^2(\theta)}\right)\frac{\sin^2(\theta)}{\sin^2(\phi)}.
\end{aligned}$$

If $\sin^2(\theta) \geq \sin^2(\phi)$ (meaning $\theta \geq \phi$ and $\cos(\theta) \leq \cos(\phi)$), then $\widetilde{\gamma}^2 \leq \gamma^2\frac{\sin^2(\theta)}{\sin^2(\phi)}$. Otherwise,

$$\widetilde{\gamma}^2 = \left(\gamma^2 + \frac{2\cos(\alpha)\cos(\beta)(\cos(\theta)-\cos(\phi))}{\sin^2(\theta)}\right)\frac{\sin^2(\theta)}{\sin^2(\phi)} \leq \gamma^2\frac{\sin^2(\theta)}{\sin^2(\phi)} + \frac{2\epsilon}{\sin^2(\phi)}.$$

In either case, we have $\sin^2(\theta)-\sin^2(\phi) \leq 3\epsilon$, so $\gamma^2 \leq 1$ and $\sin^2(\phi) \geq \kappa$ imply

$$\gamma^2\frac{\sin^2(\theta)}{\sin^2(\phi)} = \gamma^2 + \gamma^2\frac{\sin^2(\theta)-\sin^2(\phi)}{\sin^2(\phi)} \leq \gamma^2 + \frac{3\epsilon}{\kappa}.$$

In other words, we have shown $\widetilde{\gamma}^2 - \gamma^2 \leq \frac{5\epsilon}{\kappa}$. Hence,

$$p \geq_n (1-\gamma^2)^{n/2}\left(1-\frac{\widetilde{\gamma}^2-\gamma^2}{1-\gamma^2}\right)^{n/2} \geq (1-\gamma^2)^{n/2}\left(1-\frac{5\epsilon}{\kappa\kappa'}\right)^{n/2} \geq_n (1-\gamma^2)^{n/2}$$

as desired. □

Chapter 7

Quantum code sieving and its limitations in information-set decoding

Like in the previous chapter, we address the potential of quantum algorithms for speeding up sieving-based attacks. However, instead of working with lattices in $\mathbb{R}^n$, the mathematical objects of study are linear subspaces of the finite vector space $\mathbb{F}_2^n$, also known as binary linear codes. These objects form the foundation of code-based cryptography. Information-set decoding (ISD) algorithms are an important class of algorithms for attacking code-based cryptography. In this chapter, we analyze quantum analogs of a recently introduced sieving-based ISD algorithm. Specifically, we present the first quantum algorithms for the code-sieving subroutine of this ISD algorithm, and numerically analyze the obtained quantum speedups. We also consider quantum code sieving as a subroutine of ISD in the quantum setting, and show that the resulting quantum sieving-based ISD algorithm performs no better than existing quantum ISD algorithms. By highlighting inherent limitations of quantum analogs of this sieving-based ISD approach, this chapter effectively "closes" certain research paths within quantum cryptanalysis of code-based cryptography.

This chapter is based on [EEL25].

The numerical results presented in this chapter were generated using code available at the repository `https://github.com/lynnengelberts/quantum-sieving-for-codes-public`. References of the form [`.py`] link to the relevant code in the online version of this thesis.

Contents of this chapter

7.1 Overview

7.1.1 Problem setting

A fundamental problem in code-based cryptography is the task of decoding a linear code: given a k-dimensional linear subspace $\mathcal{C} \subseteq \mathbb{F}_2^n$ and $\tilde{\mathbf{c}} \in \mathbb{F}_2^n$, find $\mathbf{e} \in \mathbb{F}_2^n$ of Hamming weight $|\mathbf{e}| \leq w$ such that $\tilde{\mathbf{c}} = \mathbf{c} + \mathbf{e}$ for some $\mathbf{c} \in \mathcal{C}$. While this decoding problem is NP-hard in the worst case [BMT78], for cryptographic purposes we are more interested in its average-case complexity. For fixed n and k, the weight w can be chosen to guarantee that there is exactly one solution in expectation, giving the so-called unique decoding regime. In this regime, the decoding problem is believed to be hard for a random instance of the problem, with all known algorithms requiring exponential time. It is indeed this regime that is of particular interest for cryptography.

The best known generic decoding algorithms[1] belong to the framework of *information-set decoding (ISD)*. This class of algorithms originates from the work of Prange [Pra62], and further improvements were made using various techniques, including the meet-in-the-middle approach and its generalizations [Ste88; Dum91; SS81], representation techniques [MMT11; BJMM12], near-neighbor search techniques [MO15; Car20], and their combinations. Some approaches have been adapted to the quantum setting, resulting in *quantum ISD* algorithms [Ber10; KT17; Kir18].

The general strategy behind ISD is to repeatedly solve an easier, but closely related, decoding problem until this allows solving the original problem. The kind of subroutine used for solving these easier decoding problems is what distinguishes most ISD algorithms and their performance. Recently, a new ISD algorithm was proposed in [GJN23] using a "sieving subroutine", which was further improved and generalized in [DEEK24].

While such sieving algorithms are new in code-based cryptanalysis, we have already seen in Chapter 6 that they are abundant in lattice-based cryptanalysis, and belong to the state-of-the-art of classical *and* quantum algorithms for the Shortest Vector Problem. The most recent improvement in the classical runtime of (heuristic) lattice sieving was achieved by introducing the notion of *random product codes* (recall Section 6.2.4) to do "locality-sensitive filtering" in the Eu-

[1] For certain parameter regimes, the so-called statistical decoding attacks (also known as dual attacks in the lattice literature) are more efficient. As we are not particularly interested in a specific parameter regime, we will only consider attacks in the ISD framework.

clidean metric [BDGL16]. The only further asymptotic runtime improvements (not counting subexponential factor improvements) have come from resorting to quantum algorithms. The first quantum speedup for 2-tuple lattice sieving was obtained using a Grover-based algorithm [Laa15], which was improved using quantum-walk techniques in [CL21]. This quantum-walk algorithm was further improved in [BCSS23] using reusable-walk techniques.

Soon after the introduction of the sieving-based ISD algorithm from [GJN23], the aforementioned techniques from classical lattice sieving have been adapted to the Hamming metric in [DEEK24]. They obtained an improved sieving-based ISD algorithm with asymptotic runtime close to that of the best known classical ISD algorithms (such as [BJMM12; BM18]), and with an improved time-memory trade-off over previously known techniques. Given the overlap between code sieving and lattice sieving, the various quantum speedups in lattice sieving raise the natural question of whether similar quantum techniques can lead to improved quantum attacks on code-based cryptography, which is the focus of this chapter. More precisely, we will address the following questions:

(1) *To what extent can the code-sieving subroutine of [DEEK24] be sped up using quantum algorithms?*

(2) *Does quantum code sieving lead to a faster quantum ISD algorithm?*

7.1.2 Contributions

To answer the first question, we present two quantum algorithms for the core problem solved by the classical sieving subroutine from [DEEK24]. Inspired by the quantum state-of-the-art in heuristic lattice sieving [Laa15; CL21; BCSS23], our algorithms are based on Grover's search algorithm and on a quantum-walk-based approach in the spirit of [CL21], where we incorporate locality-sensitive filtering inside the quantum walk to improve over the Grover approach. Our main contribution here is to show how to apply and analyze these algorithms in the setting of codes.

We provide asymptotic formulas for the time and memory complexities of these quantum algorithms in terms of the parameters introduced by the different approaches, which we optimize numerically. Our numerical results show that we obtain quantum speedups over the classical subroutine from [DEEK24], and illustrate to what extent our quantum-walk algorithm improves over the Grover

approach. In addition, we show that the obtained quantum speedups align with the speedups observed in lattice sieving (see Table 7.1).

Finally, we present and analyze a natural quantum analog of the sieving-based ISD algorithm from [DEEK24] that instantiates the code-sieving subroutine with our best quantum algorithm. The resulting quantum ISD algorithm is even outperformed by the first quantum algorithm [Ber10] proposed for the decoding problem, yielding a negative answer to the second question listed above (at least, within the current sieving-based framework). The main limiting factor can be attributed to a constraint imposed by the sieving approach, which results in a lower bound on the runtime of the full quantum ISD algorithm that is too high. We also highlight why this issue does not appear in the classical setting, where sieving-based ISD outperforms the classical counterpart of [Ber10]. Moreover, we explain why two natural attempts to adapt the sieving-based ISD algorithm do not suffice to overcome the found limitations. Altogether, this suggests that other approaches to (sieving-based) ISD should be considered if it wants to compete with the best-performing quantum algorithms for the decoding problem.

The numerical results in this chapter were generated using Python code available at `https://github.com/lynnengelberts/quantum-sieving-for-codes-public`, and we refer to the relevant parts of the code using the symbol [`.py`].

7.1.3 Technical overview

Quantum algorithms for code sieving. The code-sieving algorithm presented in [DEEK24] serves as a subroutine within their ISD algorithm. This sieving subroutine repeatedly solves a so-called Near-Neighbor Search (NNS) problem, which is achieved using a subroutine that we call `FindSolutions`. The task of `FindSolutions` is to find all pairs $(\mathbf{x}, \mathbf{y})$ of vectors in a given list whose sum $\mathbf{x} + \mathbf{y}$ has a certain (Hamming) weight. Being a rather plain search problem, this naturally suggests the potential of using quantum algorithms. Indeed, our first quantum algorithm is a straightforward application of Grover's algorithm, which will be used as a baseline for comparison.

From an abstract perspective, the problem solved by `FindSolutions` is actually quite similar to the problem that is sped up by the quantum algorithms for lattice sieving. Indeed, our Grover variant does precisely what was considered in [Laa15]. Moreover, this suggests that we may be able to obtain an improved quantum algorithm for `FindSolutions` by using quantum walks in a similar manner as [CL21].

The main idea in [CL21] that enables the speedup over Grover's algorithm (in the lattice setting) is to add a layer of locality-sensitive filtering (LSF) to the quantum walk, to facilitate the search for "good" pairs by localizing the search. Although their analysis does not automatically transfer to the code setting, we show how to adapt it, thereby benefiting from this layer of LSF in a way that improves over the Grover approach.

By integrating these quantum algorithms for `FindSolutions`, we obtain quantum algorithms for NNS and code sieving, whose complexities we can analyze through numerical optimization of the parameters.

Application to quantum ISD. The main application of the code-sieving algorithm introduced in [GJN23] and improved by [DEEK24] was to use it as a subroutine of an ISD algorithm for decoding a random linear code in the unique decoding regime. For all ISD algorithms, there is a natural quantum analog obtained by applying amplitude amplification and, if applicable, replacing its classical subroutine by a quantum one. Using our best quantum algorithm for code sieving as a quantum subroutine, we therefore obtain a quantum analog of the sieving-based ISD algorithm from [DEEK24]. By evaluating its complexity numerically, we can then compare it with the quantum state-of-the-art for the decoding problem.

See Figure 7.1 for a (high-level) visualization of how the different algorithms relate to each other.

7.1.4 Organization

Section 7.2 provides chapter-specific background on the geometric properties of $\mathbb{F}_2^n$ equipped with the Hamming metric and on binary linear codes. In Section 7.3, we present the code-sieving approach from [DEEK24], including their locality-sensitive filtering techniques for $\mathbb{F}_2^n$. It also explains how the complexity of code sieving depends on the complexity of `FindSolutions`. In Section 7.4, we present our quantum algorithms for `FindSolutions` and analyze their asymptotic complexity. Section 7.5 then presents the numerical results on the asymptotic runtime and memory of the resulting quantum algorithms for the problem solved by code sieving. In Section 7.6, we adapt the classical sieving-based ISD framework from [DEEK24] to the quantum setting, and explain its limitations. Finally, in Section 7.7 we briefly discuss how to interpret the cryptanalytic implications of the results in this chapter.

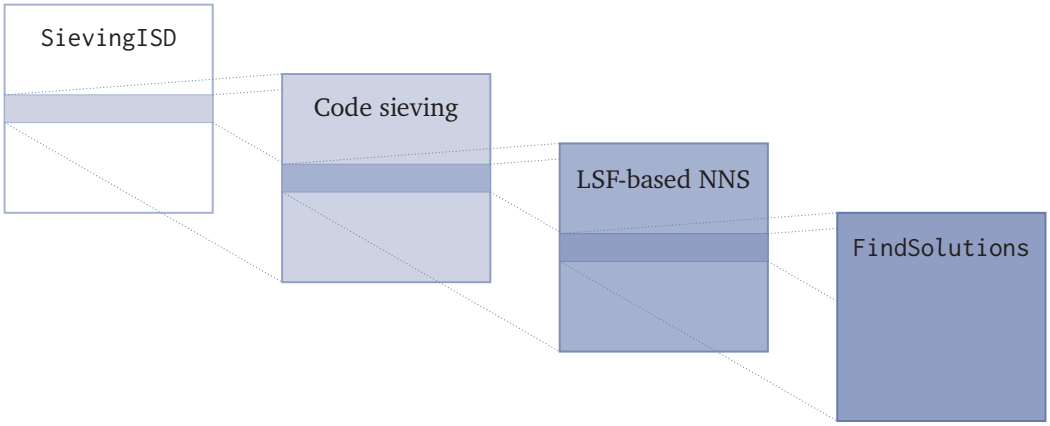


Figure 7.1: Illustration of the different layers of subroutines used by `SievingISD`. This ISD algorithm repeatedly invokes a code-sieving subroutine until it succeeds to solve the main decoding problem at hand. The code-sieving subroutine repeatedly invokes an LSF-based algorithm for solving NNS, which in turn relies on a subroutine that we call `FindSolutions`. In Section 7.4, we present quantum analogs of `FindSolutions`, resulting in quantum algorithms for NNS and for code sieving. By instantiating `SievingISD` with this quantum code-sieving subroutine, and putting amplitude amplification on top, we obtain a quantum sieving-based ISD algorithm that we analyze in Section 7.6.

7.2 Preliminaries

Notation. In this chapter, we work over the binary finite field $\mathbb{F}_2$. We write $\mathbf{x} + \mathbf{y}$ for the bitwise "XOR" operation between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$, and $\mathbf{x} \wedge \mathbf{y}$ for the bitwise "AND" operation.

7.2.1 Hamming space

For $n \in \mathbb{N}$, we consider the vector space $\mathbb{F}_2^n$ endowed with the Hamming metric. We denote the *Hamming weight* of a vector $\mathbf{x} \in \mathbb{F}_2^n$ by $|\mathbf{x}|$, that is,

$$|\mathbf{x}| := |\{i \in [n] : x_i \neq 0\}|.$$

We will repeatedly use the identity $|\mathbf{x} + \mathbf{y}| = |\mathbf{x}| + |\mathbf{y}| - 2|\mathbf{x} \wedge \mathbf{y}|$.

The code-sieving algorithm from [DEEK24] has a geometric interpretation using the following notions of (Hamming) sphere, spherical cap, and spherical wedge. This mirrors the Euclidean-metric analogs in lattice sieving from Section 6.2.3.

Definition 7.2.1 (Hamming sphere, cap, and wedge)**.** For all integers $0 \le v \le n$, we define the *weight-v Hamming sphere*

$$\mathcal{S}_v^n := \{\mathbf{c} \in \mathbb{F}_2^n : |\mathbf{c}| = v\}.$$

For all integers $0 \le \alpha \le v \le n$ and all $\mathbf{x} \in \mathbb{F}_2^n$ satisfying $\alpha \le |\mathbf{x}|$, we define the *spherical cap*

$$\mathcal{C}_v^n(\alpha \mid \mathbf{x}) := \{\mathbf{c} \in \mathcal{S}_v^n : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}.$$

For all integers $0 \le \alpha \le v \le n$ and all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ satisfying $\alpha \le \min\{|\mathbf{x}|, |\mathbf{y}|\}$, we define the (symmetric) *spherical wedge*

$$\mathcal{W}_v^n(\alpha \mid \mathbf{x}, \mathbf{y}) := \mathcal{C}_v^n(\alpha \mid \mathbf{x}) \cap \mathcal{C}_v^n(\alpha \mid \mathbf{y}) = \{\mathbf{c} \in \mathcal{S}_v^n : |\mathbf{x} \wedge \mathbf{c}| = \alpha, |\mathbf{y} \wedge \mathbf{c}| = \alpha\}.$$

The sizes or "surface areas" of these sets can be calculated as follows. In particular, the surface areas of a cap (resp. wedge) depends only on the *weight* of the vector $\mathbf{x}$ (resp. the vectors $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{x} \wedge \mathbf{y}$).

Lemma 7.2.2 (Surface areas of the Hamming sphere, cap, and wedge [DEEK24])**.** *The surface area of a Hamming sphere is*

$$|\mathcal{S}_v^n| = \binom{n}{v}.$$

The surface area of a spherical cap is

$$|\mathcal{C}_v^n(\alpha \mid \mathbf{x})| = \binom{|\mathbf{x}|}{\alpha}\binom{n - |\mathbf{x}|}{v - \alpha}.$$

The surface area of a spherical wedge is (see Figure 7.2)

$$|\mathcal{W}_v^n(\alpha \mid \mathbf{x}, \mathbf{y})| = \sum_{u=\max\{0,2\alpha-v\}}^{\min\{\alpha,w^*\}} \binom{w^*}{u}\binom{|\mathbf{x}| - w^*}{\alpha - u}\binom{|\mathbf{y}| - w^*}{\alpha - u}\binom{n - |\mathbf{x}| - |\mathbf{y}| + w^*}{v - 2\alpha + u}$$

where $w^* := |\mathbf{x} \wedge \mathbf{y}|$.

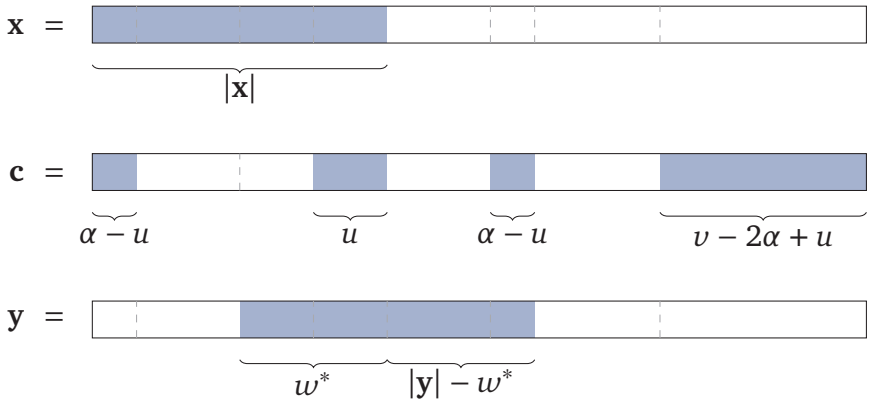


Figure 7.2: Illustration of the overlaps between vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$ and $\mathbf{c} \in \mathcal{S}_\upsilon^n$ satisfying $|\mathbf{x} \wedge \mathbf{y}| = w^*$, $|\mathbf{x} \wedge \mathbf{c}| = |\mathbf{y} \wedge \mathbf{c}| = \alpha$, and $|\mathbf{x} \wedge \mathbf{y} \wedge \mathbf{c}| = u$. The colored part corresponds to the coordinates of the binary string that are 1.

Proof. The size of the Hamming sphere and spherical cap follow immediately from their definition. For the spherical wedge, we adapt the argument from [DEEK24] to general $|\mathbf{x}|, |\mathbf{y}|, |\mathbf{x} \wedge \mathbf{y}|$ (as they only considered $|\mathbf{x}| = |\mathbf{y}| = |\mathbf{x} + \mathbf{y}| = w$). We want to count the number of $\mathbf{c} \in \mathcal{S}_\upsilon^n$ that satisfy $|\mathbf{x} \wedge \mathbf{c}| = |\mathbf{y} \wedge \mathbf{c}| = \alpha$. Such a $\mathbf{c}$ must satisfy $\max\{0, 2\alpha - \upsilon\} \leq |\mathbf{x} \wedge \mathbf{y} \wedge \mathbf{c}| \leq \min\{\alpha, w^*\}$, where the inequality $2\alpha - \upsilon \leq |\mathbf{x} \wedge \mathbf{y} \wedge \mathbf{c}|$ can be seen from Figure 7.2. In particular, we have the partition

$$\mathcal{W}_\upsilon^n(\alpha \mid \mathbf{x}, \mathbf{y}) = \bigsqcup_{u=\max\{0,2\alpha-\upsilon\}}^{\min\{\alpha,w^*\}} W_u$$

where $W_u := \{\mathbf{c} \in \mathcal{S}_\upsilon^n : |\mathbf{x} \wedge \mathbf{c}| = |\mathbf{y} \wedge \mathbf{c}| = \alpha, |\mathbf{x} \wedge \mathbf{y} \wedge \mathbf{c}| = u\}$. The claim then follows from the observation that $|W_u| = \binom{w^*}{u}\binom{|\mathbf{x}|-w^*}{\alpha-u}\binom{|\mathbf{y}|-w^*}{\alpha-u}\binom{n-|\mathbf{x}|-|\mathbf{y}|+w^*}{\upsilon-2\alpha+u}$, which can be verified using Figure 7.2. □

7.2.2 Binary linear codes

The problems and algorithms in this chapter consider binary linear codes. Recall that we are using column notation for vectors throughout this thesis.

> **Definition 7.2.3** (Binary linear code)**.** An $[n, k]$ binary linear code $\mathcal{C}$ is defined as a linear subspace of size 2^k of $\mathbb{F}_2^n$. Elements of $\mathcal{C}$ are called *codewords*.

An $[n, k]$ binary linear code $\mathcal{C}$ can be represented by a *generator matrix*: a matrix $\mathbf{G} \in \mathbb{F}_2^{k\times n}$ of rank k such that

$$\mathcal{C} = \mathcal{C}(\mathbf{G}) \coloneqq \{\mathbf{G}^\top \mathbf{m} \colon \mathbf{m} \in \mathbb{F}_2^k\}.$$

It can also be represented by a *parity-check matrix*: a matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ of rank $n - k$ such that

$$\mathcal{C} = \mathcal{C}(\mathbf{H}) \coloneqq \{\mathbf{c} \in \mathbb{F}_2^n \colon \mathbf{Hc} = \mathbf{0}\}.$$

Given a generator matrix $\mathbf{G}$ of $\mathcal{C}$, we can compute a parity-check matrix $\mathbf{H}$ satisfying $\mathbf{GH}^\top = \mathbf{0}$ in time $\mathrm{poly}(n)$ using Gaussian elimination, and conversely, we can compute $\mathbf{G}$ from $\mathbf{H}$. For example, see [Deb23].

We are primarily interested in *random binary linear codes* (sometimes simply referred to as *random codes*), which are codes obtained by sampling a parity-check matrix uniformly at random from $\mathbb{F}_2^{(n-k)\times n}$. These codes can be sampled in time polynomial in n: just flip $(n - k)n$ uniformly random bits.

Decoding problems

The central problem in code-based cryptography is often referred to as the *Decoding Problem*. There are different formulations of this problem, such as the following.

> **Problem 7.2.4** (Syndrome Decoding Problem, $\mathrm{SDP}(n, k, w)$)**.** Given a parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$ of an $[n, k]$ binary linear code $\mathcal{C}$, $w \in \mathbb{N}$, and $\mathbf{s} \in \mathbb{F}_2^{n-k}$, find $\mathbf{e} \in \mathbb{F}_2^n$ of weight $|\mathbf{e}| = w$ satisfying $\mathbf{He} = \mathbf{s}$.

This problem seems different from the decoding problem mentioned in the introduction, but they are closely related: the problem in the introduction provides the generator-matrix perspective, whereas SDP considers the parity-check matrix perspective. (When $\tilde{\mathbf{c}} = \mathbf{c} + \mathbf{e}$ for $\mathbf{c} \in \mathcal{C}$, we obtain $\mathbf{H}\tilde{\mathbf{c}} = \mathbf{0} + \mathbf{He} = \mathbf{He}$, so the problem in the introduction can be formulated as an instance of the Syndrome Decoding Problem with $\mathbf{s} = \mathbf{H}\tilde{\mathbf{c}}$.)

In this chapter, we follow [DEEK24] and consider the following reformulation of Problem 7.2.4.

> **Problem 7.2.5** (Codeword-Finding Problem, CFP(n, k, w))**.** Given a parity-check matrix of an $[n, k]$ binary linear code $\mathcal{C}$ and $w \in \mathbb{N}$, find a codeword $\mathbf{c} \in \mathcal{C}$ of weight $|\mathbf{c}| = w$.

As explained in [DEEK24], SDP(n, k, w) and CFP(n, k, w) are essentially equivalent. Namely a solution $\mathbf{c}$ to Problem 7.2.5 satisfies $\mathbf{Hc} = \mathbf{0}$, so we can solve an instance $\mathbf{H}$ of CFP(n, k, w) by applying an algorithm for SDP(n, k, w) to $\mathbf{H}$ and $\mathbf{s} = \mathbf{0}$. Conversely, given an instance $(\mathbf{H}, \mathbf{s})$ of SDP(n, k, w), let $\mathbf{H}' = [\mathbf{H} \mid \mathbf{s}]$. Then every weight-$w$ vector $\mathbf{e} \in \mathbb{F}_2^n$ satisfying $\mathbf{He} = \mathbf{s}$ can be found as a solution $\mathbf{e}' = (\mathbf{e}\ ;\ 1)$ to the instance $\mathbf{H}'$ of CFP$(n + 1, k + 1, w + 1)$.

7.3 Code sieving using locality-sensitive filtering

The quantum algorithms for code sieving that we present in this chapter are based on the code-sieving framework introduced in [GJN23] and further improved and generalized in [DEEK24] using locality-sensitive filtering techniques. In this section, we present the code-sieving algorithm from [DEEK24] and its complexity (Theorem 7.3.10).

The main computational problem addressed by code sieving is the following variant of Problem 7.2.5, where the goal is to find $N \geq 1$ codewords instead of one.

> **Problem 7.3.1** (Codeword-Finding Problem, CFP(n, k, w, N))**.** Given a parity-check matrix of an $[n, k]$ binary linear code $\mathcal{C}$ and $w \in \mathbb{N}$, find N codewords $\mathbf{x}_\mathcal{C} \in \mathcal{C}$ of weight $|\mathbf{x}_\mathcal{C}| = w$.

We focus on algorithms that solve a *random* instance of Problem 7.3.1, given by a random $[n, k]$ binary linear code $\mathcal{C}$. For a random code $\mathcal{C}$, the expected number of codewords in $\mathcal{C}$ of weight w is $\binom{n}{w}/2^{n-k}$ (for instance, see [Deb23, Chapter 2]). We therefore only consider algorithms for CFP(n, k, w, N) when the parameters satisfy $N \leq \binom{n}{w}/2^{n-k}$.

In Section 7.6, we will explain how an algorithm for Problem 7.3.1 can be used to solve the central problems in code-based cryptography.

7.3.1 Framework for code sieving

The basic idea of code sieving [GJN23; DEEK24] for solving $\mathrm{CFP}(n, k, w, N)$ is to start from a large list L_0 of random vectors of Hamming weight w and then iteratively add code constraints to obtain enough codewords of Hamming weight w. These iteratively added code constraints define a *tower of codes*: a collection

$$\mathbb{F}_2^n = \mathcal{C}_0 \supseteq \mathcal{C}_1 \supseteq \cdots \supseteq \mathcal{C}_{n-k} = \mathcal{C}$$

of codes with dimension decrements of 1, starting with the "initial" code $\mathbb{F}_2^n$ of dimension n and ending with the input code $\mathcal{C}$ of dimension k.

Each iteration $i \geq 1$ of the sieving algorithm uses the current list $L_{i-1} \subseteq \mathcal{C}_{i-1}$ to create a new list L_i of vectors in the next code $\mathcal{C}_i$. When L_{i-1} consists of random elements from $\mathcal{C}_{i-1}$, only half of its elements are expected to lie in $\mathcal{C}_i$ (because going from $\mathcal{C}_{i-1}$ to $\mathcal{C}_i$ adds one code constraint). To avoid that the list sizes are halved in each iteration, the list L_i is formed by *combining* elements from the previous list L_{i-1}. Specifically, the algorithm first constructs a list of *near-neighbors* in L_{i-1} (pairs $\mathbf{x}, \mathbf{y}$ at small Hamming distance w):

$$\{(\mathbf{x}, \mathbf{y}) \in L_{i-1}^2 \colon |\mathbf{x} + \mathbf{y}| = w\}.$$

It then searches among these near-neighbors for pairs $(\mathbf{x}, \mathbf{y})$ satisfying $\mathbf{x} + \mathbf{y} \in \mathcal{C}_i$, and adds $\mathbf{x} + \mathbf{y}$ to L_i.

The sieving-based algorithms [GJN23; DEEK24] therefore use an oracle $\mathcal{A}_{\mathrm{NNS}}$ that solves a Near-Neighbor Search problem (NNS) in $\mathbb{F}_2^n$ endowed with the Hamming metric. We define this problem as follows, as in [DEEK24, Definition 3.1]. Recall that $L \sim U(\mathcal{S}_w^n, N)$ denotes sampling N i.i.d. uniform elements from $\mathcal{S}_w^n$.

> **Problem 7.3.2** (Near-Neighbor Search, $\mathrm{NNS}(n, w, N)$)**.** Let $n, w \in \mathbb{N}$ with $w \leq n$. Given $L \sim U(\mathcal{S}_w^n, N)$, find a $1 - o(1)$ fraction of all pairs of distinct $\mathbf{x}, \mathbf{y} \in L$ with $|\mathbf{x} + \mathbf{y}| = w$. We will refer to such a pair $(\mathbf{x}, \mathbf{y})$ as a *solution pair*.

Algorithm 7.1 presents the overall sieving framework for solving $\mathrm{CFP}(n, k, w, N)$ (Problem 7.3.1) in more detail. Note that the asymptotic time complexity of this sieving algorithm is determined by the cost of one iteration, which in turn is dominated by the cost of solving $\mathrm{NNS}(n, w, N)$ (Problem 7.3.2). We therefore state the complexities of Algorithm 7.1 after presenting the state-of-the-art algorithm from [DEEK24] for $\mathrm{NNS}(n, w, N)$.

Algorithm 7.1: Code sieving using an NNS-oracle

Input: $[n, k]$ binary linear code $\mathcal{C} \subseteq \mathbb{F}_2^n$
Weight w
Output size N
Oracle $\mathcal{A}_{\mathrm{NNS}}$ for $\mathrm{NNS}(n, w, N)$

Output: $L \subseteq \mathcal{C} \cap \mathcal{S}_w^n$ of size N

1: Choose a tower of codes $\{\mathbb{F}_2^n = \mathcal{C}_0, \ldots, \mathcal{C}_{n-k} = \mathcal{C}\}$
2: Initialize a list L_0 of N independent samples from $U(\mathcal{S}_w^n)$
3: **for** $i = 1$ to $n - k$ **do** *// Sieving*
4: Compute $L_i := \{\mathbf{x} + \mathbf{y} : \mathbf{x}, \mathbf{y} \in L_{i-1} \text{ with } |\mathbf{x} + \mathbf{y}| = w\} \cap \mathcal{C}_i$ using $\mathcal{A}_{\mathrm{NNS}}$
5: Discard some elements if $|L_i| > N$
6: **return** L_{n-k}

Remark 7.3.3 (Difference with lattice sieving). Algorithm 7.1 preserves the Hamming weight of the vectors throughout the iterations, while progressively adding code constraints. Instead, lattice sieving starts with lattice vectors of large Euclidean norm, and iteratively combines them to produce shorter vectors in the same lattice. A second difference is that lattice sieving can be viewed as solving the target problem (SVP, Problem 2.3.8) directly: it produces short lattice vectors, which is the primary goal. In contrast, code sieving is intended to serve as a subroutine within an ISD algorithm, and is invoked many times during the execution of ISD (see Section 7.6).

Besides the upper bound on the output size N that is imposed by $\mathrm{CFP}(n, k, w, N)$ itself, the formulation of Algorithm 7.1 also puts a lower bound on N for the algorithm to succeed. Namely, Lemma 7.3.4 gives a lower bound on the list size N to ensure that the expected number of solution pairs in a random instance of $\mathrm{NNS}(n, w, N)$ is at least N. Consequently, if N is too small, then the expected number of near-neighbors found by the oracle $\mathcal{A}_{\mathrm{NNS}}$ in the first iteration of Algorithm 7.1 is already significantly smaller than N, so it is unlikely that the algorithm outputs N elements.

Lemma 7.3.4 (Expected number of solution pairs for $\mathrm{NNS}(n, w, N)$ [DEEK24])**.** *Let $w \in [n]$ and $N \in \mathbb{N}$. For $L \sim U(\mathcal{S}_w^n, N)$, the expected number of pairs of distinct $\mathbf{x}, \mathbf{y} \in L$ with $|\mathbf{x}+\mathbf{y}| = w$ is*

$$\binom{N}{2} \frac{\binom{w}{w/2}\binom{n-w}{w/2}}{\binom{n}{w}}.$$

In particular, this expected number is at least N if and only if $N \geq 2\frac{\binom{n}{w}}{\binom{w}{w/2}\binom{n-w}{w/2}} + 1$.

Proof. For all $\mathbf{x}, \mathbf{y} \in \mathcal{S}_w^n$, we have $|\mathbf{x}+\mathbf{y}| = w$ if and only if $|\mathbf{x} \wedge \mathbf{y}| = w/2$. Therefore, for fixed $\mathbf{x} \in \mathcal{S}_w^n$, the probability that $\mathbf{y} \sim U(\mathcal{S}_w^n)$ satisfies $|\mathbf{x} \wedge \mathbf{y}| = w/2$ is $\binom{w}{w/2}\binom{n-w}{w/2}/\binom{n}{w}$. If $L \sim U(\mathcal{S}_w^n, N)$, then the expected number of pairs of distinct $\mathbf{x}, \mathbf{y} \in L$ satisfying $|\mathbf{x}+\mathbf{y}| = w$ is thus $\binom{N}{2}\binom{w}{w/2}\binom{n-w}{w/2}/\binom{n}{w}$. The last part follows immediately. □

Among the near-neighbors $(\mathbf{x}, \mathbf{y})$ found by the oracle $\mathcal{A}_{\mathrm{NNS}}$ during an iteration i of Algorithm 7.1, only those satisfying $\mathbf{x}+\mathbf{y} \in \mathcal{C}_i$ are considered for the new list L_i. Since two subsequent codes $\mathcal{C}_{i-1}, \mathcal{C}_i$ differ by one dimension, we may expect that roughly half of the found pairs are discarded. More precisely, if $L_{i-1} \sim U(\mathcal{S}_w^n, N)$ and $N \geq 4\binom{n}{w}/\left(\binom{w}{w/2}\binom{n-w}{w/2}\right) + 1$, then Lemma 7.3.4 implies that the expected number of pairs $\mathbf{x}, \mathbf{y} \in L_{i-1}$ satisfying $\mathbf{x}+\mathbf{y} \in \mathcal{S}_w^n \cap \mathcal{C}_i$ (that is, the expected size of the next list L_i) is at least N.

The authors of [DEEK24] heuristically assume that this list size N is maintained through *all* iterations if $N = \Omega\left(\binom{n}{w}/\left(\binom{w}{w/2}\binom{n-w}{w/2}\right)\right)$ and $N \leq \binom{n}{w}/2^{n-k}$, allowing them to conclude that Algorithm 7.1 outputs a list of N elements in $\mathcal{C} \cap \mathcal{S}_w^n$, thereby solving $\mathrm{CFP}(n, k, w, N)$. Specifically, their analysis relies on the following heuristic.

> **Heuristic 7.3.5** (Binary-sieve heuristic)**.** *In each iteration of Algorithm 7.1, every element in the current list behaves as if it is an i.i.d. uniform sample from $\mathcal{S}_w^n$ in the sense that, even if some correlations are present, they affect the time and memory complexity of the algorithm by at most a factor $2^{o(n)}$. Moreover, for all $\mathbf{x} \in \mathcal{C} \cap \mathcal{S}_w^n$, the probability that $\mathbf{x}$ is part of the output is at least $\Pr_{L \sim U(\mathcal{C} \cap \mathcal{S}_w^n, N)}[\mathbf{x} \in L] 2^{-o(n)}$.*

A more precise formulation and supporting experiments are given in [DEEK24]. Note the similarity with the use of heuristics in lattice sieving (Heuristic 6.4.1).

7.3.2 Solving NNS using locality-sensitive filtering

The algorithms for solving NNS in the Hamming metric proposed in [Car20; GJN23; DEEK24] can be formulated using the locality-sensitive filtering (LSF) framework in the Hamming metric. We start by recalling this approach.

The idea of locality-sensitive filtering in $\mathbb{F}_2^n$ is motivated by the fact that vectors $\mathbf{x}, \mathbf{y}$ at Hamming distance w (near-neighbors) can be found more efficiently if we restrict our search to *local regions* of $\mathbb{F}_2^n$. The algorithm proceeds as follows. It starts with covering the space $\mathbb{F}_2^n$ with (potentially overlapping) regions, each region corresponding to a certain vector $\mathbf{c} \in C$, for some $C \subseteq \mathbb{F}_2^n$.[2] Each vector $\mathbf{x}$ from the input list L is then *filtered* according to the centers, namely, it is inserted into the *bucket* of a center $\mathbf{c}$ if and only if $|\mathbf{x} \wedge \mathbf{c}| = \alpha$ for a fixed parameter α.[3] We call this the *bucketing phase*. Upon successful completion, the algorithm has filled the buckets

$$B_\alpha(\mathbf{c}) := \{\mathbf{x} \in L : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$$

for each $\mathbf{c} \in C$. The algorithm then searches for near-neighbors within each bucket, which we refer to as the *checking phase*. This approach therefore finds all solution pairs $(\mathbf{x}, \mathbf{y})$ for which there exists $\mathbf{c} \in C$ with $|\mathbf{x} \wedge \mathbf{c}| = \alpha$ and $|\mathbf{y} \wedge \mathbf{c}| = \alpha$. Although it may fail to find some solution pairs, the probability of this happening can be controlled by a careful choice of C. Specifically, it can be shown that this approach (possibly repeated with different random choices of C) solves $\mathrm{NNS}(n, w, N)$; see Theorem 7.3.10 for a formal statement.

The resulting LSF-based algorithm for $\mathrm{NNS}(n, w, N)$ is given in Algorithm 7.2.[4] The algorithm uses two subroutines: `FindValidCenters` and `FindSolutions`. For a given vector $\mathbf{x} \in L$, the first subroutine returns the set $V_\alpha(\mathbf{x}) = \{\mathbf{c} \in C : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$ of *valid centers*. For arbitrary and large C, this could be a rather expensive task. However, [Car20; DEEK24] showed that for a suitable choice of C (using so-called random product codes, which we will define in the next section), there is a subroutine `FindValidCenters` with time complexity of order $|V_\alpha(\mathbf{x})|$ on input $\mathbf{x}$.

---

[2] Although C is taken to be a linear code in [DEEK24], we write C rather than $\mathcal{C}$ to distinguish it from CFP instances and the tower of codes in Algorithm 7.1.

[3] The analog in the lattice setting is that a unit vector is added to a bucket corresponding to some unit vector $\mathbf{c} \in \mathbb{R}^n$ if it has (absolute) inner product at least α with $\mathbf{c}$.

[4] Instead of iterating over each $\mathbf{c} \in C$ and searching for pairs $(\mathbf{x}, \mathbf{y})$ in its bucket, the checking phase presented in [DEEK24] iterates over each $\mathbf{x} \in L$ and searches for $\mathbf{y} \in B_\alpha(\mathbf{c})$ for all $\mathbf{c} \in V_\alpha(\mathbf{x})$. This results in the same asymptotic runtime for NNS.

The second subroutine, `FindSolutions`, returns many pairs $(\mathbf{x}, \mathbf{y})$ of vectors in the bucket $B_\alpha(\mathbf{c})$ satisfying $|\mathbf{x} + \mathbf{y}| = w$. Classically, this can be achieved in time $O(|B_\alpha(\mathbf{c})|^2)$ by searching through all pairs in $B_\alpha(\mathbf{c})$. The checking phase contributes most to the overall cost of the algorithm (see Theorem 7.3.10), so our goal in Section 7.4 is to speed up the `FindSolutions` subroutine via a quantum approach.

Algorithm 7.2: LSF-based algorithm for NNS(n, w, N)

Input: Input list $L \subseteq \mathcal{S}_w^n$ of size N
Set of centers $C \subseteq \mathcal{S}_v^n$ for some $v \in [n]$
Bucketing parameter α

Output: List L' of pairs $(\mathbf{x}, \mathbf{y}) \in L^2$ satisfying $|\mathbf{x} + \mathbf{y}| = w$

Initialize empty lists $B_\alpha(\mathbf{c})$ for each $\mathbf{c} \in C$
for $\mathbf{x} \in L$ **do** *// Bucketing*
    Compute $V_\alpha(\mathbf{x}) := \{\mathbf{c} \in C : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$ ▷ `FindValidCenters`
    for $\mathbf{c} \in V_\alpha(\mathbf{x})$ **do**
        Add $\mathbf{x}$ to $B_\alpha(\mathbf{c})$
Initialize an empty list L'
for $\mathbf{c} \in C$ **do** *// Checking*
    Find all $(\mathbf{x}, \mathbf{y}) \in B_\alpha(\mathbf{c})^2$ satisfying $|\mathbf{x} + \mathbf{y}| = w$ ▷ `FindSolutions`
    Add the found solution pairs to L'
return L'

7.3.3 Instantiation using random product codes in $\mathbb{F}_2^n$

To efficiently perform the `FindValidCenters` subroutine, [DEEK24] suggest to let C be a specific binary linear code for which there exists an efficient decoding algorithm. More precisely, they use the notion of *random product codes*, originally introduced in [BDGL16] for $\mathbb{R}^n$ (Section 6.2.4), and defined as follows for $\mathbb{F}_2^n$. Here, we write S_n for the set of permutations of $[n]$.

Definition 7.3.6 (Random product code in $\mathbb{F}_2^n$)**.** We define $\mathcal{R}(\mathcal{S}_v^n, b, M)$ as the family of sets $C \subseteq \mathcal{S}_v^n$ that can be written as

$$C = \sigma(C^{(1)} \times \cdots \times C^{(b)})$$

where $\sigma \in S_n$, and $C^{(i)} \subseteq \mathcal{S}_{v/b}^{n/b}$ with $|C^{(i)}| = M^{1/b}$ for each $i \in [b]$. Such a tuple $(\sigma, C^{(1)}, \ldots, C^{(b)})$ is called a *description* of C.

A *random product code (RPC)* in $\mathbb{F}_2^n$ is a random set $C \in \mathcal{R}(\mathcal{S}_v^n, b, M)$ obtained by sampling a uniformly random description $(\sigma, C^{(1)}, \ldots, C^{(b)})$ from the set of all valid descriptions. We write $\mathrm{RPC}(\mathcal{S}_v^n, b, M)$ for the resulting distribution over $\mathcal{R}(\mathcal{S}_v^n, b, M)$.

In [DEEK24, Definition 4.3], RPCs are defined without including σ, but random permutations are still used in their LSF-based algorithm and analysis; we therefore include σ in the definition. A closely related definition of random product codes was independently presented (in French) in [Car20, Section 9.1].

It was shown in [Car20; DEEK24] that an RPC in $\mathbb{F}_2^n$ has the following useful properties (similar to its analog in $\mathbb{R}^n$):

(1) **Efficient decodability:** In certain parameter regimes (in particular, if b is not too small), there is an algorithm that, on input $C \in \mathcal{R}(\mathcal{S}_v^n, b, M)$ and $\mathbf{x} \in \mathbb{F}_2^n$, computes the set $V_\alpha(\mathbf{x}) := \{\mathbf{c} \in C : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$ in time roughly equal to its size $|V_\alpha(\mathbf{x})|$. See Lemmas 7.3.7 and 7.3.8.

(2) **Random behavior:** In certain parameter regimes (in particular, if b is not too large), a random product code $C \sim \mathrm{RPC}(\mathcal{S}_v^n, b, M)$ behaves like a uniformly random subset of $\mathcal{S}_v^n$ in the following sense: for all $w, \alpha = \Theta(n)$ and all $\mathbf{x}, \mathbf{y} \in \mathcal{S}_w^n$ satisfying $|\mathbf{x} + \mathbf{y}| = w$, the probability that there exists $\mathbf{c} \in V_\alpha(\mathbf{x}) \cap V_\alpha(\mathbf{y})$ is the same, up to subexponential factors, as for $C \sim U(\mathcal{S}_v^n, M)$. See [DEEK24, Lemma 4.8]. We present a variant of this lemma in Lemma 7.3.9.

Although for fixed n these properties hold for a wider range of b, in later sections we restrict to RPCs with parameter $b = \sqrt{n}$ for simplicity, assuming without loss of generality that it is rounded to an integer if necessary.

[Car20, Section 9.1.2] presents an algorithm that computes $V_\alpha(\mathbf{x})$ in full, using an approach reminiscent of the algorithm [BDGL16, Lemma 5.1] for RPCs in $\mathbb{R}^n$.

Lemma 7.3.7 (Efficient decoder for $V_\alpha(\mathbf{x})$ (implicit in [Car20, Proposition 9.1.2]))**.** *Let $b = \omega(1)$ and $M \le 2^n$. There exists a classical algorithm that, given a description of $C \in \mathcal{R}(\mathcal{S}_v^n, b, M)$ and a target vector $\mathbf{x} \in \mathbb{F}_2^n$, returns the set $V_\alpha^{(b)}(\mathbf{x})$ in time $(|V_\alpha^{(b)}(\mathbf{x})| + 1)2^{o(n)}$.*

We will instantiate `FindValidCenters` in Algorithm 7.2 with the algorithm from [Car20], so that the bucketing phase of Algorithm 7.2 constructs the buckets $B_\alpha(\mathbf{c}) = \{\mathbf{x} \in L : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$ in full. In [DEEK24], a simpler algorithm was proposed that computes the subset

$$V_\alpha^{(b)}(\mathbf{x}) \coloneqq \{\sigma(\mathbf{c}_1, \dots, \mathbf{c}_b) \in C : \forall i \in [b], |\sigma(\mathbf{x})_i \wedge \mathbf{c}_i| = \alpha/b\} \subseteq V_\alpha(\mathbf{x})$$

where $C = \sigma(C^{(1)} \times \cdots \times C^{(b)})$. In [DEEK24], `FindValidCenters` is instantiated with this simpler algorithm, in which case the buckets $B_\alpha(\mathbf{c})$ may only be partially constructed; however, they show that this barely affects the overall complexity of solving NNS.

Lemma 7.3.8 (Efficient decoder for $V_\alpha^{(b)}(\mathbf{x})$ (implicit in [DEEK24, Lemma 4.5]))**.** *Let $b = \omega(1)$ and $M \le 2^n$. There exists a classical algorithm that, given a description of $C \in \mathcal{R}(\mathcal{S}_v^n, b, M)$ and a target vector $\mathbf{x} \in \mathbb{F}_2^n$, returns the set $V_\alpha^{(b)}(\mathbf{x})$ in time $(|V_\alpha^{(b)}(\mathbf{x})| + 1)2^{o(n)}$.*

In Section 7.4, we present a quantum-walk algorithm that uses this simpler algorithm to compute sets of the form $V_\alpha^{(b)}(\mathbf{x})$. The use of $V_\alpha^{(b)}(\mathbf{x})$ (rather than the full set $V_\alpha(\mathbf{x})$) allows us to apply the following slight modification of [DEEK24, Lemma 4.8].

Lemma 7.3.9. *Let $\mathbf{x}, \mathbf{y} \in \mathcal{S}_w^n$ satisfy $|\mathbf{x} + \mathbf{y}| = w$. Let $b = \Theta(\sqrt{n})$ and $M \coloneqq \frac{|\mathcal{S}_v^n|}{|\mathcal{C}_v^n(\alpha|\mathbf{x})|}$. Then*

$$\Pr_{C \sim \mathrm{RPC}(\mathcal{S}_v^n, b, M)}[V_\alpha^{(b)}(\mathbf{x}) \cap V_\alpha^{(b)}(\mathbf{y}) \neq \emptyset, |V_\alpha^{(b)}(\mathbf{x})| \le 2^{n/\log n}, |V_\alpha^{(b)}(\mathbf{y})| \le 2^{n/\log n}]$$
$$\ge \frac{|\mathcal{W}_v^n(\alpha \mid \mathbf{x}, \mathbf{y})|}{|\mathcal{C}_v^n(\alpha \mid \mathbf{x})|} 2^{-o(n)}.$$

Proof. Let $E_{\mathbf{x}}$ denote the event that $|V_\alpha^{(b)}(\mathbf{x})| \le 2^{n/\log n}$, and let $E_{\mathbf{y}}$ be defined analogously. Let $E_{(\mathbf{x},\mathbf{y})}$ denote the event that $V_\alpha^{(b)}(\mathbf{x}) \cap V_\alpha^{(b)}(\mathbf{y}) \neq \emptyset$. By definition of the RPC distribution, sampling $C \sim \mathrm{RPC}(\mathcal{S}_v^n, b, M)$ is the same as first sampling $\sigma \sim U(S_n)$ and then sampling $C^{(i)}$ for $i \in [b]$. For $\sigma \in S_n$, $\mathbf{x}' := \sigma(\mathbf{x})$, and $\mathbf{y}' := \sigma(\mathbf{y})$, let E_σ denote the event that $|\mathbf{x}'_i| = |\mathbf{y}'_i| = |\mathbf{x}'_i + \mathbf{y}'_i| = w/b$ for all $i \in [b]$. It suffices to prove $\Pr_{\sigma,C^{(1)},\dots,C^{(b)}}[E_{(\mathbf{x},\mathbf{y})}, E_{\mathbf{x}}, E_{\mathbf{y}}, E_\sigma] \ge \min\left\{M \cdot \frac{|\mathcal{W}_v^n(\alpha|\mathbf{x},\mathbf{y})|}{|\mathcal{S}_v^n|}, 1\right\} 2^{-o(n)}$. The proof of [DEEK24, Lemma 4.8] shows

$$\Pr_{\sigma\sim U(S_n)}[E_\sigma] \ge \left(\frac{1}{1+n/b}\right)^{3b} = 2^{-o(n)}$$

and

$$\Pr_{\sigma,C^{(1)},\dots,C^{(b)}}[E_{(\mathbf{x},\mathbf{y})} \mid E_\sigma] \ge \min\left\{M\frac{|\mathcal{W}_v^n(\alpha \mid \mathbf{x},\mathbf{y})|}{|\mathcal{S}_v^n|}, 1\right\} 2^{-o(n)} = \frac{|\mathcal{W}_v^n(\alpha \mid \mathbf{x},\mathbf{y})|}{|\mathcal{C}_v^n(\alpha \mid \mathbf{x})|} 2^{-o(n)}.$$

Since $\Pr[E_{(\mathbf{x},\mathbf{y})}, E_{\mathbf{x}}, E_{\mathbf{y}} \mid E_\sigma] \ge \Pr[E_{(\mathbf{x},\mathbf{y})} \mid E_\sigma] - \Pr[\neg E_{\mathbf{x}} \mid E_\sigma] - \Pr[\neg E_{\mathbf{y}} \mid E_\sigma]$, it therefore suffices to show $\Pr_{\sigma,C^{(1)},\dots,C^{(b)}}[\neg E_{\mathbf{x}} \mid E_\sigma] \le 2^{-\omega(n)}$ and $\Pr_{\sigma,C^{(1)},\dots,C^{(b)}}[\neg E_{\mathbf{y}} \mid E_\sigma] \le 2^{-\omega(n)}$.

If $|V_\alpha^{(b)}(\mathbf{x})| > 2^{n/\log n}$, then there exists $i \in [b]$ such that the set $V_i := \{\mathbf{c}_i \in C^{(i)} : |\sigma(\mathbf{x})_i \wedge \mathbf{c}_i| = \alpha/b\}$ has size $> 2^{n/(b\log n)}$. When $C^{(i)} \sim U(\mathcal{S}_{v/b}^{n/b}, M^{1/b})$, [DEEK24, Lemma 4.6] implies $n^{-O(1)} \le \mathbb{E}_{C^{(i)}}[|V_i|] \le n^{O(1)}$ (for our choice of M). Since all elements of $C^{(i)}$ are independent, the Chernoff bound (Corollary 2.1.4) implies $\Pr_{C^{(i)}}[|V_i| > 2^{n/(b\log n)}] \le 2^{-\omega(n)}$. Therefore, the union bound over $i \in [b]$ implies $\Pr_{\sigma,C^{(1)},\dots,C^{(b)}}[\neg E_{\mathbf{x}} \mid E_\sigma] \le 2^{-\omega(n)}$. The same argument applies to $E_{\mathbf{y}}$, so the lemma follows. □

7.3.4 Complexity of solving NNS(n, w, N) and CFP(n, k, w, N)

Using the algorithm from Lemma 7.3.7 for `FindValidCenters` and given an algorithm for `FindSolutions`, this results in the following time and memory complexities for $\mathrm{NNS}(n, w, N)$, as shown in [Car20, Cor. 8.2.6] and [DEEK24, Cor. 4.2]. By instantiating the oracle $\mathcal{A}_{\mathrm{NNS}}$ in Algorithm 7.1 with the resulting algorithm for $\mathrm{NNS}(n, w, N)$, we obtain an algorithm for $\mathrm{CFP}(n, k, w, N)$ [DEEK24, Theorem 3.2].

Theorem 7.3.10 ([Car20, Cor. 8.2.6] and [DEEK24, Cor. 4.2, Theorem 3.2])**.** *Let $n, w, N \in \mathbb{N}$ with $w = \Theta(n)$ such that $NNS(n, w, N)$ (Problem 7.3.2) is well-defined. Let $\nu, \alpha \in \mathbb{N}$ satisfy $\nu, \alpha = \Theta(n)$. For arbitrary $\mathbf{x}, \mathbf{y} \in \mathcal{S}_w^n$ satisfying $|\mathbf{x} + \mathbf{y}| = w$, define*

$$M := \frac{|\mathcal{S}_\nu^n|}{|\mathcal{C}_\nu^n(\alpha \mid \mathbf{x})|} 2^{o(n)} \qquad \text{and} \qquad R := \frac{|\mathcal{C}_\nu^n(\alpha \mid \mathbf{x})|}{|\mathcal{W}_\nu^n(\alpha \mid \mathbf{x}, \mathbf{y})|} 2^{o(n)}.$$

Let `FindSolutions` *be a classical (resp. quantum) algorithm that, on input a list $B_\alpha(\mathbf{c}) = \{\mathbf{x} \in L : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$ for some $L \subseteq \mathcal{S}_w^n$ and $\mathbf{c} \in \mathcal{S}_\nu^n$, finds all pairs $\mathbf{x}, \mathbf{y} \in B_\alpha(\mathbf{c})$ satisfying $|\mathbf{x} + \mathbf{y}| = w$.*
If `FindSolutions` *has expected runtime T over $L \sim U(\mathcal{S}_w^n, N)$, then there exists a classical (resp. quantum) algorithm that solves $NNS(n, w, N)$ in expected time*

$$R(N\, 2^{o(n)} + M\, T)$$

using R calls to Algorithm 7.2, where each call is given an independent sample $C \sim \mathrm{RPC}(\mathcal{S}_\nu^n, \sqrt{n}, M)$. It uses an expected number of $N\, 2^{o(n)}$ classical bits, plus any additional memory used by the `FindSolutions` *subroutine.*
Moreover, if $4\binom{n}{w} / \left(\binom{w}{w/2}\binom{n-w}{w/2}\right) + 1 \leq N \leq \binom{n}{w}/2^{n-k}$ and $k = \Theta(n)$ is such that the binary-sieve heuristic in [DEEK24] holds for (n, k, w, N), then there exists a classical (resp. quantum) sieving-based algorithm for $CFP(n, k, w, N)$ (Problem 7.3.1) with the same time and memory complexity, up to a polynomial factor in n.

Remark 7.3.11 (Error probability of `FindSolutions`). Theorem 7.3.10 considers a subroutine `FindSolutions` that succeeds with certainty. The claimed algorithm for $\mathrm{NNS}(n, w, N)$ invokes this subroutine $RM = 2^{O(n)}$ times, and the resulting sieving-based algorithm for $\mathrm{CFP}(n, k, w, N)$ invokes it $(n-k)RM = 2^{O(n)}$ times. Using an application of the union bound, `FindSolutions` is allowed some error probability that is inverse-exponentially small in n.[5] While the classical subroutine `FindSolutions` considered by [DEEK24] succeeds with certainty, the quantum subroutines that we present in Section 7.4 are not guaranteed to succeed, so combining them with Theorem 7.3.10 should take their error probability into account.

[5] If the sieving-based algorithm is instantiated within ISD (as we will consider in Section 7.6), then this error probability should also account for the number of times the sieving-based algorithm is invoked.

7.4 Quantum algorithms for FindSolutions

In this section, we present quantum analogs of the key subroutine `FindSolutions` in the LSF-based algorithm Algorithm 7.2 for $\mathrm{NNS}(n, w, N)$. Our quantum algorithms make use of Grover's search algorithm and quantum walks, and we analyze their time and memory complexities in Theorem 7.4.6 and Theorem 7.4.10 below. Plugging these results into Theorem 7.3.10 provides an upper bound on the quantum complexity of $\mathrm{NNS}(n, w, N)$, and in turn on the quantum complexity of $\mathrm{CFP}(n, k, w, N)$ via code sieving (Algorithm 7.1).

To formalize the problem solved by `FindSolutions`, consider Algorithm 7.2 with as input a list $L \subseteq \mathcal{S}_w^n$, a set $C \subseteq \mathcal{S}_v^n$ of centers, and a bucketing parameter α. Upon successful completion of the bucketing phase, the algorithm has created a data structure that stores, for each $\mathbf{c} \in C$, the bucket

$$B_\alpha(\mathbf{c}) := \{\mathbf{x} \in L : |\mathbf{x} \wedge \mathbf{c}| = \alpha\} \subseteq \mathcal{S}_w^n. \tag{7.1}$$

Given $\mathbf{c}$ and its bucket $B_\alpha(\mathbf{c})$, the goal of `FindSolutions` is to find all $(\mathbf{x}, \mathbf{y}) \in B_\alpha(\mathbf{c})^2$ satisfying $|\mathbf{x} + \mathbf{y}| = w$.

When L is an instance of $\mathrm{NNS}(n, w, N)$, it consists of N i.i.d. uniform samples from $\mathcal{S}_w^n$. For fixed $\mathbf{c} \in \mathcal{S}_v^n$ (independently chosen from L), the expected number of elements in $B_\alpha(\mathbf{c})$ is $N\binom{v}{\alpha}\binom{n-v}{w-\alpha}/\binom{n}{w}$. Moreover, the elements in $B_\alpha(\mathbf{c})$ are i.i.d. uniformly random over the spherical cap $\mathcal{C}_w^n(\alpha \mid \mathbf{c}) = \{\mathbf{x} \in \mathcal{S}_w^n : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$. Hence, the task of `FindSolutions` can be formulated as follows.

> **Problem 7.4.1** (Bucket search)**.** Let $B, n, w, v, \alpha \in \mathbb{N}$ satisfy $B \leq \binom{v}{\alpha}\binom{n-v}{w-\alpha}$ and $\alpha \leq \max\{v, w\} \leq n$ (referred to as *well-defined parameters*). Given $\mathbf{c} \in \mathcal{S}_v^n$ and a list $B_\alpha(\mathbf{c})$ of B i.i.d. uniform samples from $\{\mathbf{x} \in \mathcal{S}_w^n : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$, find all pairs of distinct $\mathbf{x}, \mathbf{y} \in B_\alpha(\mathbf{c})$ with $|\mathbf{x} + \mathbf{y}| = w$. We will refer to such a pair $(\mathbf{x}, \mathbf{y})$ as a *solution pair in* $B_\alpha(\mathbf{c})$.

This problem is a generalization of the Near-Neighbor Search problem (Problem 7.3.2) defined in Section 7.3, since we obtain $\mathrm{NNS}(n, w, N)$ by taking $v = n$ and $\alpha = w$ (which forces $\mathbf{c}$ to be the all-ones vector).[6] For other choices of v and α, the elements of $B_\alpha(\mathbf{c})$ are not uniformly distributed over $\mathcal{S}_w^n$ but over a proper subset.

[6] This means that, for suitable parameters, our quantum algorithms for Problem 7.4.1 also directly solve NNS (instead of via Algorithm 7.2).

We next analyze the probability that a pair of vectors $\mathbf{x}, \mathbf{y} \sim U(B_\alpha(\mathbf{c}))$ satisfies $|\mathbf{x}+\mathbf{y}| = w$, yielding an explicit formula for the expected number of solutions to Problem 7.4.1. Afterwards, we present two quantum algorithms for solving Problem 7.4.1 and analyze their cost in terms of (B, n, w, v, α). When used as the subroutine `FindSolutions` in Algorithm 7.2, the expected cost of solving $\mathrm{NNS}(n, w, N)$ then depends on the choice of v and α, since the expected value of B is determined by them. By optimizing v and α, we can minimize the resulting complexities of $\mathrm{NNS}(n, w, N)$ (and thus $\mathrm{CFP}(n, k, w, N)$), which we do in Section 7.5 to quantify the resulting quantum speedups.

7.4.1 Number of solution pairs in $B_\alpha(\mathbf{c})$

Consider parameters (B, n, w, v, α) of Problem 7.4.1 and fix $\mathbf{c} \in \mathcal{S}_v^n$. Let $B_\alpha(\mathbf{c})$ be a list of B i.i.d. uniform samples from $\{\mathbf{x} \in \mathcal{S}_w^n : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$. The probability that a pair of vectors $\mathbf{x}, \mathbf{y} \sim U(B_\alpha(\mathbf{c}))$ satisfies $|\mathbf{x}+\mathbf{y}| = w$ equals

$$\Pr_{\mathbf{x},\mathbf{y}\sim U(B_\alpha(\mathbf{c}))}[|\mathbf{x}+\mathbf{y}| = w] = \Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x}+\mathbf{y}| = w \mid |\mathbf{x}\wedge\mathbf{c}| = \alpha \text{ and } |\mathbf{y}\wedge\mathbf{c}| = \alpha].$$

By Lemma 7.4.2, this probability is independent of $\mathbf{c}$ and depends only on (n, w, v, α). Accordingly, for arbitrary $\mathbf{c} \in \mathcal{S}_v^n$, we define

$$p(n, w, v, \alpha) := \Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x}+\mathbf{y}| = w \mid |\mathbf{x}\wedge\mathbf{c}| = \alpha \text{ and } |\mathbf{y}\wedge\mathbf{c}| = \alpha]. \tag{7.2}$$

The expected number of solutions to Problem 7.4.1 is then given by

$$\binom{B}{2} p(n, w, v, \alpha).$$

We will next provide an explicit formula for $p(n, w, v, \alpha)$, and then show that, with overwhelming probability over the random choice of $B_\alpha(\mathbf{c})$ in Problem 7.4.1, the number of solution pairs in $B_\alpha(\mathbf{c})$ is close to its expectation.

Probability of forming a solution pair

Lemma 7.4.2 (Probability of forming a solution pair in $B_\alpha(\mathbf{c})$)**.** *Let $n, w, v, \alpha \in \mathbb{N}$ satisfy $\alpha \le \max\{v, w\} \le n$. For all $\mathbf{c} \in \mathcal{S}_v^n$, the probability $p(n, w, v, \alpha)$ defined in Equation (7.2) equals*

$$p(n,w,v,\alpha) = \frac{\binom{w}{w/2}\binom{n-w}{w/2}}{\binom{v}{\alpha}\binom{n-v}{w-\alpha}\binom{w}{\alpha}\binom{n-w}{v-\alpha}} \cdot |\mathcal{W}_v^n(\alpha \mid \mathbf{x}', \mathbf{y}')|$$

where $|\mathcal{W}_v^n(\alpha \mid \mathbf{x}', \mathbf{y}')| = \sum_{u=\max\{0,2\alpha-v\}}^{\min\{\alpha,w/2\}} \binom{w/2}{u}\binom{w/2}{\alpha-u}^2\binom{n-3w/2}{v-2\alpha+u}$ is the surface area of a wedge for arbitrary $\mathbf{x}', \mathbf{y}' \in \mathcal{S}_w^n$ with $|\mathbf{x}' + \mathbf{y}'| = w$.

Note that $p(n, w, n, w) = \frac{\binom{w}{w/2}\binom{n-w}{w/2}}{\binom{n}{w}}$, recovering Lemma 7.3.4.

Remark 7.4.3. We can rephrase $p(n, w, v, \alpha)$ in terms of surface areas of Hamming spheres, caps, and a wedge:

$$p(n,w,v,\alpha) = \left(\frac{|\mathcal{S}_v^n|}{|\mathcal{C}_v^n(\alpha \mid \mathbf{x}')|}\right)^2 \frac{|\mathcal{C}_w^n(w/2 \mid \mathbf{x}')|}{|\mathcal{S}_w^n|} \frac{|\mathcal{W}_v^n(\alpha \mid \mathbf{x}', \mathbf{y}')|}{|\mathcal{S}_v^n|}$$

for all $\mathbf{x}', \mathbf{y}' \in \mathcal{S}_w^n$ with $|\mathbf{x}' + \mathbf{y}'| = w$.

We prove Lemma 7.4.2 using the following lemma.[7]

Lemma 7.4.4. *Let $n, w, v, \alpha \in \mathbb{N}$ satisfy $\alpha \le \max\{v, w\} \le n$. For all $\mathbf{c} \in \mathcal{S}_v^n$ and $\mathbf{x}', \mathbf{y}' \in \mathcal{S}_w^n$ satisfying $|\mathbf{x}' + \mathbf{y}'| = w$, we have*

$$\Pr_{\mathbf{x},\mathbf{y} \sim U(\mathcal{S}_w^n)}[|\mathbf{x} \wedge \mathbf{c}| = \alpha \text{ and } |\mathbf{y} \wedge \mathbf{c}| = \alpha \mid |\mathbf{x} \wedge \mathbf{y}| = w/2] = \frac{|\mathcal{W}_v^n(\alpha \mid \mathbf{x}', \mathbf{y}')|}{|\mathcal{S}_v^n|}.$$

Proof of Lemma 7.4.4. For fixed $\mathbf{c} \in \mathcal{S}_v^n$ and $\mathbf{x}', \mathbf{y}' \in \mathcal{S}_w^n$ satisfying $|\mathbf{x}' + \mathbf{y}'| = w$, the stated probability is equal to

$$\frac{|\{(\mathbf{x}, \mathbf{y}) \in (\mathcal{S}_w^n)^2 \colon |\mathbf{x} \wedge \mathbf{c}| = \alpha, |\mathbf{y} \wedge \mathbf{c}| = \alpha, |\mathbf{x} \wedge \mathbf{y}| = w/2\}|}{|\{(\mathbf{x}, \mathbf{y}) \in (\mathcal{S}_w^n)^2 \colon |\mathbf{x} \wedge \mathbf{y}| = w/2\}|}$$
$$= \frac{\sum_u \binom{v}{\alpha}\binom{n-v}{w-\alpha}\binom{\alpha}{u}\binom{v-\alpha}{\alpha-u}\binom{w-\alpha}{w/2-u}\binom{n-(w+v-\alpha)}{w/2-\alpha+u}}{\binom{n}{w}\binom{w}{w/2}\binom{n-w}{w/2}}$$

[7] We will repeatedly make use of the fact that $\binom{a}{b}\binom{b}{d}\binom{a-b}{c-d} = \binom{a}{c}\binom{c}{d}\binom{a-c}{b-d}$ for all $a, b, c, d \in \mathbb{N}$ for which these binomial coefficients are well-defined.

where u ranges over all possible values of $|\mathbf{x} \wedge \mathbf{y} \wedge \mathbf{c}|$. Substituting the equalities $\binom{n}{v}\binom{v}{\alpha}\binom{n-v}{w-\alpha} = \binom{n}{w}\binom{w}{\alpha}\binom{n-w}{v-\alpha}$, $\binom{w}{\alpha}\binom{\alpha}{u}\binom{w-\alpha}{w/2-u} = \binom{w}{w/2}\binom{w/2}{u}\binom{w-w/2}{\alpha-u}$, and the equality $\binom{n-w}{v-\alpha}\binom{v-\alpha}{\alpha-u}\binom{(n-w)-(v-\alpha)}{w/2-(\alpha-u)} = \binom{n-w}{w/2}\binom{w/2}{\alpha-u}\binom{(n-w)-w/2}{(v-\alpha)-(\alpha-u)}$ then proves that this probability equals $|\mathcal{W}_v^n(\alpha \mid \mathbf{x}', \mathbf{y}')|/\binom{n}{v}$. □

Proof of Lemma 7.4.2. By definition of $p := p(n, w, v, \alpha)$ and using Lemma 7.4.4 for the last equality, we obtain

$$\begin{aligned} p &= \Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x} \wedge \mathbf{y}| = w/2 \mid |\mathbf{x} \wedge \mathbf{c}| = |\mathbf{y} \wedge \mathbf{c}| = \alpha] \\ &= \frac{\Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x} \wedge \mathbf{c}| = |\mathbf{y} \wedge \mathbf{c}| = \alpha \mid |\mathbf{x} \wedge \mathbf{y}| = w/2] \cdot \Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x} \wedge \mathbf{y}| = w/2]}{\Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x} \wedge \mathbf{c}| = |\mathbf{y} \wedge \mathbf{c}| = \alpha]} \\ &= \frac{|\mathcal{W}_v^n(\alpha \mid \mathbf{x}', \mathbf{y}')|}{\binom{n}{v}} \cdot \frac{\Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x} \wedge \mathbf{y}| = w/2]}{\Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x} \wedge \mathbf{c}| = |\mathbf{y} \wedge \mathbf{c}| = \alpha]} \end{aligned}$$

for arbitrary $\mathbf{x}', \mathbf{y}' \in \mathcal{S}_w^n$ satisfying $|\mathbf{x}' + \mathbf{y}'| = w$. Since $\Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x} \wedge \mathbf{y}| = w/2] = \binom{n}{w}\binom{w}{w/2}\binom{n-w}{w/2}/\binom{n}{w}^2$ and $\Pr_{\mathbf{x},\mathbf{y}\sim U(\mathcal{S}_w^n)}[|\mathbf{x} \wedge \mathbf{c}| = |\mathbf{y} \wedge \mathbf{c}| = \alpha] = \binom{v}{\alpha}^2\binom{n-v}{w-\alpha}^2/\binom{n}{w}^2$, it follows that

$$p = \frac{|\mathcal{W}_v^n(\alpha \mid \mathbf{x}', \mathbf{y}')|}{\binom{n}{v}} \frac{\binom{n}{w}\binom{w}{w/2}\binom{n-w}{w/2}}{\binom{v}{\alpha}^2\binom{n-v}{w-\alpha}^2}.$$

Notice that $\binom{n}{v}\binom{v}{\alpha}\binom{n-v}{w-\alpha} = \binom{n}{w}\binom{w}{\alpha}\binom{n-w}{v-\alpha}$, hence the desired result follows. □

Bounding the number of solution pairs

Lemma 7.4.5 (Number of solution pairs)**.** *Let (B, n, w, v, α) be well-defined parameters of Problem 7.4.1, and let $p_{\text{sol}} := p(n, w, v, \alpha)$ as in Equation (7.2). With probability $1 - 1/\Omega(\binom{B}{2}p_{\text{sol}})$ over the random choice of $B_\alpha(\mathbf{c})$ in Problem 7.4.1, the number of solution pairs in $B_\alpha(\mathbf{c})$ equals $\Theta(\binom{B}{2}p_{\text{sol}})$.*

Proof. We view $B_\alpha(\mathbf{c})$ as an ordered tuple $(\mathbf{x}_1, \ldots, \mathbf{x}_B)$. For all $i, j \in [B]$, define the random variable $X_{i,j} \in \{0, 1\}$ by setting $X_{i,j} = 1$ if and only if $i \neq j$ and $|\mathbf{x}_i + \mathbf{x}_j| = w$ (that is, $(\mathbf{x}_i, \mathbf{x}_j)$ is a solution pair). Then the number of solution pairs is $X/2$, where $X = \sum_{i,j\in[B]} X_{i,j}$, so it suffices to show X equals $\Theta(\binom{B}{2}p_{\text{sol}})$. If $\mathbb{E}[X^2] - \mathbb{E}[X]^2 = O(\mathbb{E}[X])$, then Chebyshev's inequality (Lemma 2.1.2) implies $X = \Theta(\mathbb{E}[X])$, except with probability $1/\Omega(\mathbb{E}[X])$. The lemma then follows, be-

cause $\mathbb{E}[X] = B(B-1)p_{\text{sol}}$. It therefore suffices to show $\mathbb{E}[X^2] - \mathbb{E}[X]^2 = O(\mathbb{E}[X])$. For all $i, j, k, \ell \in [B]$ with $k \neq \ell$, we have

$$\Pr[X_{k,\ell} = 1 \mid X_{i,j} = 1] = \begin{cases} \Pr[X_{k,\ell} = 1] & \text{if } \{k,\ell\} \neq \{i,j\} \\ 1 & \text{otherwise} \end{cases}$$

and thus $q_{i,j} \coloneqq \sum_{k,\ell \in [B]} (\Pr[X_{k,\ell} = 1 \mid X_{i,j} = 1] - \Pr[X_{k,\ell} = 1]) = O(1)$, because the terms in the sum can be nonzero only if $k, \ell \in [B]$ satisfy $k \neq \ell$ and $\{k,\ell\} = \{i,j\}$. Therefore,

$$\begin{aligned} \mathbb{E}[X^2] - \mathbb{E}[X]^2 &= \sum_{i,j \in [B]} \sum_{k,\ell \in [B]} \mathbb{E}[X_{i,j} X_{k,\ell}] - \sum_{i,j \in [B]} \mathbb{E}[X_{i,j}] \sum_{k,\ell \in [B]} \mathbb{E}[X_{k,\ell}] \\ &= \sum_{i,j \in [B]} \Pr[X_{i,j} = 1] q_{i,j} \\ &= O(\mathbb{E}[X]) \end{aligned}$$

as desired. □

7.4.2 Quantum algorithm using Grover's search algorithm

Our first quantum algorithm for Problem 7.4.1 (and thus for the `FindSolutions` subroutine) is a straightforward application of Grover's search algorithm [Gro96], which was described in Section 5.1. This Grover-based algorithm will serve as a baseline for comparison with the quantum-walk algorithm that we present next.

Theorem 7.4.6 (`FindSolutions` using Grover). *Let (B, n, w, v, α) be well-defined parameters of Problem 7.4.1, and let $p_{\text{sol}} \coloneqq p(n, w, v, \alpha)$ as in Equation (7.2). There exists a quantum algorithm that, given QCRAM access to an instance $B_\alpha(\mathbf{c})$ of Problem 7.4.1, finds all solution pairs in $B_\alpha(\mathbf{c})$ in time $\binom{B}{2}\sqrt{p_{\text{sol}}}\, 2^{o(n)}$ using $\text{poly}(n)$ qubits, with success probability $1 - 1/\Omega(\binom{B}{2} p_{\text{sol}})$ over the random choice of $B_\alpha(\mathbf{c})$.*

Proof. We consider a quantum algorithm that repeatedly applies Grover's algorithm (Theorem 5.1.2) to the set of all $\binom{B}{2}$ pairs in the input list $B_\alpha(\mathbf{c})$, where a pair $(\mathbf{x}, \mathbf{y})$ is considered a solution if and only if $|\mathbf{x}+\mathbf{y}| = w$ (which can be checked

in time $\mathrm{poly}(n)$). By Lemma 7.4.5, with probability $1 - 1/\Omega(\binom{B}{2} p_{\mathrm{sol}})$ over $B_\alpha(\mathbf{c})$, the number of solution pairs equals $\Theta(\binom{B}{2} p_{\mathrm{sol}})$. A coupon-collector argument (for instance, Lemma 6.2.1) then implies that $\binom{B}{2} p_{\mathrm{sol}} 2^{o(n)}$ repetitions suffice to find all solutions. Since one run of Grover's algorithm takes time $(1/\sqrt{p_{\mathrm{sol}}})2^{o(n)}$, this results in an overall runtime of $\binom{B}{2}\sqrt{p_{\mathrm{sol}}}2^{o(n)}$. Each run of Grover's algorithm uses QCRAM access to the input list $B_\alpha(\mathbf{c})$, and the number of qubits used is polynomial in n, which can be reused in the next run. □

By instantiating the subroutine `FindSolutions` in Algorithm 7.2 with this Grover-based quantum algorithm, we obtain a quantum algorithm for $\mathrm{NNS}(n, w, N)$, and consequently for $\mathrm{CFP}(n, k, w, N)$ via code sieving (Algorithm 7.1). The resulting time and memory complexities then follow from combining Theorem 7.4.6 with Theorem 7.3.10, as long as the error probability of this Grover-based subroutine is taken into account (Remark 7.3.11).

7.4.3 Quantum algorithm using quantum walks

We will now improve the Grover-based approach for solving Problem 7.4.1 using quantum-walk techniques. To speed up the search for solution pairs in $B_\alpha(\mathbf{c})$, we will add a layer of locality-sensitive filtering (LSF). Similar to how LSF is used in Algorithm 7.2 to facilitate the search for solution pairs in the input list L, we will restrict the search for solution pairs in $B_\alpha(\mathbf{c})$ to the neighborhoods (buckets) of some vectors $\mathbf{c}'$ in a set C', quantified by a bucketing parameter β. (Remember that Problem 7.4.1 is in fact a generalization of the NNS problem solved by Algorithm 7.2.) This approach was inspired by a quantum-walk algorithm for lattice sieving from [CL21], and we show that most of their ideas can be transferred from the Euclidean metric to the Hamming metric.

Remark 7.4.7 (Two layers of LSF when solving NNS). Our motivation for analyzing Problem 7.4.1 is to understand the quantum complexity of the subroutine `FindSolutions` in Algorithm 7.2. The latter algorithm considers one layer of LSF defined by a set $C \subseteq \mathcal{S}_v^n$ and bucketing parameter α, so instantiating Algorithm 7.2 with our LSF-based quantum-walk algorithm for `FindSolutions` introduces a second layer of LSF.

MNRS-style quantum walk. We consider the MNRS quantum-walk framework from [MNRS11], which was detailed in Section 5.3. For some suitable vertex size $s \leq B$, we define a quantum walk on the Johnson graph $J(B, s)$. The set $\mathcal{V}$ of vertices of the graph is therefore identified with the set of size-s subsets of the input list $B_\alpha(\mathbf{c})$:

$$\mathcal{V} = \{S \subseteq B_\alpha(\mathbf{c}) : |S| = s\}.$$

Two vertices are adjacent if and only if the corresponding subsets differ in exactly one element. We will define the set $\mathcal{M} \subseteq \mathcal{V}$ of *marked vertices* as a subset of vertices containing a solution pair:

$$\mathcal{M} \subseteq \{S \in \mathcal{V} : \exists \mathbf{x}, \mathbf{y} \in S, |\mathbf{x} + \mathbf{y}| = w\}. \tag{7.3}$$

This allows us to find solution pairs by searching for marked vertices using the quantum-walk framework from [MNRS11]. To facilitate the search for marked vertices, we equip each vertex $S \in \mathcal{V}$ with a data structure $\mathsf{data}(S)$ that incorporates LSF locally inside S, as we will explain below.

The quantum walk starts by setting up a superposition over all the vertices and their data, and then "walks" through the graph (by repeatedly applying a check and update step) until it has essentially reached a superposition over only the marked vertices. We briefly sketch the inner steps of the quantum walk:

- The setup step (of cost S) constructs a uniform superposition over $|S, \mathsf{data}(S)\rangle$ for all $S \in \mathcal{V}$. Since each vertex consists of $|S| = s$ elements, the setup cost is essentially s plus the cost of mapping $|S, 0\rangle \mapsto |S, \mathsf{data}(S)\rangle$.
- The checking step (of cost C) decides whether a given vertex S is marked. A naive Grover search allows us to decide whether $\exists \mathbf{x}, \mathbf{y} \in S$ with $|\mathbf{x} + \mathbf{y}| = w$ in time roughly $\sqrt{|S|^2} = s$. We will define the set of marked vertices $\mathcal{M}$ and the data structures such that the checking cost C is significantly cheaper than s, and thus S.
- The update step (of cost U) maps a vertex $S \in \mathcal{V}$ to a uniform superposition over its adjacent vertices S', and updates the data. By definition of the Johnson graph, each adjacent vertex S' is of the form $S' = (S \setminus \{\mathbf{x}_{\text{old}}\}) \cup \{\mathbf{x}_{\text{new}}\}$, so the update from $\mathsf{data}(S)$ to $\mathsf{data}(S')$ only has to account for the changes induced by replacing $\mathbf{x}_{\text{old}}$ with $\mathbf{x}_{\text{new}}$. We will show that this is possible without having to compute $\mathsf{data}(S')$ from scratch, so that U is significantly cheaper than the setup cost S.

By writing ε for the fraction of marked vertices and δ for the spectral gap of the graph, Theorem 5.3.2 implies that this quantum walk returns a marked vertex in time roughly

$$\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}\left(\mathsf{C} + \frac{1}{\sqrt{\delta}}\mathsf{U}\right).$$

Each run of the quantum walk finds a sufficiently random solution pair in $B_\alpha(\mathbf{c})$, so by repeating the walk enough times we can then solve Problem 7.4.1.

Remark 7.4.8 (Reusing the setup state). The final state of the quantum walk is a superposition over marked vertices, each of which contains a solution pair in $B_\alpha(\mathbf{c})$. When searching for many solutions, it seems wasteful to simply measure this state, output the found solution, and discard the remaining information in the vertex. It was recently shown in [BCSS23] that this information can indeed be reused to construct a new setup state, thereby avoiding having to pay the setup cost S again. This *reusable* (or "chained") quantum-walk technique finds m solutions at a total cost of roughly

$$\frac{m}{m'}\left(\mathsf{S} + \frac{m'}{\sqrt{\varepsilon}}\left(\mathsf{C} + \frac{1}{\sqrt{\delta}}\mathsf{U}\right)\right)$$

where $m' \in [m]$ is optimized over admissible choices. The non-reusable approach corresponds to taking $m' = 1$. By applying this reusable technique to the quantum-walk algorithm for lattice sieving in [CL21], the authors of [BCSS23] improved the time complexity from $2^{0.2570n}$ to $2^{0.2563n}$, where n denotes the lattice dimension. While this technique could also be adapted to our code setting, we omit the details, as our numerical results indicate only a minor speedup compared to the non-reusable approach (see Remark 7.5.1).

Adding a layer of LSF. We already mentioned that the detection of marked vertices will be facilitated using a layer of LSF, added to the data structures of the vertices. For some fixed set C' and bucketing parameter β, the data structure $\mathsf{data}(S)$ of S will store pairs $(\mathbf{x}, \mathbf{c}') \in S \times C'$ such that $\mathbf{x}$ and $\mathbf{c}'$ overlap in β coordinates (i.e., both are nonzero in those coordinates). To check whether a vertex S contains a solution pair, we will then search, for each $\mathbf{c}' \in C'$, among all $(\mathbf{x}, \mathbf{c}'), (\mathbf{y}, \mathbf{c}')$ that are stored in $\mathsf{data}(S)$ for solution pairs $(\mathbf{x}, \mathbf{y})$. In some sense, we thereby implement Algorithm 7.2 locally to S: the construction of its data structure can be

seen as the bucketing phase, and the search for solution pairs that share a "close" $\mathbf{c}' \in C'$ can be seen as the checking phase.

More concretely, recall that each vertex $S \in \mathcal{V}$ is a subset of $B_\alpha(\mathbf{c}) \subseteq \mathcal{S}_w^n$. Since every $\mathbf{x} \in B_\alpha(\mathbf{c})$ satisfies $|\mathbf{x} \wedge \mathbf{c}| = \alpha$, we will identify each $\mathbf{x} \in B_\alpha(\mathbf{c})$ with the v-dimensional vector $\pi_{\mathbf{c}}(\mathbf{x}) \in \mathcal{S}_\alpha^v$ obtained by projecting $\mathbf{x} \mapsto \mathbf{x} \wedge \mathbf{c}$ onto the support of $\mathbf{c}$, and ignoring the remaining $n - v$ coordinates of $\mathbf{x} \wedge \mathbf{c}$ (which are all zero). Slightly simplified, we then apply LSF as follows, where v' and β are additional parameters that affect the overall complexity (and should therefore be optimized). For a fixed set $C' \subseteq \mathcal{S}_{v'}^v$ of weight-v' vectors, we associate each vertex $S \subseteq B_\alpha(\mathbf{c})$ with the *β-buckets* $(B_\beta(\mathbf{c}') \cap S)_{\mathbf{c}' \in C'}$ defined by

$$B_\beta(\mathbf{c}') \cap S := \{\mathbf{x} \in S : |\pi_{\mathbf{c}}(\mathbf{x}) \wedge \mathbf{c}'| = \beta\}$$

and we consider the marked set

$$\mathcal{M} := \{S \in \mathcal{V} : \exists \mathbf{c}' \in C', \exists \mathbf{x}, \mathbf{y} \in B_\beta(\mathbf{c}') \cap S, |\mathbf{x} + \mathbf{y}| = w\}.$$

By storing the β-buckets in the data structure $\mathtt{data}(S)$, we can then search for solution pairs in S (specifically, check whether $S \in \mathcal{M}$) by searching among all $(\mathbf{x}, \mathbf{y}) \in \bigcup_{\mathbf{c}' \in C'} (B_\beta(\mathbf{c}') \cap S)^2$. Depending on the choice of C', this union may be sufficiently smaller than S^2, so this approach may outperform a naive Grover search among all pairs in the vertex. While this definition of $\mathcal{M}$ may overlook some solution pairs when Equation (7.3) is a strict inclusion, we will repeat the quantum walk many times for sufficiently random sets C' to ensure that we find all solution pairs and thereby solve Problem 7.4.1.

By sampling each C' as a random product code $C' \sim \mathrm{RPC}(\mathcal{S}_{v'}^v, \sqrt{v}, M')$ (of size M' that we will carefully choose below), we can analyze how many repetitions are needed, and moreover compute each set

$$V_\beta(\mathbf{x}) := \{\mathbf{c}' \in C' : |\pi_{\mathbf{c}}(\mathbf{x}) \wedge \mathbf{c}'| = \beta\}$$

in time roughly $|V_\beta(\mathbf{x})|$ using the algorithm from Lemma 7.3.7. The latter fact allows us to construct (and update) the data structure rather efficiently: we can construct the β-buckets $(B_\beta(\mathbf{c}') \cap S)_{\mathbf{c}' \in C'}$ by computing $V_\beta(\mathbf{x})$ for all $\mathbf{x} \in S$. (For technical reasons, we will actually compute a subset of $V_\beta(\mathbf{x})$, and therefore may only partially construct the β-buckets; we address this in our analysis.)

Overall complexity. So far, we have introduced four parameters: the vertex size s of the Johnson graph, the weight v' of the vectors in each $C' \subseteq \mathcal{S}^v_{v'}$, the bucketing parameter β, and the size M' of each set C'. We choose $s = M' = 1/p_{\text{cap}}$, where

$$p_{\text{cap}} = p_{\text{cap}}(v', \beta) := \Pr_{\mathbf{c}' \sim U(\mathcal{S}^v_{v'})}[|\mathbf{x}' \wedge \mathbf{c}'| = \beta]$$

for arbitrary $\mathbf{x}' \in \mathcal{S}^v_\alpha$. The overall complexity of our quantum-walk algorithm therefore depends only on v' and β.

Remark 7.4.9 (Sparsification). The choice $M' = 1/p_{\text{cap}}$ ensures that $\mathbb{E}_{C'}[|V_\beta(\mathbf{x}')|] = Mp_{\text{cap}} = 1$ for $C' \sim \text{RPC}(\mathcal{S}^v_{v'}, \sqrt{v}, M')$. This approach is referred to as *sparsification* in [Laa15; CL21], and results in the lowest time and memory complexity of the quantum-walk algorithm (for fixed v', β).

We obtain the following result, which we prove in Section 7.4.4. The parameters v' and β can be optimized to minimize Equation (7.4) for given (B, n, w, v, α).

Theorem 7.4.10 (`FindSolutions` using quantum walks)**.** *Let (B, n, w, v, α) be well-defined parameters of Problem 7.4.1, and let $p_{\text{sol}} := p(n, w, v, \alpha)$ as in Equation (7.2). Let $v', \beta \in \mathbb{N}$ satisfy $v' \leq v$ and $\beta \leq \min\{\alpha, v'\}$, and define*

$$p_{\text{cap}} := \Pr_{\substack{\mathbf{x}' \sim U(\mathcal{S}^v_\alpha), \\ \mathbf{c}' \sim U(\mathcal{S}^v_{v'})}}[|\mathbf{x}' \wedge \mathbf{c}'| = \beta]$$

$$p_{\text{wedge}} := \min_u \Pr_{\substack{\mathbf{x}', \mathbf{y}' \sim U(\mathcal{S}^v_\alpha), \\ \mathbf{c}' \sim U(\mathcal{S}^v_{v'})}}[|\mathbf{x}' \wedge \mathbf{c}'| = \beta \text{ and } |\mathbf{y}' \wedge \mathbf{c}'| = \beta \mid |\mathbf{x}' \wedge \mathbf{y}'| = u].$$

If $B \geq 1/p_{\text{cap}}$, then there exists a quantum-walk-based algorithm that, given QCRAM access to an instance $B_\alpha(\mathbf{c})$ of Problem 7.4.1, finds all solution pairs in $B_\alpha(\mathbf{c})$ in time

$$\binom{B}{2} p_{\text{sol}} \left(\frac{1}{p_{\text{cap}}} + \frac{p_{\text{cap}}}{\sqrt{p_{\text{sol}}\, p_{\text{wedge}}}} \right) 2^{o(n)} \tag{7.4}$$

using $(1/p_{\text{cap}})\, 2^{o(n)}$ qubits and QQRAM qubits, with success probability $1 - 1/\Omega(\binom{B}{2} p_{\text{sol}})$ over the random choice of $B_\alpha(\mathbf{c})$.

Explicit formulas for p_{cap} and p_{wedge} are known by Lemma 7.2.2. As with the Grover-based algorithm, the subroutine `FindSolutions` in Algorithm 7.2 can be instantiated with the above quantum-walk-based algorithm in order to obtain a quantum algorithm for $\text{NNS}(n, w, N)$, and consequently for $\text{CFP}(n, k, w, N)$ via code sieving (Algorithm 7.1). The resulting time and memory complexities then follow from combining Theorem 7.4.10 with Theorem 7.3.10, as long as the error probability of this quantum-walk-based subroutine is taken into account (Remark 7.3.11).

Remark 7.4.11 (The variant of Theorem 7.4.10 analyzed in Section 7.5). In Section 7.5, we work with a slight variant of Theorem 7.4.10 that still finds a significant fraction (at least $2/n$) of all solution pairs in $B_\alpha(\mathbf{c})$, but appears to give a better time complexity than Equation (7.4). More concretely, note that $p_{\text{sol}} = \sum_{u=\max\{0,2\alpha-v\}}^{\min\{\alpha,w/2\}} p(u)$, where $p(u) := \Pr_{\mathbf{x},\mathbf{y}\sim U(B_\alpha(\mathbf{c}))}[|\mathbf{x}+\mathbf{y}| = w \text{ and } |\mathbf{x}\wedge\mathbf{y}\wedge\mathbf{c}| = u]$. By Lemma 7.4.2, $p(u)$ is maximized by $u^* := \operatorname{argmax}_u \binom{w/2}{u}\binom{w/2}{\alpha-u}^2\binom{n-3w/2}{v-2\alpha+u}$. Moreover, since $u^* \le w/2 \le n/2$, we have $p(u^*) \le p \le p(u^*)n/2$, so at least a fraction $2/n$ of all solution pairs can be found if we only search for those satisfying $|\mathbf{x}\wedge\mathbf{y}\wedge\mathbf{c}| = u^*$. While the quantum algorithm of Theorem 7.4.10 runs a quantum-walk-based algorithm for each u (each finding all solution pairs with $|\mathbf{x}\wedge\mathbf{y}\wedge\mathbf{c}| = u$), in Section 7.5 we consider only the one for u^*, which allows us to replace the role of p_{wedge} in Equation (7.4) by $p_{\text{wedge}}(u^*) := \Pr_{\mathbf{x}',\mathbf{y}'\sim U(\mathcal{S}_\alpha^v),\mathbf{c}'\sim U(\mathcal{S}_{v'}^v)}[|\mathbf{x}'\wedge\mathbf{c}'| = \beta \text{ and } |\mathbf{y}'\wedge\mathbf{c}'| = \beta \mid |\mathbf{x}'\wedge\mathbf{y}'| = u^*]$.

7.4.4 Proof of Theorem 7.4.10

Proof. Consider an instance $B_\alpha(\mathbf{c})$ of Problem 7.4.1 with parameters (B, n, w, v, α), and let $p = p(n, w, v, \alpha)$ be according to Lemma 7.4.2. The goal is to find all solution pairs in $B_\alpha(\mathbf{c})$, that is, all elements of $\mathcal{P} := \bigsqcup_{u=\max\{0,2\alpha-v\}}^{\min\{\alpha,w/2\}} \mathcal{P}_u$, where

$$\mathcal{P}_u := \{(\mathbf{x},\mathbf{y}) \in B_\alpha(\mathbf{c})^2 : |\mathbf{x}+\mathbf{y}| = w, |\pi_{\mathbf{c}}(\mathbf{x})\wedge\pi_{\mathbf{c}}(\mathbf{y})| = u\}.$$

(As before, $\pi_{\mathbf{c}}(\mathbf{x}) \in \mathcal{S}_\alpha^v$ is obtained by projecting $\mathbf{x} \mapsto \mathbf{x}\wedge\mathbf{c}$ onto the support of $\mathbf{c}$, ignoring the remaining $n-v$ coordinates of $\mathbf{x}\wedge\mathbf{c}$.) For each u, we present a quantum algorithm that finds all elements of $\mathcal{P}_u$.

As we mentioned earlier, our quantum algorithm is inspired by a related algorithm for lattice sieving [CL21] (i.e., in $\mathbb{R}^n$), and incorporates the LSF framework

for $\mathbb{F}_2^n$ from [DEEK24] (using the properties of RPCs stated in Section 7.3.3). Fix parameters $\nu', \beta \in \mathbb{N}$ satisfying $\nu' \le \nu$ and $\beta \le \min\{\alpha, \nu'\}$, and define

$$p_{\text{cap}} := \Pr_{\substack{\mathbf{x}' \sim U(\mathcal{S}_\alpha^\nu), \\ \mathbf{c}' \sim U(\mathcal{S}_{\nu'}^\nu)}} [|\mathbf{x}' \wedge \mathbf{c}'| = \beta]$$

$$p_{\text{wedge}}(u) := \Pr_{\substack{\mathbf{x}', \mathbf{y}' \sim U(\mathcal{S}_\alpha^\nu), \\ \mathbf{c}' \sim U(\mathcal{S}_{\nu'}^\nu)}} [|\mathbf{x}' \wedge \mathbf{c}'| = \beta \text{ and } |\mathbf{y}' \wedge \mathbf{c}'| = \beta \mid |\mathbf{x}' \wedge \mathbf{y}'| = u].$$

For $C' \in \mathcal{R}(\mathcal{S}_{\nu'}^\nu, \sqrt{\nu}, 1/p_{\text{cap}})$, we define the following subset of solution pairs

$$\mathcal{P}_u(C') := \{(\mathbf{x}, \mathbf{y}) \in B_\alpha(\mathbf{c})^2 : |\mathbf{x} + \mathbf{y}| = w, |\pi_{\mathbf{c}}(\mathbf{x}) \wedge \pi_{\mathbf{c}}(\mathbf{y})| = u, V_\beta^{\text{tr}}(\mathbf{x}) \cap V_\beta^{\text{tr}}(\mathbf{y}) \neq \emptyset\}$$

where

$$V_\beta^{\text{tr}}(\mathbf{x}) = \begin{cases} V_\beta^{(b)}(\pi_{\mathbf{c}}(\mathbf{x})) & \text{if } |V_\beta^{(b)}(\pi_{\mathbf{c}}(\mathbf{x}))| = 2^{n/\log n} \\ \emptyset & \text{otherwise.} \end{cases} \tag{7.5}$$

Here, $V_\beta^{(b)}(\mathbf{x}') := \{\sigma(\mathbf{c}_1, \ldots, \mathbf{c}_b) \in C : \forall i \in [b], |\sigma(\mathbf{x}')_i \wedge \mathbf{c}_i| = \beta/b\}$ for $C' = \sigma(C^{(1)} \times \cdots \times C^{(b)})$, as defined in Section 7.3.3. (We define $\mathcal{P}_u(C')$ with respect to $V_\beta^{\text{tr}}(\mathbf{x})$, as it allows us to keep control over the time and memory complexity of our algorithm.)

We will design a quantum-walk algorithm that, for a given C', finds all elements of $\mathcal{P}_u(C')$, and show that the theorem statement follows if we apply this algorithm sufficiently many times for different random $C' \sim \text{RPC}(\mathcal{S}_{\nu'}^\nu, \sqrt{\nu}, 1/p_{\text{cap}})$, and repeat this for all $\max\{0, 2\alpha - \nu\} \le u \le \min\{\alpha, w/2\}$. More concretely, we will prove the following claims hold for each $\max\{0, 2\alpha - \nu\} \le u \le \min\{\alpha, w/2\}$:

(*I*) For all $C' \in \mathcal{R}(\mathcal{S}_{\nu'}^\nu, \sqrt{\nu}, 1/p_{\text{cap}})$, there exists a quantum-walk algorithm $\mathtt{QW}_u(C')$ that with probability $1 - 2^{-\omega(n)}$ over $B_\alpha(\mathbf{c})$ finds all elements of $\mathcal{P}_u(C')$ in time

$$T_u(C') = |\mathcal{P}_u(C')| \left(\frac{1}{p_{\text{cap}}} + B \sqrt{\frac{p_{\text{cap}}}{|\mathcal{P}_u(C')|}} \right) 2^{o(n)}$$

using QCRAM access to $B_\alpha(\mathbf{c})$, and $(1/p_{\text{cap}})\, 2^{o(n)}$ qubits and QQRAM qubits.

(*II*) Sample $C'_1, \ldots, C'_{\ell_u} \sim \text{RPC}(\mathcal{S}_{\nu'}^\nu, \sqrt{\nu}, 1/p_{\text{cap}})$ independently for sufficiently large $\ell_u = \frac{p_{\text{cap}}}{p_{\text{wedge}}(u)} 2^{o(n)}$. Then, with probability $1 - 2^{-\omega(n)}$, all $(\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u$ satisfy

$$|\{i \in [\ell_u] : (\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u(C'_i)\}| \in [1, 2^{o(n)}].$$

In particular, $\bigcup_{i\in[\ell_u]} \mathcal{P}_u(C'_i) = \mathcal{P}_u$ and $|\mathcal{P}_u| \le \sum_{i\in[\ell_u]} |\mathcal{P}_u(C'_i)| \le |\mathcal{P}_u| 2^{o(n)}$.

Let us first show that the theorem statement follows from the above claims. Consider the quantum algorithm that applies the following for all u, where $\mathtt{QW}_u$ and ℓ_u are according to (I) and (II):

1. For $i = 1, \dots, \ell_u$:
   (a) Sample $C'_i \sim \mathrm{RPC}(\mathcal{S}^{\upsilon}_{\upsilon'}, \sqrt{\upsilon}, 1/p_{\mathrm{cap}})$
   (b) Compute $\mathcal{P}_u(C'_i)$ using $\mathtt{QW}_u(C'_i)$
2. Output $\bigcup_{i\in[\ell_u]} \mathcal{P}_u(C'_i)$

By (I) and (II), this approach finds all solution pairs in $B_\alpha(\mathbf{c})$, except with probability $1 - 2^{-\omega(n)}$ over $B_\alpha(\mathbf{c})$, in time

$$T = \sum_u \sum_{i\in[\ell_u]} T_u(C'_i) = \left(\sum_u \sum_{i\in[\ell_u]} \frac{|\mathcal{P}_u(C'_i)|}{p_{\mathrm{cap}}} + \sum_u \sum_{i\in[\ell_u]} B\sqrt{|\mathcal{P}_u(C'_i)| p_{\mathrm{cap}}} \right) 2^{o(n)}.$$

(Note that we apply (I) for $\sum_u \ell_u = 2^{O(n)}$ times, so taking a union bound over all applications ensures that the error probability remains $2^{-\omega(n)}$.) By the Cauchy-Schwarz inequality, we then have

$$\sum_{i\in[\ell_u]} \sqrt{|\mathcal{P}_u(C'_i)|} \le \sqrt{\ell_u \sum_{i\in[\ell_u]} |\mathcal{P}_u(C'_i)|} = \sqrt{\ell_u |\mathcal{P}_u|} 2^{o(n)}.$$

For all u, we have $\ell_u \le \frac{p_{\mathrm{cap}}}{p_{\mathrm{wedge}}} 2^{o(n)}$, where $p_{\mathrm{wedge}} := \min_u p_{\mathrm{wedge}}(u)$ as in the theorem statement. Moreover, by Lemma 7.4.5, with probability $1 - 1/\Omega(\binom{B}{2} p_{\mathrm{sol}})$ over $B_\alpha(\mathbf{c})$, we have $\sum_u |\mathcal{P}_u| = |\mathcal{P}| \le O(\binom{B}{2} p_{\mathrm{sol}})$. Combined with another application of the Cauchy-Schwarz inequality (using that the number of possible u is upper bounded by n), we obtain

$$\sum_u \sum_{i\in[\ell_u]} \sqrt{|\mathcal{P}_u(C'_i)|} \le \sqrt{\sum_u \ell_u |\mathcal{P}_u|} 2^{o(n)} \le \sqrt{\frac{\binom{B}{2} p_{\mathrm{sol}} p_{\mathrm{cap}}}{p_{\mathrm{wedge}}}} 2^{o(n)}$$

and thus $\sum_u \sum_{i\in[\ell_u]} B\sqrt{|\mathcal{P}_u(C'_i)| p_{\mathrm{cap}}} = \binom{B}{2} p_{\mathrm{sol}} \frac{p_{\mathrm{cap}}}{\sqrt{p_{\mathrm{sol}} p_{\mathrm{wedge}}}} 2^{o(n)}$. Hence,

$$T = \binom{B}{2} p_{\mathrm{sol}} \left(\frac{1}{p_{\mathrm{cap}}} + \frac{p_{\mathrm{cap}}}{\sqrt{p_{\mathrm{sol}}\, p_{\mathrm{wedge}}}} \right) 2^{o(n)}$$

which matches the time complexity in the theorem statement. Note that this algorithm also has the desired memory complexity.

We start with proving claim (II), as it is a relatively straightforward application of the Chernoff bound (Corollary 2.1.4). Let $(\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u$. By Lemma 7.3.9, we have

$$\begin{aligned}\Pr_{C' \sim \mathrm{RPC}(\mathcal{S}^v_{v'}, \sqrt{v}, 1/p_{\mathrm{cap}})}[(\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u(C')] &= \Pr_{C'}[V^{\mathrm{tr}}_\beta(\mathbf{x}) \cap V^{\mathrm{tr}}_\beta(\mathbf{y}) \neq \emptyset] \\ &\geq \frac{|\mathcal{W}^v_{v'}(\beta \mid \pi_{\mathbf{c}}(\mathbf{x}), \pi_{\mathbf{c}}(\mathbf{y}))|}{|\mathcal{C}^v_{v'}(\beta \mid \pi_{\mathbf{c}}(\mathbf{x}))|} 2^{-o(n)} \\ &\geq \frac{p_{\mathrm{wedge}}(u)}{p_{\mathrm{cap}}} 2^{-o(n)}\end{aligned}$$

by definition of p_{cap} and $p_{\mathrm{wedge}}(u)$. As long as $\ell_u = \frac{p_{\mathrm{cap}}}{p_{\mathrm{wedge}}(u)} 2^{o(n)}$ is sufficiently large, it then follows that all $(\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u$ satisfy

$$\mathbb{E}[|\{i \in [\ell_u] \colon (\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u(C'_i)\}|] = \omega(n)$$

where the expectation is taken over independent $C'_1, \ldots, C'_{\ell_u} \sim \mathrm{RPC}(\mathcal{S}^v_{v'}, \sqrt{v}, 1/p_{\mathrm{cap}})$. Claim (II) then follows from applying Corollary 2.1.4 to each $(\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u$, followed by a union bound over all $|\mathcal{P}_u| = 2^{O(n)}$ solution pairs in $\mathcal{P}_u$.

It remains to prove claim (I). Let $C' \in \mathcal{R}(\mathcal{S}^v_{v'}, \sqrt{v}, 1/p_{\mathrm{cap}})$. By Lemma 7.3.8, there is an algorithm $\mathtt{Dec}$ that, given $\mathbf{x} \in B_\alpha(\mathbf{c})$, computes the set $V^{(b')}_\beta(\mathbf{x})$ in time $(|V^{(b')}_\beta(\mathbf{x})| + 1)2^{o(n)}$, where $b' := \sqrt{v}$. To guarantee that we obtain an algorithm that runs in time $2^{o(n)}$ for all $\mathbf{x}$, we will modify $\mathtt{Dec}$ to compute $V^{\mathrm{tr}}_\beta(\mathbf{x})$ instead. That is, we simply halt $\mathtt{Dec}$ and output $\emptyset$ when it takes too long, which happens only when $|V^{(b')}_\beta(\mathbf{x})| > 2^{n/\log n}$.

We will now present an MNRS-style quantum walk and let $\mathtt{QW}_u(C')$ run this quantum walk $|\mathcal{P}_u(C')|2^{o(n)}$ times, in order to find all elements of

$$\mathcal{P}_u(C') = \{(\mathbf{x}, \mathbf{y}) \in B_\alpha(\mathbf{c})^2 \colon (\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u, V^{\mathrm{tr}}_\beta(\mathbf{x}) \cap V^{\mathrm{tr}}_\beta(\mathbf{y}) \neq \emptyset\}.$$

Graph. Consider the Johnson graph $J(B, s)$ with vertex size $s = 1/p_{\mathrm{cap}}$. We identify its set of vertices with $\mathcal{V} = \{S \subseteq B_\alpha(\mathbf{c}) \colon |S| = s\}$.

Data. Each vertex $S \in \mathcal{V}$ is paired with a data structure $\mathtt{data}(S)$ that facilitates the search for $(\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u(C')$. Specifically, $\mathtt{data}(S)$ stores the relation

$$\{(\mathbf{x}, \mathbf{c}') \in S \times C' \colon \mathbf{c}' \in V^{\mathrm{tr}}_\beta(\mathbf{x})\}. \tag{7.6}$$

For each $\mathbf{x} \in B_\alpha(\mathbf{c})$, the set $V_\beta^{\mathrm{tr}}(\mathbf{x})$ has size $2^{o(n)}$ and can be constructed in time $2^{o(n)}$. This allows us to upper bound the number of pairs stored in each data structure $\mathsf{data}(S)$ by $s\,2^{o(n)}$ (with $s = 1/p_{\mathrm{cap}}$), and to upper bound the cost of maintaining the data structures in a uniform way over all $\mathbf{x}$.

We will store the relation defined in Equation (7.6) via a QQRAM data structure $|\mathsf{data}(S)\rangle$ that allows for efficient insertion and removal of elements, for instance using the data structure presented in [Amb07] or using a quantum radix tree [BJLM13] (with the corrections in [BLPS22]).

Marked vertices. Accordingly, we want to define the set of marked vertices as

$$\mathcal{M}' := \{S \in \mathcal{V} \colon \exists \mathbf{x}, \mathbf{y} \in S, (\mathbf{x}, \mathbf{y}) \in \mathcal{P}_u \text{ and } V_\beta^{\mathrm{tr}}(\mathbf{x}) \cap V_\beta^{\mathrm{tr}}(\mathbf{y}) \neq \emptyset\}$$

as this set consists of all $S \in \mathcal{V}$ with $S^2 \cap \mathcal{P}_u(C') \neq \emptyset$. For technical reasons, we instead let the set of marked vertices be a subset $\mathcal{M} \subseteq \mathcal{M}'$ that we will define below when we discuss the checking cost.

Next, we analyze the different terms in the time complexity of the quantum walk, according to Theorem 5.3.2.

Setup cost S. Constructing a uniform superposition over all basis states $|S\rangle$ with $S \in \mathcal{V}$ takes time $\tilde{O}(s)$. In addition, we need to compute the data for each vertex S. We will construct the QQRAM data structure $|\mathsf{data}(S)\rangle$ by computing, for each $\mathbf{x} \in S$, the set $V_\beta^{\mathrm{tr}}(\mathbf{x})$ of valid center points, and then inserting the corresponding pairs $(\mathbf{x}, \mathbf{c}')$ at the right locations of the quantum data structure. Our definition of $V_\beta^{\mathrm{tr}}(\mathbf{x})$ and choice of vertex size $s = 1/p_{\mathrm{cap}}$ therefore guarantee the following upper bound on the setup cost:

$$\mathsf{S} = s\left(1 + \max_{\mathbf{x} \in B_\alpha(\mathbf{c})} |V_\beta^{\mathrm{tr}}(\mathbf{x})|\right) 2^{o(n)} = \frac{1}{p_{\mathrm{cap}}}\, 2^{o(n)}.$$

Checking cost C. In order to check whether a given vertex $S \in \mathcal{V}$ is marked, we use QQRAM queries to $|\mathsf{data}(S)\rangle$ to do a Grover search (Theorem 5.1.2) over all $\mathbf{c}' \in C'$ and the corresponding pairs of stored vectors $\mathbf{x}, \mathbf{y} \in S$ satisfying $|\pi_{\mathbf{c}}(\mathbf{x}) \wedge \mathbf{c}'| = \beta$ and $|\pi_{\mathbf{c}}(\mathbf{y}) \wedge \mathbf{c}'| = \beta$. A triple $(\mathbf{c}', \mathbf{x}, \mathbf{y})$ counts as a solution if $|\mathbf{x} + \mathbf{y}| = w$ and $|\pi_{\mathbf{c}}(\mathbf{x}) \wedge \pi_{\mathbf{c}}(\mathbf{y})| = u$.

Note that $\mathsf{data}(S)$ stores, for each $\mathbf{c}' \in C'$, a subset of

$$B_\beta(\mathbf{c}') \cap S := \{\mathbf{x} \in S \colon |\pi_{\mathbf{c}}(\mathbf{x}) \wedge \mathbf{c}'| = \beta\}.$$

Namely, it stores precisely the set $B_\beta^{\mathrm{tr}}(\mathbf{c}') \cap S := \{\mathbf{x} \in S : \mathbf{c}' \in V_\beta^{\mathrm{tr}}(\mathbf{x})\}$. For a fixed $S \in \mathcal{V}$ and $\mathbf{c}' \in C'$, Theorem 5.1.2 implies that we can determine whether there exists a solution of the form $(\mathbf{c}', \mathbf{x}, \mathbf{y})$ in time $\sqrt{|B_\beta^{\mathrm{tr}}(\mathbf{c}') \cap S|^2}\, 2^{o(n)}$.

Recall that we have $s = |C'| = 1/p_{\mathrm{cap}}$. By definition of p_{cap}, the expected size of the full set $B_\beta(\mathbf{c}') \cap S$ is $\mathbb{E}_{C'}\left[|B_\beta(\mathbf{c}') \cap S|\right] = sp_{\mathrm{cap}} = 1$. To ensure that the overall checking cost (that is, for all $S \in \mathcal{V}$) is upper bounded by $\sqrt{|C'|}\, 2^{o(n)} = \sqrt{1/p_{\mathrm{cap}}}\, 2^{o(n)}$, we will, for each $S \in \mathcal{V}$, only perform the Grover search for those $\mathbf{c}' \in C'$ with $|B_\beta^{\mathrm{tr}}(\mathbf{c}') \cap S| \leq 2^{n/\log n}$ (the size of this set is stored in $\mathsf{data}(S)$ and can therefore be obtained with one QQRAM query). This may introduce some false negatives (some $S \in \mathcal{M}'$ are possibly not marked), but we will take care of this by adding one more condition to the definition of marked vertices: we will replace $\mathcal{M}'$ by the slightly smaller subset

$$\mathcal{M} := \{S \in \mathcal{M}' : \forall \mathbf{c}' \in C', |B_\beta^{\mathrm{tr}}(\mathbf{c}') \cap S| \leq 2^{n/\log n}\}.$$

We will show below (see Claim 7.4.12) that this restriction barely affects our probability of detecting solution pairs (specifically, ε barely changes). Intuitively, this is because the elements in our input list $B_\alpha(\mathbf{c})$ are independently distributed, so for a fixed $S \in \mathcal{V}$, the size of $|B_\beta(\mathbf{c}') \cap S|$ is close to its expectation (which is 1) with overwhelming probability.

Altogether, this results in the following upper bound on the checking cost:

$$\mathsf{C} = \frac{1}{\sqrt{p_{\mathrm{cap}}}}\, 2^{o(n)}.$$

Update cost U**.** The dominating cost in the update step will be that of updating the data when going from a vertex S to a neighbor S'. By definition of the Johnson graph, such a neighbor S' is of the form $S' = (S \setminus \{\mathbf{x}_{\mathrm{old}}\}) \cup \{\mathbf{x}_{\mathrm{new}}\}$, so we can update $\mathsf{data}(S)$ to $\mathsf{data}(S')$ as follows. First, we compute $V_\beta^{\mathrm{tr}}(\mathbf{x}_{\mathrm{old}})$, so that we can remove each $(\mathbf{x}_{\mathrm{old}}, \mathbf{c}')$ with $\mathbf{c}' \in V_\beta^{\mathrm{tr}}(\mathbf{x}_{\mathrm{old}})$ from $\mathsf{data}(S)$, giving $\mathsf{data}(S \setminus \{\mathbf{x}_{\mathrm{old}}\})$. Then, we compute $V_\beta^{\mathrm{tr}}(\mathbf{x}_{\mathrm{new}})$, so that we can add $(\mathbf{x}_{\mathrm{old}}, \mathbf{c}')$ for all $\mathbf{c}' \in V_\beta^{\mathrm{tr}}(\mathbf{x}_{\mathrm{new}})$, resulting in $\mathsf{data}(S')$. Our definition of $V_\beta^{\mathrm{tr}}(\mathbf{x})$ therefore guarantees the following upper bound on the update cost:

$$\mathsf{U} = \left(1 + \max_{\mathbf{x} \in B_\alpha(\mathbf{c})} |V_\beta^{\mathrm{tr}}(\mathbf{x})|\right) 2^{o(n)} = 2^{o(n)}.$$

Spectral gap δ. As mentioned in Section 5.3, the spectral gap of the Johnson graph $J(B,s)$ is $\delta = \Omega(\frac{1}{s})$ for all $s \geq 1$, and thus $\delta = \Omega(p_{\text{cap}})$.

Probability ε of a vertex being marked. By definition of $\mathcal{M}$, we have

$$\varepsilon = \Pr_{S\sim U(\mathcal{V})}[S \in \mathcal{M}] = \Pr_{S\sim U(\mathcal{V})}[S \in \mathcal{M}'] - \Pr_{S\sim U(\mathcal{V})}[S \in \mathcal{M}' \setminus \mathcal{M}]$$

where (using that all vertices have size $|S| = s = 1/p_{\text{cap}}$)

$$\Pr_{S\sim U(\mathcal{V})}[S \in \mathcal{M}'] = \Pr_{S\sim U(\mathcal{V})}[S^2 \cap \mathcal{P}_u(C') \neq \emptyset] = \Omega\left(\frac{|\mathcal{P}_u(C')|s^2}{B^2}\right) = \Omega\left(\frac{|\mathcal{P}_u(C')|}{B^2 p_{\text{cap}}^2}\right).$$

Since $|\mathcal{P}_u(C')|/(B^2 p_{\text{cap}}^2) \geq 1/2^{O(n)}$, the following claim implies

$$\varepsilon = \Omega\left(\frac{|\mathcal{P}_u(C')|}{B^2 p_{\text{cap}}^2}\right)$$

except with probability $2^{-\omega(n)}$ over the input $B_\alpha(\mathbf{c})$.

Claim 7.4.12. *If $sp_{\text{cap}} \geq 1$ and $|C'| = 2^{O(n)}$, then with probability $1 - 2^{-\omega(n)}$ over the input $B_\alpha(\mathbf{c})$ we have $\Pr_{S\sim U(\mathcal{V})}[S \in \mathcal{M}' \setminus \mathcal{M}] \leq 2^{-\omega(n)}$.*

Proof. By definition of $\mathcal{M}$ and the union bound, we have

$$\Pr_{S\sim U(\mathcal{V})}[S \in \mathcal{M}' \setminus \mathcal{M}] \leq \Pr_{S\sim U(\mathcal{V})}[\exists \mathbf{c}' \in C', |B_\beta(\mathbf{c}') \cap S| > sp_{\text{cap}}\, 2^{n/\log n}].$$

Since the input $B_\alpha(\mathbf{c})$ consists of i.i.d. uniform samples from $\{\mathbf{x} \in \mathcal{S}_w^n : |\mathbf{x} \wedge \mathbf{c}| = \alpha\}$ and $\mathbb{E}_{S\sim U(\mathcal{V})}[|B_\beta(\mathbf{c}') \cap S|] = sp_{\text{cap}}$ for all $\mathbf{c}' \in C'$, the Chernoff bound (Lemma 2.1.3) implies that $|B_\beta(\mathbf{c}') \cap S| \leq sp_{\text{cap}}\, 2^{n/\log n}$ except with probability $2^{-\omega(n)}$. A union bound over all $\mathbf{c}' \in C'$ then implies that the expectation of $\Pr_{S\sim U(\mathcal{V})}[\exists \mathbf{c}' \in C', |B_\beta(\mathbf{c}') \cap S| > sp_{\text{cap}}\, 2^{n/\log n}]$ over $B_\alpha(\mathbf{c})$ is at most $2^{-\omega(n)}$. The claim now follows from Markov's inequality (Lemma 2.1.1). □

Time complexity. By Theorem 5.3.2, this quantum-walk algorithm returns an essentially uniform sample $S \in \mathcal{M}$ in time (omitting subexponential factors)

$$\mathsf{S} + \frac{1}{\sqrt{\varepsilon}}\left(\mathsf{C} + \frac{1}{\sqrt{\delta}}\mathsf{U}\right) = \frac{1}{p_{\text{cap}}} + B\sqrt{\frac{p_{\text{cap}}}{|\mathcal{P}_u(C')|}}$$

where we use that $\mathsf{C} = \frac{1}{\sqrt{\delta}}\mathsf{U} = \frac{1}{\sqrt{p_{\text{cap}}}}\,2^{o(n)}$. A Grover search over all s^2 pairs in S then allows us to find a solution pair in time $1/p_{\text{cap}}$. While ε (specifically, $|\mathcal{P}_u(C')|$) may not be known to the algorithm, it can learn ε by just running the quantum walk for different guesses $\varepsilon = 1/2, 1/4, 1/8, \ldots$ until it succeeds (we can verify solution pairs), which only affects the overall cost by a factor of $2^{o(n)}$.

By the coupon-collector argument, $|\mathcal{P}_u(C')|2^{o(n)}$ repetitions suffice to output all elements of $\mathcal{P}_u(C')$, as each element of $\mathcal{P}_u(C')$ is found with probability $1/|\mathcal{P}_u(C')|$, up to subexponential factors. When the size $|\mathcal{P}_u(C')|$ is not known a priori, we can apply Lemma 6.2.1 (using that the size is always upper bounded by $2^{O(n)}$). The resulting quantum-walk algorithm $\mathtt{QW}_u(C')$ therefore has the desired time complexity (with success probability following from Claim 7.4.12 and Lemma 6.2.1).

Memory complexity. The algorithm $\mathtt{QW}_u(C')$ uses QCRAM access to the input list $B_\alpha(\mathbf{c})$. Moreover, the quantum walk is performed over a superposition of (pairs of) vertices $S \in \mathcal{V}$ and their associated data structure $|\mathtt{data}(S)\rangle$. The amount of quantum memory used by the algorithm is dominated by the vertices and their data. Each vertex consists of s elements (each of which can be represented using $2^{o(n)}$ bits), and $\mathtt{data}(S)$ stores $s\,2^{o(n)}$ elements by construction. Since $s = 1/p_{\text{cap}}$, it follows that the algorithm uses $(1/p_{\text{cap}})\,2^{o(n)}$ qubits, and uses QQRAM access to these qubits.

This completes the proof of claim (II), and thereby the proof of Theorem 7.4.10. □

7.5 Numerical results

This section summarizes our numerical results on the asymptotic complexities of the different algorithms for $\mathrm{NNS}(n, w, N)$ (Problem 7.3.2). By Theorem 7.3.10, the corresponding code-sieving algorithms for $\mathrm{CFP}(n, k, w, N)$ (Problem 7.3.1) have the same asymptotic complexities, provided that k is chosen appropriately. These numerical results are obtained using Python code [.py]. The algorithms we compare are the following:

- Classical: The algorithm from [DEEK24] using the RPC approach described in Section 7.3.3, which is the best known approach in the classical setting.
- Grover: The quantum algorithm obtained by combining Classical with the Grover-based algorithm from Section 7.4.2.
- QW: The quantum algorithm obtained by combining Classical with the quantum-walk algorithm from Section 7.4.3 using the additional layer of locality-sensitive filtering. (See Remark 7.4.11.)

Figure 7.3 shows the asymptotic runtimes of the algorithms for $\mathrm{NNS}(n, w, N)$ in $\mathbb{F}_2^n$. The figures are obtained by calculating the asymptotic runtime in 100 equidistant values of $\omega := w/n$ ranging in $[0, 0.5)$, taking $N = \Theta\left(\binom{n}{w} / \left(\binom{w}{w/2}\binom{n-w}{w/2}\right)\right)$.

Table 7.1 shows the asymptotic runtimes and memory complexities of the algorithms for $\mathrm{NNS}(n, w, N)$ in $\mathbb{F}_2^n$. Because our work is inspired by quantum algorithms for lattice sieving that address a similar NNS problem over $\mathbb{R}^n$, we also list the complexities of the corresponding lattice analogs. The listed results for $\mathbb{F}_2^n$ correspond to the *hardest instances*, that is, those values of ω for which the runtime curve in Figure 7.3 reaches its peak. (The lattice algorithms consider a NNS problem on the unit sphere in $\mathbb{R}^n$, so the norm is always fixed.) The table distinguishes different types of memory: classical bits (m_C), qubits (m_Q), QCRAM bits (m_{QCRAM}), and QQRAM bits (m_{QQRAM}), where each is listed as a constant m such that the memory complexity is $2^{mn+o(n)}$.

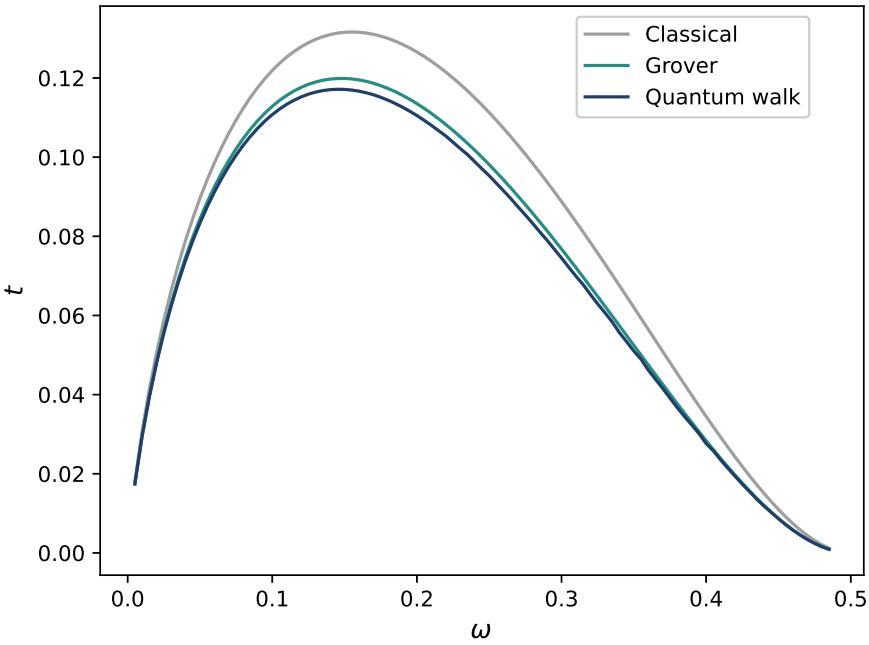


Figure 7.3: Asymptotic runtimes $2^{tn+o(n)}$ for $\mathrm{NNS}(n, w, N)$, where $w = \omega n$ and $N = \Theta\left(\binom{n}{w} \Big/ \left(\binom{w}{w/2}\binom{n-w}{w/2}\right)\right)$ for $\omega \in [0, 0.5)$.

Algorithm	t	m_C	m_Q	m_{QCRAM}	m_{QQRAM}
Codes					
Classical [DEEK24]	0.132	0.093	—	—	—
Grover	0.120	0.094	0	0.026	—
QW	0.117	0.094	0.023	0.036	0.023
Lattices					
Classical [BDGL16]	0.292	0.208	—	—	—
Grover [Laa15]	0.265	0.208	0	0.058	—
QW [CL21, Sec. 5]	0.257	0.208	0.050	0.077	0.050

Table 7.1: The exponents of the asymptotic runtime (i.e., t such that runtime is $2^{tn+o(n)}$) and corresponding memory exponents (i.e., m such that memory usage is $2^{mn+o(n)}$) for NNS in the code-based and lattice-based settings, where for the former we choose the weight ω to correspond to the hardest instances (i.e., to yield the highest runtime for the algorithm considered). A dash "—" means that the memory type is not used.

The number 0.132 in Table 7.1 refers to the runtime of the algorithm from [DEEK24] for NNS (and code sieving), and is not explicitly stated in their paper. Note that their table [DEEK24, Table 1] instead presents the (different) asymptotic complexity of the resulting sieving-based ISD algorithm.

Remark 7.5.1. The repository also includes code for estimating the complexity of a variant of QW where we incorporate the reusable quantum-walk techniques from [BCSS23] (in a similar way as they apply their techniques to [CL21]). Similar to their result on lattice sieving, we only obtain a minor improvement, namely an exponent of $t = 0.1169$ instead of 0.1171. (In lattice sieving, [BCSS23] obtain an exponent of $t = 0.2563$ instead of 0.2570.) We leave the elaboration of the analytical details of this approach as potential future work.

7.5.1 Some observations

Our numerical results show that our quantum algorithms provide a speedup in comparison with the classical runtime [DEEK24]. While this is unsurprising, it is not necessarily guaranteed. Despite the structural differences between codes and lattices (particularly, their metric), we observe that similar techniques yield comparable improvements in the asymptotic runtime for sieving in both contexts. Grover's algorithm applied to the classical version delivers the most substantial quantum speedup. Nevertheless, our quantum-walk algorithm outperforms the version using Grover. It appears that fundamentally different quantum techniques are needed to obtain more significant improvements.

While our work only focused on improving the asymptotic time complexities of sieving algorithms, it is also important to quantify the memory used by these algorithms, and minimize it. In particular, note that the improved runtime obtained when moving from classical to Grover, and subsequently to the quantum-walk algorithm, comes at the cost of an increased memory complexity and extra assumption about the memory model. Specifically, the Grover variant uses quantum memory and QCRAM access; quantum walks additionally use QQRAM, which is an even stronger assumption about the memory model. Note that this trade-off between improved runtime and assumptions on the memory model is consistent with other quantum algorithms in code-based (and lattice-based) cryptanalysis.

7.6 Quantum analogs of sieving-based ISD

Information-set decoding (ISD) algorithms are the most efficient generic attacks against the decoding problem for a wide range of parameters.[8] Each of these can be seen as an improvement of an algorithm from Prange [Pra62]. We view ISD algorithms as a framework, as formulated in [FS09].

The authors of [GJN23; DEEK24] combined the ISD framework with code sieving, resulting in a sieving-based ISD algorithm that we call `SievingISD`. In this section, we will analyze a quantum analog of `SievingISD` obtained using our quantum algorithms for code sieving as a subroutine.

Remark 7.6.1. ISD algorithms are used to solve a random instance of CFP(n, k, w) with 1 solution in expectation, achieved by invoking an oracle for CFP(n', k, w', N) for some n', w', N. To be consistent with prior work on ISD, we will denote the global ISD parameters by n, w in this section, and use n', w' for the parameters corresponding to the CFP instances solved by the oracle (which were denoted by n, w in the previous sections).

7.6.1 ISD and SievingISD

Let us first review the ISD framework. We already mentioned that the central computational problem underlying the security of code-based cryptography is the Syndrome Decoding Problem, SDP(n, k, w) (Problem 7.2.4). As explained in Section 7.2.2, we can instead consider algorithms for CFP(n, k, w) (Problem 7.2.5), which asks to find a codeword $\mathbf{c} \in \mathcal{C}$ of weight w in a given $[n, k]$ binary linear code $\mathcal{C}$. Cryptographically relevant instances of this problem correspond to those where $\mathcal{C}$ is a random $[n, k]$ binary linear code, and w is chosen such that there is only one solution to the problem on average. We refer to these instances as a *unique decoding instance* with parameters (n, k, w).

Algorithms in the ISD framework solve CFP(n, k, w) by repeatedly solving related instances of CFP(n', k, w', N) for some $n' \leq n, w' \leq w$, and $N \geq 1$. An ISD algorithm therefore makes use of an oracle $\mathcal{A}$ for CFP(n', k, w', N). We present the resulting ISD algorithm in Algorithm 7.3, using the formulation in [DEEK24]. We use the notation $\pi_{\mathbf{x}}(\cdot)$ for code puncturing, which is defined below.

[8] For a more specific range of parameters, there exist more efficient attacks such as statistical decoding [CDMT22].

Definition 7.6.2 (Code puncturing, $\pi_{\mathbf{x}}(\cdot)$)**.** For an $[n, k]$ binary linear code $\mathcal{C}$ and a binary vector $\mathbf{x} \in \mathbb{F}_2^n$ with $|\mathbf{x}| = n'$, we define by $\pi_{\mathbf{x}} : \mathbf{c} \mapsto \mathbf{c} \wedge \mathbf{x}$ the puncturing function relative to the support of $\mathbf{x}$, and we define $\pi_{\mathbf{x}}(\mathcal{C})$ to be the corresponding punctured code, which is an $[n', k]$ binary linear code.

Algorithm 7.3: Information-set decoding (ISD)

Input: $[n, k]$ linear code $\mathcal{C}$
Weight w
Oracle $\mathcal{A}$ for CFP(n', k, w', N), where $n' > k$, w', N are fixed

Output: Codeword $\mathbf{c} \in \mathcal{C}$ with $|\mathbf{c}| = w$

1: **while** True **do**
2: Sample $\mathbf{x} \sim U(\mathcal{S}_{n'}^n)$ and repeat if $\dim(\pi_{\mathbf{x}}(\mathcal{C})) \neq k$
3: $L \leftarrow \mathcal{A}(\pi_{\mathbf{x}}(\mathcal{C}), w')$
4: **if** $\exists\, \mathbf{y} \in L\colon |\pi_{\mathbf{x}}^{-1}(\mathbf{y})| = w$ **then**
5: **return** $\pi_{\mathbf{x}}^{-1}(\mathbf{y})$

The first step in the while-loop of Algorithm 7.3 succeeds after an expected number of $O(1)$ samples, because $\mathbf{x} \sim U(\mathcal{S}_{n'}^n)$ satisfies $\dim(\pi_{\mathbf{x}}(\mathcal{C})) = k$ with constant probability [Coo00]. The overall success probability of one while-loop can also be calculated, resulting in the following complexity, as (for instance) shown in [DEEK24].

Theorem 7.6.3 (ISD [DEEK24, Theorem 3.1])**.** *Let $\mathcal{C}$ be a unique decoding instance with parameters (n, k, w). For integers n', w', N satisfying $n' > k$, $n' \geq w'$, and $N \leq \binom{n'}{w'}/2^{n'-k}$, let $\mathcal{A}$ be an algorithm that returns N independent and uniformly random weight-w' codewords in a given $[n', k]$ binary linear code. If $\mathcal{A}$ has expected runtime $T_{\mathcal{A}}$, then Algorithm 7.3 returns a weight-w codeword in $\mathcal{C}$, if one exists, in expected time*

$$\tilde{O}\left(\frac{T_{\mathcal{A}}}{p_1 p_2}\right)$$

where $p_1 := \binom{n'}{w'}\binom{n-n'}{w-w'}/\binom{n}{w}$ and $p_2 := N2^{n'-k}/\binom{n'}{w'}$.

We refer to `SievingISD` as an ISD algorithm in the form of Algorithm 7.3 that uses a sieving algorithm (Algorithm 7.1) as input oracle $\mathcal{A}$. By Lemma 7.3.4 and the subsequent discussion, this requires $N = \Omega\left(\binom{n'}{w'}/\left(\binom{w'}{w'/2}\binom{n'-w'}{w'/2}\right)\right)$. Moreover, the analysis of the `SievingISD` algorithm in [DEEK24] is heuristic, as it relies on the binary-sieve heuristic mentioned in Section 7.3.1.

Remark 7.6.4. As discussed in Section 7.3, the upper bound on N in Theorem 7.6.3 is to ensure that there exist N codewords on average for a random code $[n', k]$ binary code. Note that it ensures $p_2 \leq 1$.

7.6.2 Quantum analogs of ISD and SievingISD

We will now discuss quantum analogs of the ISD framework and `SievingISD`. Prior quantum ISD algorithms [Ber10; KT17; Kir18] are based on the idea that one can apply amplitude amplification (AA) (Section 5.2) to essentially obtain a quadratic improvement over the required number of while-loops in Algorithm 7.3 to succeed. The subroutine $\mathcal{A}$ in Algorithm 7.3 is also allowed to be quantum, which may lead to further speedups. We therefore directly obtain the following quantum analog of [DEEK24, Theorem 3.1]. Here, and in the remainder of Section 7.6, we only focus on the runtime of (quantum) ISD algorithms, not their space usage.

Theorem 7.6.5 (Quantum ISD). *Let $\mathcal{C}$ be a unique decoding instance with parameters (n, k, w). For integers n', w', N satisfying $n' > k$, $n' \geq w'$, and $N \leq \binom{n'}{w'}/2^{n'-k}$, let $\mathcal{A}$ be a classical or quantum algorithm that returns N independent and uniformly random weight-w' codewords in a given $[n', k]$ binary linear code. If $\mathcal{A}$ has expected runtime $T_{\mathcal{A}}$, then there exists a quantum algorithm that returns a weight-w codeword in $\mathcal{C}$, if one exists, in expected time*

$$\tilde{O}\left(\frac{T_{\mathcal{A}}}{\sqrt{p_1 p_2}}\right)$$

where $p_1 := \binom{n'}{w'}\binom{n-n'}{w-w'}/\binom{n}{w}$ and $p_2 := N2^{n'-k}/\binom{n'}{w'}$.

Proof. (This result is folklore, but we sketch the proof for completeness.) The proof of [DEEK24, Theorem 3.1] shows that one while-loop of Algorithm 7.3 succeeds with probability $p_1 p_2$. Since we can efficiently verify solutions (that is, whether a given vector $\mathbf{x} \in \mathbb{F}_2^n$ belongs to $\mathcal{C}$ and is of weight w), amplitude amplification (Theorem 5.2.1) yields a quantum analog of Algorithm 7.3 that succeeds after $\tilde{O}(\frac{1}{\sqrt{p_1 p_2}})$ repetitions of the while-loop. The time complexity of one while-loop is dominated by the runtime of algorithm $\mathcal{A}$, so the theorem follows by assumption on $T_{\mathcal{A}}$. □

Analogous to the classical setting, if $N = \Omega\left(\binom{n'}{w'}/\left(\binom{w'}{w'/2}\binom{n'-w'}{w'/2}\right)\right)$, then we can instantiate the subroutine $\mathcal{A}$ with a (classical or quantum) sieving algorithm. This yields a natural quantum analog of `SievingISD`, and we will refer to it as *quantum* `SievingISD`. It turns out that quantum `SievingISD` does not improve over prior quantum ISD algorithms, as we will show in the next section.

7.6.3 Limitations of quantum SievingISD

We numerically illustrate that, contrary to the classical setting, quantum `SievingISD` cannot do better than the quantum ISD algorithm that we will refer to as *quantum Prange*. Quantum Prange [Ber10] is obtained by adding AA on top of the classical Prange algorithm. It was the first quantum ISD algorithm, and therefore a natural starting point for comparison.

The following lemma states the time complexity of quantum Prange. More recent quantum ISD algorithms improve upon this complexity [KT17; Kir18].

Lemma 7.6.6 (Quantum Prange [Ber10])**.** *Let $\mathcal{C}$ be a unique decoding instance with parameters (n, k, w). There exists a quantum algorithm that returns a weight-w codeword in $\mathcal{C}$, if one exists, in time*

$$\tilde{O}\left(\sqrt{\frac{\binom{n}{w}}{\binom{n-k}{w}}}\right).$$

Let us first formalize our claim that quantum `SievingISD` cannot do better than quantum Prange. Recall that any (classical or quantum) `SievingISD` algorithm requires N to satisfy

$$N = \Omega\left(\frac{\binom{n'}{w'}}{\binom{w'}{w'/2}\binom{n'-w'}{w'/2}}\right) \quad \text{and} \quad N = O\left(\frac{\binom{n'}{w'}}{2^{n'-k}}\right) \tag{7.7}$$

where the upper bound is imposed by the ISD framework itself (see Remark 7.6.4), and the lower bound is due to the use of a *sieving* subroutine of the form Algorithm 7.1 (recall Lemma 7.3.4). This lower bound turns out to be a bottleneck in being able to improve over quantum Prange, as the following claim (and its justification) illustrates.[9]

Claim 7.6.7. *Let $\mathcal{C}$ be a unique decoding instance with parameters (n, k, w). For all integers n', w', N satisfying Equation (7.7), there exists no (classical or quantum) algorithm for the oracle $\mathcal{A}$ in Algorithm 7.3 that results in a quantum ISD algorithm with better runtime than quantum Prange [Ber10].*

Justification of the claim. A trivial lower bound on the runtime of *any* algorithm for the oracle $\mathcal{A}$ in Algorithm 7.3 (with corresponding parameters n', w', N) is given by N, since the framework requires it to output N solutions. Given any such algorithm $\mathcal{A}$ and parameters n', w', N satisfying Equation (7.7), the runtime T of the resulting quantum ISD algorithm must therefore satisfy

$$T \geq \frac{N}{\sqrt{p_1 p_2}} = \sqrt{\frac{N\binom{n}{w}}{2^{n'-k}\binom{n-n'}{w-w'}}}$$

where $p_1 := \binom{n'}{w'}\binom{n-n'}{w-w'}/\binom{n}{w}$ and $p_2 := N2^{n'-k}/\binom{n'}{w'}$ as in Theorem 7.6.5. For fixed n', w', this lower bound on T is minimal when N is as small as possible, i.e., when

[9] Since our justification of the claim is partly numerical, we do not refer to it as a "theorem".

$N = \Theta(N_{n',w'})$ for $N_{n',w'} := \binom{n'}{w'} / \left(\binom{w'}{w'/2}\binom{n'-w'}{w'/2}\right)$. In the remainder, we will leave out constant factors and assume for simplicity that $N = N_{n',w'}$. It follows that T is lower bounded by

$$\min_{\substack{(n',w',N) \\ \text{satisfying} \\ \text{Equation (7.7)}}} \sqrt{\frac{N\binom{n}{w}}{2^{n'-k}\binom{n-n'}{w-w'}}} = \min_{\substack{(n',w') \\ \text{satisfying} \\ \binom{w'}{w'/2}\binom{n'-w'}{w'/2} \geq 2^{n'-k}}} \sqrt{\frac{\binom{n'}{w'}\binom{n}{w}}{\binom{w'}{w'/2}\binom{n'-w'}{w'/2}2^{n'-k}\binom{n-n'}{w-w'}}}. \quad (7.8)$$

To prove the claim, it suffices to show that this is lower bounded by the runtime $\sqrt{\binom{n}{w}/\binom{n-k}{w}}$ of quantum Prange.

Using code from our repository [.py], we numerically minimized Equation (7.8), which shows that, for all n', w' satisfying $\binom{w'}{w'/2}\binom{n'-w'}{w'/2} \geq 2^{n'-k}$, we asymptotically have

$$\min_{\substack{(n',w') \\ \text{satisfying} \\ \binom{w'}{w'/2}\binom{n'-w'}{w'/2} \geq 2^{n'-k}}} \sqrt{\frac{\binom{n'}{w'}\binom{n}{w}}{\binom{w'}{w'/2}\binom{n'-w'}{w'/2}2^{n'-k}\binom{n-n'}{w-w'}}} \geq \sqrt{\frac{\binom{n}{w}}{\binom{n-k}{w}}}$$

so Equation (7.8) is indeed lower bounded by the runtime of quantum Prange. □

In fact, the optimal values obtained through numerical optimization essentially correspond to the values characterizing quantum Prange: $n' = k$ and $w' = 0$.

Remark 7.6.8 (Comparison with the classical setting). The same argument does not apply to *classical* `SievingISD` (unsurprisingly, since [DEEK24] have shown that `SievingISD` outperforms classical Prange). In the classical setting, for all parameters n', k, w', N, a trivial lower bound on the runtime of the oracle $\mathcal{A}$ is again N, giving a (possibly non-tight) lower bound on the time complexity of classical `SievingISD` of $N/(p_1 p_2)$. Unlike in the quantum setting, the dependency (specifically, the lower bound) on N disappears as it is canceled out by p_2, namely $N/(p_1 p_2) = \binom{n}{w} / \left(\binom{n-n'}{w-w'}2^{n'-k}\right)$ since p_2 is linear in N. This argument also applies to other subroutines $\mathcal{A}$ that may be considered within an ISD algorithm (Algorithm 7.3): in the quantum setting, any lower bound on N induces a lower bound on the overall time complexity of the resulting quantum ISD algorithm (in the current formulation where $\mathcal{A}$ is required to output N elements).

7.6.4 On overcoming the limitations

The previous section illustrates limitations of the presented quantum `SievingISD` framework and implies that the framework should be adapted to outperform quantum Prange [Ber10]. Specifically, the main bottleneck appears to be the *lower bound* on the output size N of the sieving subroutine imposed by Lemma 7.3.4 (where we emphasize that in this section we use n', w' to specify the dimension and weight). Recall that the factors $T_{\mathcal{A}}$ and p_2 in the expected runtime $\tilde{O}(T_{\mathcal{A}}/\sqrt{p_1 p_2})$ of a quantum ISD algorithm both depend on N. Here, we present two natural approaches for potentially overcoming these limitations and explain why neither of them works.

Approach 1: From pairs to tuples

So far, we considered the task of finding pairs $(\mathbf{x}_1, \mathbf{x}_2)$ of vectors in our input list L such that $|\mathbf{x}_1 + \mathbf{x}_2| = w'$ (called *solution pairs*). To guarantee the existence of $N = |L|$ solution pairs, Lemma 7.3.4 implied the lower bound on N. Inspired by lattice-based cryptanalysis, an idea to obtain a smaller lower bound on N is to focus instead on finding t-tuples (for some $t > 2$): in lattice tuple sieving (recall Chapter 6), considering t-tuples for $t > 2$ allows for reducing the memory complexity.

Specifically, for $t \geq 2$, we say that a *t-sieving algorithm* is an algorithm constructed similarly to the 2-sieving algorithm in Algorithm 7.1, but generalized to t-tuples. That is, in each sieving iteration, the algorithm is instructed to find N tuples $(\mathbf{x}_1, \ldots, \mathbf{x}_t) \in L^t$ satisfying $|\mathbf{x}_1 + \cdots + \mathbf{x}_t| = w'$ given a list L of N vectors in $\mathcal{S}^{n'}_{w'}$. We refer to such tuples as *t-solutions*. For $t = 2$, we recover the original setting in which the algorithm searches for solution pairs.

When $L \sim U(\mathcal{S}^{n'}_{w'}, N)$, the expected number of t-solutions in L is given by $N^t p^{(t)}$, where $p^{(t)}$ denotes the probability that a t-tuple of independent and uniformly random vectors in $\mathcal{S}^{n'}_{w'}$ forms a t-solution. To ensure that, on average, there exist at least N t-solutions, the list size N thus needs to satisfy $N^t p^{(t)} \geq N$, which may result in a reduced lower bound on the output size N compared to the $t = 2$ case.

We refer to *quantum t-sieving ISD* as the resulting quantum ISD algorithm where we instantiate the oracle $\mathcal{A}$ with a t-sieving algorithm, where we recall that $\mathcal{A}$ aims to solve $\mathrm{CFP}(n', k, w', N)$. We keep the meaning of p_1 and p_2 from Theorem 7.6.5, and write $p_2 = Nq_2$ to highlight that p_2 depends on the output size

N,whereas p_1 and q_2 do not depend on N or t. The expected runtime of quantum t-sieving ISD is then $\tilde{O}\left(T_{\mathcal{A}}/\sqrt{p_1 q_2 N}\right)$, where $T_{\mathcal{A}}$ is the expected runtime of $\mathcal{A}$.

While the use of t-tuples for $t > 2$ may reduce the lower bound on N, we obtain the following lower bound on the runtime of quantum t-sieving ISD under an assumption that we expect to hold in general. The assumption in Proposition 7.6.9 holds, for instance, if the runtime of every t-sieving algorithm is lower bounded by the optimal runtime of 2-sieving (since $1/p^{(2)}$ is a lower bound on the output size of a 2-sieving algorithm).

Proposition 7.6.9 (A lower bound on quantum t-sieving ISD)**.** *Assume that the runtime of any t-sieving algorithm is at least $1/p^{(2)}$. Then all quantum ISD algorithms that use the aforementioned t-sieving algorithm as oracle $\mathcal{A}$ have expected runtime $\Omega\left(1/\sqrt{p_1 q_2 p^{(2)}}\right)$.*

By Section 7.6.3, the latter is (asymptotically) never better than the runtime of quantum Prange. Hence, we can conclude that, under the stated assumption, quantum t-sieving ISD does no better than quantum Prange for all $t \geq 2$.

Proof. We will omit writing $\tilde{O}(\cdot)$ throughout the proof. We analyze the cases $N \leq 1/p^{(2)}$ and $N \geq 1/p^{(2)}$ separately, where N (as usual) denotes the output size of $\mathcal{A}$. First, suppose that $N \leq 1/p^{(2)}$. By the assumption, the runtime $T_{\mathcal{A}}$ of any t-sieving algorithm is at least $1/p^{(2)}$, so the runtime of the resulting quantum ISD algorithm is lower bounded by $T_{\mathcal{A}}/\sqrt{p_1 q_2 N} \geq 1/\left(p^{(2)}\sqrt{p_1 q_2 N}\right) \geq 1/\sqrt{p_1 q_2 p^{(2)}}$. On the other hand, if $N \geq 1/p^{(2)}$, then $T_{\mathcal{A}}/\sqrt{p_1 q_2 N} \geq \sqrt{N}/\sqrt{p_1 q_2} \geq 1/\sqrt{p_1 q_2 p^{(2)}}$ since $T_{\mathcal{A}} \geq N$. □

Approach 2: Varying list sizes

The lower bound on the output size N of the sieving subroutine comes from maintaining the same list size N throughout each iteration in the sieving algorithm. A natural question to ask is whether it would be possible to vary the list size in order to overcome this intrinsic limitation of quantum `SievingISD`.

We propose the following approach for varying list sizes. Suppose that for given parameters n', w' there exists a (classical or quantum) algorithm for the subroutine $\mathcal{A}$ in the quantum ISD algorithm that iteratively applies a sieving step of the following form, where the values N_i are arbitrary. It starts by sampling a list L_0 of N_0 independent and uniformly random vectors from $\mathcal{S}^{n'}_{w'}$. For iteration $i = 1$

up to $i = n' - k$, the algorithm finds N_i pairs $(\mathbf{x}_1, \mathbf{x}_2) \in L_{i-1}^2$ satisfying $|\mathbf{x}_1 + \mathbf{x}_2| = w'$, yielding a new list L_i of size N_i. The expected number of solution pairs is $N_{i-1}^2 p$, where $p := \binom{w'}{w'/2}\binom{n'-w'}{w'/2}/\binom{n'}{w'}$ is the probability that a uniformly random pair is a solution pair (previously denoted by $p^{(2)}$). Therefore, we will only consider N_i satisfying $N_i \leq N_{i-1}^2 p$.

Remark 7.6.10. If $N_i = N_0$ for all i, then we get the same lower bound on the list size as we considered so far, namely $1/p$, resulting in the original setting. We now instead consider the situation where the N_i's might differ.

We now sketch the proof that there is no choice of $N_0, \ldots, N_{n'-k}$ satisfying $N_i \leq N_{i-1}^2 p$ for which this algorithm yields runtime better than quantum Prange (Lemma 7.6.6).

We start by observing that the overall runtime of this quantum ISD algorithm is essentially $T := T_{\max}/\sqrt{p_1 q_2 N_{n'-k}}$, where $T_{\max}$ is the maximum among the runtimes of the iterations, and p_1 and $p_2 = q_2 N_{n'-k}$ are the probabilities from Theorem 7.6.5. (Note that the output size $N_{n'-k}$ was previously denoted by N.) Since $T_{\max} \geq N_{\max} := \max_{i \geq 0} N_i$, we have $T \geq N_{\max}/\sqrt{p_1 q_2 N_{n'-k}}$. Recall from Section 7.6.3 that, for all allowed (n', w'), $1/\sqrt{p_1 q_2 p}$ is asymptotically lower bounded by the runtime of quantum Prange. Therefore, it suffices to show that there is no choice of $N_0, \ldots, N_{n'-k}$ such that $N_{\max}/\sqrt{N_{n'-k}} < 1/\sqrt{p}$.

Suppose for contradiction there is a choice of $N_0, \ldots, N_{n'-k}$ satisfying $N_i \leq N_{i-1}^2 p$ for all $i \geq 1$ and $N_{\max}/\sqrt{N_{n'-k}} < 1/\sqrt{p}$. We analyze the two cases $N_{n'-k} \geq \frac{1}{p}$ and $N_{n'-k} < \frac{1}{p}$ separately. We start with the case $N_{n'-k} \geq 1/p$. Then $N_{\max}/\sqrt{N_{n'-k}} \geq \sqrt{N_{n'-k}} \geq 1/\sqrt{p}$.

It remains to consider the case $N_{n'-k} < 1/p$. If $N_0 \geq 1/p$, then $N_0 > N_{n'-k}$, so $N_{\max}/\sqrt{N_{n'-k}} \geq N_0/\sqrt{N_{n'-k}} > \sqrt{N_0} \geq 1/\sqrt{p}$. Finally, consider the case that $N_0 < 1/p$. Note that $N_i \leq N_{i-1}^2 p$ for all $i \geq 1$ implies that $N_i \leq N_0^{2^i} p^{2^i - 1}$ for all $i \geq 0$. In particular, the size of the output list satisfies $N_{n'-k} \leq N_0^{2^{n'-k}} p^{2^{n'-k}-1}$. Therefore, we obtain that

$$\frac{N_{\max}}{\sqrt{N_{n'-k}}} \geq \frac{N_0}{\sqrt{N_{n'-k}}} \geq \sqrt{\frac{N_0^2}{N_0^{2^{n'-k}} p^{2^{n'-k}-1}}} = \frac{1}{\sqrt{p}} \frac{1}{\sqrt{N_0^{2^{n'-k}-2} p^{2^{n'-k}-2}}} > \frac{1}{\sqrt{p}}$$

since $N_0 p < 1$. That is, in all possible cases we showed that $N_{\max}/\sqrt{N_{n'-k}} \geq 1/\sqrt{p}$, so we reached a contradiction. We conclude that there is no suitable choice for the N_i that enables to outperform quantum Prange.

7.7 Discussion

The recent introduction of sieving techniques to code-based cryptanalysis naturally raised the question of whether this may lead to faster quantum attacks on code-based cryptography, especially given the resemblance to lattice sieving, where quantum speedups have been shown. The results in this chapter establish that code-based schemes remain resilient to quantum attacks based on code sieving, despite quantum speedups for the code-sieving subroutine itself. As we explained in Section 7.6.3, this discrepancy between the performance of quantum code sieving and the performance of the resulting quantum sieving-based ISD algorithm may be attributed to the current code-sieving framework, specifically the imposed lower bound on the number N of elements output by the code-sieving subroutine.

It is important to briefly explain why this discrepancy does *not* arise in the classical setting (recall Remark 7.6.8), which can be understood from the use of amplitude amplification in quantum ISD algorithms. More precisely, to solve a decoding problem in an $[n, k]$ binary linear code, ISD algorithms use a subroutine $\mathcal{A}$ that solves a random instance of $\mathrm{CFP}(n', k, w', N)$ (Problem 7.3.1), where n', w', N are parameters that should be optimized to minimize the overall complexity. This subroutine $\mathcal{A}$ is called many times (on different random instances) in order to solve the target decoding problem. The overall complexity of an ISD algorithm then depends both on the expected runtime $T_{(n',w',N)}$ of $\mathcal{A}$, and on the probability $p_{(n',w',N)}$ that a single call to $\mathcal{A}$ succeeds.

In the quantum setting, we can consider a *quantum* subroutine $\mathcal{A}^{(Q)}$ whose expected runtime $T^{(Q)}_{(n',w',N)}$ may be smaller than that of $\mathcal{A}$, and amplitude amplification yields essentially a quadratic speedup in the number of calls to $\mathcal{A}^{(Q)}$. Roughly speaking, this implies that in the classical setting the parameters (n', w, N) should be optimized to minimize

$$\frac{1}{p_{(n',w',N)}} T_{(n',w',N)}$$

whereas in the quantum setting they should minimize

$$\frac{1}{\sqrt{p_{(n',w',N)}}} T^{(Q)}_{(n',w',N)}.$$

If $T^{(Q)}_{(n',w',N)} \gg \sqrt{T_{(n',w',N)}}$ (which appears to be a common scenario for the subroutines in state-of-the-art ISD approaches), then the parameters that are optimal in the classical setting may not be optimal in the quantum setting. In this

chapter, we have shown that this situation occurs for the code-sieving subroutine from [DEEK24], and for natural modifications of it. As mentioned in Remark 7.6.8, our arguments apply more generally to other subroutines for which there is a large lower bound on their output size N.

It therefore remains an important open question whether existing quantum ISD algorithms can be further improved. Are there ways to modify the existing code-sieving framework to enable faster quantum attacks on code-based cryptography? Are there promising quantum variants of ISD that do not require a subroutine to output N elements? Finally, it would be interesting to identify other applications of our quantum algorithms for the Near-Neighbor Search problem in $\mathbb{F}_2^n$ (endowed with the Hamming metric) beyond code sieving.

Bibliography

[AAAA+25] Rajeev Acharya et al. “Quantum Error Correction Below the Surface Code Threshold”. In: *Nature* 638.8052 (2025), pp. 920–926 (cited on p. 7).

[Aar09] Scott Aaronson. “Quantum Copy-Protection and Quantum Money”. In: *Proceedings of the IEEE Conference on Computational Complexity (CCC 2009)*. IEEE, July 2009, pp. 229–242. URL: http://dx.doi.org/10.1109/CCC.2009.42 (cited on p. 8).

[ABCC+24] Gorjan Alagic, Maxime Bros, Pierre Ciadoux, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Hamilton Silberg, Daniel Smith-Tone, and Noah Waller. *Status Report on the Second Round of the Additional Digital Signature Schemes for the NIST Post-Quantum Cryptography Standardization Process*. NIST Interagency/Internal Report (NISTIR) 8528. National Institute of Standards and Technology, Oct. 2024 (cited on p. 10).

[ABCC+25] Gorjan Alagic, Maxime Bros, Pierre Ciadoux, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Hamilton Silberg, Daniel Smith-Tone, and Noah Waller. *Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process*. NIST Interagency/Internal Report (NISTIR) 8545. National Institute of Standards and Technology, Mar. 2025 (cited on p. 10).

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. “A Subfield Lattice Attack on Overstretched NTRU Assumptions – Cryptanalysis of Some FHE and Graded Encoding Schemes”. In: *Advances in Cryptology – CRYPTO 2016*. Ed. by Matthew Robshaw and Jonathan Katz.

Vol. 9814. Lecture Notes in Computer Science. Springer, 2016, pp. 153–178 (cited on pp. 98, 124).

[ABDK+21] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation*. https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf. Accessed: 12-05-2025. 2021 (cited on pp. 91, 133, 134).

[ABNA+23] ANSSI, BSI, NLNCSA, Swedish National Communications Security Authority, and Swedish Armed Forces. *Position Paper on Quantum Key Distribution*. Technical Report. Accessed: 2026-01-30. Dec. 2023. URL: https://open.overheid.nl/documenten/797c7e8e-9c70-4a98-bfb4-11cb5f19515f/file (cited on p. 8).

[ACKS21] Divesh Aggarwal, Yanlin Chen, Rajendra Kumar, and Yixin Shen. "Improved (Provable) Algorithms for the Shortest Vector Problem via Bounded Distance Decoding". In: *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*. Vol. 187. LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 4:1–4:20 (cited on pp. 49, 55).

[ACKS25] Divesh Aggarwal, Yanlin Chen, Rajendra Kumar, and Yixin Shen. "Improved Classical and Quantum Algorithms for the Shortest Vector Problem via Bounded Distance Decoding". In: *SIAM Journal on Computing* 54.2 (2025), pp. 233–278 (cited on p. 155).

[AD21] Martin Albrecht and Léo Ducas. "Lattice Attacks on NTRU and LWE: A History of Refinements". In: *Computational Cryptography: Algorithmic Aspects of Cryptology*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021, pp. 15–40 (cited on pp. 106, 115, 134).

[ADHK+19] Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. "The general sieve kernel and new records in lattice reduction". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 717–746 (cited on pp. 94, 109, 134, 156, 202).

[ADRS15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. "Solving the Shortest Vector Problem in 2^n Time Using Discrete Gaussian Sampling: Extended Abstract". In: *Proceedings of the ACM Symposium on Theory of Computing (STOC 2015)*. ACM, 2015, pp. 733–742 (cited on pp. 49, 55, 155).

[ADS15] Divesh Aggarwal, Daniel Dadush, and Noah Stephens-Davidowitz. "Solving the Closest Vector Problem in 2^n Time – The Discrete Gaussian Strikes Again!" In: *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS 2015)*. IEEE Computer Society, 2015, pp. 563–582 (cited on pp. 49, 55).

[AFFP14] Martin R Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. "Lazy Modulus Switching for the BKW Algorithm on LWE". In: *Public-Key Cryptography – PKC 2014*. Springer. 2014, pp. 429–445 (cited on p. 51).

[Ajt96] Miklós Ajtai. "Generating Hard Instances of Lattice Problems (Extended Abstract)". In: *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*. 1996, pp. 99–108 (cited on pp. 39–41, 47).

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. "A sieve algorithm for the shortest lattice vector problem". In: *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*. ACM, 2001, pp. 601–610. URL: https://doi.org/10.1145/380752.380857 (cited on p. 155).

[ALS21] Divesh Aggarwal, Zeyong Li, and Noah Stephens-Davidowitz. "A $2^{n/2}$-Time Algorithm for $\sqrt{n}$-SVP and $\sqrt{n}$-Hermite SVP, and an Improved Time-Approximation Tradeoff for (H)SVP". In: *Advances in Cryptology – EUROCRYPT 2021*. Vol. 12696. Lecture Notes in Computer Science. Springer, 2021, pp. 467–497 (cited on pp. 49, 55–57, 61, 69, 70).

[Amb04] Andris Ambainis. "Quantum Walk Algorithm for Element Distinctness". In: *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS 2004)*. 2004, pp. 22–31 (cited on pp. 146, 205).

[Amb07] Andris Ambainis. “Quantum walk algorithm for element distinctness”. In: *SIAM Journal on Computing* 37.1 (2007), pp. 210–239. ISSN: 0097-5397 (cited on p. 245).

[Amb10] Andris Ambainis. “Quantum Search with Variable Times”. In: *Theory of Computing Systems* 47.3 (2010), pp. 786–807 (cited on pp. 151, 173).

[AR05] Dorit Aharonov and Oded Regev. “Lattice Problems in NP ∩ coNP”. In: *Journal of the ACM (JACM)* 52.5 (2005), pp. 749–765 (cited on pp. 39, 47, 87).

[AS18] Divesh Aggarwal and Noah Stephens-Davidowitz. “Just Take the Average! An Embarrassingly Simple 2^n-Time Algorithm for SVP (and CVP)”. In: *1st Symposium on Simplicity in Algorithms (SOSA 2018)*. Vol. 61. Open Access Series in Informatics. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, 12:1–12:19 (cited on pp. 49, 55).

[AWHT16] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. “Improved Progressive BKZ Algorithms and Their Precise Cost Estimation by Sharp Simulator”. In: *Advances in Cryptology – EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. Lecture Notes in Computer Science. Springer, 2016, pp. 789–819 (cited on pp. 91, 134).

[Bab86] László Babai. “On Lovász’ Lattice Reduction and the Nearest Lattice Point Problem”. In: *Combinatorica* 6.1 (1986), pp. 1–13 (cited on p. 108).

[Ban93] Wojciech Banaszczyk. “New Bounds in Some Transference Theorems in the Geometry of Numbers”. In: *Mathematische Annalen* 296 (1993), pp. 625–635 (cited on p. 59).

[Ban95] Wojciech Banaszczyk. “Inequalites for Convex Bodies and Polar Reciprocal Lattices in $\mathbb{R}^n$”. In: *Discrete & Computational Geometry* 13 (1995), pp. 217–231 (cited on p. 60).

[Bay00] Eva Bayer-Fluckiger. “Cyclotomic Modular Lattices”. In: *Journal de théorie des nombres de Bordeaux* 12.2 (2000), pp. 273–280 (cited on p. 133).

[BB14] Charles H. Bennett and Gilles Brassard. “Quantum Cryptography: Public Key Distribution and Coin Tossing”. In: *Theoretical Computer Science* 560 (2014). Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84, pp. 7–11. ISSN: 0304-3975 (cited on p. 8).

[BBHT98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. “Tight Bounds on Quantum Searching”. In: *Fortschritte der Physik* 46.4-5 (1998), pp. 493–505 (cited on pp. 141, 142).

[BCSS23] Xavier Bonnetain, André Chailloux, André Schrottenloher, and Yixin Shen. “Finding many Collisions via Reusable Quantum Walks”. In: *Proceedings of the Advances in Cryptology – EUROCRYPT 2023*. Vol. 14008. 2023, pp. 221–251 (cited on pp. 16, 150, 156–158, 203, 205, 214, 238, 251).

[BCWZ99] Harry Buhrman, Richard Cleve, Ronald de Wolf, and Christof Zalka. “Bounds for Small-Error and Zero-Error Quantum Algorithms”. In: *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS 1999)*. IEEE Computer Society, 1999, pp. 358–368 (cited on p. 142).

[BDEF+17] {Daniel J.} Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, {Martin M.} Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, and Peter Schwabe. *SPHINCS+ – Submission to the NIST post-quantum cryptography project*. 2017. URL: https://sphincs.org (cited on p. 10).

[BDF18] Guillaume Bonnoron, Léo Ducas, and Max Fillinger. “Large FHE Gates from Tensored Homomorphic Accumulator”. In: *Progress in Cryptology – AFRICACRYPT 2018*. Ed. by Antoine Joux, Abderrahmane Nitaj, and Tajjeeddine Rachidi. Vol. 10831. Lecture Notes in Computer Science. Springer, 2018, pp. 217–251 (cited on pp. 110, 133).

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. “New Directions In Nearest Neighbor Searching with Applications to Lattice Sieving”. In: *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*. 2016, pp. 10–24 (cited on

pp. 16, 134, 135, 157, 158, 161, 166–169, 171, 214, 226, 227, 250).

[BDKL+18] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. "CRYSTALS – Kyber: A CCA-secure module-lattice-based KEM". In: *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P 2018)*. 2018, pp. 353–367 (cited on p. 10).

[Ben82] Paul A. Benioff. "Quantum Mechanical Hamiltonian Models of Turing Machines". In: *Journal of Statistical Physics* 29.3 (1982), pp. 515–546 (cited on p. 6).

[Ber10] Daniel J. Bernstein. "Grover vs. McEliece". In: *Post-Quantum Cryptography (PQCrypto 2010)*. Ed. by Nicolas Sendrier. Vol. 6061. Lecture Notes in Computer Science. Springer, 2010, pp. 73–80 (cited on pp. 213, 215, 254–256, 258).

[BGJM+20] Alessandro Budroni, Qian Guo, Thomas Johansson, Erik Mårtensson, and Paul Stankovski Wagner. "Making the BKW Algorithm Practical for LWE". In: *Progress in Cryptology – INDOCRYPT 2020*. Springer. 2020, pp. 417–439 (cited on p. 83).

[BH12] Andries E. Brouwer and Willem H. Haemers. *Spectra of Graphs*. Universitext. Springer New York, 2012 (cited on p. 150).

[BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. "Quantum amplitude amplification and estimation". In: *Quantum Computation and Information*. Vol. 305. Contemporary Mathematics. American Mathematical Society, 2002, pp. 53–74. URL: http://dx.doi.org/10.1090/conm/305/05215 (cited on pp. 15, 141–143).

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. "Quantum Algorithm for the Collision Problem". In: *Latin American Symposium on Theoretical Informatics (LATIN 1998)*. Lecture Notes in Computer Science. arXiv:quant-ph/9705002. 1998, pp. 163–169 (cited on p. 205).

[BJLM13] Daniel J. Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. "Quantum Algorithms for the Subset-Sum Problem". In: *Post-Quantum Cryptography (PQCrypto 2013)*. Ed. by Philippe

Gaborit. Vol. 7932. Lecture Notes in Computer Science. Springer, 2013, pp. 16–33 (cited on p. 245).

[BJMM12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. "Decoding Random Binary Linear Codes in $2^{n/20}$: How 1+ 1 = 0 Improves Information Set Decoding". In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. 2012, pp. 520–536. URL: https://doi.org/10.1007/978-3-642-29011-4%5C_31 (cited on pp. 213, 214).

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. "Noise-Tolerant Learning, the Parity Problem, and the Statistical Query Model". In: *Journal of the ACM (JACM)* 50.4 (2003), pp. 506–519 (cited on pp. 47, 50, 54).

[BLPR+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. "Classical Hardness of Learning with Errors". In: *Proceedings of the ACM Symposium on Theory of Computing (STOC 2013)*. ACM, 2013, pp. 575–584 (cited on p. 61).

[BLPS22] Harry Buhrman, Bruno Loff, Subhasree Patro, and Florian Speelman. "Memory Compression with Quantum Random-Access Gates". In: vol. 232. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 10:1–10:19. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TQC.2022.10 (cited on p. 245).

[BLS16] Shi Bai, Thijs Laarhoven, and Damien Stehlé. "Tuple lattice sieving". In: *LMS Journal of Computational Mathematics* 19.A (2016), pp. 146–162 (cited on pp. 156, 157, 201, 202).

[BM18] Leif Both and Alexander May. "Decoding Linear Codes with High Error Rate and Its Impact for LPN Security". In: *Post-Quantum Cryptography (PQCrypto 2018)*. Ed. by Tanja Lange and Rainer Steinwandt. Vol. 10786. Lecture Notes in Computer Science. Springer, 2018, pp. 25–46 (cited on p. 214).

[BMNG+21] Ryan Babbush, Jarrod R. McClean, Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven. "Focus beyond Quadratic Speedups for Error-Corrected Quantum Advantage". In: *PRX*

Quantum 2.1 (Mar. 2021). ISSN: 2691-3399. URL: http://dx.doi.org/10.1103/PRXQuantum.2.010103 (cited on p. 205).

[BMT78] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. "On the inherent intractability of certain coding problems". In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386 (cited on p. 213).

[BNP17] Joppe W Bos, Michael Naehrig, and Joop Van De Pol. "Sieving for Shortest Vectors in Ideal Lattices: A Practical Perspective". In: *International Journal of Applied Cryptography* 3.4 (2017), pp. 313–329 (cited on pp. 91, 134).

[BSS22] Xavier Bonnetain, André Schrottenloher, and Ferdinand Sibleyras. *Beyond Quadratic Speedups in Quantum Attacks on Symmetric Schemes*. 2022 (cited on p. 7).

[BSW18] Shi Bai, Damien Stehlé, and Weiqiang Wen. "Measuring, Simulating and Exploiting the Head Concavity Phenomenon in BKZ". In: *Advances in Cryptology – ASIACRYPT 2018*. Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11272. Lecture Notes in Computer Science. Springer, 2018, pp. 369–404 (cited on pp. 93, 106, 119, 126, 134).

[BW25] Koen de Boer and Wessel van Woerden. *Lattice-Based Cryptography: A Survey on the Security of the Lattice-Based NIST Finalists*. Cryptology ePrint Archive, Paper 2025/304. 2025. URL: https://eprint.iacr.org/2025/304 (cited on pp. 106, 108).

[Car20] Kévin Carrier. "Recherche de presque-collisions pour le décodage et la reconnaissance de codes correcteurs (Near-collisions finding problem for decoding and recognition of error correcting codes)". PhD thesis. Sorbonne University, France, 2020. URL: https://tel.archives-ouvertes.fr/tel-03370678 (cited on pp. 213, 225, 227–230).

[CDMT22] 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*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13794. Lecture Notes in Computer Science. 2022, pp. 477–507. URL: https://doi.org/10.1007/978-3-031-22972-5%5C_17 (cited on p. 252).

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. "Recovering Short Generators of Principal Ideals in Cyclotomic Rings". In: *Advances in Cryptology – EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Springer, 2016, pp. 559–585 (cited on pp. 122, 124).

[Che13] Yuanmi Chen. "Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe". 2013PA077242. PhD thesis. 2013, 1 vol. (133 p.) URL: http://www.theses.fr/2013PA077242 (cited on pp. 97, 106, 119).

[Che52] Herman Chernoff. "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations". In: *The Annals of Mathematical Statistics* 23.4 (1952), pp. 493–507 (cited on p. 24).

[CL21] André Chailloux and Johanna Loyer. "Lattice Sieving via Quantum Random Walks". In: *Advances in Cryptology – ASIACRYPT 2021*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13093. Lecture Notes in Computer Science. 2021, pp. 63–91. URL: https://doi.org/10.1007/978-3-030-92068-5%5C_3 (cited on pp. 17, 150, 156, 158, 205, 214–216, 236, 238, 240, 241, 250, 251).

[CL23] André Chailloux and Johanna Loyer. "Classical and Quantum 3 and 4-Sieves to Solve SVP with Low Memory". In: *International Conference on Post-Quantum Cryptography*. Springer. 2023, pp. 225–255. URL: https://eprint.iacr.org/2023/200.pdf (cited on pp. 16, 157, 158, 160, 161, 203).

[CN11] Yuanmi Chen and Phong Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 1–20 (cited on p. 134).

[Coo00] Colin Cooper. "On the distribution of rank of a random matrix over a finite field". In: *Random Structures & Algorithms* 17.3-4 (2000), pp. 197–212 (cited on p. 253).

[DD12] Léo Ducas and Alain Durmus. "Ring-LWE in Polynomial Rings". In: *Public Key Cryptography – PKC 2012*. Ed. by Marc Fischlin, Johannes Buchmann, and Mark Manulis. Vol. 7293. Lecture Notes in Computer Science. Springer, 2012, pp. 34–51 (cited on p. 110).

[DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. "LWE with Side Information: Attacks and Concrete Security Estimation". In: *Advances in Cryptology – CRYPTO 2020*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. Lecture Notes in Computer Science. Springer, 2020, pp. 329–358 (cited on p. 135).

[Deb23] Thomas Debris-Alazard. *Code-Based Cryptography: Lecture Notes*. 2023. arXiv: 2304.03541 [cs.CR]. URL: https://arxiv.org/abs/2304.03541 (cited on pp. 220, 221).

[DEEK24] Léo Ducas, Andre Esser, Simona Etinski, and Elena Kirshanova. "Asymptotics and Improvements of Sieving for Codes". In: *Advances in Cryptology – EUROCRYPT 2024*. Ed. by Marc Joye and Gregor Leander. Vol. 14656. Lecture Notes in Computer Science. Springer, 2024, pp. 151–180 (cited on pp. 17, 213–219, 221, 222, 224–230, 242, 249–255, 257, 262).

[DEL25] Léo Ducas, Lynn Engelberts, and Johanna Loyer. "Wagner's Algorithm Provably Runs in Subexponential Time for SIS^{∞}". In: *Advances in Cryptology – CRYPTO 2025*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16000. Lecture Notes in Computer Science. Implementation available at https://github.com/lducas/Provable-Wagner-SIS. Springer, 2025, pp. 353–384 (cited on pp. ix, 13, 45).

[DEP23] Léo Ducas, Thomas Espitau, and Eamonn W Postlethwaite. "Finding Short Integer Solutions When the Modulus is Small". In: *Annual International Cryptology Conference*. Springer. 2023, pp. 150–176 (cited on pp. 82, 84).

[DEP25] Léo Ducas, Lynn Engelberts, and Paola de Perthuis. "Predicting Module-Lattice Reduction". In: *Advances in Cryptology – ASIACRYPT 2025*. Lecture Notes in Computer Science. Implementation available at https://github.com/lducas/mBKZ/. Springer, 2025 (cited on pp. ix, 13, 89, 135).

[Deu85] David Deutsch. “Quantum Theory, the Church–Turing Principle, and the Universal Quantum Turing Machine”. In: *Proceedings of the Royal Society of London. Series A* 400 (1985), pp. 97–117 (cited on p. 6).

[DH76] Whitfield Diffie and Martin E. Hellman. “New Directions in Cryptography”. In: *IEEE Trans. Inf. Theory* 22.6 (1976), pp. 644–654 (cited on p. 4).

[DKLL+18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. “CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 238–268 (cited on pp. 10, 41, 47, 84).

[DL25] Léo Ducas and Johanna Loyer. “Lattice Reduction via Dense Sublattices: A Cryptanalytic No-Go”. In: *IACR Communications in Cryptology* 2.3 (Oct. 6, 2025). ISSN: 3006-5496 (cited on p. 98).

[DP16] Léo Ducas and Thomas Prest. “Fast Fourier Orthogonalization”. In: *Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation*. 2016, pp. 191–198 (cited on p. 110).

[DPPW22] Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel P. J. van Woerden. “Hawk: Module LIP Makes Lattice Signatures Fast, Compact and Simple”. In: *Advances in Cryptology – ASIACRYPT 2022*. Ed. by Shweta Agrawal and Dongdai Lin. Vol. 13794. Lecture Notes in Computer Science. Springer, 2022, pp. 65–94 (cited on pp. 98, 124, 133).

[DSW21] Léo Ducas, Marc Stevens, and Wessel P. J. van Woerden. “Advanced Lattice Sieving on GPUs, with Tensor Cores”. In: *Advances in Cryptology – EUROCRYPT 2021*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. Lecture Notes in Computer Science. Springer, 2021, pp. 249–279 (cited on p. 134).

[Duc18] Léo Ducas. “Shortest Vector from Lattice Sieving: A Few Dimensions for Free”. In: *Advances in Cryptology – EUROCRYPT 2018*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. Lecture Notes in Computer Science. Springer, 2018, pp. 125–145 (cited on pp. 91, 134, 135).

[Dum91] Ilya Dumer. “On minimum distance decoding of linear codes”. In: *Proceedings of the Fifth Joint Soviet–Swedish International Workshop on Information Theory*. 1991, pp. 50–52 (cited on p. 213).

[DW21] Léo Ducas and Wessel P. J. van Woerden. “NTRU Fatigue: How Stretched is Overstretched?” In: *Advances in Cryptology – ASIACRYPT 2021*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13093. Lecture Notes in Computer Science. Springer, 2021, pp. 3–32 (cited on pp. 97, 98, 122, 124).

[DW22] Léo Ducas and Wessel P. J. van Woerden. “On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography”. In: *Advances in Cryptology – EUROCRYPT 2022*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. Lecture Notes in Computer Science. Springer, 2022, pp. 643–673 (cited on p. 133).

[ECGH+25] Lynn Engelberts, Yanlin Chen, Amin Shiraz Gilani, Maya-Iggy van Hoof, Stacey Jeffery, and Ronald de Wolf. “An Improved Quantum Algorithm for 3-Tuple Lattice Sieving”. In: *CoRR* abs/2510.08473 (2025). arXiv: 2510.08473 (cited on pp. ix, 15, 153, 158).

[EEL25] Lynn Engelberts, Simona Etinski, and Johanna Loyer. “Quantum Sieving for Code-Based Cryptanalysis and its Limitations for ISD”. In: *Designs, Codes and Cryptography* 93.6 (2025). Implementation available at https://github.com/lynnengelberts/quantum-sieving-for-codes-public, pp. 1611–1644 (cited on pp. ix, 15, 211).

[EFGR+22] Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. “Mitaka: A Simpler, Parallelizable, Maskable Variant of Falcon”. In: *Advances in Cryptology – EUROCRYPT 2022*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13277. Lecture Notes in Computer Science. Springer, 2022, pp. 222–253 (cited on p. 133).

[Eke91] Artur K. Ekert. “Quantum Cryptography Based on Bell’s Theorem”. In: *Phys. Rev. Lett.* 67 (6 Aug. 1991), pp. 661–663. URL: https://link.aps.org/doi/10.1103/PhysRevLett.67.661 (cited on p. 8).

[Emd81] Peter van Emde Boas. *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*. Report. Department of Mathematics. University of Amsterdam. 1981 (cited on pp. 39, 40).

[ETWY22] Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. "Shorter Hash-and-Sign Lattice-Based Signatures". In: *Annual International Cryptology Conference*. Springer. 2022, pp. 245–275 (cited on p. 82).

[Eve] Jan-Hendrik Evertse. *Chapter 3: Algebraic numbers and algebraic number fields*. Chapter 3 of lecture notes for the course *Diophantine Approximation*. https://pub.math.leidenuniv.nl/~evertsejh/dio19-3.pdf (cited on p. 99).

[EWY23a] Thomas Espitau, Alexandre Wallet, and Yang Yu. "On Gaussian Sampling, Smoothing Parameter and Application to Signatures". In: *Advances in Cryptology – ASIACRYPT 2023*. Vol. 14444. Lecture Notes in Computer Science. Springer, 2023, pp. 65–97 (cited on pp. 55, 65, 78).

[EWY23b] Thomas Espitau, Alexandre Wallet, and Yang Yu. "On Gaussian Sampling, Smoothing Parameter and Application to Signatures". In: *Advances in Cryptology – ASIACRYPT 2023*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14444. Lecture Notes in Computer Science. Springer, 2023, pp. 65–97 (cited on p. 133).

[Fey82] Richard P. Feynman. "Simulating Physics with Computers". In: *International Journal of Theoretical Physics* 21.6/7 (1982), pp. 467–488 (cited on p. 6).

[Fey85] Richard P. Feynman. "Quantum Mechanical Computers". In: *Optics News* 11 (1985), pp. 11–20 (cited on p. 6).

[FS09] Matthieu Finiasz and Nicolas Sendrier. "Security Bounds for the Design of Code-Based Cryptosystems". In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. 2009, pp. 88–105. URL: https://doi.org/10.1007/978-3-642-10366-7%5C_6 (cited on p. 252).

[FS10] Claus Fieker and Damien Stehlé. "Short Bases of Lattices over Number Fields". In: *Algorithmic Number Theory (ANTS-IX)*. Ed. by Guillaume Hanrot, François Morain, and Emmanuel Thomé. Springer Berlin Heidelberg, 2010, pp. 157–173 (cited on pp. 104, 108).

[GJMS17] Qian Guo, Thomas Johansson, Erik Mårtensson, and Paul Stankovski. "Coded-BKW with Sieving". In: *Advances in Cryptology – ASIACRYPT 2017*. Springer. 2017, pp. 323–346 (cited on pp. 51, 83).

[GJN23] Qian Guo, Thomas Johansson, and Vu Nguyen. "A New Sieving-Style Information-Set Decoding Algorithm". In: *IACR Cryptology ePrint Archive* (2023). URL: https://eprint.iacr.org/2023/247 (cited on pp. 17, 213, 214, 216, 221, 222, 225, 252).

[GMSS99] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert. "Approximating Shortest Lattice Vectors is Not Harder Than Approximating Closest Lattice Vectors". In: *Information Processing Letters* 71.2 (1999), pp. 55–61. ISSN: 0020-0190 (cited on p. 40).

[GN08a] Nicolas Gama and Phong Q. Nguyen. "Finding Short Lattice Vectors within Mordell's Inequality". In: *Proceedings of the ACM Symposium on Theory of Computing (STOC 2008)*. Ed. by Cynthia Dwork. ACM, 2008, pp. 207–216 (cited on pp. 39, 91, 92).

[GN08b] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction". In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Springer, 2008, pp. 31–51 (cited on p. 91).

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for Hard Lattices and New Cryptographic Constructions". In: *Proceedings of the ACM Symposium on Theory of Computing (STOC 2008)*. ACM, 2008, pp. 197–206 (cited on pp. 55, 63, 65, 201).

[Gro96] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by Gary L. Miller. ACM, 1996, pp. 212–219 (cited on pp. 7, 15, 141, 142, 150, 235).

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. "Quantum Singular Value Transformation and Beyond: Exponential Improvements for Quantum Matrix Arithmetics". In: *Proceedings of the 51st ACM Symposium on the Theory of Computing*. arXiv:1806.01838. 2019, pp. 193–204 (cited on pp. 144, 145).

[GSVV24] Nihar Gargava, Vlad Serban, Maryna Viazovska, and Ilaria Viglino. "Effective Module Lattices and their Shortest Vectors". 2024. arXiv: 2402.10305 [math.NT]. URL: https://arxiv.org/abs/2402.10305 (cited on pp. 97, 118, 119).

[Hei21] Max Heiser. *Improved Quantum Hypercone Locality Sensitive Filtering in Lattice Sieving*. Cryptology ePrint Archive, Paper 2021/1295. 2021. URL: https://eprint.iacr.org/2021/1295 (cited on p. 203).

[HK17] Gottfried Herold and Elena Kirshanova. "Improved Algorithms for the Approximate k-List Problem in Euclidean Norm". In: *Public-Key Cryptography – PKC 2017*. Vol. 10174. Lecture Notes in Computer Science. Springer, 2017, pp. 16–40 (cited on pp. 157, 159, 170, 201, 202).

[HKL18] Gottfried Herold, Elena Kirshanova, and Thijs Laarhoven. "Speed-ups and time-memory trade-offs for tuple lattice sieving". In: *Public-Key Cryptography – PKC 2018*. Vol. 10769. Lecture Notes in Computer Science. Springer, 2018, pp. 407–436 (cited on p. 158).

[HKM18] Gottfried Herold, Elena Kirshanova, and Alexander May. "On the Asymptotic Complexity of Solving LWE". In: *Designs, Codes and Cryptography* 86.1 (2018), pp. 55–83 (cited on pp. 14, 47, 54, 87).

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. "Analyzing Blockwise Lattice Algorithms Using Dynamical Systems". In: *Advances in Cryptology – CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 447–464 (cited on p. 108).

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. "NTRU: A Ring-Based Public Key Cryptosystem". In: *Algorithmic Number Theory (ANTS-III)*. Ed. by Joe Buhler. Vol. 1423. Lecture Notes in Computer Science. Springer, 1998, pp. 267–288 (cited on p. 91).

[HS14] Shai Halevi and Victor Shoup. "Algorithms in HElib". In: *Advances in Cryptology – CRYPTO 2014*. Ed. by Juan A. Garay and Rosario Gennaro. Springer Berlin Heidelberg, 2014, pp. 554–571 (cited on pp. 133, 135).

[HWKK+25] Ga Hee Hong, Joo Woo, Jonghyun Kim, Minkyu Kim, Hochang Lee, and Jong Hwan Park. *More NTRU+Sign Signatures from Cyclotomic Trinomials*. Cryptology ePrint Archive, Paper 2025/612. 2025. URL: https://eprint.iacr.org/2025/612 (cited on pp. 133, 135).

[Iir20] Iiridayn. *Answer to "When to stop enumerating a fixed set of unknown cardinality via random sampling?"* Cross Validated (Stack Exchange). https://stats.stackexchange.com/a/486813, accessed October 2025. 2020 (cited on p. 163).

[Kah96] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, 1996 (cited on p. 3).

[KF15] Paul Kirchner and Pierre-Alain Fouque. "An Improved BKW Algorithm for LWE with Applications to Cryptography and Lattices". In: *Advances in Cryptology – CRYPTO 2015*. Vol. 9215. Lecture Notes in Computer Science. Springer. 2015, pp. 43–62 (cited on pp. 13, 14, 47, 51, 54, 83, 86, 87).

[Kir18] Elena Kirshanova. "Improved Quantum Information Set Decoding". In: *Post-Quantum Cryptography (PQCrypto 2018)*. Ed. by Tanja Lange and Rainer Steinwandt. Vol. 10786. Lecture Notes in Computer Science. 2018, pp. 507–527. URL: https://doi.org/10.1007/978-3-319-79063-3%5C_24 (cited on pp. 213, 254, 255).

[KK24] Alexander Karenin and Elena Kirshanova. "Finding Dense Submodules with Algebraic Lattice Reduction". In: *Progress in Cryptology – AFRICACRYPT 2024*. Ed. by Serge Vaudenay and Christophe Petit. Vol. 14861. Lecture Notes in Computer Science. Springer, 2024, pp. 403–427 (cited on pp. 92, 115, 133, 135).

[Kle00] Philip N. Klein. "Finding the Closest Lattice Vector when it's Unusually Close". In: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San*

Francisco, CA, USA. Ed. by David B. Shmoys. ACM/SIAM, 2000, pp. 937–941 (cited on p. 201).

[KLL15] Shelby Kimmel, Cedric Yen-Yu Lin, and Han-Hsuan Lin. "Oracles with Costs". In: *Proceedings of the 10th Conference on the Theory of Quantum Computation, Communication and Cryptography*. Vol. 44. LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015, pp. 1–26 (cited on pp. 151, 160, 173, 204).

[KMPM19] Elena Kirshanova, Erik Mårtensson, Eamonn W. Postlethwaite, and Subhayan Roy Moulik. "Quantum Algorithms for the Approximate k-List Problem and their Application to Lattice Sieving". In: *Advances in Cryptology – ASIACRYPT 2019*. Vol. 11921. Lecture Notes in Computer Science. Springer, 2019, pp. 521–551 (cited on pp. 157, 158).

[Kob87] Neal Koblitz. "Elliptic Curve Cryptosystems". In: *Mathematics of Computation* 48 (1987), pp. 203–209 (cited on p. 6).

[Kpq25] KpqC. *Selected Algorithms from the KpqC Competition Round 2*. https://kpqc.or.kr/competition_02.html. Accessed: 14-05-2025. 2025 (cited on p. 133).

[KT17] Ghazal Kachigar and Jean-Pierre Tillich. "Quantum Information Set Decoding Algorithms". In: *Post-Quantum Cryptography (PQCrypto 2017)*. Ed. by Tanja Lange and Tsuyoshi Takagi. Vol. 10346. Lecture Notes in Computer Science. 2017, pp. 69–89. URL: https://doi.org/10.1007/978-3-319-59879-6%5C_5 (cited on pp. 213, 254, 255).

[Laa15] Thijs Laarhoven. "Search Problems in Cryptography: From Fingerprinting to Lattice Sieving". PhD thesis. Eindhoven University of Technology, The Netherlands, 2015. URL: https://thijs.com/docs/phd-final.pdf (cited on pp. 17, 134, 156, 214, 215, 240, 250).

[Lam79] Leslie Lamport. "Constructing Digital Signatures from a One Way Function". In: (1979) (cited on p. 8).

[LF06] Éric Levieil and Pierre-Alain Fouque. "An Improved LPN Algorithm". In: *Security and Cryptography for Networks*. Springer. 2006, pp. 348–359 (cited on p. 83).

[LK14] Yongjae Lee and Woo Chang Kim. *Concise Formulas for the Surface Area of the Intersection of Two Hyperspherical Caps*. Technical Report. KAIST, Feb. 2014 (cited on p. 167).

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Miklós Lovász. "Factoring Polynomials with Rational Coefficients". In: *Mathematische Annalen* 261 (1982), pp. 515–534 (cited on pp. 39, 91, 92, 201).

[LM18] Thijs Laarhoven and Artur Mariano. "Progressive lattice sieving". In: *International Conference on Post-Quantum Cryptography*. Springer. 2018, pp. 292–311 (cited on pp. 91, 134).

[LMP15] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. "Finding Shortest Lattice Vectors Faster Using Quantum Search". In: *Designs, Codes and Cryptography* 77 (Dec. 2015) (cited on p. 156).

[LN24] Jianwei Li and Phong Q. Nguyen. "A Complete Analysis of the BKZ Lattice Reduction Algorithm". In: *Journal of Cryptology* 38.1 (2024), p. 12 (cited on p. 97).

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: vol. 60. 6. Earlier version in Advances in Cryptology – EUROCRYPT 2010. Association for Computing Machinery, Nov. 2013 (cited on p. 91).

[LPSW19] Changmin Lee, Alice Pellet-Mary, Damien Stehlé, and Alexandre Wallet. "An LLL Algorithm for Module Lattices". In: *Advances in Cryptology – ASIACRYPT 2019*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11922. Lecture Notes in Computer Science. Springer, 2019, pp. 59–90 (cited on pp. 15, 92, 102, 104, 107).

[LS15] Adeline Langlois and Damien Stehlé. "Worst-Case to Average-Case Reductions for Module Lattices". In: *Designs, Codes and Cryptography* 75.3 (2015), pp. 565–599 (cited on p. 91).

[LS19] Vadim Lyubashevsky and Gregor Seiler. "NTTRU: Truly Fast NTRU Using NTT". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019.3 (2019), pp. 180–201 (cited on pp. 133, 135).

[Lyu05] Vadim Lyubashevsky. "The Parity Problem in the Presence of Noise, Decoding Random Linear Codes, and the Subset Sum Problem". In: *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer. 2005, pp. 378–389 (cited on pp. 49, 54).

[MABB+22] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Charles Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Joppe Bos, Aurore Dion, Jérôme Lacan, Jean-Michel Robert, and Pascal Veron. *HQC Algorithm Specifications and Supporting Documentation*. NIST PQC Round 4 submission, specification document. 2022. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions (cited on p. 10).

[Man80] Yuri I. Manin. *Vychislimoe i nevychislimoe (Computable and Noncomputable)*. Russian. Soviet Radio, 1980, pp. 13–15 (cited on p. 6).

[Man99] Yuri I. Manin. "Classical Computing, Quantum Computing, and Shor's Factoring Algorithm". In: *arXiv preprint* (Mar. 1999). eprint: quant-ph/9903008 (cited on p. 6).

[Mar03] Jacques Martinet. *Perfect Lattices in Euclidean Spaces*. Vol. 327. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2003. URL: http://link.springer.com/10.1007/978-3-662-05167-2 (cited on p. 36).

[Mar78] Jacques Martinet. "Tours de corps de classes et estimations de discriminants". In: *Inventiones mathematicae* 44 (1978), pp. 65–73 (cited on p. 133).

[McE78] Robert J. McEliece. "A Public Key Cryptosystem Based on Algebraic Coding Theory". In: 1978. URL: https://api.semanticscholar.org/CorpusID:56502909 (cited on p. 8).

[Mil86] Victor S. Miller. "Use of Elliptic Curves in Cryptography". In: *Advances in Cryptology – CRYPTO 1985*. Ed. by Hugh C. Williams. Springer Berlin Heidelberg, 1986, pp. 417–426 (cited on p. 6).

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. "Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. 2011, pp. 107–124 (cited on p. 213).

[MNRS11] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. "Search via Quantum Walk". In: *SIAM Journal on Computing* 40.1 (2011), pp. 142–164 (cited on pp. 15, 146–149, 237).

[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 Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. 2015, pp. 203–228. URL: https://doi.org/10.1007/978-3-662-46800-5%5C_9 (cited on p. 213).

[MP13] Daniele Micciancio and Chris Peikert. "Hardness of SIS and LWE with Small Parameters". In: *Advances in Cryptology – CRYPTO 2013*. Vol. 8042. Lecture Notes in Computer Science. Springer, 2013, pp. 21–39 (cited on pp. 55, 61, 69).

[MP24] Michele Mosca and Marco Piani. *Quantum Threat Timeline Report 2024*. Technical Report. Accessed: 2026-01-30. Global Risk Institute in Financial Services and evolutionQ Inc., Dec. 2024. URL: https://globalriskinstitute.org/publication/2024-quantum-threat-timeline-report/ (cited on p. 7).

[MR07] Daniele Micciancio and Oded Regev. "Worst-Case to Average-Case Reductions Based on Gaussian Measures". In: *SIAM Journal on Computing* 37.1 (2007), pp. 267–302. Earlier version in FOCS 2004 (cited on pp. 47, 60, 62).

[MS20] Tamalika Mukherjee and Noah Stephens-Davidowitz. "Lattice Reduction for Modules, or How to Reduce ModuleSVP to ModuleSVP". In: *Advances in Cryptology – CRYPTO 2020*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. Lecture Notes in Computer Science. Springer, 2020, pp. 213–242 (cited on pp. 15, 92, 102, 135).

[MV10] Daniele Micciancio and Panagiotis Voulgaris. “Faster Exponential Time Algorithms for the Shortest Vector Problem”. In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010*. Ed. by Moses Charikar. SIAM, 2010, pp. 1468–1480 (cited on pp. 134, 202).

[MW16] Daniele Micciancio and Michael Walter. “Practical, Predictable Lattice Basis Reduction”. In: *Advances in Cryptology – EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. Lecture Notes in Computer Science. Springer, 2016, pp. 820–849 (cited on pp. 39, 91).

[Nat20] National Cyber Security Centre. *Quantum Security Technologies*. White Paper. Accessed: 2026-01-30. National Cyber Security Centre, 2020. URL: https://www.ncsc.gov.uk/whitepaper/quantum-security-technologies (cited on p. 8).

[Nat21] National Security Agency. *Quantum Key Distribution (QKD) and Quantum Cryptography (QC)*. Web Guidance. Accessed: 2026-01-30. National Security Agency, 2021. URL: https://www.nsa.gov/Cybersecurity/Quantum-Key-Distribution-QKD-and-Quantum-Cryptography-QC/ (cited on p. 8).

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000 (cited on pp. 27–29).

[Neu92] Jürgen Neukirch. *Algebraische Zahlentheorie*. German. Springer Berlin Heidelberg, 1992 (cited on pp. 128, 129).

[NIS16] National Institute of Standards and Technology (NIST). *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf (cited on pp. 9, 85).

[NIS22a] National Institute of Standards and Technology (NIST). *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. 2022. URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf (cited on p. 9).

[NIS22b] National Institute of Standards and Technology (NIST). *Post-Quantum Cryptography: Selected Algorithms*. https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms. Accessed: 29-04-2025. 2022 (cited on pp. 9, 91, 132).

[NIS24a] National Institute of Standards and Technology (NIST). *FIPS 203. Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Aug. 2024 (cited on p. 10).

[NIS24b] National Institute of Standards and Technology (NIST). *FIPS 204. Module-Lattice-Based Digital Signature Standard*. Aug. 2024 (cited on p. 10).

[NIS24c] National Institute of Standards and Technology (NIST). *FIPS 205. Stateless Hash-Based Digital Signature Standard*. Aug. 2024 (cited on p. 10).

[NV08] Phong Q. Nguyen and Thomas Vidick. "Sieve Algorithms for the Shortest Vector Problem Are Practical". In: *Journal of Mathematical Cryptology* 2.2 (2008), pp. 181–207 (cited on pp. 155, 156, 201, 202).

[Pat23] Subhasree Patro. "Quantum Fine-Grained Complexity". PhD thesis. University of Amsterdam, 2023. URL: https://eprints.illc.uva.nl/id/eprint/2231/ (cited on p. 29).

[Pei08] Chris Peikert. "Limits on the Hardness of Lattice Problems in l_p Norms". In: *Computational Complexity* 17.2 (2008), pp. 300–351. Earlier version in CCC 2007 (cited on pp. 62, 69).

[Pei09] Chris Peikert. "Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem: Extended Abstract". In: *Proceedings of the ACM Symposium on Theory of Computing (STOC 2009)*. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 333–342 (cited on pp. 40, 41).

[Pei10] Chris Peikert. "An Efficient and Parallel Gaussian Sampler for Lattices". In: *Advances in Cryptology – CRYPTO 2010*. Vol. 6223. Lecture Notes in Computer Science. Springer, 2010, pp. 80–97 (cited on pp. 55, 61).

[PFHK+17] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhan. *FALCON. Technical report, National Institute of Standards and Technology*. https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. Accessed: 28-04-2025. 2017 (cited on pp. 133, 135).

[PFHK+18] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. *Falcon: Fast-Fourier Lattice-Based Compact Signatures over NTRU*. Cryptology ePrint Archive, Report 2018/998. https://eprint.iacr.org/2018/998. 2018 (cited on p. 10).

[PR06] Chris Peikert and Alon Rosen. "Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices". In: *Theory of Cryptography*. Vol. 3876. Lecture Notes in Computer Science. Springer, 2006, pp. 145–166. Full version in ECCC TR05-158 (cited on pp. 55, 60).

[Pra62] Eugene Prange. "The use of information sets in decoding cyclic codes". In: *IRE Transactions on Information Theory* 8.5 (1962), pp. 5–9 (cited on pp. 213, 252).

[PT25] Paola de Perthuis and Filip Trenkić. *Refined Modelling of the Primal Attack, and Variants Against Module-Learning With Errors*. Cryptology ePrint Archive, Paper 2025/2195. 2025. URL: https://eprint.iacr.org/2025/2195 (cited on p. 135).

[PV21] Eamonn W. Postlethwaite and Fernando Virdia. "On the Success Probability of Solving Unique SVP via BKZ". In: *Public-Key Cryptography – PKC 2021*. Ed. by Juan A. Garay. Vol. 12710. Lecture Notes in Computer Science. Springer, 2021, pp. 68–98 (cited on p. 135).

[Reg09] Oded Regev. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: *Journal of the ACM* 56.6 (2009). Earlier version in STOC 2005. arXiv:2401.03703, pp. 1–40 (cited on pp. 40, 41, 47).

[Rog56] C. A. Rogers. “The Number of Lattice Points in a Set”. In: *Proceedings of the London Mathematical Society* s3-6.2 (1956), pp. 305–320 (cited on p. 97).

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Communications of the ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782 (cited on pp. 4, 6).

[Sam13] Pierre Samuel. *Algebraic Theory of Numbers: Translated from the French by Allan J. Silberger*. Courier Corporation, 2013 (cited on p. 128).

[San08] Miklos Santha. *Quantum walk based search algorithms*. 2008. arXiv: 0808.0059 [quant-ph]. URL: https://arxiv.org/abs/0808.0059 (cited on p. 148).

[Sch03] Claus-Peter Schnorr. “Lattice Reduction by Random Sampling and Birthday Methods”. In: *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 – March 1, 2003, Proceedings*. Ed. by Helmut Alt and Michel Habib. Vol. 2607. Lecture Notes in Computer Science. Springer, 2003, pp. 145–156 (cited on p. 105).

[Sch87] Claus-Peter Schnorr. “A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms”. In: *Theoretical Computer Science* 53 (1987), pp. 201–224 (cited on pp. 15, 39, 91, 105, 155).

[SE94] Claus-Peter Schnorr and M. Euchner. “Lattice basis reduction: Improved practical algorithms and solving subset sum problems”. In: *Mathematical Programming* 66 (1994), pp. 181–199 (cited on pp. 15, 105, 155).

[Sho97] Peter Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (1997). Earlier version in FOCS 1994. arXiv:quant-ph/9508027, pp. 1484–1509 (cited on p. 6).

[Sie45] Carl Ludwig Siegel. “A Mean Value Theorem in Geometry of Numbers”. In: *Annals of Mathematics* 46.2 (1945), pp. 340–347 (cited on p. 99).

[Söd11] Anders Södergren. "On the Poisson distribution of lengths of lattice vectors in a random lattice". In: *Mathematische Zeitschrift* 269.3 (2011), pp. 945–954 (cited on pp. 97, 106).

[SS11] Damien Stehlé and Ron Steinfeld. "Making NTRU as Secure as Worst-Case Problems over Ideal Lattices". In: *Advances in Cryptology – EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Vol. 6632. Lecture Notes in Computer Science. Springer, 2011, pp. 27–47 (cited on p. 91).

[SS81] Richard Schroeppel and Adi Shamir. "A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems". In: *SIAM Journal on Computing* 10.3 (1981), pp. 456–464 (cited on p. 213).

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. "Efficient Public Key Encryption Based on Ideal Lattices". In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 617–635 (cited on p. 91).

[Ste88] Jacques Stern. "A method for finding codewords of small weight". In: *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings*. Ed. by Gérard D. Cohen and Jacques Wolfmann. Vol. 388. Lecture Notes in Computer Science. Springer, 1988, pp. 106–113 (cited on p. 213).

[The23] The FPLLL development team. "fplll, a lattice reduction library, Version: 5.4.5". Available at https://github.com/fplll/fplll. 2023. URL: https://github.com/fplll/fplll (cited on pp. 94, 109).

[Uni48] United Nations General Assembly. *Universal Declaration of Human Rights, Article 12*. Adopted 10 December 1948, Resolution 217 A (III). Accessed: 2026-01-30. 1948. URL: https://www.un.org/en/about-us/universal-declaration-of-human-rights (cited on p. 3).

[Wag02] David Wagner. "A Generalized Birthday Problem". In: *Advances in Cryptology – CRYPTO 2002*. Vol. 2442. Lecture Notes in Computer Science. Springer. 2002, pp. 288–304 (cited on pp. 14, 47, 49, 50, 54).

[Was82] Lawrence C. Washington. *Introduction to Cyclotomic Fields*. 1982 (cited on p. 120).

[WBLW25] Yu Wei, Lei Bi, Xianhui Lu, and Kunpeng Wang. "Memory-Efficient BKW Algorithm for Solving the LWE Problem". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2025, pp. 331–362 (cited on p. 86).

[Wol23] Ronald de Wolf. *Quantum Computing: Lecture Notes*. 2023. arXiv: 1907.09415 [quant-ph]. URL: https://arxiv.org/abs/1907.09415 (cited on p. 29).

[XWWG+24] Wenwen Xia, Leizhang Wang, Geng Wang, Dawu Gu, and Baocang Wang. "A Refined Hardness Estimation of LWE in Two-Step Mode". In: *Public-Key Cryptography – PKC 2024*. Ed. by Qiang Tang and Vanessa Teague. Vol. 14603. Lecture Notes in Computer Science. Springer, 2024, pp. 3–35 (cited on p. 134).

[YD17] Yang Yu and Léo Ducas. "Second order statistical behavior of LLL and BKZ". In: *International Conference on Selected Areas in Cryptography*. Springer. 2017, pp. 3–22 (cited on pp. 93, 126).

[YLC14] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. "Fixed-Point Quantum Search with an Optimal Number of Queries". In: *Physical Review Letters* 113.21 (2014). arXiv:1409.3305, p. 210501 (cited on pp. 144, 145).

Nederlandse samenvatting

Klassieke en quantumcryptanalyse van roosters en codes

Dit promotieonderzoek biedt nieuwe inzichten in de veiligheid van cryptografische mechanismen. Bij het gebruik van online diensten (zoals e-mail, sociale media en internetbankieren) wordt vaak aangenomen dat gegevens en communicatie privé blijven en alleen leesbaar zijn voor wie ze bedoeld zijn. Het is echter niet eenvoudig om praktische mechanismen te ontwikkelen om de veiligheid van onze digitale infrastructuur te waarborgen. Het vakgebied *cryptografie* ontwikkelt zulke mechanismen, ook wel cryptografische schema's en protocollen genoemd.

Een encryptieschema is een voorbeeld van een cryptografisch schema en stelt twee partijen in staat om een privébericht uit te wisselen via een openbaar kanaal, zoals het internet. Voordat het bericht wordt verzonden wordt het door middel van een wiskundige berekening onleesbaar gemaakt (versleuteld; Engels: encrypted). Nadat het versleutelde bericht is aangekomen bij de bedoelde ontvanger wordt de berekening teruggedraaid (ontsleuteld) met behulp van een bijpassende geheime sleutel om het oorspronkelijke bericht te herstellen.

Uiteraard wordt het hele doel van encryptie tenietgedaan wanneer iemand het oorspronkelijke bericht kan achterhalen zonder daartoe bevoegd te zijn. Dit zou bijvoorbeeld kunnen gebeuren als de sleutel eenvoudig te raden is of als het encryptieschema bepaalde eigenschappen heeft die benut kunnen worden om het bericht deels of geheel te onthullen. Daarom streeft men naar cryptografische schema's die garanderen dat dergelijke aanvallen onmogelijk zijn, of in elk geval zo tijdrovend of kostbaar dat ze praktisch onhaalbaar zijn.

Voor de meeste cryptografische schema's die in de praktijk worden gebruikt is het niet bekend hoe we wiskundig kunnen bewijzen dat ze aan zulke sterke veiligheidsgaranties voldoen. Wel kunnen we de veiligheid van een schema beter begrijpen door de mogelijke aanvallen te koppelen aan het oplossen van veel bestudeerde computationele problemen. Dit zijn taken die door een computer

kunnen worden opgelost met behulp van een *algoritme*, een stapsgewijze reeks instructies.

Zo kan een aanvaller proberen een versleuteld bericht te ontcijferen met een algoritme dat alle mogelijke sleutels uitprobeert. Cryptografische schema's zijn daarom zo bedacht dat het aantal mogelijke sleutels groot genoeg is om zo'n aanval onpraktisch te maken. Er zijn echter vaak ook andere aanvallen mogelijk. Om snellere en geavanceerdere aanvallen te bedenken kan het helpen om de ontsleutelingstaak als een abstract wiskundig probleem te formuleren. Door nieuwe algoritmen voor zo'n probleem te bedenken en de benodigde rekentijd en computergeheugen te bepalen kan men beoordelen hoe moeilijk het is om het probleem op te lossen. Deze aanpak biedt vervolgens meer inzicht in de veiligheid van onze online gegevens en communicatie.

Als experts voldoende overtuigd zijn dat er geen goed werkende algoritmen bestaan voor het oplossen van een bepaald computationeel probleem, dan draagt dit bij aan het vertrouwen in de daaraan gekoppelde cryptografische schema's. Over het algemeen ontstaat zulk vertrouwen door vele uitgebreide, maar desondanks onsuccesvolle pogingen om de veiligheid van een cryptografisch schema te ondermijnen, bijvoorbeeld door nieuwe algoritmen voor de bijbehorende computationele problemen te bedenken. Al deze inspanningen behoren tot het vakgebied van *cryptanalyse*. Een cryptanalyst houdt zich dus bezig met het analyseren van cryptografische schema's vanuit het perspectief van een aanvaller.

Een mogelijke aanvaller kan over meer rekenkracht beschikken dan de conventionele computers die men dagelijks gebruikt. Ons fysieke universum wordt namelijk op fundamenteel niveau beschreven door de quantummechanica. Deze natuurkundige theorie maakt een andere manier van rekenen mogelijk die niet realiseerbaar is op een klassieke (dat wil zeggen, niet-quantum) computer. Zo ontstond in de jaren tachtig van de twintigste eeuw het idee van een computer die berekeningen uitvoert volgens de wetten van de quantummechanica: de *quantumcomputer*. Deze computer kan quantumalgoritmen uitvoeren die kunnen leiden tot zogenoemde quantumversnellingen voor het oplossen van bepaalde computationele problemen.

De eerste aanwijzing dat quantumversnellingen praktische implicaties zouden kunnen hebben, kwam in 1994, toen Peter Shor een quantumalgoritme ontwikkelde dat twee veel bestudeerde wiskundige problemen aanzienlijk (namelijk exponentieel) sneller oplost dan alle bekende klassieke algoritmen. De veiligheid van diverse cryptografische schema's die op grote schaal worden gebruikt, waaronder RSA en cryptografie gebaseerd op elliptische krommen, berust op

de veronderstelling dat deze problemen niet efficiënt kunnen worden opgelost door bestaande computers. Het door Shor ontwikkelde quantumalgoritme maakt daarmee een groot deel van de momenteel gebruikte cryptografie vatbaar voor aanvallers met voldoende krachtige quantumcomputers: zo'n aanvaller zou versleutelde berichten kunnen onderscheppen en ontsleutelen.

De recente vooruitgang in quantumhardware maakt deze bedreiging voor cryptografie urgenter dan ooit en onderstreept de noodzaak van *postquantumcryptografie*: cryptografie gebaseerd op computationele problemen waarvoor geen efficiënte klassieke of quantumalgoritmen bekend zijn om ze op te lossen en waarvan wordt aangenomen dat deze ook niet bestaan. De afgelopen jaren zijn er diverse postquantum schema's voorgesteld en geanalyseerd, mede gestimuleerd door een standaardiseringsinitiatief uit 2016 van het Amerikaanse National Institute of Standards and Technology (NIST).

Een aantal van deze postquantum schema's zijn inmiddels gestandaardiseerd, maar de volledige overstap naar postquantumcryptografie wordt beschouwd als een complexe en tijdrovende taak. Bovendien blijft het van cruciaal belang om de cryptanalyse van deze standaarden en andere voorgestelde postquantum schema's voort te zetten. Dit is nodig om het vertrouwen in de veiligheid te vergroten, vooral omdat deze nieuwere postquantum schema's en de bijbehorende computationele problemen veel minder grondig zijn onderzocht dan de schema's die ze vervangen. Zoals blijkt uit Shors ontdekking, is het mogelijk dat nieuwe (quantum)aanvallen blijven opduiken.

In dit proefschrift richten we ons op de cryptanalyse (dat wil zeggen, de veiligheidsanalyse) van twee belangrijke klassen binnen de postquantumcryptografie:

- *Roostergebaseerde cryptografie*, die gebruik maakt van wiskundige objecten genaamd roosters (Engels: lattices)
- *Codegebaseerde cryptografie*, die gebruik maakt van foutcorrigerende codes (Engels: error-correcting codes)

Deze klassen behoren tot de meest veelbelovende kandidaten voor cryptografie bestand tegen aanvallen door zowel klassieke als quantumcomputers. Bovendien omvatten ze meerdere cryptografische schema's die door NIST zijn geselecteerd voor standaardisering.

Om bij te dragen aan het begrip van de veiligheid van deze kandidaten ontwikkelen en analyseren we algoritmen voor specifiek hun onderliggende computationele problemen. Hierbij bouwt ons onderzoek vaak voort op varianten van bestaande aanvalsstrategieën.

In deel I van het proefschrift presenteren we onze resultaten die behoren tot de *klassieke cryptanalyse* van roostergebaseerde cryptografie, dat wil zeggen, we analyseren welke aanvallen mogelijk zijn met een klassieke computer.

We bestuderen eerst het *Short Integer Solution probleem (SIS)*, een fundamenteel probleem binnen de roostergebaseerde cryptografie dat aan de basis ligt van de veiligheid van de NIST-standaard ML-DSA. Door een bestaand algoritme vanuit een abstracter perspectief te bekijken en vervolgens technieken gebaseerd op de discrete Gaussverdeling toe te passen, verkrijgen we een sneller algoritme dat subexponentiële rekentijd heeft voor bepaalde parameter regimes van SIS. Deze resultaten zijn asymptotisch en gelden voor voldoende grote invoer, maar lijken (gelukkig, vanuit cryptografisch oogpunt) op dit moment geen bedreiging te vormen voor de concrete veiligheid van ML-DSA.

Onze tweede wetenschappelijke bijdrage is gemotiveerd door de aanwezigheid van extra algebraïsche structuur in veel roostergebaseerde schema's (waaronder alle huidige standaarden). Om te onderzoeken of deze structuur kan worden benut in aanvallen richten we ons op een algoritme genaamd *module-BKZ*. Dit is een gestructureerde variant van het BKZ-algoritme, een veelgebruikt hulpmiddel in de cryptanalyse van roostergebaseerde cryptografie. Door middel van een heuristische analyse, die wordt ondersteund door experimenten, ontwikkelen we een model waarmee we de prestaties van module-BKZ en BKZ kunnen vergelijken. Op basis hiervan identificeren we algebraïsche eigenschappen die bepaalde gestructureerde roosters mogelijk kwetsbaarder maken voor aanvallen dan voorheen werd aangenomen.

In deel II verschuiven we naar het perspectief van een aanvaller met toegang tot een quantumcomputer en richten we ons op de *quantumcryptanalyse* van zowel roostergebaseerde als codegebaseerde cryptografie.

We analyseren eerst het *Shortest Vector Problem (SVP)*, een centraal probleem binnen de roostergebaseerde cryptografie dat nauw verwant is aan het eerdergenoemde SIS-probleem. Momenteel zijn *zeefmethoden* (Engels: sieving methods) de snelste bekende klassieke algoritmen voor het oplossen van SVP. Daarom onderzoeken we in welke mate deze zeefmethoden kunnen worden versneld met behulp van quantumalgoritmen. We presenteren een quantumalgoritme voor het oplossen van SVP dat een verbeterde afweging tussen rekentijd en computergeheugen biedt. Dit resultaat is bereikt door geneste quantumprocedures zorgvuldig te combineren met klassieke voorverwerking.

Daarnaast onderzoeken we of soortgelijke technieken kunnen leiden tot snellere quantumaanvallen op codegebaseerde cryptografie. Onlangs zijn de klassieke

zeefmethoden voor roosters namelijk uitgebreid naar de context van codegebaseerde cryptografie. Hierop voortbouwend ontwikkelen we quantumanalogen voor deze nieuwe zeefmethoden. Hoewel we quantumversnellingen bereiken vergeleken met het snelste klassieke zeefalgoritme, tonen we ook aan waarom deze versnellingen niet leiden tot een verbeterde quantumaanval op codegebaseerde cryptografie.

Welke lessen kunnen we hieruit trekken? Breder bekeken benadrukt dit proefschrift het belang van een passend abstractieniveau en geschikte hulpmiddelen voor cryptanalyse. Voor de asymptotische resultaten met betrekking tot het SIS-probleem en de quantumzeefmethoden voor roosters en codes was het nuttig om de algoritmen vanuit een overkoepelend, abstract perspectief te benaderen. Daarentegen kwam onze analyse van het module-BKZ-algoritme juist voort uit een diepere blik op de specifieke algebraïsche structuur die aanwezig is in veel roostergebaseerde cryptografische schema's, gecombineerd met concrete numerieke experimenten.

Ook laten de resultaten zien dat asymptotische verbeteringen niet altijd leiden tot verbeterde aanvallen in de praktijk, omdat de werkelijke aanvalskosten van algoritmen veel lager kunnen uitpakken dan asymptotische bovengrenzen in theorie suggereren. Ondanks dat een algoritme een asymptotische verbetering biedt kan een ander algoritme dus alsnog de voorkeur krijgen voor praktische aanvallen. Dit onderstreept de noodzaak om theoretische analyses waar mogelijk aan te vullen met experimenten en realistische heuristische modellen.

Kortom, de resultaten van dit proefschrift beantwoorden een aantal natuurlijke vragen binnen de roostergebaseerde en codegebaseerde cryptanalyse en wijzen op verschillende richtingen voor toekomstig onderzoek. Op deze manier dragen we bij aan de overstap naar een digitale infrastructuur die veilig blijft zelfs wanneer cryptografisch relevante quantumcomputers bestaan.

Summary

Classical and quantum cryptanalysis of lattices and codes

In online environments, most people expect their data and communications to remain private and to be readable only by the intended recipients. While this expectation is often taken for granted, it is not easy to design practical mechanisms for securing our online data and communications. The field that develops such mechanisms is called *cryptography*, and these mechanisms are often referred to as cryptographic schemes and protocols. For example, an encryption scheme enables two parties to send a private message over a public channel such as the internet: before the message is sent, it is hidden (encrypted) using some mathematical operation that can be reversed (decrypted) using an associated secret key. The intended recipient can then recover the original message if they know this secret key. Of course, the purpose of encryption is defeated if the message can be recovered by someone without prior knowledge of the secret key, such as by guessing the key or by exploiting some patterns in the encryption method that reveal part of the message. We therefore want some guarantees that such attacks are impossible or at least computationally too expensive to be performed in practice.

For most cryptographic schemes used in practice, we do not know how to mathematically prove such strong security guarantees. Fortunately, the ability to attack a scheme can often be related to solving a well-studied computational problem. If it is widely believed that no efficient algorithms exist for this computational problem, then this belief may contribute to confidence in the scheme's security. More generally, confidence is built through extensive but unsuccessful attempts to find weaknesses in the scheme and its related computational problems. These efforts fall under the field of *cryptanalysis*, which analyzes cryptographic constructions by adopting the mindset of an attacker.

An attacker may have capabilities beyond ordinary computation, since our physical universe is understood to be fundamentally *quantum*. The concept of a quantum computer stems from the 1980s, and employs a model of computation based on the laws of quantum mechanics. These laws enable forms of computation that are

not feasible on a classical (i.e., non-quantum) computer, and in some cases lead to so-called quantum speedups for certain computational problems. The first sign that quantum speedups could have practical implications came in 1994, when Peter Shor designed a quantum algorithm that solves the integer factorization and discrete logarithm problems exponentially faster than all known classical algorithms. The security of various cryptographic schemes that have been widely deployed, including RSA and elliptic-curve cryptography, relies on the assumed difficulty of these problems. Shor's result therefore renders a large portion of currently-used cryptography insecure against attackers with sufficiently powerful quantum computers.

Recent progress in quantum hardware has made this threat to cryptography more urgent and has necessitated the development of *post-quantum cryptography*: cryptography based on computational problems for which no efficient classical *or* quantum algorithms are believed to exist. Many candidates for post-quantum cryptography have been proposed and analyzed, stimulated by a standardization effort that was initiated by the US National Institute of Standards and Technology (NIST) in 2016. While a few post-quantum schemes have now been standardized, the migration to post-quantum cryptography is considered a difficult and time-consuming task. It also remains vitally important to continue cryptanalysis of these standards in order to increase our confidence in their security, not least because the newer post-quantum schemes and their underlying computational problems have received far less scrutiny than the schemes that they replace. Indeed, as illustrated by Shor's discovery, new (quantum) attacks may continue to emerge in the future.

In this thesis, we focus on the cryptanalysis of *lattice-based* and *code-based cryptography*, two of the most promising candidates for post-quantum cryptography, from which multiple schemes have been selected for standardization by NIST. We investigate their security by designing and analyzing algorithms for their underlying computational problems, often building on variants of existing attack strategies.

In Part I, we present our contributions to *classical* lattice-based cryptanalysis. We study the Short Integer Solution problem (SIS), a foundational problem in lattice-based cryptography that underlies the security of the NIST standard ML-DSA. Using a more abstract view of a prior algorithm and discrete-Gaussian techniques, we obtain a subexponential-time algorithm for nontrivial instances of SIS. Fortunately for cryptography, this asymptotic result does not seem to threaten the concrete security of ML-DSA. Our second contribution is motivated by the fact that most practical lattice-based schemes (including all current standards)

consider lattices equipped with additional algebraic structure. We analyze whether this structure can be exploited in lattice attacks, focusing on a structured analog of the BKZ algorithm called module-BKZ. Using a heuristic analysis supported by experiments, we develop a model that enables a performance comparison between module-BKZ and BKZ, allowing us to identify algebraic properties that appear to make structured instances more vulnerable to attack.

In Part II, we switch to the perspective of an attacker with access to a quantum computer, and consider the *quantum* cryptanalysis of lattice-based and code-based cryptography. We explore how quantum algorithms can speed up lattice sieving methods, which are currently the fastest known algorithms for solving the Shortest Vector Problem (SVP), another central problem in lattice-based cryptography closely related to SIS. Roughly speaking, we present a quantum algorithm that achieves a better time-memory trade-off for SVP, using a carefully nested quantum algorithm combined with classical preprocessing. In addition, we investigate whether similar techniques can lead to faster quantum attacks on code-based cryptography. Building on recent work that extended the classical framework of lattice sieving to the setting of codes, we show how to also translate their quantum analogs to code sieving. While we achieve quantum speedups for code sieving itself, we show that this does not result in an improved quantum attack on code-based cryptography.

Conceptually, this thesis illustrates the importance of finding the right level of abstraction and an appropriate set of tools for cryptanalysis. Our asymptotic results on SIS, quantum lattice sieving, and quantum code sieving benefited from viewing the targeted problem and prior algorithms from a high-level perspective. On the other hand, our analysis of the practical performance of module-BKZ was guided by taking a closer look at the internal algebraic structure of lattices used in cryptographic constructions, and by running concrete numerical experiments. Another key takeaway is that asymptotic speedups do not necessarily lead to faster attacks in practice, since the concrete cost of competing algorithms may be much lower than asymptotic upper bounds suggest. This highlights the significance of complementing theory with experiments and heuristic models where available.

Taken together, the results in this thesis address some natural questions in lattice-based and code-based cryptanalysis, and identify several next steps towards a better understanding of lattice-based and code-based cryptography.

Titles in the ILLC Dissertation Series:

ILLC DS-2021-01: **Yfke Dulek**
Delegated and Distributed Quantum Computation

ILLC DS-2021-02: **Elbert J. Booij**
The Things Before Us: On What it Is to Be an Object

ILLC DS-2021-03: **Seyyed Hadi Hashemi**
Modeling Users Interacting with Smart Devices

ILLC DS-2021-04: **Sophie Arnoult**
Adjunction in Hierarchical Phrase-Based Translation

ILLC DS-2021-05: **Cian Guilfoyle Chartier**
A Pragmatic Defense of Logical Pluralism

ILLC DS-2021-06: **Zoi Terzopoulou**
Collective Decisions with Incomplete Individual Opinions

ILLC DS-2021-07: **Anthia Solaki**
Logical Models for Bounded Reasoners

ILLC DS-2021-08: **Michael Sejr Schlichtkrull**
Incorporating Structure into Neural Models for Language Processing

ILLC DS-2021-09: **Taichi Uemura**
Abstract and Concrete Type Theories

ILLC DS-2021-10: **Levin Hornischer**
Dynamical Systems via Domains: Toward a Unified Foundation of Symbolic and Non-symbolic Computation

ILLC DS-2021-11: **Sirin Botan**
Strategyproof Social Choice for Restricted Domains

ILLC DS-2021-12: **Michael Cohen**
Dynamic Introspection

ILLC DS-2021-13: **Dazhu Li**
Formal Threads in the Social Fabric: Studies in the Logical Dynamics of Multi-Agent Interaction

ILLC DS-2021-14: **Álvaro Piedrafita**
On Span Programs and Quantum Algorithms

ILLC DS-2022-01: **Anna Bellomo**
Sums, Numbers and Infinity: Collections in Bolzano's Mathematics and Philosophy

ILLC DS-2022-02: **Jan Czajkowski**
Post-Quantum Security of Hash Functions

ILLC DS-2022-03: **Sonia Ramotowska**
Quantifying quantifier representations: Experimental studies, computational modeling, and individual differences

ILLC DS-2022-04: **Ruben Brokkelkamp**
How Close Does It Get?: From Near-Optimal Network Algorithms to Suboptimal Equilibrium Outcomes

ILLC DS-2022-05: **Lwenn Bussière-Carae**
No means No! Speech Acts in Conflict

ILLC DS-2022-06: **Emma Mojet**
Observing Disciplines: Data Practices In and Between Disciplines in the 19th and Early 20th Centuries

ILLC DS-2022-07: **Freek Gerrit Witteveen**
Quantum information theory and many-body physics

ILLC DS-2023-01: **Subhasree Patro**
Quantum Fine-Grained Complexity

ILLC DS-2023-02: **Arjan Cornelissen**
Quantum multivariate estimation and span program algorithms

ILLC DS-2023-03: **Robert Paßmann**
Logical Structure of Constructive Set Theories

ILLC DS-2023-04: **Samira Abnar**
Inductive Biases for Learning Natural Language

ILLC DS-2023-05: **Dean McHugh**
Causation and Modality: Models and Meanings

ILLC DS-2023-06: **Jialiang Yan**
Monotonicity in Intensional Contexts: Weakening and: Pragmatic Effects under Modals and Attitudes

ILLC DS-2023-07: **Yiyan Wang**
Collective Agency: From Philosophical and Logical Perspectives

ILLC DS-2023-08: **Lei Li**
Games, Boards and Play: A Logical Perspective

ILLC DS-2023-09: **Simon Rey**
Variations on Participatory Budgeting

ILLC DS-2023-10: **Mario Giulianelli**
Neural Models of Language Use: Studies of Language Comprehension and Production in Context

ILLC DS-2023-11: **Guillermo Menéndez Turata**
Cyclic Proof Systems for Modal Fixpoint Logics

ILLC DS-2023-12: **Ned J.H. Wontner**
Views From a Peak: Generalisations and Descriptive Set Theory

ILLC DS-2024-01: **Jan Rooduijn**
Fragments and Frame Classes: Towards a Uniform Proof Theory for Modal Fixed Point Logics

ILLC DS-2024-02: **Bas Cornelissen**
Measuring musics: Notes on modes, motifs, and melodies

ILLC DS-2024-03: **Nicola De Cao**
Entity Centric Neural Models for Natural Language Processing

ILLC DS-2024-04: **Ece Takmaz**
Visual and Linguistic Processes in Deep Neural Networks: A Cognitive Perspective

ILLC DS-2024-05: **Fatemeh Seifan**
Coalgebraic fixpoint logic Expressivity and completeness result

ILLC DS-2024-06: **Jana Sotáková**
Isogenies and Cryptography

ILLC DS-2024-07: **Marco Degano**
Indefinites and their values

ILLC DS-2024-08: **Philip Verduyn Lunel**
Quantum Position Verification: Loss-tolerant Protocols and Fundamental Limits

ILLC DS-2024-09: **Rene Allerstorfer**
Position-based Quantum Cryptography: From Theory towards Practice

ILLC DS-2024-10: **Willem Feijen**
Fast, Right, or Best? Algorithms for Practical Optimization Problems

ILLC DS-2024-11: **Daira Pinto Prieto**
Combining Uncertain Evidence: Logic and Complexity

ILLC DS-2024-12: **Yanlin Chen**
On Quantum Algorithms and Limitations for Convex Optimization and Lattice Problems

ILLC DS-2024-13: **Jaap Jumelet**
Finding Structure in Language Models

ILLC DS-2025-01: **Julian Chingoma**
On Proportionality in Complex Domains

ILLC DS-2025-02: **Dmitry Grinko**
Mixed Schur-Weyl duality in quantum information

ILLC DS-2025-03: **Rochelle Choenni**
Multilinguality and Multiculturalism: Towards more Effective and Inclusive Neural Language Models

ILLC DS-2025-04: **Aleksi Anttila**
Not Nothing: Nonemptiness in Team Semantics

ILLC DS-2025-05: **Niels M. P. Neumann**
Adaptive Quantum Computers: decoding and state preparation

ILLC DS-2025-06: **Alina Leidinger**
Towards Language Models that benefit us all: Studies on stereotypes, robustness, and values

ILLC DS-2025-07: **Zhi Zhang**
Advancing Vision and Language Models through Commonsense Knowledge, Efficient Adaptation and Transparency

ILLC DS-2025-08: **Sophie Klumper**
The Gap and the Gain: Improving the Approximate Mechanism Design Frontier in Constrained Environments

ILLC DS-2025-09: **Jordi Rudo Weggemans**
Quantum versus classical resources in computational complexity

ILLC DS-2026-01: **Bryan Eikema**
A Sampling-Based Exploration of Neural Text Generation Models

ILLC DS-2026-02: **Marten Folkertsma**
Empowering Quantum Computation with: Measurements, Catalysts, and Guiding States

ILLC DS-2026-03: **Valentin Richard**
Presuppositional and Dynamic Aspects of Questions

ILLC DS-2026-04: **Puyu Yang**
Bringing Science to the Public: The Role of Wikipedia in Scientific Communication

ILLC DS-2026-05: **Johannes Kloibhofer**
Cycles with Annotations: Non-Wellfounded Proof Theory of Modal Fixpoint Logics

ILLC DS-2026-06: **Danish Kashaev**
Approximation via duality in offline, online and strategic settings

ILLC DS-2026-07: **Oskar van der Wal**
Taking a Step Back: Measuring and Mitigating Bias in Language Models

ILLC DS-2026-08: **Lynn Engelberts**
Classical and Quantum Cryptanalysis of Lattices and Codes