Bridging Early and Modern Lattice Cryptosystems: A Theoretical Journey through SVP, LWE and Kyber

by

Daniel Santana

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2025

© Daniel Santana 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis offers an in-depth theoretical study of lattice-based cryptography, tracing its evolution from early foundational systems to modern standardized constructions. Initially, we examine and emphasize the significance of the Ajtai-Dwork cryptosystem and its foundational worst-case to average-case reductions based on lattice problems. The thesis then details Regev's Learning with Errors (LWE) problem and shows its impact on the development of practical public-key schemes while maintaining the security guarantees from the well-studied worst-case lattice problems.

Further, we discuss the structured variants such as Ring-LWE and Module-LWE, showing how they improve the efficiency and scalability of the lattice-based schemes while maintaining the security foundations of the worst-case to average-case reductions. This analysis then culminates in the study of ML-KEM, a lattice-based scheme recently standardized by NIST, examining its specific design choices, optimizations, and security proofs. Finally, we investigate algorithmic methods for solving the fundamental lattice problems, analysing exact Shortest Vector Problem (SVP) solvers including Kannan's enumeration algorithm and the AKS sieve algorithm, in order to understand their implications on the hardness assumptions underlying the lattice-based cryptographic security.

In summary, this work shows that lattice-based cryptography offers a secure and efficient foundation for (post-quantum) cryptographic schemes, with strong theoretical bases and practical implementations.

Acknowledgements

I want to thank everyone who helped me complete this thesis and supported me along the way. First, I'd like to thank my supervisors, Dr. Rafael Oliveira, who guided me and always pushed me to think deeper, and Dr. Mohammad Hajiabadi, for his steady support and direction.

I am also grateful to Prof. Sam Jaques, who first introduced me to the world of Post-Quantum Cryptography and whose teachings inspired the direction of this thesis.

I also want to thank my family – Claudia, João and Felipe – and especially my wife Pamela, whose support and belief in me gave me strength throughout this journey.

Special thanks go to Rasoul Akhavan Mahdavi, Pranshu Kumar and Steven Lee, who listened patiently to my endless talks about lattices and were always there to help me when I needed it. Their insights and discussions were invaluable in shaping my understanding of the subject.

Finally, I couldn't have finished my master's degree without the encouragement and support from: Alexandru Bara, Aswad Tariq, Ben Schneider, Sina Kamali, Camryn Steckel, and several others friends and colleagues from the CrySP lab. Your collaboration, insightful discussions, and necessary distractions made this journey enjoyable.

Dedication

To Pamela, my parents Claudia and João, and my brother Felipe, with all my love and gratitude.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures x

List of Tables xi

List of Algorithms xii

1 Introduction 1

1.1 Post-Quantum Cryptography . . . 1
1.2 Start of Lattice-based Cryptography . . . 1
1.3 Learning with Errors . . . 2
1.4 Scalability and Efficiency . . . 2
1.5 NIST PQC Standardization . . . 3
1.6 Overview of the Thesis . . . 3

2 Background 5

2.1 Notation . . . 5
2.2 Probability and Complexity . . . 6
2.2.1 Chernoff Bound . . . 7
2.3 Lattices Theory and Hard Problems . . . 7
2.3.1 Lattice Definitions and Bases . . . 7

2.3.2 Minkowski's Theorem and Successive Minima . . . 8
2.3.3 Dual Lattice . . . 9
2.3.4 Discrete Gaussian and Smoothing Parameter . . . 10
2.3.5 Lattice Problems . . . 11
2.3.6 Lattice Reductions . . . 13
2.4 Cryptographic Notions . . . 15
2.4.1 Security Parameter and Adversary Model . . . 15
2.4.2 Public-Key Encryption (PKE) . . . 16
2.4.3 IND-CPA and IND-CCA Security for Public-Key Encryption . . . 16
2.4.4 Key Encapsulation Mechanism (KEM) . . . 18
2.4.5 IND-CCA Security for KEMs . . . 18

3 Early Cryptosystems **20**
3.1 Ajtai-Dwork Cryptosystem . . . 20
3.1.1 Background . . . 21
3.1.2 Public-key Encryption Scheme . . . 21
3.1.3 Main Results . . . 22
3.1.4 Proofs . . . 25

4 LWE, Ring-LWE and Module-LWE **44**
4.1 Regev's LWE Cryptosystem . . . 44
4.1.1 LWE Problem . . . 45
4.1.2 Main Result . . . 45
4.1.3 Public-key Encryption Scheme . . . 50
4.1.4 Alternative Form . . . 51
4.1.5 Analysis of the Cryptosystem . . . 52
4.1.6 Proofs . . . 57
4.2 Ring-LWE . . . 82
4.2.1 Background and Definitions . . . 82
4.2.2 Main Result . . . 85
4.2.3 Public-key Encryption Scheme . . . 88
4.3 Module-LWE . . . 90
4.3.1 Background and Definitions . . . 90
4.3.2 Module-LWE Problem . . . 91
4.3.3 Main Result . . . 93
4.3.4 Public-key Encryption Scheme . . . 94

5 Module Lattice KEM **96**

5.1 Introduction . . . 96
5.2 Kyber PKE . . . 97
5.2.1 Simplified Kyber PKE . . . 97
5.2.2 Differences from Previous Schemes . . . 99
5.3 Security . . . 100
5.4 Optimizations . . . 108
5.4.1 Public Key Generation . . . 108
5.4.2 Secret and Error Sampling . . . 109
5.4.3 NTT Multiplication . . . 111
5.4.4 Compression . . . 117
5.5 Full Scheme: `K-PKE` . . . 119
5.6 Security of `K-PKE` (Revisited) . . . 121
5.7 IND-CCA on `K-PKE`? . . . 122
5.8 FO Transform . . . 123
5.8.1 Security of `Simp-KEM` . . . 125
5.9 ML-KEM . . . 127
5.9.1 Security of `ML-KEM` . . . 129
5.10 Parameter Sets . . . 130
5.11 Conclusion . . . 130

6 SVP solvers **131**

6.1 Kannan's Enumeration SVP algorithm . . . 131
6.1.1 SVP Algorithm . . . 131
6.1.2 Enumeration Procedure . . . 132
6.1.3 Select Basis Procedure . . . 136
6.1.4 Analysis of Kannan's SVP Algorithm . . . 138
6.2 AKS Sieve Algorithm for the SVP . . . 146
6.2.1 SVP Algorithm . . . 146
6.2.2 Pre-processing . . . 147
6.2.3 Sampling Procedure . . . 147
6.2.4 Sieving Procedure . . . 148
6.2.5 Analysis of the AKS Sieve Algorithm . . . 149

7 Conclusion **154**

References **157**

List of Figures

4.1 Steps for the reduction from Worst-case SIVP/GapSVP to Average-case (Decision) LWE 50

6.1 Tossing of $\boldsymbol{y}$ 151

List of Tables

5.1 Parameter sets for the ML-KEM scheme, based on the NIST PQC standard [MLK24]. 130

List of Algorithms

1 Babai's Rounding Off Algorithm, [Bab86] . . . 14
2 Babai's Nearest Plane Algorithm, [Bab86] . . . 15
3 Simp-PKE.KeyGen() . . . 98
4 Simp-PKE.Enc(pk, μ) . . . 98
5 Simp-PKE.Dec(sk, $\boldsymbol{c}$) . . . 99
6 Simp-PKE.KeyGen() (with public key matrix $\boldsymbol{A}$ generated from a seed) . . . 109
7 Simp-PKE.Enc(pk, μ) (with public key matrix $\boldsymbol{A}$ generated from a seed) . . . 109
8 Simp-PKE.KeyGen() (with small vectors $\boldsymbol{s}, \boldsymbol{e}$ generated by CBD) . . . 110
9 Simp-PKE.Enc(pk, μ) (with small vectors $\boldsymbol{s}, \boldsymbol{e}$ generated by CBD) . . . 111
10 Simp-PKE.KeyGen() (with NTT multiplication) . . . 116
11 Simp-PKE.Enc(pk, μ) (with NTT multiplication) . . . 117
12 Simp-PKE.Dec(sk, $\boldsymbol{c}$) (with NTT multiplication) . . . 117
13 K-PKE.KeyGen() . . . 119
14 K-PKE.Enc(pk, μ) . . . 120
15 K-PKE.Dec(sk, $\boldsymbol{c}$) . . . 120
16 Simp-KEM.KeyGen() . . . 123
17 Simp-KEM.Encaps(pk) . . . 124
18 Simp-KEM.Decaps(sk, $\boldsymbol{c}$, pk) . . . 124
19 ML-KEM.KeyGen() . . . 128
20 ML-KEM.Encaps(pk) . . . 128
21 ML-KEM.Decaps(sk, $\boldsymbol{c}$) . . . 129
22 SHORTEST . . . 133
23 ENUMERATE . . . 134
24 LIST . . . 135
25 SELECT_BASIS($n; \boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_{n+1}}$) . . . 137
26 AKS SVP . . . 146
27 SAMPLE . . . 147
28 SIEVE . . . 148

Chapter 1

Introduction

1.1 Post-Quantum Cryptography

The advent of quantum computers has the potential to break many of the cryptographic schemes that are currently in use. In particular, Shor's algorithm [Sho94] can solve the integer factorization and discrete logarithm problem in polynomial time, which are the basis of many asymmetric, or public-key, cryptographic schemes, such as RSA and ECC. This has led to the research and development into Post-Quantum Cryptography (PQC) schemes, which are cryptosystems believed to be secure even against quantum adversaries, and rely on different underlying problems to base their security on. Among the various candidates for PQC are lattice-based cryptosystems, which have their security based on the conjectured hardness of mathematical problems on lattices.

Lattice-based cryptography has been a subject of research for several years, but it gained significant attention from their ability to base security on *worst-case* hardness assumptions, that is, breaking even a single instance of a cryptographic scheme would imply the ability to solve the hardest instances of a lattice problem. This strong security guarantee, as well as the apparent resistance to known quantum algorithms, has made lattice-based cryptography a leading candidate and focus for post-quantum cryptography.

1.2 Start of Lattice-based Cryptography

The theoretical foundations of lattice-based cryptography were laid out by Ajtai in 1996 [Ajt96], who demonstrated the first worst-case to average-case reduction for lattice problems. This was a significant breakthrough, since it showed that a cryptographic scheme that was designed around an average-case problem (i.e. randomly generated instances), could be as hard to break as solving the hardest instances of an underlying lattice problem, such as the Shortest Vector Problem (SVP). Ajtai's work introduced a hard problem, now known as the Short Integer Solution (SIS), which became a basis for several modern lattice-based cryptographic schemes.

Following this, the Ajtai-Dwork (AD) cryptosystem [AD97] was introduced as the first lattice-based public-key cryptosystem to provably show security based on the worst-case

hardness of a lattice problem, specifically the unique Shortest Vector Problem (uSVP). The AD cryptosystem was a significant milestone, establishing the theoretical foundation for lattice-based public-key cryptography, that is, showing that if a random instance of their scheme could be distinguished, it implied that a polynomial-time algorithm could solve the worst-case unique Shortest Vector Problem (uSVP). While the AD cryptosystem was a theoretical breakthrough, it had several limitations, such as very large public key sizes and high computational costs, making it impractical for real-world applications. However, it laid the theoretical groundwork for future lattice-based cryptographic schemes, which demonstrated the potential of lattices as a foundation for secure cryptographic systems.

1.3 Learning with Errors

Despite the theoretical advances, its practical limitations showed the need for more efficient lattice-based constructions. In 2005, Regev introduced the Learning With Errors (LWE) problem [Reg05], which provided a simpler yet powerful foundation for cryptography by showing a worst-case to average-case reduction from standard lattice problems like GapSVP and SIVP to the average-case LWE problem. This implied that schemes built on LWE would inherit the hardness of these lattice problems, making them secure against both classical and quantum attacks.

Regev's work also introduced a basic public-key encryption scheme based on LWE, where the security relies on the difficulty of distinguishing between LWE samples, which are noisy linear equations, and uniformly random samples. This noise, or error term added to the linear equations is necessary to ensure security, since it prevents an adversary to use linear algebra techniques, such as Gaussian elimination, to solve the equations and recover the secret key. However, it is important to note that the cryptosystem was still impractical, as it required large key sizes and computational costs, which made it unsuitable for real-world applications. Several improvements were made to the LWE-based schemes throughout the years, but it wasn't until the introduction of the Ring-LWE problem [LPR12] that practical lattice-based cryptography became feasible.

1.4 Scalability and Efficiency

By leveraging lattices with additional algebraic structure and applying them to the LWE problem, Lyubashevsky, Peikert, and Regev [LPR12] introduced the Ring-LWE problem, which used polynomial rings to create more efficient cryptographic schemes. This was a very promising solution, since it allowed for smaller key sizes, and, for example, could encrypt n (degree of the polynomial ring) bits of data with a single ciphertext, as opposed to the LWE problem, which required n ciphertexts to encrypt the same amount of data. Also, by leveraging the ring structure, using the Number Theoretic Transform (NTT), the polynomial multiplications could be performed very efficiently, (in $O(n \log n)$ time, instead of $O(n^2)$), which made the schemes even more practical.

The only issue, however, was that the Ring-LWE problem introduced additional structure that could potentially be exploited by an adversary, and its hardness was proved under the worst-case hardness of the SIVP problem in *ideal lattices*, which do not share the same

hardness guarantees as problems in *general lattices*. To this point, there are no known attacks that can exploit this additional structure, and the best known algorithms perform no better than in the general lattice, but it still a potential risk to be considering when designing cryptosystems based solely on the Ring-LWE problem.

This led to the development, or generalization, of the Ring-LWE to the Module-LWE problem [BV11], which can be seen as bridging the gap between the Ring-LWE and (Plain) LWE problems. The Module-LWE problem looks like the LWE problem, however, the elements of the matrices and vectors are now elements of a polynomial ring, like the Ring-LWE problem. This generalization allowed for a trade-off between the efficiency and security of the scheme, making use of the optimizations from the Ring-LWE problem, NTT multiplications and less ciphertexts to communicate the same message, but also allowing the use of different module ranks, removing part of the structure that could be exploited by an adversary. This is the direct basis for the more efficient lattice-based schemes.

1.5 NIST PQC Standardization

In 2016, with the growing need and interest for post-quantum cryptography, the National Institute of Standards and Technology (NIST) initiated a process to standardize post-quantum cryptographic algorithms, making a call for proposals for post-quantum cryptographic schemes[1], and, in 2022, announced the selected algorithms to be standardized[2]. Among them, there are several lattice-based schemes, such as CRYSTALS-Kyber ([SAB+20]), which was the first selected in the key encapsulation mechanism (KEM) category, being standardized in 2024 in the FIPS 203 as ML-KEM [MLK24].[3]

Kyber, or ML-KEM, is a culmination of decades of research in lattice-based cryptography, instantiating the Module-LWE problem, offering practical efficiency and provable security. Note that the ML-KEM is not a direct implementation of the Module-LWE problem, but rather a specific instantiation of it, along with the application of the Fujisaki-Okamoto (FO) transform [FO99] to achieve IND-CCA security, and several other optimizations to improve the performance of the scheme.

1.6 Overview of the Thesis

This thesis embarks on a theoretical journey, discussing the foundations and development of lattice-based cryptography from its earlier concepts to its modern standardized forms. We highlight how the hardness of lattice problems has been used to build secure cryptographic schemes, such as the Ajtai-Dwork cryptosystem, the Learning With Errors problem, as well as the Ring-LWE and Module-LWE problems, culminating in the NIST standardization of the ML-KEM scheme, and end with two (exact) SVP solvers, Enumeration and Sieving.

The thesis is structured as follows:

[1] Available in: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals

[2] Available in: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms

[3] In March 2025, NIST announced the HQC algorithm as another KEM standard, which is not lattice-based.

- Chapter 2: Background provides the necessary mathematical and cryptographic background, including notation, definitions related to lattices and their hard problems, as well as the cryptographic notions like IND-CPA and IND-CCA security.
- Chapter 3: Early Cryptosystems, discusses the early lattice-based cryptosystem, focusing on the Ajtai-Dwork cryptosystem and its proofs and connections to the unique Shortest Vector Problem (uSVP).
- Chapter 4: LWE, Ring-LWE and Module-LWE, introduces the Learning With Errors (LWE) problem, and its evolution though the Ring-LWE and Module-LWE problems, focusing on the security reductions and their respective cryptographic schemes.
- Chapter 5: Module Lattice KEM, discusses the NIST standardization of the ML-KEM scheme, its construction, design choices, optimizations, and security proofs.
- Chapter 6: SVP solvers, presents two exact Shortest Vector Problem (SVP) solvers, Kannan's Enumeration and AKS Sieve algorithms, which are fundamental for showing the hardness of lattice problems and their applications in cryptography.

By navigating this theoretical journey, we aim to provide a comprehensive understanding of the evolution of lattice-based cryptography, showing how the reductions and constructions are proved secure, how they have led to the development of practical cryptographic schemes that are now being standardized and adopted in the post-quantum era, and how the best known algorithms to solve the underlying hard problems are still exponential, making them suitable for cryptographic applications.

Chapter 2

Background

2.1 Notation

We define here some of the notation used throughout this paper, however, some will be defined in the corresponding sections to avoid cluttering this section. This is not exhaustive, but it is meant to be a reference for the most common symbols used in the paper. We follow mostly the notation used in [Pei15] and [SAB+20].

$\mathcal{R}$	A ring, usually $\mathbb{Z}[X]/(X^n+1)$, where n is the degree of the polynomial, and $\mathbb{Z}$ is the ring of integers.
$\mathcal{R}_q$	A ring $\mathcal{R}$ with coefficients reduced modulo q, usually the ring $\mathbb{Z}_q[X]/(X^n+1)$.
v	Regular font is used for elements of a ring $\mathcal{R}$ or $\mathcal{R}_q$ (includes elements in $\mathbb{Z}, \mathbb{Z}_q$).
$\boldsymbol{v}$	Bold letters is used for (column) vectors with coefficients in a ring $\mathcal{R}$ or $\mathcal{R}_q$ (or, in $\mathbb{Z}$ or $\mathbb{Z}_q$).
$\boldsymbol{A}$	A matrix of elements in a ring $\mathcal{R}$ or $\mathcal{R}_q$, or, an ordered set of column vectors $[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$.
$\boldsymbol{v}^T, \boldsymbol{A}^T$	The transpose of a vector or matrix, respectively.
$\mathcal{L}, \Lambda$	A lattice (refer to Section 2.3.1), which is a discrete additive subgroup of $\mathbb{R}^n$.
$\mathcal{L}^*$	The dual lattice of $\mathcal{L}$, defined in Def. 2.3.3.
$\lvert\cdot\rvert$	The absolute value of a number.
$\lVert\cdot\rVert = \lVert\cdot\rVert_2$	The ℓ_2 norm, or Euclidean norm, of a vector.
$B_n(r)$	The ball in $\mathbb{R}^n$ of radius r around the origin, defined as $B_n(r) = \{ \boldsymbol{x} \in \mathbb{R}^n : \lVert\boldsymbol{x}\rVert \leqslant r \}$.

$B_n(r, \boldsymbol{x})$	The ball in $\mathbb{R}^n$ of radius r around the point $\boldsymbol{x}$, defined as $B_n(r, \boldsymbol{x}) = \{ \boldsymbol{y} \in \mathbb{R}^n : \|\boldsymbol{y} - \boldsymbol{x}\| \leqslant r \}$.
$\langle \cdot , \cdot \rangle$	The inner product of two vectors, defined as $\langle \boldsymbol{x} , \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i$ for $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$.
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	Sets of algorithms, usually with a specific purpose, such as $\mathcal{A}$ for an adversary, $\mathcal{B}$ for a reduction.
$\mathsf{Adv}^{cpa}_{\texttt{K-PKE}}(\mathcal{A})$	The advantage of an adversary $\mathcal{A}$ in breaking the IND-CPA security of the `K-PKE` scheme (See Def. 2.2.1).
$\leftarrow$	The assignment operator, used to assign a value to a variable, e.g., $\boldsymbol{x} \leftarrow \text{Alg}(\boldsymbol{v})$ means that the algorithm Alg is run with input $\boldsymbol{v}$ and the output is assigned to the variable $\boldsymbol{x}$. The algorithm is deterministic.
$\leftarrow\!\!\$$	The sampling operator, used to sample a value from a distribution, e.g., $\boldsymbol{x} \leftarrow\!\!\$ \mathcal{D}$ means that $\boldsymbol{x}$ is sampled from the distribution $\mathcal{D}$. The sampling is usually done uniformly at random. It is also being used for probabilistic algorithms, e.g., $\boldsymbol{x} \leftarrow\!\!\$ \text{Alg}(\boldsymbol{v})$ means that the algorithm Alg is run with input $\boldsymbol{v}$ and the output is assigned to the variable $\boldsymbol{x}$. The algorithm is probabilistic.

2.2 Probability and Complexity

Here we define some of the basic notions of probability and complexity that will be used throughout the paper, essential for defining and proving the security of the schemes presented in this work.

We begin by defining the notion of a Probabilistic Polynomial Time (PPT) Algorithm, which is an algorithm that runs in polynomial time in every input of length size n, the security parameter. All adversaries are assumed to be PPT unless stated otherwise.

When we talk about adversaries, or algorithms, we are usually interested in their *advantage* in distinguishing between two distributions, or in breaking a cryptographic scheme, defined as follows:

Definition 2.2.1 (Advantage)**.** *The advantage of an adversary $\mathcal{A}$ in distinguishing between two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ is defined as:*

$$\mathsf{Adv}_n^{\mathcal{D}_0, \mathcal{D}_1}(\mathcal{A}) := |\Pr[\mathcal{A}(\boldsymbol{x}) = 1 \mid \boldsymbol{x} \leftarrow\!\!\$ \mathcal{D}_0] - \Pr[\mathcal{A}(\boldsymbol{x}) = 1 \mid \boldsymbol{x} \leftarrow\!\!\$ \mathcal{D}_1]|$$

where $\boldsymbol{x}$ is the input to the adversary $\mathcal{A}$ and n is the security parameter.

A common requirement for a cryptographic scheme is that the *advantage* of any PPT adversary in breaking the scheme is negligible, $\mathsf{negl}(n)$, which is defined as a function that is asymptotically smaller than any polynomial in n, i.e. $\mathsf{negl}(n) < \frac{1}{p(n)}$ for any polynomial $p(n)$ (i.e. $\mathsf{negl}(n) < n^{-\omega(1)}$), for sufficiently large n.

2.2.1 Chernoff Bound

Several of the results in this paper makes use of the Chernoff bound, which is a powerful tool to bound the probability of the sum of independent random variables deviating from its expected value.

There are several versions of the Chernoff bound, but we will mostly use the following ones, with the latter applying Hoeffding's inequality, giving us the following inequalities:

Lemma 2.2.2 (Chernoff's bound)**.** *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables and $X = \sum_{i=1}^{n} X_i$ with expectation $\mu = E[X]$. For any $0 < \delta < 1$,*

$$\Pr[X \leqslant (1-\delta)\mu] < \exp\left(-\frac{\delta^2\mu}{2}\right)$$
$$\Pr[X \geqslant (1+\delta)\mu] < \exp\left(-\frac{\delta^2\mu}{3}\right)$$
$$\Rightarrow \Pr[|X-\mu| \geqslant \delta\mu] < 2\exp\left(-\frac{\delta^2\mu}{3}\right)$$

Equivalently, the sum of n independent trials deviates from its mean by more than $\delta\mu$ with a bounded probability.

Lemma 2.2.3 (Chernoff–Hoeffding's bound)**.** *Let $X_1, \ldots, X_n$ be independent $\{0,1\}$-valued random variables and $X = \sum_{i=1}^{n} X_i$ with expectation $\mu = E[X]$. For any $t > 0$,*

$$\Pr[X \leqslant \mu - t] < \exp\left(-\frac{2t^2}{n}\right)$$
$$\Pr[X \geqslant \mu + t] < \exp\left(-\frac{2t^2}{n}\right)$$
$$\Rightarrow \Pr[|X-\mu| \geqslant t] < 2\exp\left(-\frac{2t^2}{n}\right)$$

Equivalently, the sum of n independent trials deviates from its mean by more than t except with a bounded probability.

Throughout the paper we invoke this bound to argue that repeated experiments or samplings behave very close to their expected values.

2.3 Lattices Theory and Hard Problems

2.3.1 Lattice Definitions and Bases

A lattice $\mathcal{L}$ is defined as a discrete additive subgroup of $\mathbb{R}^n$, that is, $\mathbf{0} \in \mathcal{L}$, and $-\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{y} \in \mathcal{L}$ for every $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}$, and also, for every lattice point, there is a neighbourhood in $\mathbb{R}^n$ which it is the only lattice point. The simplest example of a lattice is $\mathbb{Z}^n$. In this paper, we will consider the lattices to be full rank, spanning the whole space $\mathbb{R}^n$.

A lattice can be finitely generated by linear combinations of a basis, represented by a set of linearly independent vectors $\boldsymbol{B} := \{ \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \}$:

$$\mathcal{L} = \mathcal{L}(\boldsymbol{B}) := \left\{ \sum_{i=1}^{n} \beta_i \boldsymbol{b}_i : \beta_i \in \mathbb{Z} \right\}$$

A basis of a lattice is not unique, and, if $\boldsymbol{B}$ and $\boldsymbol{B'}$ are two bases of the same lattice, then $|\det(\boldsymbol{B})| = |\det(\boldsymbol{B'})|$. This is true because if we write $\boldsymbol{B'}$ as a linear combination of the vectors of the basis $\boldsymbol{B}$, then we can say that $\boldsymbol{B'} = \boldsymbol{BU}$, where $\boldsymbol{U} \in \mathbb{Z}^{n \times n}$ is a matrix with integer coefficients and $\det(\boldsymbol{U}) = \pm 1$.

Volume and Fundamental Region

We can also define the volume of a lattice $\mathrm{Vol}(\mathcal{L}) = |\det(\boldsymbol{B})|$, and from this we can see that the volume is a property of a lattice, being the same for any basis of the same lattice. The volume of the lattice is the volume of the fundamental region, called fundamental parallelepiped, defined by

$$\mathcal{P}(\boldsymbol{B}) := \boldsymbol{B} \cdot [0,1)^n = \sum_{i=1}^{n} \boldsymbol{b}_i \cdot [0,1)$$

Another fundamental parallelepiped that is commonly used is the centered fundamental parallelepiped, defined as $\mathcal{P}(\boldsymbol{B}) = \boldsymbol{B} \cdot [-\frac{1}{2}, \frac{1}{2})^n$

2.3.2 Minkowski's Theorem and Successive Minima

Minimum distance and Successive Minima

Because a lattice is a discrete subgroup, there exists a shortest non-zero vector, and we can define the minimum distance $\lambda_1(\mathcal{L})$ as:

$$\lambda_1(\mathcal{L}) := \min_{\boldsymbol{v} \in \mathcal{L} \setminus \{ 0 \}} \|\boldsymbol{v}\|$$

We can also define the ith successive minimum $\lambda_i(\mathcal{L})$ as the radius of the smallest ball $B_n(r)$ that contains at least i (for $i = 1, \ldots, n$) linearly independent lattice vectors.

Theorem 2.3.1 (Minkowski's first theorem)**.** *For any full-rank lattice $\mathcal{L}$ of rank n,*

$$\lambda_1(\mathcal{L}) \leqslant \sqrt{n} (\det(\mathcal{L}))^{1/n}$$

Definition 2.3.2 (Hermite Constant)**.** *The Hermite constant γ_n is the maximum over all n-rank lattices $\mathcal{L}$ of $\gamma_n = \max(\lambda_1(\mathcal{L})/vol(\mathcal{L})^{1/n})^2$. Asymptotically γ_n satisfies:*

$$\frac{n}{2\pi e} + \frac{\log(\pi n)}{2\pi e} \leqslant \gamma_n \leqslant \frac{1.744n}{2\pi e} + o(n) \tag{2.1}$$

Remark (Why γ_n matters)**.** *Hermite's constant gives the best universal constant relating the determinant (volume) to the shortest nonzero vector:*

$$\lambda_1(\mathcal{L}) \;\leq\; \sqrt{\gamma_n}\;\det(\mathcal{L})^{1/n}.$$

So if we know $\det(\mathcal{L})$*, we immediately get a universal bound on how short a nonzero vector can be (and the packing radius is at most half of that). This also gives a clean baseline to compare algorithms against (e.g. Babai's, LLL), and the right scale for estimates elsewhere (e.g., dual lengths and Gaussian/smoothing bounds in the LWE parts). As* n *grows,* $\sqrt{\gamma_n} \approx \sqrt{n/(2\pi e)}$*, so the order of magnitude is clear.*

2.3.3 Dual Lattice

Definition 2.3.3 (Dual Lattice)**.** *Let* $\mathcal{L}$ *be a full-rank lattice in* $\mathbb{R}^n$*, the dual lattice* $\mathcal{L}^*$ *is defined as:*

$$\mathcal{L}^* := \{\, \boldsymbol{v} \in \mathbb{R}^n \;\mid\; \langle \boldsymbol{v}\,,\, \boldsymbol{w}\rangle \in \mathbb{Z}, \forall \boldsymbol{w} \in \mathcal{L}\,\}$$

In words, the dual lattice, sometimes called the *reciprocal*, is the set of points whose inner product with the vectors in $\mathcal{L}$ are integers. For any vector $\boldsymbol{w} \in \mathcal{L}$, this integer inner product are points that form a set of hyperplanes perpendicular to $\boldsymbol{w}$ separated by a distance of $1/\|\boldsymbol{w}\|$.

This definition allows for some important relationships between the original lattice $\mathcal{L}$ and its dual $\mathcal{L}^*$, which are fundamental in lattice theory and cryptographic applications. These however will not be proved in this work, but can be found in several places, such as Regev's [Reg04c] or Micciancio's [Mic14] lecture notes.

Definition 2.3.4 (Dual Basis)**.** *For a basis* $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^{m\times n}$*, define the dual basis* $\boldsymbol{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n) \in \mathbb{R}^{m\times n}$ *as the unique basis that satisfies:*

- $span(\boldsymbol{D}) = span(\boldsymbol{B})$
- $\boldsymbol{B}^T\boldsymbol{D} = I$

Thus, from the second condition, we can say that $\boldsymbol{D} := (\boldsymbol{B}^T)^{-1}$ is a basis of $\mathcal{L}^*$ and that each vector of $\boldsymbol{D}$ is such that $\langle \boldsymbol{b}_i\,,\, \boldsymbol{d}_j\rangle = \delta_{ij}$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.

We can also say that, for any lattice $\mathcal{L}$,

$$(\mathcal{L}^*)^* = \mathcal{L} \qquad \text{and} \qquad \det(\mathcal{L}^*) = 1/\det(\mathcal{L}).$$

The following theorem is due to Banaszczyk [Ban93].

Theorem 2.3.5 (Transference Theorem (Banaszczyk))**.** *For any lattice* $\mathcal{L}$*,*

$$1 \leqslant \lambda_1(\mathcal{L}) \cdot \lambda_n(\mathcal{L}^*) \leqslant n$$

2.3.4 Discrete Gaussian and Smoothing Parameter

In several of the definitions and algorithms in this paper, we will use the discrete Gaussian distribution, which is a distribution over a lattice $\mathcal{L}$ that assigns a probability to each point in the lattice in a way that is similar to the Gaussian distribution. The definitions shown here are adapted from [Pei15].

Definition 2.3.6 ((Continuous) Gaussian Distribution)**.** *For any positive integer n and real width $r > 0$, which is taken as $r = 1$ when not specified, the Gaussian function is defined as:*

$$\rho_r(\boldsymbol{x}) := \exp\left(\frac{-\pi\|\boldsymbol{x}\|^2}{r^2}\right) = \rho(\boldsymbol{x}/r),$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $\rho_r(\boldsymbol{x}) = \prod_{i=1}^n \rho_r(x_i)$.

The (continuous) Gaussian distribution $\mathcal{D}_r$ is defined as the probability distribution over $\mathbb{R}^n$ with probability density function proportional to $\rho_r(\boldsymbol{x})$:

$$f(\boldsymbol{x}) := \frac{\rho_r(\boldsymbol{x})}{\int_{\mathbb{R}^n} \rho_r(\boldsymbol{y})d\boldsymbol{y}} = \frac{\rho_r(\boldsymbol{x})}{r^n}$$

We can then define the discrete Gaussian distribution over a lattice $\mathcal{L}$ as follows:

Definition 2.3.7 (Discrete Gaussian Distribution)**.** *For a lattice coset $\boldsymbol{c} + \mathcal{L} \in \mathbb{R}^n$ and a real width $r > 0$, the discrete Gaussian distribution $\mathcal{D}_{\boldsymbol{c}+\mathcal{L},r}$ is defined as the Gaussian distribution restricted to the coset $\boldsymbol{c} + \mathcal{L}$:*

$$\mathcal{D}_{\boldsymbol{c}+\mathcal{L},r}(\boldsymbol{x}) \propto \begin{cases} \rho_r(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \boldsymbol{c} + \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

Lastly, we define the smoothing parameter of a lattice $\mathcal{L}$, which is a measure of the smallest amount of gaussian noise that can be added to a lattice point such that it smooths out the lattice structure. It was introduced by Micciancio and Regev [MR04] and it is used in several proofs with the notion that when the width of the Discrete Gaussian distribution is larger than the smoothing parameter ($r > \eta_\epsilon(\mathcal{L})$), then the Discrete Gaussian distribution behaves like a (continuous) Gaussian distribution $\mathcal{D}_r$. The formal definition is as follows:

Definition 2.3.8 (Smoothing Parameter)**.** *The smoothing parameter $\eta_\epsilon(\mathcal{L})$ of a lattice $\mathcal{L}$, for a tolerance $\epsilon > 0$, is defined as the smallest radius r such that*

$$\rho_{1/r}(\mathcal{L}^*) \leqslant 1 + \epsilon.$$

There are two theorems that relate the smoothing parameter minimum distance and successive minima of a lattice, which are important in the context of lattice-based cryptography.

Theorem 2.3.9 ([Ban93], [MR04]). *For any lattice $\mathcal{L}$ and tolerance $\epsilon > 0$, the smoothing parameter satisfies:*

$$\eta_{2^{-n}}(\mathcal{L}) \leqslant \frac{\sqrt{n}}{\lambda_1(\mathcal{L}^*)}.$$

Theorem 2.3.10 ([MR04]). *For any lattice $\mathcal{L}$ and tolerance $\epsilon \in (0, 1/2)$, the smoothing parameter satisfies:*

$$\eta_\epsilon(\mathcal{L}) \leqslant \lambda_n(\mathcal{L}) \cdot \sqrt{\log O(n/\epsilon)}.$$

2.3.5 Lattice Problems

In this section we define the most common lattice problems, which are the basis for the security of many lattice-based cryptographic schemes, such as the ones presented in this paper. Most of the definitions described here are adapted from [Pei15].

Definition 2.3.11 (Shortest Vector Problem (SVP)). *Given an arbitrary basis $\boldsymbol{B}$ of a lattice $\mathcal{L}$, find a shortest non-zero vector $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v}\| = \lambda_1(\mathcal{L})$.*

In lattice cryptography, we are usually interested in the approximate version of these lattice problems, so we begin by defining the approximate Shortest Vector Problem (SVP_γ) and the Gap SVP (GapSVP_γ), which are the approximate versions of the Shortest Vector Problem (SVP) and the decisional aproximate version of the SVP, respectively. Normally, cryptosystems are designed to be secure assuming the hardness of the decision version of the problem, such as the GapSVP_γ, in the worst case.

Definition 2.3.12 (Approximate SVP (SVP_γ)). *Given a lattice basis $\boldsymbol{B}$ of an n-dimensional lattice $\mathcal{L}$, and a parameter $\gamma = \gamma(n) > 1$, find a non-zero vector $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ such that*

$$\|\boldsymbol{v}\| \leqslant \gamma \cdot \lambda_1(\mathcal{L}(\boldsymbol{B})).$$

Definition 2.3.13 (Decisional Approximate SVP (GapSVP_γ)). *Given a lattice basis $\boldsymbol{B}$ of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$ and a parameter $\gamma = \gamma(n) > 1$, decide whether the shortest vector $\lambda_1(\mathcal{L})$ satisfies $\lambda_1(\mathcal{L}) \leqslant 1$ or $\lambda_1(\mathcal{L}) \geqslant \gamma(n)$.*

In certain cases, such as the Ajtai-Dwork cryptosystem [Ajt96], the worst-case to average-case reductions are made with respect to the unique shortest vector problem, defined as follows:

Definition 2.3.14 (Unique Shortest Vector Problem (uSVP_γ)). *The let $\mathcal{L} \subset \mathbb{R}^n$ be any lattice having a "γ-unique" shortest vector, that is, the length of the shortest vector $\boldsymbol{v} \in \mathcal{L}$ is at least a γ factor smaller than the lengths of all lattice vectors not parallel to $\boldsymbol{v}$:*

$$\lambda_2(\mathcal{L}) \geqslant \gamma \cdot \lambda_1(\mathcal{L}).$$

Find a shortest vector $\boldsymbol{v} \in \mathcal{L}$, such that $\|\boldsymbol{v}\| = \lambda_1(\mathcal{L})$.

Remark. *The* uSVP_γ *is a promise problem, because the input must satisfy that condition, otherwise any answer is correct. And, it also is a worst-case problem, because there is no distribution over the lattice or its basis. An algorithm that solves* uSVP_γ *must work on* ***any*** $\mathcal{L}$ *satisfying the promise.*

Next, we define other lattice problems that are also used in lattice-based cryptography, such as the Shortest Independent Vectors Problem (SIVP), the Closest Vector Problem (CVP), and the Bounded Distance Decoding (BDD) problem. Here we define their approximate versions, which are denoted by the γ parameter, and we will denote them as SIVP_γ, CVP_γ, and BDD_γ, respectively.

Definition 2.3.15 (Approximate Shortest Independent Vectors Problem (SIVP_γ))**.** *Given a basis* $\boldsymbol{B}$ *of a full-rank, n-dimensional lattice* $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$*, output a set of n linearly independent vectors* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathcal{L}(\boldsymbol{B})$ *such that*

$$\|\boldsymbol{v}_i\| \leqslant \gamma \cdot \lambda_n(\mathcal{L}) \qquad , \forall i = 1, \ldots, n.$$

Definition 2.3.16 (Closest Vector Problem (CVP_γ))**.** *Given a basis* $\boldsymbol{B}$ *of an n-dimensional lattice* $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$ *and a target vector* $\boldsymbol{t} \in \mathbb{R}^n$*, find a vector* $\boldsymbol{v} \in \mathcal{L}$ *such that*

$$dist(\boldsymbol{v}, \boldsymbol{t}) \leqslant \gamma \cdot \min_{\boldsymbol{w} \in \mathcal{L}} dist(\boldsymbol{w}, \boldsymbol{t}),$$

where $dist(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$ *is the distance between two vectors* $\boldsymbol{x}$ *and* $\boldsymbol{y}$*.*

Definition 2.3.17 (Bounded Distance Decoding (BDD_γ))**.** *Given a basis* $\boldsymbol{B}$ *of an n-dimensional lattice* $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$ *and a target vector* $\boldsymbol{t} \in \mathbb{R}^n$*, with the guarantee that* $dist(\boldsymbol{t}, \mathcal{L}) \leqslant d = \lambda_1(\mathcal{L})/(2\gamma)$*, find the* ***unique*** *lattice vector* $\boldsymbol{v} \in \mathcal{L}$ *such that* $dist(\boldsymbol{v}, \boldsymbol{t}) \leqslant d$*.*

Remark. *The Bounded Distance Decoding problem is very similar to the CVP, however it is a promise problem, guaranteed that the target vector* $\boldsymbol{t}$ *is close enough to a lattice point, within a given distance d, and also that there is a unique lattice point within that distance.*

NP-hardness of the Lattice Problems

SVP: The Shortest Vector Problem (SVP) is NP-hard, and it is one of the most studied problems in lattice theory. The best non-randomized result is by [vEB81], where the they indicate that for the ℓ_∞ norm, the problem is NP-hard. For the ℓ_2 norm, the reduction is randomized.

Theorem 2.3.18 ([Kho04])**.** *For every* $\varepsilon > 0$*, there is a constant* $p(\varepsilon)$ *such that for all integers* $p \geqslant p(\varepsilon)$*, there is no polynomial time approximation to* SVP_p[1] *within ratio* $p^{1-\varepsilon}$ *provided* $NP \nsubseteq ZPP$*.*

[1]The length of the vectors can be measured in any ℓ_p norm ($p \geqslant 1$) and the corresponding problem is denoted by SVP_p

2.3.6 Lattice Reductions

In several of the definitions and algorithms in this paper, we will use the notion of lattice reductions, which is a process that transforms a basis of a lattice into a more convenient one, usually with shorter vectors, while preserving the lattice structure. Here we define the most common types of lattice reductions which are referenced in this work.

Definition 2.3.19 (Gram-Schmidt Orthogonalization (GSO))**.** *The Gram-Schmidt orthogonalization (GSO) is a process that takes a basis of a lattice and produces an orthogonal basis. Let* $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ *be a basis of a lattice* $\mathcal{L} \in \mathbb{R}^{n \times m}$*.*

The Gram-Schmidt orthogonalization (GSO) of $\boldsymbol{B}$ *is* $\boldsymbol{b}^* = (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*)$*, where* $\boldsymbol{b}_i^*$ *is* $\pi_i(\boldsymbol{b}_i)$*, the projection of* $\boldsymbol{b}_i$ *onto the orthogonal complement of the span of* $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1}$*, defined as:*

$$\begin{aligned} \boldsymbol{b}_1^* &= \boldsymbol{b}_1, \\ \boldsymbol{b}_i^* &= \boldsymbol{b}_i - \sum_{j=1}^{i-1} \mu_{j,i} \boldsymbol{b}_j^*, \quad i = 2, \ldots, n, \end{aligned}$$

where $\mu_{j,i} = \frac{\langle \boldsymbol{b}_i, \boldsymbol{b}_j^* \rangle}{\langle \boldsymbol{b}_j^*, \boldsymbol{b}_j^* \rangle}$*.*

Now consider the following definitions of lattice bases, which are used to define the different types of lattice reductions. Let $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ be a basis of a lattice $\mathcal{L}$:

1. **Size-reduced**: $\boldsymbol{B}$ is *size-reduced* if $|\mu_{j,i}| \leqslant 1/2$ for $1 \leqslant i < j \leqslant n$.
2. ξ**-LLL-reduced**: $\boldsymbol{B}$ is ξ*-LLL-reduced* it is **size-reduced** and $\xi \|\boldsymbol{b}_i^*\|^2 \leqslant \|\mu_{i+1,i}\boldsymbol{b}_i^* + \boldsymbol{b}_{i+1}^*\|^2$ for $1 \leqslant i < n$, $\xi \in (\frac{1}{4}, 1]$. Then, we have that $\|\boldsymbol{b}_1\| \leqslant 2^{(n-1)/4} \cdot \mathrm{vol}(\mathcal{L})^{1/n}$. The LLL algorithm [LLL82] considers $\xi = \frac{3}{4}$.
3. **SVP-reduced**: $\boldsymbol{B}$ is *SVP-reduced* if $\|\boldsymbol{b}_1\| = \lambda_1(\mathcal{L})$.
4. δ**-SVP-reduced**: $\boldsymbol{B}$ is δ*-SVP-reduced* if $\|\boldsymbol{b}_1\|^2 \leqslant \delta\lambda_1(\mathcal{L})^2$.
5. **HKZ-reduced**: $\boldsymbol{B}$ is *HKZ-reduced* if it is **size-reduced** and $\boldsymbol{B}_{[i,n]}$ is **SVP-reduced** for $i = 1, \ldots, n$.
6. β**-BKZ-reduced**: $\boldsymbol{B}$ is β*-BKZ-reduced* if it is **size-reduced** and $\boldsymbol{B}_{[i,\min\{\, i+\beta-1, n\,\}]}$ is **SVP-reduced** for $i = 1, \ldots, n$.
7. (δ, β)**-BKZ-reduced**: $\boldsymbol{B}$ is (δ, β)*-BKZ-reduced* if it is **size-reduced** and the block $\boldsymbol{B}_{[i,\min\{\, i+\beta-1, n\,\}]}$ is δ**-SVP-reduced** for $i = 1, \ldots, n$.
8. δ**-SVP-oracle**: Given a lattice $\mathcal{L}$ with a basis $\boldsymbol{B}$, outputs a primitive vector α such that $\|\alpha\boldsymbol{B}\|^2 \leqslant \delta\lambda_1(\mathcal{L})^2$, and therefore, $\|\alpha\boldsymbol{B}\| \leqslant \sqrt{\delta\gamma_\beta} \cdot \mathrm{vol}(\mathcal{L})^{1/\beta}$ (rank β). LLL algorithm efficiently achieves $\|\alpha\boldsymbol{B}\| \leqslant 2^{(\beta-1)/4} \cdot \mathrm{vol}(\mathcal{L})^{1/\beta}$.

Babai's Rounding Off Algorithm

Considering the hard lattice problems and basis-reduction techniques defined above, the first algorithm that gives us a concrete approximation to the CVP_γ is Babai's Nearest Plane Algorithm (or also the simpler Rounding-Off Algorithm, [Bab86]). This is a simple algorithm that takes a basis $\boldsymbol{B}$ of a lattice $\mathcal{L}$, or more precisely, a ξ-LLL-reduced basis, and a target vector $\boldsymbol{t} \in \mathbb{R}^n$, and outputs a vector $\boldsymbol{v} \in \mathcal{L}$ that is the closest to the target vector $\boldsymbol{t}$, within a factor of $2^{n/2}$ (or 2^n in the simpler approach).

We begin by showing the simpler, weaker approximation, which is easier to understand, and already provides a good enough approximation to the CVP_γ problem in certain cases. This algorithm rounds the coordinates of the target vector $\boldsymbol{t}$ in the basis $\boldsymbol{B}$.

Algorithm 1 Babai's Rounding Off Algorithm, [Bab86]

Input: A basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of a lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$
Input: A target vector $\boldsymbol{t} \in \mathbb{R}^n$
Output: A vector $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v} - \boldsymbol{t}\| \leqslant 2^n \cdot \mathrm{dist}(\boldsymbol{t}, \mathcal{L})$

$\boldsymbol{\beta} \leftarrow \boldsymbol{B}^{-1} \cdot \boldsymbol{t}$ ▷ Compute the coordinates of $\boldsymbol{t}$ in the basis $\boldsymbol{B}$
for $i = 1$ to n **do**
    $\alpha_i \leftarrow \lfloor \beta_i \rceil$ ▷ Round the coordinates to the nearest integer
$\boldsymbol{v} \leftarrow \sum_{i=1}^{n} \alpha_i \boldsymbol{b}_i$ ▷ Reconstruct the vector in the lattice
return $\boldsymbol{v}$

Remark. *Note that, given an ξ-LLL-reduced basis $\boldsymbol{B}$, the above algorithm Babai's Rounding Off Algorithm, [Bab86] returns a vector $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v} - \boldsymbol{t}\| \leqslant 2^n \cdot \lambda_1(\mathcal{L})$.*

Since $\boldsymbol{B}$ is ξ-LLL-reduced, we have that $\|\boldsymbol{b}_i\| \leqslant 2^{n/2} \cdot \lambda_1(\mathcal{L})$, and therefore, the distance between the target vector $\boldsymbol{t}$ and the reconstructed vector $\boldsymbol{v}$ is bounded by:

$$\|\boldsymbol{v} - \boldsymbol{t}\| = \left\| \sum_{i=1}^{n} (\alpha_i - \beta_i) \boldsymbol{b}_i \right\| \leqslant \sum_{i=1}^{n} |\alpha_i - \beta_i| \cdot \|\boldsymbol{b}_i\| \leqslant n/2 \cdot 2^{n/2} \cdot \lambda_1(\mathcal{L}) \leqslant 2^n \cdot \lambda_1(\mathcal{L}),$$

where we used the fact that $|\alpha_i - \beta_i| \leqslant 1/2$ for all $i = 1, \ldots, n$.

The second algorithm is Babai's Nearest Plane Algorithm, which is a more powerful algorithm that iteratively moves the target vector $\boldsymbol{t}$ onto the nearest plane defined by the basis vectors of the lattice $\mathcal{L}$ using the Gram-Schmidt orthogonalization. The algorithm is shown in Algorithm 2, following the presentation from Regev's Notes [Reg04b].

Algorithm 2 Babai's Nearest Plane Algorithm, [Bab86]

Input: A basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of a lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$
Input: A target vector $\boldsymbol{t} \in \mathbb{R}^n$
Output: A vector $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v} - \boldsymbol{t}\| \leqslant 2^{n/2} \cdot \mathrm{dist}(\boldsymbol{t}, \mathcal{L})$

$\boldsymbol{w} \leftarrow \boldsymbol{t}$
for $i = n$ to 1 **do**
  $\boldsymbol{w} \leftarrow \boldsymbol{w} - \left\lfloor \frac{\langle \boldsymbol{w}, \boldsymbol{b}_i^* \rangle}{\|\boldsymbol{b}_i^*\|^2} \right\rceil \boldsymbol{b}_i$ ▷ centers its $\boldsymbol{b}_i^*$-coordinate in $(-\frac{1}{2}, \frac{1}{2}]$
$\boldsymbol{v} \leftarrow \boldsymbol{t} - \boldsymbol{w}$
  ▷ Remove the residual $\boldsymbol{w}$ to obtain the lattice point closest to $\boldsymbol{t}$ found by the loop
return $\boldsymbol{v}$

Remark (Quality of Babai's nearest plane output)**.** *After the loop, the residual* $\boldsymbol{w}$ *can be written in the (orthogonal) Gram–Schmidt basis as* $\boldsymbol{w} = \sum_{i=1}^{n} \theta_i \boldsymbol{b}_i^*$ *with each* $\theta_i \in (-\frac{1}{2}, \frac{1}{2}]$*; hence* $\|\boldsymbol{v} - \boldsymbol{t}\| = \|\boldsymbol{w}\| \leq \frac{1}{2}\sqrt{\sum_i \|\boldsymbol{b}_i^*\|^2}$*. When* $\boldsymbol{B}$ *is* ξ*-LLL-reduced (e.g.,* $\delta = 3/4$*), standard LLL inequalities imply* $\sqrt{\sum_i \|\boldsymbol{b}_i^*\|^2} \leq 2^{n/2} \cdot \mathrm{dist}(\boldsymbol{t}, \mathcal{L})$*, yielding the stated* $2^{n/2}$*-approximation for CVP.*

Moreover, Babai is exact *whenever the target lies in a ball around a lattice point: if* $\mathrm{dist}(\boldsymbol{t}, \mathcal{L}) \leq \frac{1}{2} \min_i \|\boldsymbol{b}_i^*\|$*, then* $\boldsymbol{v}$ *is the true closest lattice vector.*

2.4 Cryptographic Notions

Finally, we define some of the cryptographic notions that are used throughout the paper allowing us to define and prove the security of the schemes presented in this work. Most of the definitions are adapted from [Pei15].

2.4.1 Security Parameter and Adversary Model

The security parameter n is a positive integer that is used to define the security of a cryptographic scheme. It is usually related to the size of the keys, the size of the messages, and regulates the running time of all algorithms involved in the scheme, including the adversaries. As denoted in Sec. 2.2, normally the algorithms have a running time that is polynomial in the security parameter (PPT), and the adversaries (attacker, usually denoted as $\mathcal{A}$), are required to have a negligible advantage (Def 2.2.1), in the security parameter, in breaking the scheme.

When we talk about the advantage of an adversary in breaking a scheme, we are usually referring to the advantage in distinguishing between the **real** distribution of the scheme and a **random** distribution, which is usually defined as the *ideal* distribution.

2.4.2 Public-Key Encryption (PKE)

A public-key encryption scheme is comprised of three algorithms that allow a user to encrypt and decrypt messages using a public key and a secret key. We denote a public-key encryption scheme as $\mathtt{PKE} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$, over a message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, where:

- **Key Generation** ($\mathtt{KeyGen}()$): Given a security parameter, usually not shown as an input, outputs a key pair, a public key pk and a secret key sk.
- **Encryption** ($\mathtt{Enc}(\mathsf{pk}, m; r)$): Given a public key pk, a message $m \in \mathcal{M}$, and randomness r[2], outputs a ciphertext $c \in \mathcal{C}$.
- **Decryption** ($\mathtt{Dec}(\mathsf{sk}, c)$): Given a secret key sk and a ciphertext $c \in \mathcal{C}$, outputs a message $m \in \mathcal{M}$ or the failure symbol $\perp$.

Correctness. We say that a public-key encryption scheme is *correct* if, generating a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$, and encrypting a message $m \in \mathcal{M}$ with the public key pk, the decryption algorithm returns the same message m with all but negligible probability.

That is, we can say that for every $m \in \mathcal{M}$,

$$\Pr\left[\mathtt{Dec}(\mathsf{sk}, \mathtt{Enc}(\mathsf{pk}, m)) = m \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()\right] = 1 - \mathsf{negl}(n),$$

where $\mathsf{negl}(n)$ is a negligible function in the security parameter n.

2.4.3 IND-CPA and IND-CCA Security for Public-Key Encryption

For a public-key encryption scheme $\mathtt{PKE} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$, we also define the security notions of *indistinguishability under chosen-plaintext attack* (IND-CPA) and *indistinguishability under chosen-ciphertext attack* (IND-CCA).

IND-CPA is the security notion that guarantees that an adversary cannot distinguish between two encryptions of two different messages, even if it can query the encryption oracle with other messages. The formal definition considers an experiment that runs the adversary $\mathcal{A}$ and outputs a bit b' that indicates whether the adversary guessed correctly which of the two messages was encrypted. The experiment is defined as follows:

Definition 2.4.1 (IND-CPA Security)**.** *Consider the following experiment between a challenger and an adversary $\mathcal{A}$:*

1. *The challenger generates a key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$.

[2]The randomness r may be implicitly included, and not shown as an input in the encryption algorithm.

2. *$\mathcal{A}$ receives the public key* pk *and may obtain encryptions under* pk *(i.e., for every message $m \in \mathcal{M}$, $\mathcal{A}$ can use the public key* pk *to generate a ciphertext $c = \mathit{Enc}(\mathsf{pk}, m)$).*

3. *$\mathcal{A}$ outputs two equal-length messages $m_0, m_1 \in \mathcal{M}$.*

4. *The challenger samples a bit $b \leftarrow\!\!\$\ \{0,1\}$ and encrypts the corresponding message $\boldsymbol{c}^* = \mathit{Enc}(\mathsf{pk}, m_b)$, and gives $\boldsymbol{c}^*$ to $\mathcal{A}$.*

5. *$\mathcal{A}$ continues its oracle queries and outputs a bit b'.*[3]

6. *The experiment outputs 1 if $b' = b$, and 0 otherwise.*

The encryption scheme PKE *is said to be IND-CPA secure if it is infeasible for any PPT adversary $\mathcal{A}$ to distinguish between the two messages, or, more formally, that the advantage of $\mathcal{A}$ in the above experiment is negligible:*

$$\mathsf{Adv}^{\mathit{ind\text{-}cpa}}_{\mathit{PKE}}(\mathcal{A}) = \left| \Pr\left[b' = b\right] - \frac{1}{2} \right| \leqslant \mathsf{negl}(n),$$

where $\mathsf{negl}(n)$ is a negligible function in the security parameter n.

IND-CCA is a stronger security notion than IND-CPA, which guarantees that the adversary cannot distinguish between two encryptions of two different messages, even if it can query the *decryption oracle* with ciphertexts that are not the challenge ciphertext. The formal definition is similar to the IND-CPA, but with access to a decryption oracle that does not include the challenge ciphertext. The experiment is defined as follows:

Definition 2.4.2 (IND-CCA Security)**.** *Consider the following experiment between a challenger and an adversary $\mathcal{A}$:*

1. *The challenger generates a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathit{KeyGen}()$.*

2. *$\mathcal{A}$ receives the public key* pk *and may adaptively obtain encryptions under* pk *(i.e., for every message $m \in \mathcal{M}$, $\mathcal{A}$ can query the encryption oracle and receive the ciphertext $c = \mathit{Enc}(\mathsf{pk}, m)$).*

3. *$\mathcal{A}$ may also query the decryption oracle with ciphertexts $\boldsymbol{c}' \in \mathcal{C}$, obtaining $\mathit{Dec}(\mathsf{sk}, \boldsymbol{c}')$.*

4. *$\mathcal{A}$ outputs two equal-length messages $m_0, m_1 \in \mathcal{M}$.*

5. *The challenger samples a bit $b \leftarrow\!\!\$\ \{0,1\}$ and encrypts the corresponding message $\boldsymbol{c}^* = \mathit{Enc}(\mathsf{pk}, m_b)$, and gives $\boldsymbol{c}^*$ to $\mathcal{A}$.*

6. *$\mathcal{A}$ continues its oracle queries, including decryption queries, but* ***excluding the challenge ciphertext $\boldsymbol{c}^*$****, and outputs a bit b'.*

[3] In certain cases, we denote that the adversary $\mathcal{A}$, instead of outputting a bit b', either "accepts" or "rejects" the challenge, and that is equivalent to outputting 1 or 0, respectively.

7. *The experiment outputs* 1 *if* $b' = b$*, and* 0 *otherwise.*

The encryption scheme PKE *is said to be IND-CCA secure if it is infeasible for any PPT adversary* $\mathcal{A}$ *to distinguish between the two messages, or, more formally, that the advantage of* $\mathcal{A}$ *in the above experiment is negligible:*

$$\mathsf{Adv}^{\mathit{ind\text{-}cca}}_{\mathit{PKE}}(\mathcal{A}) = \left| \Pr\left[b' = b\right] - \frac{1}{2} \right| \leqslant \mathsf{negl}(n),$$

where $\mathsf{negl}(n)$ *is a negligible function in the security parameter* n*.*

2.4.4 Key Encapsulation Mechanism (KEM)

As an alternative to public-key encryption, we can define a key-encapsulation mechanism (KEM), which is a cryptographic primitive that allows two parties to establish a shared secret key using a public key and a secret key. It is similar to a public-key encryption scheme, but instead of encrypting messages, it encapsulates a random key. This key can then be used to encrypt messages using a symmetric encryption scheme. A KEM is defined as a tuple of three algorithms, denoted as $\mathtt{KEM} = (\mathtt{KeyGen}, \mathtt{Encaps}, \mathtt{Decaps})$, where:

- **Key Generation** ($\mathtt{KeyGen}()$): Given a security parameter, outputs a key pair, a public key pk and a secret key sk.
- **Encapsulation** ($\mathtt{Encaps}(\mathsf{pk})$): Given a public key pk, outputs a pair $(K, \boldsymbol{c})$, where K is the shared secret key and $\boldsymbol{c}$ is the encapsulation, or ciphertext. It is a probabilistic algorithm.
- **Decapsulation** ($\mathtt{Decaps}(\mathsf{sk}, \boldsymbol{c})$): Given a secret key sk and an encapsulation $\boldsymbol{c}$, outputs either the shared secret key K or the failure symbol $\perp$.

Correctness. We say that a key encapsulation mechanism is *correct* if, generating a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$, and encapsulating a key, obtaining the encapsulation $(K, \boldsymbol{c})$ with the public key pk, the decapsulation algorithm returns the same key K with all but negligible probability.

$$\forall (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}(), \quad \forall (K, \boldsymbol{c}) \leftarrow\!\!\$\ \mathtt{Encaps}(\mathsf{pk}) : \quad \Pr\left[\mathtt{Decaps}(\mathsf{sk}, \boldsymbol{c}) = K\right] = 1 - \mathsf{negl}(n),$$

where $\mathsf{negl}(n)$ is a negligible function in the security parameter n.

2.4.5 IND-CCA Security for KEMs

Normally, the security notion defined for a key encapsulation mechanism (KEM) is the IND-CCA security, which is similar to the IND-CCA security for public-key encryption, allowing the adversary access to a decryption oracle, or *decapsulation oracle*, but with a different challenge. In this case, the adversary is trying to distinguish between a real key

encapsulation and a random key of equal length, rather than distinguishing between two messages. That is, the adversary is challenged to guess whether a given ciphertext-key pair $(K, \boldsymbol{c}^*)$ is a real encapsulation of a key K or it is a random key. The formal definition of the experiment is as follows:

Definition 2.4.3 (IND-CCA Security for KEMs)**.** *Consider the following experiment between a challenger and an adversary $\mathcal{A}$:*

1. *The challenger generates a key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathtt{KeyGen}()$.
2. *$\mathcal{A}$ receives the public key* pk *and may obtain encapsulations under* pk *(i.e., $\mathcal{A}$ can query the encapsulation oracle and receive* $(K, \boldsymbol{c}) \leftarrow\!\!\$\ \mathtt{Encaps}(\mathsf{pk})$*).*
3. *$\mathcal{A}$ may also query the decapsulation oracle with encapsulations $\boldsymbol{c}$, obtaining the decapsulation from* $\mathtt{Decaps}(\mathsf{sk}, \boldsymbol{c})$.
4. *The challenger samples a bit* $b \leftarrow\!\!\$\ \{0, 1\}$ *and samples a key-ciphertext pair* $(K^*, \boldsymbol{c}^*) \leftarrow\!\!\$\ \mathtt{Encaps}(\mathsf{pk})$*. If* $b = 1$*, then the challenger outputs the real key-ciphertext pair* $(K^*, \boldsymbol{c}^*)$*, otherwise it outputs a random key K' of the same length as K with the ciphertext* $(K', \boldsymbol{c}^*)$.
5. *$\mathcal{A}$ continues its oracle queries, including decapsulation queries, but* ***excluding the challenge encapsulation $\boldsymbol{c}^*$****, and outputs a bit b'.*
6. *The experiment outputs* 1 *if* $b' = b$*, and* 0 *otherwise.*

The key encapsulation mechanism $\mathtt{KEM}$ *is said to be IND-CCA secure if it is infeasible for any PPT adversary $\mathcal{A}$ to distinguish between the two cases, or, more formally, that the advantage of $\mathcal{A}$ in the above experiment is negligible:*

$$\mathsf{Adv}^{\mathit{ind\text{-}cca}}_{\mathtt{KEM}}(\mathcal{A}) = \left| \Pr\left[b' = b\right] - \frac{1}{2} \right| \leqslant \mathsf{negl}(n),$$

where $\mathsf{negl}(n)$ *is a negligible function in the security parameter n.*

Chapter 3

Early Public Key Cryptosystems

3.1 Ajtai-Dwork Cryptosystem

In their work, Ajtai and Dwork [AD97] created the first public-key encryption scheme with security based on the hardness of a worst-case problem, in this case, the uSVP_γ (Def. 2.3.14). It was a significant breakthrough in cryptography, and more specifically, in lattice-based cryptography, particularly when discussing the security and evolution towards post-quantum cryptography schemes like the LWE problem and its variants (Secs. 4.1, 4.2, 4.3), and culminating in the ML-KEM (Kyber, 5) standard. To achieve this, they rely on a worst-case to average-case reduction for lattice problems, initially proposed by Ajtai in [Ajt96], ensuring that every instance of the cryptosystem reduces to a problem conjectured to be extremely hard, or, in other words, that breaking a random instance of the cryptosystem is at least as hard as solving a worst-case instance of an underlying lattice problem.

As an overview, their approach first shows that if an adversary $\mathcal{A}$ can *distinguish* between two distributions, which represent the encryption of 0 or 1 in a cryptosystem, then a reduction $\mathcal{B}$ can use $\mathcal{A}$ to solve a certain instance of the Hidden Hyperplanes Problem ((HHP), Def. 3.1.3), which is an average-case problem. This establishes that the security of the scheme relies on the hardness of solving the HHP. Then, they show that if $\mathcal{B}$ can solve the HHP with certain parameters, then, another reduction $\mathcal{C}$, using $\mathcal{B}$, can solve the uSVP_γ, for some $\gamma = \text{poly}(n)$, which is a worst-case problem ($\mathcal{C}$ is able to solve uSVP_γ for *any* lattice that satisfies the constraints). This sequence of reductions establishes a connection between the cryptosystem, the HHP, and the uSVP_γ. Therefore, since the uSVPγ problem is believed to be very hard to solve for appropriately chosen parameters, the cryptosystem's security is tightly bound to this hardness. As long as the conjectured difficulty of uSVPγ holds, the cryptosystem remains secure against adversarial attacks.

Even though the Ajtai-Dwork cryptosystem laid groundwork for lattice-based cryptography and was of great theoretical significance, it also had several issues that made it impractical for real-world applications. As one of the main issues, the scheme was completely impractical in terms of efficiency, with very large public key sizes ($O(n^4)$) and secret key and ciphertexts ($O(n^2)$), as well as running time. Note that this is even acknowledged

in the paper, where the authors state they "made no attempt to optimize the efficiency of the cryptosystem".

3.1.1 Background

The following definitions are from [AD97].

Definition 3.1.1 ((d, M)-Lattices)**.** *Assume n is a positive integer, $M > 0$, $d > 0$, $d, M \in \mathbb{R}$, and $\mathcal{L} = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ is a lattice which has an $n-1$-dimensional sublattice $\mathcal{L}'$ with the following properties:*

1. *$\mathcal{L}'$ has a basis of length at most M; (i.e. $\max_{i=1}^{n-1} \|\boldsymbol{b}_i\| \leqslant M$)*
2. *If H is the $(n-1)$-dimensional subspace of $\mathbb{R}^n$ containing $\mathcal{L}'$, and $H' \neq H$ is a coset of H intersecting $\mathcal{L}$, then the distance of H and H' is at least d.*

Then we say that $\mathcal{L}$ is a (d, M)-lattice. If $d > M$, then $\mathcal{L}'$ is unique. In this case $\mathcal{L}'$ will be denoted by $\mathcal{L}^{(d,M)}$. The minimum distance between H and a coset of H intersecting $\mathcal{L}$ will be denoted by $d_{\mathcal{L}}$.

Definition 3.1.2 (Factor Lattice)**.** *Suppose that $\mathcal{L}$ is a lattice in $\mathbb{R}^n$ and H is a k-dimensional subspace of $\mathbb{R}^n$ so that $\mathcal{L}' = H \cap \mathcal{L}$ is a k-dimensional lattice in H. The factor lattice $\mathcal{L}/\mathcal{L}'$ will be the lattice we get from projecting $\mathcal{L}$ orthogonally onto $H^{\perp}$.*

Definition 3.1.3 (Hidden Hyperplanes Problem (HHP))**.** *Let $c > 5$ be a real number, and let $\mathcal{D}_{\mathcal{L}}$ be a distribution on the set of (d, M) lattices for which $d > n^c M$ and $d \leqslant d_{\mathcal{L}} \leqslant 2d$. The hidden hyperplane assumption for $\mathcal{D}_{\mathcal{L}}$ says that, given a basis for a random (d, M) lattice $\mathcal{L} \in \mathcal{D}_{\mathcal{L}}$, it is computationally infeasible to compute $\mathcal{L}^{(d,M)}$.*

Remark. *The hidden hyperplane assumption is related to the uSVP (Def. 2.3.14) as follows. If Λ is a lattice with an n^c-unique shortest vector $\boldsymbol{u}$, then $\mathcal{L} = \Lambda^*$ is a (d, M) lattice for some $d > n^{c'} M$ and $d_{\mathcal{L}} = \|\boldsymbol{u}\|^{-1}$, with $c' \approx c - 2$. (See Lemma 3.1.13 for proof.)*

3.1.2 Public-key Encryption Scheme

In their paper, the constants used are: $\mathcal{D}_1 = 3, \mathcal{D}_2 = 8, \mathcal{D}_3 = 3, R = n^{-\mathcal{D}_1} = n^{-3}$ and $\mathcal{K} = 2^{n \log n}$

Private Key

The private key is a random vector $\boldsymbol{u} \in B_n(1)$, chosen uniformly from the unit ball.

Public Key

Given the private key $\boldsymbol{u}$, we can define a distribution $\mathcal{H}_{\boldsymbol{u},R,m}$ of points in an n-dimensional cube $\mathcal{Q}$ of sides $\mathcal{K} = 2^{n \log n}$ with the following construction:

- Sample a point $\boldsymbol{y} \in \{\, \boldsymbol{x} \in \mathcal{Q} \mid \langle \boldsymbol{x}\,,\, \boldsymbol{u} \rangle \in \mathbb{Z} \,\}$. This construction defines a point from one of the parallel $(n-1)$-dimensional hyperplanes which $\boldsymbol{u}$ is orthogonal to.
- Sample a small perturbation $\boldsymbol{z}$. The perturbation is defined as the sum of m vectors sampled independently and with a uniform distribution from $B_n(R)$.[1]
- Output $\boldsymbol{y} + \boldsymbol{z}$.

Then, the public key is a set of vectors $\{\, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \,\}$ and $\{\, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_n \,\}$ sampled independently at random from the distribution $\mathcal{H}_{\boldsymbol{u},R,n}$ described above, and with $\mathcal{P} = \mathcal{P}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$ having a width of at least $n^{-2}\mathcal{K}$ (The width is defined as the *minimum* distance between the each $\boldsymbol{w}_i$ and the subspace generated by $\{\, \boldsymbol{w}_j \mid j \neq i \,\}$).

In other words, the public key is defined by m vectors close to the parallel hyperplanes orthogonal to $\boldsymbol{u}$ and a parallelepiped that is not very "thin".

Encryption

The cryptosystem only allows the encryption of either 0 or 1. This is done differently for each case:

- Encryption of 0: Uniformly select $\alpha_1, \ldots, \alpha_m \in \{\, 0, 1 \,\}$ and output the ciphertext as $\boldsymbol{c} = \sum_{i=1}^{m} \alpha_i \boldsymbol{v}_i \bmod \mathcal{P}$. That is, the sum of a random subset of points given in the public key modulo the parallelepiped defined by $\mathcal{P}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$. Let the random variable $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ be a random encryption of 0.
- Encryption of 1: Choose a random point inside the $\mathcal{P}$. Let the random variable $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ be a random encryption of 1.

Decryption

For the decryption, given the ciphertext $\boldsymbol{c}$, and using the private key $\boldsymbol{u}$, verify whether $\langle \boldsymbol{u}\,,\, \boldsymbol{c} \rangle \approx 0 \bmod 1$, or, in other words, if the result is close to an integer, and if so, decrypt $\boldsymbol{c}$ as 0. More precisely, if the result is within n^{-1} of 0 or 1. Otherwise, decrypt $\boldsymbol{c}$ as 1.[2]

3.1.3 Main Results

Proof Sketch

Ajtai and Dwork prove the security of their cryptosystem by using two main reductions:

1. If an adversary $\mathcal{A}$ can *distinguish* (w.p. at least n^{-c_1}) between two distributions, which represent the encryption of 0 or 1 in a cryptosystem, with a bias greater than n^{-c_2}, then a reduction $\mathcal{B}$ can use $\mathcal{A}$ to *distinguish* (w.p. at least n^{-c_3}) random variables *uniform* and $\mathcal{H}_{\boldsymbol{u},R,n}$ with a bias of a least $n^{-1/4}$ in time polynomial in n.

[1] For concreteness, Ajtai and Dwork use $R = n^{-3}$.

[2] Note that this may result in decryption errors.

2. If an adversary $\mathcal{B}$ can *distinguish* (w.p. at least n^{-c_3}) random variables *uniform* and $\mathcal{H}_{\boldsymbol{u},R,n}$ with a bias of a least n^{-c_4}, then a reduction $\mathcal{C}$ can use $\mathcal{B}$ to solve *any* instance of size at most n^{c_5} of the $n^{\mathcal{D}_2}$-uSVP in time polynomial in n and with probability greater than $1 - 2^n$.

The first reduction proves that breaking the security of the cryptosystem is at least as hard solving the Hidden Hyperplanes problem, while the second one shows that solving the latter is at least as hard solving the uSVPγ for $\gamma = \text{poly}(n)$.

Next follows an overview of the steps used to prove each of the reductions, with the proofs of each step in section 3.1.4.

Part 1 - Cryptosystem to HHP

Considering $\boldsymbol{u}$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ are picked at random as described in the protocol, suppose there exists an adversary $\mathcal{A}$ that can, with a probability of at least n^{-c_1}, distinguish two random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ with a bias of at least n^{-c_2}. Recall that $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ are the random encryptions of 0 and 1, respectively.

1. Amplify the bias that $\mathcal{A}$ can distinguish the random variables to at least $\frac{1}{2} - n^{-2}$:

   (a) Create a circuit $\mathcal{A}'$ that produces long independent sequences of the random variables, applies $\mathcal{A}$ for n^{c_2+1} (n times initial bias) and take majority of decisions. For the next steps, take the initial $\mathcal{A}$ to be the $\mathcal{A}'$, that is, an algorithm that wins distinguishes with bias greater than $\frac{1}{2} - n^{-2}$.

2. Define algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to decide whether a random variable ξ is *unif* or $\mathcal{H}_{\boldsymbol{u},R,n}$. Take mt values of ξ, $t = n^{2c_1}$, and create t blocks $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ of size m each. The general idea is that if $\xi = \mathcal{H}_{\boldsymbol{u},R,n}$, then B is a proper public key and for some sequences (i.e. $B \in Y$) $\mathcal{A}$ would be able to correctly identify the random variables. Otherwise, B would be random and the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ would be as well.

   (a) Define a set Y of sequences $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ that allows a sequence to be determined if its in the set in polynomial time, and, if the sequence is in the set, then $\mathcal{A}$ distinguishes the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ with bias greater than $\frac{1}{2} - n^{-1}$.[3]

   (b) Setup two counters: one counts the number of sequences of $B \in Y$ (sequences which $\mathcal{A}$ can distinguish the random variables) and the other counts the number of sequences which have a correct identification from $\mathcal{A}$ (this is only done for sequences $B \in Y$)[4].

[3] This is used in the next steps in order to separate the probability that $\mathcal{A}$ distinguishes the random variables.

[4] See step (d)

(c) For each B, if $B \in Y$, produce as many random variables of $\mathcal{S}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ and $\mathcal{E}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ as needed for the input of $\mathcal{A}$ and give them to $\mathcal{A}$. Increment the counter if $\mathcal{A}$ correctly identifies the samples.

(d) If the number of sequences $B \in Y$ is greater than $n^{-c_1}\frac{t}{2}$ and the number of correct identifications by $\mathcal{A}$ is greater than $\frac{3}{4}$ of the number of sequences $B \in Y$, then output that $\xi = \mathcal{H}_{\boldsymbol{u},R,n}$, otherwise, $\xi = \mathit{unif}$.

Therefore, with non-negligible probability, and considering $\boldsymbol{u}$ picked at random as described in the protocol, $\mathcal{B}$ can use $\mathcal{A}$ to distinguish unif and $\mathcal{H}_{\boldsymbol{u},R,n}$.

Part 2 - HHP to uSVP_γ

Considering $\boldsymbol{u}$ picked at random as described in the protocol, suppose there exists an algorithm $\mathcal{B}$ that can, with a probability of at least n^{-c_3}, distinguish two random variables $\mathcal{H}_{\boldsymbol{u},R,m}$ and unif with a bias of at least n^{-c_4}. We now describe an algorithm $\mathcal{C}$ that can, given as input any lattice with an $n^{\mathcal{D}_2}$-unique shortest vector $\boldsymbol{v}$, using the algorithm $\mathcal{B}$ as oracle, find this vector $\boldsymbol{v}$.

1. Given a lattice $\mathcal{L}$, by Lemma 3.1.10 we can find a $U_i = \theta\nu$, where $\theta \in \mathbb{R}$ and ν is an orthogonal linear transformation, such that $U_i\boldsymbol{v} \in X$, where $\boldsymbol{v}$ is shortest vector we are trying to find and X is the set of $\boldsymbol{u}$ which $\mathcal{B}$ can distinguish the random variables $\mathcal{H}_{\boldsymbol{u},R,m}$ and unif. Assume that i is now fixed with this property.

2. $\boldsymbol{v}$ is an $n^{\mathcal{D}_2}$-unique shortest vector of $\mathcal{L} \iff U_i\boldsymbol{v}$ is an $n^{\mathcal{D}_2}$-unique shortest vector of $U_i\mathcal{L}$ (because $U_i = \theta\nu$, $\theta \in \mathbb{R}$ and ν is an orthogonal linear transformation).

3. Considering the definition (d, M)-lattice (Def. 3.1.1). If Λ is a lattice with an n^c-unique shortest vector $\boldsymbol{u}$, then $\mathcal{L} = \Lambda^*$ is a (d, M)-lattice with $d \geqslant n^{c'}M$ and $d_{\mathcal{L}} = \|\boldsymbol{u}\|^{-1}$.

4. Let $J = (U_i\mathcal{L})^*$. Since $U_i\mathcal{L}$ has an $n^{\mathcal{D}_2}$-unique shortest vector, then J is a $(1, n^{-\mathcal{D}'_2})$ lattice, where $\mathcal{D}'_2 \approx \mathcal{D}_2 - 1$.

5. By (Lemma 3.1.17 - dist $\mathcal{H}$ and ξ) the distance of the distributions $\mathcal{H}_{\boldsymbol{u},R,m}$ and $\xi_{(J,K,R)}$ is exponentially small and $\eta_K = \mathit{unif}$, therefore if $\mathcal{B}$ can distinguish $\mathcal{H}_{\boldsymbol{u},R,m}$ and unif, it can also distinguish $\xi_{(J,K,R)}$ and η_K.

   (a) When we compute $\mathrm{pert}(R, m) := \boldsymbol{w} = \sum_{i=1}^{m} \boldsymbol{t}_i$, where $\boldsymbol{t}_i \leftarrow\!\!\$\ B_n(R)$, we write $\boldsymbol{t}_i = \boldsymbol{r}_i + \boldsymbol{s}_i$, where $\boldsymbol{r}_i$ is the component orthogonal to H. If we first randomize $\boldsymbol{r}_i$, then $\boldsymbol{s}_i$ is chosen from an $(n-1)$ dimensional ball of radius $R' = \sqrt{R^2 - \|\boldsymbol{r}_i\|^2}$.

      i. Let ξ_i be the distance from $\boldsymbol{t}_i$ to H. Let $\eta = \sum_{i=1}^{m} \xi_i$. Let I denote an interval contained in $[-\frac{mR}{40}, \frac{mR}{40}]$. If $\eta \in I$, then, w.h.p. there are at least $m/4$ values $\boldsymbol{s}_i$, such that they are chosen from $B_{n-1}(R')$, with $R' \geqslant n^2M$.

ii. Let $J^{(d,M)} = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$ and $\mathcal{P}' = \mathcal{P}^-(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$. For $n' \geqslant n$, assume each $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{n'}$ is chosen from $B_{n-1}(R')$, with $R' \geqslant n^2 M$. If we tile H with $\mathcal{P}'$, then, w.h.p., at least one of $\boldsymbol{s}_i$, $i = 1, \ldots, n'$, is chosen from a tile that is completely contained in its respective ball. In this case, $\boldsymbol{s}_i$ is uniformly distributed in $\mathcal{P}'$.

iii. Consider $\boldsymbol{w}$ *bad* if either less than $m/4$ $\boldsymbol{s}_i$ are chosen from balls of $R' \geqslant n^2 M$ (item (i) fails), or more than $m/4$ $\boldsymbol{s}_i$ are chosen from balls of $R' \geqslant n^2 M$ *but* none of these $\boldsymbol{s}_i$ are chosen from a tile completely contained in its respective ball (item (i) succeeds but item (ii) fails). The distance between projecting $\boldsymbol{w}$ onto H modulo $\mathcal{P}'$ and the uniform distribution on $\mathcal{P}'$ is given by the case where $\boldsymbol{w}$ is *bad*, since when $\boldsymbol{w}$ is *good*, $\boldsymbol{s}_i$ is uniform in $\mathcal{P}'$.

(b) Take a polynomial amount of points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m'} \in K'U^n$, with $K' = n^c d_J$. Let $\boldsymbol{a}_{ij} = \boldsymbol{p}_i - \boldsymbol{p}_j$, for $1 \leqslant i < j \leqslant m'$. There are at most n^c hyperplanes H', parallel to H, that intersect $K'U^n \cap J$. Therefore, a polynomial fraction of $\boldsymbol{a}_{ij}$ will be such that both $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are in the same hyperplane H', and thus $\boldsymbol{a}_{ij}$ is in H.

i. Let $\boldsymbol{v} \in \{\, \boldsymbol{a}_{ij} \mid 1 \leqslant i < j \leqslant m' \,\}$. Define the random variable $\delta_{\boldsymbol{v}} := \boldsymbol{u} + \alpha \boldsymbol{v} + \boldsymbol{w}$, where $\boldsymbol{u} \leftarrow\$\ KU^n \cap J, \alpha \leftarrow\$\ [0,1]$ and $\boldsymbol{w} := \mathrm{pert}(R, m)$. If $\boldsymbol{v} \in H$, then the projection of $\delta_{\boldsymbol{v}} (= \mathcal{H}_{\boldsymbol{u},R,m})$ onto H will be almost the same as the projection of $\xi_{J,K,R}$ onto H. If $\boldsymbol{v} \notin H$, then $\delta_{\boldsymbol{v}}$ will be essentially the same distribution as η_K.

6. If $\mathcal{B}$ can distinguish $\xi_{(J,K,R)}$ and η_K in Λ then an algorithm $\mathcal{B}'$, using $\mathcal{B}$, finds $J^{(d,M)}$ on Λ.

(a) The algorithm $\mathcal{B}'$ can, using $\mathcal{B}$, find a sequence of $n-1$ linearly independent vectors $\boldsymbol{v} \in H$, and thus, a basis of a lattice in H, or $J^{(d,M)}$.

7. With $J^{(d,M)}$, we can find the direction of $\boldsymbol{u}$ (orthogonal to $J^{(d,M)}$), and the distance $d_J = \|\boldsymbol{u}\|^{-1}$ in between hyperplanes (by taking the gcd of the distances of the hyperplane H that contains $J^{(d,M)}$ to points in the lattice J). Therefore yielding the vector $\boldsymbol{u} = U_i \boldsymbol{v}$, and thus $\boldsymbol{v}$, the shortest vector of $\mathcal{L}$.

3.1.4 Proofs

In this section, we will build upon the overview including the necessary proofs for each step. Most of the proofs are from [AD97], with some additional from [Ajt96].

Theorem 3.1.4 (Main Theorem)**.** *For all $c_1, c_2, c_3, c_4 > 0$, there exists a c_5 and a probabilistic algorithm $\mathcal{C}$ so that for all sufficiently large n, condition 1 implies condition 2, where*

1. *$\mathcal{A}$ is a probabilistic circuit of size n^{c_1} so that if we pick random vectors $\boldsymbol{u}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ as described in the protocol for generating the public and private keys, then, with a probability of at least n^{-c_2}, $\mathcal{A}$ distinguishes the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ over $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ with a bias of at least n^{-c_2}.*

2. *$\mathcal{C}$, using $\mathcal{A}$ as an oracle, can solve any instance of size at most n^{c_4} of the $n^{\mathcal{D}_2}$-unique shortest vector problem in time n^{c_5} with a probability at least $1 - 2^{-n}$.*

Part 1 - Cryptosystem to HHP

Lemma 3.1.5 (Cryptosystem to HHP)**.** *For all c_1, c_2, there exists c_2, c_3 and let an algorithm $\mathcal{A}$ which, if we pick random vectors $\boldsymbol{u}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ as described in the protocol for generating the public and private keys, then, with a probability of at least n^{-c_1}, $\mathcal{A}$ distinguishes the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ over $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ with a bias of at least n^{-c_2}.*

Suppose we pick random vector $\boldsymbol{u}$ as described in the protocol, then there exists a probabilistic algorithm $\mathcal{B}$ such that, using $\mathcal{A}$ as an oracle, with a probability of at least n^{-c_3}, $\mathcal{B}$ distinguishes the random variables $\mathcal{H}_{\boldsymbol{u},n^{-\mathcal{D}_1},n}$ and $unif$ with a bias of at least n^{-c_4} in time n^{c_5}.

Considering $\boldsymbol{u}$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ are picked at random as described in the protocol, suppose there exists an adversary $\mathcal{A}$ that can, with a probability of at least n^{-c_1}, distinguish two random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ with a bias of at least n^{-c_2}.

Step 1. "Amplify the bias that $\mathcal{A}$ can distinguish the random variables to at least $\frac{1}{2} - n^{-2}$"

Lemma 3.1.6. *Suppose $\mathcal{A}$ is a probabilistic circuit which if we pick the random vectors $\boldsymbol{u}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ as described in the protocol for generating the public and private keys, then, with a probability of at least n^{-c_1}, $\mathcal{A}$ distinguishes the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ over $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ with a bias of at least n^{-c_2}.*

Then there exists another circuit $\mathcal{A}'$ that uses $\mathcal{A}$ which distinguishes the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ over $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ with a bias of at least $1 - n^{-2}$.

Proof. Consider the following definition of the algorithm $\mathcal{A}'$: $\mathcal{A}'$ produces long independent sequences of both the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$, applies $\mathcal{A}$ for $N = n^{c_2+1}$ (i.e. n times the inverse of the initial bias) and takes majority of decision. For each result, let X_i denote the result of each trial, with $X_i = 1$ if $\mathcal{A}$ succeeds, $X_i = 0$ otherwise. Using Chernoff-Hoeffdings inequality, the probability that less than half of the trials is successful is given by:

$$\Pr\left[\sum_{i=1}^{N} X_i \leqslant \frac{N}{2}\right] \leqslant \exp\left(-\frac{n}{n^{c_2}+2}\right) \leqslant n^{-2}$$

Hence, with the majority vote, $\mathcal{A}'$ wins with bias greater than $1 - n^{-2}$.

□

Remark. *For the next steps, take the initial $\mathcal{A}$ to be the $\mathcal{A}'$, that is, an algorithm that wins distinguishes with bias greater than $\frac{1}{2} - n^{-2}$.*

Step 2. "Define algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to decide whether a random variable ξ is *unif* or $\mathcal{H}_{\boldsymbol{u},R,n}$."

Proof of Lemma 3.1.5.

Step 2.1. "Define a set Y of sequences $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ that allows a sequence to be determined if its in the set in polynomial time, and, if the sequence is in the set, then $\mathcal{A}$ distinguishes the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ with bias greater than $\frac{1}{2} - n^{-1}$."

Lemma 3.1.7. *Suppose $\mathcal{A}$ is a probabilistic circuit which if we pick the random vectors $\boldsymbol{u}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ as described in the protocol for generating the public and private keys, then, with a probability of at least n^{-c_1}, $\mathcal{A}$ distinguishes the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ over $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ with a bias of at least $1 - n^{-2}$.*

Then, there exists a set Y of sequences $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ with the property that a sequence y can be decided in polynomial time if it is in the set Y, and for each sequence $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m \in Y$, $\mathcal{A}$ distinguishes $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ with a bias of at least $\frac{1}{2} - n^{-1}$.

Proof. Let X be the set of sequences $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ that $\mathcal{A}$ distinguishes the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ with a bias of at least $1 - n^{-2}$, i.e. we are removing the randomness related to the choice of the sequence $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$. Since $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ can be generated in polynomial time, then there is a polynomial time algorithm which, for a fixed sequence $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ can approximate the bias of $\mathcal{A}$, with a polynomial small error, and using only a polynomial amount of times the algorithm $\mathcal{A}$. Therefore, there is a set $Y \supseteq X$ which a sequence $y \in Y$ can be decided in polynomial time, with probability exponentially close to one. For each sequence $y \in Y$, $\mathcal{A}$ distinguishes the random variables with a bias of at least $\frac{1}{2} - n^{-1}$ □

Remark. *This set Y is used in the next steps in order to separate the probability that $\mathcal{A}$ distinguishes the random variables.*

Step 2.2. "Take mt values of ξ, $t = n^{2c_1}$, and create t blocks $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ of size m each. The general idea is that if $\xi = \mathcal{H}_{\boldsymbol{u},R,n}$, then B is a proper public key and for some sequences (i.e. $B \in Y$) $\mathcal{A}$ would be able to correctly identify the random variables. Otherwise, B would be random and the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ would be as well."

This is the definition of the algorithm $\mathcal{B}$. $\mathcal{B}$ gets mt values of the the random variable ξ which is either *unif* or $\mathcal{H}_{\boldsymbol{u},R,n}$ and partitions these variables into t blocks of m variables each, creating t blocks $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$.

Step 2.3. "Setup two counters: one counts the number of sequences of $B \in Y$ (sequences which $\mathcal{A}$ can distinguish the random variables) and the other counts the number of sequences which have a correct identification from $\mathcal{A}$ (this is only done for sequences $B \in Y$)[5]."

[5] See step (2.5)

Step 2.4. "For each B, if $B \in Y$, produce as many random variables of $\mathcal{S}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ and $\mathcal{E}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ as needed for the input of $\mathcal{A}$ and give them to $\mathcal{A}$. Increment the counter if $\mathcal{A}$ correctly identifies the samples."

For each fixed block $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$, $\mathcal{B}$ generates many independent values of the random variables $\mathcal{S}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ and $\mathcal{E}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ as needed for $\mathcal{A}$ to give a correct output. $\mathcal{B}$ then computes two counters, one to count the number of sequences that are in the set $\mathcal{A}$ can distinguish random variables from $(B \in Y)$, and one to count the number of sequences that $\mathcal{A}$ correctly identified the random variables. This second counter is only used for sequences that are in the set $\mathcal{A}$ can distinguish.

Let $f(B)$ be the counter of sequences $B \in Y$. Let $g(B)$ be the counter that, the sequence $B \in Y$ **and** $\mathcal{A}$ correctly identifies the random variables $\mathcal{S}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ and $\mathcal{E}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ generated from the block B. Therefore, if $B \notin Y$, $f(B) = g(B) = 0$. If $B \in Y$, then $f(B) = 1$ and we give the random variables $\mathcal{E}$ and $\mathcal{S}$ to $\mathcal{A}$. If $\mathcal{A}$ correctly identifies the random variables, set $g(B) = 1$, otherwise, $g(B) = 0$.

Step 2.5. "If the number of sequences $B \in Y$ is greater than $n^{-c_1}\frac{t}{2}$ and the number of correct identifications by $\mathcal{A}$ is greater than $\frac{3}{4}$ of the number of sequences $B \in Y$, then output that $\xi = \mathcal{H}_{\boldsymbol{u},R,n}$, otherwise, $\xi = \mathit{unif}$."

Finally, let $f_0 = \sum f(B)$ and $g_0 = \sum g(B)$, where the sum is taken over all t blocks. If $f_0 > \frac{1}{2}n^{-c_1}t$ and $g_0 > \frac{3}{4}f_0$, then $\mathcal{B}$ outputs that the random variable $\xi = \mathcal{H}_{\boldsymbol{u},R,n}$ and $\xi = unif$ otherwise.

For this analysis, we fix $\boldsymbol{u}$, and therefore we have that $B_{\boldsymbol{u}}$ if we randomize each $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ as described in the protocol, then with a probability of at least n^{-2c_1}, we have that $\mathcal{A}$ distinguishes the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ with a bias of at least $1/2 - n^{-c_1}$.

Lemma 3.1.8. *If $\mathcal{B}$ functions as described above, then:*

1. *If $\xi = \mathcal{H}_{\boldsymbol{u},R,n}$, then with probability exponentially close to 1, $f_0 > \frac{1}{2}n^{-c_1}t$ **and** $g_0 > \frac{3}{4}f_0$;*
2. *If $\xi = unif$, then with probability exponentially close to 1, either $f_0 < \frac{1}{2}n^{-c_1}t$ **or** $g_0 < \frac{3}{4}f_0$.*

Proof. Proof of 1. Considering the case where $\xi = \mathcal{H}_{\boldsymbol{u},R,n}$, since $Y \supseteq X$, and, by Lemma 3.1.5 we are considering that $P(X) \geqslant n^{-c_1}$, then $P(Y) \geqslant n^{-c_1}$. If the sequence $\langle \boldsymbol{b}_1, \ldots, \boldsymbol{b}_m \rangle \in Y$, then that implies that $f(B) = 1$, and thus $P(f(B) = 1) > n^{-c_1}$. Considering $f(B) = 1$, then by definition of Y, $\mathcal{A}$ distinguishes $\mathcal{S}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ and $\mathcal{E}_{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m}$ with a bias of at least $\frac{1}{2} - n^{-1}$. Therefore, the probability $\mathcal{A}$ gives the right answer thus $g(B) = 1$ given $f(B) = 1$ is at least $1 - n^{-1}$.

In other words, using Chernoff bounds, the probability that the sum of the independent variables $f(B)$ is less than half of the expected value ($\mathbb{E}[f_0] = t \cdot p = n^{-c_1}t$) is given by:

$$\Pr\left[f_0 \leqslant \frac{1}{2}n^{-c_1}t\right] \leqslant \exp\left(-\frac{2(\frac{1}{2}n^{-c_1}t)^2}{t}\right) = \exp\left(-\frac{tn^{-2c_1}}{2}\right) = e^{-\frac{1}{2}} \qquad (t = n^{2c_1})$$

Thus, the probability $f_0 > \frac{1}{2}n^{-c_1}t$ is at least $1 - e^{-\frac{1}{2}}$.

Now, conditioning on $f(B) = 1$, then

$$\Pr[g(B) = 1 \mid f(B) = 1] \geqslant 1 - n^{-1}.$$

If $f_0 = f$, then g_0 is the sum of f independent random variables, each with a probability of at least $1 - n^{-1}$ of being equal to 1, so:

$$\mathbb{E}[g_0 \mid f_0 = f] \geqslant f \cdot (1 - n^{-c_1}).$$

Therefore, by Chernoff bounds:

$$\Pr\left[g_0 \leqslant \frac{3}{4}f_0 \mid f_0\right] \leqslant \exp\left(-2(1/4 - n^{-1})^2 f_0\right)$$

Thus, the probability $g_0 > \frac{3}{4}f_0$ is exponentially close to 1. Proving part 1.

Proof of 2. For the second part, we are now assuming that $\xi = unif$, then, for almost all choices of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$, the random variables $\mathcal{S}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ and $\mathcal{E}_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_m}$ are almost identical (the distance of their distribution is smaller than 2^{-n}. This implies the inequality, since $\mathcal{A}$ is trying to distinguish two random variables that are exponentially close, and therefore, the bias of the decision is exponentially small. Here, we need to prove that $\Pr[g(B) = 1 | f(B) = 1] < \frac{1}{2} + \frac{1}{8}$, and that implies that with a probability exponentially close to one, either $f_0 < \frac{1}{2}n^{-c_1}t$ or $g_0 < \frac{3}{4}f_0$. □

This completes the proof of Lemma 3.1.5 (Cryptosystem to HHP). ■

Part 2 - HHP to uSVP_γ

Lemma 3.1.9 (HHP to uSVP_γ)**.** *For all c_1, c_2, there exists c_3, c_4 and let an algorithm $\mathcal{B}$ which, if we pick a random vector $\boldsymbol{u}$ as described in the protocol for generating the public and private keys, then, with a probability of at least n^{-c_1}, $\mathcal{B}$ distinguishes the random variables $\mathcal{H}_{\boldsymbol{u},n^{-\mathcal{D}_1},n}$ and $unif$ with a bias of at least n^{-c_2}.*

Then there exists a probabilistic algorithm $\mathcal{C}$ such that, using $\mathcal{B}$ as an oracle, for all sufficiently large n, can solve any instance of size at most n^{c_3} of the $n^{\mathcal{D}_2}$-unique shortest vector problem in time n^{c_4} with a probability greater than $1 - 2^{-n}$.

Step 1. "Given a lattice $\mathcal{L}$, by Lemma 3.1.10 we can find a $U_i = \theta\nu$, where $\theta \in \mathbb{R}$ and ν is an orthogonal linear transformation, such that $U_i\boldsymbol{v} \in X$, where $\boldsymbol{v}$ is shortest vector we are trying to find and X is the set of $\boldsymbol{u}$ which $\mathcal{B}$ can distinguish the random variables $\mathcal{H}_{\boldsymbol{u},R,m}$ and *unif*. Assume that i is now fixed with this property."

Lemma 3.1.10. *For all $c_1 > 0$, there is a $c_2 > 0$ and a probabilistic algorithm which generates a random variable ν in polynomial time so that*

1. *each value of ν can be written in the form of $\theta\nu_1$, where $\theta \in \mathbb{R}$ and ν_1 is an orthogonal linear transformation of $\mathbb{R}^n$;*
2. *If X is a Lebesgue measurable subset of the unit ball of $\mathbb{R}^n$ whose density in it is at least n^{c_1} and $\boldsymbol{v} \in \mathbb{R}^n$ with $2^{-n^2} \leqslant \|\boldsymbol{v}\| \leqslant 2^{n^2}$, then $\Pr[\nu\boldsymbol{v} \in X] > n^{-c_2}$.*

Proof. Let us assume that μ is a random variable taking its values with the uniform distribution on the set of orthogonal linear transformations of $\mathbb{R}^n$ according to this distribution. Consider the following property of μ: If $\boldsymbol{v} \in \mathbb{R}^n$, $\|\boldsymbol{v}\| = 1$ is fixed, then $\mu\boldsymbol{v}$ has a uniform distribution on the set of vectors with length 1, and there are several ways to generate μ in polynomial time, e.g. randomize sequentially the vectors $\mu\boldsymbol{e}_1, \ldots, \mu\boldsymbol{e}_n$, and after $\mu\boldsymbol{e}_1, \ldots, \mu\boldsymbol{e}_i$ has been selected, $\mu\boldsymbol{e}_{i+1}$ is chosen from the uniform distribution from the set of all unit vectors orthogonal to $\mu\boldsymbol{e}_1, \ldots, \mu\boldsymbol{e}_i$.

Let $\beta \in [0,1]$ be a random variable defined in the following way: we first take a vector $\boldsymbol{w}$ with uniform distribution on the unit ball of $\mathbb{R}^n$, and let $\beta = \|\boldsymbol{w}\|$. Let γ be the random variable which takes the value $(1+1/n)^i$ with a probability $\frac{1}{2n^4+1}$ for $i = -n^4, \ldots, -1, 0, 1, \ldots, n^4$. Finally assume μ, β, γ are independent and define ν, ν_1, θ as follows: $\nu_1 = \mu, \theta = \gamma\beta, \nu = \gamma\beta\mu$. Assume that $\boldsymbol{v} \in \mathbb{R}^n$ os fixed with $2^{-n^2} \leqslant \|\boldsymbol{v}\| \leqslant 2^{n^2}$. According to the definition of γ, there is a γ_0 such that the probability of $\gamma = \gamma_0$ is $\frac{1}{2n^2+1}$ and $1 \leqslant \gamma_0\|\boldsymbol{v}\| \leqslant (1+1/n)$.

Now, to estimate $\Pr[\nu\boldsymbol{v} \in X \mid \gamma = \gamma_0]$, since μ, β, γ are independent, this is the same as $\Pr[\gamma_0\beta\mu\boldsymbol{v} \in X]$. $\mu\boldsymbol{v}$ has uniform distribution on the set of all vectors with length $\|\boldsymbol{v}\|$, and so, by definition of β, $\gamma_0\beta\mu\boldsymbol{v}$ has uniform distribution on the ball around 0 with radius $\gamma_0\boldsymbol{v}$. Since this ball contains the unit ball and the ratio of their volumes is at most $(1+1/n)^n \leqslant 3$, we get a point in X with a probability of at least $\frac{1}{3}n^{-c_1}$, therefore we have that $\Pr[\nu\boldsymbol{v} \in X \mid \gamma = \gamma_0] \geqslant \frac{1}{3}n^{-c_1}$ and thus, $\Pr[\nu\boldsymbol{v} \in X] \geqslant \frac{1}{3}n^{-c_1}\frac{1}{2n^2+1} \geqslant n^{-c_2}$. □

Step 2. "$\boldsymbol{v}$ is an $n^{\mathcal{D}_2}$-unique shortest vector of $\mathcal{L} \iff U_i\boldsymbol{v}$ is an $n^{\mathcal{D}_2}$-unique shortest vector of $U_i\mathcal{L}$."

Lemma 3.1.11. *Let $U_i = \theta\nu$, where $\theta \in \mathbb{R}$ and ν is an orthogonal linear transformation.$\boldsymbol{v}$ is an $n^{\mathcal{D}_2}$-unique shortest vector of $\mathcal{L} \iff U_i\boldsymbol{v}$ is an $n^{\mathcal{D}_2}$-unique shortest vector of $U_i\mathcal{L}$.*

Proof. Considering that $U_i = \theta\nu$, $\theta \in \mathbb{R}$ and ν is an orthogonal linear transformation, the orthogonal linear transformation maintains the lengths of all vectors and angles between different vectors. That is, if $\boldsymbol{v} \in \mathcal{L}$ is such that $\|\boldsymbol{v}\| = \lambda_1(\mathcal{L})$, then $\|\nu\boldsymbol{v}\| = \|\boldsymbol{v}\| = \lambda_1(\mathcal{L})$.

The scaling θ scales all vectors of the lattice by the same factor, thus, if $\boldsymbol{w} \in \mathcal{L}$ is such that $\|\boldsymbol{w}\| = \lambda_2(\mathcal{L}) \geqslant n^{\mathcal{D}_2}\lambda_1(\mathcal{L})$, then $\|\theta\boldsymbol{w}\| = \theta\lambda_2(\mathcal{L}) = \lambda_2(\theta\mathcal{L}) \geqslant n^{\mathcal{D}_2}\lambda_1(\theta\mathcal{L})$. □

Step 3. "If Λ is a lattice with an n^c-unique shortest vector $\boldsymbol{u}$, then $\mathcal{L} = \Lambda^*$ is a (d, M)-lattice with $d \geqslant n^{c'}M$ and $d_{\mathcal{L}} = \|\boldsymbol{u}\|^{-1}$."

Lemma 3.1.12. *Let $\boldsymbol{B}, \boldsymbol{D} \in \mathbb{R}^{m\times n}$ be a pair of dual bases, then, for all $i = 1, \ldots, n$, $\boldsymbol{B}' = (\pi_i(\boldsymbol{b}_i), \ldots, \pi_i(\boldsymbol{b}_n))$ and $\boldsymbol{D}' = (\boldsymbol{d}_i, \ldots, \boldsymbol{d}_n)$ are also dual bases.*

Proof. Let us prove by induction for $i \geqslant 2$. Let $i = 2$, $\boldsymbol{B}' = (\pi_2(\boldsymbol{b}_2), \ldots, \pi_2(\boldsymbol{b}_n))$ and $\boldsymbol{D}' = (\boldsymbol{d}_2, \ldots, \boldsymbol{d}_n)$. From Def. 2.3.4, we need to prove that $\text{span}(\boldsymbol{D}') = \text{span}(\boldsymbol{B}')$ and $\boldsymbol{B}'^T\boldsymbol{D}' = I$.

First, $\boldsymbol{B}$ and $\boldsymbol{D}$ span the same vector space W. The span of $\boldsymbol{B}'$ is the orthogonal complement of $\boldsymbol{b}_1$ in W. By the definition of the dual basis, each vector $\boldsymbol{d}_2, \ldots, \boldsymbol{d}_n$ is orthogonal to $\boldsymbol{b}_1$, and linearly independent, therefore they also span the orthogonal complement of $\boldsymbol{b}_1$ in W, and thus $\boldsymbol{B}'$ and $\boldsymbol{D}'$ have the same linear span.

Now let us prove that $\boldsymbol{B}'^T\boldsymbol{D}' = I$ by showing that $\langle \pi_2(\boldsymbol{b}_i)\,,\, \boldsymbol{d}_j\rangle = \delta_{ij}, \forall i, j > 1$, where $\delta_{ij} = 1$ for $i = j$ and 0 otherwise. By the definition of $\pi_2(\cdot)$, we have $\pi_2(\boldsymbol{b}_i) = \boldsymbol{b}_i - \mu_{i,1}\boldsymbol{b}_1$, therefore,

$$\begin{aligned}
\langle \pi_2(\boldsymbol{b}_i)\,,\, \boldsymbol{d}_j\rangle &= \langle \boldsymbol{b}_i - \mu_{i,1}\boldsymbol{b}_1\,,\, \boldsymbol{d}_j\rangle \\
&= \langle \boldsymbol{b}_i\,,\, \boldsymbol{d}_j\rangle - \mu_{i,1}\langle \boldsymbol{b}_1\,,\, \boldsymbol{d}_j\rangle \\
&= \langle \boldsymbol{b}_i\,,\, \boldsymbol{d}_j\rangle - \cancel{\mu_{i,1}\delta_{1j}} && (j > 1, \delta_{1j} = 0) \\
&= \delta_{ij}
\end{aligned}$$

□

Lemma 3.1.13. *If Λ is a lattice with an n^c-unique shortest vector $\boldsymbol{u}$, then $\mathcal{L} = \Lambda^*$ is a (d, M)-lattice with $d \geqslant n^{c'}M$ and $d_{\mathcal{L}} = \|\boldsymbol{u}\|^{-1}$.*

Proof. Let $\boldsymbol{u} \in \Lambda$ be the shortest non-zero vector of Λ, there exists a basis $A = \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ of Λ such that $\boldsymbol{u} = \boldsymbol{a}_1$ is one of the basis vectors. Consider the basis of the dual lattice $\mathcal{L} = \Lambda^*$ defined by $B = \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ such that $\langle \boldsymbol{b}_i\,,\, \boldsymbol{a}_j\rangle = \delta_{i,j}$. Thus, for $i = j = 1$, we have that $\langle \boldsymbol{b}_1\,,\, \boldsymbol{u}\rangle = 1$ and every other $\boldsymbol{b}_i, i = 2, \ldots, n$ is orthogonal to $\boldsymbol{u}$, defining an n-dimensional hyperplane H orthogonal to $\boldsymbol{u}$. By definition, $\langle \boldsymbol{b}_1\,,\, \boldsymbol{u}\rangle = 1$ implies that the distance of $\boldsymbol{b}_1$ to the hyperplane H is:

$$d_{\mathcal{L}} = \frac{\langle \boldsymbol{b}_1\,,\, \boldsymbol{u}\rangle}{\|\boldsymbol{u}\|} = \|\boldsymbol{u}\|^{-1}$$

Now consider the lattice $\mathcal{L} \cap H = \mathcal{L}(\boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$. This lattice is the dual of the factor lattice[6] $\Lambda_1 = \Lambda/\Lambda_u$ (See Lemma 3.1.12). By Banaszczyk's transference theorem (Theorem 2.3.5), we have that:

$$\lambda_1(\Lambda_1) \cdot \lambda_n(\mathcal{L}_1) \leqslant n$$

By the definition of Λ, the factor lattice Λ_1 has $\lambda_1(\Lambda_1) \geqslant (n^c - 1)\|\boldsymbol{u}\|$ and, thus:

$$\lambda_n(\mathcal{L}_1) \leqslant \frac{n}{(n^c - 1)\|\boldsymbol{u}\|} = n^{-c'}\|\boldsymbol{u}\|^{-1}$$

This way, $\mathcal{L}_1$ has a basis with elements no longer than $M = n^{-c'}\|\boldsymbol{u}\|^{-1}$, and the distance between hyperplanes is $d_{\mathcal{L}} = \|\boldsymbol{u}\|^{-1}$ which is $d_{\mathcal{L}} = n^{c'}M$.

□

Step 4. "Let $J = (U_i\mathcal{L})^*$. Since $U_i\mathcal{L}$ has an $n^{\mathcal{D}_2}$-unique shortest vector, then J is a $(1, n^{-\mathcal{D}'_2})$ lattice, where $\mathcal{D}'_2 \approx \mathcal{D}_2 - 1$."

By Lemma 3.1.13, if we take a lattice with an $n^{\mathcal{D}_2}$-unique shortest vector $\boldsymbol{u}$ with $\|\boldsymbol{u}\| = 1$, then the dual lattice J will have a basis with elements no longer than $M = n^{-\mathcal{D}'_2}\|\boldsymbol{u}\|^{-1} = n^{-\mathcal{D}'_2}$ and the distance between hyperplanes will be $d_J = \|\boldsymbol{u}\|^{-1} = 1$. Thus, the dual lattice J will be a $(1, n^{-\mathcal{D}'_2})$ lattice, with $\mathcal{D}'_2 \approx \mathcal{D}_2 - 1$.

Step 5. "The distance of the distributions $\mathcal{H}_{\boldsymbol{u},R,m}$ and $\xi_{(J,K,R)}$ is exponentially small and $\eta_K = \mathit{unif}$, therefore if $\mathcal{B}$ can distinguish $\mathcal{H}_{\boldsymbol{u},R,m}$ and unif, it can also distinguish $\xi_{(J,K,R)}$ and η_K."

In the next few paragraphs, we want to prove that the distance of the distributions $\mathcal{H}_{\boldsymbol{u},R,m}$ and $\xi_{(J,K,R)}$ is exponentially small, so let us define again the variables we are using for ease of reading.

- Let $\mathrm{pert}(R, m) := \sum_{i=1}^{m} \boldsymbol{t}_i$, where each $\boldsymbol{t}_i \hookleftarrow\$ B_n(R), \forall i = 1, \ldots, m$.
  - $\mathrm{pert}(R, m)$ represents a perturbation generated by adding m vectors sampled from a ball of radius R.
  - For this exposition, consider $R = n^3M$.
- Let $\mathcal{H}_{\boldsymbol{u},R,m} := \boldsymbol{y} + \boldsymbol{w}$, $\boldsymbol{y} \hookleftarrow\$ \{\, KU^n \mid \langle \boldsymbol{y}, \boldsymbol{u} \rangle \in \mathbb{Z} \,\}$ and $\boldsymbol{w} = \mathrm{pert}(R, m)$.
  - In words, $\boldsymbol{y}$ represents a vector in any of the $(n-1)$-dimensional hyperplanes $H_i := \{\, \boldsymbol{y} \in \mathbb{R}^n \mid \langle \boldsymbol{y}, \boldsymbol{u} \rangle = i, i \in \mathbb{Z} \,\}$ that are orthogonal to the vector $\boldsymbol{u}$. The minimum distance between the hyperplanes is $d = \|\boldsymbol{u}\|^{-1}$.

[6] Def. 3.1.2

- KU^n represents a large cube with sides K. For this exposition, consider $K = 2^{n \log n}$.
- Therefore, $\mathcal{H}_{\boldsymbol{u},R,m}$ represents vectors inside the "large cube", in any of the parallel hyperplanes orthogonal to $\boldsymbol{u}$, with an added perturbation of $\text{pert}(R, m)$.

- Let $\xi_{(J,K,R)} := \boldsymbol{x} + \boldsymbol{w}$, $\boldsymbol{x} \leftarrow\$ \{ KU^n \cap J \}$ and $\boldsymbol{w} = \text{pert}(R, m)$.
  - For a lattice J, $\boldsymbol{x}$ represents any point of the lattice inside the "large cube".
  - Therefore, $\xi_{(J,K,R)}$ represents lattice J vectors inside the "large cube", with an added perturbation of $\text{pert}(R, m)$.

The main idea for this proof is that if $\boldsymbol{w}$ is not too far from H, then the distribution of the projection of $\boldsymbol{w}$ onto H is almost uniform. Therefore, conditioned on $\text{pert}(R, m)$, the distribution of the projection of $\xi_{(J,K,R)}$ onto H is almost uniform as well, and thus not far from $\mathcal{H}_{\boldsymbol{u},R,m}$. When we compute $\boldsymbol{w} := \sum^m \boldsymbol{t}_i$, we write each $\boldsymbol{t}_i := \boldsymbol{r}_i + \boldsymbol{s}_i$, where $\boldsymbol{r}_i$ is the component orthogonal to H, then if we first randomize $\boldsymbol{r}_i$, $\boldsymbol{s}_i$ will be chosen from the $(n-1)$-dimensional ball of radius $R' := \sqrt{R^2 - \|\boldsymbol{r}_i\|^2}$. If $\boldsymbol{r}_i$ is not too large, then, with high probability, $\boldsymbol{s}_i$ is contained in a tile of H (i.e. a parallepiped of the lattice J) that is completely inside the ball $B_{n-1}(R')$ which $\boldsymbol{s}_i$ was chosen from, and thus, this $\boldsymbol{s}_i$ is chosen uniformly in the parallelepiped.

Step 5.1. "Let ξ_i be the distance from $\boldsymbol{t}_i$ to H. Let $\eta = \sum_{i=1}^m \xi_i$. Let I denote an interval contained in $[-\frac{mR}{40}, \frac{mR}{40}]$. If $\eta \in I$, then, w.h.p. there are at least $m/4$ values $\boldsymbol{s}_i$, such that they are chosen from $B_{n-1}(R')$, with $R' \geqslant n^2 M$."

Lemma 3.1.14. *Let $\boldsymbol{w} = \sum_{i=1}^m \boldsymbol{t}_i$, where each $\boldsymbol{t}_i$ is chosen at random from $B_n(R)$. There is a $c > 0$ and n_0 so that if $n \geqslant n_0$ and R are fixed, $\epsilon > 0$, and I is an interval of length ϵ contained in $[-\frac{mR}{40}, \frac{mR}{40}]$, then the following holds. For $1 \leqslant i \leqslant m$, let ξ_i be the signed distance from H to $\boldsymbol{t}_i$. Let us assume a coordinate system in which we can write $\boldsymbol{t}_i = \boldsymbol{r}_i + \boldsymbol{s}_i$, where $\boldsymbol{r}_i = (0, 0, \ldots, 0, \xi_i)$. Let G be the event "there are at least $m/4$ integers $i \in [1, m]$ such that $\boldsymbol{s}_i$ is chosen from an $(n-1)$-dimensional ball of radius at least $n^2 M$." Let $\eta = \sum_{i=1}^m \xi_i$. Then the conditional probability of G with the condition $\eta \in I$ is at least $1 - 2^{-cm}$.*

Remark. *Lemma 3.1.14 above states that, if $\boldsymbol{w}$ is not too far from H, i.e. more than $\frac{mR}{40}$, then with high probability at least $1/4$ of the $\boldsymbol{s}_i$ are chosen from a ball $B_{n-1}(R')$, where $R' \geqslant n^2 M$. This Lemma and its proof are available in [AD97], Lemma 5.1.*

Step 5.2. "For $n' \geqslant n$, assume each $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{n'}$ is chosen from $B_{n-1}(R')$, with $R' \geqslant n^2 M$. If we tile H with $\mathcal{P}'$, then, w.h.p., at least one of $\boldsymbol{s}_i$, $i = 1, \ldots, n'$, is chosen from a tile that is completely contained in its respective ball. In this case, $\boldsymbol{s}_i$ is uniformly distributed in $\mathcal{P}'$."

Lemma 3.1.15. *Let $J = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in_R \Lambda$, where Λ is a distribution on the set of (d, M) lattices and $J^{(d,M)} = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$. Let $m = c_0 n$, for some $c_0 \geqslant 4$, let $R = n^3 M$, and let $\mathcal{P}' = \mathcal{P}^-(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$. For $n' \geqslant n$, assume each of $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{n'}$ is chosen from $(n-1)$ dimensional ball of radius at least $n^2 M$. If we tile H with copies of $\mathcal{P}'$, then, w.h.p. $(1 - 2^{-n' \log n})$ at least one of $\boldsymbol{s}_i$ is chosen from a tile completely contained in its respective ball.*

Proof. Recall that, by Lemma 3.1.13, $\max_{i \leqslant n-1} \|\boldsymbol{b}_i\| \leqslant M$. Let S_i be the ball from which $\boldsymbol{s}_i$ is chosen. For any vector $\boldsymbol{s}_i$, let $\mathcal{P}_i$ denote the copy of $\mathcal{P}'$ containing $\boldsymbol{s}_i$. Let C be the set of vectors $\boldsymbol{s}_i$ such that $\mathcal{P}_i$ lies entirely within S_i. For a tile to be partially contained in S_i, it has to be close enough to the boundary of S_i, that is, within a distance of at most $\sqrt{n-1}M$ to the boundary. Therefore, we can define the region inside S_i, with a width of $\sqrt{n-1}M$ such that any tile that falls in this region will be partially contained in S_i, let this region be called the boundary region. The volume of this region will be proportional to the volume of the $(n-1)$ dimensional ball of radius $R' = n^2 M$ minus the ball of radius $R_B = (n^2 M) - \sqrt{n-1}M$, i.e. the shell of width $\sqrt{n-1}M$, which we can approximate to the volume of the $(n-2)$ dimensional ball times the width, considering $\sqrt{n-1}M \ll n^2 M$.

$$\text{Vol(Boundary Region)} \propto ((n^2 M)^{n-1} - (n^2 M - \sqrt{n-1}M)^{n-1}) \approx (n^2 M)^{n-2} \sqrt{n-1} M$$

Therefore, by a volume ratio argument, we can bound the probability an $\boldsymbol{s}_i$ will fall in this region, and thus, not be in the set C.

$$\frac{\text{Vol(Boundary Region)}}{\text{Vol}(S_i)} = \frac{(n^2 M)^{n-2} \sqrt{n-1} M}{(n^2 M)^{n-1}} = \frac{\sqrt{n-1}}{n^2} \leqslant \frac{1}{n}$$

Thus, we can say that $\Pr[\boldsymbol{s}_i \notin C] \leqslant 1/n$, and hence, for $n' \geqslant n$ values of $\boldsymbol{s}_i$,

$$\Pr[\text{all } \boldsymbol{s}_i \notin C] \leqslant (1/n)^{n'} \Rightarrow \Pr[\text{at least one } \boldsymbol{s}_i \in C] \geqslant 1 - (1/n)^{n'} = 1 - 2^{-n' \log n}$$

□

Consider $\boldsymbol{w}$ *bad* if either less than $m/4$ $\boldsymbol{s}_i$ are chosen from balls of $R' \geqslant n^2 M$ (step 5.1 fails), or more than $m/4$ $\boldsymbol{s}_i$ are chosen from balls of $R' \geqslant n^2 M$ *but* none of these $\boldsymbol{s}_i$ are chosen from a tile completely contained in its respective ball (step 5.1 succeeds but step 5.2 fails).

Step 5.3. "The distance between projecting $\boldsymbol{w}$ onto H modulo $\mathcal{P}'$ and the uniform distribution on $\mathcal{P}'$ is given by the case where $\boldsymbol{w}$ is *bad*, since when $\boldsymbol{w}$ is *good*, $\boldsymbol{s}_i$ is uniform in $\mathcal{P}'$."

Lemma 3.1.16. *Let $J = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in_R \Lambda$, where Λ is a distribution on the set of (d, M) lattices and $J^{(d,M)} = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$. Let $m = c_0 n$, for some $c_0 \geqslant 4$, let $R = n^3 M$, and let $\mathcal{P}' = \mathcal{P}^-(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$. For $\epsilon > 0$ and for any interval I of length ϵ contained in $[-\frac{mR}{40}, \frac{mR}{40}]$, let $\boldsymbol{w} = \sum_{i=1}^{m} \boldsymbol{t}_i$, where each $\boldsymbol{t}_i \in B_n(R)$.*

Then the distribution obtained by projecting $\boldsymbol{w}$ onto H modulo P' differs from the uniform distribution on P' by at most $2^{-c_1 m}$ for some $c_1 > 0$, even with the condition that the signed distance of $\boldsymbol{w}$ to H is in I.

Proof. Let $\boldsymbol{w} := \sum^{m} \boldsymbol{t}_i$, and we write each $\boldsymbol{t}_i := \boldsymbol{r}_i + \boldsymbol{s}_i$, where $\boldsymbol{r}_i$ is the component orthogonal to H. We say that a $\boldsymbol{w}$ is *bad* if either:

1. fewer than $m/4$ of the $\boldsymbol{s}_i$ are chosen from $(n-1)$ dimensional balls of radius at least $n^2 M$; or
2. at least $m/4$ of the $\boldsymbol{s}_i$ are chosen from $(n-1)$ dimensional balls of radius at least $n^2 M$ but none of these $\boldsymbol{s}_i$ are chosen from a tile completely contained in its respective ball.

Otherwise, we say that $\boldsymbol{w}$ is *good*.

By Lemma 3.1.14, w.p. at least $1 - 2^{-cm}$, at least $m/4$ values of $\boldsymbol{s}_i$ are chosen from $(n-1)$ dimensional balls of radius $R' \geqslant n^2 M$. Let $n' \geqslant m/4 \geqslant n$, then, by Lemma 3.1.15, w.p. at least $1 - 2^{-n' \log n} \geqslant 1 - 2^{-\frac{m}{4} \log n}$, at least one of these $\boldsymbol{s}_i$ is chosen from a copy of $\mathcal{P}'$ that is completely contained in its ball. In this case, this $\boldsymbol{s}_i$ is uniformly distributed in $\mathcal{P}'$, and so is the projection of $\boldsymbol{w} = \sum \boldsymbol{t}_i$. The probability that $\boldsymbol{w}$ is *good* is given by $\Pr[\boldsymbol{w} \text{ is } \textit{good}] \geqslant 1 - 2^{-cm} \cdot 2^{-\frac{m}{4} \log n} = 1 - 2^{-c_3 m}$, for some $c_3 > 0$.

Now, let us tile $\mathcal{P}'$ with small parallelepipeds, and for any given tile a, let $\Pr[a]$ denote the probability that the projection of $\boldsymbol{w}$ is in a and let $\textit{unif}(a)$ denote the probability that a point chosen uniformly from $\mathcal{P}'$ is in a. Let A denote the subset of tiles in which $\Pr[a] \geqslant \textit{unif}(a)$, and B denote the subset in which $\Pr[a] < \textit{unif}(a)$. The distance of the distribution of the projection of $\boldsymbol{w}$ from the uniform distribution on $\mathcal{P}'$ is given by:

$$\sup \{ \; |\Pr[A] - \textit{unif}(A)| + |\Pr[B] - \textit{unif}(B)| \; \}$$

Therefore, for the set of tiles A, we can calculate the distance as:

$$
\begin{aligned}
&\sum_{a\in A}\left[\Pr\left[a\right]-\mathit{unif}(a)\right]\\
&=\sum_{a\in A}\left[\Pr\left[a\mid \boldsymbol{w}\text{ is }\mathit{good}\right]\Pr\left[\boldsymbol{w}\text{ is }\mathit{good}\right]+\Pr\left[a\mid \boldsymbol{w}\text{ is }\mathit{bad}\right]\Pr\left[\boldsymbol{w}\text{ is }\mathit{bad}\right]-\mathit{unif}(a)\right]\\
&=\sum_{a\in A}\left[\mathit{unif}(a)(1-2^{-c_3m})+\Pr\left[a\mid \boldsymbol{w}\text{ is }\mathit{bad}\right]\Pr\left[\boldsymbol{w}\text{ is }\mathit{bad}\right]-\mathit{unif}(a)\right]\\
&=\sum_{a\in A}\left[\mathit{unif}(a)(1-2^{-c_3m}-1)+\Pr\left[a\mid \boldsymbol{w}\text{ is }\mathit{bad}\right]\Pr\left[\boldsymbol{w}\text{ is }\mathit{bad}\right]\right]\\
&=\sum_{a\in A}\left[\mathit{unif}(a)(-2^{-c_3m})+\Pr\left[a\mid \boldsymbol{w}\text{ is }\mathit{bad}\right]\Pr\left[\boldsymbol{w}\text{ is }\mathit{bad}\right]\right]\\
&\leqslant\sum_{a\in A}\left[\mathit{unif}(a)(-2^{-c_3m})\right]+\Pr\left[\boldsymbol{w}\text{ is }\mathit{bad}\right]\\
&\leqslant\sum_{a\in A}\left[\mathit{unif}(a)(-2^{-c_3m})\right]+2^{-c_3m}\\
&\leqslant 2^{-c_3m}\left(1-\sum_{a\in A}\mathit{unif}(a)\right)\leqslant 2^{-c_3m}
\end{aligned}
$$

Since the analysis for the set of tiles B is the same, the total distance between the distributions is at most $2\cdot 2^{-c_3m}\leqslant 2^{-c_1m}$, for some $c_1>0$. □

Corollary 3.1.1. *Let $J=\mathcal{L}(\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n)\in_R\Lambda$, where Λ is a distribution on the set of (d,M) lattices and $J^{(d,M)}=\mathcal{L}(\boldsymbol{b}_1,\ldots,\boldsymbol{b}_{n-1})$. Let $m=c_0n$, for some $c_0\geqslant 4$, let $R=n^3M$, and let $\mathcal{P}'=\mathcal{P}^-(\boldsymbol{b}_1,\ldots,\boldsymbol{b}_{n-1})$. For $\epsilon>0$ and for any interval I of length ϵ contained in $[-\frac{mR}{40},\frac{mR}{40}]$, let $\boldsymbol{w}=\sum_{i=1}^m\boldsymbol{t}_i$, where each $\boldsymbol{t}_i\in B_n(R)$.*

Consider the distribution $\mathcal{D}$ obtained by choosing $\boldsymbol{p}\in_R H$ and projecting $\boldsymbol{p}+\boldsymbol{w}$ onto H. The distance of $\mathcal{D}$ from the uniform distribution in H is at most 2^{-c_1m}, for some $c_1>0$, even conditioned on the signed distance of $\boldsymbol{w}$ from H being in I. Moreover, if H' is any coset of H intersecting J, then the Lemma holds for any distribution for $\boldsymbol{p}$ in which, if we tile H with copies of $\mathcal{P}'$, each copy of $\mathcal{P}'$ is equally likely to contain $\boldsymbol{p}$.

Proof. By Lemma 3.1.16, de distribution obtained by projecting $\boldsymbol{w}$ onto H mod $\mathcal{P}'$ is within 2^{-c_1m} of the uniform distribution on $\mathcal{P}'$. By randomizing the starting point $\boldsymbol{p}$, we eliminate the need to reduce modulo $\mathcal{P}'$, since the probability that $\boldsymbol{p}$ is in any particular copy of $\mathcal{P}'$ is the same. □

Remark. *In order to apply the Lemma to the distribution $\xi_{J,K,R}$, we still need to bound the ratio of lattice points in KU^n near the surface of the cube. Since we are considering a cube with sides $K\geqslant 2^{c_4n}d_J$ and the $\max\|\boldsymbol{b}_i\|=d_J$, that would define a boundary region as a shell of the original cube with exponentially small width, and thus, the ratio of points in this region will be exponentially small compared to the ones in the whole cube. This proof can be seen in its entirety in [AD97], Lemma 3.9 on page 26.*

Now, for the next step, we take a polynomial amount of points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m'} \in K'U^n$, with $K' = n^c d_J$. Let $\boldsymbol{a}_{ij} = \boldsymbol{p}_i - \boldsymbol{p}_j$, for $1 \leqslant i < j \leqslant m'$. There are at most n^c hyperplanes H', parallel to H, that intersect $K'U^n \cap J$. Therefore, a polynomial fraction of $\boldsymbol{a}_{ij}$ will be such that both $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are in the same hyperplane H' (at least $(\frac{1}{n^c})^2\binom{m'}{2}$), and in this case $\boldsymbol{a}_{ij} \in H$.

Step 5.4. "Let $\boldsymbol{v} \in \{\, \boldsymbol{a}_{ij} \mid 1 \leqslant i < j \leqslant m' \,\}$. Define the random variable $\delta_{\boldsymbol{v}} := \boldsymbol{u} + \alpha\boldsymbol{v} + \boldsymbol{w}$, where $\boldsymbol{u} \leftarrow\!\$\ KU^n \cap J, \alpha \leftarrow\!\$\ [0,1]$ and $\boldsymbol{w} := \mathrm{pert}(R, m)$. If $\boldsymbol{v} \in H$, then the projection of $\delta_{\boldsymbol{v}} (= \mathcal{H}_{\boldsymbol{u},R,m})$ onto H will be almost the same as the projection of $\xi_{J,K,R}$ onto H. If $\boldsymbol{v} \notin H$, then $\delta_{\boldsymbol{v}} (= \mathit{unif})$ will be essentially the same distribution as η_K."

Lemma 3.1.17. *Let $J = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in_R \Lambda$, where Λ is a distribution on the set of (d, M) lattices and $J^{(d,M)} = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$. Let $\boldsymbol{v} \in J$ satisfy $\|\boldsymbol{v}\| \leqslant K^{1/c}$. Then:*

1. *if $\boldsymbol{v} \in H$, then the distance between the distributions $\xi_{J,K,R}$ and $\delta_{\boldsymbol{v}}$ is at most 2^{-cn} for some $c > 0$;*
2. *if $\boldsymbol{v} \notin H$, then the distance between the distributions η_K and $\delta_{\boldsymbol{v}}$ is at most 2^{-cn} for some $c > 0$;*

Proof. Let $\mathcal{P}' = \mathcal{P}^-(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$. Let $\boldsymbol{w} = \sum_{i=1}^{m} \boldsymbol{t}_i$, where each $\boldsymbol{t}_i \in B_n(R), R = n^3 M$. Define the random variables $\delta_{\boldsymbol{v}} := \boldsymbol{u}_0 + \alpha\boldsymbol{v} + \boldsymbol{w}_0$ and $\xi_{J,K,R} := \boldsymbol{u}_1 + \boldsymbol{w}_1$, where $\boldsymbol{u}_0, \boldsymbol{u}_1 \leftarrow\!\$\ KU^n \cap J, \alpha \leftarrow\!\$\ [0,1]$ and $\boldsymbol{w}_0, \boldsymbol{w}_1$ are copies of $\boldsymbol{w}$.

Proof of (1): Suppose that, instead of sampling from $KU^n \cap J$, we were to choose $\boldsymbol{u} \in_R J$. Fix any coset H' of H intersecting J and tile H' with copies of $\mathcal{P}'$. Since $\boldsymbol{v} \in H$, $\boldsymbol{u}_0 + \alpha\boldsymbol{v}$ is uniformly chosen from among these tiles, as is $\boldsymbol{u}_1$. For $\epsilon > 0$, let us partition the space $\mathbb{R}^n$ into strips of width ϵ parallel to H. For each strip σ, the difference between the projection of $\boldsymbol{w}_0$ onto H and the uniform distribution on H is exponentially small and so is $\boldsymbol{w}_1$ (by Corollary 3.1.1).

Let $\delta'_{\boldsymbol{v}} = \boldsymbol{u}_0 + \alpha\boldsymbol{v} + \boldsymbol{w}_0$ and $\xi = \boldsymbol{u}_1 + \boldsymbol{w}_1$, where both $\boldsymbol{u}_0, \boldsymbol{u}_1 \leftarrow\!\$\ J$. With a similar approach to the proof of Lemma 3.1.16, let us tile each strip σ with n-dimensional small parallelepipeds with a height of ϵ perpendicular to H. For any given tile T, let A denote the subset of tiles in which $\Pr[\xi \in T] \geqslant \Pr[\delta_{\boldsymbol{v}} \in T]$, and B the subset which holds the opposite. When we define a strip, we specify $\sigma = (H_\sigma, I_\sigma)$ by naming H_σ as the coset of H intersecting J nearest to σ and I_σ the interval containing the distances of the points in σ from H_σ. For the set of tiles A, the distance of the distributions ξ and $\delta_{\boldsymbol{v}}$ is calculated over all strips σ and over all tiles $T \in A$.

$$\sum_{\sigma}\sum_{T\in A\cap\sigma}\left(\Pr\left[\xi\in T\mid\xi\in\sigma\right]\Pr\left[\xi\in\sigma\right]-\Pr\left[\delta'_{\boldsymbol{v}}\in T\mid\delta'_{\boldsymbol{v}}\in\sigma\right]\Pr\left[\delta'_{\boldsymbol{v}}\in\sigma\right]\right) \tag{3.1}$$

$$=\sum_{\sigma}\sum_{T\in A\cap\sigma}\left(\Pr\left[\xi\in T\mid\xi\in\sigma\right]-\Pr\left[\delta'_{\boldsymbol{v}}\in T\mid\delta'_{\boldsymbol{v}}\in\sigma\right]\right)\Pr\left[\boldsymbol{u}\in H_\sigma\right]\Pr\left[\boldsymbol{w}\in I_\sigma\right] \tag{3.2}$$

$$=\sum_{H_\sigma}\Pr\left[\boldsymbol{u}\in H_\sigma\right]\sum_{I_\sigma}\sum_{T\in A\cap(H_\sigma,I_\sigma)}\left(\Pr\left[\xi\in T\mid\xi\in\sigma\right]-\Pr\left[\delta'_{\boldsymbol{v}}\in T\mid\delta'_{\boldsymbol{v}}\in\sigma\right]\right)\Pr\left[\boldsymbol{w}\in I_\sigma\right] \tag{3.3}$$

$$\leqslant 2\cdot 2^{-c'm}\sum_{I_\sigma}\Pr\left[\boldsymbol{w}\in I_\sigma\right]\leqslant 2^{-cm} \tag{3.4}$$

Where step 3.2 comes from the fact that each $\delta'_{\boldsymbol{v}}=\boldsymbol{u}_0+\alpha\boldsymbol{v}+\boldsymbol{w}_0$ (respectively $\xi=\boldsymbol{u}_1+\boldsymbol{w}_1$) falls in the strip σ only if $\boldsymbol{u}_0+\alpha\boldsymbol{v}\in H_\sigma$ and $\boldsymbol{w}_0\in I_\sigma$, (respectively $\boldsymbol{u}_1\in H_\sigma$ and $\boldsymbol{w}_1\in I_\sigma$). Since $\boldsymbol{v}\in H$, for ξ we only need $\boldsymbol{u}_0\in H_\sigma$. In step 3.3, we split the strip σ into (H_σ,I_σ). Step 3.4 comes from the fact that the probability that $\boldsymbol{w}$ is at a distance greater than $\frac{mR}{40}$ is exponentially small in m, hence, w.h.p. $\boldsymbol{u}\in H_\sigma$ and by Corollary 3.1.1, which bounds the distance of "$\boldsymbol{p}+\boldsymbol{w}$" (i.e. our ξ and $\delta_{\boldsymbol{v}}$) from the uniform distribution on H for strips of distance at most $\frac{mR}{40}$ from a coset of H intersecting J.

Since the analysis for the set of tiles B is the same, the total distance between the distributions is at most 2^{-cn}, for some $c>0$, proving part 1.

Proof of (2): Using the same approach, let A denote the subset of tiles in which $\Pr\left[\eta_K\in T\right]\geqslant\Pr\left[\delta_{\boldsymbol{v}}\in T\right]$, and B the subset which holds the opposite. As with the remark after Corollary 3.1.1, since the ratio of lattice points near the surface of the cube KU^n is exponentially small, we can restrict the analysis to only those that are not in the "boundary region". For the set of tiles A, we can calculate the distance as:

$$\Pr\left[\eta_K\in A\right]-\Pr\left[\delta_{\boldsymbol{v}}\in A\right] \tag{3.5}$$

$$=\sum_{T\in A}\left[\frac{\mathrm{Vol}(T)}{\mathrm{Vol}(KU^n)}-\Big(\Pr\left[\delta_{\boldsymbol{v}}\in T\mid\boldsymbol{w}_1\text{ is }\textit{good}\right]\Pr\left[\boldsymbol{w}_1\text{ is }\textit{good}\right]\right.$$

$$\left.+\Pr\left[\delta_{\boldsymbol{v}}\in T\mid\boldsymbol{w}_1\text{ is }\textit{bad}\right]\Pr\left[\boldsymbol{w}_1\text{ is }\textit{bad}\right]\Big)\right] \tag{3.6}$$

$$\leqslant\sum_{T\in A}\left[\frac{\mathrm{Vol}(T)}{\mathrm{Vol}(KU^n)}-\Pr\left[\delta_{\boldsymbol{v}}\in T\mid\boldsymbol{w}_1\text{ is }\textit{good}\right]\Pr\left[\boldsymbol{w}_1\text{ is }\textit{good}\right]\right]+\Pr\left[\boldsymbol{w}_1\text{ is }\textit{bad}\right] \tag{3.7}$$

$$\leqslant\sum_{T\in A}\left[\frac{\mathrm{Vol}(T)}{\mathrm{Vol}(KU^n)}-\Pr\left[\delta_{\boldsymbol{v}}\in T\mid\boldsymbol{w}_1\text{ is }\textit{good}\right]\Pr\left[\boldsymbol{w}_1\text{ is }\textit{good}\right]\right]+2^{-\frac{m}{4}\log\frac{m}{4}}\leqslant 2^{-cm} \tag{3.8}$$

Step 3.6 comes from the fact that η_K is uniform over KU^n so its probability is proportional to its volume and for $\delta_{\boldsymbol{v}}$ we can separate into the cases where $\boldsymbol{w}_1$ is *good* or *bad*. For

Step 3.7, we are considering the fact that $\Pr[\boldsymbol{w}_1 \text{is } \textit{bad}]$ is exponentially small, and thus its contribution to the distance is very small as well. For the last step 3.8, since $\boldsymbol{v} \notin H$, with probability at least $1 - 2^{-c_4 n}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ are not in the same coset, it follows from Corollary 3.1.1 that the difference is bounded by $2^{-c_5 n}$, for some $c_5 > 0$. Since the analysis for the set of tiles B is the same, the total distance between the distributions is at most 2^{-cn}, for some $c > 0$, proving part 2.

□

Remark. *We showed that the distance of the distributions $\delta_{\boldsymbol{v}}$, for $\boldsymbol{v} \in H$, and $\xi_{(J,K,R)}$ is exponentially small. For the proof of the theorem, we needed to show the distance to the distribution $\mathcal{H}_{\boldsymbol{u},R,m}$. However, it is important to note that $\delta_{\boldsymbol{v}} = \boldsymbol{u} + \alpha\boldsymbol{v} + \boldsymbol{w}$ is essentially the same as the distribution $\mathcal{H}_{\boldsymbol{u},R,m} = \boldsymbol{x} + \boldsymbol{w}$, since for $\boldsymbol{v} \in H$, $\boldsymbol{u} + \alpha\boldsymbol{v}$ is uniformly chosen in H, as is the vector $\boldsymbol{x}$.*

Step 6. "If $\mathcal{B}$ can distinguish $\xi_{(J,K,R)}$ and η_K in Λ then an algorithm $\mathcal{B}'$, using $\mathcal{B}$, finds $J^{(d,M)}$ on Λ.

Step 6.1. "The algorithm $\mathcal{B}'$ can, using $\mathcal{B}$, find a sequence of $n-1$ linearly independent vectors $\boldsymbol{v} \in H$, and thus, a basis of a lattice in H, or $J^{(d,M)}$."

Consider the following description of the algorithm $\mathcal{B}'$. Take a polynomial amount of points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m'} \in K'U^n$, with $K' = n^c d_J$. Let $\boldsymbol{a}_{ij} = \boldsymbol{p}_i - \boldsymbol{p}_j$, for $1 \leqslant i < j \leqslant m'$. There are at most n^c hyperplanes H', parallel to H, that intersect $K'U^n \cap J$. Therefore, a polynomial fraction of $\boldsymbol{a}_{ij}$ will be such that both $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are in the same hyperplane H' (at least $(\frac{1}{n^c})^2\binom{m'}{2}$), and in this case $\boldsymbol{a}_{ij} \in H$. By Lemma 3.1.17, using the algorithm $\mathcal{B}$, $\mathcal{B}'$ can determine which of the differences $\boldsymbol{a}_{ij}$ are in H. If we choose m' large enough, then $\mathcal{B}'$ can find a sufficiently large amount of $\boldsymbol{a}_{ij} \in H$, e.g. $3n$, and then, w.h.p. there are $n-1$ linearly independent vectors among them, forming a basis for a lattice in H, and thus, a basis for some $J^{(d,M)}$.

Step 6.2. "Extension to general lattices".

In the previous step, we showed that if there is an algorithm $\mathcal{B}$ that can distinguish $\xi_{J,K,R}$ and η_K, then an algorithm $\mathcal{B}'$ can find H in polynomial time. However, we considered that the lattice given was presented by a basis of length at most polynomial in n, which allowed sampling of lattice points inside a smaller cube $K'U^n, K' = n^c d_J$ that was intersected by only a polynomial number of hyperplanes H', which in turn allowed a basis of H to be found in polynomial time. To remove this condition of the given lattice basis length, we can use the distinguisher $\mathcal{B}$ to find random vectors close to H, and then "grow" these vectors into long vectors still close to H. These long vectors are used to find an approximation of H, and, if this approximation is close enough, then the vector orthogonal to the approximation will be very close to $\boldsymbol{u}_H$.

In order find and then "grow" these long vectors, we run an algorithm iteratively, using $\mathcal{B}$ in every step to verify if the vector is still close enough to H. Consider the following

description of the algorithm. Let σ denote the set of points in $\mathbb{R}^n$ of a distance at most d from H. Each step of the algorithm has a starting point $\boldsymbol{s}$, which initially is the origin, and the goal is to find a point $\boldsymbol{v} \in \sigma \cap B(2\sqrt{n}d, \boldsymbol{s})$, that is, a point farther from the origin than $\boldsymbol{s}$ but still inside the region σ (note that the ball described here is around the starting point $\boldsymbol{s}$). Then $2\boldsymbol{v}$ becomes the new starting point and the process continues. Note that the $\boldsymbol{s}$ will always be within $2d$ from H. The algorithm $\mathcal{B}$ will be used in every step by sampling the distribution $\delta_{\boldsymbol{v}} = \boldsymbol{u} + \alpha\boldsymbol{v} + \boldsymbol{w}$ and verifying whether $\boldsymbol{v}$ is near H or not. If $\boldsymbol{v} \notin \sigma$, then, w.h.p. this will be detected. If $\boldsymbol{v}$ is "very close" to H, then, w.h.p. this is detected as being in σ. Lastly, if $\boldsymbol{v} \in \sigma$, then there is a probability polynomial in n^{-1} that the algorithm does not detects as if being outside σ.

To test whether $\boldsymbol{s} + \boldsymbol{v}'$, for a random $\boldsymbol{v}'$, is outside the region σ, we test each multiple of $\boldsymbol{v}'$ added to the initial starting point $\boldsymbol{s}$, i.e. $\boldsymbol{s} + \boldsymbol{v}', \boldsymbol{s} + \frac{n^{c_1}-1}{n^{c_1}}\boldsymbol{v}', \frac{n^{c_1}-2}{n^{c_1}}\boldsymbol{v}', \ldots, \frac{1}{n^{c_1}}\boldsymbol{v}'$, to see if it is near a coset $H' \neq H \in \mathcal{L}$. If any multiple of $\boldsymbol{v}'$ tests positive, then $\boldsymbol{v}'$ is discarded and the procedure is repeated for a new random $\boldsymbol{v}'$. If no test is positive and $\|\boldsymbol{s} + \boldsymbol{v}'\| \geqslant \ell$, then we set $\boldsymbol{v}_i = \boldsymbol{s} + \boldsymbol{v}'$, however, if $\|\boldsymbol{s} + \boldsymbol{v}'\| < \ell$, then we set the next starting point to $2(\boldsymbol{s} + \boldsymbol{v}')$.

Lemma 3.1.18. *There exist $c_3, c_4, c_5 > 0$ such that with probability $1-n^{-c_4}$, if the procedure to find the next starting point, as described above, is run from a starting point $\boldsymbol{s}$ within distance $2d$ of H, then within n^{c_5} iterations the procedure produces an output that, with probability $1 - n^{c_3}$, is in σ.*

Proof. Recall that $d < n^{c_6}M$ for some $c_6 > 0$, and that $R = n^3M$. Therefore, for some $c_1 > 0$, $d \leqslant n^{c_1}R$. Let $\varepsilon = n^{-c_2}$ represent the distinguishing advantage of $\mathcal{A}$.

Let e_0 be the probability that the test incorrectly handles inputs within distance αd of $H \cap V^{n-i+1}$. Let e_1 be the probability that it makes errors on inputs within distance $\frac{d}{4}n^{-c+c_2}$ of a coset of $H' \cap V^{n-i+1}$, where $H' \neq H$.

If $\boldsymbol{s} + \boldsymbol{v}'$, for some random $\boldsymbol{v}'$, is within distance αd of $H \cap V^{n-i+1}$, then $\boldsymbol{v}'$ will not be discarded, as no multiple tested yields the uniform distribution. Thus, with probability $1 - n^{c_1}e_0$, that is, the number of multiples tested (n^{c_1}) multiplied by the error probability e_0, $\boldsymbol{v}'$ will not be discarded, and it is close enough to H.

If $\boldsymbol{s} + \boldsymbol{v}'$ is outside σ, then at least one of the multiples is within distance $\frac{d}{4}n^{-c+c_2}$ of a coset of $H' \cap V^{n-i+1}$, where $H' \neq H$ and $\|\boldsymbol{v}'\| \leqslant 2\sqrt{n}d$. Since with probability e_1 it fails, then with probability $1 - e_1$ it correctly identifies the point as being far from H and it will be discarded.

Therefore, we need to show that with a sufficiently high probability, $\boldsymbol{s} + \boldsymbol{v}'$ is within αd of H, since in this case, the distinguisher correctly identifies that the vector is close to H.

Let G be an $m-1$-dimensional hyperplane passing through the origin and let $Y = 2\sqrt{n}d$ be the m-dimensional ball centered at a point $\boldsymbol{p}$ of distance at most $2d$ from G. Calculating the ratio of the volume of the ball Y to the volume of a strip parallel to a hyperplane G at distance $2d \leqslant y \leqslant 3d$ from the hyperplane that passes through $\boldsymbol{p}$, we are given

$$\text{Volume Ratio} = \frac{\int_{2d/Y}^{3d/Y} (\sqrt{1-x^2})^{m-1} dx}{\int_0^1 (\sqrt{1-x^2})^{m-1} dx},$$

The numerator is bounded below by:

$$\int_{2d/Y}^{3d/Y} (\sqrt{1-x^2})^{m-1} dx \approx \frac{d}{Y} (\sqrt{1-(3d/Y)^2})^{m-1} = \frac{1}{2\sqrt{n}} (1 - \frac{9}{4n})^{\frac{m-1}{2}} \geqslant \frac{1}{2\sqrt{n}} e^{-9/2}$$

While the denominator is bounded above by $\int_0^1 (\sqrt{1-x})^{m-1} dx \leqslant c''\sqrt{m}$, where $c'' > 0$.

To bound the probability, we need to multiply by α, since we want the vector to fall within αd of H and not d, yielding probability polynomial in n^{-1}, showing that there is a sufficient probability that we select $\boldsymbol{v}'$ such that $\boldsymbol{s} + \boldsymbol{v}'$ is in the desired region.

□

Corollary 3.1.2. *There exists c_5 such that and for all $c_6 \geqslant 0$ there is a procedure that, using $\mathcal{A}$ as an oracle, with probability at least $1 - n^{-c_5}$ generates a vector $\boldsymbol{v}$ within distance d of H and having length at least $2^{n^{c_7}}$ in time polynomial in n^{c_6}.*

The goal is then to construct an approximation $\hat{H}$ to H by finding $n-1$ mutually orthogonal long lattice vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n-1}$ of length at least ℓ and at a distance less than d of H. This is done using Corollary 3.1.2. Once we find $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{i-1}$, the next vector $\boldsymbol{v}_i$ will be searched for in the $(n-i+1)$ dimensional subspace orthogonal to all previous vectors. Hence, when searching to "grow" the vectors, we search only in the subspace orthogonal to previously found vectors, e.g. when growing the vector $\boldsymbol{v}_i$, we pick a random $\boldsymbol{v}' \in B_{n-i+1}(2\sqrt{n}d)$, where $B_{n-i+1}(2\sqrt{n}d)$ is in the space orthogonal to $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{i-1}$, such that $\|\boldsymbol{s} + \boldsymbol{v}'\| \geqslant \|\boldsymbol{s}\|$ and $\boldsymbol{s} + \boldsymbol{v}'$ is not detected as being outside σ. We then construct the approximation $\hat{H}$ that yields a vector $\hat{\boldsymbol{u}_H}$ that is very close to the desired vector $\boldsymbol{u}_H$, by choosing the appropriate length ℓ, such that the length of vectors $\|\boldsymbol{v}_i\| \geqslant \ell, i = 1, \ldots, n-1$. This is shown in [AD97], after Corollary 5.1 of Appendix 1.

Lemma 3.1.19. *Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ be a basis of the lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and $\boldsymbol{b}'_1, \ldots, \boldsymbol{b}'_n$ its dual basis, with $\|\boldsymbol{b}'_i\| \leqslant N, i = 1, \ldots, n$. Let $\boldsymbol{v} \in \mathcal{L}$ and $\boldsymbol{u} = \sum_{i=1}^n \beta_i \boldsymbol{b}_i \in \mathbb{R}^n$ and consider $\|\boldsymbol{u} - \boldsymbol{v}\| < \frac{1}{2N}$. If we round the coefficients β_i to the closest integer α_i, then $\boldsymbol{v} = \sum_{i=1}^n \alpha_i \boldsymbol{b}_i$.*

Proof. Let $\beta_i = \langle \boldsymbol{b}'_i, \boldsymbol{u} \rangle, i = 1, \ldots, n$. If $\boldsymbol{v} = \sum_{i=1}^n \gamma_i \boldsymbol{b}_i$, then $\gamma_i = \langle \boldsymbol{b}'_i, \boldsymbol{v} \rangle$. Therefore, we need to show that $|\beta_i - \gamma_i| < 1/2$.

$$|\beta_i - \gamma_i| = |\langle \boldsymbol{b}'_i, \boldsymbol{v} - \boldsymbol{u} \rangle| \leqslant \|\boldsymbol{b}'_i\| \cdot \|\boldsymbol{v} - \boldsymbol{u}\|$$

Since $\|\boldsymbol{b}'_i\| \leqslant N$ and $\|\boldsymbol{v} - \boldsymbol{u}\| < 1/2N$ implies $|\beta_i - \gamma_i| < N/2N = 1/2$ □

Using this result, we can apply this to $\mathcal{L}^*$, dual lattice of $\mathcal{L} \in_R \Lambda$. Note that $\mathcal{L} = (\mathcal{L}^*)^*$ has a basis of length at most $N = 2nd$, with $\boldsymbol{u}_H \in \mathcal{L}^*$ and $\hat{\boldsymbol{u_H}}$ close to $\boldsymbol{u}_H$. Lemma 3.1.19 says that we need $\|\boldsymbol{u}_H - \hat{\boldsymbol{u_H}}\| < 1/2N$, then if we write $\hat{\boldsymbol{u_H}}$ as a combination of the basis vectors of $\mathcal{L}^*$ and round to the nearest integer, we obtain $\boldsymbol{u}_H$. Therefore, we need to ensure that the projection in the subspace orthogonal to $\boldsymbol{u}_H$ satisfies the bounds $\sqrt{1-\boldsymbol{x}_n^2} < 1/2N$. We then have that $|\boldsymbol{x}_n| \geqslant 1 - |\frac{n\epsilon}{\alpha}| \geqslant 1 - |\frac{2dn}{\ell}|$, and choosing $\ell > 16dN^2n$ suffices.

Step 7. With $J^{(d,M)}$, we can find the direction of $\boldsymbol{u}$ (orthogonal to $J^{(d,M)}$), and the distance $d_J = \|\boldsymbol{u}\|^{-1}$ in between hyperplanes (by taking the gcd of the distances of the hyperplane H that contains $J^{(d,M)}$ to points in the lattice J). Therefore yielding the vector $\boldsymbol{u} = U_i\boldsymbol{v}$, and thus $\boldsymbol{v}$, the shortest vector of $\mathcal{L}$.

Lemma 3.1.20. *Let $J^{(d,M)} \subset \mathcal{L}$ be a sublattice of dimension $d-1$, spanned by linearly independent vectors of $\mathcal{L}$, and let $\boldsymbol{u}$ be a vector orthogonal to $J^{(d,M)}$.*

1. *The distance between adjacent parallel hyperplanes orthogonal to $\boldsymbol{u}$ is $d_J = \|\boldsymbol{u}\|^{-1}$.*
2. *The gcd of the distances from the hyperplane $H_0 = H$ to lattice points in J is d_J.*
3. *From $\boldsymbol{u}$, the shortest vector $\boldsymbol{v} \in \mathcal{L}$ can be recovered as $\boldsymbol{u} = U_i\boldsymbol{v}$.*

Proof. By definition, $J^{(d,M)} \subset \mathcal{L}$ is a $(d-1)$-dimensional sublattice. The hyperplane $H = H_0$ containing $J^{(d,M)}$ can be represented as

$$H_0 = \{\, \boldsymbol{x} \in \mathbb{R}^n : \langle \boldsymbol{u}\,,\, \boldsymbol{x}\rangle = 0 \,\},$$

where $\boldsymbol{u}$ is orthogonal to $J^{(d,M)}$. Parallel hyperplanes to H_0 are defined as

$$H_k = \{\, \boldsymbol{x} \in \mathbb{R}^n : \langle \boldsymbol{u}\,,\, \boldsymbol{x}\rangle = k \,\}, \quad k \in \mathbb{Z}.$$

The distance between consecutive hyperplanes H_k and H_{k+1} is $\|\boldsymbol{u}\|^{-1}$, as $\boldsymbol{u}$ determines the perpendicular direction.

For any lattice point $\boldsymbol{p} \in J$, the distance from $\boldsymbol{p}$ to H_0 is given by

$$d_{H_0}(\boldsymbol{p}) = \frac{\langle \boldsymbol{u}\,,\, \boldsymbol{p}\rangle}{\|\boldsymbol{u}\|}.$$

Since J is a lattice, these distances are integer multiples of $d_J = \|\boldsymbol{u}\|^{-1}$. By taking the gcd of such distances for points $\boldsymbol{p}_i \in J$, we recover d_J.

Knowing $d_J = \|\boldsymbol{u}\|^{-1}$, we can recover $\boldsymbol{u}$ as the vector orthogonal to $J^{(d,M)}$. Specifically, $\boldsymbol{u}$ lies in the orthogonal complement of $J^{(d,M)}$, and its magnitude is determined by d_J.

Since $\boldsymbol{u} = U_i\boldsymbol{v}$ for some U_i, the shortest vector $\boldsymbol{v} \in \mathcal{L}$ can be reconstructed.

□

This completes the proof of Lemma 3.1.9. ■

Proof of Theorem 3.1.4 (Main Theorem)

Proof of Theorem 3.1.4. Condition 1 of the theorem is the same condition as Lemma 3.1.5, i.e. the existence of the algorithm $\mathcal{A}$ that distinguishes encryption of 0s and 1s. Which, by Lemma 3.1.5, implies the existence of an algorithm $\mathcal{B}$ that satisfies the condition of Lemma 3.1.9, i.e. $\mathcal{B}$ distinguishes the random variables $\mathcal{H}_{\boldsymbol{u},n^{-\mathcal{D}_1},n}$ and $unif$. In turn, by Lemma 3.1.9, it implies the existence of the algorithm $\mathcal{C}$ that solves the unique shortest vector problem with the conditions outlined. Thus, we can make a single algorithm from the two mentioned and prove the condition 2 of the theorem.

■

Chapter 4

LWE, Ring-LWE and Module-LWE

4.1 Regev's LWE Cryptosystem

In his groundbreaking work, Regev [Reg05] introduced the Learning With Errors (LWE) problem and a reduction from worst-case lattice problems—specifically, the two problems GapSVP_γ (Def. 2.3.13) and SIVP_γ (Def. 2.3.15)—to the average-case LWE problem. Using this reduction, he constructed a public-key encryption scheme that is provably secure under the hardness of LWE. Consequently, the scheme's security is also based on the quantum hardness[1] of GapSVP_γ and SIVP_γ, for some $\gamma = \tilde{\mathcal{O}}(n^{3/2})$.

As an overview, Regev's approach begins by defining the Learning With Errors (LWE) distribution and related problems (Decision and Search) (See Sec. 4.1.1), which involves either distinguishing samples from a given list of equations with added errors from random samples or recovering the secret vector that satisfies these equations. The paper shows that if an adversary $\mathcal{A}$ can solve the average-case (search) LWE problem (Def. 4.1.2)—such as recovering the secret from a polynomial number of LWE samples—then a quantum reduction $\mathcal{B}$ can solve the GapSVP_γ and SIVP_γ problems on arbitrary lattices. Finally, Regev constructs a public-key encryption scheme, where the secret key corresponds to the secret vector and the public key consists of the LWE samples. Encryption is performed by encoding a bit onto an LWE ciphertext, while decryption is enabled by using the secret key to recover the plaintext despite the noise added in the ciphertext. The security of the cryptosystem is then proved by showing that if an adversary can break the encryption scheme, it can be used to solve the LWE problem, reducing the security of the scheme to the hardness of LWE.

Thus, as long as the GapSVP_γ and SIVP_γ problems, for the appropriate γ, remain computationally infeasible for quantum and classical adversaries, the cryptosystem offers strong security guarantees. This reduction provided a foundation for quantum-resistant encryption schemes that derive their security from the hardness of lattice problems, and has since been the basis for many other lattice-based cryptographic constructions.

[1] Regev's reduction in [Reg05] relies on a quantum algorithm to solve the GapSVP_γ problem. However, a later result by Peikert [Pei15] demonstrated that the reduction can be adapted to rely solely on classical algorithms.

4.1.1 LWE Problem

The LWE distribution and problems are defined as follows, with the definitions taken from [Reg05].

Definition 4.1.1 (LWE Distribution)**.** *Let n be a positive integer, $q \geqslant 2$ be some integer and χ be a distribution over $\mathbb{Z}_q$. For a secret vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, the LWE distribution $A_{\boldsymbol{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is generated by choosing $\boldsymbol{a} \leftarrow\$ \mathbb{Z}_q^n$ uniformly at random, $e \leftarrow\$ \chi$ and outputting the pair $(\boldsymbol{a}, \langle \boldsymbol{a}\,,\, \boldsymbol{s} \rangle + e \mod q)$.*

Remark. *Another way to define the LWE distribution is by using a probability density function ϕ on $\mathbb{T} = [0,1)$, with $\mathbb{T}$ being a torus. In this case, the LWE distribution is defined as follows:*

For a probability density function ϕ on $\mathbb{T}$, we denote by $A_{\boldsymbol{s},\phi}$ the distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ generated by choosing $\boldsymbol{a} \leftarrow\$ \mathbb{Z}_q^n$ uniformly at random, $e \leftarrow\$ \phi$ and outputting the pair $(\boldsymbol{a}, \langle \boldsymbol{a}\,,\, \boldsymbol{s} \rangle / q + e \mod 1)$.

From the LWE distribution defined above, we derive the two versions of the problem:

Definition 4.1.2 ((Search) $\mathrm{LWE}_{n,q,\chi,m}$ Problem)**.** *Let n be a positive integer, $q \geqslant 2$ be some integer and χ be a distribution over $\mathbb{Z}_q$. Given m independent samples from the LWE distribution $A_{\boldsymbol{s},\chi}$ for a secret $\boldsymbol{s} \in \mathbb{Z}_q^n$ and an error distribution χ over $\mathbb{Z}_q$, find the secret $\boldsymbol{s}$.*

Definition 4.1.3 (Decision $\mathrm{LWE}_{n,q,\chi,m}$ Problem)**.** *Let n be a positive integer, $q \geqslant 2$ be some integer and χ be a distribution over $\mathbb{Z}_q$. Given m independent samples of the form $(\boldsymbol{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where every sample is distributed according to either:*

1. *the LWE distribution $A_{\boldsymbol{s},\chi}$ for a secret $\boldsymbol{s} \in \mathbb{Z}_q^n$ (fixed for all samples), or*
2. *uniformly at random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$,*

decide which is the case with non-negligible advantage.

4.1.2 Main Result

For this section, we are mostly following the exposition in [Reg05], however, we will change the notation to make it consistent with the rest of the paper. In particular, we will use n to denote the dimension of the secret vector, q to denote the modulus of the LWE samples, χ to denote a discrete Gaussian distribution, and m to denote the number of LWE samples, and parametrize LWE as: $\mathrm{LWE}_{n,q,\chi,m}$, (or, $\mathrm{LWE}_{n,q,\chi}$, with the number of samples m being sometimes left out). The main result of [Reg05] is the following theorem (adapted)[2]:

Theorem 4.1.4 (Regev's Main Result)**.** *Let n, q be positive integers, for any $m = poly(n)$ and any discrete Gaussian distribution χ of parameter $\alpha q > 2\sqrt{n}$, where $0 < \alpha < 1$, if there exists an oracle that solves the average-case $LWE_{n,q,\chi,m}$, then there exists a quantum polynomial-time algorithm that solves the worst-case $GapSVP_\gamma$ and $SIVP_\gamma$ for $\gamma = \tilde{\mathcal{O}}(n^{3/2})$.*

[2]The original theorem uses the $\mathrm{LWE}_{n,q,\chi,m}$ oracle to solve the DGS problem. Then, Regev shows a reduction from DGS to GapSVP_γ and SIVP_γ.

Proof Sketch

As a very high level overview, the aim of the proof is to show that given an LWE oracle, one can devise a quantum algorithm that solves GapSVP and SIVP. In fact, the algorithm achieves this by solving the discrete Gaussian sampling problem (DGS), defined as follows:

Definition 4.1.5 (Discrete Gaussian Sampling problem (DGS))**.** *Given an n-dimensional lattice $\mathcal{L}$, and a parameter $r > \sqrt{2n} \cdot \eta_\epsilon(\mathcal{L})/\alpha$, output a sample from $D_{\mathcal{L},r}$.*

Solving the DGS for some r, close to the lower bound, yields short lattice vectors, of norm roughly $\sqrt{n}r$. For concreteness, consider $\alpha = 1/n$, $q = n^2$, and hence we obtain lattice vectors of length in the order of

$$\sqrt{n}r \approx \sqrt{n} \cdot \sqrt{2n} \cdot \eta_\epsilon(\mathcal{L}) \cdot n = \sqrt{2}n^2 \cdot \eta_\epsilon(\mathcal{L}).$$

Let $r_i := r \cdot (\alpha q/\sqrt{n})^i$. The idea of the algorithm is to start with a sufficiently large distribution, with r_{3n}, which is easy to sample from $D_{\mathcal{L},r_{3n}}$, and then iteratively, for $i = 3n, 3n-1, \ldots, 1$ with the help of the LWE oracle, transform samples from $D_{\mathcal{L},r_i}$ into samples from $D_{\mathcal{L},r_{i-1}}$, where $r_{i-1} < r_i/2$ by our choice of parameters. After these iterations, the resulting distribution will output a sample from $D_{\mathcal{L},r_0}$ for some $r_0 = r$ small enough, and thus, solving the DGS.

This iterative step itself is broken into two parts:

1. Classical Part: Using the given samples from $D_{\mathcal{L},r_i}$, and the LWE oracle, solve a Bounded Distance Decoding (BDD) problem (Def. 2.3.17) on the dual lattice $\mathcal{L}^*$ to within some distance $d \approx \alpha q/r_i$. More precisely, given any point within distance d of the lattice, the algorithm should output the unique closest lattice point.

2. Quantum Part: The BDD oracle on $\mathcal{L}^*$ is then used to prepare a quantum superposition. After applying the Quantum Fourier Transform (QFT) to this superposition and measuring it, the algorithm outputs a sample from $D_{\mathcal{L},r_{i-1}}$.

Finally, Regev shows additional reductions and explains how solving DGS leads to solving GapSVP and SIVP, thus proving the theorem.

Next follows a more detailed overview of each step in the iterative part of the reduction, with the proofs given in section 4.1.6.

Part 1 – Classical Step

Considering we have samples from $D_{\mathcal{L},r_i}$, and the $\mathrm{LWE}_{n,q,\Psi_\alpha,m}$ oracle:

1. Reduce BDD to " mod q" BDD.

(a) Iteratively "peel off" lower order digits mod q. After n iterations the point returned is within distance d/q^n of the lattice and Babai's algorithm (Babai's Nearest Plane Algorithm, [Bab86]) can be used to recover the closest lattice point.

(b) After obtaining the iteration $n+1$, backtrack to reconstruct each of the previous iterations, until the original point is recovered.

2. Verify candidate LWE solutions using Statistical Testing.

   (a) Given a candidate solution $\boldsymbol{s}'$ and samples from $A_{\boldsymbol{s},\Psi_\alpha}$, shift the samples and observe the distribution of the shifted samples.

   (b) Apply a statistical test, using periodicity of the shifted samples to verify if $\boldsymbol{s}'$ is the correct solution.

3. Handle LWE samples with lower error (Ψ_β, for $\beta \leqslant \alpha$) by adding artificial noise.

4. Reduce " mod q" BDD to LWE – Given samples from $D_{\mathcal{L},r_i}$, and the LWE oracle, solve a " mod q" BDD on $\mathcal{L}^*$ with distance $d \leqslant \alpha q/\sqrt{2}r_i$.[3]

   (a) Produce LWE samples from $D_{\mathcal{L},r_i}$ and the target vector $\boldsymbol{x}$ close to the lattice $\mathcal{L}^*$.

   (b) Use the LWE oracle to find the closest lattice point to $\boldsymbol{x}$ modulo q on the dual lattice $\mathcal{L}^*$ (Solve $\text{BDD}^{(q)}_{\mathcal{L}^*,d}$).

Part 2 – Quantum Step

Considering we have access to a BDD oracle on the dual lattice $\mathcal{L}^*$ with distance $d = \frac{\alpha q}{\sqrt{2}r_i} = \frac{\sqrt{n}}{\sqrt{2}r_{i-1}}$:

1. Create a Discrete Quantum Gaussian State for a large enough r.

   (a) Create a one dimensional quantum state $\sum_{i=-\sqrt{nr}}^{\sqrt{nr}} e^{-\pi i^2/(\sqrt{2}r)^2} |i\rangle$ and extend to n dimensions by tensoring the states.

   (b) Project it onto the lattice $\mathcal{L}$ by reducing the state modulo $\mathcal{P}(\mathcal{L})$, measuring, obtaining a superposition over a coset of the lattice $(\mathcal{L} + \boldsymbol{y})$, and then shifting the coset back to the lattice. The resulting state is exponentially close to $\sum_{\boldsymbol{x}\in\mathcal{L}} \rho_{r/\sqrt{2}}(\boldsymbol{x}) |\boldsymbol{x}\rangle$.

2. Show that a "truncated" version of the state ($\|\boldsymbol{x}\| < \sqrt{n}$) is exponentially close to the full state.

[3] From the previous steps, we have shown that it is sufficient to solve " mod q" BDD (instead of BDD) and that the LWE samples given to the $\text{LWE}_{n,q,\Psi_\alpha,m}$ oracle can have lower error Ψ_β for $\beta \leqslant \alpha$.

(a) In a scaled lattice where $\lambda_1(\mathcal{L}) > 2\sqrt{n}$, consider only $\boldsymbol{x} \in \mathcal{L}, \|\boldsymbol{x}\| \leqslant \sqrt{n}$, and show that the state is exponentially close to the full state.

3. Reduce DGS to BDD – Use the BDD oracle on $\mathcal{L}^*$ with distance $d = \frac{\sqrt{n}}{\sqrt{2}r_{i-1}}$ to generate samples from the distribution $D_{\mathcal{L},r_{i-1}}$.

   (a) Rescale the lattice $\mathcal{L}^*$ so that w.l.o.g. $d = \sqrt{n}$ and choose a sufficiently large integer $R \geqslant 2^{3n}\lambda_n(\mathcal{L}^*)$ to ensure that the Discrete Quantum Gaussian State is efficiently generated using step 1, with $r = 1/\sqrt{2}$.

   (b) Compute $\boldsymbol{x}$ mod $\mathcal{P}(\mathcal{L}^*)$ on another register and use the BDD oracle to recover the "truncated" full state $\boldsymbol{x}$. The state returned is exponentially close to

$$\sum_{\boldsymbol{x} \in \mathcal{L}^*/R\cap\mathcal{P}(\mathcal{L}^*)} \sum_{\boldsymbol{y}\in\mathcal{L}^*} \rho(\boldsymbol{x}-\boldsymbol{y})\,|\boldsymbol{x}\rangle\,.$$

   (c) Apply the QFT and measure the state to obtain, with high probability, a sample close to the scaled lattice $R\mathcal{L}$.

   (d) Use Babai's Nearest Plane Algorithm, [Bab86] to recover the closest lattice point to the sample – this can be done since the lattice is scaled by a large enough factor.

   (e) The probability of any $\boldsymbol{x}$ is proportional to $\rho(\boldsymbol{x})^2$, and thus equivalent to sampling from $D_{\mathcal{L},r_{i-1}}$ (up to scaling).

Remark. *At this point, we have showed that with samples from a very large distribution $D_{\mathcal{L},r_{3n}}$, which can be sampled efficiently, and a LWE oracle, we can obtain samples from a smaller distribution $D_{\mathcal{L},r_0}$, where $r_0 = r$, solving the DGS problem. The next step is to show how solving DGS leads to solving the GapSVP$_\gamma$ and SIVP$_\gamma$ problems, that is, reduce GapSVP$_\gamma$ and SIVP$_\gamma$ to DGS.*

Reduction: GapSVP and SIVP to DGS

1. SIVP to DGS

   (a) Call the DGS oracle enough times, and, with high probability there are n short linearly independent vectors among the returned vectors.

2. GapSVP to DGS

   (a) Define a variant of the GapCVP$_\gamma$ as GapCVP$'_\gamma$, which adds a promise where when $dist(\boldsymbol{x}, \mathcal{L}) > \gamma(n) \cdot d$, then $\lambda_1(\mathcal{L}) > \gamma(n) \cdot d$.

   (b) There is a polynomial time reduction from GapSVP$_\gamma$ to GapCVP$'_\gamma$. Hence it is enough to show a reduction from GapCVP$'_\gamma$ to DGS.

   (c) Use NP verifier from [AR04] to show a polynomial time reduction from the variant GapCVP$'_{100\sqrt{n}\cdot\gamma(n)}$ to DGS$_{\sqrt{n}\cdot\gamma(n)/\lambda_1(\mathcal{L}^*)}$.

Remark. *The previous steps show that given an LWE oracle, we can solve the DGS problem, and that solving DGS leads to solving GapSVP$_\gamma$ and SIVP$_\gamma$. This means the (Search) LWE problem is at least as hard as the GapSVP$_\gamma$ and SIVP$_\gamma$ problems. However, for most practical applications, such as the public-key encryption scheme, we are not interested in the Search version of the LWE problem, but rather the Decision version, that is, we want to distinguish between LWE samples $A_{\boldsymbol{s},\chi}$ and uniform samples from $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The next step is to show that the Decision LWE problem is also hard, and this is done by a series of reductions from the Search LWE problem to the Decision LWE problem.*

Reduction: Search to Decision LWE

Through a series of reductions, Regev shows that if there is an efficient algorithm that solves the average-case Decision LWE problem, then there is an efficient algorithm that solves the worst-case Search LWE problem. In other words, the different variants of the LWE problem are as hard as the (Search) LWE problem, and thus, if one of them is hard (Search LWE is as hard as GapSVP$_\gamma$ and SIVP$_\gamma$ problems), then all of them are hard.

1. (weak) Average-case $\implies$ (strong) Worst-case Decision LWE

   The first reduction is a worst-case to average-case reduction, where, if there is an efficient algorithm that can distinguish between LWE samples and uniform samples for a non-negligible fraction of secret keys (a weak distinguisher - average-case), then there is an efficient algorithm that can distinguish between LWE samples and uniform samples for *all* secret keys (a strong distinguisher - worst-case). For this reduction, Regev builds the strong distinguisher by randomly shifting the secret key and leveraging the weak distinguisher, repeating the process for a polynomial number of times.

2. (strong) Decision LWE $\implies$ Search LWE

   The next reduction shows that if there is an efficient algorithm that can distinguish between LWE samples and uniform samples for all secret keys (a strong distinguisher), then there is an efficient algorithm that can solve the Search LWE problem, i.e. recover the secret key from LWE samples. This is done by using the strong distinguisher as a subroutine to find the secret key one component at a time.

3. Discrete noise $\implies$ Continuous noise Search LWE

   The last reduction shows that if there is an efficient algorithm that can solve the Search LWE problem for discrete error distributions, then there is an efficient algorithm that can solve the Search LWE problem for continuous error distributions. This is step is done by simply discretizing the continuous error samples.

Overview of the Proof

The figure Fig. 4.1 represents an overview of all steps shown in [Reg05] in order to prove hardness of the LWE.

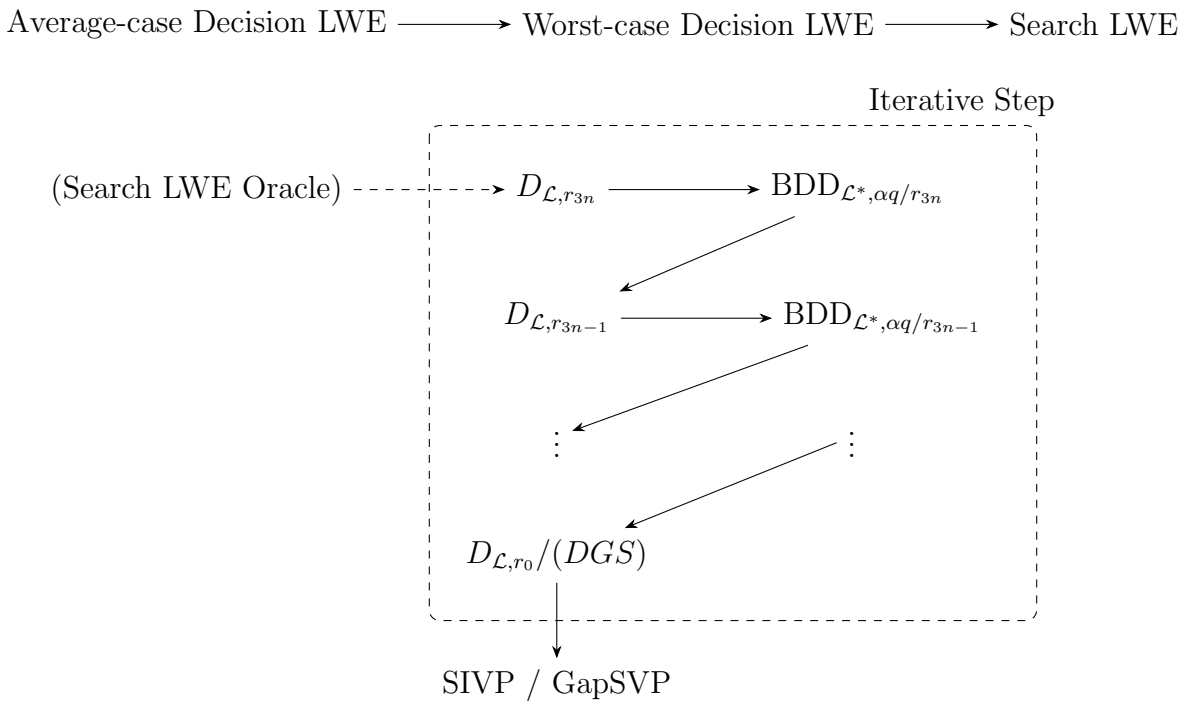


Figure 4.1: Steps for the reduction from Worst-case SIVP/GapSVP to Average-case (Decision) LWE

4.1.3 Public-key Encryption Scheme

Regev's cryptosystem is a public-key encryption scheme that is based on the hardness of the LWE problem for the given set of parameters n, q, χ, m. The scheme presented on the paper uses n as the security parameter and dimension of the secret key, $q > 2$, a prime number with $n^2 < q < 2n^2$, as the modulus and $m = (1+\epsilon)(n+1)\log q$, for some constant $\epsilon > 0$ as the number of LWE samples. The probability distribution χ is a discrete Gaussian distribution of parameter αq for some $\alpha = o(1/(\sqrt{n}\log n))$.

Private Key

The private key is a uniformly random LWE secret vector $\boldsymbol{s} \in \mathbb{Z}_q^n$.

Public Key

Sample m vectors $\boldsymbol{a}_i \in \mathbb{Z}_q^n \leftarrow\!\$\, \mathcal{U}$ independently and m values $e_i \in \mathbb{Z}_q \leftarrow\!\$\, \chi$, for $i = 1, \ldots, m$. The public key is the set of LWE samples $(\boldsymbol{a}_i, b_i := \langle \boldsymbol{a}_i\,,\, \boldsymbol{s}\rangle + e_i \bmod q)_{i=1}^m$.

Encryption

To encrypt a message $\mu \in \{\,0, 1\,\}$, take a the sum of a random set S of the samples in the public key, and output the ciphertext $\boldsymbol{c} = (\boldsymbol{a}, b) = (\sum_{i\in S} \boldsymbol{a}_i, \sum_{i\in S} b_i + \mu\lfloor q/2\rceil) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where the additions are performed $\bmod q$.

Decryption

To decrypt a ciphertext $\boldsymbol{c} = (\boldsymbol{a}, b)$, using the secret key $\boldsymbol{s}$, compute $b - \langle \boldsymbol{a}\,,\, \boldsymbol{s} \rangle \bmod q$ and output 0 if the result is closer to 0 than to $q/2$, and 1 otherwise.

$$b - \langle \boldsymbol{a}\,,\, \boldsymbol{s} \rangle \bmod q = \langle \boldsymbol{a}\,,\, \boldsymbol{s} \rangle + e + \mu\lfloor q/2 \rceil - \langle \boldsymbol{a}\,,\, \boldsymbol{s} \rangle \approx \mu\lfloor q/2 \rceil \bmod q \tag{4.1}$$

Where $e = \sum_{i \in S} \in \mathbb{Z}_q$ is small relative to q (i.e. $e < q/4$).

4.1.4 Alternative Form

The public-key encryption scheme can be expressed in an alternative form that is more convenient for later comparisons.

Private Key

The private key is done similarly to the previous one, however, instead of a sampling a uniformly random LWE secret vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, we sample from the distribution χ, that is,

$$\boldsymbol{s} \leftarrow\!\!\$\ \chi \in \mathbb{Z}_q^n$$

Now, the private key is a random **short** vector in $\mathbb{Z}_q^n$.

Public Key

The public key is the same as before, but we view the m vectors $\boldsymbol{a}_i \in \mathbb{Z}_q^n$ as a matrix $A \in \mathbb{Z}_q^{m \times n}$ and the m values $b_i \in \mathbb{Z}_q$ as a vector $\boldsymbol{b} \in \mathbb{Z}_q^m$. The public key is then pair

$$(A, \boldsymbol{b} := A\boldsymbol{s} + \boldsymbol{e}_1 \bmod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m,$$

where $\boldsymbol{e}_1 = (e_1, \ldots, e_m) \in \mathbb{Z}_q^m$ is a vector of small errors, where each $e_i \leftarrow\!\!\$\ \chi$ independently.

Encryption

To encrypt a message $\mu \in \{\,0, 1\,\}$, we initially sampled a random subset S of the rows of $\boldsymbol{a}_i$'s and b_i's. Here the process is similar, but we can express as sampling a short vector $\boldsymbol{r} \leftarrow\!\!\$\ \chi \in \mathbb{Z}_q^n$ and also two small errors: $\boldsymbol{e}_2 \leftarrow\!\!\$\ \chi \in \mathbb{Z}_q^m$ and $e_3 \leftarrow\!\!\$\ \chi \in \mathbb{Z}_q$. The ciphertext output is $(\boldsymbol{u}, v) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$, computed as follows:

$$\begin{aligned} \boldsymbol{u}^T &= \boldsymbol{r}^T A + \boldsymbol{e}_2^T \bmod q, \\ v &= \boldsymbol{r}^T \boldsymbol{b} + e_3 + \mu\lfloor q/2 \rceil \bmod q, \end{aligned}$$

where the multiplication by $\boldsymbol{r}^T$ is done in order to "select" a random subset of the rows of A and $\boldsymbol{b}$.

Decryption

To decrypt the ciphertext $(\boldsymbol{u}, v) \in \mathbb{Z}_q^m \times \mathbb{Z}_q$, using the secret key $\boldsymbol{s}$, compute:

$$\begin{aligned} v - \boldsymbol{u}^T\boldsymbol{s} &= \boldsymbol{r}^T(A\boldsymbol{s} + \boldsymbol{e}_1) + e_3 + \mu\lfloor q/2 \rceil - (\boldsymbol{r}^T A + \boldsymbol{e}_2^T)\boldsymbol{s} \bmod q \\ &= \cancel{\boldsymbol{r}^T A\boldsymbol{s}} + \boldsymbol{r}^T\boldsymbol{e}_1 + e_3 - \cancel{\boldsymbol{r}^T A\boldsymbol{s}} - \boldsymbol{e}_2^T\boldsymbol{s} + \mu\lfloor q/2 \rceil \bmod q \\ &= (\boldsymbol{r}^T\boldsymbol{e}_1 - \boldsymbol{e}_2^T\boldsymbol{s} + e_3) + \mu\lfloor q/2 \rceil \bmod q \end{aligned}$$

where, similar to the representation in the previous section, we have that the error term $(\boldsymbol{r}^T\boldsymbol{e}_1 - \boldsymbol{e}_2^T\boldsymbol{s} + e_3) \in \mathbb{Z}_q$ is small relative to q, with magnitude less than $q/4$ with high probability and the message $\mu \in \{0, 1\}$ can be recovered by checking if the result is closer to 0 or to $q/2$.

4.1.5 Analysis of the Cryptosystem

Correctness

For the decryption to be correct, we need that $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle \bmod q$ should be closer to $\mu\lfloor q/2 \rfloor$ for the correct message $\mu \in \{0, 1\}$, that is, $e + \mu\lfloor q/2 \rfloor = \sum_{i \in S} e_i + m\lfloor q/2 \rfloor \approx \mu\lfloor q/2 \rfloor$ as shown in Equation 4.1. This is true for $e < q/4$, but more specifically, we can show that the decryption is correct with high probability with the following Lemma. Let $|a|$ denote the distance of a from the nearest integer multiple of q, that is, $|a| \in (-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor)$.

Lemma 4.1.6. *Let $\delta > 0$. Assume that for any $k \in \{0, 1, \ldots, m\}$ the sum of k samples e_i from the distribution χ, $e = \sum_{i=1}^k e_i$, satisfies:*

$$\Pr_{e_i \leftarrow \$ \chi}[|e| < \lfloor q/2 \rfloor/2] > 1 - \delta$$

Then, the probability of decryption error is at most δ.

Proof. Consider first an encryption of $\mu = 0$, with $c = (\boldsymbol{a}, b) = (\sum_{i \in S} \boldsymbol{a}_i, \sum_{i \in S} b_i)$. Since $b_i = \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i$, we have that $b - \langle \boldsymbol{a}, \boldsymbol{s} \rangle = \sum_{i \in S} e_i$. Where the distribution is the sum of $|S|$ samples from χ, which, by the assumption of the Lemma, is less than $\lfloor q/2 \rfloor/2$ with probability at least $1 - \delta$. In this case, it is closer to 0 than to $\lfloor q/2 \rfloor$ and the decryption is correct. The proof for $\mu = 1$ is analogous. □

Lemma 4.1.7. *For the parameters n, q, χ, m as defined in the cryptosystem, it holds that for any $k \in \{0, 1, \ldots, m\}$ the sum of k samples e_i from the distribution χ, $e = \sum_{i=1}^k e_i$, satisfies:*

$$\Pr_{e_i \leftarrow \$ \chi}\left[|e| < \frac{\lfloor q/2 \rfloor}{2}\right] > 1 - \delta.$$

Proof. A sample from the distribution χ is produced by first drawing a real number x from a continuous noise distribution Ψ_α (which is supported on, say, $[0,1]$ and has parameter α) and then mapping it to an integer in $\mathbb{Z}_q$ by scaling and discretizing. More precisely, if $x_i \hookleftarrow\$ \Psi_\alpha$, for $i = 1, \ldots, m$, then we set

$$e_i = \lfloor q x_i \rfloor.$$

Consider the sum of k independent samples:

$$e = \sum_{i=1}^{k} e_i = \sum_{i=1}^{k} \lfloor q x_i \rfloor.$$

Discretization error. Since we are discretizing, the floor operation satisfies

$$0 \leqslant q x_i - \lfloor q x_i \rfloor < 1,$$

the difference between the discretized sum and the ideal sum is bounded by

$$\sum_{i=1}^{k} (q x_i - \lfloor q x_i \rfloor) < k.$$

That is, the total rounding error is at most k. Since $k \leq m$, and by our parameter choice we have $m = (1+\epsilon)(n+1)\log q \approx n \log q$. Since $q \leqslant 2n^2$, then $m \approx n \log n$. We can choose m such that $m \leqslant q/32$. Therefore, the error introduced by discretization is at most $q/32$.

Ideal error. Now, let us examine the ideal error sum

$$e' = \sum_{i=1}^{k} q x_i \bmod q,$$

Here, it is enough to show that this sum is bounded by $q/16$, since the total error will be the ideal sum with the added discretization error, thus, we want to show that:

$$e < \frac{q}{16} + \frac{q}{32} < \frac{\lfloor q/2 \rfloor}{2}$$

Note that working modulo q we have

$$e' = q \cdot \left(\sum_{i=1}^{k} x_i \bmod 1 \right) \bmod q.$$

Thus, the condition $e' < \frac{q}{16}$ is equivalent to

$$\left| \sum_{i=1}^{k} x_i \bmod 1 \right| < \frac{1}{16}.$$

Since each x_i is drawn from Ψ_α, the sum $\sum_{i=1}^{k} x_i$ has standard deviation approximately

$$\sigma = \sqrt{k}\,\alpha \leq \sqrt{m}\,\alpha.$$

Under our chosen parameters, with $\alpha = \frac{1}{\sqrt{n \log^2 n}}$, and since $m \approx n \log n$, then,

$$\sqrt{m}\,\alpha \approx \frac{\sqrt{n \log n}}{\sqrt{n \log^2 n}} = \frac{1}{\sqrt{\log n}}.$$

By a concentration argument, the probability that $\left|\sum_{i=1}^{k} x_i \bmod 1\right| < \frac{1}{16}$ is at least $1 - \delta$ for some negligible δ.

Returning to the ideal error, this implies that with probability at least $1 - \delta$

$$e' = q \cdot \left(\sum_{i=1}^{k} x_i \bmod 1\right) < \frac{q}{16}.$$

Total error probablity. Thus, taking into account the additional error from discretization (which is at most $q/32$), the total error is bounded by

$$e < \frac{q}{16} + \frac{q}{32} < \frac{q}{4}.$$

We then conclude that:

$$\Pr_{e_i \leftarrow \$ \chi}\left[|e| < \frac{\lfloor q/2 \rfloor}{2}\right] > 1 - \delta.$$

□

Security

To prove the security of the cryptosystem, Regev shows that any adversary that can break the encryption scheme, that is, distinguish between the encryptions of 0 and 1, can also solve the LWE problem. More precisely, breaking the scheme is to distinguish between the distribution of the LWE samples and the uniform distribution. The proof is based on the following Lemma:

Lemma 4.1.8. *For any n, q, χ, m as defined in the cryptosystem, if there exists an adversary $\mathcal{A}$ that can distinguish between the encryptions of 0 and 1 with non-negligible advantage, then there exists an adversary $\mathcal{B}$ that can distinguish between the distribution of the LWE samples $A_{\boldsymbol{s},\chi}$ and the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ with non-negligible advantage.*

Proof. Let $p_0(\mathcal{A})$ and $p_1(\mathcal{A})$ be the probabilities that $\mathcal{A}$ "accepts", outputs 1, the encryptions of 0 and 1, respectively, on input the public key $pk = (\boldsymbol{a}_i, b_i)_{i=1}^{m}$ and a ciphertext $\boldsymbol{c} = (\boldsymbol{a}, b)$. The probability is taken over the randomness in the choice of public and private keys, as well as the randomness in the encryption algorithm. Let $p_u(\mathcal{A})$ be the probability that $\mathcal{A}$ accepts the ciphertext $\boldsymbol{c}$ when it is uniformly random in $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Setting up $\mathcal{A}$'s advantage. Therefore, considering this notation, the hypothesis of the Lemma is that

$$|p_0(\mathcal{A}) - p_1(\mathcal{A})| \geqslant \frac{1}{n^c},$$

that is, the advantage of $\mathcal{A}$ is non-negligible.

Creating adversary $\mathcal{A}'$ with advantage over uniform ciphertexts. We now show that this implies the existence of another adversary $\mathcal{A}'$ for which $|p_0(\mathcal{A}') - p_u(\mathcal{A}')| \geqslant \frac{1}{2n^c}$. By our hypothesis, we have that $|p_0(\mathcal{A}) - p_1(\mathcal{A})| \geqslant \frac{1}{n^c}$, and therefore, by the triangle inequality, we have that:

$$\frac{1}{n^c} \leqslant |p_0(\mathcal{A}) - p_1(\mathcal{A})| \leqslant |p_0(\mathcal{A}) - p_u(\mathcal{A})| + |p_u(\mathcal{A}) - p_1(\mathcal{A})|$$

Therefore, either $|p_0(\mathcal{A}) - p_u(\mathcal{A})| \geqslant \frac{1}{2n^c}$ or $|p_u(\mathcal{A}) - p_1(\mathcal{A})| \geqslant \frac{1}{2n^c}$. If the former is true, then we can set $\mathcal{A}' = \mathcal{A}$, otherwise, we can set $\mathcal{A}'$ to, on input $(pk, \boldsymbol{c} = (\boldsymbol{a}, b))$, call $\mathcal{A}$ on with input $(pk, \boldsymbol{c} = (\boldsymbol{a}, b + \lfloor q/2 \rfloor))$. Thus, our assumption on $\mathcal{A}'$ is that:

$$|p_0(\mathcal{A}') - p_u(\mathcal{A}')| \geqslant \frac{1}{2n^c}$$

Averaging over Public Key generation. Now we define a probability $p_0(\boldsymbol{s})$ that $\mathcal{A}'$ accepts on input $(pk, \boldsymbol{c})$, where the public key is generated from the LWE samples $A_{\boldsymbol{s},\chi}$, and $\boldsymbol{c}$ is the encryption of 0. We also define a probability $p_u(\boldsymbol{s})$, where the public key is still generated from the LWE samples $A_{\boldsymbol{s},\chi}$, but the ciphertext is uniformly random.

From our assumption on $\mathcal{A}'$, we have that

$$|\,\mathbb{E}\,[p_0(\boldsymbol{s})] - \mathbb{E}\,[p_u(\boldsymbol{s})]\,| \geqslant \frac{1}{2n^c}. \tag{4.2}$$

Define a set $Y := \{\, \boldsymbol{s} \;\mid\; |p_0(\boldsymbol{s}) - p_u(\boldsymbol{s})| \geqslant \frac{1}{4n^c} \,\}$. Which, from the expected value of the difference, we can see that:

$$\begin{aligned}
\mathbb{E}\,[p_0(\boldsymbol{s}) - p_u(\boldsymbol{s})] &= \Pr[\boldsymbol{s} \in Y] \cdot \mathbb{E}\,[p_0(\boldsymbol{s}) - p_u(\boldsymbol{s}) \mid \boldsymbol{s} \in Y] \\
&\quad + \Pr[\boldsymbol{s} \notin Y] \cdot \mathbb{E}\,[p_0(\boldsymbol{s}) - p_u(\boldsymbol{s}) \mid \boldsymbol{s} \notin Y] \\
&\leqslant \alpha \cdot 1 + (1 - \alpha)\frac{1}{4n^c} \quad (\text{where } \alpha = \Pr[\boldsymbol{s} \in Y])
\end{aligned}$$

Where the last inequality follows from the expectation being a probability (and hence bounded by 1, $\mathbb{E}\,[p_0(\boldsymbol{s}) - p_u(\boldsymbol{s}) \mid \boldsymbol{s} \in Y] \leqslant 1$) and from the definition of Y,

$$\mathbb{E}\,[p_0(\boldsymbol{s}) - p_u(\boldsymbol{s}) \mid \boldsymbol{s} \notin Y] \leqslant \frac{1}{4n^c}.$$

Therefore, we have that:

$$\begin{aligned}\frac{1}{2n^c} \leqslant \mathbb{E}\left[p_0(\boldsymbol{s}) - p_u(\boldsymbol{s})\right] &\leqslant \alpha + (1-\alpha)\frac{1}{4n^c} \\ \frac{1}{2n^c} &\leqslant \alpha(1 - \frac{1}{4n^c}) + \frac{1}{4n^c} \\ \Rightarrow \Pr\left[\boldsymbol{s} \in Y\right] = \alpha &\geqslant \frac{1}{4n^c}\end{aligned}$$

Therefore, since the probability of $\boldsymbol{s} \in Y$ is non-negligible, it is enough to show an adversary $\mathcal{B}$ that can distinguish between the distribution of the LWE samples $A_{\boldsymbol{s},\chi}$ and the uniform distribution over $\mathbb{Z}_q^n$ for any $\boldsymbol{s} \in Y$.

Construction of $\mathcal{B}$. Lets describe the adversary $\mathcal{B}$. We are given a distribution $\mathcal{R}$ which is either $A_{\boldsymbol{s},\chi}$ or Uniform. Take m samples from $\mathcal{R}$ to generate a public key $pk = (\boldsymbol{a}_i, b_i)_{i=1}^m$. Let $p_0(pk)$ be the probability that $\mathcal{A}'$ accepts the encryption of 0 on input $(pk, \boldsymbol{c})$, where $\boldsymbol{c}$ is the encryption of 0. Let $p_u(pk)$ be the probability that $\mathcal{A}'$ accepts on input $(pk, \boldsymbol{c})$, where $\boldsymbol{c}$ is uniformly random. We then apply $\mathcal{A}'$ a polynomial number of times to get the estimates of $p_0(pk)$ and $p_u(pk)$, and output 1 if $|p_0(pk) - p_u(pk)| \geqslant \frac{1}{16n^c}$ and 0 otherwise. The idea here is that we are using $\mathcal{A}'$ as a subroutine to distinguish between the two distributions, and if the public key is generated from an LWE sample with $\boldsymbol{s} \in Y$, then $\mathcal{A}'$ has a non-negligible advantage, and would allow $\mathcal{B}$ to distinguish between the two distributions, however, if the public key is generated from a uniform distribution, then $\mathcal{A}'$ has a negligible advantage, and $\mathcal{B}$ would output 0.

$\mathcal{R}$ is Uniform: In this case, the public key is generated from a uniform distribution and $\mathcal{B}$ should output 0 with high probability. As an application of the leftover hash Lemma[4], when the public key is generated from a uniform distribution, the distance between the distributions of the encryptions of 0 and uniform random ciphertexts, is negligible. Thus, with high probability ($\geqslant 1 - 2^{-n}$):

$$|p_0(pk) - p_u(pk)| \leqslant 2^{-n} \leqslant \frac{1}{16n^c},$$

and $\mathcal{B}$ will output 0 with high probability.

$\mathcal{R}$ is $A_{\boldsymbol{s},\chi}$: In this case, the public key is generated from the LWE samples $A_{\boldsymbol{s},\chi}$, and $\mathcal{B}$ should output 1 for $\boldsymbol{s} \in Y$ with non-negligible probability. Notice that the probability $p_0(\boldsymbol{s})$ is the average over the randomness in the choice of the public key $p_0(pk)$, where pk is generated from LWE samples $A_{\boldsymbol{s},\chi}$, and, by the definition of Y, we have that $|p_0(\boldsymbol{s}) - p_u(\boldsymbol{s})| \geqslant \frac{1}{4n^c}$, therefore, using the same averaging argument as before,

[4] A special case of the Lemma is shown and proved in [Reg05] Claim 5.3

$$\Pr\left[\Pr\left[|p_0(pk) - p_u(pk)| \geqslant \frac{1}{8n^c}\right]\right] \geqslant \frac{1}{8n^c}$$

Therefore, with non-negligible probability, $\mathcal{B}$ will choose such a public key, that results in a non-negligible advantage for $\mathcal{A}'$. Since the difference is greater than $\frac{1}{16n^c}$, $\mathcal{B}$ will output 1 with non-negligible probability. □

4.1.6 Proofs

In this section, we will build upon the overview including the necessary proofs for each step. Most of proofs are from [Reg05].

Proof of Theorem 4.1.4 (Regev's Main Result)

Proof. **Input and goal.** The input to the algorithm is an n-dimensional lattice $\mathcal{L}$ and a parameter $r \geqslant \sqrt{2n} \cdot \eta_\epsilon(\mathcal{L})/\alpha$, where $0 < \alpha < 1$. The goal is to sample from the discrete Gaussian distribution $D_{\mathcal{L},r}$, and then use that to solve the GapSVPγ and SIVPγ problems. Let $r_i = r \cdot (\alpha q/\sqrt{n})^i$ for $i = 3n, 3n-1, \ldots, 1$.

Setup. The algorithm starts by producing n^c samples from the distribution $D_{\mathcal{L},r_{3n}}$ for some constant c, defined in Lemma 4.1.11 (Classical step). Since $\alpha q > 2\sqrt{n}$, then:

$$\begin{aligned}
r_{3n} &= r \cdot (\alpha q/\sqrt{n})^{3n} > r \cdot 2^{3n} \\
&> 2^{3n} \cdot \sqrt{2n} \cdot \eta_\epsilon(\mathcal{L})/\alpha && \text{(by the choice of } r\text{)} \\
&> 2^{3n} \cdot \sqrt{2n} \cdot \eta_\epsilon(\mathcal{L}) && \text{(since } \alpha < 1\text{)} \\
&> 2^{3n} \cdot \sqrt{2n} \cdot \frac{c}{\lambda_1(\mathcal{L}^*)} && \text{(by Claim 4.1.16)} \\
&> 2^{3n} \cdot \sqrt{2n} \cdot \frac{c\lambda_n(\mathcal{L})}{n} && \text{(by Thm 2.3.5 (Transference Theorem (Banaszczyk)))} \\
&> 2^{2n} \cdot \lambda_n(\mathcal{L})
\end{aligned}$$

Therefore, we can efficiently generate samples from $D_{\mathcal{L},r_{3n}}$, according to Lemma 4.1.9 (Bootstrapping).

Iterative Step. The algorithm then proceeds with the iterative step, for $i = 3n, 3n-1, \ldots, 1$. By using the n^c samples from $D_{\mathcal{L},r_i}$, the algorithm produces n^c samples from $D_{\mathcal{L},r_{i-1}}$, as shown in Lemmas 4.1.11 (Classical step) and 4.1.22 (Quantum step), for part 1 and 2, respectively. At the end of the loop, the algorithm outputs a sample from $D_{\mathcal{L},r_0} = D_{\mathcal{L},r}$.

Conclusion. Finally, according to the reductions in Lemmas 4.1.28 (GapSVP) and 4.1.25 (SIVP), the algorithm can solve the GapSVPγ and SIVPγ problems, respectively.

□

Setup

Lemma 4.1.9 (Bootstrapping)**.** *Given any n-dimensional lattice $\mathcal{L}$ and a parameter $r \geqslant 2^{2n} \cdot \lambda_n(\mathcal{L})$, there exists an efficient algorithm that outputs a sample from a distribution that is within statistical distance $2^{-\Omega(n)}$ from $D_{\mathcal{L},r}$.*

Proof. Using the LLL algorithm, we can find a basis $\boldsymbol{B} = \{ \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \}$ of the lattice $\mathcal{L}$ of length at most $2^n \cdot \lambda_n(\mathcal{L})$. Let $\mathcal{P}(\mathcal{L})$ be the fundamental parallelpiped of $\mathcal{L}$ defined by the basis B.

Sampling Algorithm. The algorithm proceeds by sampling a point $\boldsymbol{y}$ from the continuous gaussian distribution v_r, with density proportional to $\rho_r(\boldsymbol{y}) = \exp(-\pi\|\boldsymbol{y}\|^2/r^2)$. Then outputs the lattice point $\boldsymbol{x} = \boldsymbol{y} - (\boldsymbol{y} \bmod \mathcal{P}(\mathcal{L}))$, $\boldsymbol{x} \in \mathcal{L}$. Since $\|\boldsymbol{y} \bmod \mathcal{P}(\mathcal{L})\| \leqslant \text{diameter}(\mathcal{P}(\mathcal{L})) \leqslant n2^n\lambda_n(\mathcal{L})$, then rounding error is bounded.

Now we show that the distribution of $\boldsymbol{x}$ is close to $D_{\mathcal{L},r}$.

Probability under $D_{\mathcal{L},r}$. Since almost all of the probability mass of $D_{\mathcal{L},r}$ is concentrated on lattice points with norm at most $\sqrt{nr}$, we can focus on lattice points with $\|\boldsymbol{x}\| \leqslant \sqrt{nr}$. The probability under $D_{\mathcal{L},r}$ of sampling a point $\boldsymbol{x}$ is given by definition:

$$D_{\mathcal{L},r}(\boldsymbol{x}) = \frac{\rho_r(\boldsymbol{x})}{\rho_r(\mathcal{L})}$$

By Poisson Summation Formula, we have that $\rho_r(\mathcal{L}) = \det(\mathcal{L}^*) \cdot r^n \cdot \rho_{1/r}(\mathcal{L}^*) \geqslant \det(\mathcal{L}^*) \cdot r^n$, and therefore we have that:

$$D_{\mathcal{L},r}(\boldsymbol{x}) \leqslant \frac{\rho_r(\boldsymbol{x})}{\det(\mathcal{L}^*) \cdot r^n} = \det(\mathcal{L}) \cdot v_r(\boldsymbol{x}) \tag{4.3}$$

Probability under the sampling algorithm. For any lattice point, the probability of sampling a point $\boldsymbol{x}$ is given by the parallelpiped it falls in:

$$\Pr[\boldsymbol{x}] = \int_{\boldsymbol{x}+\mathcal{P}(\mathcal{L})} v_r(\boldsymbol{y}) d\boldsymbol{y}$$

By Lemma 4.1.10, we have $\rho_r(\boldsymbol{y}) \geqslant (1 - \pi(2lt + l^2)/r^2)\rho_r(\boldsymbol{x})$, where $\|\boldsymbol{y} - \boldsymbol{x}\| \leqslant l$ and $\|\boldsymbol{x}\| \leqslant t$. Therefore:

$$\rho_r(\boldsymbol{y}) \geqslant (1 - \pi(2 \cdot (n2^n\lambda_n(\mathcal{L})) \cdot \sqrt{nr} + (n2^n\lambda_n(\mathcal{L}))^2)/r^2)\rho_r(\boldsymbol{x}) \geqslant (1 - 2^{-\Omega(n)})\rho_r(\boldsymbol{x}),$$

and thus:

$$\Pr[\boldsymbol{x}] \geqslant (1 - 2^{-\Omega(n)}) \int_{\boldsymbol{x}+\mathcal{P}(\mathcal{L})} v_r(\boldsymbol{y}) d\boldsymbol{y} = (1 - 2^{-\Omega(n)}) \det(\mathcal{L}) v_r(\boldsymbol{x}) \tag{4.4}$$

Finally, by equations 4.3 and 4.4, for all lattice points $\boldsymbol{x}$ with $\|\boldsymbol{x}\| \leqslant \sqrt{nr}$, we have that:

$$D_{\mathcal{L},r}(\boldsymbol{x}) - \Pr[\boldsymbol{x}] \leqslant (1 - (1 - 2^{-\Omega(n)})) \det(\mathcal{L}) v_r(\boldsymbol{x}) = 2^{-\Omega(n)} \det(\mathcal{L}) v_r(\boldsymbol{x})$$

And therefore, the statistical distance between the two distributions is:

$$\Delta(D_{\mathcal{L},r}, \Pr) = \frac{1}{2} \sum_{\boldsymbol{x} \in \mathcal{L}} |D_{\mathcal{L},r}(\boldsymbol{x}) - \Pr[\boldsymbol{x}]| \leqslant 2^{-\Omega(n)},$$

where the last inequality follows from the probability density function

$$\sum_{\boldsymbol{x} \in \mathcal{L}} \det(\mathcal{L}) v_r(\boldsymbol{x}) = \sum_{\boldsymbol{x} \in \mathcal{L}} \int_{\boldsymbol{x} + \mathcal{P}(\mathcal{L})} v_r(\boldsymbol{y}) d\boldsymbol{y} = \int_{\mathbb{R}^n} v_r(\boldsymbol{y}) d\boldsymbol{y} = 1.$$

□

Lemma 4.1.10. *For all $r, l, t > 0$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. Let $\boldsymbol{x}$ be a lattice point with $\|\boldsymbol{x}\| \leqslant t$ and let $\boldsymbol{y}$ be a point with $\|\boldsymbol{y} - \boldsymbol{x}\| \leqslant l$, then, the density of the Gaussian distribution at $\boldsymbol{y}$:*

$$\rho_r(\boldsymbol{y}) \geqslant (1 - \pi(2lt + l^2)/r^2) \rho_r(\boldsymbol{x}).$$

Proof. Using the inequality $e^{-z} \geqslant 1 - z$,

$$\begin{aligned}
\rho_r(\boldsymbol{y}) &= \exp(-\pi \|\boldsymbol{y}\|^2 / r^2) \\
&\geqslant \exp(-\pi (\|\boldsymbol{x}\| + \|\boldsymbol{y} - \boldsymbol{x}\|)^2 / r^2) \\
&= \exp(-\pi (\|\boldsymbol{x}\|^2 + 2\|\boldsymbol{x}\|\|\boldsymbol{y} - \boldsymbol{x}\| + \|\boldsymbol{y} - \boldsymbol{x}\|^2) / r^2) \\
&= \exp(-\pi \|\boldsymbol{x}\|^2 / r^2) \exp(-\pi (2lt + l^2) / r^2) \\
&= \rho_r(\boldsymbol{x}) \exp(-\pi (2lt + l^2) / r^2) \\
&\geqslant (1 - \pi (2lt + l^2) / r^2) \rho_r(\boldsymbol{x})
\end{aligned}$$

□

Part 1 – Classical Step

Lemma 4.1.11 (Classical step)**.** *Let $\epsilon > 0$ be a negligible function of n, $q \geqslant 2 \in \mathbb{Z}$ and $\alpha \in (0, 1) \in \mathbb{R}$. Assuming we have access to an algorithm $\mathcal{A}$ that solves $LWE_{n,q,\chi,m}$ given a polynomial number of samples, then there exists a constant $c > 0$ and an efficient algorithm that, given an n-dimensional lattice $\mathcal{L}$ and a parameter $r \geqslant \sqrt{2} q \cdot \eta_\epsilon(\mathcal{L})$ and n^c samples from the distribution $D_{\mathcal{L},r}$, solves BDD on $\mathcal{L}^*$ with distance $d = \frac{\alpha q}{\sqrt{2} r}$.*

We will separate the proof into four steps following the overview in Section 4.1.2 Part 1. The complete proof of the Lemma is given at the end of this section.

Step 1. "Reduce BDD to " mod q" BDD."

Lemma 4.1.12. *There exists an efficient algorithm that, given an n-dimensional lattice $\mathcal{L}$, a number $d < \lambda_1(\mathcal{L})/2$ and an integer $q \geqslant 2$, using an oracle that returns the coefficients of the closest lattice point modulo q on $\mathcal{L}$ within distance d ($BDD^{(q)}_{\mathcal{L},d}$), solves $BDD_{\mathcal{L},d}$.*

Proof. The input is a point $\boldsymbol{x}$ within distance d of $\mathcal{L}$. Define a sequence of points $\boldsymbol{x}_1 = \boldsymbol{x}$, $\boldsymbol{x}_2, \ldots$ and, assuming we have $\boldsymbol{x}_i$, define $\boldsymbol{a}_i$ as the coefficient vector of the closest lattice point to $\boldsymbol{x}_i$. Let $\kappa_{\mathcal{L}}(\boldsymbol{x}_i)$ be the closest lattice point to $\boldsymbol{x}_i$, therefore we have that:

$$\kappa_{\mathcal{L}}(\boldsymbol{x}_i) = \mathcal{L}\boldsymbol{a}_i \Rightarrow \boldsymbol{a}_i = \mathcal{L}^{-1}\kappa_{\mathcal{L}}(\boldsymbol{x}_i).$$

On input $\boldsymbol{x}_i$, the oracle $\mathrm{BDD}^{(q)}_{\mathcal{L},d}$ returns $\boldsymbol{a}_i \bmod q$.

We also define:

$$\boldsymbol{x}_{i+1} = \frac{\boldsymbol{x}_i - \mathcal{L}(\boldsymbol{a}_i \bmod q)}{q}$$

The idea is that we are decomposing the coefficients in terms of the modulus q, since we can uniquely write every integer vector as:

$$\begin{aligned} \boldsymbol{a}_i &= (\boldsymbol{a}_i \bmod q) + q \cdot \boldsymbol{a}_{i+1} \\ \Rightarrow \mathcal{L}\boldsymbol{a}_i &= \mathcal{L}(\boldsymbol{a}_i \bmod q) + q \cdot \mathcal{L}\boldsymbol{a}_{i+1} \\ \Rightarrow \mathcal{L}\boldsymbol{a}_{i+1} &= \frac{\mathcal{L}\boldsymbol{a}_i - \mathcal{L}(\boldsymbol{a}_i \bmod q)}{q}, \end{aligned}$$

which is the closest lattice point to $\boldsymbol{x}_{i+1}$.

And, for every step of the iteration, notice that if the distance from $\boldsymbol{x}_i$ to the lattice point $\mathcal{L}\boldsymbol{a}_i$ is d', then the distance from $\boldsymbol{x}_{i+1}$ to the lattice point $\mathcal{L}\boldsymbol{a}_{i+1}$ is d'/q. Therefore, after n steps, the distance from $\boldsymbol{x}_{n+1}$ to the lattice point $\mathcal{L}\boldsymbol{a}_{n+1}$ is at most d/q^n.

At $\boldsymbol{x}_{n+1}$, we can apply Babai's Nearest Plane Algorithm, [Bab86] to find a lattice point $\mathcal{L}\boldsymbol{a}_{n+1}$ within distance $2^n \frac{d}{q^n}$ of $\boldsymbol{x}_{n+1}$. Since $2^n \frac{d}{q^n} \leqslant d < \lambda_1(\mathcal{L})/2$, then $\mathcal{L}\boldsymbol{a}_{n+1}$ is the unique closest lattice point to $\boldsymbol{x}_{n+1}$, allowing us to recover $\boldsymbol{a}_{n+1}$.

With $\boldsymbol{a}_{n+1}$ and $(\boldsymbol{a}_n \bmod q)$ – which we can recover using the $\mathrm{BDD}^{(q)}_{\mathcal{L},d}$ oracle on $\boldsymbol{x}_n$ – we can recover $\boldsymbol{a}_n = (\boldsymbol{a}_n \bmod q) + q\boldsymbol{a}_{n+1}$. Therefore, we repeat this process for every $i = n-1, n-2, \ldots, 1$ until we recover $\boldsymbol{a}_1$, and output the closest lattice point to $\boldsymbol{x}_1 = \boldsymbol{x}$ which is $\mathcal{L}\boldsymbol{a}_1$.

□

Step 2. "Verify candidate LWE solutions using Statistical Testing."

Lemma 4.1.13. *Let $q \geqslant 1$ be an integer. There exists an efficient algorithm that, given a candidate solution $\boldsymbol{s}'$ and LWE samples from $A_{\boldsymbol{s},\Psi_\alpha}$, for some $\boldsymbol{s} \in \mathbb{Z}_q^n$ and $\alpha \in (0,1)$, outputs whether $\boldsymbol{s} = \boldsymbol{s}'$ with probability exponentially close to 1.*

Proof. The idea is to shift the LWE samples by $\boldsymbol{s}'$ and observe the distribution of the shifted samples, applying a statistical test to verify if $\boldsymbol{s}'$ is the correct solution. Each LWE sample is of the form $(\boldsymbol{a}, x = \langle \boldsymbol{a}\,,\, \boldsymbol{s}\rangle/q + e \bmod 1)$, where $\boldsymbol{a} \hookleftarrow\$\ \mathbb{Z}_q^n$ and $e \hookleftarrow\$\ \Psi_\alpha$ on the torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. Here we are choosing to look at the LWE on the torus so each x is in the interval $[0,1)$ and tools from Fourier analysis can be used more easily.

We define the following algorithm: Define the shifted samples as

$$y = x - \langle \boldsymbol{a}\,,\, \boldsymbol{s}'\rangle/q \bmod 1 \in \mathbb{T}.$$

Since $x = \langle \boldsymbol{a}\,,\, \boldsymbol{s}\rangle/q + e \bmod 1$, then $y = e - \langle \boldsymbol{a}\,,\, \boldsymbol{s} - \boldsymbol{s}'\rangle/q \bmod 1$. This results in two cases:

1. If $\boldsymbol{s} = \boldsymbol{s}'$, then $y = e \bmod 1$, which is exactly the error distribution Ψ_α.
2. If $\boldsymbol{s} \neq \boldsymbol{s}'$, then $y = e - \langle \boldsymbol{a}\,,\, \boldsymbol{s} - \boldsymbol{s}'\rangle/q \bmod 1$. Since $\boldsymbol{s} - \boldsymbol{s}' \neq 0$, then $\langle \boldsymbol{a}\,,\, \boldsymbol{s} - \boldsymbol{s}'\rangle/q$ will be periodic with some period $1/k$ for some integer $k > 1$. Therefore, the distribution ξ of y will also be periodic with period $1/k$.

Take n samples from the distribution ξ then compute $z := 1/n \sum_{i=1}^{n} \cos(2\pi y_i)$. If $z > 0.02$, then output $\boldsymbol{s} = \boldsymbol{s}'$, otherwise output $\boldsymbol{s} \neq \boldsymbol{s}'$.

Lets analyze the algorithm. In order to distinguish the two cases, we can look at the periodicity of ξ by calculating the expected value of $\cos(2\pi y)$. With the idea of Fourier analysis to extract the periodicity of the distribution, where we can write:

$$\tilde{z} := \mathop{\mathbb{E}}_{y \hookleftarrow\$ \xi}[\cos(2\pi y)] = \int_0^1 \cos(2\pi y) \cdot \xi(y) dy = \mathrm{Re}\left[\int_0^1 \exp(2\pi i y)\xi(y) dy\right]$$

1. If $\boldsymbol{s} = \boldsymbol{s}'$, then $y = e \bmod 1$, and $\xi = \Psi_\alpha$. Therefore, we are taking the real part of the Fourier transform of the Gaussian distribution, which is a Gaussian function as well, with $\tilde{z} = \exp(-\pi\alpha^2)$.

   Since $\alpha < 1$, then $\tilde{z} > \exp(-\pi) \approx 0.043 > 0.04$.

2. If $\boldsymbol{s} \neq \boldsymbol{s}'$, then $y = e - \langle \boldsymbol{a}\,,\, \boldsymbol{s} - \boldsymbol{s}'\rangle/q \bmod 1$. Since $\boldsymbol{s} - \boldsymbol{s}' \neq 0$, then for at least one i, $a_i \cdot (s_i - s_i')/q \neq 0$. We also know that $a_i \in \mathbb{Z}_q$, and $s_i - s_i' \in \mathbb{Z}_q$, therefore $a_i \cdot (s_i - s_i')/q$ is periodic with period $1/k$ for some $k > 1$ and ξ is also periodic with period $1/k$.

   Therefore, the Fourier transform of ξ will be given by:

$$\int_0^1 \exp(2\pi i y)\xi(y)dy = \int_0^1 \exp(2\pi i(y+1/k))\xi(y)dy = \exp(2\pi i/k)\int_0^1 \exp(2\pi i y)\xi(y)dy,$$

where the second equality follows from the periodicity of ξ ($\xi(y) = \xi(y+1/k)$). Therefore, for any $k \geqslant 2$, this equality holds only if $\tilde{z} = 0$.

Therefore, by taking the average over n samples, we can use Chernoff bounds to show that the probability of error is exponentially small.

$$\Pr\left[|z - \tilde{z}| > 0.02\right] \leqslant 2\exp(-2 \cdot 0.02^2 \cdot n/4),$$

thus, the probability of error is exponentially small and the algorithm outputs the correct solution with probability exponentially close to 1.

□

Step 3. "Handle LWE samples with lower error (Ψ_β, for $\beta \leqslant \alpha$) by adding artificial noise."

Lemma 4.1.14. *Let $q \geqslant 2$ be an integer, and $\alpha \in (0,1)$. Assume we have access to an oracle $\mathcal{A}$ that solves $LWE_{n,q,\Psi_\alpha,m}$, for some $m = poly(n)$. Then, there exists an efficient algorithm $\mathcal{B}$ that given samples from $A_{\boldsymbol{s},\Psi_\beta}$ for some $\beta \leqslant \alpha$, outputs $\boldsymbol{s}$ with probability exponentially close to 1.*

Proof. The idea of the proof is to add the right amount of artificial noise to the samples, in order to increase the error to Ψ_α, transforming the $A_{\boldsymbol{s},\Psi_\beta}$ samples into $A_{\boldsymbol{s},\Psi_\alpha}$ samples (or something sufficiently close to it). We start with each sample of the form $(\boldsymbol{a}, x = \langle \boldsymbol{a}\,,\, \boldsymbol{s}\rangle/q + e \bmod 1)$, where $\boldsymbol{a} \hookleftarrow \mathbb{Z}_q^n$ and $e \hookleftarrow \Psi_\beta$. If we add independent noise $e' \hookleftarrow \Psi_{\sqrt{\gamma}}$ to each sample x, then the new error distribution will be $\Psi_{\sqrt{\beta^2+\gamma}}$. Therefore, by choosing γ such that $\sqrt{\beta^2+\gamma} \approx \alpha$, we can transform the samples into $A_{\boldsymbol{s},\Psi_\alpha}$ samples to match the error distribution the oracle expects.

In order to choose the right γ, define the discretized set Z of all the integer multiples of $n^{-2c}\alpha^2$ in the interval $[0, \alpha^2]$, where n^c is an upper bound on the number of samples the oracle needs. For each $\gamma \in Z$, the algorithm $\mathcal{B}$ takes n^c samples from $A_{\boldsymbol{s},\Psi_\beta}$ and adds independent noise $e' \hookleftarrow \Psi_{\sqrt{\gamma}}$ to each sample. The samples are now transformed into $A_{\boldsymbol{s},\Psi_{\sqrt{\beta^2+\gamma}}}$ samples, and the algorithm $\mathcal{A}$ can be used to solve the LWE problem and obtain a candidate solution $\boldsymbol{s}'$. This is done n times.

With the candidate $\boldsymbol{s}'$, using the algorithm shown in Lemma 4.1.13, we can verify if $\boldsymbol{s}' = \boldsymbol{s}$ (is the correct solution) with probability exponentially close to 1. If the answer is yes, then output $\boldsymbol{s}'$ as the correct solution, otherwise it continues to the next $\gamma \in Z$.

Whenever $\mathcal{B}$ outputs the a solution, since, by Lemma 4.1.13 the probability of error is exponentially small, then the solution output is correct with probability exponentially close to 1. Therefore, we need to show that at least one of the iterations $\mathcal{B}$ will output some value. Let us consider the smallest $\gamma \in Z$ such that $\gamma \geqslant \alpha^2 - \beta^2$. Since $\gamma \in Z$, then $\gamma = k \cdot n^{-2c}\alpha^2$ for some integer k and thus $\gamma \leqslant \alpha^2 - \beta^2 + n^{-2c}\alpha^2$.

Let $\alpha' = \sqrt{\beta^2 + \gamma}$ be our effective noise level, and since $\gamma \geqslant \alpha^2 - \beta^2 \Rightarrow \gamma + \beta^2 \geqslant \alpha^2$, thus $\alpha' \geqslant \alpha$. Therefore,

$$\alpha \leqslant \alpha' \leqslant \sqrt{\alpha^2 + n^{-2c}\alpha^2} \leqslant (1 + n^{-2c})\alpha.$$

By Claim 4.1.18, the statistical distance between $\Psi_{\alpha'}$ and Ψ_α is given by:

$$\Delta(\Psi_\alpha, \Psi_{\alpha'}) \leqslant 9(\frac{\alpha'}{\alpha} - 1) \leqslant 9(\frac{1 + n^{-2c}}{1} - 1) = 9n^{-2c},$$

and, for n^c samples, the total statistical distance is at most $9n^{-c}$. Therefore, for the choice of γ, the algorithm $\mathcal{A}$ will output $\boldsymbol{s} = \boldsymbol{s}'$ with probability at least $1 - 9n^{-c}/2 - 2^{-\Omega(n)} \geqslant 1/2$. By repeating the process n times, the probability $\boldsymbol{s}$ is not found in any of the calls becomes at most $(\frac{1}{2})^n = 2^{-n}$.

□

Step 4. "Main Procedure – Reduce " mod q" BDD to LWE – Given samples from $D_{\mathcal{L},r_i}$, and the LWE oracle, solve a " mod q" BDD on $\mathcal{L}^*$ with distance $d \leqslant \alpha q/\sqrt{2}r_i$."

Remark. From the previous steps, we have shown that it is sufficient to solve " mod q" BDD (instead of BDD) and that the LWE samples given to the $\text{LWE}_{n,q,\Psi_\alpha,m}$ oracle can have lower error Ψ_β for $\beta \leqslant \alpha$.

Lemma 4.1.15. *Let $q \geqslant 2$ be an integer, $\alpha \in (0,1)$ and $\epsilon > 0$ be negligible functions of n. Assuming we have access to an oracle $\mathcal{A}$ that solves $LWE_{n,q,\Psi_\beta,m}$, for all $\beta \leqslant \alpha$ and for some $m = poly(n)$, then there exists an efficient algorithm $\mathcal{B}$ that, given an n-dimensional lattice $\mathcal{L}$, a number $r > \sqrt{2}q \cdot \eta_\epsilon(\mathcal{L})$, and n^c samples from the distribution $D_{\mathcal{L},r}$, solves $BDD^{(q)}_{\mathcal{L}^*,d}$ with distance $d = \frac{\alpha q}{\sqrt{2}r}$.*

Proof. Given a vector $\boldsymbol{x}$ that is known to be within distance $d \leqslant \frac{\alpha q}{\sqrt{2}r}$ of the dual lattice $\mathcal{L}^*$, the goal of this proof is show how to transform samples from $D_{\mathcal{L},r}$ into samples from $A_{\boldsymbol{s},\Psi_\beta}$ for some $\beta \leqslant \alpha$, where the secret $\boldsymbol{s}$ is defined as $\boldsymbol{s} = (\mathcal{L}^*)^{-1} \cdot \kappa_{\mathcal{L}^*}(\boldsymbol{x}) \bmod q$, that is, the coefficients of the closest lattice point to $\boldsymbol{x}$, modulo q, with $\kappa_{\mathcal{L}^*}(\boldsymbol{x})$ being the closest lattice point to $\boldsymbol{x}$ on the dual lattice $\mathcal{L}^*$. Once we have the samples from $A_{\boldsymbol{s},\Psi_\beta}$, we can use the oracle $\mathcal{A}$ to solve the LWE problem and obtain the secret $\boldsymbol{s}$, and thus, find the closest lattice point to $\boldsymbol{x}$ modulo q on the dual lattice $\mathcal{L}^*$, as desired, solving $\text{BDD}^{(q)}_{\mathcal{L}^*,d}$.

Step 4.1. "Produce LWE samples from $D_{\mathcal{L},r}$ and the target vector $\boldsymbol{x}$ close to the lattice $\mathcal{L}^*$."

The procedure to produce the LWE samples from $D_{\mathcal{L},r}$ and the target vector $\boldsymbol{x}$ is as follows:

1. Sample a vector $\boldsymbol{v} \in \mathcal{L}$ from the distribution $D_{\mathcal{L},r}$.
2. Compute $\boldsymbol{a} = \mathcal{L}^{-1}\boldsymbol{v} \bmod q$. This is the coefficient vector of the lattice point $\boldsymbol{v}$ modulo q. $\boldsymbol{a} \in \mathbb{Z}_q^n$.
3. Compute the second component of the LWE sample as $\langle \boldsymbol{x}\,,\,\boldsymbol{v}\rangle/q + e \bmod 1$, where e is sampled from a normal distribution with standard deviation $\alpha/(2\sqrt{\pi})$.
4. Output the sample $(\boldsymbol{a}, \langle \boldsymbol{x}\,,\,\boldsymbol{v}\rangle/q + e \bmod 1)$.

In order to show that the samples produced are indeed LWE samples, we need to show that $\boldsymbol{a} \hookleftarrow\$\ \mathbb{Z}_q^n$ (is uniformly distributed) and that the second component resembles $\langle \boldsymbol{s}\,,\,\boldsymbol{a}\rangle/q + e \bmod 1$ for some secret $\boldsymbol{s} \in \mathbb{Z}_q^n$, and $e \hookleftarrow\$\ \Psi_\beta$ for some $\beta \leqslant \alpha$. Where the LWE oracle will be able to recover the secret $\boldsymbol{s}$ from the samples.

First, we claim that $\boldsymbol{a}$ is very close to uniform. By the structure of the Discrete Gaussian, the probability of sampling each $\boldsymbol{a} \in \mathbb{Z}_q^n$ is proportional to $\rho_r(q\mathcal{L} + \mathcal{L}\boldsymbol{a})$. This comes from the fact that, first, for each $\boldsymbol{v}$, the probability of sampling $\boldsymbol{v}$ is proportional to $\rho_r(\boldsymbol{v})$. Because of the modulus, since $\boldsymbol{a} = \mathcal{L}^{-1}\boldsymbol{v} \bmod q$, we can partition the lattice into cosets of the sublattice $q\mathcal{L}$, where for each vector $\boldsymbol{v}$ there is a unique coset representative $\mathcal{L}\boldsymbol{a}$ such that $\boldsymbol{v} = q\mathcal{L} + \mathcal{L}\boldsymbol{a}$. Therefore, the probability of sampling $\boldsymbol{a}$ is proportional to $\rho_r(q\mathcal{L} + \mathcal{L}\boldsymbol{a})$.

By the smoothing parameter $\eta_\epsilon(q\mathcal{L})$, since $r > \sqrt{2}q \cdot \eta_\epsilon(\mathcal{L})$ and $\eta_\epsilon(q\mathcal{L}) = q \cdot \eta_\epsilon(\mathcal{L})$, then our Discrete Gaussian has a standard deviation of larger than the smoothing parameter for the lattice $q\mathcal{L}$. With that, and by Claim 4.1.17, we have that the probability of sampling each $\boldsymbol{a}$, which is proportional to $\rho_r(q\mathcal{L} + \mathcal{L}\boldsymbol{a}) \in (\frac{r}{q})^n \det \mathcal{L}^* \cdot (1 \pm \epsilon)$, is independent of $\boldsymbol{a}$ with negligible error. Therefore, since this probability is nearly constant for all $\boldsymbol{a}$, then $\boldsymbol{a}$ is very close to uniform.

Now, we need to show that the second component of the LWE sample $(\langle \boldsymbol{x}\,,\,\boldsymbol{v}\rangle/q + e \bmod 1)$ resembles $\langle \boldsymbol{a}\,,\,\boldsymbol{s}\rangle/q + e \bmod 1$ for some secret $\boldsymbol{s} \in \mathbb{Z}_q^n$, and $e \hookleftarrow\$\ \Psi_\beta$ for some $\beta \leqslant \alpha$. We begin by decomposing $\boldsymbol{x} = \kappa_{\mathcal{L}^*}(\boldsymbol{x}) + \boldsymbol{x}'$, where $\kappa_{\mathcal{L}^*}(\boldsymbol{x})$ is the closest lattice point to $\boldsymbol{x}$ on the dual lattice $\mathcal{L}^*$, and $\boldsymbol{x}'$ is the "residual" vector, with $\|\boldsymbol{x}'\| \leqslant \frac{\alpha q}{\sqrt{2}r}$. Then the second component can be written as:

$$\langle \boldsymbol{x}\,,\,\boldsymbol{v}\rangle/q + e = \langle \boldsymbol{x}'\,,\,\boldsymbol{v}\rangle/q + \langle \kappa_{\mathcal{L}^*}(\boldsymbol{x})\,,\,\boldsymbol{v}\rangle/q + e$$

By definition on the dual lattice, $\mathcal{L}^* = (\mathcal{L}^{-1})^T$, therefore we can write:

$$\langle \kappa_{\mathcal{L}^*}(\boldsymbol{x}) \, , \, \boldsymbol{v} \rangle = \langle (\mathcal{L}^*)^{-1} \kappa_{\mathcal{L}^*}(\boldsymbol{x}) \, , \, \mathcal{L}^{-1} \boldsymbol{v} \rangle \qquad (= (\kappa_{\mathcal{L}^*}(\boldsymbol{x}))^T ((\mathcal{L}^*)^{-1})^T \cdot \mathcal{L}^{-1} \boldsymbol{v})$$

Let $\boldsymbol{s} = (\mathcal{L}^*)^{-1} \cdot \kappa_{\mathcal{L}^*}(\boldsymbol{x}) \bmod q$, and $\boldsymbol{a} = \mathcal{L}^{-1} \boldsymbol{v} \bmod q$, then:

$$\langle \kappa_{\mathcal{L}^*}(\boldsymbol{x}) \, , \, \boldsymbol{v} \rangle / q \bmod 1 = \langle (\mathcal{L}^*)^{-1} \kappa_{\mathcal{L}^*}(\boldsymbol{x}) \, , \, \mathcal{L}^{-1} \boldsymbol{v} \rangle / q \bmod 1 = \langle \boldsymbol{s} \, , \, \boldsymbol{a} \rangle / q \bmod 1$$

Now, for the remaining part $(\langle \boldsymbol{x}' \, , \, \boldsymbol{v} \rangle / q + e)$, we know that $\|\boldsymbol{x}'\| \leqslant \frac{\alpha q}{\sqrt{2}r}$, and thus the contribution of $\langle \boldsymbol{x}' \, , \, \boldsymbol{v} \rangle / q$ is small relative to the problem, $\boldsymbol{v}$ is sampled from $D_{\mathcal{L},r}$, where conditioned on $\boldsymbol{a}$, the distribution becomes $D_{q\mathcal{L}+\mathcal{L}\boldsymbol{a},r}$, and e is sampled from a normal distribution with standard deviation $\alpha/(2\sqrt{\pi}) = (\alpha/\sqrt{2})/\sqrt{2\pi}$.

Let $\boldsymbol{z} = \boldsymbol{x}'/q$, then $\|\boldsymbol{z}\| \leqslant \frac{\alpha}{\sqrt{2}r}$ and when we compare it to the smoothing parameter:

$$\frac{1}{\sqrt{1/r^2 + (\|\boldsymbol{z}\|/\alpha)^2}} \geqslant \frac{1}{\sqrt{1/r^2 + 1/2r^2}} \geqslant \frac{r}{\sqrt{2}} \geqslant \eta_\epsilon(\mathcal{L}),$$

therefore, by Corollary 4.1.1, the sum $\langle \boldsymbol{x}' \, , \, \boldsymbol{v} \rangle / q + e$ is statistically close to a distribution Ψ_β, for:

$$\beta = \sqrt{\left(\frac{r\|\boldsymbol{x}'\|}{q}\right)^2 + \frac{\alpha^2}{2}}.$$

Since $\|\boldsymbol{x}'\| \leqslant \frac{\alpha q}{\sqrt{2}r}$, then $\beta \leqslant \alpha$, as required for the LWE oracle.

Therefore, the samples produced becomes:

$$\langle \boldsymbol{x} \, , \, \boldsymbol{v} \rangle / q + e \bmod 1 = \langle \boldsymbol{s} \, , \, \boldsymbol{a} \rangle / q + (e + \langle \boldsymbol{x}' \, , \, \boldsymbol{v} \rangle / q) \bmod 1,$$

where $e + \langle \boldsymbol{x}' \, , \, \boldsymbol{v} \rangle / q$ is statistically close to Ψ_β, for $\beta \leqslant \alpha$, and the sample pair $(\boldsymbol{a}, \langle \boldsymbol{x} \, , \, \boldsymbol{v} \rangle / q + e \bmod 1)$ is within statistical distance of the LWE distribution $A_{\boldsymbol{s},\Psi_\beta}$. We run this procedure n^c times to obtain n^c samples from $A_{\boldsymbol{s},\Psi_\beta}$.

Step 4.2. "Use the LWE oracle to find the closest lattice point to $\boldsymbol{x}$ modulo q on the dual lattice $\mathcal{L}^*$ (Solve $\mathrm{BDD}^{(q)}_{\mathcal{L}^*,d}$)."

Finally, we use the LWE oracle to solve the LWE problem and obtain the secret $\boldsymbol{s} := (\mathcal{L}^*)^{-1} \cdot \kappa_{\mathcal{L}^*}(\boldsymbol{x}) \bmod q$, which is the coefficient vector of the closest lattice point to $\boldsymbol{x}$ modulo q on the dual lattice $\mathcal{L}^*$. This solves the $\mathrm{BDD}^{(q)}_{\mathcal{L}^*,d}$ problem, as desired. □

Proof of Lemma 4.1.11 – Classical Step

Proof of Lemma 4.1.11.

We begin with Lemma 4.1.15, which we showed that given an oracle that solves $\mathrm{LWE}_{n,q,\Psi_\beta,m}$, for all $\beta \leqslant \alpha$ that there is an efficient algorithm that solves $\mathrm{BDD}^{(q)}_{\mathcal{L}^*,d}$ within distance $d = \frac{\alpha q}{\sqrt{2}r_i}$ using n^c samples from $D_{\mathcal{L},r_i}$, for $r_i > \sqrt{2}q \cdot \eta_\epsilon(\mathcal{L})$.

With Lemma 4.1.14, we showed that given an oracle that solves $\mathrm{LWE}_{n,q,\Psi_\alpha,m}$, there is an efficient algorithm that solves $\mathrm{LWE}_{n,q,\Psi_\beta,m}$ for all $\beta \leqslant \alpha$, that is, if we receive samples from $A_{\boldsymbol{s},\Psi_\beta}$ for some $\beta \leqslant \alpha$, we can still use the LWE oracle for Ψ_α to solve the problem.

Also, with Lemma 4.1.12, we showed that given an oracle that solves $\mathrm{BDD}^{(q)}_{\mathcal{L}^*,d}$, there is an efficient algorithm that solves $\mathrm{BDD}_{\mathcal{L}^*,d}$, that is, the main Lemma only requires solving the " mod q" version of the BDD problem.

Therefore, by combining the three Lemmas, we have shown that given an oracle that solves $\mathrm{LWE}_{n,q,\Psi_\alpha,m}$, there is an efficient algorithm that solves $\mathrm{BDD}_{\mathcal{L}^*,d}$, for $d = \frac{\alpha q}{\sqrt{2}r_i}$, using n^c samples from $D_{\mathcal{L},r_i}$, for $r_i > \sqrt{2}q \cdot \eta_\epsilon(\mathcal{L})$. In other words, we can use n^c samples from $D_{\mathcal{L},r_i}$, along with the $\mathrm{LWE}_{n,q,\Psi_\alpha,m}$ oracle, to solve the $\mathrm{BDD}_{\mathcal{L}^*,\frac{\alpha q}{\sqrt{2}r_i}}$ problem, as desired. □

Auxiliary Claims. Next, we will establish the claims and Corollary used in the proof of the Theorem 4.1.4 and Lemmas 4.1.14 and 4.1.15. The proofs are from [Reg05].

Smoothing Parameter lower bound – Claim 2.13, [Reg05]

Claim 4.1.16. *For any lattice $\mathcal{L}$ and any $\epsilon > 0$,*

$$\eta_\epsilon(\mathcal{L}) \geqslant \sqrt{\frac{\ln 1/\epsilon}{\pi}} \cdot \frac{1}{\lambda_1(\mathcal{L}^*)} \geqslant \sqrt{\frac{\ln 1/\epsilon}{\pi}} \cdot \frac{\lambda_n(\mathcal{L})}{n}.$$

In particular, for any $\epsilon(n) = o(1)$ and any constant $c > 0$,

$$\eta_\epsilon(\mathcal{L}) > \frac{c}{\lambda_1(\mathcal{L}^*)} \geqslant \frac{c\lambda_n(\mathcal{L})}{n}.$$

Proof. Let $\boldsymbol{v} \in \mathcal{L}^*$ be a vector of length $\lambda_1(\mathcal{L}^*)$ and let $s = \eta_\epsilon(\mathcal{L})$. Then, since $\mathcal{L}^*$ contains $\boldsymbol{v}$, we can write

$$\epsilon = \rho_{1/s}(\mathcal{L}^* \setminus \{\,0\,\}) \geqslant \rho_{1/s}(\boldsymbol{v}) = e^{-\pi(s\lambda_1(\mathcal{L}^*))^2},$$

where, by the Gaussian function, we have that $\rho_{1/s}(\boldsymbol{v}) = e^{-\pi\|\boldsymbol{v}\|^2/(1/s)^s} = e^{-\pi(s\lambda_1(\mathcal{L}^*))^2}$. If we solve for s, we have:

$$\epsilon \geqslant e^{-\pi(s\lambda_1(\mathcal{L}^*))^2}$$
$$\Rightarrow \ln \epsilon \geqslant -\pi(s\lambda_1(\mathcal{L}^*))^2$$
$$\Rightarrow \eta_\epsilon(\mathcal{L}) = s \geqslant \sqrt{\frac{\ln 1/\epsilon}{\pi}} \cdot \frac{1}{\lambda_1(\mathcal{L}^*)}.$$

The second inequality follows from Theorem 2.3.5, where $\lambda_1(\mathcal{L}^*)\lambda_n(\mathcal{L}) \leqslant n$, thus giving us the complete claim.

□

Discrete Gaussian above Smoothing Parameter invariant under shifts – Claim 3.8, [Reg05]

The following claim shows that when working with Discrete Gaussians of width r larger than the smoothing parameter, it is invariant under shifts by lattice points.

Claim 4.1.17. *For any lattice $\mathcal{L}$, $\boldsymbol{c} \in \mathbb{R}^n$, $\epsilon > 0$ and $r \geqslant \eta_\epsilon(\mathcal{L})$,*

$$\rho_r(\mathcal{L} + \boldsymbol{c}) \in r^n \det \mathcal{L}^* \cdot (1 \pm \epsilon).$$

Proof. By definition, we have that: $\rho_r(\mathcal{L} + \boldsymbol{c}) = \sum_{\boldsymbol{x}\in\mathcal{L}} \rho_r(\boldsymbol{x} + \boldsymbol{c})$. If we shift the Gaussian by $\boldsymbol{c}$, we can write that:

$$\rho_r(\mathcal{L} + \boldsymbol{c}) = \sum_{\boldsymbol{x}\in\mathcal{L}} \rho_r(\boldsymbol{x} + \boldsymbol{c}) = \sum_{\boldsymbol{x}\in\mathcal{L}} \rho_{r,-\boldsymbol{c}}(\boldsymbol{x}),$$

meaning we are working with the Gaussian centered at $-\boldsymbol{c}$. By applying the Poisson Summation Formula ($f(\mathcal{L}) = \det \mathcal{L}^* \hat{f}(\mathcal{L}^*)$), we can write:

$$\sum_{\boldsymbol{x}\in\mathcal{L}} \rho_{r,-\boldsymbol{c}}(\boldsymbol{x}) = \det \mathcal{L}^* \sum_{\boldsymbol{y}\in\mathcal{L}^*} \hat{\rho}_{r,-\boldsymbol{c}}(\boldsymbol{y})$$

With the Fourier Transform, the sum becomes:

$$\sum_{\boldsymbol{y}\in\mathcal{L}^*} \hat{\rho}_{r,-\boldsymbol{c}}(\boldsymbol{y}) = r^n \sum_{\boldsymbol{y}\in\mathcal{L}^*} e^{2\pi i\langle \boldsymbol{c},\boldsymbol{y}\rangle} \rho_{1/r}(\boldsymbol{y}),$$

where, by the smoothing parameter, we have that $\rho_{1/r}(\mathcal{L}^*\setminus\{\,0\,\}) = \sum_{\boldsymbol{y}\in\mathcal{L}^*\setminus\{\,0\,\}} \rho_{1/r}(\boldsymbol{y}) \leqslant \epsilon$. Therefore, the previous sum becomes:

$$\sum_{\boldsymbol{y}\in\mathcal{L}^*} e^{2\pi i\langle \boldsymbol{c},\boldsymbol{y}\rangle} \rho_{1/r}(\boldsymbol{y}) = \rho_{1/r}(0) + \sum_{\boldsymbol{y}\in\mathcal{L}^*\setminus\{\,0\,\}} e^{2\pi i\langle \boldsymbol{c},\boldsymbol{y}\rangle} \rho_{1/r}(\boldsymbol{y}) \approx 1 + \epsilon,$$

Therefore, going back to the original expression, results in:

$$\rho_r(\mathcal{L} + \boldsymbol{c}) = r^n \det \mathcal{L}^* (1 \pm \epsilon)$$

□

Ψ_β, with a small change in β – Claim 2.2, [Reg05]

Claim 4.1.18. *For any $0 < \alpha < \beta \leqslant 2\alpha$,*

$$\Delta(\Psi_\alpha, \Psi_\beta) \leqslant 9\left(\frac{\beta}{\alpha} - 1\right).$$

Proof. We want to show that the statistical distance between two normal variables, with standard deviations $\alpha/\sqrt{2\pi}$ and $\beta/\sqrt{2\pi}$, is at most $9(\beta/\alpha - 1)$. When we apply a function, in this case, modulo 1 in order to transform the variables into Ψ_α and Ψ_β, the statistical distance can not increase, and therefore the claim holds. Without loss of generality, assume $\alpha = 1$ and $\beta = 1 + \epsilon$ for some $\epsilon \in (0, 1]$. The statistical distance between the two normal distributions is given by:

$$\begin{aligned}
&\int_{\mathbb{R}} \left| e^{-\pi x^2} - \frac{1}{1+\epsilon} e^{-\pi x^2/(1+\epsilon)^2} \right| dx \\
&\leqslant \int_{\mathbb{R}} \left| e^{-\pi x^2} - e^{-\pi x^2/(1+\epsilon)^2} \right| dx + \int_{\mathbb{R}} \left| \left(1 - \frac{1}{1+\epsilon}\right) e^{-\pi x^2/(1+\epsilon)^2} \right| dx \\
&= \int_{\mathbb{R}} \left| e^{-\pi x^2} - e^{-\pi x^2/(1+\epsilon)^2} \right| dx + \epsilon \cdot \int_{\mathbb{R}} \left| \left(\frac{1}{1+\epsilon}\right) e^{-\pi x^2/(1+\epsilon)^2} \right| dx \\
&= \int_{\mathbb{R}} \left| e^{-\pi(1-1/(1+\epsilon)^2)x^2} - 1 \right| e^{-\pi x^2/(1+\epsilon)^2} dx + \epsilon
\end{aligned}$$

Since $1 - z \leqslant e^{-z}$ for all $z \geqslant 0$, then the integral is upper bounded by:

$$\left| e^{-\pi(1-1/(1+\epsilon)^2)x^2} - 1 \right| \leqslant \pi(1 - 1/(1+\epsilon)^2)x^2 \leqslant 2\pi\epsilon x^2,$$

Hence the integral is upper bounded by:

$$2\pi\epsilon \int_{\mathbb{R}} x^2 e^{-\pi x^2/(1+\epsilon)^2} dx + \epsilon = \epsilon(1+\epsilon)^3 + \epsilon \leqslant 9\epsilon.$$

□

Adding continuous Gaussian noise to discrete Gaussian Distribution – Claim 3.9 and Corollary 3.10, [Reg05]

The following claim is used to show that when adding noise (continuous gaussian of width s) to a discrete gaussian distribution of width r, where both are greater than the smoothing parameter, the resulting distribution is statistically close to a continuous gaussian of width $\sqrt{r^2 + s^2}$.

Claim 4.1.19. *Let $\mathcal{L}$ be an n-dimensional lattice, let $\boldsymbol{u} \in \mathbb{R}^n$ be any vector, let $r, s > 0$ be any two real numbers, and let $t := \sqrt{r^2 + s^2}$. Assume that $rs/t = 1/\sqrt{1/r^2 + 1/s^2} \geqslant \eta_\epsilon(\mathcal{L})$ for some $\epsilon < 1/2$. Consider the continuous distribution Y on $\mathbb{R}^n$ obtained by sampling from $D_{\mathcal{L}+\boldsymbol{u},r}$ and adding independent noise from v_s. Then the statistical distance between Y and v_t is at most 4ϵ.*

This claim is used to establish the following Corollary. The complete proof can be found in (Claim 3.9, [Reg05]).

Corollary 4.1.1. *Let $\mathcal{L}$ be an n-dimensional lattice, let $\boldsymbol{z}, \boldsymbol{u} \in \mathbb{R}^n$ be any two vectors, let $r, \alpha > 0$ be two real numbers. Assume that $1/\sqrt{1/r^2 + (\|\boldsymbol{z}\|/\alpha)^2} \geqslant \eta_\epsilon(\mathcal{L})$ for some $\epsilon < 1/2$. Then, the distribution of $\langle \boldsymbol{z}, \boldsymbol{v} \rangle + e$ where $\boldsymbol{v}$ is sampled from $D_{\mathcal{L}+\boldsymbol{y},r}$ and e is sampled from a normal distribution with standard deviation $\alpha/\sqrt{2\pi}$ is within statistical distance 4ϵ of a normal distribution with standard deviation $\sqrt{(r\|\boldsymbol{z}\|)^2 + \alpha^2}/\sqrt{2\pi}$. In particular, since the statistical distance cannot be increased by applying a function, the distribution of $\langle \boldsymbol{z}, \boldsymbol{v} \rangle + e \bmod 1$ is within statistical distance 4ϵ of $\Psi_{\sqrt{(r\|\boldsymbol{z}\|)^2+\alpha^2}}$.*

Proof. The proof begins with the observation that the distribution of $\langle \boldsymbol{z}, \boldsymbol{v} \rangle + e$ is the same as the distribution of $\langle \boldsymbol{z}, \boldsymbol{v} + \boldsymbol{h} \rangle$, where $\boldsymbol{h}$ is distributed as a continuous Gaussian $v_{\alpha/\|\boldsymbol{z}\|}$. By Claim 4.1.19, the statistical distance between the distribution of $\langle \boldsymbol{z}, \boldsymbol{v} + \boldsymbol{h} \rangle$ and the continious Gaussian $v_{\sqrt{r^2+(\alpha/\|\boldsymbol{z}\|)^2}}$ is at most 4ϵ. Taking th inner product with $\boldsymbol{z}$, leads to a normal distribution with mean 0 and standard deviation $\sqrt{(r\|\boldsymbol{z}\|)^2 + \alpha^2}/\sqrt{2\pi}$. □

Part 2 – Quantum Step

We will separate the proof into three steps following the overview in Section 4.1.2 Part 2. The complete proof is given at the end of this section, Lemma 4.1.22.

Step 1. "Create a Discrete Quantum Gaussian State for a large enough r."[5]

Lemma 4.1.20. *There exists an efficient quantum algorithm that, given an n-dimensional lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ and a real number $r > 2^{2n}\lambda_n(\mathcal{L})$, outputs a state that is within ℓ_2-distance $2^{-\Omega(n)}$ of the normalized state corresponding to*

$$\sum_{\boldsymbol{x} \in \mathcal{L}} \sqrt{\rho_r(\boldsymbol{x})} \, |\boldsymbol{x}\rangle = \sum_{\boldsymbol{x} \in \mathcal{L}} \rho_{\sqrt{2}r}(\boldsymbol{x}) \, |\boldsymbol{x}\rangle \, .$$

Proof. We begin by creating a "one-dimensional Gaussian state", that is, a quantum state over the integers with amplitudes proportional to a Gaussian distribution. The state is given by:

$$\sum_{x=-\sqrt{nr}}^{\sqrt{nr}} e^{-\pi x^2/(\sqrt{2}r)^2} \, |x\rangle \, ,$$

[5] This step is similar to the setup, or bootstrapping, step shown in Lemma 4.1.9.

which can be created efficiently using a technique shown by [GR02].[6]

This procedure is then repeated n times, and the n-dimensional state is obtained by tensoring the states. This yields the following state:

$$\sum_{\boldsymbol{x}\in\{-\sqrt{nr},\ldots,\sqrt{nr}\}^n} \rho_{r/\sqrt{2}}(\boldsymbol{x})\,|\boldsymbol{x}\rangle\,.$$

This state is only defined over the box $[-\sqrt{nr},\sqrt{nr}]^n$, however, by a tail bound shown by [Ban93], this state is within ℓ_2-distance $2^{-\Omega(n)}$ of the state

$$\sum_{\boldsymbol{x}\in\mathbb{Z}^n} \rho_{r/\sqrt{2}}(\boldsymbol{x})\,|\boldsymbol{x}\rangle\,.$$

Next, using the LLL algorithm, we compute a reduced basis and project the state onto the lattice $\mathcal{L}$ by reducing the state modulo $\mathcal{P}(\mathcal{L})$ in a new register ($\boldsymbol{x} \bmod \mathcal{P}(\mathcal{L})$). This vector $\boldsymbol{x}$ is decomposed as $\boldsymbol{x} = \boldsymbol{x}' + \boldsymbol{y}$, where $\boldsymbol{x}' \in \mathcal{L}$ and $\boldsymbol{y} \in \mathcal{P}(\mathcal{L})$. After measuring this register, we obtain a vector $\boldsymbol{y}$, and because of the LLL reduction, $\|\boldsymbol{y}\| \leqslant \operatorname{diam}(\mathcal{P}(\mathcal{L})) \leqslant n \cdot 2^n \lambda_n(\mathcal{L})$, and the state collapses into a superposition over the coset $\mathcal{L} + \boldsymbol{y}$, resulting in the state

$$\sum_{\boldsymbol{x}\in\mathcal{L}+\boldsymbol{y}} \rho_{r/\sqrt{2}}(\boldsymbol{x})\,|\boldsymbol{x}\rangle\,.$$

Finally, in order to obtain the desired state, we need to shift the coset back to the lattice $\mathcal{L}$. This can be done by subtracting the value measured $\boldsymbol{y}$ from the register, resulting in the state

$$\sum_{\boldsymbol{x}\in\mathcal{L}+\boldsymbol{y}} \rho_{r/\sqrt{2}}(\boldsymbol{x})\,|\boldsymbol{x}-\boldsymbol{y}\rangle \Rightarrow \sum_{\boldsymbol{x}\in\mathcal{L}} \rho_{r/\sqrt{2}}(\boldsymbol{x}+\boldsymbol{y})\,|\boldsymbol{x}\rangle\,.$$

This state is essentially the same as the desired state, with an added $\boldsymbol{y}$ term in the amplitude. However, again by the tail bound, for our desired state, almost all of the norm is concentrated on points with norm at most $\sqrt{n}r$, where the amplitude squared is $\rho_r(\boldsymbol{x})/\rho_r(\mathcal{L})$. Therefore, by Poisson Summation Formula ($\rho_r(\mathcal{L}) = r^n \det(\mathcal{L}^*) \cdot \rho_{1/r}(\mathcal{L}^*) \geqslant r^n \det(\mathcal{L}^*)$), the amplitude is bounded by

$$\frac{\rho_r(\boldsymbol{x})}{\rho_r(\mathcal{L})} \leqslant \frac{\rho_r(\boldsymbol{x})}{r^n \det(\mathcal{L}^*)} = \det(\mathcal{L}) \cdot v_r(\boldsymbol{x}).$$

In the state generated by the procedure, the amplitude is given by $\rho_r(\boldsymbol{x}+\boldsymbol{y})/\rho_r(\mathcal{L}+\boldsymbol{y})$. By Poisson Summation Formula, we have that

[6] The technique is based on reducing the problem to being able to compute partial sums of the Gaussian function with good precision, i.e. $\sum_{x=a}^{b} e^{-\pi x^2/r^2}$, for any $a, b \in \{-\sqrt{nr},\ldots,\sqrt{nr}\}$.

$$\begin{aligned}\rho_r(\mathcal{L}+\boldsymbol{y}) &= r^n \det(\mathcal{L}^*)\cdot \sum_{\boldsymbol{z}\in\mathcal{L}^*} e^{2\pi i\langle \boldsymbol{y},\boldsymbol{z}\rangle}\rho_{1/r}(\boldsymbol{z}) \\ &\leqslant r^n \det(\mathcal{L}^*)\cdot \left(\overbrace{e^{2\pi i\langle \boldsymbol{y},0\rangle}\rho_{1/r}(0)}^{1} + \sum_{\boldsymbol{z}\in\mathcal{L}^*\setminus\{0\}} e^{2\pi i\langle \boldsymbol{y},\boldsymbol{z}\rangle}\rho_{1/r}(\boldsymbol{z})\right) \\ &= r^n \det(\mathcal{L}^*)\cdot (1+2^{-\Omega(n)})\end{aligned}$$

Where the last inequality follows from the definition of $\rho_{1/r}(\boldsymbol{z}) = e^{-\pi\frac{\|\boldsymbol{z}\|^2}{(1/r)^2}}$, the fact that $\|\boldsymbol{z}\| \geqslant \lambda_1(\mathcal{L}^*) \geqslant 1/\lambda_n(\mathcal{L})$ (By Transference Theorem (Banaszczyk)) and that $r > 2^{2n}\lambda_n(\mathcal{L})$, and thus $\|\boldsymbol{z}\|^2 r^2 \geqslant 2^{4n}$. For the numerator, we have that, by Lemma 4.1.10, $\rho_r(\boldsymbol{x}+\boldsymbol{y}) \geqslant (1-2^{-\Omega(n)})\rho_r(\boldsymbol{x})$, and thus, putting everything together, we have that:

$$\frac{\rho_r(\boldsymbol{x}+\boldsymbol{y})}{\rho_r(\mathcal{L}+\boldsymbol{y})} \geqslant \frac{(1-2^{-\Omega(n)})\rho_r(\boldsymbol{x})}{r^n\det(\mathcal{L}^*)\cdot(1+2^{-\Omega(n)})} \geqslant \frac{(1-2^{-\Omega(n)})\rho_r(\boldsymbol{x})}{r^n\det(\mathcal{L}^*)} = (1+2^{-\Omega(n)})\cdot\det(\mathcal{L})\cdot v_r(\boldsymbol{x})$$

Therefore, the state generated is within ℓ_2-distance $2^{-\Omega(n)}$ of the desired state.

□

Step 2. "Show that a "truncated" version of the state ($\|\boldsymbol{x}\| < \sqrt{n}$) is exponentially close to the full state."

Claim 4.1.21. *Let $R \geqslant 1$ be an integer and let $\mathcal{L}$ be an n-dimensional lattice such that $\lambda_1(\mathcal{L}) > 2\sqrt{n}$. Then the ℓ_2-distance between the normalized quantum states corresponding to*

$$|\vartheta_1\rangle = \sum_{\boldsymbol{x}\in\mathcal{L}/R, \|\boldsymbol{x}\|<\sqrt{n}} \rho(\boldsymbol{x})\,|\boldsymbol{x} \bmod \mathcal{P}(\mathcal{L})\rangle\,,$$

and

$$|\vartheta_2\rangle = \sum_{\boldsymbol{x}\in\mathcal{L}/R} \rho(\boldsymbol{x})\,|\boldsymbol{x} \bmod \mathcal{P}(\mathcal{L})\rangle = \sum_{\boldsymbol{x}\in\mathcal{L}/R\cap\mathcal{P}(\mathcal{L})}\sum_{\boldsymbol{y}\in\mathcal{L}} \rho(\boldsymbol{x}-\boldsymbol{y})\,|\boldsymbol{x}\rangle\,,$$

is at most $2^{-\Omega(n)}$.

Proof. Let $\mathcal{L}/R$ be the lattice $\mathcal{L}$ scaled down by a factor R. The goal is to prove that the truncated state over only $\boldsymbol{x} \in \mathcal{L}/R$ with $\|\boldsymbol{x}\| < \sqrt{n}$ is exponentially close to the full state. The proof is based on the ℓ_2 norm of the states, where we show the weight of the contribution of the "tail". Let $Z := \||\vartheta_1\rangle\|_2$.

In order to estimate Z, we can use the fact that $\lambda_1(\mathcal{L}) > 2\sqrt{n}$, and thus, each $\boldsymbol{x} \in \mathcal{L}/R$ with $\|\boldsymbol{x}\| < \sqrt{n}$ only appears once in the sum. Therefore, we can write:

$$Z^2 = \sum_{\boldsymbol{x}\in\mathcal{L}/R, \|\boldsymbol{x}\|<\sqrt{n}} \rho(\boldsymbol{x})^2 = \rho(\sqrt{2}\mathcal{L}/R \cap \sqrt{2n}B_n),$$

which, by [Ban93], Lemma 1.5, which says that $\rho_r(\mathcal{L} \setminus \sqrt{n}rB_n) \leqslant 2^{-2n} \cdot \rho_r(\mathcal{L})$, where $\mathcal{L} \setminus \sqrt{n}rB_n$ denotes the set of points in $\mathcal{L}$ with norm larger than $\sqrt{n}r$, we can write:

$$(1 - 2^{-2n}) \cdot \rho(\sqrt{2}\mathcal{L}/R) \leqslant Z^2 \leqslant \rho(\sqrt{2}\mathcal{L}/R)$$

Therefore we can express the difference between the norms as:

$$\begin{aligned}
\| |\vartheta_1\rangle - |\vartheta_2\rangle \|_2 &= \| |\vartheta_1\rangle - |\vartheta_1\rangle \|_1 && \text{(by the monotonicity of norms)} \\
&= \sum_{\boldsymbol{x} \in \mathcal{L}/R, \|\boldsymbol{x}\| \geqslant \sqrt{n}} \rho(\boldsymbol{x}) \\
&\leqslant 2^{-2n} \cdot \rho(\mathcal{L}/R) && \text{(by [Ban93], Lemma 1.5)} \\
&\leqslant 2^{-2n} \cdot 2^{n/2} \cdot \rho(\sqrt{2}\mathcal{L}/R) && \text{(by [Ban93], Lemma 1.4)} \\
&\leqslant 2^{-n} \cdot \rho(\sqrt{2}\mathcal{L}/R)
\end{aligned}$$

Since $Z \approx \rho(\sqrt{2}\mathcal{L}/R)$, then the relative error is at most 2^{-n}, and thus the state $|\vartheta_1\rangle$, which "removes the tail" of the distribution (only has $\|\boldsymbol{x}\| < \sqrt{n}$), is exponentially close to the full state $|\vartheta_2\rangle$. □

Step 3. "Main Procedure – Reduce DGS to BDD – Use the BDD oracle on $\mathcal{L}^*$ with distance $d = \frac{\sqrt{n}}{\sqrt{2}r_{i-1}}$ to generate samples from the distribution $D_{\mathcal{L},r_{i-1}}$."

Lemma 4.1.22 (Quantum step)**.** *There exists an efficient quantum algorithm that, given an n-dimensional lattice $\mathcal{L}$ and a number $d < \lambda_1(\mathcal{L}^*)/2$, and an oracle that solves the $\mathrm{BDD}_{\mathcal{L}^*,d}$ problem, outputs a sample from the distribution $D_{\mathcal{L},\sqrt{n}/(\sqrt{2}d)}$.*

Proof. Given an n-dimensional lattice $\mathcal{L}$, and a distance parameter $d < \lambda_1(\mathcal{L}^*)/2$, we want to show that there exists an efficient quantum algorithm that, given an oracle that solves the $\mathrm{BDD}_{\mathcal{L}^*,d}$ problem, outputs a sample from the distribution $D_{\mathcal{L},\sqrt{n}/(\sqrt{2}d)}$.

Step 3.1. "Rescale the lattice $\mathcal{L}^*$ so that w.l.o.g. $d = \sqrt{n}$ and choose a sufficiently large integer $R \geqslant 2^{3n}\lambda_n(\mathcal{L}^*)$ to ensure that the Discrete Quantum Gaussian State is efficiently generated using step 1, with $r = 1/\sqrt{2}$."

We begin by rescaling the lattice $\mathcal{L}^*$ so that we can assume without loss of generality that $d = \sqrt{n}$. This is done to make the proof easier to analyze. We also choose a parameter $R \geqslant 2^{3n}\lambda_n(\mathcal{L}^*)$ to ensure that the Discrete Quantum Gaussian State is efficiently generated using step 1 (Lemma 4.1.20). Step 1 shows that we can generate a state that is exponentially close to the desired state $\sum_{\boldsymbol{x} \in \mathcal{L}} \rho_{r/\sqrt{2}}(\boldsymbol{x}) |\boldsymbol{x}\rangle$, however we need $r > 2^{2n}\lambda_n(\mathcal{L})$, and thus, when we rescale the lattice using R, and assume $r = 1/\sqrt{2}$, we satisfy the condition.

Therefore, we can use Lemma 4.1.20 on the rescaled dual lattice $\mathcal{L}^*/R$, with parameter $r = 1/\sqrt{2}$, to obtain the state (or sufficiently close to it)

$$\sum_{\boldsymbol{x}\in\mathcal{L}^*/R} \rho_{\sqrt{2}\cdot\frac{1}{\sqrt{2}}}(\boldsymbol{x})\,|\boldsymbol{x}\rangle = \sum_{\boldsymbol{x}\in\mathcal{L}^*/R} \rho(\boldsymbol{x})\,|\boldsymbol{x}\rangle\,.$$

Then, by [Ban93] (Lemma 1.5), almost all of the mass of the Gaussian is concentrated on points with norm at most $\sqrt{n}$, and thus we can write:

$$\sum_{\boldsymbol{x}\in\mathcal{L}^*/R,\|\boldsymbol{x}\|<\sqrt{n}} \rho(\boldsymbol{x})\,|\boldsymbol{x}\rangle\,.$$

Step 3.2 "Compute $\boldsymbol{x}$ mod $\mathcal{P}(\mathcal{L}^*)$ on another register and use the BDD oracle to recover the "truncated" full state $\boldsymbol{x}$. The state returned is exponentially close to $\sum_{\boldsymbol{x}\in\mathcal{L}^*/R\cap\mathcal{P}(\mathcal{L}^*)}\sum_{\boldsymbol{y}\in\mathcal{L}^*}\rho(\boldsymbol{x}-\boldsymbol{y})\,|\boldsymbol{x}\rangle$."

We continue by computing in another register $\boldsymbol{x}$ mod $\mathcal{P}(\mathcal{L}^*)$. This produces the state

$$\sum_{\boldsymbol{x}\in\mathcal{L}^*/R,\|\boldsymbol{x}\|<\sqrt{n}} \rho(\boldsymbol{x})\,|\boldsymbol{x},\boldsymbol{x} \bmod \mathcal{P}(\mathcal{L}^*)\rangle\,.$$

Now, with the $\mathrm{BDD}_{\mathcal{L}^*,d}$ oracle, we can "uncompute" the value $\boldsymbol{x}$ from the second register. If we decompose $\boldsymbol{x}$ as $\boldsymbol{x}=\boldsymbol{x}'+\boldsymbol{y}$, where $\boldsymbol{x}'\in\mathcal{L}^*$ and $\boldsymbol{y}\in\mathcal{P}(\mathcal{L}^*)$, then the second register contains $\boldsymbol{y}\in\mathcal{P}(\mathcal{L}^*)$, and when we apply the oracle, we obtain the closest vector, which in this case is $\boldsymbol{x}'\in\mathcal{L}^*$. The $\boldsymbol{x}'$ obtained is unique since $\|\boldsymbol{x}\|<\sqrt{n}$ and we know that $\lambda_1(\mathcal{L}^*)>2\sqrt{n}$. Therefore, once we obtain $\boldsymbol{x}'$, we can combine it with the second register $\boldsymbol{y}$ to recover the state $\boldsymbol{x}=\boldsymbol{x}'+\boldsymbol{y}$. That is, we can use this information to cancel out the first register, and leave only the second register, which contains $\boldsymbol{y}=\boldsymbol{x}$ mod $\mathcal{P}(\mathcal{L}^*)$. The state is now:

$$\sum_{\boldsymbol{x}\in\mathcal{L}^*/R,\|\boldsymbol{x}\|<\sqrt{n}} \rho(\boldsymbol{x})\,|\boldsymbol{x} \bmod \mathcal{P}(\mathcal{L}^*)\rangle\,.$$

However, by Claim 4.1.21 (Step 2), this "truncated" state is exponentially close to the full state, and thus we can write the state as:

$$\sum_{\boldsymbol{x}\in\mathcal{L}^*/R} \rho(\boldsymbol{x})\,|\boldsymbol{x} \bmod \mathcal{P}(\mathcal{L}^*)\rangle = \sum_{\boldsymbol{x}\in\mathcal{L}^*/R\cap\mathcal{P}(\mathcal{L}^*)}\sum_{\boldsymbol{y}\in\mathcal{L}^*} \rho(\boldsymbol{x}-\boldsymbol{y})\,|\boldsymbol{x}\rangle\,.$$

Step 3.2 "Apply the QFT and measure the state to obtain, with high probability, a sample close to the scaled lattice $R\mathcal{L}$."

Before applying the QFT, we can rewrite the state using a natural mapping between $\mathcal{L}^*/R\cap\mathcal{P}(\mathcal{L}^*)$ and $\mathbb{Z}_R^n$. Each vector of the set $\mathcal{L}^*/R\cap\mathcal{P}(\mathcal{L}^*)$ can be written as $\boldsymbol{x}=B^*/R\cdot\boldsymbol{s}$, where $\boldsymbol{s}\in\mathbb{Z}^n$, however, since we are working with the $\mathcal{P}(\mathcal{L}^*)$, then each $\boldsymbol{x}$ has to be written as $\boldsymbol{x}=B^*\cdot\boldsymbol{z}$, with $\boldsymbol{z}\in[0,1)^n$ by definition of the parallelepiped. Therefore, we can write:

$$\boldsymbol{x} = B^*/R \cdot \boldsymbol{s} = B^* \cdot \boldsymbol{z} \Rightarrow 0 \leqslant s_i/R < 1 \Rightarrow s_i \in [0, R)^n, i = 1, 2, \dots n$$

Therefore, there is a natural mapping between $\mathcal{L}^*/R \cap \mathcal{P}(\mathcal{L}^*)$ and $\mathbb{Z}_R^n$, and hence we can rewrite the state as:

$$\sum_{\boldsymbol{s} \in \mathbb{Z}_R^n} \sum_{\boldsymbol{z} \in \mathbb{Z}^n} \rho(\mathcal{L}^* \boldsymbol{s}/R - \mathcal{L}^* \boldsymbol{z}) \left| \boldsymbol{s} \right\rangle .$$

We can now apply the quantum Fourier Transform (QFT) on $\mathbb{Z}_R^n$, resulting in a state where the amplitude of each $|\boldsymbol{t}\rangle$ is proportional to:

$$\begin{aligned}
&\sum_{\boldsymbol{s} \in \mathbb{Z}_R^n} \sum_{\boldsymbol{z} \in \mathbb{Z}^n} \rho(\mathcal{L}^* \boldsymbol{s}/R - \mathcal{L}^* \boldsymbol{z}) \cdot \exp\left(2\pi i \cdot \langle \boldsymbol{s}\,,\, \boldsymbol{t} \rangle / R\right) \\
&= \sum_{\boldsymbol{s}' \in \mathbb{Z}^n} \rho(\mathcal{L}^* \boldsymbol{s}'/R) \cdot \exp\left(2\pi i \cdot \langle \boldsymbol{s}'\,,\, \boldsymbol{t} \rangle / R\right) && \text{(where } \boldsymbol{s}' = \boldsymbol{s}/R - \boldsymbol{z}\text{, and } \boldsymbol{s}' \in \mathbb{Z}^n\text{)} \\
&= \sum_{\boldsymbol{x} \in \mathcal{L}^*/R} \rho(\boldsymbol{x}) \cdot \exp\left(2\pi i \cdot \langle (\mathcal{L}^*)^{-1} \boldsymbol{x}\,,\, \boldsymbol{t} \rangle\right) && \text{(where } \boldsymbol{x} = \mathcal{L}^* \boldsymbol{s}'/R \Rightarrow \boldsymbol{s}' = R(\mathcal{L}^*)^{-1} \boldsymbol{x}\text{)} \\
&= \sum_{\boldsymbol{x} \in \mathcal{L}^*/R} \rho(\boldsymbol{x}) \cdot \exp\left(2\pi i \cdot \langle \boldsymbol{x}\,,\, \mathcal{L} \boldsymbol{t} \rangle\right) \\
&= \sum_{\boldsymbol{x} \in \mathcal{L}^*/R} \rho(\boldsymbol{x} - \mathcal{L} \boldsymbol{t}) \\
&= \det(R\mathcal{L}) \sum_{\boldsymbol{y} \in R\mathcal{L}} \rho(\boldsymbol{y} - \mathcal{L} \boldsymbol{t})
\end{aligned}$$

where the last step is by using Poisson Summation Formula ($\rho(\mathcal{L}') = \det(\mathcal{L}'^*)\hat{\rho}(\mathcal{L}'^*)$), with $\mathcal{L}' = \mathcal{L}^*/R \Rightarrow \mathcal{L}'^* = R\mathcal{L}$.

Finally, the state generated is:

$$\sum_{\boldsymbol{x} \in \mathcal{P}(R\mathcal{L}) \cap \mathcal{L}} \sum_{\boldsymbol{y} \in R\mathcal{L}} \rho(\boldsymbol{y} - \boldsymbol{x}) \left| \boldsymbol{x} \right\rangle$$

Also, since the shortest vector $\lambda_1(R\mathcal{L}) = R\lambda_1(\mathcal{L}) \geqslant R/\lambda_n(\mathcal{L}^*) \geqslant 2^{3n}$ by the choice of R, then by Claim 4.1.21, then the state is exponentially close to

$$\sum_{\boldsymbol{x} \in \mathcal{L}, \|\boldsymbol{x}\| < \sqrt{n}} \rho(\boldsymbol{x}) \left| \boldsymbol{x} \bmod \mathcal{P}(R\mathcal{L}) \right\rangle .$$

Finally, we measure this state, obtaining $\boldsymbol{x} \bmod \mathcal{P}(R\mathcal{L})$ for some vector $\boldsymbol{x}$, such that $\|\boldsymbol{x}\| < \sqrt{n}$.

Step 3.3 "Use Babai's algorithm to recover the closest lattice point to the sample – this can be done since the lattice is scaled by a large enough factor."

Hence, the recovered value is within $\sqrt{n}$ of the lattice $R\mathcal{L}$, and that $\lambda_1(R\mathcal{L})$ is very large ($\geqslant 2^{3n}$), and therefore we can use Babai's Nearest Plane Algorithm, [Bab86], to recover the closest lattice point $\boldsymbol{x}$.

Step 3.4 "The probability of any $\boldsymbol{x}$ is proportional to $\rho(\boldsymbol{x})^2$, and thus equivalent to sampling from $D_{\mathcal{L},r_{i-1}}$ (up to scaling)."

The probability of recovering each $\boldsymbol{x} \in \mathcal{L}$ is proportional to $\rho(\boldsymbol{x})^2 = \rho_{1/\sqrt{2}}(\boldsymbol{x})$,

and this is exactly the weight function of the Discrete Gaussian we are interested in sampling from, $D_{\mathcal{L},1/\sqrt{2}}$.

Finally, by [Ban93], Lemma 1.5, almost all of the mass of $D_{\mathcal{L},1/\sqrt{2}}$ is concentrated on points with norm less than $\sqrt{n}$, then the probability of the outputs is exponentially close to the desired distribution $D_{\mathcal{L},1/\sqrt{2}}$. □

GapSVP and SIVP to DGS

SIVP The reduction to SIVP is based on the idea that if we call the DGS oracle enough times, then with high probability, there are n short linearly independent vectors among them. The following Lemma shows that the probability of sampling a vector from $D_{\mathcal{L},r}$ in a new subspace is non-negligible.

Lemma 4.1.23. *Let $\mathcal{L}$ be an n-dimensional lattice and let $r \geqslant \sqrt{2}\eta_\epsilon(\mathcal{L})$, with $\epsilon \leqslant 1/10$. Then, for any subspace H of dimension at most $n-1$, the probability of sampling a vector $\boldsymbol{x}$ from $D_{\mathcal{L},r}$ such that $\boldsymbol{x} \notin H$ is at least $1/10$.*

Proof. Assume, without loss of generality, that H is orthogonal to the first coordinate axis, i.e. $x_1 = 0$. Then, we want to show that the probability of sampling a vector $\boldsymbol{x}$ from $D_{\mathcal{L},r}$ such that $x_1 \neq 0$ is at least $1/10$.

The expected value of the weight distribution along the first coordinate is given by:

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim D_{\mathcal{L},r}}\left[e^{-\pi x_1^2/r^2}\right] = \sum_{\boldsymbol{x}\in\mathcal{L}} e^{-\pi x_1^2/r^2}\cdot\frac{\rho_r(\boldsymbol{x})}{\rho_r(\mathcal{L})} \tag{4.5}$$

$$= \frac{1}{\rho_r(\mathcal{L})}\cdot\sum_{\boldsymbol{x}\in\mathcal{L}} e^{-\pi x_1^2/r^2}\cdot e^{-\pi\|\boldsymbol{x}\|^2/r^2} \tag{4.6}$$

$$= \frac{1}{\rho_r(\mathcal{L})}\cdot\sum_{\boldsymbol{x}\in\mathcal{L}} e^{-\pi x_1^2/r^2}\cdot e^{-\pi x_1^2/r^2}\cdot e^{-\pi x_2^2/r^2}\cdots e^{-\pi x_n^2/r^2} \tag{4.7}$$

$$= \frac{1}{\rho_r(\mathcal{L})}\cdot\sum_{\boldsymbol{x}\in\mathcal{L}} e^{-\pi(\sqrt{2}x_1)^2/r^2}\cdot e^{-\pi x_2^2/r^2}\cdots e^{-\pi x_n^2/r^2} \tag{4.8}$$

$$= \frac{\det(\mathcal{L}^*)}{\rho_r(\mathcal{L})}\cdot\sum_{\boldsymbol{y}\in\mathcal{L}^*}\left(\frac{r}{\sqrt{2}}\right) e^{-2\pi(y_1)^2/(\sqrt{2}r)^2}\cdot re^{-\pi y_2^2/r^2}\cdots re^{-\pi y_n^2/r^2} \tag{4.9}$$

$$= \frac{\det(\mathcal{L}^*)r^n}{\sqrt{2}\rho_r(\mathcal{L})}\cdot\sum_{\boldsymbol{y}\in\mathcal{L}^*} e^{-\pi(y_1)^2/(\sqrt{2}r)^2}\cdot e^{-\pi y_2^2/r^2}\cdots e^{-\pi y_n^2/r^2} \tag{4.10}$$

$$\leqslant \frac{\det(\mathcal{L}^*)r^n}{\sqrt{2}\rho_r(\mathcal{L})}\cdot\sum_{\boldsymbol{y}\in\mathcal{L}^*} e^{-\pi(y_1)^2/(\sqrt{2}r)^2}\cdot e^{-\pi y_2^2/(\sqrt{2}r)^2}\cdots e^{-\pi y_n^2/(\sqrt{2}r)^2} \tag{4.11}$$

$$= \frac{\det(\mathcal{L}^*)r^n}{\sqrt{2}\rho_r(\mathcal{L})}\cdot\rho_{\sqrt{2}/r}(\mathcal{L}^*) \tag{4.12}$$

$$\leqslant \frac{\det(\mathcal{L}^*)r^n}{\sqrt{2}\cdot\det(\mathcal{L}^*)r^n}\cdot\rho_{\sqrt{2}/r}(\mathcal{L}^*) \tag{4.13}$$

$$\leqslant \frac{1}{\sqrt{2}}\cdot(1+\epsilon) \tag{4.14}$$

$$\tag{4.15}$$

Where steps 4.9 and 4.13 follow from Poisson Summation Formula ($\rho_r(\mathcal{L}) = \det(\mathcal{L}^*)\cdot r^n\cdot\rho_{1/r}(\mathcal{L}^*)$), transforming the sum into a sum over the dual lattice $\mathcal{L}^*$ and since $\rho_r(\mathcal{L}) = \det(\mathcal{L}^*)\cdot r^n\cdot\rho_{1/r}(\mathcal{L}^*) \geqslant \det(\mathcal{L}^*)\cdot r^n$. The last inequality follows from the smoothing parameter, where $r \geqslant \sqrt{2}\eta_\epsilon(\mathcal{L})$, and thus, $\frac{r}{\sqrt{2}} \geqslant \eta_\epsilon(\mathcal{L}^*)$, and $\rho_{1/\eta_\epsilon(\mathcal{L}^*)}(\mathcal{L}^*) \leqslant (1+\epsilon)$. Therefore, for $\epsilon \leqslant 1/10$, we have that $\frac{1+\epsilon}{\sqrt{2}} \leqslant 0.9$.

□

Before the main Lemma, we can derive the following Corollary, in which, by this previous Lemma, regardless of what happened in the previous step, the probability of sampling a vector that is not in the subspace of the previous vectors is at least 1/10, and thus, the probability that n^2 sampled vectors does not contain a linearly independent set of n vectors is exponentially small.

Corollary 4.1.2. *Let $\mathcal{L}$ be an n-dimensional lattice and let $r \geqslant \sqrt{2}\eta_\epsilon(\mathcal{L})$, with $\epsilon \leqslant 1/10$. Then, the probability that a set of n^2 vectors chosen independently from $D_{\mathcal{L},r}$ does not contain a linearly independent set of n vectors is exponentially small.*

For the main Lemma, we define a generalization of the SIVP, which is defined in 2.3.15. The definition here is the same as the one presented in [Reg05].

Definition 4.1.24. *An instance of $GIVP^{\varphi}_{\gamma}$ is given by an n-dimensional lattice $\mathcal{L}$. The goal is to output a set of n linearly independent lattice vectors of length at most $\gamma(n) \cdot \varphi(\mathcal{L})$.*

Remark. When the function φ is chosen to be $\varphi = \lambda_n$, then the problem is equivalent to the SIVP problem.

Lemma 4.1.25 (SIVP)**.** *For any $\epsilon \leqslant 1/10$ and any $\varphi(\mathcal{L}) \geqslant \sqrt{2}\eta_\epsilon(\mathcal{L})$, there is a polynomial time reduction from $GIVP_{2\sqrt{n}\varphi}$ to DGS_{φ}.*

Remark. The Lemma gives the reduction from GIVP to DGS, which, by taking $\varphi = \sqrt{2n} \cdot \eta_\epsilon(\mathcal{L})/\alpha$, from the assumptions of the Theorem 4.1.4, (Sampling $D_{\mathcal{L},r}$, with $r \geqslant \sqrt{2}\eta_\epsilon(\mathcal{L})/\alpha$), gives us an algorithm for $\mathrm{GIVP}_{2\sqrt{2n}\cdot\eta_\epsilon(\mathcal{L})/\alpha}$. Finally, since $\eta_\epsilon(\mathcal{L}) \leqslant \sqrt{\omega(\log n)} \cdot \lambda_n(\mathcal{L})$, this also solves $\mathrm{SIVP}_{\tilde{O}(n/\alpha)}$.

Proof. The idea of the proof is to call the DGS oracle enough times, and with high probability, there are n short linearly independent vectors among them. The issue with creating a proof as it is, is that we don't know if the function φ is efficiently computable, and thus we don't know how to choose right value r to give to the DGS oracle. Thus, the solution is to use many different values of r and take the shortest set of n linearly independent vectors.

The algorithm is the following. With input $\mathcal{L}$, we apply the LLL ([LLL82]) algorithm to obtain a reduced basis $S = \{ \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \}$ of $\mathcal{L}$, such that $\|\boldsymbol{b}_i\| \leqslant 2^n \lambda_n(\mathcal{L})$. Let $(\tilde{\lambda}_n(\mathcal{L}))$ denote the longest vector obtained, hence, we have that $\lambda_n(\mathcal{L}) \leqslant \tilde{\lambda}_n(\mathcal{L}) \leqslant 2^n \lambda_n(\mathcal{L})$. Then, for each $i = 1, 2, \ldots, 2n$, we call the DGS oracle (n^2) times with the pair $(\mathcal{L}, r_i)$, where $r_i = (\tilde{\lambda}_n(\mathcal{L})) \cdot 2^{-i}$ and let S_i be the set of vectors sampled. Look for a set of n linearly independent vectors in S_i and output the shortest set of n linearly independent vectors among all S_i.

For correctness, consider the following cases. First, if $\varphi(\mathcal{L}) \geqslant \tilde{\lambda}_n(\mathcal{L})$, then the set S is already shorter than $2\sqrt{n}\varphi(\mathcal{L})$, and thus, we can output S as the solution. Otherwise, if $\varphi(\mathcal{L}) < \tilde{\lambda}_n(\mathcal{L})$, then there exists and $i \in \{ 1, 2, \ldots, 2n \}$ such that $\varphi(\mathcal{L}) \leqslant r_i \leqslant 2\varphi(\mathcal{L})$ (by the Pigeon Hole Principle, and since $\varphi(\mathcal{L}) \geqslant \sqrt{2}\eta_\epsilon(\mathcal{L})$ and $\eta_\epsilon(\mathcal{L})$ has a lower bound $\eta_\epsilon(\mathcal{L}) \geqslant c\lambda_n(\mathcal{L})/n$), and thus, by the previous Lemma, the set S_i contains with high probability n linearly independent vectors of length at most $2\sqrt{n}\varphi(\mathcal{L})$, and hence the algorithm outputs a set of n linearly independent vectors of length at most $2\sqrt{n}\varphi(\mathcal{L})$. □

GapSVP The reduction to GapSVP is actually in two steps, using a reduction from GapSVP to GapCVP′ from [GMSS99] and using an NP verifier for coGapCVP, from [AR04], that is, an efficient verifier that solves the complement of GapCVP′, and if $\mathrm{dist}(\boldsymbol{x}, \mathcal{L}) \leqslant d$, the verifier rejects, and if $\mathrm{dist}(\boldsymbol{x}, \mathcal{L}) > 100\sqrt{n}d$, the verifier accepts. The definitions for decision version of CVP (Def. 2.3.16), GapCVP and GapCVP' are the following:

Definition 4.1.26. *An instance of GapCVP$_\gamma$ is given by an n-dimensional lattice $\mathcal{L}$, a vector $\boldsymbol{t}$ and a distance parameter $d > 0$. If dist$(\boldsymbol{t}, \mathcal{L}) \leqslant d$, the answer is "yes" and if dist$(\boldsymbol{t}, \mathcal{L}) > \gamma d$, the answer is "no".*

Definition 4.1.27. *An instance of GapCVP$'_\gamma$ is given by an n-dimensional lattice $\mathcal{L}$, a vector $\boldsymbol{t}$ and a distance parameter $d > 0$. If dist$(\boldsymbol{t}, \mathcal{L}) \leqslant d$, the answer is "yes". The answer is "no" if dist$(\boldsymbol{t}, \mathcal{L}) > \gamma d$ **and** $\lambda_1(\mathcal{L}) > \gamma d$.*

Lemma 4.1.28 (GapSVP)**.** *For any $\gamma \geqslant 1$, there is a polynomial time reduction from GapCVP$'_{100\sqrt{n}\gamma}$ to DGS$_{\sqrt{n}\gamma/\lambda_1(\mathcal{L}^*)}$.*

Proof. The proof is based on the algorithm by NP verifier by [AR04], which, on input an n-dimensional lattice $\mathcal{L}$, a vector $\boldsymbol{t}$, a distance parameter d and a sequence of vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N \in \mathcal{L}^*$, for some $N = \text{poly}(n)$, always **rejects** when dist$(\boldsymbol{t}, \mathcal{L}) \leqslant d$, and **accepts** when dist$(\boldsymbol{t}, \mathcal{L}) > 100\sqrt{n}d$ and $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$ are chosen from $D_{\mathcal{L}^*, 1/(100d)}$, with probability close to one.

For the reduction, on input an n-dimensional lattice $\mathcal{L}$, a vector $\boldsymbol{t}$ and a distance parameter $d > 0$, we call the DGS oracle N times with the lattice $\mathcal{L}^*$ and the value $r = 1/(100d)$ to obtain the vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$. We then apply the algorithm by [AR04] described above, and if the algorithm **rejects**, we output "yes".

For correctness, consider the case when dist$(\boldsymbol{t}, \mathcal{L}) \leqslant d$. In this case, the algorithm always rejects, independently of the vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$. In the other case, when dist$(\boldsymbol{t}, \mathcal{L}) > 100\sqrt{n}d$, since $r = 1/(100d) > \sqrt{n}\gamma/\lambda_1(\mathcal{L}^*)$ (by the promise of GapCVP$'_\gamma$, we have that $\lambda_1(\mathcal{L}^*) > 100\sqrt{n}\gamma d$), then the samples $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$ are valid samples from $D_{\mathcal{L}^*, 1/(100d)}$ and thus, the algorithm accepts with probability close to one. □

Reduction: Search to Decision LWE

Finally, we can reduce the search LWE problem to the (average-case) decision LWE by a series of steps. The reduction shows that the variants of the LWE problem are as hard as the Search LWE, and, a (weak) distinguisher for the average-case decision LWE problem, that can distinguish between the LWE samples and uniform samples for a non-negligible fraction of secret keys, can be used to actually find the secret key. The proofs follow [Reg05].

Step 1. "(weak) Average-case $\implies$ (strong) Worst-case Decision LWE"

Lemma 4.1.29 (Average-case to Worst-case Decision LWE)**.** *Let $n, q \geqslant 1$ be integers and let χ be a distribution over $\mathbb{Z}_q$. Assume we have access to a distinguisher $\mathcal{A}$ that can distinguish between the LWE samples $A_{\boldsymbol{s},\chi}$ and uniform samples U for a non-negligible fraction of all possible secret keys $\boldsymbol{s} \in \mathbb{Z}_q^n$. Then, there exists a polynomial time algorithm $\mathcal{B}$ that, for all $\boldsymbol{s}$, accepts with probability exponentially close to 1 on inputs from $A_{\boldsymbol{s},\chi}$ and rejects with probability exponentially close to 1 on inputs from U.*

Proof. The high-level idea of this proof is that we can use the (weak) distinguisher $\mathcal{A}$, that can distinguish for a non-negligible fraction of secret keys, by applying a transformation to

the LWE samples, randomly shifting the secret key $\boldsymbol{s}$ until we find a key that is accepted by $\mathcal{A}$ with high probability.

We begin by defining the following transformation $f_{\boldsymbol{t}} : \mathbb{Z}_q^n \times \mathbb{Z}_q \to \mathbb{Z}_q^n \times \mathbb{Z}_q$, for any $\boldsymbol{t} \in \mathbb{Z}_q^n$:

$$f_{\boldsymbol{t}}(\boldsymbol{a}, b) = (\boldsymbol{a}, b + \langle \boldsymbol{a}\,,\, \boldsymbol{t} \rangle)$$

Since, by definition of $A_{\boldsymbol{s},\chi}$ (with samples of the form $(\boldsymbol{a}, b)$, where $b = \langle \boldsymbol{a}\,,\, \boldsymbol{s} \rangle + e$), the function $f_{\boldsymbol{t}}$ transforms a sample $(\boldsymbol{a}, \langle \boldsymbol{a}\,,\, \boldsymbol{s} \rangle + e)$ into the sample $(\boldsymbol{a}, \langle \boldsymbol{a}\,,\, \boldsymbol{s} \rangle + e + \langle \boldsymbol{a}\,,\, \boldsymbol{t} \rangle = \langle \boldsymbol{a}\,,\, \boldsymbol{s} + \boldsymbol{t} \rangle + e)$, and therefore, the distribution $A_{\boldsymbol{s},\chi}$ is transformed into the distribution $A_{\boldsymbol{s}+\boldsymbol{t},\chi}$. The function also transforms the uniform distribution U into itself.

Assume that, for a non-negligible fraction n^{-c_1} of all possible secret keys $\boldsymbol{s}$, the distinguisher $\mathcal{A}$ has an acceptance probability that differs on inputs from $A_{\boldsymbol{s},\chi}$ and U by at least n^{-c_2}. Consider the distinguisher $\mathcal{B}$ defined as follows:

Let R denote the unknown input distribution, either $A_{\boldsymbol{s},\chi}$ or U, and let $\mathcal{B}$ be the algorithm that, on input $(\boldsymbol{a}, b)$, chooses a random $\boldsymbol{t} \in \mathbb{Z}_q^n$ and estimates the acceptance probability of $\mathcal{A}$ on U and $f_{\boldsymbol{t}}(R)$, by calling $\mathcal{A}$ for $O(n^{2c_2+1})$ times on each distribution. Repeat this process for $O(n^{c_1+1})$ times.

If, in any of the iterations, the estimated acceptance probabilities of $\mathcal{A}$ on U and $f_{\boldsymbol{t}}(R)$ differ by at least $n^{-c_2}/2$, then $\mathcal{B}$ accepts the input, since this means that one of the $\boldsymbol{t}$ values is such that $\mathcal{A}$ accepts with non-negligible probability on $f_{\boldsymbol{t}}(R)$, and thus, the input is from $A_{\boldsymbol{s},\chi}$. Otherwise, if the procedure completes all iterations without accepting, then $\mathcal{B}$ rejects the input, since this means that the acceptance probabilities of $\mathcal{A}$ on U and $f_{\boldsymbol{t}}(R)$ are close to each other for all $\boldsymbol{t}$ values.

Let us consider each case separately:

1. R **is from** U**:** If the input is from U, then $f_{\boldsymbol{t}}(U) = U$, and therefore $\mathcal{B}$ will perform estimates of the acceptance probability of the same distribution. Let P_U be the acceptance probability of $\mathcal{A}$ on U, and P_1 be the estimate of the acceptance probability of either $\mathcal{A}$ on U or $f_{\boldsymbol{t}}(U)$. Then, we have that the probability of this estimate deviating more than $n^{-c_2}/8$ from the acceptance probability for $O(n^{2c_2+1})$ calls is given by:

$$\Pr\left[|P_U - P_1| > n^{-c_2}/8\right] \leqslant 2 \cdot \exp(-2(n^{-c_2}/8)^2 \cdot n^{2c_2+1}) \leqslant O(\exp(-n))$$

   Therefore, the probability the estimate P_1 deviates more than $n^{-c_2}/8$ is exponentially small. Therefore, by union bound, the sum both estimates P_1 and P_2 deviate more than $2n^{-c_2}/8 < n^{-c_2}/2$, is exponentially small, and thus, $\mathcal{B}$ will reject the input with exponentially high probability.

2. R **is from** $A_{\boldsymbol{s},\chi}$**:** In this case, the input is from $A_{\boldsymbol{s},\chi}$ for some $\boldsymbol{s}$. In each of the n^{c_1+1} iterations, the algorithm $\mathcal{B}$ chooses a random $\boldsymbol{t} \in \mathbb{Z}_q^n$ and estimates the acceptance probability of $\mathcal{A}$ on $f_{\boldsymbol{t}}(A_{\boldsymbol{s},\chi}) = A_{\boldsymbol{s}+\boldsymbol{t},\chi}$. Since $\boldsymbol{s} + \boldsymbol{t}$ is uniformly distributed over $\mathbb{Z}_q^n$,

then, the probability that $\boldsymbol{t}$ is such that $\boldsymbol{s}+\boldsymbol{t}$ is one of the secret keys for which $\mathcal{A}$ has non-negligible acceptance probability is at least n^{-c_1}. Therefore, the probability that *none* of the $O(n^{c_1+1})$ iterations chooses a $\boldsymbol{t}$ such that $\boldsymbol{s}+\boldsymbol{t}$ is one of the secret keys is given by:

$$\Pr\left[\textit{no} \text{ iteration chooses correct } \mathbf{t}\right] \leqslant (1-n^{-c_1})^{O(n^{c_1+1})} \leqslant \exp(-n^{c_1}+n^{c_1+1}) = \exp(-n),$$

which is exponentially small. Therefore, with probability exponentially close to 1, *at least one* of the iterations will choose a "good" $\boldsymbol{t}$ for which $\mathcal{A}$ has an acceptance probability of at least n^{-c_2}, and, since the estimates are within $n^{-c_2}/8$ of the true acceptance probability, then the acceptance probability will be greater than $n^{-c_2}/2$ and $\mathcal{B}$ will accept the input.

□

Step 2. "(strong) Decision LWE $\implies$ Search LWE"

Lemma 4.1.30 (Decision to Search LWE)**.** *Let $n \geqslant 1$ be some integer, let $2 \leqslant q \leqslant poly(n)$ be a prime, and let χ be a distribution over $\mathbb{Z}_q$. Assume we have access to a distinguisher $\mathcal{A}$ that can distinguish between the LWE samples $A_{\boldsymbol{s},\chi}$ and uniform samples U for all $\boldsymbol{s}$ with probability exponentially close to 1. Then, there exists a polynomial time algorithm $\mathcal{B}$ that given samples from $A_{\boldsymbol{s},\chi}$ for some $\boldsymbol{s}$, outputs $\boldsymbol{s}$ with probability exponentially close to 1.*

Proof. The high-level idea of this proof is that, by applying a transformation to the LWE samples, we can "isolate" each component of the secret key $\boldsymbol{s}$ and use the distinguisher $\mathcal{A}$, that can distinguish samples from $A_{\boldsymbol{s},\chi}$ and uniform samples for all $\boldsymbol{s}$, to recover the secret key.

Consider the following transformation for finding s_1. For any $k \in \mathbb{Z}_q$, given a pair $(\boldsymbol{a}, b)$, we output the pair $(\boldsymbol{a}+(\ell,0,\ldots,0), b+k\cdot\ell)$, where $\ell \in \mathbb{Z}_q$ is chosen uniformly at random. This transformation takes the uniform distribution U to itself. Now lets consider the effect of the transformation on the LWE samples.

1. $k = s_1$**:** Suppose we have a sample $(\boldsymbol{a} = (a_1, a_2, \ldots, a_n), b = a_1 s_1 + \ldots, +a_n s_n + e)$. The transformed sample is

$$\begin{aligned} \boldsymbol{a}' &= (a_1+\ell, a_2, \ldots, a_n) \\ b &= a_1 s_1 + a_2 s_2 + \cdots + a_n s_n + e + \ell s_1 \\ &= (a_1+\ell)s_1 + \cdots + a_n s_n \end{aligned}$$

   Since ℓ is uniformly random over $\mathbb{Z}_q$, then $a_1+\ell$ is also uniformly random over $\mathbb{Z}_q$, and thus, the transformed sample is also a sample from $A_{\boldsymbol{s},\chi}$, with a different first component of $\boldsymbol{a}$.

2. $k \neq s_1$**:** In this case, the transformed sample is

$$\begin{aligned} \boldsymbol{a}' &= (a_1 + \ell, a_2, \ldots, a_n) \\ b &= a_1 s_1 + a_2 s_2 + \cdots + a_n s_n + e + \ell k \\ &= \langle \boldsymbol{a}' \, , \, \boldsymbol{s} \rangle + e + \ell(k - s_1) \end{aligned}$$

Since $k \neq s_1$, then the transformed b' adds the term $\ell(k - s_1)$, which is uniformly random over $\mathbb{Z}_q$ and independent, making b' uniformly random over $\mathbb{Z}_q$ as well. Therefore, the transformed sample $(\boldsymbol{a}', b')$ is uniformly random over $\mathbb{Z}_q^n \times \mathbb{Z}_q$, or from U.

Therefore, the transformed sample is from $A_{\boldsymbol{s},\chi}$ if and only if $k = s_1$, and thus, we can use the distinguisher $\mathcal{A}$ to distinguish between the transformed samples from $A_{\boldsymbol{s},\chi}$ and uniform samples. Consider the algorithm $\mathcal{B}$ described below.

In order to find s_1, $\mathcal{B}$ iterates through all possible values of $k \in \mathbb{Z}_q$ and applies the transformation to the LWE samples, obtaining the transformed samples $(\boldsymbol{a}', b')$ and gives them to $\mathcal{A}$. If $k = s_1$, then $\mathcal{A}$ accepts with high probability. Otherwise, for all other values of $k \neq s_1$, the transformed samples are from U and $\mathcal{A}$ rejects with high probability. Therefore, $\mathcal{B}$ outputs the value k for which $\mathcal{A}$ accepts.

Then, we can repeat the process for the other components of $\boldsymbol{s}$, by applying a similar transformation for each s_i, for $i = 2, \ldots, n$. Since the $q \leqslant \text{poly}(n)$, then the algorithm $\mathcal{B}$, for each component, will call $\mathcal{A}$ at most $O(q)$ times, and thus, the total number of calls to $\mathcal{A}$ is $O(nq)$, and the algorithm $\mathcal{B}$ runs in polynomial time, and outputs $\boldsymbol{s}$ with probability exponentially close to 1.

□

Step 3. "Discrete noise $\implies$ Continuous noise Search LWE"

Lemma 4.1.31 (Discrete to Continuous LWE)**.** *Let $n, q \geqslant 1$ be integers, let ϕ be some probability density function on $\mathbb{T}$ and let $\bar{\phi}$ be its discretization on $\mathbb{Z}_q$. Assume we have access to a distinguisher $\mathcal{A}$ that solves $LWE_{q,\bar{\phi}}$, then there exists a polynomial time algorithm $\mathcal{B}$ that solves $LWE_{q,\phi}$.*

Proof. Consider the algorithm $\mathcal{B}$ that, on input samples from $A_{\boldsymbol{s},\phi}$, for some $\boldsymbol{s} \in \mathbb{Z}_q^n$, discretizes the second element to obtain samples from $A_{\boldsymbol{s},\bar{\phi}}$. The algorithm $\mathcal{B}$ then calls the distinguisher $\mathcal{A}$ on the transformed samples. Whenever $\mathcal{A}$ outputs a secret key $\boldsymbol{s}'$, $\mathcal{B}$ outputs $\boldsymbol{s}'$ as the output of the algorithm. □

This concludes the proofs for the reductions from the search LWE problem to the average-case decision LWE problem, showing that the average-case decision LWE problem is at least as hard as the search LWE problem.

4.2 Ring-LWE

The Ring Learning with Errors (Ring-LWE) problem, introduced in [LPR12] is a variant of the Learning with Errors (LWE) problem introduced by [Reg05] (Sec. 4.1). It improves upon the LWE problem by leveraging ring structures, allowing for more efficient implementations. The authors show in [LPR12] a *quantum* reduction from the Ring-LWE problem to approximate SVP (SVP_γ) in *ideal lattices*. Ideal lattices, on the other hand, are a special class of lattices that have additional structure, and, even though there is no proof of hardness of the SVP in ideal lattices (as there is for general lattices – See Sec. 2.3.5), there are no known algorithms that take advantage of this structure to provide significantly faster algorithms for solving the SVP in ideal lattices. In a subsequent paper [LPR13], the authors expand on this work and provide a toolkit of algorithms and analytical techniques to be used in ring-based cryptographic applications, including a few cryptosystems based on the Ring-LWE problem.

As an overview, Lyubashevsky et al. begin by defining the Ring-LWE distribution and related problems (Decision and Search), similar to Regev's work on LWE, then show that if an adversary $\mathcal{A}$ can solve the average-case (Decision) Ring-LWE problem, then a quantum reduction $\mathcal{B}$ can solve the SIVP_γ (or SVP_γ) on ideal lattices. This is done mainly in two steps: first, they show a *quantum* reduction from the SVP_γ in the worst case on ideal lattices to the (Search) Ring-LWE problem, then they show a *classical* search to decision reduction for the Ring-LWE problem, which is useful for the security proofs of cryptosystems based on this problem.

Several of the steps in the proofs are similar to those in [Reg05], following the same general structure, including the iterative steps to reduce the Ring-LWE problem to the SIVP_γ. Most of these however, have slight modifications to account for the ring structure and ideal lattices.

4.2.1 Background and Definitions

Polynomial Rings

The quotient ring $\mathcal{R} = \mathbb{Z}[X]/(f(X))$ is the set of polynomials with integer coefficients modulo a polynomial $f(X)$, where $f(X)$ is a monic polynomial of degree n (i.e., $f(X) = X^n + \ldots + f_0$). That is, $\mathcal{R}$ is the set of polynomials of the form $a(X) = a_0 + a_1 X + \ldots + a_{n-1}X^{n-1}$, where $a_i \in \mathbb{Z}$, with addition and multiplication performed modulo $f(X)$. Let q be a prime modulus, and let $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/(f(X))$ be the ring of polynomials with coefficients in $\mathbb{Z}_q$.

One representation of $\mathcal{R}_q$ is as a vector of dimension n over $\mathbb{Z}_q$, where the i-th coordinate of the vector corresponds to the coefficient of X^i in the polynomial. For example, the polynomial $a(X) = 3 + 2X + 4X^2$ can be represented as the vector $(3, 2, 4) \in \mathbb{Z}_q^3$.

Polynomial Ring Operations Under the coefficient embedding described above, the ring operations can be defined as follows:

Addition The addition of two polynomials $a(X), b(X) \in \mathcal{R}_q$ is defined as the polynomial $c(X) = a(X) + b(X) = \sum_{i=0}^{n-1}(a_i + b_i)X^i$, where a_i and b_i are the coefficients of $a(X)$ and $b(X)$, respectively. The addition is performed modulo q. That is, the addition of two polynomials is done by adding the coefficients of the polynomials mod q, or in other words, the addition of two vectors in $\mathbb{Z}_q^n$.

Multiplication For multiplication of two polynomials in the ring, however, the easiest way is to define a matrix for the first polynomial $\mathcal{M}_a$, and then do a matrix-vector multiplication with the second polynomial. To do that, consider the following:

$$a(X) \cdot b(X) \bmod f(X) = a(X) \cdot \left(\sum_{i=0}^{n-1} b_i X^i \right) \bmod f(X) = \sum_{i=0}^{n-1} (a(X)X^i \bmod f(X)) \cdot b_i$$

Therefore, we can define the matrix $\mathcal{M}_a$ as follows:

$$\mathcal{M}_a = \begin{bmatrix} a(X) & a(X)X \bmod f(X) & \cdots & a(X)X^{n-1} \bmod f(X) \end{bmatrix}$$

where each polynomial $a(X)X^i \bmod f(X)$ is represented as a vector of coefficients. Finally, the multiplication of two polynomials $a(X), b(X) \in \mathcal{R}_q$ can be computed as:

$$a(X) \cdot b(X) \bmod f(X) = \mathcal{M}_a \cdot \boldsymbol{b},$$

where $\boldsymbol{b}$ is the vector of coefficients of $b(X)$, and $\mathcal{M}_a$ is the matrix defined above.

Consider the following example: let $a(X) = 3X + 2$ and $b(X) = X + 4$, with the modulus $f(X) = X^2 + 1$. Then, the vector representations of the polynomials are $\boldsymbol{a} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$ and $\boldsymbol{b} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$. The matrix $\mathcal{M}_a$ is given by:

$$\mathcal{M}_a = \begin{bmatrix} 2 & -3 \\ 3 & 2 \end{bmatrix},$$

where the first column is simply the coefficients of $a(X)$ and the second column is obtained by multiplying $a(X)$ by X and reducing modulo $f(X)$:

$$\begin{aligned} a(X)X \bmod f(X) &= (3X + 2)X \bmod f(X) \\ &= 3X^2 + 2X \bmod f(X) \\ &= 3(-1) + 2X \bmod f(X) \\ &= -3 + 2X \end{aligned}$$

and we have used the fact that $X^2 \equiv -1 \bmod f(X)$ to reduce the polynomial.

The first column is simply the coefficients of $a(X)$, and the second column is obtained by multiplying $a(X)$ by X and reducing modulo $f(X)$. The product is given by:

$$c(X) = a(X) \cdot b(X) \bmod f(X) = \mathcal{M}_a \cdot \boldsymbol{b} = \begin{bmatrix} 2 & -3 \\ 3 & 2 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 8-3 \\ 12+2 \end{bmatrix} = \begin{bmatrix} 5 \\ 14 \end{bmatrix} = \boldsymbol{c}$$

Now, doing the product as polynomials, we have:

$$\begin{aligned} c(X) = a(X) \cdot b(X) \bmod f(X) &= (3X+2)(X+4) \bmod f(X) \\ &= 3X^2 + 12X + 2X + 8 \bmod f(X) \\ &= 3X^2 + 14X + 8 \bmod f(X) \\ &= 14X + 8 - 3 = 14X + 5 \end{aligned}$$

The result of the multiplication is $c(X) = 14X + 5$, which is the same as the result obtained from the matrix multiplication $\boldsymbol{c} = \begin{bmatrix} 5 \\ 14 \end{bmatrix}$.

Ideal and Ideal Lattice

An Ideal $\mathcal{I}$ in a ring is an additive subgroup of the ring that is closed under multiplication by any element of the ring. For the rings considered in this work, ideals are finitely generated by all integer linear combinations of some basis $\{ u_1, \ldots, u_n \}$, where $u_i \in \mathcal{R}$. The ideal lattice can be defined as the n-dimensional lattice under an embedding $\sigma(\mathcal{I})$ generated by the basis $\{ \sigma(u_1), \ldots, \sigma(u_n) \}$, where σ is an embedding that maps the ring elements to a vector space. An example is the coefficient embedding described above, where the ideal lattice is represented as a vector space over $\mathbb{Z}_q^n$.

Ideal Lattice Problems Similar to the lattice problems defined in Sec. 2.3.5, we can define the ideal lattice problems as follows, where the problems are defined in the context of a fractional ideal $\mathcal{I}$ in a number field K instead of a lattice $\mathcal{L}$ in $\mathbb{Z}^n$. The definitions are taken from [LPR12].

Definition 4.2.1 (Ideal SVP)**.** *Let K be a number field of degree n with some geometric norm (e.g., the Euclidean norm), and let $\gamma \geqslant 1$. The K-SVP$_\gamma$ problem is, given a fractional ideal $\mathcal{I}$ in K, find some non-zero element $\boldsymbol{x} \in \mathcal{I}$ such that $\|\boldsymbol{x}\| \leqslant \gamma \cdot \lambda_1(\mathcal{I})$.*

Definition 4.2.2 (Ideal SIVP)**.** *Let K be a number field of degree n with some geometric norm (e.g., the Euclidean norm), and let $\gamma \geqslant 1$. The K-SIVP$_\gamma$ problem is, given a fractional ideal $\mathcal{I}$ in K, find n linearly independent, non-zero elements in $\mathcal{I}$ whose norms are all at most $\gamma \cdot \lambda_n(\mathcal{I})$.*

Definition 4.2.3 (Ideal BDD)**.** *Let K be a number field of degree n with some geometric norm (e.g., the Euclidean norm), let $\mathcal{I}$ be a fractional ideal in K, and let $d < \lambda_1(\mathcal{I})/2$. The K-BDD$_{\mathcal{I},d}$ problem in the given norm is: given $\mathcal{I}$ and a vector $\boldsymbol{y}$ of the form $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$, where $\boldsymbol{x} \in \mathcal{I}$ and $\|\boldsymbol{e}\| \leqslant d$, find $\boldsymbol{x}$.*

Ring-LWE Problem

Similar to the LWE problem, the Ring-LWE distribution and problems are defined as follows. The definitions are taken from [LPR12].[7]

Let $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ with $\mathcal{R} = \mathbb{Z}[X]/(f(X))$ where $f(X)$ is monic of degree n; all ring operations below are in $\mathcal{R}_q$.

Definition 4.2.4 (Ring-LWE Distribution)**.** *For a secret $\boldsymbol{s} \in \mathcal{R}_q$ and an error distribution χ over $\mathcal{R}_q$, a sample from the Ring-LWE distribution $A_{\boldsymbol{s},\chi}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is generated by choosing $\boldsymbol{a} \hookleftarrow \mathcal{R}_q$ uniformly at random, $\boldsymbol{e} \hookleftarrow \chi$, and outputting the pair $(\boldsymbol{a}, \boldsymbol{a}\cdot\boldsymbol{s}+\boldsymbol{e} \bmod q)$.*

From the definition above, we can see that the Ring-LWE problem is a generalization of the LWE problem, where the secret $\boldsymbol{s}$, the error $\boldsymbol{e}$, and the vector $\boldsymbol{a}$ are all polynomials in the ring $\mathcal{R}_q$. From the distribution, we derive the two versions of the problem:

Definition 4.2.5 ((Search) Ring-LWE$_{n,q,f(X),\chi}$ Problem (R-LWE))**.** *Given $m = \text{poly}(n)$ independent samples from the Ring-LWE distribution $A_{\boldsymbol{s},\chi}$ for an unknown secret $\boldsymbol{s} \in \mathcal{R}_q$ and an error distribution χ over $\mathcal{R}_q$, output $\boldsymbol{s}$ with non-negligible probability (in n).*

Definition 4.2.6 (Decision Ring-LWE$_{n,q,f(X),\chi}$ Problem (R-DLWE))**.** *Given $m = \text{poly}(n)$ independent samples of the form $(\boldsymbol{a}_i, \boldsymbol{b}_i) \in \mathcal{R}_q \times \mathcal{R}_q$, where every sample is distributed according to either:*

1. *the Ring-LWE distribution $A_{\boldsymbol{s},\chi}$ for a secret $\boldsymbol{s} \hookleftarrow \mathcal{R}_q$ (fixed for all samples), or*
2. *the uniform distribution over $\mathcal{R}_q \times \mathcal{R}_q$,*

decide which is the case with non-negligible advantage.

4.2.2 Main Result

The main result of [LPR12] is the following theorem (adapted), which shows a quantum reduction from the Ring-LWE problem to the SVP$_\gamma$ problem in ideal lattices, that is, the Ring-LWE problem is at least as hard as the SVP$_\gamma$ problem in ideal lattices.

Let $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ with $\mathcal{R} = \mathbb{Z}[X]/(f(X))$ where $f(X)$ is monic of degree n; all ring operations below are in $\mathcal{R}_q$.

Theorem 4.2.7. *Let K be a mth cyclotomic number field of degree $n = \varphi(m)$ and $\mathcal{R} = \mathcal{O}_K$. Let $\alpha < \sqrt{\log n/n}$, and let $q \geqslant 2$, $q \equiv 1 \bmod m$ be a prime bounded by a polynomial in n such that $\alpha q \geqslant \omega(\sqrt{\log n})$.*

There is a polynomial time quantum reduction from $\tilde{O}(\sqrt{n}/\alpha)$-approximate (worst-case) SIVP (equivalently SVP) on ideal lattices in K to the (average-case) problem Decision Ring-LWE$_{n,q,f(x),\chi}$ problem over $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$.

[7] The definitions are slightly modified to use the ring $\mathcal{R}_q$ instead of the a dual fractional ideal $\mathcal{R}_q^\vee$, defined in the paper. The dual ring is used in order for the proofs to be easier to analyze, where the errors $\boldsymbol{e}^\vee$ are near spherical in the canonical embedding used.

Overview

Most steps of the proof are similar to the one presented in Regevs work on LWE, with the main difference being that the Ring-LWE problem is defined in the context of a ring, and therefore, the reduction is done in the context of a ring as well. The proof is done in two main components that are independent:

1. ***quantum* reduction from the worst-case SIVP_γ problem to the Search Ring-LWE problem:** This part follows the same structure as the one in [Reg05], with an iterative quantum reduction in two main parts:
   (a) Classical Part: Reduction from BDD on ideals to R-LWE.
   (b) Quantum Part: Essentially the same as the one in [Reg05], reproduced in Lemma 4.1.22 (Quantum step), and used here as a black box, shows a reduction from the DGS problem on ideal lattices to the BDD problem on ideals.
2. **reduction from the Search Ring-LWE problem to the Decision Ring-LWE problem:** This part shows the equivalence between the Ring-LWE variants and thus, that the R-LWE distribution is pseudorandom. Similar to the one in [Reg05] this reduction is done in several steps, with the main difference being that the ring structure introduces the need for additional steps.

Reduction: Search to Decision Ring-LWE

Through a series of reductions, [LPR12] show that the (average-case) Decision Ring-LWE is at least as hard as the Search Ring-LWE problem, and therefore, if there is a (weak) distinguisher for the average-case Decision Ring-LWE problem, then we can solve the (worst-case) Search Ring-LWE problem. In other words, the different variants of the Ring-LWE problem are as hard as the (Search) Ring-LWE problem, and thus, if one of them is hard (Search Ring-LWE is as hard as SIVP_γ or SVP_γ problems on ideal lattices), then all of them are hard.

1. **Automorphism shuffle ([LPR12], Lemma 5.5).** The first step is to show that the (search) $\mathrm{RLWE}_{q,\Psi}$ problem can be reduced to the (search) $\mathfrak{q}_i$-$\mathrm{RLWE}_{q,\Psi}$ problem, that is, the problem of recovering the secret modulo a fixed prime-ideal factor: recover $s \bmod \mathfrak{q}_i R^\vee$. This is done by using the fact that the ring $\mathcal{R}_q$ can be split into a product $\langle q \rangle = \prod_{i=1}^{k} \mathfrak{q}_i$, where $\mathfrak{q}_i$ are the prime-ideal factors of the ring and that there are field automorphisms τ_k of the ring that permute the ideals above q, that is, $\tau_k(\mathfrak{q}_i) = \mathfrak{q}_{j \cdot k^{-1}}$.

   For the reduction, starting from the *search* variant of $\mathrm{RLWE}_{q,\psi}$ we first "project" the secret onto a single prime-ideal factor $\mathfrak{q}_i \mid \langle q \rangle$. Given samples $(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}_q \times \mathbb{T}$ and a uniformly random field-automorphism τ_k of K, we output $(\tau_k(\boldsymbol{a}), \tau_k(\boldsymbol{b})) \in \mathcal{R}_q \times \mathbb{T}$. Because τ_k simply permutes the ideals above q, this turns knowledge of $s \bmod \mathfrak{q}_i$ for one i into knowledge of $s \bmod \mathfrak{q}_j$ for *every* j. Using the Chinese Remainder

Theorem, we can then reconstruct $s \in \mathcal{R}_q^\vee$ from the s mod $\mathfrak{q}_i$ for all i. Hence a solver for the new problem $\mathfrak{q}_i$-RLWE$_{q,\Psi}$ (recovering s modulo a fixed $\mathfrak{q}_i$) suffices to solve the original search problem.

2. **Search → (weak) decision ([LPR12], Lemma 5.9).**

   The next step reduces the search problem, for a fixed ideal factor $\mathfrak{q}_i$, to a decision problem, also relative to the fixed ideal factor $\mathfrak{q}_i$. This problem is defined as WD RLWE$^i_{q,\Psi}$, where, given access to a (hybrid) distribution $A^j_{\boldsymbol{s},\psi}$ for some secret $\boldsymbol{s} \in \mathcal{R}_q^\vee$, and $j \in \{\, i-, i \,\}$, find j. The $A^j_{\boldsymbol{s},\psi}$ represents a hybrid distribution where, for input $(\boldsymbol{a}, \boldsymbol{b}) \hookleftarrow\$\ A_{\boldsymbol{s},\psi}$, output $(\boldsymbol{a}, \boldsymbol{b} + \boldsymbol{h}/q)$, with $\boldsymbol{h} \hookleftarrow\$\ \mathcal{R}_q$ uniformly at random for all $j \leqslant i$ (i.e. for the first i components of $\boldsymbol{h}$) and 0 mod the remaining $\mathfrak{q}_j\mathcal{R}^\vee$, with $A^0_{\boldsymbol{s},\psi}$ being the original distribution.

   The idea of the reduction to recover the secret $\boldsymbol{s}$ mod $\mathfrak{q}_i\mathcal{R}^\vee$ is to try each of its possible values ($q = poly(n)$ possible values) by modifying the samples $(\boldsymbol{a}, \boldsymbol{b})$ so that they are distributed according to $A^{i-}_{\boldsymbol{s},\psi}$ in the correct value of $\boldsymbol{s}$ mod $\mathfrak{q}_i\mathcal{R}^\vee$, or $A^i_{\boldsymbol{s},\psi}$ in all other cases, and then using the decision oracle to distinguish between the two distributions.

3. **Worst-case decision → average-case decision ([LPR12], Lemma 5.12).**

   Here we are assuming we have an oracle that can distinguish between the two distributions $A^{i-}_{\boldsymbol{s},\psi}$ and $A^i_{\boldsymbol{s},\psi}$ only for a non-negligible fraction of the secrets $\boldsymbol{s} \in \mathcal{R}_q^\vee$ and error distributions $\psi \in \Psi$ (the weak distinguisher), defined as the (average-case) decision problem D RLWE$^i_{q,\Psi}$.

   Similar to the worst-case to average-case reduction in [Reg05], we can use the weak distinguisher to build a strong one that works for all secrets $\boldsymbol{s} \in \mathcal{R}_q^\vee$ by randomly shifting the secret $\boldsymbol{s}$ by a uniformly random element $\boldsymbol{s}' \in \mathcal{R}_q^\vee$, and then using the weak distinguisher to distinguish between the two distributions $A^{i-}_{\boldsymbol{s},\psi}$ and $A^i_{\boldsymbol{s},\psi}$ for the shifted secret $\boldsymbol{s} + \boldsymbol{s}'$. This is repeated for a polynomial number of times. The idea is that if in *any* of the attempts the (shifted) distributions present a non-negligible difference in the acceptance probability, then it means we have found a secret $\boldsymbol{s} + \boldsymbol{s}' \in S$, that is, the secret is in the set of secrets that the weak distinguisher can distinguish. Since there is a non-negligible fraction of secrets $\boldsymbol{s} \in \mathcal{R}_q^\vee$ that the weak distinguisher can distinguish, then (by repeating the process for a polynomial number of times) there is a non-negligible chance that we will find a such secret $\boldsymbol{s} + \boldsymbol{s}'$.

4. **Hybrid argument ([LPR12], Lemma 5.14).**

   The last step is to show that an oracle for the (average-case) decision problem D RLWE$_{q,\Psi}$, that distinguishes between the $A_{\boldsymbol{s},\psi}$ and the uniform distribution with non-negligible advantage for some pair $(\boldsymbol{s}, \psi)$, can be used as an oracle to solve the (average-case) decision problem D RLWE$^i_{q,\gamma}$, that is, the problem of distinguishing between the two distributions $A^{i-}_{\boldsymbol{s},\psi}$ and $A^i_{\boldsymbol{s},\psi}$ for a fixed secret.

   Here, we make use of the hybrid argument, considering that $A^0_{\boldsymbol{s},\psi} = A_{\boldsymbol{s},\psi}$ and $A^{m-1}_{\boldsymbol{s},\psi}$ is within negligible distance from the uniform distribution. The idea is to gradually

randomize the distribution one prime-ideal at a time, producing the sequence of distributions $A^0_{\boldsymbol{s},\psi}, A^1_{\boldsymbol{s},\psi}, \ldots, A^{m-1}_{\boldsymbol{s},\psi}$, where each consecutive pair $A^{i-1}_{\boldsymbol{s},\psi}$ and $A^i_{\boldsymbol{s},\psi}$ differ only in the $\mathfrak{q}_i$ component. Since the oracle can distinguish between the two distributions $A_{\boldsymbol{s},\psi}$ and the uniform distribution with non-negligible advantage, then there must be a point in the sequence which the oracle distinguishes between $A^{i-1}_{\boldsymbol{s},\psi}$ and $A^i_{\boldsymbol{s},\psi}$ with non-negligible advantage, and that coincides with the distributions $A^{i-}_{\boldsymbol{s},\psi}$ and $A^i_{\boldsymbol{s},\psi}$ needed to solve the decision problem D RLWE$^i_{q,\Psi}$.

This completes the reduction from the Search Ring-LWE problem to the Decision Ring-LWE problem, and therefore, putting the pieces together, we can conclude that the (average-case) Decision Ring-LWE problem is at least as hard as the (worst-case) Search Ring-LWE.

4.2.3 Public-key Encryption Scheme

Briefly mentioned in [LPR12], with the correctness and security proofs given in [LPR13], the authors describe an adaptation of the LWE-based public-key encryption scheme to the ring setting, making it more efficient and practical. Here we describe a simple version of the cryptosystem in order to illustrate the main benefits of the Ring-LWE based cryptosystems. The security of the scheme is based on the hardness of the Ring-LWE problem, and therefore, on the hardness of the SIVP$_\gamma$ problem in ideal lattices.

Private Key

The private key is a secret $\boldsymbol{s} \leftarrow\!\!\$\ \chi \in \mathcal{R}_q$, where χ is a distribution over $\mathcal{R}_q$.

Public Key

Sample a uniformly random polynomial $\boldsymbol{a} \leftarrow\!\!\$\ \mathcal{R}_q$, and a small error polynomial $\boldsymbol{e} \leftarrow\!\!\$\ \chi \in \mathcal{R}_q$. The public key is the pair $(\boldsymbol{a}, \boldsymbol{b} = \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e} \bmod \mathcal{R}) \in \mathcal{R}_q \times \mathcal{R}_q$.

Encryption

To encrypt an n-bit message $m \in \{0,1\}^n$, we view it as an element of the ring $\mu \in \mathcal{R}_q$, using the bits as coefficients. Take three random small elements $\boldsymbol{r}, \boldsymbol{e}_1, \boldsymbol{e}_2 \leftarrow\!\!\$\ \chi \in \mathcal{R}_q$, and output the pair $(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{R}_q \times \mathcal{R}_q$ computed as follows:

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{a} \cdot \boldsymbol{r} + \boldsymbol{e}_1 \bmod q, \\ \boldsymbol{v} &= \boldsymbol{b} \cdot \boldsymbol{r} + \boldsymbol{e}_2 + \lfloor \frac{q}{2} \rceil \cdot \mu \bmod q, \end{aligned}$$

Decryption

To decrypt the ciphertext $(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{R}_q \times \mathcal{R}_q$, using the secret key $\boldsymbol{s}$, compute:

$$
\begin{aligned}
\boldsymbol{v} - \boldsymbol{u} \cdot \boldsymbol{s} &= \boldsymbol{b} \cdot \boldsymbol{r} + \boldsymbol{e}_2 + \lfloor \frac{q}{2} \rceil \cdot \mu - (\boldsymbol{a} \cdot \boldsymbol{r} + \boldsymbol{e}_1) \cdot \boldsymbol{s} \\
&= (\cancel{\boldsymbol{a} \cdot \boldsymbol{s} \cdot \boldsymbol{r}} + \boldsymbol{e} \cdot \boldsymbol{r} + \boldsymbol{e}_2 - \cancel{\boldsymbol{a} \cdot \boldsymbol{r} \cdot \boldsymbol{s}} - \boldsymbol{e}_1 \cdot \boldsymbol{s}) + \lfloor \frac{q}{2} \rceil \cdot \mu \\
&= (\boldsymbol{e} \cdot \boldsymbol{r} + \boldsymbol{e}_2 - \boldsymbol{e}_1 \cdot \boldsymbol{s}) + \lfloor \frac{q}{2} \rceil \cdot \mu
\end{aligned}
$$

where, similar to the LWE case, the error term $(\boldsymbol{e} \cdot \boldsymbol{r} + \boldsymbol{e}_2 - \boldsymbol{e}_1 \cdot \boldsymbol{s}) \in \mathcal{R}_q$ has coefficients with magnitude less than $q/4$, and the bits of the message μ can be recovered by rounding each coefficient of the polynomial to either 0 or $\lfloor \frac{q}{2} \rceil$.

The trade-off for the Ring-LWE based cryptosystem compared to the LWE-based cryptosystem is that each sample in the Ring-LWE case $(\boldsymbol{a}, \boldsymbol{b}) \in \mathcal{R}_q \times \mathcal{R}_q$, allows the encryption of a message with n bits, while in the LWE case, each sample $(\boldsymbol{a}, b) \in \mathbb{Z}^n \times \mathbb{Z}$ can only encrypt a single bit. Thus, the Ring-LWE based cryptosystem replaces n samples of the LWE problem with a single sample, reducing the public key size by a factor of n. This, however, comes at the cost of introducing a structure to the problem, which can potentially be exploited by an adversary.

One topic that is not covered in this section, but worth mentioning, is the use of the Number Theoretic Transform (NTT) to speed up the polynomial operations in the Ring-LWE based cryptosystems, which is a common technique used in practice to improve the performance of these cryptosystems. The NTT allows for efficient multiplication of polynomials in the ring, reducing the complexity of the operations from $O(n^2)$ to $O(n \log n)$, making the Ring-LWE (and also Module-LWE (Sec. 4.3)) based cryptosystems more practical for real-world applications, making great use of the ring structure to both speed up the operations and reduce the key sizes. This topic will be covered in more detail in the next chapter, Sec. 5.4.3, where we will discuss the NTT and its effects on the performance of the Ring-LWE based cryptosystems.

4.3 Module-LWE

The Module Learning with Errors (Module-LWE, or M-LWE) problem, introduced in [BV11] and [BGV12] is a variant or generalization of the Ring Learning with Errors. It interpolates between the highly structured Ring-LWE (Sec. 4.2) and the standard LWE (Sec. 4.1) problems, replacing the single secret polynomial in the Ring-LWE ($s \in \mathcal{R}_q$) with a vector of secret polynomials ($\boldsymbol{s} \in \mathcal{R}_q^k$), or a *module vector*, similar to the LWE problem (where $\boldsymbol{s} \in \mathbb{Z}_q^n$), now in the context of a ring instead of integers. This makes use of the efficiency gained from the Ring-LWE problem, while sacrificing enough of the structure to make the problem harder and regain stronger security guarantees. In particular, [LS15] show a (quantum) worst-case to average-case reduction from the M-LWE problem to the SIVP_γ problem in *module lattices*, which are a generalization of *ideal lattices* (Sec. 4.2.1), strictly containing them, and thus, being at least as hard as the problem in ideal lattices. This reduction follows both the original LWE reductions from [Reg05] as well as the Ring-LWE reductions from [LPR12], combining the two approaches to achieve the generalization to the module setting.

Similar to the Ring-LWE problem, there are no known algorithms that take advantage of the structure of the module lattices, and therefore, the M-LWE problem is believed to be hard, even in the quantum setting, though it has been less studied than the plain LWE problem. However, [AD17] shows that the M-LWE problem can be reduced to a large modulus Ring-LWE problem, that is, Ring-LWE with modulus q^k is at least as hard as the M-LWE problem with modulus q and module rank k in the same ring, and also that with the same ring dimension n and polynomial $f(X)$, increasing the modulus q in the Ring-LWE also increases the difficulty of the problem. Therefore, even though it is not known if SIVP_γ in *module lattices* is strictly harder than SIVP_γ in *ideal lattices*, solving the M-LWE problem is harder than solving the Ring-LWE problem with the same parameters, making the M-LWE problem a good candidate for cryptographic applications.

4.3.1 Background and Definitions

Module and Module Lattice

A *module* over a ring $\mathcal{R}$, or $\mathcal{R}$-*module*, is a ring generalization of a vector space, where the scalars are elements of the ring $\mathcal{R}$ instead of a field. A module, similar to a vector space, is defined as an abelian group $\mathcal{M}$ under addition, with a scalar multiplication operation $\mathcal{R} \times \mathcal{M} \to \mathcal{M}$. Where in a vector space a basis generates dimension n over a field $\mathcal{F}$, in a module, the basis generates *rank* k over the ring $\mathcal{R}$, where k is the number of elements in the basis.

The definition of the module lattice is similar to the definition of an ideal lattice in Sec. 4.2.1, where, with an embedding σ from K to $\mathbb{R}^n$, and a fractional ideal $\mathcal{I}$ in a number field K, the ideal lattice is the set $\sigma(\mathcal{I})$. For a module $\mathcal{M}$ over a ring $\mathcal{R}$, we define the map $(\sigma, \ldots, \sigma)$ from the module $\mathcal{M} \subseteq K^k$ to $\mathbb{R}^{nk}$ as the embedding of the module into a vector space, and the module lattice as the set $\sigma(\mathcal{M})$.

Module Lattice Problems Analogous to the definitions of the lattice problems in Sec. 2.3.5 and 4.2.1, we can define the module lattice problems as follows, where the problems are defined in the context of a module $\mathcal{M}$ over a ring $\mathcal{R}$. The definitions are taken from [LS15].

Definition 4.3.1 (Module SVP)**.** *Let $\mathcal{M}$ be a module over a ring $\mathcal{R}$, and let $\gamma \geqslant 1$. The Mod-SVP$_\gamma$ problem is, given a module $\mathcal{M} \subseteq K^k$, with the input being a generating set of $\mathcal{M}$, find some non-zero element $\boldsymbol{x} \in \mathcal{M}$ such that $\|\boldsymbol{x}\| \leqslant \gamma \cdot \lambda_1(\mathcal{M})$, where $\|\cdot\|$ is some geometric norm (e.g., the Euclidean norm).*

Definition 4.3.2 (Module SIVP)**.** *Let $\mathcal{M}$ be a module over a ring $\mathcal{R}$, and let $\gamma \geqslant 1$. The Mod-SIVP$_\gamma$ problem is, given a module $\mathcal{M} \subseteq K^k$ with rank k, with the input being a generating set of $\mathcal{M}$, find k $\mathcal{R}$-linearly independent, non-zero elements in $\mathcal{M}$ whose norms are all at most $\gamma \cdot \lambda_{kn}(\mathcal{M})$.*

4.3.2 Module-LWE Problem

The definitions of the Module-LWE distribution and problems are analogous to the definitions of the LWE and Ring-LWE (Def. 4.1.1 and 4.2.4), with the definitions taken from [LS15].[8]

Let $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ with $\mathcal{R} = \mathbb{Z}[X]/(f(X))$ where $f(X)$ is monic of degree n; all ring operations below are in $\mathcal{R}_q$.

Definition 4.3.3 (Module-LWE (M-LWE) Distribution)**.** *For a secret $\boldsymbol{s} \in \mathcal{R}_q^k$, and an error distribution χ over $\mathcal{R}_q$, a sample from the M-LWE distribution $A_{\boldsymbol{s},\chi}^{(\mathcal{M})}$ over $\mathcal{R}_q^k \times \mathcal{R}_q$ obtained by choosing $\boldsymbol{a} \hookleftarrow \mathcal{R}_q^k$ uniformly at random, $\boldsymbol{e} \hookleftarrow \chi$ and outputting the pair $(\boldsymbol{a}, \langle \boldsymbol{a}\,,\, \boldsymbol{s}\rangle + e)$.*[9]

We can see from the definition above that the Module-LWE problem bridges the gap between the Ring-LWE problem and the LWE problem, where the secret $\boldsymbol{s}$ is a vector as in the LWE problem, but each component of the vector is an element of the ring $\mathcal{R}_q$ as in the Ring-LWE problem. From this definition, we can derive the two versions of the problem:

Definition 4.3.4 ((Search) Module-LWE Problem (M-LWE))**.** *Given ℓ independent samples from the Module-LWE distribution $A_{\boldsymbol{s},\chi}^{(\mathcal{M})}$ for an unknown secret $\boldsymbol{s} \in \mathcal{R}_q^k$ and an error distribution χ over $\mathcal{R}_q$, output the secret $\boldsymbol{s}$ with non-negligible probability.*

Definition 4.3.5 (Decision Module-LWE Problem (D-M-LWE))**.** *Given ℓ independent samples of the form $(\boldsymbol{a}_i, b_i) \in \mathcal{R}_q^k \times \mathcal{R}_q$, where every sample is distributed according to either:*

[8] The definitions are slightly modified to use the ring $\mathcal{R}_q$ here for simplicity, instead of the a dual fractional ideal $\mathcal{R}_q^\vee$ as in the original paper.

[9] Here, $\langle \cdot\,,\, \cdot \rangle$ is the inner product in the module $\mathcal{M} \subseteq K^k$ (i.e. $\langle \boldsymbol{a}\,,\, \boldsymbol{s}\rangle = \sum_{i=1}^k a_i \cdot s_i$, with each $a_i, s_i \in \mathcal{R}_q$ and the multiplication is multiplication in the ring $\mathcal{R}_q$ as shown in 4.2.1).

1. *the M-LWE distribution* $A_{\boldsymbol{s},\chi}^{(\mathcal{M})}$ *for a secret* $\boldsymbol{s} \in \mathcal{R}_q^k$ *(fixed for all samples), or*

2. *uniformly at random from* $\mathcal{R}_q^k \times \mathcal{R}_q$,

decide which is the case with non-negligible advantage.

Similar to the other definitions, the M-LWE problem also can be interpreted as the plain LWE problem in matrix form, where each of the ℓ samples $(\boldsymbol{a}_i, b_i)$ is a row in a matrix $\boldsymbol{A} \in \mathcal{R}_q^{\ell \times k}$, and the secret $\boldsymbol{s} \in \mathcal{R}_q^k$ is the vector we want to recover. The M-LWE problem then becomes:

$$(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) \in \mathcal{R}_q^{\ell \times k} \times \mathcal{R}_q^{\ell},$$

where $\boldsymbol{e}$ is a vector of errors sampled from the error distribution χ.

Using the same approach for the multiplication in the ring $\mathcal{R}_q$ (as seen in Sec. 4.2.1), we can represent each element of the matrix $\boldsymbol{A}$ as a matrix and each element of the vector $\boldsymbol{s}$ as a vector, and thus, the multiplication $(\boldsymbol{A}\boldsymbol{s})$ is done in the ring $\mathcal{R}_q$ as well, becoming a matrix-vector multiplication $(\mathcal{M}_{\boldsymbol{A}} \cdot \mathcal{V}_{\boldsymbol{s}})$ in the module over the ring $\mathcal{R}_q$, where $\mathcal{M}_{\boldsymbol{A}}$ is the matrix representation of the samples $\boldsymbol{A}$ and $\mathcal{V}_{\boldsymbol{s}}$ is the vector representation of the secret $\boldsymbol{s}$. Consider the following example for clarity:

Example. Let $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^2+1)$ and let $k = \ell = 2$, a visual representation of the M-LWE problem is as follows:

$$\begin{aligned}
\boldsymbol{A} &= \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \in \mathcal{R}_q^{2\times 2}, \\
\boldsymbol{s} &= \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \in \mathcal{R}_q^2, \\
\boldsymbol{e} &= \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \in \mathcal{R}_q^2, \\
\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} &= \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_{1,1}s_1 + a_{1,2}s_2 + e_1 \\ a_{2,1}s_1 + a_{2,2}s_2 + e_2 \end{pmatrix} \in \mathcal{R}_q^2
\end{aligned}$$

Since each row is now a polynomial with integer coefficients, we can represent the multiplication as follows:

$$\begin{aligned}
\boldsymbol{b} &= \begin{pmatrix} \mathcal{M}_{a_{1,1}} \cdot \mathcal{V}_{s_1} + \mathcal{M}_{a_{1,2}} \cdot \mathcal{V}_{s_2} + \mathcal{V}_{e_1} \\ \mathcal{M}_{a_{2,1}} \cdot \mathcal{V}_{s_1} + \mathcal{M}_{a_{2,2}} \cdot \mathcal{V}_{s_2} + \mathcal{V}_{e_2} \end{pmatrix} \\
&= \begin{pmatrix} \mathcal{M}_{a_{1,1}} & \mathcal{M}_{a_{1,2}} \\ \mathcal{M}_{a_{2,1}} & \mathcal{M}_{a_{2,2}} \end{pmatrix} \cdot \begin{pmatrix} \mathcal{V}_{s_1} \\ \mathcal{V}_{s_2} \end{pmatrix} + \begin{pmatrix} \mathcal{V}_{e_1} \\ \mathcal{V}_{e_2} \end{pmatrix} \\
&= \mathcal{M}_{\boldsymbol{A}} \cdot \mathcal{V}_{\boldsymbol{s}} + \mathcal{V}_{\boldsymbol{e}},
\end{aligned}$$

Finally, we can represent the above example as:

$$\boldsymbol{b} = \begin{pmatrix} \begin{pmatrix} b_1^{(1)} \\ b_1^{(2)} \end{pmatrix} \\ \\ \begin{pmatrix} b_2^{(1)} \\ b_2^{(2)} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} a_{1,1}^{(1,1)} & a_{1,1}^{(1,2)} \\ a_{1,1}^{(2,1)} & a_{1,1}^{(2,2)} \end{pmatrix} & \begin{pmatrix} a_{1,2}^{(1,1)} & a_{1,2}^{(1,2)} \\ a_{1,2}^{(2,1)} & a_{1,2}^{(2,2)} \end{pmatrix} \\ \\ \begin{pmatrix} a_{2,1}^{(1,1)} & a_{2,1}^{(1,2)} \\ a_{2,1}^{(2,1)} & a_{2,1}^{(2,2)} \end{pmatrix} & \begin{pmatrix} a_{2,2}^{(1,1)} & a_{2,2}^{(1,2)} \\ a_{2,2}^{(2,1)} & a_{2,2}^{(2,2)} \end{pmatrix} \end{pmatrix} \cdot \begin{pmatrix} \begin{pmatrix} s_1^{(1)} \\ s_1^{(2)} \end{pmatrix} \\ \\ \begin{pmatrix} s_2^{(1)} \\ s_2^{(2)} \end{pmatrix} \end{pmatrix} + \begin{pmatrix} \begin{pmatrix} e_1^{(1)} \\ e_1^{(2)} \end{pmatrix} \\ \\ \begin{pmatrix} e_2^{(1)} \\ e_2^{(2)} \end{pmatrix} \end{pmatrix}$$

Therefore, generalizing for the matrix $\boldsymbol{A} \in \mathcal{R}_q^{\ell \times k}$, we can see the M-LWE problem in terms of linear algebra:

$$\mathcal{M}_{\boldsymbol{A}} = \underbrace{\left\{ \left(\begin{array}{c|c|c} \mathcal{M}_{a_{1,1}} & \cdots & \mathcal{M}_{a_{1,k}} \\ \hline \vdots & \ddots & \vdots \\ \hline \mathcal{M}_{a_{\ell,1}} & \cdots & \mathcal{M}_{a_{\ell,k}} \end{array} \right) \right.}_{k \text{ blocks}} \left. \vphantom{\left(\begin{array}{c|c|c} \mathcal{M}_{a_{1,1}} & \cdots & \mathcal{M}_{a_{1,k}} \\ \hline \vdots & \ddots & \vdots \\ \hline \mathcal{M}_{a_{\ell,1}} & \cdots & \mathcal{M}_{a_{\ell,k}} \end{array} \right)} \right\} n \times n \text{ block}$$
$$\ell \text{ blocks}$$

This results in a matrix with dimension $N = n \cdot k$, where n is the degree of the polynomial $f(X)$, and k is the rank of the module $\mathcal{M}$. Where fixing the polynomial $f(X)$ and the degree n still lets us choose the rank k of the module, resulting in a tunable trade-off between structure and efficiency.

Relation to LWE and Ring-LWE. Setting $k = n$ and $f(X) = X$ recovers standard LWE, while $k = \ell = 1$ recovers Ring-LWE and intermediate k values yield a trade-off between structure and efficiency.

4.3.3 Main Result

One of the main results of [LS15] is the following theorem (adapted), which shows a quantum reduction from the Module-LWE problem to the SIVP_γ problem in module lattices, that is, the Module-LWE problem is at least as hard as the SIVP_γ problem in *module lattices*.

Let $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ with $\mathcal{R} = \mathbb{Z}[X]/(f(X))$ where $f(X)$ is monic of degree n; all ring operations below are in $\mathcal{R}_q$.

Theorem 4.3.6. *Let $\mathcal{R}_q$ be a cyclotomic ring $\mathcal{R}_q = \mathbb{Z}_q[X]/f(X)$, of degree $n = \varphi(m)$ and with coefficients in $\mathbb{Z}_q$. Fix the rank $k \geqslant 1$, and let $\mathcal{M} = \mathcal{R}_q^k$ be the module over the ring $\mathcal{R}_q$. Let $\alpha < \sqrt{\log n/n}$, and let $q \geqslant 2$ be a prime modulus satisfying:*

$$q \equiv 1 \ mod\ m, \qquad q \leqslant poly(nk), \qquad and \quad \alpha q \geqslant 2\sqrt{k}\omega(\sqrt{\log n})$$

There is a polynomial time quantum reduction from $\tilde{O}\big(\sqrt{8nk^2}\,\omega(\sqrt{\log n})/\alpha\big)$-approximate (worst-case) SIVP on module lattices in the cyclotomic field K to the (average-case) Decision Module-LWE problem.

This theorem is proved in the same way as the original LWE reductions from [Reg05], with the reduction from Mod-SIVP to Mod-DGS being almost directly adapted the plain LWE reduction, while the reduction from Mod-DGS to M-LWE needing some extra steps to adapt to the Module setting, similar to the Ring-LWE reduction from [LPR12]. However, this second reduction remains in the same overall structure, with the iterative steps that decrease the size of of r in the Mod-DGS $D_{\mathcal{M},r}$ problem, using a classical and a quantum step. The classical step contains a reduction from the Mod-BDD to the Mod-DGS problem assuming an M-LWE oracle, and the quantum step contains a reduction from the Mod-DGS to the Mod-BDD problem.

4.3.4 Public-key Encryption Scheme

Here we describe a simple public-key encryption scheme based on the Module-LWE problem, as an adaptation of the plain LWE-based public-key encryption scheme (Sec. 4.1.3) to the module setting, with the main differences being the use of the module $\mathcal{M}$ instead of the vector space $\mathbb{Z}^n$, and the use of the ring $\mathcal{R}_q$ instead of the integers $\mathbb{Z}_q$. The scheme is based on the Module-LWE problem, and therefore, on the hardness of the SIVP_γ problem in *module lattices*. This scheme will serve as a basis for the construction of the Kyber (ML-KEM) cryptosystem (KEM) in Sec. 5, and will be explored in more detail in the following chapter.

Consider the $\text{M-LWE}_{\ell,k,q,f(X),\chi}$ parameters for the PKE described below, where k is the module rank, ℓ is the number of samples, q is the modulus, $f(X)$ is a polynomial of degree n, and χ is a distribution over the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/f(X)$. The scheme is defined as follows:

Private Key

The private key consists of a secret vector $\boldsymbol{s} \leftarrow\!\$\ \chi^k \subseteq \mathcal{R}_q^k$, where each component is sampled independently from the error distribution χ over $\mathcal{R}_q$.

Public Key

Sample a uniformly random matrix $\boldsymbol{A} \leftarrow\!\$\ \mathcal{R}_q^{\ell\times k}$, and a small error vector $\boldsymbol{e} \leftarrow\!\$\ \chi^\ell \subseteq \mathcal{R}_q^\ell$. The public key is the pair $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) \in \mathcal{R}_q^{\ell\times k} \times \mathcal{R}_q^\ell$.

Encryption

To encrypt an n-bit message $m \in \{0,1\}^n$, we view it as an element of the ring $\mu \in \mathcal{R}_q$, using the bits as coefficients. Take three random small elements $\boldsymbol{r}, \boldsymbol{e}_1 \leftarrow\!\$\ \chi \in \mathcal{R}_q^\ell$ and $\boldsymbol{e}_2 \leftarrow\!\$\ \chi \in \mathcal{R}_q$ (where $\boldsymbol{e}_1, \boldsymbol{e}_2$ are small error terms), and output the pair $(\boldsymbol{u}, v) \in \mathcal{R}_q^\ell \times \mathcal{R}_q$ computed as follows:

$$\boldsymbol{u}^T = \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T,$$
$$v = \boldsymbol{r}^T \cdot \boldsymbol{b} + \boldsymbol{e}_2 + \lfloor \frac{q}{2} \rceil \cdot \mu,$$

Decryption

To decrypt the ciphertext $(\boldsymbol{u}, v) \in \mathcal{R}_q^\ell \times \mathcal{R}_q$, using the secret key $\boldsymbol{s}$, compute:

$$\begin{aligned} v - \boldsymbol{u}^T \cdot \boldsymbol{s} &= \boldsymbol{r}^T \cdot \boldsymbol{b} + \boldsymbol{e}_2 + \lfloor \frac{q}{2} \rceil \cdot \mu - (\boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T) \cdot \boldsymbol{s} \\ &= \boldsymbol{r}^T \cdot (\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) + \boldsymbol{e}_2 + \lfloor \frac{q}{2} \rceil \cdot \mu - (\boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T) \cdot \boldsymbol{s} \\ &= (\cancel{\boldsymbol{r}^T \boldsymbol{A}\boldsymbol{s}} + \boldsymbol{r}^T \boldsymbol{e} + \boldsymbol{e}_2 - \cancel{\boldsymbol{r}^T \boldsymbol{A}\boldsymbol{s}} - \boldsymbol{e}_1^T \boldsymbol{s}) + \lfloor \frac{q}{2} \rceil \cdot \mu \\ &= (\boldsymbol{r}^T \boldsymbol{e} + \boldsymbol{e}_2 - \boldsymbol{e}_1^T \boldsymbol{s}) + \lfloor \frac{q}{2} \rceil \cdot \mu \end{aligned}$$

where, similar to the LWE and R-LWE cases, the error term $(\boldsymbol{r}^T \boldsymbol{e} + \boldsymbol{e}_2 - \boldsymbol{e}_1^T \boldsymbol{s}) \in \mathcal{R}_q$ has coefficients with magnitude less than $q/4$, and the bits of the message μ can be recovered by rounding each coefficient of the polynomial to either 0 or $\lfloor \frac{q}{2} \rceil$.

Relative to the Ring-LWE based cryptosystem (Sec. 4.2.3), the Module-LWE based cryptosystem lets us choose the rank k of the module, resulting in a lattice dimension of $N = nk$, and matching the desired security level, while still avoiding the possible vulnerabilities of the $(k = 1)$ Ring-LWE problem. The choice of the rank k allows us to balance between the efficiency of the scheme (i.e., the public key size and the ciphertext size) and the security level, where a larger k results in a larger public key and ciphertext size, but also a higher security level. The message size remains the same as in the Ring-LWE case, where the message is encoded as an element in the ring $\mathcal{R}_q$, allowing for n bits to be encrypted in each instance of the Module-LWE problem, similar to the Ring-LWE case, enjoying the same efficiency in terms of message size.

Chapter 5

Module Lattice KEM

5.1 Introduction

In this chapter, we present a Key Encapsulation Mechanism (KEM) based on the Module-LWE problem, which was introduced in the ML-KEM – which stands for Module Lattice Key Encapsulation Mechanism – specification [MLK24]. The ML-KEM specification is the first public-key primitive to be formally standardized by NIST in the Post-Quantum Cryptography (PQC) standardization process, issued as FIPS 203 in August 2024 [MLK24]. The ML-KEM instantiates the CRYSTALS-Kyber algorithm [SAB+20], an IND-CCA secure key encapsulation mechanism (KEM) whose security is provably reduced to the hardness of the Module LWE problem [LS15] (Sec. 4.3). It is specified with three parameter sets (see table 5.1), `ML-KEM-512`, `ML-KEM-768` and `ML-KEM-1024`, where the numbers denote the secret key size in bits, and provide security level roughly equivalent to AES-128, AES-192 and AES-256, respectively.

As a high level overview, the Kyber algorithm is instantiated in a mainly two steps, using an IND-CPA secure public key encryption (PKE) scheme and applying a Fujisaki-Okamoto transform [FO99] to obtain an IND-CCA secure KEM scheme.

1. `K-PKE` **Scheme** (Sec. 5.5): an IND-CPA secure public key encryption (PKE) scheme is constructed with its security reduced to the Module-LWE problem. In the presentation, we begin with a simplified version of the scheme (`Simp-PKE`, Sec. 5.2.1), which is a PKE based on the Module-LWE problem as shown in Sec. 4.3.4, and then we introduce optimizations to the scheme to obtain the full `K-PKE` scheme, which is the one specified in the ML-KEM specification [MLK24].

2. `ML-KEM` **Scheme** (Sec. 5.9): the `K-PKE` scheme is then transformed into an IND-CCA secure key encapsulation mechanism (KEM) scheme by applying a Fujisaki-Okamoto transform [FO99] (see Sec. 5.8), which is a generic transformation that turns an IND-CPA secure PKE scheme into an IND-CCA secure KEM scheme (in the Random Oracle Model, ROM). The resulting reduction preserves the worst-case to average-case guarantees of the Module-LWE problem to approximate SIVP on module lattices.

Kyber's underlying hardness assumption is the Module-LWE problem, which, since no known attacks exploit the module structure, can be analyzed as a special case of the Learning With Errors (LWE) problem [SAB+20]. Because the number of LWE samples is limited to an attacker in the parameter set (i.e. $m = (k+1)n$), combinatorial attacks such as BKW [BKW00] and linearization attacks (e.g. Arora-Ge [AG11]) are not applicable. The best known strategies are therefore the primal and dual lattice-reduction attacks, which transform the problem into a lattice problem and then solve an instance of the Shortest Vector Problem (SVP) in the lattice [SAB+20].

The presentation follows more closely the ML-KEM specification [MLK24], however we reference the Kyber specification [SAB+20] for details on the algorithms and security proofs.

5.2 Kyber PKE

Written as the `K-PKE` scheme in the ML-KEM specification [MLK24], the Kyber PKE scheme is a public key encryption scheme based on the Module-LWE problem, similar to the one presented in Sec. 4.3.4. Note that this scheme is not IND-CCA secure, but rather IND-CPA secure, and thus, it is not to be used in a standalone manner, but rather as a building block for the Kyber KEM scheme.

5.2.1 Simplified Kyber PKE

We begin constructing the `K-PKE` scheme by recalling the PKE scheme presented in Sec. 4.3.4, which brings us the necessary building blocks for the actual subroutine used in the ML-KEM specification. Here we are considering the M-LWE$_{\ell,k,q,f(X),\chi}$ problem, where ℓ is the number of samples, k is the rank of the module, q is the modulus, $f(X)$ is the polynomial defining the ring $R_q = \mathbb{Z}_q[X]/(f(X))$ (with $f(X) = X^n + 1$), and χ is the secret/error distribution. Since in all implementations of Kyber / ML-KEM, the ring R_q is the same ($f(X) = X^n + 1$), and the module rank and the number of samples are the same ($k = \ell$), we will denote the problem as M-LWE$_{k,k,q,\chi}$.

Key Generation

Private Key: Sample a vector

$$\boldsymbol{s} \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k,$$

where each component s_i is sampled independently from the distribution χ over $\mathcal{R}_q$.

Public Key: Sample a matrix $\boldsymbol{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k\times k}$, and a small error vector $\boldsymbol{e} \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k$. The public key is then given by

$$(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) \in \mathcal{R}_q^{k\times k} \times \mathcal{R}_q^k.$$

Algorithm 3 `Simp-PKE.KeyGen()`

Output: Secret key $\mathsf{sk} = \boldsymbol{s} \in \mathcal{R}_q^k$
Output: Public key $\mathsf{pk} = (\boldsymbol{A}, \boldsymbol{b}) \in \mathcal{R}_q^{k \times k} \times \mathcal{R}_q^k$
1: $\boldsymbol{s} \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k$ ▷ Sample secret vector $\boldsymbol{s} \in \mathcal{R}_q^k$ from the error distribution χ
2: $\boldsymbol{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k \times k}$ ▷ Sample a uniformly random matrix $\boldsymbol{A} \in \mathcal{R}_q^{k \times k}$
3: $\boldsymbol{e} \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k$ ▷ Sample a small error vector $\boldsymbol{e} \in \mathcal{R}_q^k$ from the error distribution χ
4: $\boldsymbol{b} \leftarrow \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$ ▷ Compute the public key $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \in \mathcal{R}_q^k$
5: $\mathsf{sk} := \boldsymbol{s}$
6: $\mathsf{pk} := (\boldsymbol{A}, \boldsymbol{b})$
7: **return** $(\mathsf{sk}, \mathsf{pk})$

The algorithm can be summarized as follows:

$$((\boldsymbol{A}, \boldsymbol{b}), \boldsymbol{s}) = (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \texttt{Simp-PKE.KeyGen()}$$

Encryption

To encrypt a message $m \in \{0,1\}^n$, we view the message as an element of the ring $\mu \in R_q$, and we sample three small elements $\boldsymbol{r}, \boldsymbol{e}_1 \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k$ and $e_2 \leftarrow\!\!\$\ \chi \in \mathcal{R}_q$. The ciphertext is generated as:

$$\begin{aligned} \boldsymbol{u}^T &= \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T, \\ v &= \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu, \end{aligned}$$

Output the ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$.

Algorithm 4 `Simp-PKE.Enc`(pk, μ)

Input: Public key $\mathsf{pk} = (\boldsymbol{A}, \boldsymbol{b}) \in \mathcal{R}_q^{k \times k} \times \mathcal{R}_q^k$
Input: Message $\mu \in \{0,1\}^n$
Output: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$
1: $\boldsymbol{r} \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k$ ▷ Sample a small random vector $\boldsymbol{r} \in \mathcal{R}_q^k$ from the error distribution χ
2: $\boldsymbol{e}_1 \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k$ ▷ Sample a small error vector $\boldsymbol{e}_1 \in \mathcal{R}_q^k$ from the error distribution χ
3: $e_2 \leftarrow\!\!\$\ \chi \in \mathcal{R}_q$ ▷ Sample a small error $e_2 \in \mathcal{R}_q$ from the error distribution χ
4: $\boldsymbol{u}^T \leftarrow \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T$ ▷ Compute $\boldsymbol{u}^T = \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T \in \mathcal{R}_q^k$
5: $v \leftarrow \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu$ ▷ Compute $v = \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu \in \mathcal{R}_q$
6: $\boldsymbol{c} := (\boldsymbol{u}^T, v)$
7: **return** $\boldsymbol{c}$

The algorithm is summarized as follows:

$$\boldsymbol{c} = (\boldsymbol{u}^T, v) \leftarrow\!\!\$\ \texttt{Simp-PKE.Enc}(\mathsf{pk}, \mu)$$

Decryption

To decrypt the ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$, using the secret key $\boldsymbol{s}$, compute:

$$\begin{aligned} v - \boldsymbol{u}^T \cdot \boldsymbol{s} &= \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu - (\boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T) \cdot \boldsymbol{s} \\ &= \boldsymbol{r}^T \cdot (\boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}) + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu - (\boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T) \cdot \boldsymbol{s} \\ &= (\cancel{\boldsymbol{r}^T \boldsymbol{A}\boldsymbol{s}} + \boldsymbol{r}^T \boldsymbol{e} + e_2 - \cancel{\boldsymbol{r}^T \boldsymbol{A}\boldsymbol{s}} - \boldsymbol{e}_1^T \boldsymbol{s}) + \lfloor \frac{q}{2} \rceil \cdot \mu \\ &= (\boldsymbol{r}^T \boldsymbol{e} + e_2 - \boldsymbol{e}_1^T \boldsymbol{s}) + \lfloor \frac{q}{2} \rceil \cdot \mu \end{aligned}$$

Since all parts that make the error term $(\boldsymbol{r}^T \boldsymbol{e} + e_2 - \boldsymbol{e}_1^T \boldsymbol{s}) \in \mathcal{R}_q$ are small, it should have coefficients with magnitude less than $q/4$, and the bits of the message μ can be recovered by rounding each coefficient of the polynomial to either 0 or $\lfloor \frac{q}{2} \rceil$.

Algorithm 5 `Simp-PKE.Dec(sk, c)`

Input: Secret key $\mathsf{sk} = \boldsymbol{s} \in \mathcal{R}_q^k$
Input: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$
Output: Message $\mu \in \{0,1\}^n$

1: $(\boldsymbol{u}^T, v) \leftarrow \boldsymbol{c}$
2: $w \leftarrow v - \boldsymbol{u}^T \cdot \boldsymbol{s}$ ▷ Compute $v - \boldsymbol{u}^T \cdot \boldsymbol{s} \in \mathcal{R}_q$
3: $\mu \leftarrow \text{Round}(w)$ ▷ Round each coefficient of w to either 0 or $\lfloor \frac{q}{2} \rceil$, recovering the message $\mu \in \{0,1\}^n$
4: **return** μ

The algorithm is summarized as follows:

$$\mu \leftarrow \mathtt{Simp\text{-}PKE.Dec}(\mathsf{sk}, \boldsymbol{c})$$

5.2.2 Differences from Previous Schemes

The Simplified Kyber PKE scheme presents a trade-off between the previous schemes (LWE at Sec. 4.1.1 and Ring-LWE at Sec. 4.2.3), as it was already discussed in the end of Sec. 4.3.4. However, this implementation also allows for flexibility, since with a single fixed polynomial ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, the rank k of the module can be adjusted to the desired security level, from $k = 1$ (Ring-LWE), highly structured, to some $k > 1$, increasing actual dimension of the underlying lattice to $N = kn$, increasing the security level, albeit also increasing the size of the public key and ciphertext, but without needing to change the proof of security and ring structure.

5.3 Security

The proof of security can be constructed using a series of game hops, proving the IND-CPA security of the simplified scheme. The following Lemma follows the one presented in the Kyber Specification [SAB+20], where the advantage of an algorithm is defined as in Def. 2.2.1.

Lemma 5.3.1 (Simplified Kyber-PKE IND-CPA Security)**.** *For any adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$, with roughly the same running time of $\mathcal{A}$, such that if $\mathcal{A}$ can break the IND-CPA security of the Simplified Kyber PKE scheme with non-negligible probability, then $\mathcal{B}$ can solve the Module-LWE problem with non-negligible probability.*

$$\mathsf{Adv}^{\mathrm{cpa}}_{K\text{-}PKE}(\mathcal{A}) \leqslant 2\mathsf{Adv}^{\mathrm{mlwe}}_{k+1,k,q,\chi}(\mathcal{B})$$

Proof. For the proof, consider the following games *Game*0, *Game*1, *Game*2, where *Game*0 is the real game, *Game*1 is the game where the adversary is given a random public key, and *Game*2 is the game where the adversary is also given a random ciphertext. We also show the procedure for the Module-LWE problem, which is used to construct the adversary $\mathcal{B}$.

First lets begin with the Decision Game for the Module-LWE problem (D-M-LWE), which is defined as follows:

Decision Game for Module-LWE D-M-LWE

$\mathrm{D\text{-}M\text{-}LWE}_{k,k,q,\chi}$

Challenger		**Adversary**
$\boldsymbol{s} \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k$		
$\boldsymbol{A} \leftarrow\!\!\$\ \mathcal{R}_q^{k\times k}, \boldsymbol{e} \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k$		
$b \leftarrow\!\!\$\ \{0,1\}$		
if $b = 0$:		
$\quad \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$		
else:		
$\quad \boldsymbol{b} \leftarrow\!\!\$\ \mathcal{R}_q^k$		
	$(\boldsymbol{A}, \boldsymbol{b})$ $\longrightarrow$	
	b' $\longleftarrow$	
Return $(b' = b)$		

In words, the challenger samples a secret vector $\boldsymbol{s}$, a uniformly random matrix $\boldsymbol{A}$ and a small error vector $\boldsymbol{e}$. Then, it samples a bit $b \in \{0,1\}$, and if $b = 0$, it computes $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e}$,

i.e. a valid Module-LWE sample, and if $b = 1$, it samples $\boldsymbol{b}$ uniformly at random. The adversary $\mathcal{A}$ is given the pair $(\boldsymbol{A}, \boldsymbol{b})$, and it outputs a guess b', which is either 0 or 1. The game returns true if the guess is correct.

Game 0: IND-CPA Game for Simplified Kyber PKE IND-CPA.`Simp-PKE`

Now, we show the IND-CPA game for the Simplified Kyber PKE scheme, which is defined as follows, this is the first game *Game*0:

Game 0: IND-CPA Game for `Simp-PKE`

Challenger		**Adversary** $\mathcal{A}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$$ `Simp-PKE.KeyGen()`		
	$\xrightarrow{\quad \mathsf{pk} \quad}$	
..... Challenge Phase		
	$\xleftarrow{\quad (\mu_0, \mu_1) \quad}$	
$b \leftarrow\!\!\$ \{0,1\}$		
$\boldsymbol{c}^* \leftarrow\!\!\$$ `Simp-PKE.Enc`(pk, μ_b)		
	$\xrightarrow{\quad \boldsymbol{c}^* \quad}$	
..... Guess Phase		
	$\xleftarrow{\quad b' \quad}$	
Return $(b' = b)$		

The adversary $\mathcal{A}$ is given the public key pk, and it outputs two messages $\mu_0, \mu_1 \in \{0,1\}^k$, and receives a challenge ciphertext $\boldsymbol{c}^* = (\boldsymbol{u}^T, v)$, which is an encryption of one of the messages μ_b. The adversary then outputs a guess b', which is either 0 or 1, whether it thinks the challenge ciphertext $\boldsymbol{c}^*$ is an encryption of μ_0 or μ_1. The game returns true if the guess is correct.

The first game *Game*0 is the real game, where the adversary $\mathcal{A}$ interacts with the challenger as described above.

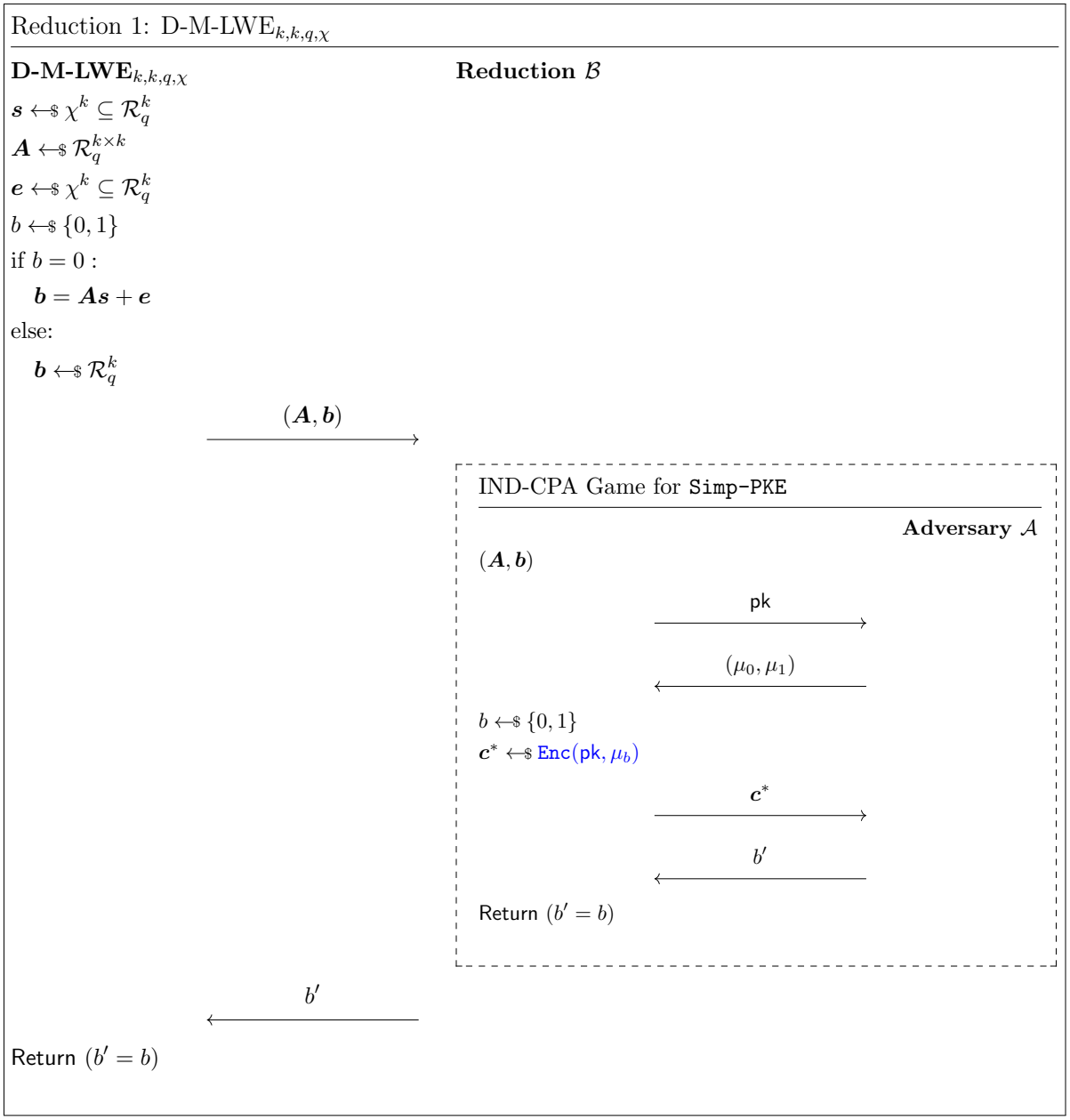
Game 1: IND-CPA Game for Simplified Kyber PKE IND-CPA.`Simp-PKE`

In the second game *Game*1, the adversary is given a random public key, and is defined as follows, where the difference is shown in teal:

Game 1: IND-CPA Game for `Simp-PKE`

Challenger		**Adversary** $\mathcal{A}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$$ `Simp-PKE.KeyGen()`		
$\mathsf{pk} \leftarrow\!\!\$ \mathcal{R}_q^{k\times k} \times \mathcal{R}_q^k$		
	$\xrightarrow{\quad \mathsf{pk} \quad}$	
..... Challenge Phase		
	$\xleftarrow{\quad (\mu_0, \mu_1) \quad}$	
$b \leftarrow\!\!\$ \{0,1\}$		
$\boldsymbol{c}^* \leftarrow\!\!\$$ `Simp-PKE.Enc`(pk, μ_b)		
	$\xrightarrow{\quad \boldsymbol{c}^* \quad}$	
..... Guess Phase		
	$\xleftarrow{\quad b' \quad}$	
Return $(b' = b)$		

Reduction 1 to Module-LWE Now consider the first step of comparing the reduction to the Decision Module-LWE problem to both Games *Game*0 and *Game*1. Let the Challenger be the D-M-LWE$_{k,k,q,\chi}$ challenger, let $\mathcal{B}$ be the reduction and let $\mathcal{A}$ be the adversary that breaks the IND-CPA security of the Simplified Kyber PKE scheme. The reduction $\mathcal{B}$ interacts with the challenger as follows:



The reduction above shows that the adversary $\mathcal{A}$ can be used as a subroutine in the decision game for the Module-LWE problem, where the reduction $\mathcal{B}$ directly forwards the public key pk received from the challenger to the adversary $\mathcal{A}$, and then encrypts either μ_0 or μ_1 depending on the bit b sampled and forwards the ciphertext $\boldsymbol{c}^*$ to the adversary. The reduction then receives the guess b' from the adversary, and then returns true if the guess is correct, i.e. if $b' = b$. Whenever the D-M-LWE game forwards the LWE sample $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e})$ to the reduction $\mathcal{B}$, the adversary $\mathcal{A}$ sees exactly the game *Game*0, which is the original IND-CPA game for the Simplified Kyber PKE, and whenever the D-M-LWE game forwards a random sample $(\boldsymbol{A}, \boldsymbol{b})$ to the reduction $\mathcal{B}$, the adversary $\mathcal{A}$ sees exactly the game *Game*1.

Thus, we can conclude that the advantage of the reduction $\mathcal{B}$ in solving the Module-LWE problem is given by:

$$\begin{aligned} \mathsf{Adv}^{\mathrm{d-mlwe}}_{k,k,q,\chi}(\mathcal{B}) &= \left| \Pr\left[b' = b \mid \text{LWE Sample}\right] - \Pr\left[b' = b \mid \text{Random Sample}\right] \right| && (5.1) \\ &= \left| \quad p_0 \quad - \quad p_1 \quad \right| && (5.2) \end{aligned}$$

where p_0 and p_1 are the probabilities of the adversary $\mathcal{A}$ winning the IND-CPA game for the Simplified Kyber PKE scheme in the case of a LWE sample (i.e. *Game*0) and a random sample (i.e. *Game*1) for the public key, respectively.

Game 2: IND-CPA Game for Simplified Kyber PKE IND-CPA.`Simp-PKE`

In the next hop, game *Game*2, the adversary is given a random ciphertext as well as a random public key from the *Game*1, and is defined as follows, where the difference is shown in purple from the previous game *Game*1:

Game 2: IND-CPA Game for `Simp-PKE`

Challenger		**Adversary** $\mathcal{A}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$$ `Simp-PKE.KeyGen()`		
$\mathsf{pk} \leftarrow\!\!\$ \mathcal{R}_q^{k\times k} \times \mathcal{R}_q^k$		
	$\xrightarrow{\quad \mathsf{pk} \quad}$	
.....	Challenge Phase	.....
	$\xleftarrow{\quad (\mu_0, \mu_1) \quad}$	
$b \leftarrow\!\!\$ \{0,1\}$		
$\boldsymbol{c}^* \leftarrow\!\!\$ \mathcal{R}_q^k \times \mathcal{R}_q$		
	$\xrightarrow{\quad \boldsymbol{c}^* \quad}$	
.....	Guess Phase	.....
	$\xleftarrow{\quad b' \quad}$	
$\mathsf{Return}\ (b' = b)$		

Note that in the game *Game*1, the adversary is given the correct challenge ciphertext

$$\begin{aligned}\boldsymbol{u}^T &= \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T,\\ v &= \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu,\\ \boldsymbol{c}^* &= (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q,\end{aligned}$$

Let:

$$\hat{\boldsymbol{A}} = \begin{pmatrix} \boldsymbol{A}^T \\ \boldsymbol{b}^T \end{pmatrix} \leftarrow\!\$\, \mathcal{R}_q^{(k+1)\times k},$$

since in *Game1*, $\boldsymbol{b}$ is already uniformly random $\boldsymbol{b} \leftarrow\!\$\, \mathcal{R}_q^k$, therefore, we can rewrite the ciphertext as:

$$\begin{aligned}\boldsymbol{c}^* = \begin{pmatrix} \boldsymbol{u} \\ v \end{pmatrix} &= \begin{pmatrix} \boldsymbol{A}^T \\ \boldsymbol{b}^T \end{pmatrix} r + \begin{pmatrix} \boldsymbol{e}_1 \\ e_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \lfloor \frac{q}{2} \rceil \cdot \mu \end{pmatrix}\\ &= \hat{\boldsymbol{A}} \cdot \boldsymbol{r} + \begin{pmatrix} \boldsymbol{e}_1 \\ e_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \lfloor \frac{q}{2} \rceil \cdot \mu \end{pmatrix}\\ &= \hat{\boldsymbol{A}} \cdot \boldsymbol{r} + \hat{\boldsymbol{e}} + \begin{pmatrix} 0 \\ \lfloor \frac{q}{2} \rceil \cdot \mu \end{pmatrix},\end{aligned}$$

where, the ciphertext is a valid Module-LWE sample, added to some 'one-time pad' message $\lfloor \frac{q}{2} \rceil \cdot \mu$, and therefore we can represent the sample as:

$$(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}} := \hat{\boldsymbol{A}}\boldsymbol{r} + \hat{\boldsymbol{e}}), \qquad \text{where } \hat{\boldsymbol{b}} = \begin{pmatrix} \boldsymbol{u} \\ \tilde{v} \end{pmatrix},$$

where, $\tilde{v} = \boldsymbol{b}^T \cdot \boldsymbol{r} + e_2$ is the part of the ciphertext without the message.

Now, in the game *Game2*, the adversary is given a random ciphertext $\boldsymbol{c}^* \in \mathcal{R}_q^{k+1}$, which can be seen as part of the Module-LWE sample ($\hat{\boldsymbol{b}}$) however, this time, completely random, i.e. $\hat{\boldsymbol{b}} \leftarrow\!\$\, \mathcal{R}_q^{k+1}$, and thus, the adversary is given a random Module-LWE sample $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}) \in \mathcal{R}_q^{k+1\times k} \times \mathcal{R}_q^{k+1}$, this time with an extra sample $(k+1)$.

The intuition then tells us that if the adversary $\mathcal{A}$ can distinguish between LWE samples and random samples, it can also distinguish between the ciphertexts $\boldsymbol{c}^*$ and random ciphertexts, so we can also use a similar reduction to the Module-LWE problem, now with an extra sample $(k+1)$.

Reduction 2 to Module-LWE Now consider the next step of comparing the reduction to the Decision Module-LWE problem to both Games *Game1* and *Game2*. Note that the difference now is that the adversary is given a random ciphertext as well, besides the random public key, and from the discussion above, we can see that the ciphertext in the

game *Game1* is a valid Module-LWE sample, and the ciphertext in the game *Game2* is a random Module-LWE sample, both with extra samples.

From Game1 onward, $(\boldsymbol{A}, \boldsymbol{b})$ *is already uniform. In the next reduction we only* reuse *the uniform challenge matrix* $\hat{\boldsymbol{A}}$ *to* couple *the ciphertext with the pk (so that* $\boldsymbol{u} = \boldsymbol{A}^T\boldsymbol{r} + \boldsymbol{e}_1$*); this does not change the pk distribution.*

Consider the following reduction:

Reduction 2: D-M-LWE$_{k+1,k,q,\chi}$

D-M-LWE$_{k+1,k,q,\chi}$

$\boldsymbol{s} \leftarrow\!\!\$\ \chi^k \subseteq \mathcal{R}_q^k$

$\hat{\boldsymbol{A}} \leftarrow\!\!\$\ \mathcal{R}_q^{(k+1)\times k}$

$\boldsymbol{e} \leftarrow\!\!\$\ \chi^{k+1} \subseteq \mathcal{R}_q^{k+1}$

$b \leftarrow\!\!\$\ \{0,1\}$

if $b = 0$:

$\quad \hat{\boldsymbol{b}} = \hat{\boldsymbol{A}}\boldsymbol{s} + \hat{\boldsymbol{e}}$

else:

$\quad \hat{\boldsymbol{b}} \leftarrow\!\!\$\ \mathcal{R}_q^{k+1}$

$(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})$ →

Reduction $\mathcal{B}$

IND-CPA Game for `Simp-PKE`

Adversary $\mathcal{A}$

$\begin{pmatrix} \boldsymbol{A}^T \\ \boldsymbol{b}^T \end{pmatrix} = \hat{\boldsymbol{A}}$

$\begin{pmatrix} \boldsymbol{u} \\ \tilde{v} \end{pmatrix} = \hat{\boldsymbol{b}}$

$(\boldsymbol{A}, \boldsymbol{b})$ →

← (μ_0, μ_1)

$b \leftarrow\!\!\$\ \{0,1\}$

$v = \tilde{v} + \lfloor \frac{q}{2} \rceil \cdot \mu_b$

$\boldsymbol{c}^* = (\boldsymbol{u}^T, v)$

$\boldsymbol{c}^*$ →

← b'

Return $(b' = b)$

← b'

Return $(b' = b)$

Because $(\boldsymbol{A}, \boldsymbol{b})$ *is uniform in Game1, reusing the top* k *rows of* $\hat{\boldsymbol{A}}$ *gives us an identical distribution for pk; it only preserves the conditional law of* $\boldsymbol{u}$ *given* $\boldsymbol{A}$*. Thus in the* $b = 0$ *case the ciphertext is real, while in* $b = 1$ *it is uniform and independent of the adversary's guess.*

The reduction above shows the adversary $\mathcal{A}$ being used as a subroutine in the decision game for the Module-LWE problem (with an extra sample $(k+1)$), where the reduction $\mathcal{B}$ uses the initial matrix $\hat{\boldsymbol{A}}$ received from the challenger to break it down to the original public key $\boldsymbol{A}, \boldsymbol{b}$ and also uses the $\hat{\boldsymbol{b}}$ to create the ciphertext $\boldsymbol{c}^* = (\boldsymbol{u}^T, v)$, where $\boldsymbol{u} = \boldsymbol{A}^T\boldsymbol{r} + \boldsymbol{e}_1$ and $v = \tilde{v} + \lfloor \frac{q}{2} \rceil \cdot \mu_b$, and $\tilde{v}$ is the part of the ciphertext without the message. The reduction then receives the guess b' from the adversary, and returns true if the guess is correct, i.e. if $b' = b$.

Whenever the D-M-LWE$_{k+1}$ game forwards the **LWE sample** $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}} := \hat{\boldsymbol{A}}\boldsymbol{s} + \hat{\boldsymbol{e}})$, since $\hat{\boldsymbol{A}}$ is a random matrix, then the public key $(\boldsymbol{A}, \boldsymbol{b})$ is is also random, and the ciphertext is correctly generated, and thus the adversary $\mathcal{A}$ sees exactly the game *Game1*. Whenever the challenger forwards the completely **random sample** $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})$, both the public key and ciphertext given to the adversary $\mathcal{A}$ are random, and thus the adversary $\mathcal{A}$ sees exactly the game *Game2* and cannot distinguish between the two games apart from random guessing, i.e. $p_2 = \Pr[b' = b \mid \text{Random Sample on Reduction 2}] = \frac{1}{2}$.

Thus, we can conclude that the advantage of the reduction $\mathcal{B}$ in solving the D-M-LWE$_{k+1}$ problem is given by:

$$\begin{aligned} \mathsf{Adv}^{\mathrm{d-mlwe}}_{k+1,k,q,\chi}(\mathcal{B}) &= \left| \Pr[b' = b \mid \text{LWE Sample}] - \Pr[b' = b \mid \text{Random Sample}] \right| && (5.3) \\ &= \left| \qquad p_1 \qquad - \qquad p_2 \qquad \right| && (5.4) \end{aligned}$$

where, p_1 and p_2 are the probabilities of the adversary $\mathcal{A}$ winning the IND-CPA game for the Simplified Kyber PKE scheme in the case of a proper constructed ciphertext (i.e.*Game1*) and a random ciphertext (i.e. *Game2*), respectively.

Consider the following Lemma about the relation between the two advantages for an adversary $\mathcal{B}$ $\mathsf{Adv}^{\mathrm{d-mlwe}}_{k,k,q,\chi}(\mathcal{B})$ and $\mathsf{Adv}^{\mathrm{d-mlwe}}_{k+1,k,q,\chi}(\mathcal{B})$.

Lemma 5.3.2. *Let* n, k, q *be integers and and let* χ *be a distribution over* $\mathcal{R}_q$*. For every pair of integers* $\ell < \ell'$ *and for every distinguisher* $\mathcal{A}$ *there exists a reduction* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathrm{d-mlwe}}_{\ell,k,q,\chi}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathrm{d-mlwe}}_{\ell',k,q,\chi}(\mathcal{B}).$$

Proof. On input $(\boldsymbol{A}, \boldsymbol{b}) \in \mathcal{R}_q^{\ell' \times k} \times \mathcal{R}_q^{\ell'}$ (the challenge for the ℓ'-sample MLWE game), the distinguisher $\mathcal{B}$ keeps the *first* ℓ rows of $\boldsymbol{A}$ and the first ℓ coordinates of $\boldsymbol{b}$. Denote the truncated pair by $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}) \in \mathcal{R}_q^{\ell \times k} \times \mathcal{R}_q^{\ell}$. It then runs $\mathcal{A}(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}})$ and outputs the same bit.

If $(\boldsymbol{A}, \boldsymbol{b})$ is an M-LWE$_{\ell',k,q,\chi}$ sample, then, because χ is a product distribution—$(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}})$ is distributed exactly as an MLWE(ℓ, k, q, χ) sample. If $(\boldsymbol{A}, \boldsymbol{b})$ is uniform, the truncation is still uniform.

Therefore $\Pr[\mathcal{B} \text{ outputs } 1 \mid \text{LWE sample}] = \Pr[\mathcal{A} \text{ outputs } 1 \mid \text{LWE sample}]$ and the same holds for the random case, so both games give the *same* advantage value.

Hence, with a distinguisher $\mathcal{A}$ that breaks the ℓ-sample MLWE game, we can construct a reduction $\mathcal{B}$ that breaks the ℓ'-sample MLWE game, or, the ℓ-sample MLWE game is at least as hard as the ℓ'-sample MLWE game, and thus we have, for every pair of integers $\ell < \ell'$:

$$\mathsf{Adv}^{\mathrm{d-mlwe}}_{\ell,k,q,\chi}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathrm{d-mlwe}}_{\ell',k,q,\chi}(\mathcal{B}).$$

□

Now, we can combine the two reductions, and conclude that:

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{ind-cpa}}_{\texttt{Simp-PKE}}(\mathcal{A}) &= |p_0 - \frac{1}{2}| \\
&\leqslant |p_0 - p_1| + |p_1 - \frac{1}{2}| && \text{(Triangle Inequality)} \\
&\leqslant |p_0 - p_1| + |p_1 - p_2| \\
&\leqslant \mathsf{Adv}^{\mathrm{d-mlwe}}_{k,k,q,\chi}(\mathcal{B}) + \mathsf{Adv}^{\mathrm{d-mlwe}}_{k+1,k,q,\chi}(\mathcal{B}) && \text{(from Eq. (5.2) and Eq. (5.4))} \\
&\leqslant 2\mathsf{Adv}^{\mathrm{d-mlwe}}_{k+1,k,q,\chi}(\mathcal{B}) && \text{(from Lemma 5.3.2)}
\end{aligned}$$

□

5.4 Optimizations

Now that we have the IND-CPA security of the Simplified Kyber PKE scheme, we can introduce some optimizations to the scheme, which do not change the overall security proof, or understanding of the scheme, but rather improve the performance of the scheme, i.e. the size of the public key and ciphertext, as well as the speed of the encryption and decryption algorithms.

5.4.1 Public Key Generation

The first step is to generate the public key matrix $\boldsymbol{A}$ from a seed, instead of sampling it uniformly at random. This is done in the ML-KEM using an extendable output function (XOF) to generate the matrix $\boldsymbol{A}$ from a seed, which is a fixed-length input, making it much smaller than the matrix itself and thus reducing the size of the public key.

If the matrix $\boldsymbol{A} \in \mathcal{R}_q^{k \times k}$ is sent directly, it is of size $k^2 \cdot n \cdot \log(q)$ (i.e. $(2^2 \cdot 256 \cdot \log(2^{12})) = 1536$ bytes for `ML-KEM-512`), while, if we use a seed ρ and expand it, the size is only s (e.g. $s = 32$ bytes). Deriving $\boldsymbol{A}$ from a seed via an XOF is computationally cheap (and typically faster than obtaining truly uniform bits); its cost is negligible compared with the NTT and polynomial arithmetic. The practical trade-off is a tiny recomputation cost to regenerate $\boldsymbol{A}$ versus a large reduction in public-key size.

The algorithm for generating the public key matrix $\boldsymbol{A}$ from a seed ρ is as follows, with the difference from the previously shown algorithms (Alg. 3 (`Simp-PKE.KeyGen()`) and Alg. 4 (`Simp-PKE.Enc(pk, μ)`)) in orange:

Algorithm 6 `Simp-PKE.KeyGen()` (with public key matrix $\boldsymbol{A}$ generated from a seed)

Input: Seed ρ ▷ Seed for the public key matrix $\boldsymbol{A}$
Output: Secret key $\mathsf{sk} = \boldsymbol{s} \in \mathcal{R}_q^k$
Output: Public key $\mathsf{pk} = (\rho, \boldsymbol{b})$

1: $\boldsymbol{s} \leftarrow\$ \chi^k \subseteq \mathcal{R}_q^k$ ▷ Sample secret vector $\boldsymbol{s} \in \mathcal{R}_q^k$ from the error distribution χ
2: $\boldsymbol{A} \leftarrow \mathsf{XOF}(\rho)$ ▷ Generate uniformly random matrix $\boldsymbol{A} \in \mathcal{R}_q^{k \times k}$ from the seed ρ using a XOF
3: $\boldsymbol{e} \leftarrow\$ \chi^k \subseteq \mathcal{R}_q^k$ ▷ Sample a small error vector $\boldsymbol{e} \in \mathcal{R}_q^k$ from the error distribution χ
4: $\boldsymbol{b} \leftarrow \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$ ▷ Compute the public key $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \in \mathcal{R}_q^k$
5: $\mathsf{sk} := \boldsymbol{s}$
6: $\mathsf{pk} := (\rho, \boldsymbol{b})$
7: **return** $(\mathsf{sk}, \mathsf{pk})$

Algorithm 7 `Simp-PKE.Enc(`pk, μ`)` (with public key matrix $\boldsymbol{A}$ generated from a seed)

Input: Public key $\mathsf{pk} = (\rho, \boldsymbol{b})$
Input: Message $\mu \in \{0,1\}^n$
Output: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$

1: $\boldsymbol{r} \leftarrow\$ \chi^k \subseteq \mathcal{R}_q^k$ ▷ Sample a small random vector $\boldsymbol{r} \in \mathcal{R}_q^k$ from the error distribution χ
2: $\boldsymbol{e}_1 \leftarrow\$ \chi^k \subseteq \mathcal{R}_q^k$ ▷ Sample a small error vector $\boldsymbol{e}_1 \in \mathcal{R}_q^k$ from the error distribution χ
3: $e_2 \leftarrow\$ \chi \in \mathcal{R}_q$ ▷ Sample a small error $e_2 \in \mathcal{R}_q$ from the error distribution χ
4: $\boldsymbol{A} \leftarrow \mathsf{XOF}(\rho)$ ▷ Expand matrix $\boldsymbol{A} \in \mathcal{R}_q^{k \times k}$ from the seed ρ using a XOF
5: $\boldsymbol{u}^T \leftarrow \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T$ ▷ Compute $\boldsymbol{u}^T = \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T \in \mathcal{R}_q^k$
6: $v \leftarrow \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu$ ▷ Compute $v = \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu \in \mathcal{R}_q$
7: $\boldsymbol{c} := (\boldsymbol{u}^T, v)$
8: **return** $\boldsymbol{c}$

5.4.2 Secret and Error Sampling

All of the secret and error terms in the PKE scheme are sampled from a Centered Binomial Distribution (CBD) instead of sampling from the error distribution χ directly. This is done to ensure that the sampled vectors are small, but also to ensure that the sampling is efficient and easier to implement in practice. The CBD sampling is done using a function CBD, which is defined as follows:

Definition 5.4.1 (Centered Binomial Distribution: CBD_η)**.** *Let η be a positive integer. An element from the Centered Binomial Distribution CBD_η is generated by sampling 2η independent uniform bits $a_1, \ldots, a_\eta, b_1, \ldots, b_\eta \leftarrow\$ \{0,1\}$ and outputting:*

$$\sum_{i=1}^{\eta} a_i - \sum_{i=1}^{\eta} b_i \ mod \ q$$

If the output is an element of the ring $\mathcal{R}_q$, we simply repeat the sampling for every coefficient of the polynomial, i.e. n times, and output the polynomial as a vector of coefficients.

Note that the values η for the ML-KEM scheme are chosen such that the sampled vectors are small enough to ensure that the probability of decription failure is negligible, and are usually chosen to be either 2 or 3, depending on the security level and the parameters of the scheme. This results in a vector of coefficients where each coefficient is in the range $[-\eta, \eta]$. It The value η is also separated into two parts, η_1 and η_2, where η_1 is the value used for the key generation and secret sampling (i.e. the secret key $\boldsymbol{s}$, the error vector $\boldsymbol{e}$ and the random secret $\boldsymbol{r}$), and η_2 is the value used for the error sampling (i.e. $\boldsymbol{e}_1$ and e_2 in the encryption algorithm).

Algorithm 8 `Simp-PKE.KeyGen()` (with small vectors $\boldsymbol{s}, \boldsymbol{e}$ generated by CBD)

Input: Seed ρ ▷ Seed for the public key matrix $\boldsymbol{A}$
Output: Secret key $\mathsf{sk} = \boldsymbol{s} \in \mathcal{R}_q^k$
Output: Public key $\mathsf{pk} = (\rho, \boldsymbol{b})$

1: **for** $i = 0$ to $k - 1$ **do**
2: $\quad \boldsymbol{s}_i \leftarrow\$ \mathsf{CBD}_{\eta_1}$ ▷ Sample secret vector $\boldsymbol{s} \in \mathcal{R}_q^k$ from CBD_{η_1}

3: $\boldsymbol{A} \leftarrow \mathsf{XOF}(\rho)$
▷ Generate uniformly random matrix $\boldsymbol{A} \in \mathcal{R}_q^{k \times k}$ from the seed ρ using a XOF

4: **for** $i = 0$ to $k - 1$ **do**
5: $\quad \boldsymbol{e}_i \leftarrow\$ \mathsf{CBD}_{\eta_1}$ ▷ Sample a small error vector $\boldsymbol{e} \in \mathcal{R}_q^k$ from CBD_{η_1}

6: $\quad \boldsymbol{b} \leftarrow \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$ ▷ Compute the public key $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \in \mathcal{R}_q^k$

7: $\quad \mathsf{sk} := \boldsymbol{s}$
8: $\quad \mathsf{pk} := (\rho, \boldsymbol{b})$
9: $\quad$ **return** $(\mathsf{sk}, \mathsf{pk})$

Algorithm 9 `Simp-PKE.Enc`(pk, μ) (with small vectors $\boldsymbol{s}, \boldsymbol{e}$ generated by CBD)

Input: Public key $\mathsf{pk} = (\rho, \boldsymbol{b})$
Input: Message $\mu \in \{0,1\}^n$
Output: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$

1: **for** $i = 0$ to $k-1$ **do**
2: $\quad \boldsymbol{r}_i \leftarrow\$ \mathsf{CBD}_{\eta_1}$ ▷ Sample a small random vector $\boldsymbol{r} \in \mathcal{R}_q^k$ from CBD_{η_1}
3: **for** $i = 0$ to $k-1$ **do**
4: $\quad \boldsymbol{e}_{1,i} \leftarrow\$ \mathsf{CBD}_{\eta_2}$ ▷ Sample a small error vector $\boldsymbol{e}_1 \in \mathcal{R}_q^k$ from CBD_{η_2}
5: $e_2 \leftarrow\$ \mathsf{CBD}_{\eta_2}$ ▷ Sample a small error $e_2 \in \mathcal{R}_q$ from CBD_{η_2}

6: $\boldsymbol{A} \leftarrow \mathsf{XOF}(\rho)$ ▷ Expand matrix $\boldsymbol{A} \in \mathcal{R}_q^{k \times k}$ from the seed ρ using a XOF

7: $\boldsymbol{u}^T \leftarrow \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T$ ▷ Compute $\boldsymbol{u}^T = \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T \in \mathcal{R}_q^k$
8: $v \leftarrow \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu$ ▷ Compute $v = \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu \in \mathcal{R}_q$

9: $\boldsymbol{c} := (\boldsymbol{u}^T, v)$
10: **return** $\boldsymbol{c}$

5.4.3 NTT Multiplication

While the implementation of the ring $\mathcal{R}_q$ brings some performance improvements in terms of the size of the public key and ciphertext, the multiplication of polynomials in the ring $\mathcal{R}_q$, as shown in Sec. 4.2.1, is not very efficient. As discussed on Sec. 4.3.1, the representation of the product $\boldsymbol{As}$, where $\boldsymbol{A} \in \mathcal{R}_q^{k \times k}$ and $\boldsymbol{s} \in \mathcal{R}_q^k$, is done by multiplying the matrix $\mathcal{M}_{\boldsymbol{A}} \in \mathbb{Z}_q^{nk \times nk}$ with the vector $\mathcal{V}_{\boldsymbol{s}} \in \mathbb{Z}_q^{nk}$, where n is the degree of the polynomial, and k is the rank of the module, which results in a multiplication on the order of $O(n^2)$.

To improve the performance of the multiplication of polynomials in the ring $\mathcal{R}_q$, we can use the Number Theoretic Transform (NTT), which is a special case of the FFT (Fast Fourier Transform) over the field $\mathbb{Z}_q^*$, instead of the complex numbers, and reduce the multiplication to component-wise products of the transformed coefficients (frequency domain), which can be done in $O(n \log n)$ time. This is done by exploiting the structure of the ring $\mathcal{R}_q$, and more specifically the choice of the modulus q and the function $f(X) = X^n + 1$, which is irreducible over $\mathbb{Z}_q$, and thus allows us to compute the NTT of the polynomials in the ring $\mathcal{R}_q$ efficiently.

Roots of unity and choice of paramenters

Let n be a power of 2 and let q be a prime such that $q \equiv 1 \bmod n$, then the multiplicative group $\mathbb{Z}_q^*$ has an element ζ of order n, which is a primitive n-th root of unity, i.e. $\zeta^n \equiv 1 \bmod q$ and $\zeta^k \neq 1 \bmod q$ for $0 < k < n$. Note here that $\zeta^{n/2} = -1$.

Therefore, this permits a binary factorization of the polynomial $f(X) = X^n + 1$ into quadratic factors as:

$$X^n + 1 = \prod_{i=0}^{n/2-1} (X^2 - \zeta^{2i+1}),$$

where, by applying the Chinese Remainder Theorem, it provides an isomorphism between the ring $\mathcal{R}_q$ and the product of the quadratic rings $\mathcal{Q}_i$ as:

$$\mathcal{R}_q \approx \prod_{i=0}^{n/2-1} \mathbb{Z}_q[X]/(X^2 - \zeta^{2i+1}).$$

Example: For clarity, consider the following example, with $q = 137$ and $n = 2^3 = 8$, where n divides $q - 1 = 136 = 8 \cdot 17$, and $2n = 16$ does not.

The primitive 8-th root of unity is $\zeta = 10$, which satisfies $\zeta^8 \equiv 1 \bmod 137$ and therefore $\zeta^4 = -1 \bmod 137$.

The polynomial $f(X) = X^8 + 1$ can be factored as:

$$\begin{aligned} f(X) &= X^8 + 1 \\ &= (X^8 - \zeta^4) \\ &= (X^4 - \zeta^2)(X^4 + \zeta^2) \\ &= (X^4 - \zeta^2)(X^4 - \zeta^6) && (\text{since } \zeta^6 \equiv (\zeta^4)(\zeta^2) \bmod 137) \\ &= (X^2 - \zeta)(X^2 + \zeta)(X^2 - \zeta^3)(X^2 + \zeta^3) \\ &= (X^2 - \zeta)(X^2 - \zeta^5)(X^2 - \zeta^3)(X^2 - \zeta^7) \end{aligned}$$

and thus, we have the isomorphism:

$$\mathcal{R}_q = \frac{\mathbb{Z}_{137}[X]}{X^8 + 1} \approx \frac{\mathbb{Z}_{137}[X]}{X^2 - \zeta} \times \frac{\mathbb{Z}_{137}[X]}{X^2 - \zeta^3} \times \frac{\mathbb{Z}_{137}[X]}{X^2 - \zeta^5} \times \frac{\mathbb{Z}_{137}[X]}{X^2 - \zeta^7}$$

ML-KEM Parameters: In the ML-KEM scheme, the parameters are chosen such that $n = 2^8 = 256$ and $q = 3329$, where $q - 1 = 3328 = 13 \cdot 2^8$, and thus n divides $q - 1$, but not $2n$. The root of unity is $\zeta = 17$, which satisfies $\zeta^{256} \equiv 1$ and $\zeta^{128} \equiv -1 \bmod 3329$.

Definition

This then lets us define the NTT transform as a mapping from the ring $\mathcal{R}_q$ to the product of the quadratic rings $\mathcal{Q}_i$, which can be computed efficiently using the roots of unity ζ.

Definition 5.4.2 (NTT Transform: NTT)**.** *Let n be a power of 2 and let q be a prime such that $q \equiv 1$ mod n. Define $\mathcal{Q}_i = \mathbb{Z}_q[X]/(X^2 - \zeta^i)$ for odd i in the range $0 \leqslant i < n$, where ζ is a primitive n-th root of unity in $\mathbb{Z}_q^*$. Define $\mathcal{T}_q = \mathcal{Q}_1 \times \mathcal{Q}_2 \times \ldots \times \mathcal{Q}_{n-1}$.*

The NTT transform is defined as the function $\mathsf{NTT} : \mathcal{R}_q \to \mathcal{T}_q$*, which maps a polynomial* $a(X) \in \mathcal{R}_q$ *to the vector:*

$$\mathsf{NTT}(a) = \hat{a} = (a \ mod \ (X^2 - \zeta), a \ mod \ (X^2 - \zeta^3), \ldots, a \ mod \ (X^2 - \zeta^{n-1})).$$

where, the output is a vector of coefficients in the product of the quadratic rings $\mathcal{T}_q$*, i.e.* $\hat{a} \in \mathcal{T}_q$*.*

Remark (vector-space view)**.** *Both* $\mathcal{R}_q$ *and* $\mathcal{T}_q$ *are additively isomorphic to* $\mathbb{Z}_q^n$ *(via the coefficient embedding and the CRT, respectively). Equivalently, one can view* NTT *as a linear bijection* $\mathbb{Z}_q^n \to \mathbb{Z}_q^n$*: a coefficient vector* $(a_0, \ldots, a_{n-1})$ *maps to a vector that can be partitioned into* $n/2$ *pairs, where each pair is the coefficient vector of* $a \bmod (X^2 - \zeta^i)$ *in the corresponding quadratic quotient ring* $\mathcal{Q}_i$*.*

Note that in the ML-KEM scheme ([SAB+20]), we identify both domain and codomain with $\mathcal{R}_q$ via the coefficient embedding and store $\widehat{a}$ back as a polynomial; with this identification it is convenient to write $\mathsf{NTT} : \mathcal{R}_q \to \mathcal{R}_q$.

$$\mathsf{NTT}(a) = \hat{a} = \hat{a_0} + \hat{a_1}X + \ldots + \hat{a_{n-1}}X^{n-1},$$

where the resulting polynomial is then used as the canonical representation of the polynomial in the ring $\mathcal{R}_q$, i.e. $\hat{a} \in \mathcal{R}_q$, with the vector of coefficients $\hat{a} = (\hat{a_0}, \hat{a_1}, \ldots, \hat{a_{n-1}}) \in \mathbb{Z}_q^n$.

The NTT transform is also invertible, and thus we can define the inverse NTT transform that brings the vector of coefficients back to the polynomial in the ring $\mathcal{R}_q$.

$$\mathsf{NTT}^{-1}(\hat{a}) = a(X) = a_0 + a_1X + \ldots + a_{n-1}X^{n-1}$$

NTT Operations

Both addition and multiplication of polynomials in the ring $\mathcal{R}_q$ can be done efficiently using the NTT transform, as both operations can be reduced to component-wise operations on the transformed coefficients. In the normal ring $\mathcal{R}_q$, addition was already done component-wise, however, the real benefit comes from the multiplication of the polynomials.

Consider the two polynomials $a(X)$ and $b(X)$ in the ring $\mathcal{R}_q$, where $a(X) = a_0 + a_1X + \ldots + a_{n-1}X^{n-1}$ and $b(X) = b_0 + b_1X + \ldots + b_{n-1}X^{n-1}$, then the multiplication of the two polynomials can be done as follows:

$$\begin{aligned} a(X) \cdot b(X) &= (a_0 + a_1X + \ldots + a_{n-1}X^{n-1}) \cdot (b_0 + b_1X + \ldots + b_{n-1}X^{n-1}) \\ &= c(x) + r(x)(X^n + 1), \end{aligned}$$

which, by reducing modulo one of the quadratic factors $\mathcal{Q}_i$ with $X^2 - \zeta^i$, results in the following:

$$
\begin{aligned}
& a(X) \bmod \mathcal{Q}_i \cdot b(X) \bmod \mathcal{Q}_i = c(X) \bmod \mathcal{Q}_i + r(X)(X^n+1) \bmod \mathcal{Q}_i \\
\Rightarrow \quad & \hat{a_i} \quad \cdot \quad \hat{b_i} \quad = \quad \hat{c_i}
\end{aligned}
$$

Therefore, since this holds for every quadratic factor $\mathcal{Q}_i$, we can conclude that:

$$\hat{c} = (\hat{c_0}, \hat{c_1}, \ldots, \hat{c_{n-1}}) = (\hat{a_1} \cdot \hat{b_1}, \hat{a_3} \cdot \hat{b_3}, \ldots, \hat{a_{n-1}} \cdot \hat{b_{n-1}}) = \hat{a} \cdot \hat{b},$$

where each $\hat{a_i}$ is a degree $\leqslant 1$ polynomial in the quadratic ring $\mathcal{Q}_i$, and thus each multiplication is done as:

$$
\begin{aligned}
\hat{a_i} \cdot \hat{b_i} &= (a_{i,0} + a_{i,1}X) \cdot (b_{i,0} + b_{i,1}X) \\
&= a_{i,0}b_{i,0} + (a_{i,0}b_{i,1} + a_{i,1}b_{i,0})X + a_{i,1}b_{i,1}X^2 \\
&= (a_{i,0}b_{i,0} + a_{i,1}b_{i,1}\zeta^i) + (a_{i,0}b_{i,1} + a_{i,1}b_{i,0})X,
\end{aligned}
$$

Example: For clarity, consider the following (toy) example:

Let $n = 4$, $q = 13$, and thus n divides $q - 1 = 12 = 3 \cdot 4$, but not $2n = 8$. The primitive 4-th root of unity is $\zeta = 5$, since $\zeta^2 = 25 \equiv -1 \bmod 13$ and $\zeta^4 = 625 \equiv 1 \bmod 13$.

Therefore, the quadratic factors are

$$\mathcal{Q}_1 = \frac{\mathbb{Z}_{13}[X]}{(X^2 - \zeta)} = \frac{\mathbb{Z}_{13}[X]}{(X^2 - 5)}, \qquad \mathcal{Q}_2 = \frac{\mathbb{Z}_{13}[X]}{(X^2 - \zeta^3)} = \frac{\mathbb{Z}_{13}[X]}{(X^2 - 8)},$$

Let $a(X) = 1 + 2X + X^2$ and $b(X) = 3 + X + X^3$. Consider the multiplication of the two polynomials:

$$
\begin{aligned}
c(X) &= a(X) \cdot b(X) \\
&= (1 + 2X + X^2) \cdot (3 + X + X^3) \\
&= 3 + X + X^3 + 6X + 2X^2 + 2X^4 + 3X^2 + X^3 + X^5 \\
&= 3 + 7X + 5X^2 + 2X^3 + 2(X^4) + X(X^4) \\
&= 3 + 7X + 5X^2 + 2X^3 - 2 - X \qquad \text{(since } X^4 \equiv -1 \bmod X^4 + 1\text{)} \\
&= 1 + 6X + 5X^2 + 2X^3. \qquad (5.5)
\end{aligned}
$$

Now consider how the NTT multiplication would work. First, we compute the NTT transform of the two polynomials:

$$
\begin{aligned}
\mathsf{NTT}(a) &= ((1+2X+X^2) \bmod \mathcal{Q}_1, (1+2X+X^2) \bmod \mathcal{Q}_2) \\
&= ((1+2X+5), (1+2X+8)) \quad (\text{since } X^2 \equiv 5 \bmod \mathcal{Q}_1 \text{ and } X^2 \equiv 8 \bmod \mathcal{Q}_2) \\
&= (6+2X, 9+2X) \\
\mathsf{NTT}(b) &= ((3+X+X^3) \bmod \mathcal{Q}_1, (3+X+X^3) \bmod \mathcal{Q}_2) \\
&= ((3+X+5X), (3+X+8X)) \\
&\qquad\qquad (\text{since } X^3 \equiv 5X \bmod \mathcal{Q}_1 \text{ and } X^3 \equiv 8X \bmod \mathcal{Q}_2) \\
&= (3+6X, 3+9X).
\end{aligned}
$$

Therefore, multiplying the two transformed polynomials is done as follows, componentwise in each quadratic ring $\mathcal{Q}_i$:

$$
\begin{aligned}
\hat{c_1} &= \hat{a_1} \cdot \hat{b_1} = (6+2X) \cdot (3+6X) \\
&= 18 + 36X + 6X + 12X^2 \\
&= 18 + 42X + 12X^2 \equiv 5 + 3X + 12(5) \\
&= 5 + 3X + 8 \\
&\equiv 3X \\
\hat{c_2} &= \hat{a_2} \cdot \hat{b_2} = (9+2X) \cdot (3+9X) \\
&= 27 + 6X + 81X + 18X^2 \\
&= 27 + 87X + 18X^2 \equiv 1 + 9X + 18 * (8) \\
&= 2 + 9X
\end{aligned}
$$

Finally, we can compute the inverse NTT transform to get the resulting polynomial in the ring $\mathcal{R}_q$:

$$
\begin{aligned}
\mathsf{NTT}^{-1}(\hat{c}) &= \mathsf{NTT}^{-1}((3X, 2+9X)) \\
&= c_L(X) + c_R(X)X^2 \\
&= \frac{c_0 + c_1}{2} + \frac{c_0 - c_1}{2\zeta} X^2 \\
&= \frac{3X + (2+9X)}{2} + \frac{3X - (2+9X)}{2 \cdot 5} X^2 \\
&= 1 + 6X - (2X^2 + 6X^3) \cdot 4 \\
&= 1 + 6X - 8X^2 - 11X^3 \\
&= 1 + 6X + 5X^2 + 2X^3
\end{aligned}
$$

This matches the result from Eq. 5.5, showing how the NTT multiplication works in practice.

NTT Multiplication in `Simp-PKE`

Finally, we can now use the NTT multiplication in the `Simp-PKE` scheme to compute every product in the ring $\mathcal{R}_q$ efficiently. Also, since the matrix $\boldsymbol{A}$ is generated from a seed, we

can compute already sample it in the NTT domain (i.e. $\hat{\boldsymbol{A}} = \mathsf{NTT}(\boldsymbol{A})$), saving us the need to compute the NTT transform every time we want to multiply a vector with the matrix $\boldsymbol{A}$. The other vectors, however, are still sampled in the normal domain, since they are small in the $\mathcal{R}_q$ ring, and not necessarily in the NTT domain.

This results in the following algorithms, with the difference from the previously shown algorithms (Alg. 8 (`Simp-PKE.KeyGen()` (with small vectors $\boldsymbol{s}, \boldsymbol{e}$ generated by CBD)) and Alg. 9 (`Simp-PKE.Enc(pk, μ)` (with small vectors $\boldsymbol{s}, \boldsymbol{e}$ generated by CBD))) in orange:

Algorithm 10 `Simp-PKE.KeyGen()` (with NTT multiplication)

Input: Seed ρ ▷ Seed for the public key matrix $\boldsymbol{A}$
Output: Secret key $\mathsf{sk} = \hat{\boldsymbol{s}} \in \mathcal{R}_q^k$
Output: Public key $\mathsf{pk} = (\rho, \hat{\boldsymbol{b}})$

1: **for** $i = 0$ to $k - 1$ **do**
2: $\quad \boldsymbol{s}_i \leftarrow\$ \mathsf{CBD}_{\eta_1}$ ▷ Sample secret vector $\boldsymbol{s} \in \mathcal{R}_q^k$ from CBD_{η_1}

3: $\hat{\boldsymbol{A}} \leftarrow \mathsf{XOF}(\rho)$
▷ Generate uniformly random matrix $\hat{\boldsymbol{A}} \in \mathcal{R}_q^{k \times k}$ from the seed ρ using a XOF in the NTT domain
4: **for** $i = 0$ to $k - 1$ **do**
5: $\quad \boldsymbol{e}_i \leftarrow\$ \mathsf{CBD}_{\eta_1}$ ▷ Sample a small error vector $\boldsymbol{e} \in \mathcal{R}_q^k$ from CBD_{η_1}

6: $\hat{\boldsymbol{s}} \leftarrow \mathsf{NTT}(\boldsymbol{s})$
7: $\hat{\boldsymbol{e}} \leftarrow \mathsf{NTT}(\boldsymbol{e})$

8: $\hat{\boldsymbol{b}} \leftarrow \hat{\boldsymbol{A}} \cdot \hat{\boldsymbol{s}} + \hat{\boldsymbol{e}}$ ▷ Compute the public key $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \in \mathcal{R}_q^k$ in the NTT domain

9: $\mathsf{sk} := \hat{\boldsymbol{s}}$
10: $\mathsf{pk} := (\rho, \hat{\boldsymbol{b}})$
11: **return** $(\mathsf{sk}, \mathsf{pk})$

Algorithm 11 `Simp-PKE.Enc(pk, μ)` (with NTT multiplication)

Input: Public key $\mathsf{pk} = (\rho, \hat{\boldsymbol{b}})$
Input: Message $\mu \in \{0,1\}^n$
Output: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$

for $i = 0$ to $k-1$ **do**
  $\boldsymbol{r}_i \leftarrow\$ \mathsf{CBD}_{\eta_1}$ ▷ Sample a small random vector $\boldsymbol{r} \in \mathcal{R}_q^k$ from CBD_{η_1}
for $i = 0$ to $k-1$ **do**
  $\boldsymbol{e}_{1,i} \leftarrow\$ \mathsf{CBD}_{\eta_2}$ ▷ Sample a small error vector $\boldsymbol{e}_1 \in \mathcal{R}_q^k$ from CBD_{η_2}
$e_2 \leftarrow\$ \mathsf{CBD}_{\eta_2}$ ▷ Sample a small error $e_2 \in \mathcal{R}_q$ from CBD_{η_2}

$\hat{\boldsymbol{A}} \leftarrow \mathsf{XOF}(\rho)$ ▷ Expand matrix $\hat{\boldsymbol{A}} \in \mathcal{R}_q^{k \times k}$ from the seed ρ using a XOF in the NTT domain

$\hat{\boldsymbol{r}} \leftarrow \mathsf{NTT}(\boldsymbol{r})$

$\boldsymbol{u}^T \leftarrow \mathsf{NTT}^{-1}(\hat{\boldsymbol{r}}^T \cdot \hat{\boldsymbol{A}}) + \boldsymbol{e}_1^T$ ▷ Compute $\boldsymbol{u}^T = \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T \in \mathcal{R}_q^k$
$v \leftarrow \mathsf{NTT}^{-1}(\hat{\boldsymbol{r}}^T \cdot \hat{\boldsymbol{b}}) + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu$ ▷ Compute $v = \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu \in \mathcal{R}_q$

$\boldsymbol{c} := (\boldsymbol{u}^T, v)$
return $\boldsymbol{c}$

Algorithm 12 `Simp-PKE.Dec(sk, c)` (with NTT multiplication)

Input: Secret key $\mathsf{sk} = \hat{\boldsymbol{s}} \in \mathcal{R}_q^k$
Input: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$
Output: Message $\mu \in \{0,1\}^n$

$(\boldsymbol{u}^T, v) \leftarrow \boldsymbol{c}$
$\hat{\boldsymbol{u}}^T \leftarrow \mathsf{NTT}(\boldsymbol{u}^T)$
$w \leftarrow v - \mathsf{NTT}^{-1}(\hat{\boldsymbol{u}}^T \cdot \hat{\boldsymbol{s}})$
$\mu \leftarrow \mathrm{Round}(w)$
  ▷ Round each coefficient of w to either 0 or $\lfloor \frac{q}{2} \rceil$, recovering the message $\mu \in \{0,1\}^n$
return μ

5.4.4 Compression

Another important optimization in the ML-KEM scheme is the compression of the ciphertext, which, besides bringing the ciphertext size down, adds extra error to the scheme, which has to be accounted for in the security analysis and decryption failure probability. The compression and decompression algorithms simply rescales the interval into a smaller or larger range, respectively, considering the number of bits needed in each stage, which are used as the values d_u and d_v (for the ciphertext $\boldsymbol{u}$ and v, respectively) in the scheme.

$$\texttt{Compress}_d(x) = \left\lfloor \frac{2^d}{q} \cdot x \right\rceil \mod 2^d,$$
$$\texttt{Decompress}_d(x) = \left\lfloor \frac{q}{2^d} \cdot x \right\rceil \mod q,$$

That is, the compression algorithm rescales the value x from the range $[0, q)$ to the range $[0, 2^d)$, and then rounds to the nearest integer, while the decompression algorithm rescales the value x from the range $[0, 2^d)$ to the range $[0, q)$, and then rounds to the nearest integer.

This also allows us to use the same compression algorithm in the rounding of the message μ in the decryption algorithm (`Round` algorithm), which is done by simply compressing the value w to 1 bit in order to recover the message μ.

The compression and decompression algorithms introduce some error. Consider the following encryption and decryption steps:

$$\begin{aligned}
\text{Encryption: } \boldsymbol{c}_1 &= \texttt{Compress}_{d_u}(\boldsymbol{u}^T) \\
\boldsymbol{c}_2 &= \texttt{Compress}_{d_v}(v) \\
\text{Decryption: } \boldsymbol{u}' &= \texttt{Decompress}_{d_u}(\boldsymbol{c}_1) \\
v' &= \texttt{Decompress}_{d_v}(\boldsymbol{c}_2) \\
w &= v' - \boldsymbol{u}' \cdot \boldsymbol{s} \\
\mu &= \texttt{Compress}_1(w)
\end{aligned}$$

We can represent the error introduced by the compression and decompression algorithms as follows:

$$\begin{aligned}
\boldsymbol{u}' &= \texttt{Decompress}_{d_u}(\texttt{Compress}_{d_u}(\boldsymbol{u}^T)) = \boldsymbol{u}^T + \boldsymbol{e}' \\
v' &= \texttt{Decompress}_{d_v}(\texttt{Compress}_{d_v}(v)) = v + e'',
\end{aligned}$$

and thus, the decryption results in:

$$\begin{aligned}
w &= v' - \boldsymbol{u}' \cdot \boldsymbol{s} \\
&= (v + e'') - (\boldsymbol{u}^T + \boldsymbol{e}'^T) \cdot \boldsymbol{s} \\
&= v - \boldsymbol{u}^T \cdot \boldsymbol{s} + e'' - \boldsymbol{e}'^T \cdot \boldsymbol{s} \\
&= \underbrace{(\boldsymbol{r}^T \boldsymbol{e} + e_2 - \boldsymbol{e}_1^T \boldsymbol{s} + (e'' - \boldsymbol{e}'^T \cdot \boldsymbol{s}))}_{e} + \lfloor \frac{q}{2} \rceil \cdot \mu,
\end{aligned}$$

where now, the error term, e, is the one that needs to be less than $q/4$ in order to not have a decryption failure.

5.5 Full Scheme: K-PKE

Finally, this brings us to the final algorithms for the K-PKE scheme, which now include the compression and decompression algorithms in the encryption and decryption steps, respectively. Here we also add a hash function G to the key generation algorithm, which is used to generate the seed ρ for the public key matrix $\boldsymbol{A}$, and the noise key σ which is used in the PRFs. [1]

Algorithm 13 K-PKE.KeyGen()

Output: Secret key $\mathsf{sk} = \hat{\boldsymbol{s}} \in \mathcal{R}_q^k$

Output: Public key $\mathsf{pk} = (\rho, \hat{\boldsymbol{b}}) \in B^{32} \times \mathcal{R}_q^k$

$\zeta \leftarrow\$ B^{32}$ ▷ Master seed

$(\rho, \sigma) \leftarrow G(\zeta)$ ▷ Derive public-key seed ρ and noise key σ

for $i = 0$ to $k - 1$ **do**

  $\boldsymbol{s}_i \leftarrow \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, \text{“s”}, i))$

  ▷ Generate secret vector $\boldsymbol{s} \in \mathcal{R}_q^k$ from deterministic CBD_{η_1} via PRF

$\hat{\boldsymbol{A}} \leftarrow \mathsf{XOF}(\rho)$

  ▷ Generate uniformly random matrix $\hat{\boldsymbol{A}} \in \mathcal{R}_q^{k \times k}$ from the seed ρ using a XOF in the NTT domain

for $i = 0$ to $k - 1$ **do**

  $\boldsymbol{e}_i \leftarrow \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, \text{“e”}, i))$

  ▷ Sample a small error vector $\boldsymbol{e} \in \mathcal{R}_q^k$ from deterministic CBD_{η_1} via PRF

$\hat{\boldsymbol{s}} \leftarrow \mathsf{NTT}(\boldsymbol{s})$

$\hat{\boldsymbol{e}} \leftarrow \mathsf{NTT}(\boldsymbol{e})$

$\hat{\boldsymbol{b}} \leftarrow \hat{\boldsymbol{A}} \cdot \hat{\boldsymbol{s}} + \hat{\boldsymbol{e}}$ ▷ Compute the public key $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \in \mathcal{R}_q^k$ in the NTT domain

$\mathsf{sk} := \hat{\boldsymbol{s}}$

$\mathsf{pk} := (\rho, \hat{\boldsymbol{b}})$

return $(\mathsf{sk}, \mathsf{pk})$

[1] Note that there still are some differences from the ones in the original paper, which are mainly due to implementation details, such as encoding the values before sending them over the wire, and random coins used in the PRFs and XOFs, which are not shown here.

Algorithm 14 `K-PKE.Enc(pk, μ)`

Input: Public key $\mathsf{pk} = (\rho, \hat{\boldsymbol{b}})$
Input: Message $\mu \in \{0,1\}^n$
Output: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$

$\sigma \leftarrow\$ B^{32}$ ▷ PRF key for deterministic coins
for $i = 0$ to $k-1$ **do**
  $\boldsymbol{r}_i \leftarrow \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, \text{“r”}, i))$ ▷ Sample a small random vector $\boldsymbol{r} \in \mathcal{R}_q^k$ from deterministic CBD_{η_1} via PRF
  for $i = 0$ to $k-1$ **do**
    $\boldsymbol{e}_{1,i} \leftarrow \mathsf{CBD}_{\eta_2}(\mathsf{PRF}(\sigma, \text{“e1”}, i))$ ▷ Sample a small error vector $\boldsymbol{e}_1 \in \mathcal{R}_q^k$ from deterministic CBD_{η_2} via PRF
  $e_2 \leftarrow \mathsf{CBD}_{\eta_2}(\mathsf{PRF}(\sigma, \text{“e2”}, 0))$ ▷ Sample a small error $e_2 \in \mathcal{R}_q$ from deterministic CBD_{η_2} via PRF
  $\hat{\boldsymbol{A}} \leftarrow \mathsf{XOF}(\rho)$ ▷ Expand matrix $\hat{\boldsymbol{A}} \in \mathcal{R}_q^{k \times k}$ from the seed ρ using a XOF in the NTT domain
  $\hat{\boldsymbol{r}} \leftarrow \mathsf{NTT}(\boldsymbol{r})$
  $\boldsymbol{u}^T \leftarrow \mathsf{NTT}^{-1}(\hat{\boldsymbol{r}}^T \cdot \hat{\boldsymbol{A}}) + \boldsymbol{e}_1^T$ ▷ Compute $\boldsymbol{u}^T = \boldsymbol{r}^T \cdot \boldsymbol{A} + \boldsymbol{e}_1^T \in \mathcal{R}_q^k$
  $v \leftarrow \mathsf{NTT}^{-1}(\hat{\boldsymbol{r}}^T \cdot \hat{\boldsymbol{b}}) + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu$ ▷ Compute $v = \boldsymbol{r}^T \cdot \boldsymbol{b} + e_2 + \lfloor \frac{q}{2} \rceil \cdot \mu \in \mathcal{R}_q$
  $\boldsymbol{c}_1 \leftarrow \mathtt{Compress}_{d_u}(\boldsymbol{u})$ ▷ Compress the ciphertext $\boldsymbol{u}$ to d_u bits
  $\boldsymbol{c}_2 \leftarrow \mathtt{Compress}_{d_v}(v)$ ▷ Compress the ciphertext v to d_v bits
  $\boldsymbol{c} := (\boldsymbol{c}_1^T, \boldsymbol{c}_2)$
  return $\boldsymbol{c}$

Algorithm 15 `K-PKE.Dec(sk, c)`

Input: Secret key $\mathsf{sk} = \hat{\boldsymbol{s}} \in \mathcal{R}_q^k$
Input: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$
Output: Message $\mu \in \{0,1\}^n$

$(\boldsymbol{c}_1^T, \boldsymbol{c}_2) \leftarrow \boldsymbol{c}$
$\boldsymbol{u}' \leftarrow \mathtt{Decompress}_{d_u}(\boldsymbol{c}_1)$ ▷ Decompress the ciphertext $\boldsymbol{u}$ from d_u bits to $\mathcal{R}_q^k$
$v' \leftarrow \mathtt{Decompress}_{d_v}(\boldsymbol{c}_2)$ ▷ Decompress the ciphertext v from d_v bits to $\mathcal{R}_q$
$\hat{\boldsymbol{u}'}^T \leftarrow \mathsf{NTT}(\boldsymbol{u}'^T)$
$w \leftarrow v' - \mathsf{NTT}^{-1}(\hat{\boldsymbol{u}'}^T \cdot \hat{\boldsymbol{s}})$
$\mu \leftarrow \mathtt{Compress}_1(w)$
▷ Compress each coefficient of w to either 0 or 1, recovering the message $\mu \in \{0,1\}^n$
return μ

5.6 Security of K-PKE (Revisited)

After the introduction of the several optimizations, the security of the K-PKE scheme is still based on the hardness of the MLWE problem, however, it is now also based on the security of the symmetric primitives used in the scheme, such as the PRF. Consider the following Lemma, from ([SAB+20], Theorem 1).

Lemma 5.6.1 (K-PKE IND-CPA Security)**.** *Suppose* XOF *is a random oracle. For any adversary $\mathcal{A}$, there exists adversaries $\mathcal{B}$ and $\mathcal{C}$, with roughly the same running time of $\mathcal{A}$, such that if $\mathcal{A}$ can break the IND-CPA security of the K-PKE scheme with non-negligible probability, then*

$$\mathsf{Adv}^{\mathrm{cpa}}_{K\text{-}PKE}(\mathcal{A}) \leqslant 2\mathsf{Adv}^{\mathrm{mlwe}}_{n,k+1,k,q,\chi}(\mathcal{B}) + \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathcal{C})$$

Proof. The proof of this Lemma follows the same structure as the proof of Lemma 5.3.1, and the first part of the reduction, which is the reduction from the IND-CPA security of the K-PKE scheme to the hardness of the MLWE problem, will not be included here. However, additionally, we now have to consider the security of the symmetric primitives used in the scheme, such as the PRF, which is used to generate the random coins for the sampling of the values in the scheme, such as $\boldsymbol{s}, \boldsymbol{e}, \boldsymbol{r}, \boldsymbol{e}_1, e_2$.

Even though each sampling is represented directly as a call to their respective CBD_η function ($x \leftarrow\!\!\$\ \mathsf{CBD}_\eta$), in reality, this call is done to a deterministic function that includes a random coin and the PRF, which is then used to sample the values from the CBD distribution deterministically ($x \leftarrow \mathsf{CBD}_\eta(\mathsf{PRF}(\sigma))$), that is, the PRF is used as a keyed randomness for the CBD. This means that we can create a game hop before the ones in the proof of Lemma 5.3.1, where we replace the PRF with a random oracle, and thus, we can reduce the security of the K-PKE scheme to the security of the PRF. Suppose that this hop is done from *Game* 0 and *Game* 1, replacing the PRF with a random oracle, then we can define the advantage of an adversary $\mathcal{C}$ as follows:

In both K-PKE.KeyGen and K-PKE.Enc, all noise draws are implemented as

$$x \leftarrow \mathsf{CBD}_\eta(\mathsf{PRF}(\sigma, \mathsf{tag}))$$

with domain-separation tags (s, e, r, e1, e2) and a secret key σ. We introduce *Game* 1 by replacing $\mathsf{PRF}(\sigma, \mathsf{tag})$ with a uniform random function (URF) of the same interface (equivalently, replacing the coin bytes by uniform strings of the same lengths before applying the deterministic CBD_η map). Let p_0, p_1 be the adversary's win probabilities in *Game* 0 (PRF) and *Game* 1 (URF), then

$$\begin{aligned}\mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathcal{C}) &= \left|\Pr[b' = b \mid \text{Randomness is PRF}] - \Pr[b' = b \mid \text{Randomness is RO}]\right| \\ &= \left|p_0 - p_1\right|,\end{aligned}$$

where p_0 is the probability that the adversary $\mathcal{C}$ wins in the game *Game* 0, and p_1 is the probability that the adversary $\mathcal{C}$ wins in the game *Game* 1.

Next, we can use the steps from the proof of Lemma 5.3.1 to create the game hops for *Game* 2, where the public key is replaced with a random matrix, and *Game* 3, where the ciphertext is replaced with a random vector, and thus, we can define the advantage of the K-PKE scheme as follows:

$$\begin{aligned}
\mathsf{Adv}^{\text{ind-cpa}}_{\text{K-PKE}}(\mathcal{A}) &= |p_0 - \frac{1}{2}| \\
&\leqslant |p_0 - p_1| + |p_1 - p_2| + |p_2 - \frac{1}{2}| && \text{(Triangle Inequality)} \\
&\leqslant |p_0 - p_1| + |p_1 - p_2| + |p_2 - p_3| && (\textit{Game}3 \text{ is completely random}) \\
&\leqslant \mathsf{Adv}^{\text{prf}}_{\mathsf{PRF}}(\mathcal{C}) + \mathsf{Adv}^{\text{mlwe}}_{k,k,q,\chi}(\mathcal{B}) + \mathsf{Adv}^{\text{mlwe}}_{k+1,k,q,\chi}(\mathcal{B}) && \text{(from Lemma 5.3.1)} \\
&\leqslant 2\mathsf{Adv}^{\text{mlwe}}_{k+1,k,q,\chi}(\mathcal{B}) + \mathsf{Adv}^{\text{prf}}_{\mathsf{PRF}}(\mathcal{C}) && \text{(from Lemma 5.3.2)}
\end{aligned}$$

□

5.7 IND-CCA on K-PKE?

Over the last few sections, we have seen how the K-PKE scheme is constructed, and how it is secure under the IND-CPA security model, which is based on the hardness of the MLWE problem and the security of the PRF used in the scheme. However, we have not yet discussed whether the K-PKE scheme is IND-CCA secure (Def. 2.4.2), that is, whether it is secure against chosen ciphertext attacks. In fact, the K-PKE scheme is not IND-CCA secure, and an adversary can leverage the access to the decryption oracle to break the security of the scheme. The main reason for this is that the ciphertext $\boldsymbol{c}$ is malleable, that is, the adversary can modify the ciphertext $\boldsymbol{c}$ to create a new ciphertext $\boldsymbol{c}' \neq \boldsymbol{c}$ that decrypts to a valid message μ'. Since the decryption algorithm rounds the value w to recover the message, the adversary can modify the ciphertext $\boldsymbol{c}$ just enough so that the message μ' returned is equal to the original message μ, breaking the IND-CCA security of the scheme.

For example, consider the following attack, where the adversary has access to the decryption oracle K-PKE.Dec(sk, ·):

1. The adversary receives the public key pk and the ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v)$.
2. The adversary modifies the ciphertext to $\boldsymbol{c}' = (\boldsymbol{u}^T, v + e'')$, where e'' is a small value that does not change the value of w significantly.
3. The adversary queries the decryption oracle with K-PKE.Dec(sk, $\boldsymbol{c}'$).
4. The oracle computes

$$w' = v + e'' - \boldsymbol{u}^T \cdot \boldsymbol{s} = \lfloor \frac{q}{2} \rceil \cdot \mu + \underbrace{e + e''}_{small \ (\leqslant q/4)}$$

where e is the error term from the original ciphertext $\boldsymbol{c}$. Since e'' is small, the value w' will most likely still be close to $\lfloor \frac{q}{2} \rceil \cdot \mu$, and thus, the rounding operation will return the message $\mu' = \mu$.

5. The adversary receives the message μ' from the oracle, which is equal to the original message μ.

6. The adversary returns the bit from μ_b such that $\mu' = \mu_b$, thus breaking the IND-CCA security of the scheme.

Therefore, the `K-PKE` scheme is not IND-CCA secure and cannot be used directly in a key encapsulation mechanism (KEM) that requires IND-CCA security. However, we can use the `K-PKE` scheme as a building block for a KEM by applying the FO transform (Sec. 5.8), which will allow us to achieve the desired IND-CCA security.

5.8 FO Transform

The FO-transform (Fujisaki-Okamoto transform, [FO99]) is a well-known, generic method to transform an IND-CPA secure public key encryption scheme into an IND-CCA secure key encapsulation mechanism (KEM, Sec. 2.4.4, Def. 2.4.3). The way it achieves this is by making the decryption oracle only return the useful information (i.e the decryption message) if the ciphertext is valid, and the adversary already knows the message that generated it. That is, the FO-transform ensures that the decryption oracle provides no useful information to the adversary, preventing it from using it to break the security of the scheme.

Consider the following Simplified KEM (`Simp-KEM`) algorithms that use the `K-PKE` scheme as a building block through the FO-transform to create an IND-CCA secure KEM:

Algorithm 16 `Simp-KEM.KeyGen()`

Output: Secret key $\mathsf{sk} = \hat{\boldsymbol{s}} \in \mathcal{R}_q^k$

Output: Public key $\mathsf{pk} = (\rho, \hat{\boldsymbol{b}}) \in B^{32} \times \mathcal{R}_q^k$

1: $(\mathsf{pk}, \mathsf{sk}) \leftarrow$ `K-PKE.KeyGen()`
▷ Generate the public and secret keys using the `K-PKE.KeyGen()` algorithm

2: **return** $(\mathsf{sk}, \mathsf{pk})$

Algorithm 17 `Simp-KEM.Encaps(pk)`

Input: Public key $\mathsf{pk} = (\rho, \hat{\boldsymbol{b}})$
Output: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$
Output: Shared key $K \in B^{32}$

1: $\mu \leftarrow\$ \{0,1\}^n$ ▷ Sample a random message $\mu \in \{0,1\}^n$
2: $(K, \sigma) \leftarrow \mathcal{H}(\mu, \mathsf{pk})$ ▷ Generate a shared key K and a random coin σ for the generation of the secrets in the `K-PKE.Enc` using a hash function $\mathcal{H}$
3: $\boldsymbol{c} \leftarrow$ `K-PKE.Enc`$(\mathsf{pk}, \mu; \sigma)$ ▷ Encrypt the message μ with the public key pk using the `K-PKE.Enc` algorithm, with the random coin σ.[2]
4: **return** $(K, \boldsymbol{c})$

Algorithm 18 `Simp-KEM.Decaps(sk, c, pk)`

Input: Secret key $\mathsf{sk} = \hat{\boldsymbol{s}} \in \mathcal{R}_q^k$
Input: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$
Input: Public key $\mathsf{pk} = (\rho, \hat{\boldsymbol{b}})$
Output: Shared key $K \in B^{32}$

1: $\mu' \leftarrow$ `K-PKE.Dec`$(\mathsf{sk}, \boldsymbol{c})$ ▷ Decrypt the ciphertext $\boldsymbol{c}$ using the secret key sk
2: $(K', \sigma') \leftarrow \mathcal{H}(\mu', \mathsf{pk})$
▷ Generate a shared key K' and a random coin σ' from the decrypted message μ'
3: $\boldsymbol{c}' \leftarrow$ `K-PKE.Enc`$(\mathsf{pk}, \mu'; \sigma')$
▷ **Re-encrypt** the message μ' with the obtained random coin σ' to obtain a new ciphertext $\boldsymbol{c}'$
4: **if** $\boldsymbol{c}' \neq \boldsymbol{c}$ **then**
5: $K' \leftarrow \perp$ ▷ If the re-encrypted ciphertext $\boldsymbol{c}'$ is not equal to the original ciphertext $\boldsymbol{c}$, then the decryption failed, and we return an invalid key $K' = \perp$ [3]
6: **return** K'

Note that the FO-transform mainly consists of derandomizing the ciphertext generation, or making it "re-generable". This is introduced in the `Simp-KEM.Encaps` by generating the random coin σ from the message μ and the public key pk ($\mathcal{H}(\mu, \mathsf{pk})$, Line. 2), which allows us to "lock" the ciphertext to both message and public key. Also, since the random coin σ is generated deterministically from the message μ and the public key pk, the adversary cannot tweak the ciphertext (i.e. by adding some small error values to $\boldsymbol{u}$ or v), while still having it decrypt to the same message μ, as the random coin σ will not match the one used to generate the ciphertext $\boldsymbol{c}$.

[2]Note that the `K-PKE.Enc` algorithm is augmented with the random coin σ as an additional parameter, which is used to generate the random values in the scheme, as discussed in Sec. 5.6.

[3]This is the key point of the FO-transform, where we ensure that the decryption oracle only returns a valid key if the ciphertext is valid and the message is the same as the one used to generate it.

Then, in the Simp-KEM.Decaps, we re-encrypt the message μ' with the random coin σ' generated from the decrypted message μ' and the public key pk, and check if the re-encrypted ciphertext $\boldsymbol{c}'$ is equal to the original ciphertext $\boldsymbol{c}$. If they are not equal, then we return an invalid key $K' = \perp$, which ensures that the adversary will only be able to obtain a valid key from the decapsulation oracle if the ciphertext was properly generated to begin with, and thus, the adversary already knows the message that generated it (as well as the shared key $(K, \sigma) \leftarrow \mathcal{H}(\mu, \mathsf{pk})$). This makes it so that the Decapsulation oracle does not leak any useful information to the adversary, making it essentially useless for the adversary to query it.

5.8.1 Security of Simp-KEM

Reduction in the ROM The security of the Simp-KEM scheme follows from the security of the K-PKE scheme, which we give a proof sketch based on the ones shown in [HHK17]. The following Lemma is taken from ([SAB+20], Theorem 2), which uses both the Lemma of security of the K-PKE scheme (Lemma 5.6.1) and the (generic) security of the FO-transform shown in ([HHK17], Thm 3.6).

Lemma 5.8.1 (Simp-KEM IND-CCA Security)**.** *Suppose* $\mathsf{XOF}, \mathcal{H}$ *are random oracles. For any adversary* $\mathcal{A}$*, there exists adversaries* $\mathcal{B}$ *and* $\mathcal{C}$*, with roughly the same running time of* $\mathcal{A}$*, and that makes at most* q_{RO} *queries to the random oracles, such that*

$$\mathsf{Adv}^{\mathrm{cca}}_{\mathit{Simp\text{-}KEM}}(\mathcal{A}) \leqslant 2\mathsf{Adv}^{\mathrm{mlwe}}_{k+1,k,q,\chi}(\mathcal{B}) + \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathcal{C}) + 4\delta \cdot q_{RO},$$

where δ *is the decryption failure probability of the* K-PKE *scheme.*

Proof. Consider the following advantage of an adversary $\mathcal{A}$ against the IND-CCA security of the Simp-KEM scheme obtained by generic result of the FO-transform from ([HHK17], Thm 3.6).

$$\mathsf{Adv}^{\mathrm{cca}}_{\mathtt{Simp\text{-}KEM}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathrm{ind-cpa}}_{\mathtt{K\text{-}PKE}}(\mathcal{B}) + \frac{q_{RO}}{|\mathcal{M}|} + \delta(2q_{RO} + q_D),$$

where q_D is the number of decapsulation queries made by the adversary $\mathcal{A}$, and $|\mathcal{M}|$ is the size of the message space (i.e. the number of possible messages $\mu \in \{0,1\}^n$).

First we observe that the queries to the decapsulation oracle always call the random oracle $\mathcal{H}$ to generate the random coin σ', which gives us $q_D \leqslant q_{RO}$. Also, if we consider the size of the message space to be $|\mathcal{M}| = 2^n$, with $n = 256$, then the term $\frac{q_{RO}}{|\mathcal{M}|}$ is negligible, with $\frac{1}{|\mathcal{M}|} \leqslant \delta$, leaving us with the following inequality:

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{cca}}_{\mathtt{Simp\text{-}KEM}}(\mathcal{A}) &\leqslant \mathsf{Adv}^{\mathrm{ind-cpa}}_{\mathtt{K\text{-}PKE}}(\mathcal{B}) + \frac{q_{RO}}{|\mathcal{M}|} + \delta(2q_{RO} + q_D) \\
&\leqslant \mathsf{Adv}^{\mathrm{ind-cpa}}_{\mathtt{K\text{-}PKE}}(\mathcal{B}) + \frac{q_{RO}}{|\mathcal{M}|} + 3\delta \cdot q_{RO} \\
&\leqslant \mathsf{Adv}^{\mathrm{ind-cpa}}_{\mathtt{K\text{-}PKE}}(\mathcal{B}) + 4\delta \cdot q_{RO}
\end{aligned}$$

Next, we can use the result of Lemma 5.6.1 to bound the IND-CPA advantage of the K-PKE scheme as follows, inserting it into the previous inequality and giving us the desired result:

$$\mathsf{Adv}^{\mathrm{cca}}_{\texttt{Simp-KEM}}(\mathcal{A}) \leqslant 2\mathsf{Adv}^{\mathrm{mlwe}}_{k+1,k,q,\chi}(\mathcal{B}) + \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathcal{C}) + 4\delta \cdot q_{RO}$$

□

Reduction in the QROM Similarly, also using the results from ([HHK17], Thm 4.6), we can derive the following Lemma ([SAB+20], Theorem 3), which shows that the Simp-KEM scheme is IND-CCA secure in the QROM, that is, when the adversaries are also given *quantum* access to the random oracles.

Lemma 5.8.2 (Simp-KEM IND-CCA Security in the QROM)**.** *Suppose* $\mathsf{XOF}, \mathcal{H}$ *are quantum random oracles. For any adversary* $\mathcal{A}$*, there exists adversaries* $\mathcal{B}$ *and* $\mathcal{C}$*, with roughly the same running time of* $\mathcal{A}$*, and that makes at most* q_{QRO} *queries to the quantum random oracles, such that*

$$\mathsf{Adv}^{\mathrm{cca}}_{\textit{Simp-KEM}}(\mathcal{A}) \leqslant 4q_{RO} \cdot \sqrt{\mathsf{Adv}^{\mathrm{mlwe}}_{k+1,k,q,\chi}(\mathcal{B})} + \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathcal{C}) + 8\delta \cdot q^2_{RO},$$

where δ *is the decryption failure probability of the K-PKE scheme.*

Proof. The proof of this Lemma follows the same structure as the proof of Lemma 5.8.1, using the results from ([HHK17], Thm 4.6) along with the results from Lemma 5.6.1. Consider the following advantage of an adversary $\mathcal{A}$ against the IND-CCA security of the Simp-KEM scheme obtained by generic result of the FO-transform in the QROM from ([HHK17], Thm 4.6).

$$\mathsf{Adv}^{\mathrm{cca}}_{\texttt{Simp-KEM}}(\mathcal{A}) \leqslant (2q_{\mathcal{H}'} + q_{\mathcal{H}} + 2q_D) \cdot \sqrt{\mathsf{Adv}^{\mathrm{ind-cpa}}_{\texttt{K-PKE}}(\mathcal{B})} + \delta,$$

where $q_{\mathcal{H}'}, q_{\mathcal{H}}$ is the number of queries made to the quantum random oracles $|\mathcal{H}'\rangle$ and $|\mathcal{H}\rangle$, respectively, and q_D is the number of decapsulation queries made by the adversary $\mathcal{A}$.

Let us denote the number of queries to the quantum random oracles as $q_{QRO} = q_{\mathcal{H}'} + q_{\mathcal{H}}$, and note that the queries to the decapsulation oracle call at most one of the quantum random oracles, thus $q_D \leqslant q_{\mathcal{H}'} + q_{\mathcal{H}} = q_{RO}$. Also, in ([HHK17], Lemma 4.3), it is shown that the decryption failure probability δ can be bounded by $8\delta \cdot q^2_{RO}$, and thus, we can rewrite the advantage of the adversary $\mathcal{A}$ as follows:

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{cca}}_{\texttt{Simp-KEM}}(\mathcal{A}) &\leqslant (2q_{\mathcal{H}'} + q_{\mathcal{H}} + 2q_D) \cdot \sqrt{\mathsf{Adv}^{\mathrm{ind-cpa}}_{\texttt{K-PKE}}(\mathcal{B})} + \delta \\
&\leqslant (2q_{RO} + 2q_{RO}) \cdot \sqrt{\mathsf{Adv}^{\mathrm{ind-cpa}}_{\texttt{K-PKE}}(\mathcal{B})} + \delta \\
&\leqslant 4q_{RO} \cdot \sqrt{\mathsf{Adv}^{\mathrm{ind-cpa}}_{\texttt{K-PKE}}(\mathcal{B})} + 8\delta \cdot q^2_{RO} \qquad \text{(from [HHK17], Lemma 4.3)}
\end{aligned}$$

Finally, we use the result of Lemma 5.6.1 to bound the IND-CPA advantage of the K-PKE scheme, inserting it into the previous inequality and giving us the desired result.

□

5.9 ML-KEM

Finally, we can use the Simp-KEM (Sec. 5.8) scheme as a building block for the ML-KEM scheme, which is the final key encapsulation mechanism (KEM) used in the NIST PQC standard. It is based on the Simp-KEM scheme, but it adds some additional features to make it more efficient and secure, that is, it is the KEM generated from a slightly modified version of the FO-transform applied to the K-PKE scheme. The main differences from the Simp-KEM scheme are the following:

1. (Pre-)hashing the public key to before hashing it to generate the random coin σ in the Simp-KEM.Encaps(pk) algorithm;

$$\begin{aligned} & (K, \sigma) \leftarrow \mathcal{H}(\mu, \mathsf{pk}) && \text{(Alg. Simp-KEM.Encaps(pk), Line 2)} \\ \mapsto \quad & (K, \sigma) \leftarrow \mathcal{H}(\mu, \mathcal{H}_2(\mathsf{pk})) \end{aligned}$$

   - This allows the scheme to be more efficient to store the hashed pk ($\mathcal{H}_2(\mathsf{pk})$) in order to reuse it in the Encaps and Decaps algorithms, making the subsequent hashing operations faster.

2. Hiding valid or invalid decapsulation responses by returning pseudo-random values in case of decapsulation failure, instead of returning an invalid key $\perp$. This is done by sampling and storing a random value $z \leftarrow\!\!\$\ B^{32}$ along with the secret key sk in the key generation algorithm, and then using it to hide the decapsulation response in case of decapsulation failure in the decapsulation algorithm:

$$\begin{aligned} & \text{If } \boldsymbol{c}' \neq \boldsymbol{c}, \text{ then } \textbf{return } K = \perp && \text{(Alg. Simp-KEM.Decaps(sk, } \boldsymbol{c}\text{, pk), Line 4)} \\ \mapsto \quad & \text{If } \boldsymbol{c}' \neq \boldsymbol{c}, \text{ then } \textbf{return } K = \mathcal{H}_3(z, \boldsymbol{c}), \end{aligned}$$

   where $\mathcal{H}_3$ is another hash function used to hash the random value z and the ciphertext $\boldsymbol{c}$.

   - Note that the shared key K returned is now a hash of the random value z and the ciphertext $\boldsymbol{c}$, which makes it change every time the decapsulation oracle is queried for different ciphertexts, but staying the same for repeated queries with the same ciphertext.
   - This makes the decapsulation oracle behave the same as when the ciphertext is valid.

Now, we can define the ML-KEM scheme, following the presentation from [MLK24]. Note that the presentation here is slightly different from the one in the original specification, as it is simplified to focus on the main algorithms and their logic, while still being equivalent to the one presented in the specification. Mainly, the original specification introduces an internal algorithm for each of the steps, such as ML-KEM.KeyGen_internal, ML-KEM.Encaps_internal and ML-KEM.Decaps_internal, which have the main logic, and are called from the main algorithms, that are responsible for input checks and rejection, output formatting, and sampling of some random values. Let $\mathcal{H}, \mathcal{G}, \mathcal{J}$ be hash functions used in the scheme, we can define the ML-KEM algorithms as follows:

Algorithm 19 ML-KEM.KeyGen()

Output: Secret key $\mathsf{sk} = (\mathsf{sk}_{\mathtt{PKE}}, \mathsf{pk}, \mathcal{H}(\mathsf{pk}), z) \in \mathcal{R}_q^k \times B^{32} \times \mathcal{R}_q^k \times B^{32} \times B^{32}$
Output: Public key $\mathsf{pk} = (\rho, \hat{\boldsymbol{b}}) \in B^{32} \times \mathcal{R}_q^k$

1: $d \leftarrow\$ B^{32}$ ▷ Sample a random value $d \in B^{32}$
2: $z \leftarrow\$ B^{32}$ ▷ Sample a random value $z \in B^{32}$

3: $(\mathsf{pk}_{\mathtt{PKE}}, \mathsf{sk}_{\mathtt{PKE}}) \leftarrow$ K-PKE.KeyGen(d) ▷ Key Generation of the K-PKE scheme, with the randomness generated from the random value d

4: $\mathsf{pk} \leftarrow \mathsf{pk}_{\mathtt{PKE}}$ ▷ pk is the public key of the K-PKE scheme
5: $\mathsf{sk} \leftarrow (\mathsf{sk}_{\mathtt{PKE}}, \mathsf{pk}, \mathcal{H}(\mathsf{pk}), z)$ ▷ sk includes pk, hashed public key $\mathcal{H}(\mathsf{pk})$, and the random value z
6: **return** $(\mathsf{sk}, \mathsf{pk})$

Algorithm 20 ML-KEM.Encaps(pk)

Input: Public key $\mathsf{pk} = (\rho, \hat{\boldsymbol{b}})$
Output: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$
Output: Shared key $K \in B^{32}$

1: $\mu \leftarrow\$ \{0,1\}^n$ ▷ Sample a random message $\mu \in \{0,1\}^n$
2: $(K, \sigma) \leftarrow \mathcal{G}(\mu, \mathcal{H}(\mathsf{pk}))$ ▷ Derive a shared key K and a random coin σ for the generation of the secrets in the K-PKE.Enc using a hash function $\mathcal{G}$

3: $\boldsymbol{c} \leftarrow$ K-PKE.Enc(pk, μ, σ) ▷ Encrypt the message μ with the public key pk using the K-PKE.Enc algorithm, with the random coin σ.[4]
4: **return** $(K, \boldsymbol{c})$

[4]Note that the K-PKE.Enc algorithm is augmented with the random coin σ as an additional parameter, which is used to generate the random values in the scheme, as discussed in Sec. 5.6.

Algorithm 21 `ML-KEM.Decaps`$(\mathsf{sk}, \boldsymbol{c})$

Input: Secret key $\mathsf{sk} = (\mathsf{sk}_{\mathtt{PKE}}, \mathsf{pk}, \mathcal{H}(\mathsf{pk}), z)$
Input: Ciphertext $\boldsymbol{c} = (\boldsymbol{u}^T, v) \in \mathcal{R}_q^k \times \mathcal{R}_q$
Output: Shared key $K \in B^{32}$

1: $(\mathsf{sk}_{\mathtt{PKE}}, \mathsf{pk}, \mathcal{H}(\mathsf{pk}), z) \leftarrow \mathsf{sk}$ ▷ Extract the secret key components from sk

2: $\mu' \leftarrow \mathtt{K\text{-}PKE.Dec}(\mathsf{sk}_{\mathtt{PKE}}, \boldsymbol{c})$ ▷ Decrypt the ciphertext $\boldsymbol{c}$ using the `K-PKE` secret key $\mathsf{sk}_{\mathtt{PKE}}$

3: $(K', \sigma') \leftarrow \mathcal{H}(\mu', \mathsf{pk})$
▷ Generate a shared key K' and a random coin σ' from the decrypted message μ'

4: $\boldsymbol{c}' \leftarrow \mathtt{K\text{-}PKE.Enc}(\mathsf{pk}, \mu'; \sigma')$
▷ **Re-encrypt** the message μ' with the obtained random coin σ' to obtain a new ciphertext $\boldsymbol{c}'$

5: **if** $\boldsymbol{c}' \neq \boldsymbol{c}$ **then**

6: $K' \leftarrow \mathcal{J}(z, \boldsymbol{c})$ ▷ If the re-encrypted ciphertext $\boldsymbol{c}'$ is not equal to the original ciphertext $\boldsymbol{c}$, then the decryption failed and we return a (pseudo-)random key (implicit rejection).

7: **return** K'

5.9.1 Security of `ML-KEM`

The security of the `ML-KEM` scheme follows from the security of the `Simp-KEM` scheme, presented in Sec. 5.8, Lemma 5.8.1 and Lemma 5.8.2, which in turn follows from the security of the `K-PKE` scheme, presented in Sec. 5.6, Lemma 5.6.1. The optimizations and tweaks made to the `ML-KEM` scheme do not affect the security of the scheme, as they are only related to the efficiency and usability of the scheme, and do not change the underlying security assumptions. The resulting theorems for the security of the `ML-KEM` scheme are as follows:

Theorem 5.9.1 (`ML-KEM` IND-CCA Security)**.** *Suppose* $\mathsf{XOF}, \mathcal{G}, \mathcal{H}$ *are random oracles. For any adversary* $\mathcal{A}$*, there exists adversaries* $\mathcal{B}$ *and* $\mathcal{C}$*, with roughly the same running time of* $\mathcal{A}$*, and that makes at most* q_{RO} *queries to the random oracles, such that*

$$\mathsf{Adv}^{\mathrm{cca}}_{\mathit{Simp\text{-}KEM}}(\mathcal{A}) \leqslant 2\mathsf{Adv}^{\mathrm{mlwe}}_{n,k+1,k,q,\chi}(\mathcal{B}) + \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathcal{C}) + 4\delta \cdot q_{RO},$$

where δ *is the decryption failure probability of the* `K-PKE` *scheme.*

Proof. The proof follows directly from Lemma 5.8.1, since the `ML-KEM` scheme is based on the `Simp-KEM` scheme, and the optimizations made to the `ML-KEM` scheme do not affect the security of the scheme. □

Theorem 5.9.2 (`ML-KEM` IND-CCA Security in the QROM)**.** *Suppose* $\mathsf{XOF}, \mathcal{G}, \mathcal{H}$ *are quantum random oracles. For any adversary* $\mathcal{A}$*, there exists adversaries* $\mathcal{B}$ *and* $\mathcal{C}$*, with roughly*

the same running time of $\mathcal{A}$, and that makes at most q_{QRO} queries to the quantum random oracles, such that

$$\mathsf{Adv}^{\mathrm{cca}}_{\textit{Simp-KEM}}(\mathcal{A}) \leqslant 4q_{RO} \cdot \sqrt{\mathsf{Adv}^{\mathrm{mlwe}}_{n,k+1,k,q,\chi}(\mathcal{B})} + \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}}(\mathcal{C}) + 8\delta \cdot q^2_{RO},$$

where δ is the decryption failure probability of the K-PKE scheme.

Proof. The proof follows directly from Lemma 5.8.2, since the ML-KEM scheme is based on the Simp-KEM scheme, and the optimizations made to the ML-KEM scheme do not affect the security of the scheme. □

5.10 Parameter Sets

Following the presentation from the NIST PQC standard, for clarity and completeness, we describe here the parameter sets for the ML-KEM scheme, which have been chosen to provide a good balance between security and performance, while providing the required security levels based on the categories defined in the NIST PQC standard. The parameter sets assigns the values for the parameters $k, \eta_1, \eta_2, d_u,$ and $, d_v$, which depend on the security level, as well as the two constants $n = 256$ and $q = 3329$, fixing the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^{256}+1)$ for all the parameter sets. The parameters are k for the rank of the module, η_1 and η_2 for the parameter used in the sampling of the small vectors (i.e. CBD_{η_1}), d_u and d_v for the number of bits for the compression of the vectors $\boldsymbol{u}$ and v, respectively, and δ for the decryption failure probability of the K-PKE scheme. The parameters are summarized in Table 5.1.

Level	**Algorithm**	n	q	k	η_1	η_2	d_u	d_v	δ	Security Strength (bits)
1	ML-KEM-512	256	3329	2	3	2	10	4	2^{-139}	128
3	ML-KEM-768	256	3329	3	2	2	10	4	2^{-164}	192
5	ML-KEM-1024	256	3329	4	2	2	11	5	2^{-174}	256

Table 5.1: Parameter sets for the ML-KEM scheme, based on the NIST PQC standard [MLK24].

5.11 Conclusion

In this chapter, we have revisited the security of the K-PKE scheme, which is the public key encryption scheme used in the ML-KEM key encapsulation mechanism (KEM) from the NIST PQC standard, as well as shown the necessary steps, design decisions and optimizations to transform it into an IND-CCA secure KEM using the (slightly modified) FO-transform. We have also presented the ML-KEM scheme, concluded with the security proofs for the scheme, as well as game reduction arguments, along with proofs for the security of the K-PKE scheme, which ultimately shows how the security is based on the hardness of the MLWE problem.

Chapter 6

SVP solvers

6.1 Kannan's Enumeration SVP algorithm

This is a review of the paper on Kannan's SVP algorithm [Kan83], which presents a deterministic procedure (shown in the procedure ENUMERATE) for exact SVP solvers that enumerates all lattice points within a smaller region. This procedure is discussed in Sec. 6.1.2. The main result is an algorithm that has time complexity of $2^{\mathcal{O}(n \log n)}$, or $n^{\mathcal{O}(n)}$ as it is shown in Theorem 6.1.3.

6.1.1 SVP Algorithm

The complete algorithm computes a HKZ-reduced[1] basis and then uses the ENUMERATE subroutine, with its time depending on the quality of the basis. The idea of the algorithm is to pre-compute a basis with better quality (LLL then HKZ) before calling the ENUMERATE subroutine. The algorithm is shown in Algorithm SHORTEST.

In several instances throughout the SHORTEST algorithm and its subroutines, there is a step that lifts vectors from a lower-dimensional lattice, orthogonal to the first vector, to the original lattice. This step is explained in the Proposition 6.1.1.

Proposition 6.1.1. *Let $\mathcal{L}_1$ be a lattice formed by the basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, $\boldsymbol{b}'_i$ be the projection of $\boldsymbol{b}_i$ orthogonal to $\boldsymbol{b}_1$. Let $\mathcal{L}_2$ be the lattice with basis $\boldsymbol{B}_2 = (\boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_n)$, orthogonal to $\boldsymbol{b}_1$, and $\boldsymbol{B}_3 = (\bar{\boldsymbol{b}_2}, \ldots, \bar{\boldsymbol{b}_n})$ be the basis returned by running SHORTEST$(\mathcal{L}_2)$. Finally, let $\boldsymbol{a}_i$ be the vector in $\mathcal{L}_1$ such that $\boldsymbol{a}_i$'s projection perpendicular to $\boldsymbol{b}_1$ is $\bar{\boldsymbol{b}_i}$ and $\langle \boldsymbol{a}_i \, , \boldsymbol{b}_1 \rangle \in \left(-\frac{\langle \boldsymbol{b}_1 \, , \boldsymbol{b}_1 \rangle}{2}, \frac{\langle \boldsymbol{b}_1 \, , \boldsymbol{b}_1 \rangle}{2} \right]$. The vector $\boldsymbol{a}_i \in \mathcal{L}_1$ is unique and can be found by adding rational multiples of $\boldsymbol{b}_1$ to $\bar{\boldsymbol{b}_1}$.*

Proof. First, since $\boldsymbol{b}'_i$ is the projection of $\boldsymbol{b}_i$ orthogonal to $\boldsymbol{b}_1$, we can write $\boldsymbol{b}_i = \boldsymbol{b}'_i + \frac{\langle \boldsymbol{b}_i \, , \boldsymbol{b}_1 \rangle}{\langle \boldsymbol{b}_1 \, , \boldsymbol{b}_1 \rangle} \boldsymbol{b}_1$.

Let $\boldsymbol{v}$ be an element in $\mathcal{L}_2$ returned after the call to SHORTEST$(\mathcal{L}_2)$. Thus, we can write $\boldsymbol{v}$ in terms of $\boldsymbol{B}_3$: $\boldsymbol{v} = \sum_{i=2}^{n} \alpha_i \bar{\boldsymbol{b}_i}$. However, since $\boldsymbol{B}_2$ and $\boldsymbol{B}_3$ form the same lattice, we can also write it in terms of $\boldsymbol{B}_2$: $\boldsymbol{v} = \sum_{i=2}^{n} \beta_i \boldsymbol{b}'_i$. In order to lift $\boldsymbol{v}$ to $\mathcal{L}_1$:

[1] See Sec. 2.3.6

$$\boldsymbol{v} = \sum_{i=2}^{n} \beta_i \boldsymbol{b}_i' = \sum_{i=2}^{n} \beta_i \left(\boldsymbol{b}_i - \frac{\langle \boldsymbol{b}_i \, , \, \boldsymbol{b}_1 \rangle}{\langle \boldsymbol{b}_1 \, , \, \boldsymbol{b}_1 \rangle} \boldsymbol{b}_1 \right)$$
$$= \sum_{i=2}^{n} \beta_i \boldsymbol{b}_i - \frac{\langle \beta_i \boldsymbol{b}_i \, , \, \boldsymbol{b}_1 \rangle}{\langle \boldsymbol{b}_1 \, , \, \boldsymbol{b}_1 \rangle} \boldsymbol{b}_1$$
$$\Rightarrow \boldsymbol{a}_i := \boldsymbol{v} + \sum_{i=2}^{n} \frac{\langle \beta_i \boldsymbol{b}_i \, , \, \boldsymbol{b}_1 \rangle}{\langle \boldsymbol{b}_1 \, , \, \boldsymbol{b}_1 \rangle} \boldsymbol{b}_1 = \sum_{i=2}^{n} \beta_i \boldsymbol{b}_i \in \mathcal{L}_1.$$

Thus, to find the unique element $\boldsymbol{a}_i$ in $\mathcal{L}_1$, we need to add to $\bar{\boldsymbol{b}_i}$ a rational multiple of $\boldsymbol{b}_1$ that can be obtained from the components of the projections of $\boldsymbol{b}_i'$ into $\boldsymbol{b}_1$. □

6.1.2 Enumeration Procedure

The algorithm uses a procedure to enumerate short lattice points that finds all lattice points in a ball of some radius R, centered at the origin. In the SVP algorithm, it is used to find all lattice points which have euclidean norm at most $\|\boldsymbol{b}_1\|$.

Proposition 6.1.2. *Consider a Lattice $\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ such that its basis satisfies:*

1. *$\boldsymbol{B}_{[i,m]}$ is SVP-reduced for $i = 2, \ldots, m$;*

and let $\boldsymbol{y} = \sum_{i=1}^{m} \alpha_i \boldsymbol{b}_i$ be the shortest vector of $\mathcal{L}$. Suppose $\beta_{i+1}, \ldots, \beta_m$ are fixed integers. Then there is an easily computed integer β_i^0 such that for all integers $\alpha_1, \ldots, \alpha_{i-1}$ and β_i satisfying:

$$\sum_{j=1}^{i-1} \alpha_j \boldsymbol{b}_j + \beta_i \boldsymbol{b}_i + \sum_{j=i+1}^{m} \beta_j \boldsymbol{b}_j \leqslant \|\boldsymbol{b}_1\|$$

then we must have

$$\beta_i \in \left[\beta_i^0, \beta_i^0 + 2 \frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|} \right]$$

Proof. For any vector $\boldsymbol{v}$, let $\hat{\boldsymbol{v}}$ be the component of $\boldsymbol{v}$ along $\boldsymbol{b}_i(i)$ (i.e. $\hat{\boldsymbol{v}} = \frac{\langle \boldsymbol{v} \, , \, \boldsymbol{b}_i(i) \rangle}{\langle \boldsymbol{b}_i(i) \, , \, \boldsymbol{b}_i(i) \rangle} \boldsymbol{b}_i(i)$). Let $\boldsymbol{w} = \sum_{j=1}^{i-1} \alpha_j \boldsymbol{b}_j + \beta_i \boldsymbol{b}_i + \sum_{j=i+1}^{m} \beta_j \boldsymbol{b}_j$. Therefore, we have that $\hat{\boldsymbol{w}} = \beta_i \boldsymbol{b}_i(i) + \sum_{j=i+1}^{m} \beta_j \hat{\boldsymbol{b}}_j$, with $\beta_j, j = i+1, \ldots, m$ fixed. Since the last sum is fixed, we can write $\hat{\boldsymbol{w}} = \beta_i \boldsymbol{b}_i(i) + t\boldsymbol{b}_i(i)$, where t is a scalar. Thus, $\hat{\boldsymbol{w}} = (\beta_i + t)\boldsymbol{b}_i(i)$, and since we want to find $\|\boldsymbol{w}\| \leqslant \|\boldsymbol{b}_1\|$, then $\|\hat{\boldsymbol{w}}\| \leqslant \|\boldsymbol{b}_1\|$. $\{ (\beta_i + t)\boldsymbol{b}_i(i) \mid \beta_i \in \mathbb{Z} \}$ are colinear points with distance $\|\boldsymbol{b}_i(i)\|$ between adjacent points. Therefore:

$$|\beta_i + t| \leqslant \frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|}$$
$$\Rightarrow -\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|} - t \leqslant \beta_i \leqslant \frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|} - t$$

Algorithm 22 SHORTEST

Input: An integer lattice basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$

Output: An HKZ-reduced basis of the same lattice.

▷ Size-Reduced ($|\mu_{j,i}| \leqslant 1/2$ for $1 \leqslant i < j \leqslant n$.) and $\boldsymbol{B}_{[i,n]}$ is *SVP-reduced* for $i = 1, \ldots, n$.

1: **if** $n = 1$ **then**, return $(\boldsymbol{b}_1)$.

2: LLL-reduce the basis $\boldsymbol{B}$. ▷ Size-reduced and $\frac{3}{4}\|b_i^*\|^2 \leqslant \|\mu_{i+1,i}b_i^* + b_{i+1}^*\|^2$ for $1 \leqslant i < n$

3: Compute the projections $(\boldsymbol{b}'_i)_{i \geqslant 2}$ of the $\boldsymbol{b}_i$'s orthogonal to $\boldsymbol{b}_1$.

4: $(\bar{\boldsymbol{b}_2}, \ldots, \bar{\boldsymbol{b}_n}) \leftarrow$ SHORTEST$(\boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_n)$

▷ HKZ-reduce the $(n-1)$-dimensional lattice generated by $(\boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_n)$ (Recursively call the same procedure on lower dimensional lattice).

5: **for** $i = 2$ to n **do**

6: Find the unique element $\boldsymbol{a_i}$ in $\mathcal{L}$ such that $\boldsymbol{a_i}$'s projection perpendicular to $\boldsymbol{b}_1$ is $\bar{\boldsymbol{b_i}}$ and $\langle \boldsymbol{a}_i \, , \boldsymbol{b}_1 \rangle \in \left(-\frac{\langle \boldsymbol{b}_1 \, , \boldsymbol{b}_1 \rangle}{2}, \frac{\langle \boldsymbol{b}_1 \, , \boldsymbol{b}_1 \rangle}{2} \right]$.

7: $\boldsymbol{b_i} \leftarrow \boldsymbol{a_i}$.

▷ Extend the obtained $(\bar{\boldsymbol{b_i}})_{i \geqslant 2}$ into vectors of $\mathcal{L}$ by adding to them rational multiples of $\boldsymbol{b}_1$, in such a way that we have $|\mu_{i,1}| \leqslant 1/2$ for any $i \geqslant 2$. Refer to Proposition 6.1.1

8: **if** $\|\boldsymbol{b}_2\| < \frac{\sqrt{3}}{2}\|\boldsymbol{b}_1\|$ **then**, swap $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ and go to step 3.

▷ We want $\boldsymbol{b}_1$ to be shorter or not much longer than $\boldsymbol{b}_2$

9: **if** $\|\boldsymbol{b}_j(j)\| \geqslant \|\boldsymbol{b}_1\|$ for some j **then**
$j_0 \leftarrow$ minimum such j

10: **else**
$j_0 \leftarrow n + 1$.

11: Call ENUMERATE$(\boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_{j_0-1}})$ to obtain $\boldsymbol{v_1}$ = a shortest nonzero vector in $\mathcal{L}(\boldsymbol{b_1}, \ldots, \boldsymbol{b_{j_0-1}})$.

▷ Call the Enumeration Subroutine to find all lattice vectors of length at most $\|\boldsymbol{b}_1\|$. This is done on a sub-basis to make enumeration faster

12: $\{\boldsymbol{b_1}, \ldots, \boldsymbol{b_n}\} \leftarrow$ SELECT_BASIS$(n, \boldsymbol{v_1}, \boldsymbol{b_1}, \ldots, \boldsymbol{b_n})$.

13: Compute the projections $(\boldsymbol{b}'_i)_{i \geqslant 2}$ of the b_i's orthogonally to $\boldsymbol{b}_1$.

14: $(\bar{\boldsymbol{b}_2}, \ldots, \bar{\boldsymbol{b}_n}) \leftarrow$ SHORTEST$(\boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_n)$

▷ HKZ-reduce the $(n-1)$-dimensional lattice generated by $(\boldsymbol{b}'_2, \ldots, \boldsymbol{b}'_n)$ (Recursively call the same procedure on lower dimensional lattice).

15: **for** $i = 2$ to n **do**
Find the unique element $\boldsymbol{a_i}$ in $\mathcal{L}$ such that $\boldsymbol{a_i}$'s projection perpendicular to $\boldsymbol{b_1}$ is $\bar{\boldsymbol{b_i}}$ and $\langle \boldsymbol{a}_i \, , \boldsymbol{b}_1 \rangle \in \left(-\frac{\langle \boldsymbol{b}_1 \, , \boldsymbol{b}_1 \rangle}{2}, \frac{\langle \boldsymbol{b}_1 \, , \boldsymbol{b}_1 \rangle}{2} \right]$.

▷ Extend the obtained $(\bar{\boldsymbol{b_i}})_{i \geqslant 2}$ into vectors of $\mathcal{L}$ by adding to them rational multiples of $\boldsymbol{b}_1$, in such a way that we have $|\mu_{i,1}| \leqslant 1/2$ for any $i \geqslant 2$. Refer to Proposition 6.1.1

16: Return $(\boldsymbol{b_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n})$.

Let $\beta_i^0 = -\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|} - t$, then:

$$\beta_i^0 \leqslant \beta_i \leqslant \beta_i^0 + 2\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|}$$

Thus, the proposition holds. □

In words, if we fix the values of the last components of the vector, we can find the range of values for the i-th component that will make the vector have squared norm less than $\|\boldsymbol{b}_1\|$. In the enumeration procedure, this is used recursively by starting from the last component, and finding all values for the next to last component that satisfy the proposition, and so on. It is a depth-first search of all possible coefficients of the vector.

For example, the first call to the procedure will be by finding α_m in the range

$$\left[-\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_m(m)\|}, \frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_m(m)\|}\right].$$

Then, for each value of α_m, we will find all values of α_{m-1} that satisfy the proposition, and so on.

The procedure is shown in Algorithms 23 and 24.

Algorithm 23 `ENUMERATE`

Input: An integer basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$. The basis satisfies:

1. $\boldsymbol{B}_{[i,m]}$ is *SVP-reduced* for $i = 2, \ldots, m$;
2. $\|\boldsymbol{b}_1\| \leqslant 2/\sqrt{3}\|\boldsymbol{b}_2\|$;
3. $\|\boldsymbol{b}_2 - \boldsymbol{b}_2(2)\| \leqslant \|\boldsymbol{b}_1/2\|$. ($\boldsymbol{b}_1$ is not much longer than $\boldsymbol{b}_2$ and the projection of $\boldsymbol{b}_2$ into $\boldsymbol{b}_1$ is short ($\mu_{2,1} \leqslant 1/2$))

Output: The shortest vector of $\mathcal{L}(B)$.

1: **if** $m = 1$ **then return** $\boldsymbol{b}_1$
2: **for** each $\alpha_m \in \left[-\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_m(m)\|}, \frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_m(m)\|}\right]$ **do**
$T \leftarrow T \cup$ `LIST`(m-1)
▷ Call `LIST`(m-1): `LIST`(m-1) returns a list of candidates $(\alpha_1, \ldots, \alpha_m)$ as per `PROPOSITION` 6.1.2
3: **return** Shortest nonzero vector in the set $\{\sum_{j=1}^m \alpha_j \boldsymbol{b}_j : (\alpha_1, \ldots, \alpha_n) \in T\}$

Corollary 6.1.1. *The list returned by the `LIST` subroutine contains the shortest vector of the lattice.*

Proof. The `LIST` subroutine returns all possible integer combinations of the basis vectors that satisfy the proposition 6.1.2. Since the shortest vector of the lattice is the one that satisfies the proposition, it must be in the list. □

Algorithm 24 `LIST`

Input: An integer basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$. ▷ When this procedure is called, the integers $\alpha_{k+1}, \alpha_{k+2}, \ldots, \alpha_m$ are fixed

Output: A list of candidates $(\alpha_1, \ldots, \alpha_m)$

1: **if** $k = 0$ **then** $T \leftarrow T \cup \{ (\alpha_1, \ldots, \alpha_m) \}$

2: Compute β_k^0 as per `PROPOSITION 6.1.2`

3: **for** each integer $\alpha_k \in \left[\beta_k^0, \beta_k^0 + 2\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_k(k)\|}\right]$ **do**
$T \leftarrow T \cup$ `LIST`(k-1) ▷ Call `LIST`(k-1)

4: **return** T

Proposition 6.1.3. *The `ENUMERATE` procedure returns a shortest nonzero vector in the lattice $\mathcal{L}(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$.*

Proof. By Corollary 6.1.1, we have that the list of candidates the `LIST` subroutine returns contains the shortest vector of the lattice, thus, the shortest vector in this list returned by `ENUMERATE` is the shortest vector in the lattice. □

Lemma 6.1.1. *Given an integer basis $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ of a lattice that satisfies:*

1. *$\boldsymbol{B}_{[i,m]}$ is SVP-reduced for $i = 2, \ldots, m$;*
2. *$\|\boldsymbol{b}_1\| \leqslant 2/\sqrt{3}\|\boldsymbol{b}_2\|$;*
3. *$\|\boldsymbol{b}_2 - \boldsymbol{b}_2(2)\| \leqslant \|\boldsymbol{b}_1/2\|$.*

The `ENUMERATE` procedure tries out at most $2^m \cdot \prod_{i=2}^m \left(\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|}\right)$ sets of values, which in turn is at most $(2(m-1))^{\frac{m-1}{2}}$.

Proof. From Proposition 6.1.2, for each $\alpha_i, i = 1, \ldots, m$, there are $2\left(\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|}\right)$ candidates, and therefore, in total, there are at most $2^m \cdot \prod_{i=2}^m \left(\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|}\right)$ candidates. We also know that the determinant of the lattice $\mathcal{L}'(\boldsymbol{b}_2(2), \boldsymbol{b}_3(2), \ldots, \boldsymbol{b}_m(2))$ is $|\det(\mathcal{L}')| = \prod_{i=2}^m \|\boldsymbol{b}_i(i)\|$. Therefore, by Minkowski's theorem:

$$
\begin{aligned}
\|\boldsymbol{b}_2(2)\| &\leqslant \sqrt{m-1} \cdot |\det(\mathcal{L}')|^{1/m-1} && \text{(Minkowski's Theorem)} \\
\Rightarrow \frac{\|\boldsymbol{b}_2(2)\|^{m-1}}{|\det(\mathcal{L}')|} &\leqslant (\sqrt{m-1})^{m-1} \\
\Rightarrow \frac{\|\boldsymbol{b}_2(2)\|^{m-1}}{\prod_{i=2}^m \|\boldsymbol{b}_i(i)\|} &\leqslant (\sqrt{m-1})^{m-1} \\
\Rightarrow \prod_{i=2}^m \left(\frac{\|\boldsymbol{b}_2(2)\|}{\|\boldsymbol{b}_i(i)\|}\right) &\leqslant (\sqrt{m-1})^{m-1}
\end{aligned}
$$

Also, from conditions 2 and 3, we have $\|\boldsymbol{b}_2\| \geqslant \frac{\sqrt{3}}{2}\|\boldsymbol{b}_1\|$ and $\|\boldsymbol{b}_2 - \boldsymbol{b}_2(2)\| \leqslant \frac{\|\boldsymbol{b}_1\|}{2}$, we also have that:

$$\begin{aligned}
&\boldsymbol{b}_2 = \boldsymbol{b}_2(2) + \alpha\boldsymbol{b}_1 \Rightarrow \boldsymbol{b}_2 - \boldsymbol{b}_2(2) = \alpha\boldsymbol{b}_1 \\
&\quad \Rightarrow \|\boldsymbol{b}_2 - \boldsymbol{b}_2(2)\| = |\alpha|\|\boldsymbol{b}_1\| \\
&\quad \Rightarrow |\alpha| \leqslant \frac{1}{2} \qquad \text{(Condition 3: } \|\boldsymbol{b}_2 - \boldsymbol{b}_2(2)\| \leqslant \tfrac{\|\boldsymbol{b}_1\|}{2}\text{)}
\end{aligned}$$

Thus,

$$\begin{aligned}
\|\boldsymbol{b}_2\|^2 &= \|\boldsymbol{b}_2(2)\|^2 + \alpha^2\|\boldsymbol{b}_1\|^2 \\
&\leqslant \|\boldsymbol{b}_2(2)\|^2 + \frac{1}{4}\|\boldsymbol{b}_1\|^2 \\
\Rightarrow \|\boldsymbol{b}_2(2)\|^2 &\geqslant \|\boldsymbol{b}_2\|^2 - \frac{1}{4}\|\boldsymbol{b}_1\|^2 \\
&\geqslant \frac{3}{4}\|\boldsymbol{b}_1\|^2 - \frac{1}{4}\|\boldsymbol{b}_1\|^2 \qquad \text{(Condition 2: } \|\boldsymbol{b}_2\| \geqslant \tfrac{\sqrt{3}}{2}\|\boldsymbol{b}_1\|\text{)} \\
&\geqslant \frac{1}{2}\|\boldsymbol{b}_1\|^2 \\
\Rightarrow \|\boldsymbol{b}_2(2)\| &\geqslant \frac{1}{\sqrt{2}}\|\boldsymbol{b}_1\| \\
\Rightarrow \sqrt{2}\|\boldsymbol{b}_2(2)\| &\geqslant \|\boldsymbol{b}_1\|
\end{aligned}$$

Therefore:

$$\begin{aligned}
(\sqrt{2})^{m-1}\prod_{i=2}^{m}\left(\frac{\|\boldsymbol{b}_2(2)\|}{\|\boldsymbol{b}_i(i)\|}\right) &\leqslant (\sqrt{2})^{m-1}\cdot(\sqrt{m-1})^{m-1} \\
\Rightarrow \prod_{i=2}^{m}\left(\frac{\sqrt{2}\|\boldsymbol{b}_2(2)\|}{\|\boldsymbol{b}_i(i)\|}\right) &\leqslant (2(m-1))^{\frac{m-1}{2}} \\
\Rightarrow \prod_{i=2}^{m}\left(\frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_i(i)\|}\right) &\leqslant (2(m-1))^{\frac{m-1}{2}} \qquad \square
\end{aligned}$$

6.1.3 Select Basis Procedure

The SELECT_BASIS subroutine is used to return a basis of the lattice that is SVP-reduced, that is, by giving $n+1$ vectors of the lattice (the first one being the shortest vector found by the ENUMERATE subroutine and the rest being the basis vectors), it returns a new basis with the first vector being the shortest vector found by the ENUMERATE subroutine and the rest forming a basis of the lattice. This algorithm finds the shortest vector of $\mathcal{L}$ in the

Algorithm 25 SELECT_BASIS$(n; \boldsymbol{b_1}, \boldsymbol{b_2}, \ldots, \boldsymbol{b_{n+1}})$

Input: $\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n+1})$ vectors that span an n-dimensional subspace
Output: A basis $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ of the lattice $\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n+1})$

1: **if** $n = 0$ **then return** the empty set $\emptyset$ ▷ Termination for recursion

2: **if** $\boldsymbol{b}_1$ is independent of $\boldsymbol{b}_2, \ldots, \boldsymbol{b}_{n+1}$ **then** $\boldsymbol{a}_1 \leftarrow \boldsymbol{b}_1$ ▷ $\boldsymbol{b}_1$ is linearly independent, and must be on basis

3: **else** find $\alpha_2, \ldots, \alpha_{n+1}$ such that $\sum_{j=2}^{n+1} \alpha_j \boldsymbol{b}_j = \boldsymbol{b}_1$ ▷ If $\boldsymbol{b}_1$ is dependent on other vectors, find the shortest vector in $\mathcal{L}$ in its direction

M: least common multiple of the denominators of the α_j's

$\gamma \leftarrow \gcd(M\alpha_2, \ldots, M\alpha_{n+1})$

4: **if** M/γ is integral **then** $\boldsymbol{a}_1 \leftarrow \boldsymbol{b}_1$

5: **else** find $p, q \in \mathbb{Z}$ such that $\frac{M}{\gamma} - \left\lfloor \frac{M}{\gamma} \right\rfloor = \frac{p}{q}$

$a_1 \leftarrow \frac{1}{q} b_1$

▷ The proof that a_1 is in the lattice is in Lemma 6.1.2

Let $\bar{\boldsymbol{b}}_2, \ldots, \bar{\boldsymbol{b}}_{n+1}$ be the projections of $\boldsymbol{b}_2, \ldots, \boldsymbol{b}_{n+1}$ orthogonally to $\boldsymbol{a}_1$.

6: **if** any of the $\bar{\boldsymbol{b}}_i$'s is zero **then** delete it to get a basis $\bar{\boldsymbol{a}}_2, \ldots, \bar{\boldsymbol{a}}_n$ for $\mathcal{L}(\bar{\boldsymbol{b}}_2, \ldots, \bar{\boldsymbol{b}}_{n+1})$ ▷ If any vector is zero, then it was dependent on $\boldsymbol{a}_1$

7: **else** Call SELECT_BASIS$(n-1; \bar{\boldsymbol{b}}_2, \ldots, \bar{\boldsymbol{b}}_{n+1})$ to get $\bar{\boldsymbol{a}}_2, \ldots, \bar{\boldsymbol{a}}_n$ as the basis for $\mathcal{L}(\bar{\boldsymbol{b}}_2, \ldots, \bar{\boldsymbol{b}}_{n+1})$ ▷ Otherwise, recursively call the procedure on the sublattice

8: Find $\boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ in $\mathcal{L}$ such that $\boldsymbol{a}_i$'s projection perpendicular to $\boldsymbol{a}_1$ is $\bar{\boldsymbol{a}}_i$ and $|\langle \boldsymbol{a}_i \, , \, \boldsymbol{a}_1 \rangle| \leqslant 1/2 |\langle \boldsymbol{a}_1 \, , \, \boldsymbol{a}_1 \rangle|$.

▷ Lift back vectors into original Space. Refer to Proposition 6.1.1

9: **return** $(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n)$

direction of $\boldsymbol{b}_1$, calls it $\boldsymbol{a}_1$, then projects $\mathcal{L}$ orthogonal to $\boldsymbol{a}_1$ to get $\bar{\mathcal{L}}$ and recursively calls the procedure to find the basis of this $\bar{\mathcal{L}}$. It is shown in Algorithm 25.

Lemma 6.1.2. *If $\boldsymbol{b}_1$ is dependent on $\boldsymbol{b}_2, \ldots, \boldsymbol{b}_{n+1}$, and M/γ is not integral. Find $p, q \in \mathbb{Z}$ such that $\frac{M}{\gamma} - \left\lfloor \frac{M}{\gamma} \right\rfloor = \frac{p}{q}$. Then, $\boldsymbol{a}_1 = \frac{1}{q}\boldsymbol{b}_1$ is in the lattice.*

Proof. $\boldsymbol{b}_1$ is in the lattice by definition and $\frac{M}{\gamma}\boldsymbol{b}_1$ is also in the lattice since all coefficients are integers. Let $\frac{M}{\gamma}\boldsymbol{b}_1 = \frac{p}{q}\boldsymbol{b}_1$. Thus, since $\frac{p}{q}\boldsymbol{b}_1$ and $\boldsymbol{b}_1 = \frac{q}{q}\boldsymbol{b}_1$ are in the Lattice, and p and q are relatively prime, $\frac{1}{q}\boldsymbol{b}_1$ is also in the lattice. □

Proposition 6.1.4. *Given a set of vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n+1}$ that form a lattice and span an n-dimensional subspace, SELECT_BASIS$(n; \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n+1})$ returns a basis $(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n)$ which is a basis of the lattice $\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n+1})$.*

Proof. Since the vector $\boldsymbol{a}_1$ returned by SELECT_BASIS is the shortest vector of $\mathcal{L}$ in the direction of $\boldsymbol{b}_1$ (See Lemma 6.1.2), and every other vector is orthogonal to to the previous one, the basis returned will have rank n and be a basis of of the lattice $\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n+1})$. □

6.1.4 Analysis of Kannan's SVP Algorithm

Proposition 6.1.5. *The vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ returned by SHORTEST$(n; \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$ form a basis of $\mathcal{L}(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$.*

Proof. Let us analyze step by step and prove by induction.

- **Step 1:** Trivial.
- **Step 2:** The LLL algorithm is called, and it returns a reduced basis of the Lattice.
- **Step 3:** The projections of $\boldsymbol{b}_i$ orthogonal to $\boldsymbol{b}_1$ form a Lattice $\mathcal{L}'$ orthogonal to $\boldsymbol{b}_1$.
- **Step 4:** SHORTEST$(i-1; \ldots)$ is called, and, by induction, the vectors returned $(\bar{\boldsymbol{b}}_i)$ also form a basis of the lattice $\mathcal{L}'$, which is orthogonal to $\boldsymbol{b}_1$.
- **Steps 5-7:** We lift the vectors $(\bar{\boldsymbol{b}}_i)$ back to the original space by adding to them a rational multiple of $\boldsymbol{b}_1$ such that the new vector is in the original Lattice $\mathcal{L}$. With this, we have a basis of the lattice orthogonal to $\boldsymbol{b}_1$, however with all elements belonging to the original Lattice $\mathcal{L}$. Therefore, by adding $\boldsymbol{b}_1$ to the basis, the lifted vectors along with $\boldsymbol{b}_1$ form a basis of the original lattice $\mathcal{L}$.
- **Steps 8-10:** Do not change the basis.
- **Step 11:** The ENUMERATE procedure is called, and it returns a shortest nonzero vector of the lattice $\boldsymbol{v}_1$. It does not change the basis.

- **Step 12:** The SELECT_BASIS procedure is called, and, by Proposition 6.1.4, it returns a basis $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ which is a basis of the lattice $\mathcal{L}(\boldsymbol{v}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, which is the original Lattice $\mathcal{L}$.
- **Step 13:** The projections of $\boldsymbol{b}_i$ orthogonal to $\boldsymbol{b}_1$ form a Lattice $\mathcal{L}'$ orthogonal to $\boldsymbol{b}_1$.
- **Step 14:** SHORTEST$(i-1;\ldots)$ is called, and, by induction, the vectors returned $(\bar{\boldsymbol{b}_i})$ form a basis of the lattice $\mathcal{L}'$, which is orthogonal to $\boldsymbol{b}_1$.
- **Step 15:** This step is similar to steps 5-7. We lift the vectors $(\bar{\boldsymbol{b}_i})$ back to the original space by adding to them a rational multiple of $\boldsymbol{b}_1$ such that the new vector is in the original Lattice $\mathcal{L}$. With this, we have a basis of the lattice orthogonal to $\boldsymbol{b}_1$, however with all elements belonging to the original Lattice $\mathcal{L}$. Therefore, by adding $\boldsymbol{b}_1$ to the basis, the lifted vectors along with $\boldsymbol{b}_1$ form a basis of the original lattice $\mathcal{L}$.

Therefore, the vectors returned form a basis of the original Lattice $\mathcal{L}$.

□

Proposition 6.1.6. *The vectors returned by SHORTEST satisfy:*

1. *for $j = 1, 2, \ldots, n-1$, $v_j(j)$ is the shortest vector in L_j, which is the projection of the lattice $\mathcal{L}$ onto V_j.*
2. *$\|\boldsymbol{v}_i(j) - \boldsymbol{v}_i(j+1)\| \leqslant \frac{\|\boldsymbol{v}_j(j)\|}{2}$, for $i \geqslant j \geqslant 1$ (the projections of a vector onto successive subspaces remain close and the difference is at most half of the shortest vector in the subspace)*

Proof. For the first part, at the end of step 7, the vectors $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ are Size-Reduced ($|\mu_{j,i}| \leqslant 1/2$ for $1 \leqslant i < j \leqslant n$.) and $\boldsymbol{B}_{[i,n]}$ is *SVP-reduced* for $i = 1, \ldots, n$. When the algorithm passes step 8, $\|\boldsymbol{b}_1\| \leqslant 2/\sqrt{3}\|\boldsymbol{b}_2\|$ and thus, at step 11 the vectors $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{j_0-1})$ satisfy all necessary conditions for the ENUMERATE procedure. The procedure returns the shortest vector in the lattice (see Proposition 6.1.3), and that is placed as the first vector of the basis. In the next recursive call of SHORTEST, at step 14, the routine is called on the projection of the lattice onto the subspace orthogonal to the first and shortest vector. Thus, the first vector of the basis returned by the routine is the shortest vector in the projected lattice.

For the second part, from step 6 of the algorithm, we have that $|\mu_{i,1}| \leqslant 1/2$ for $i = 2, \ldots, n$. Also, from the size-reduction condition of the LLL algorithm step, we have that $\mu_{j,i} \leqslant 1/2$ for $1 \leqslant i < j \leqslant n$. We also know that $\|\boldsymbol{v}_j(j)\|$ is the smallest vector in the projection of the lattice onto the subspace orthogonal to the first $j-1$ vectors. Thus, for every recursive call to the SHORTEST routine, we have:

$$\boldsymbol{v}_i(j) = \boldsymbol{b}_i - \sum_{k=1}^{j-1} \mu_{i,k} \boldsymbol{b}_k^*$$

$$\boldsymbol{v}_i(j+1) = \boldsymbol{b}_i - \sum_{k=1}^{j} \mu_{i,k}\boldsymbol{b}_k^*$$

$$\Rightarrow \|\boldsymbol{v}_i(j) - \boldsymbol{v}_i(j+1)\| = \left\|\mu_{i,j}\boldsymbol{b}_j^*\right\| \leqslant \frac{\left\|\boldsymbol{b}_j^*\right\|}{2} \leqslant \frac{\|\boldsymbol{v}_j(j)\|}{2}$$

□

Proposition 6.1.7. *Let j_0 be as defined in step 9 of SHORTEST (Alg. 22), then a shortest nonzero integer combination of $(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{j_0-1})$ is also a shortest nonzero integer combination of $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n$.*

Proof. Suppose the $\boldsymbol{v} = \sum_{i=1}^{n} \alpha_i \boldsymbol{b}_i$ is a shortest nonzero vector and one of $\alpha_{j_0}, \alpha_{j_0+1}, \ldots, \alpha_n$ is nonzero. Therefore, $\boldsymbol{v}'$, the projection of $\boldsymbol{v}$ onto the subspace perpendicular to the span of $(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{j_0-1})$ is nonzero and $\|\boldsymbol{v}'\| \geqslant \|\boldsymbol{b}_{j_0}(j_0)\|$. We also have that $\|\boldsymbol{v}\| \geqslant \|\boldsymbol{v}'\| \geqslant \|\boldsymbol{b}_{j_0}(j_0)\| \geqslant \|\boldsymbol{b}_1\|$. Thus, $\boldsymbol{v}$ is not the shortest vector in the lattice and we have a contradiction. Thus, the shortest vector in the subspace perpendicular to $(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{j_0-1})$ is also the shortest vector in the lattice.

□

Proposition 6.1.8. *The procedure SHORTEST calls itself in lower dimensional lattices at most $\frac{5}{2}n$ times.*

Proof. By step 2 of the algorithm, the basis is LLL-reduced, which returns $\|\boldsymbol{b}_1\| \leqslant 2^{(n-1)/2} \cdot \lambda_1$. Each loop of steps 3 to 8 of the algorithm reduces $\|\boldsymbol{b}_1\|$ by a factor of $\sqrt{3}/2$. Therefore, every 5 loops, $\|\boldsymbol{b}_1\|$ is reduced by a factor of 2. Thus, it needs $\frac{n-1}{2} \cdot 5 \approx \frac{5}{2}n$ loops to reduce $\|\boldsymbol{b}_1\|$ to λ_1. Finally, at step 14, the procedure is called once more. □

Proposition 6.1.9. *The call to the ENUMERATE procedure in SHORTEST tries out at most $(2n)^{n/2}$ sets of values.*

Proof. By Lemma 6.1.1, given an input that satisfies the conditions of the ENUMERATE procedure, it tries out at most $(2(m-1))^{(m-1)/2}$ sets of values. Since the input to the ENUMERATE is the basis formed by the vectors $(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{j_0-1})$, and $j_0 \leqslant n+1$, we have that the number of sets of values tried out is at most $(2n)^{n/2}$. □

Theorem 6.1.3. *When given an input vectors with integer components, and maximum length B, the algorithm SHORTEST runs in time $n^{\mathcal{O}(n)}$.*[2]

Proof. Let $T(n)$ be the maximum number of arithmetic operations performed by the routine SHORTEST$(n; \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n)$. Then, we have that all steps, except the recursive calls to SHORTEST$(n-1; \ldots)$ and the call to ENUMERATE take a polynomial number of operations. Therefore, considering q some polynomial, we have:

[2] Kannan leaves the time complexity at $\mathcal{O}\left((4n^{3/2})^n (\log B)(\log\log B)\right)$

$$
\begin{aligned}
T(n) &\leqslant \frac{5}{2}n \cdot T(n-1) + q(n) + (2n)^{n/2} \\
&\leqslant \frac{5}{2}n \cdot T(n-1) + 2(2n)^{n/2} && \text{(By Proposition 6.1.9)}
\end{aligned}
$$

Analyzing the recursion:

$$
\begin{aligned}
T(n) &\leqslant \frac{5}{2}n \cdot T(n-1) + (2n)^{n/2} \\
&\leqslant \frac{5}{2}n \left(\frac{5}{2}(n-1) \cdot T(n-2) + (2(n-1))^{(n-1)/2} \right) + (2n)^{n/2} \\
&\leqslant \frac{5}{2}n \Bigg(\frac{5}{2}(n-1) \Bigg(\frac{5}{2}(n-2)\, T(n-3) + \big(2(n-2)\big)^{(n-2)/2} \Bigg) \\
&\qquad + \big(2(n-1)\big)^{(n-1)/2} \Bigg) + \big(2n\big)^{n/2} \\
&\;\;\vdots \\
&\leqslant \left(\frac{5}{2}\right)^n \cdot n! \cdot T(1) + \sum_{i=0}^{n-1} \left(\frac{5}{2}\right)^i c \cdot (2(n-i))^{\frac{n-i}{2}} \\
&\leqslant \left(\frac{5}{2}\right)^n \cdot n! + n! \cdot \sum_{i=0}^{n-1} \left(\frac{5}{2}\right)^i \cdot \frac{1}{(n-i)!} \cdot (2n)^{\frac{n}{2}} \\
&\leqslant \left(\frac{5}{2}\right)^n \cdot n! + n! \cdot (2n)^{n/2} \sum_{i=0}^{n-1} \left(\frac{5}{2}\right)^i \cdot \frac{1}{(n-i)!} \\
&\leqslant \left(\frac{5}{2}\right)^n \cdot n! + n! \cdot (2n)^{n/2} \sum_{i=0}^{n-1} \left(\frac{5}{2}\right)^i \\
&\leqslant \left(\frac{5}{2}\right)^n \cdot n! + n! \cdot (2n)^{n/2} \cdot \left(\frac{5}{2}\right)^n
\end{aligned}
$$

Thus,

$$
\begin{aligned}
T(n) &= \mathcal{O}\left(\left(\frac{5}{2}\right)^n \cdot n! \cdot (2n)^{n/2} \right) && (n! \leqslant n^n) \\
&= \mathcal{O}\left(\left(\frac{5}{2}\right)^n \cdot 2^{n/2} \cdot \left(n^{3/2}\right)^n \right) \\
&= \mathcal{O}\left(\left(\frac{5}{\sqrt{2}}\right)^n \cdot \left(n^{3/2}\right)^n \right) \\
&= \mathcal{O}\left(\left(\frac{5}{\sqrt{2}} \cdot n^{3/2}\right)^n \right)
\end{aligned}
$$

Thus $T(n) = \mathcal{O}\left((\frac{5}{\sqrt{2}} n^{3/2})^n\right)$[3].

By Lemma 6.1.4, each operation is performed on operands of size at most $\mathcal{O}(n^2(\log B + \log n))$. Assuming that each operation on size n operands takes $\mathcal{O}(n \log n)$ operations, and $\log B >> n$, we have that the total time complexity is:

$$\begin{aligned}
\text{Time Complexity} &= \mathcal{O}\left(\left((3 \cdot 9n^{3/2})^n\right) \cdot \left(n^2(\log B + \log n)\right) \cdot (2\log n + \log\log B + \log\log n)\right) \\
&= \mathcal{O}\left((27n^{3/2})^n \cdot n^2 \cdot \log B \cdot \log\log B\right) \\
&= \mathcal{O}\left((27n^{3/2})^n \cdot \log B \cdot \log\log B\right) \\
&= n^{\mathcal{O}(n)}
\end{aligned}$$

□

Size of Numbers in the Algorithm

Proposition 6.1.10. $\max_{i=1,2,\ldots,n} \|\boldsymbol{b}_i(i)\|$ *do not increase in steps 9 through 12 of* `SHORTEST` *routine.*

Proof. Let $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n$ be the basis of the lattice at step 9 of the algorithm and let $\boldsymbol{v}_1$ the shortest vector returned by the `ENUMERATE` procedure. Let $\boldsymbol{u}_1 = \boldsymbol{v}_1, \boldsymbol{u}_2 = \boldsymbol{b}_1, \boldsymbol{u}_3 = \boldsymbol{b}_2, \ldots, \boldsymbol{u}_{n+1} = \boldsymbol{b}_n$. Since the vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{n+1}$ form a lattice and span an n-dimensional subspace, exactly one $\boldsymbol{u}_j(j) = 0$, assume this is $\boldsymbol{u}_j(j)$. Also, let $\hat{\boldsymbol{b}}_1, \hat{\boldsymbol{b}}_2, \ldots, \hat{\boldsymbol{b}}_n$ be the basis returned by the `SELECT_BASIS`$(\boldsymbol{v}_1, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ routine on step 12 of the algorithm, with $\hat{\boldsymbol{b}}_1 = \boldsymbol{v}_1$. For $i = 2, 3, \ldots, j-1$, the algorithm chooses $\hat{\boldsymbol{b}}_i(i)$ such that $\left\|\hat{\boldsymbol{b}}_i(i)\right\| \leqslant \|\boldsymbol{u}_i(i)\|$ so it follows that:

$$\left\|\hat{\boldsymbol{b}}_i(i)\right\| \leqslant \|\boldsymbol{u}_i(i)\| \leqslant \|\boldsymbol{b}_{i-1}(i-1)\|, \quad \text{for } i = 2, 3, \ldots, j-1 \tag{6.1}$$

For $i = j, j+1, \ldots, n$, the vectors $\hat{\boldsymbol{b}}_i(i)$ are chosen as the remaining $\boldsymbol{b}_i(i)$ vectors, so it follows that:

$$\left\|\hat{\boldsymbol{b}}_i(i)\right\| = \|\boldsymbol{b}_i(i)\|, \quad \text{for } i = j, j+1, \ldots, n \tag{6.2}$$

and finally:

$$\left\|\hat{\boldsymbol{b}}_1(1)\right\| = \|\boldsymbol{v}_1(1)\| \leqslant \|\boldsymbol{b}_1(1)\| \tag{6.3}$$

□

Proposition 6.1.11. $\max_{i=1,2,\ldots,n} \|\boldsymbol{b}_i(i)\|$ *never increases in the* `SHORTEST` *routine.*

[3] Kannan [Kan83] proves a more relaxed bound $T(n) = \mathcal{O}\left((3 \cdot 9n^{3/2})^n\right)$

Proof. Considering the algorithm, step by step, and by induction on the calls to the SHORTEST routine. For $n = 1$ the algorithm returns $\boldsymbol{b}_1$, thus the proposition holds. For $n > 1$, at step 2, the LLL algorithm does not increase the quantity $\max_{i=1,2,\ldots,n} \|\boldsymbol{b}_i(i)\|$ [LLL82]. At step 4, the recursion call to SHORTEST, the induction hypothesis suffices. At step 8, where, it can only decrease $\boldsymbol{b}_1(1)$ (if $\|\boldsymbol{b}_2\| < \sqrt{3}/2\|\boldsymbol{b}_1\|$) or stay the same, and $\boldsymbol{b}_2(2)$ does not increase, and the rest $\boldsymbol{b}_i(i), i = 3, \ldots, n$ do not change. For steps 9 through 12, the proposition holds from the previous proposition 6.1.10. For step 14, the induction hypothesis suffices. □

Let $d_i := \det\left(\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_i)^T \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_i)\right)$. This can also be seen as the matrix with entries $\langle \boldsymbol{b}_j \, , \, \boldsymbol{b}_k \rangle$, for $1 \leqslant j, k \leqslant i$. Since the original basis vectors had integers coordinates, then this is true for any other basis and d_i is an integer. Therefore, we also have that:

$$d_i = \prod_{j=1}^{i} \|\boldsymbol{b}_j(j)\|_2^2 \tag{6.4}$$

Proposition 6.1.12. *All numbers produced by SHORTEST algorithm are rationals of the form p/q where p and q are integers and q is one of the d_i.*

Proof. $\boldsymbol{b}_j(i) = \boldsymbol{b}_j - \sum_{k=1}^{i-1} \mu_{j,k} \boldsymbol{b}_k$ and $\mu_{j,k} \in \mathbb{R}$, for $j \geqslant i \geqslant 2$. Since we know that $\langle \boldsymbol{b}_\ell \, , \, \boldsymbol{b}_j(i) \rangle = 0$, for $1 \geqslant \ell \geqslant i - 1$, we have that:

$$\langle \boldsymbol{b}_\ell \, , \, \boldsymbol{b}_j \rangle = \sum_{k=1}^{i-1} \mu_{j,k} \langle \boldsymbol{b}_\ell \, , \, \boldsymbol{b}_k \rangle, \quad \text{for } \ell = 1, 2, \ldots, i-1 \tag{6.5}$$

These are $(i - 1)$ independent equations with $(i - 1)$ variables $\mu_{j,k}$, and with a matrix with determinant d_{i-1}. Thus, $(d_{i-1} \cdot \mu_{j,k})$ is an integer. Hence, $(d_{i-1} \cdot \boldsymbol{b}_j(i))$ is an integer vector. □

Proposition 6.1.13. *The vectors produced by the SHORTEST$(n; \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ algorithm, except in the call to LLL (step 2), ENUMERATE (step 11) and SELECT_BASIS (step 12), have length at most $n\sqrt{B}$, where each $\boldsymbol{b}_i$ consists of integral vectors such that $\|\boldsymbol{b}_i\|_2 \leqslant \sqrt{B}$.*

Proof. Since $\|\boldsymbol{b}_i(i)\|$ can increase for $i > 1$, we have to prove that the vectors produced by the recursion on SHORTEST$(i; \boldsymbol{a}_1, \ldots, \boldsymbol{a}_i)$, where $\boldsymbol{a}_j = \boldsymbol{b}_{n-i+j}(n - i + j)$, for $j = 1, 2, \ldots, i$, are not too long. Let $\sqrt{B}$ be the maximum length of the vectors $\boldsymbol{b}_i$, which were the original input to the algorithm. For $i = 1$, it is trivial. Therefore, consider the following proof, step by step, for $i \geqslant 2$:

- **Step 1:** Trivial.
- **Step 2:** The LLL algorithm is called, and this step is excluded from the proof.
- **Step 3:** Does not change the length of the vectors.

- **Step 4:** SHORTEST$(i-1;\ldots)$ is called, and by induction, the vectors produced have length at most $(i-1)\sqrt{B}$.
- **Steps 5-7:** When lifting the vectors back to the original space, we add a multiple of the vectors such that it is no more than half of the shortest vector in the subspace. Thus, the length of the vectors are at most $(i-1)\sqrt{B}+\sqrt{B}=i\sqrt{B}$.
- **Steps 8-10:** Do not affect the length of the vectors.
- **Step 11:** The ENUMERATE procedure is called, and this step is excluded from the proof.
- **Step 12:** The SELECT_BASIS procedure is called, and this step is excluded from the proof, however, it is important to note that the basis returned $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_i$ such that $\|\boldsymbol{a}_\ell(\ell)\| \leqslant \sqrt{B}$ for $\ell=1,2,\ldots,i$, from Proposition 6.1.10.
- **Steps 13-15:** These steps are similar to steps 3-7. When lifting the vectors back to the original space, we add a multiple of the vectors such that it is no more than half of the shortest vector in the subspace. Thus, the length of the vectors are at most $(i-1)\sqrt{B}+\sqrt{B}=i\sqrt{B}$.

□

Proposition 6.1.14. *The execution of the LLL algorithm in step 2 of the SHORTEST, all numbers produced are bounded by* $(nB)^{cn^2}$*, for some constant* c*.*

Proof. Whenever the LLL algorithm is called, all vectors $(\boldsymbol{a}_1,\ldots,\boldsymbol{a}_i)$ have rational components with common denominator d, where d is the one of the d_i by Proposition 6.1.12. Since the initial vectors are bounded by $\sqrt{B}$, by Equation 6.4, we have:

$$d \leqslant d_i = \prod_{j=1}^{i} \|\boldsymbol{b}_j(j)\|_2^2 \leqslant B^i. \tag{6.6}$$

By Propostion 6.1.13, throughout the execution of SHORTEST, the length of the vectors is at most $n\sqrt{B}$:

$$\|\boldsymbol{a}_i\| \leqslant n\sqrt{B} \tag{6.7}$$

Also, In the LLL algorithm, all numbers produced when input vectors $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_i$ are bounded by $(\max_{i=1,2,\ldots,n}\|\boldsymbol{a}_i(i)\|)^{cn}$ (Proposition 1.26 [LLL82])[4].

[4]The proposition states that on inputs such that $\|\boldsymbol{b}_i\|^2 \leqslant B$:

1. $\|\boldsymbol{b}_i^*\|^2 \leqslant \|\boldsymbol{b}_i\|^2 \leqslant B$;
2. $\max_i^n\{\|\boldsymbol{b}_i^*\|^2\}$ does not increase during execution of the algorithm;
3. Throughout the course of the algorithm: $\|\boldsymbol{b}_i\|^2 \leqslant n^2(4B)^n$

Thus: $\|\boldsymbol{b}_i\|^2 \leqslant n^2(4\max_i^n\{\|\boldsymbol{b}_i^*\|^2)^n\} \leqslant (\max_i^n\{\|\boldsymbol{b}_i^*\|^2)^{cn}$

Finally, the LLL algorithm behaves identically on input $d\boldsymbol{a}_1, \ldots, d\boldsymbol{a}_i$ and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_i$, since both sets are of integers vectors, thus the bounds proved in the LLL paper are valid. In the former case, with inputs $(d\boldsymbol{a}_1, \ldots, d\boldsymbol{a}_i)$, the numbers produced during execution are bounded by $(\max_{i=1,2,\ldots,n} \|d\boldsymbol{a}_i(i)\|)^{cn}$. Which is at most

$$(\max_i^n \|d\boldsymbol{a}_i(i)\|)^{cn} \leqslant (B^{n-1} \cdot n\sqrt{B})^{cn} \leqslant (nB)^{cn^2} \tag{6.8}$$

□

Proposition 6.1.15. *The calls to* ENUMERATE *(step 11) and* SELECT_BASIS *(step 12) do not produce any number of magnitude greater than* $n^{cn}B$.

Proof. In the ENUMERATE procedure, each α_k is an integer such that $\alpha_k \leqslant \frac{\|\boldsymbol{b}_1\|}{\|\boldsymbol{b}_k(k)\|}$ (By Proposition 6.1.2). Therefore, we have:

$$\begin{aligned}
\|\boldsymbol{b}_1\| &\leqslant \sqrt{n} \cdot |\det(\mathcal{L})|^{1/n} && \text{(By Minkowski's theorem)} \\
\Rightarrow \frac{\|\boldsymbol{b}_1\|^n}{\prod_{i=1}^n \|\boldsymbol{b}_i(i)\|} &\leqslant n^{n/2} \\
\Rightarrow \prod_{i=1}^n \alpha_i &\leqslant n^{n/2} \\
\Rightarrow \alpha_k &\leqslant n^{n/2} \cdot \frac{1}{\prod_{i \neq k} \alpha_i} \leqslant n^{n/2}
\end{aligned}$$

Thus, all multipliers of the vectors are at most n^{cn} for some constant c. In the SELECT_BASIS procedure, since we always try to find the shortest vector in the direction of the first vector, and the procedure is called recursively on the basis vectors orthogonal to the first vector, the multipliers are bounded by n^{cn}. □

Lemma 6.1.4. *In the* SHORTEST$(n; b_1, b_2, \ldots, b_n)$ *algorithm, with inputs* $b_1, b_2, \ldots, b_n$, *independent vectors with components of length at most* $\sqrt{B}$, *all numbers produced can be represented in* $\mathcal{O}(n^2(\log B + \log n))$ *bits. Furthermore, the numbers produced by the algorithm except the call to LLL (step 2) can be represented in* $\mathcal{O}(n \log n + \log B)$ *bits.*

Proof. By proposition 6.1.14, the numbers produced by the LLL algorithm are bounded by $(nB)^{cn^2}$, for some constant c, thus, the numbers of bits produced by the SHORTEST algorithm are bounded by $\mathcal{O}(n^2(\log B + \log n))$. During the execution of the algorithm, except during the call to LLL, all numbers are bounded by n^{cn} by proposition 6.1.15. Thus, the numbers produced by the algorithm except the call to LLL are bounded by $\mathcal{O}(n \log n + \log B)$. □

6.2 AKS Sieve Algorithm for the SVP

Sieving is a randomized algorithm used to find short vectors in a lattice. The general idea is that when you subtract any two lattice vectors, the result is also a lattice vector. Considering this, the algorithm works by sampling a large enough set of lattice vectors in a ball of radius R and subtracting them. Then, it recursively chooses the pairs of vectors that are shorter than the current vectors and repeats the process.

The first provable algorithm was proposed by [AKS01]. The idea of the algorithm is to (uniformly) choose a large enough set of lattice points inside a sufficiently large ball and perturb the points by a small value (Shown in `SAMPLE` procedure). Then, recursively apply the `SIEVE` procedure, which produces another set of vectors with the longest being about half of the original length while still maintaining the set large enough. This is done until the set of vectors are inside a ball of constant radius and that there is a reasonable chance that a short vector and its nearest neighbor are chosen by this procedure, where the difference results in the shortest vector.

6.2.1 SVP Algorithm

First, we give an overview of the algorithm, which is presented in Algorithm 26 (`AKS SVP`). Throughout the presentation, if the norm is not specified, it is in regards to ℓ_2 ($\|\cdot\| = \|\cdot\|_2$). This exposition follows more closely [Reg04a], rather than [AKS01] for simplicity.

Algorithm 26 `AKS SVP`

Input: $\boldsymbol{B} = \{\, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \,\}$ and considering that the Lattice $\mathcal{L} = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ satisfies:

$2 \leqslant \lambda_1(\mathcal{L}) < 3$;

Output: A shortest non-zero vector of $\mathcal{L}(\boldsymbol{B})$.

$List \leftarrow \emptyset$

$R_0 \leftarrow n \cdot \max_i \|\boldsymbol{b}_i\|$ ▷ Initial Radius

$N \leftarrow 2^{8n} \log R_0$ ▷ Number of Samples

for $i = 1$ to N **do** ▷ Sample N pairs of vectors and add to $List$

  Add `SAMPLE`($\boldsymbol{B}$) to $List$

  ▷ Returns a pair $(\boldsymbol{y}, \boldsymbol{x} = \boldsymbol{z} + \boldsymbol{y})$, where $\boldsymbol{y} \in B_n(2)$ is a small perturbation and $\boldsymbol{z} \in \mathcal{L}(\boldsymbol{B})$ and $\boldsymbol{x} \in B_n(R_0)$

$R \leftarrow R_0$

while $R > 6$ **do** ▷ Runs the SIEVE algorithm until the radius is less than a constant

  $List \leftarrow$ `SIEVE`($List, R$)

  $R \leftarrow R/2 + 2$

Compute $\boldsymbol{v} \in \mathcal{L}$ *s.t.* $\|\boldsymbol{v}\| = \min_{i,j \in List} \{\, \|(\boldsymbol{x}_i - \boldsymbol{y}_i) - (\boldsymbol{x}_j - \boldsymbol{y}_j)\| \mid (\boldsymbol{x}_i - \boldsymbol{y}_i) \neq (\boldsymbol{x}_j - \boldsymbol{y}_j) \,\}$

▷ Consider all pairs of vectors $(\boldsymbol{y}_i, \boldsymbol{x}_i) \in List$ and return minimal non-zero difference of $\boldsymbol{z}_i$'s.

return $\boldsymbol{v}$

6.2.2 Pre-processing

In order to facilitate analysis, there is a simplifying assumption that $2 \leqslant \lambda_1(\mathcal{L}) < 3$. Consider the following Lemma:

Lemma 6.2.1. *There exists a constant $k \in \{1, \ldots, 2n\}$ such that for any Lattice $\mathcal{L}(\boldsymbol{B})$ there is a scaled lattice $\mathcal{L}(\boldsymbol{B}_k)$ such that $2 \leqslant \lambda_1(\mathcal{L}(\boldsymbol{B}_k)) < 3$.*

Proof. Given any basis $\boldsymbol{B}$, we can apply the LLL algorithm ([LLL82]) to obtain a vector $\boldsymbol{b}_1$ such that:

$$\begin{aligned} &\lambda_1(\mathcal{L}) \leqslant \|\boldsymbol{b}_1\| \leqslant 2^{(n-1)/2}\lambda_1(\mathcal{L}) < 2^n\lambda_1(\mathcal{L}) \\ \Rightarrow\ &\lambda_1(\mathcal{L}) \leqslant \|\boldsymbol{b}_1\| < 2^n\lambda_1(\mathcal{L}) \end{aligned}$$

Then, we can create a family of lattices $\mathcal{L}(\boldsymbol{B}_k)$ such that the original basis $\boldsymbol{B}$ is scaled as follows:

$$\boldsymbol{B}_k = \frac{(3/2)^k}{\|\boldsymbol{b}_1\|}\boldsymbol{B}$$

This creates lattices $\mathcal{L}(\boldsymbol{B}_k)$ where their shortest vector is scaled by the factor $\frac{(3/2)^k}{\|\boldsymbol{b}_1\|}$, and, therefore:

$$\begin{aligned} \lambda_1(\mathcal{L}(\boldsymbol{B}_k)) &= \frac{(3/2)^k}{\|\boldsymbol{b}_1\|}\lambda_1(\mathcal{L}(\boldsymbol{B})) \\ \Rightarrow \frac{(3/2)^k}{2^n} &< \lambda_1(\mathcal{L}(\boldsymbol{B}_k)) \leqslant (3/2)^k \end{aligned}$$

Then, there exists a $k \in \{1, \ldots, 2n\}$ such that:

$$2 \leqslant \lambda_1(\mathcal{L}(\boldsymbol{B}_k)) < 3$$

□

6.2.3 Sampling Procedure

The sampling procedure shown below in SAMPLE is similar to the one shown in [Reg04a], where the perturbation $\boldsymbol{y} \in B_n(2)$ is sampled first, and then, using an algorithm similar to Babai's Rounding Off Algorithm, [Bab86] we find a point $\boldsymbol{x} \in \mathcal{P}(\boldsymbol{B})$ and $\boldsymbol{z} \in \mathcal{L}$.

Algorithm 27 SAMPLE

Input: $\boldsymbol{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$.
Output: A lattice vector $\boldsymbol{z}$ and its perturbed variant $\boldsymbol{x} = \boldsymbol{z} + \boldsymbol{y}$.

1: Sample $\boldsymbol{y} \sim B_n(2)$ ▷ Sample a perturbation uniformly from a ball of radius 2
2: $\boldsymbol{x} \leftarrow \boldsymbol{y} \bmod \mathcal{P}(\boldsymbol{B})$ ▷ Calculate the Perturbed lattice point in $\mathcal{P}(\boldsymbol{B})$
3: **return** pair $(\boldsymbol{y}, \boldsymbol{x} = \boldsymbol{z} + \boldsymbol{y})$

In order to calculate $\boldsymbol{x} = \boldsymbol{y} \bmod \mathcal{P}(\boldsymbol{B})$, consider the following steps. First express $\boldsymbol{y}$ in terms of the lattice basis as

$$\boldsymbol{y} := \sum_{i=1}^{n} \alpha_i \boldsymbol{b}_i \qquad \text{, with } \alpha_i \in \mathbb{R},\ \forall i \in [n]$$

Then, we can calculate $\boldsymbol{z}$ as the integer part of each coefficient:

$$\boldsymbol{z} = -\sum_{i=1}^{n} \lfloor \alpha_i \rfloor \boldsymbol{b}_i$$

and finally:

$$\boldsymbol{x} = \boldsymbol{z} + \boldsymbol{y} = \sum_{i=1}^{n} (\alpha_i - \lfloor \alpha_i \rfloor)\boldsymbol{b}_i = \sum_{i=1}^{n} \beta_i \boldsymbol{b}_i \in \mathcal{P}(\boldsymbol{B}) \qquad \text{, with } \beta_i \in [0, 1[$$

Therefore, we have that $\boldsymbol{x} \in \mathcal{P}(\boldsymbol{B})$ and $\boldsymbol{z} \in \mathcal{L}(\boldsymbol{B})$

Since $R_0 = n \cdot \max_i \|\boldsymbol{b}_i\|$, then $\mathcal{P}(\boldsymbol{B})$ is contained in $B_n(R_0)$ and $\boldsymbol{x} \in B_n(R_0)$

6.2.4 Sieving Procedure

The `SIEVE` procedure is the core of the algorithm, where the vectors in a ball of radius R are subtracted from other vectors denoted as centers in a way that the resulting vectors are shorter than $R/2$.

Algorithm 28 `SIEVE`

Input: Radius R and a list $List$ of pairs $(\boldsymbol{y}_r, \boldsymbol{x}_r = \boldsymbol{z}_r + \boldsymbol{y}_r)$.
Output: A list of pairs $List'$

1: $Centers \leftarrow \emptyset$ ▷ "Representatives"
2: $List' \leftarrow \emptyset$
3: **for** $(\boldsymbol{y}_r, \boldsymbol{x}_r) \in List$ **do**
4: **if** $\exists (\boldsymbol{y}_s, \boldsymbol{x}_s) \in Centers$ such that $\|\boldsymbol{x}_r - \boldsymbol{x}_s\| \leqslant R/2$ **then**
▷ Find pair such that distance is small
5: Add $(\boldsymbol{y}_r, \boldsymbol{x}_r - (\boldsymbol{x}_s - \boldsymbol{y}_s))$ to $List'$.
▷ If there is such pair, add the new vector ($\in \mathcal{L}$), centred, with the smaller size to the $List$
6: **else**
7: Add $(\boldsymbol{y}_r, \boldsymbol{x}_r)$ to $Centers$. ▷ Otherwise, create new "Center"
8: **return** $List'$

Remark. *In the pseudocode of the `SIEVE` procedure, the variable Centers stores the actual representative points* $(\boldsymbol{y}_s, \boldsymbol{x}_s)$.

In the Lemma 6.2.2 below, Centers denotes the set of indices $s \in [N]$ *of those representative points.*

Lemma 6.2.2. *There is an algorithm that, for any set of points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in B_n(R)$, we can find a subset $Centers \subset [N]$ of size at most 5^n such that:*

$$(\forall r \in [N] \setminus Centers)(\exists s \in Centers)\ \|\boldsymbol{x}_r - \boldsymbol{x}_s\| \leqslant R/2$$

The running time is polynomial in the input size.

Proof. Consider the SIEVE procedure. We initialize the $Centers$ to be the empty set and, for each point in the set $r \in [N]$, if there is a point $s \in Centers$ such that the distance $\|\boldsymbol{x}_r - \boldsymbol{x}_s\| \leqslant R/2$, then we "recenter" the point $\boldsymbol{x}_r$ by subtracting $\boldsymbol{x}_s$ and add it to the list $List'$. Otherwise, we add $\boldsymbol{x}_r$ to the set $Centers$.

To show that $|Centers| \leqslant 5^n$, we can use the fact that the distance between two points in $Centers$ is at least $R/2$. Therefore, if we consider balls of radius $R/4$ around each point in $Centers$, then the balls are disjoint. On the other hand, their union is contained in $B_n(5R/4)$. Therefore, the number of balls (and $|Centers|$) is at most:

$$|Centers| \leqslant \frac{\mathrm{vol}(B_n(5R/4))}{\mathrm{vol}(B_n(R/4))} = 5^n$$

Finally, the running time of the algorithm is polynomial in the input size, since for every element in $[N]$, we have to check all elements in $Centers$ to see if they are close enough. □

Lemma 6.2.3. *During the execution of the SIEVE procedure, if $r \in List'$ then the decomposition of $\boldsymbol{x}_r$ as $\boldsymbol{z}_r + \boldsymbol{y}_r$ is never used.*

Proof. During step 4 of the procedure, the only condition checked is the distance between $\boldsymbol{x}_r$ and $\boldsymbol{x}_s$. If the condition passes, the pair $(\boldsymbol{y}_s, \boldsymbol{x}_x) \in Centers$ is used to update the new $\boldsymbol{x}_r \in List'$. This is done by subtracting a vector $\boldsymbol{x}_s - \boldsymbol{y}_s = \boldsymbol{z}_s \in \mathcal{L}$ to the old $\boldsymbol{x}_r$. □

6.2.5 Analysis of the AKS Sieve Algorithm

Lemma 6.2.4. *The total running time of the AKS Sieve algorithm is $2^{\mathcal{O}(n)}$.*

Proof. The algorithm starts with vectors in a ball of radius R_0, and, after each call to the SIEVE procedure, the ball is reduced by a factor of 2 (plus a small constant). Therefore, the number of calls to the SIEVE procedure is at most $2 \log R_0$.

By Lemma 6.2.2, each call to the SIEVE procedure takes a running time polynomial in the input size, and thus, for an initial input of $N = 2^{8n} \log R_0$, the total running time is $2 \log R_0 \cdot 2^{\mathcal{O}(n)} = 2^{\mathcal{O}(n)}$. □

Lemma 6.2.5. *During every call to the SIEVE procedure, we have that $\forall (\boldsymbol{y}_r, \boldsymbol{x}_r = \boldsymbol{z}_r + \boldsymbol{y}_r) \in List$, $\|\boldsymbol{x}_r\| \leqslant R$ and $\boldsymbol{z}_r \in \mathcal{L}$.*

Proof. Initially, by the SAMPLE procedure (Lemma), we have that $\forall (\boldsymbol{y}_r, \boldsymbol{x}_r) \in List$:

- $\boldsymbol{x}_r - \boldsymbol{y}_r = \boldsymbol{z}_r \in \mathcal{L}$
- $\|\boldsymbol{x}_r\| \leqslant \|\boldsymbol{z}_r\| + \|\boldsymbol{y}_r\| \leqslant \sqrt{n}\max_i\|\boldsymbol{b}_i\| + 2 \leqslant n\max_i\|\boldsymbol{b}_i\| = R_0$

Then, in every call to the SIEVE procedure,

- Initially: $\|\boldsymbol{x}_r\| \leqslant R$ and $\boldsymbol{x}_r - \boldsymbol{y}_r = \boldsymbol{z}_r \in \mathcal{L}$
- Step: Find $s \in Centers, \|\boldsymbol{x}_r - \boldsymbol{x}_s\| \leqslant R/2$
- Update: $\boldsymbol{x}_r = \boldsymbol{x}_r - (\boldsymbol{x}_s - \boldsymbol{y}_s)$:

$$\begin{aligned}\|\boldsymbol{x}_r\| &= \|\boldsymbol{x}_r - (\boldsymbol{x}_s - \boldsymbol{y}_s)\| \\ &= \|(\boldsymbol{x}_r - \boldsymbol{x}_s) + \boldsymbol{y}_s\| \\ &\leqslant \|\boldsymbol{x}_r - \boldsymbol{x}_s\| + \|\boldsymbol{y}_s\| \\ &\leqslant R/2 + 2\end{aligned}$$

Thus, since for the next call to SIEVE, $R \leftarrow R/2 + 2$, the (new) $\|\boldsymbol{x}_r\| \leqslant R$.

Also, for the (new) $\boldsymbol{z}_r$, we have:

$$\begin{aligned}\boldsymbol{z}_r &= \boldsymbol{x}_r - \boldsymbol{y}_r \\ &= (\boldsymbol{x}_r - (\boldsymbol{x}_s - \boldsymbol{y}_s)) - \boldsymbol{y}_r \\ &= (\boldsymbol{x}_r - \boldsymbol{y}_r) - (\boldsymbol{x}_s - \boldsymbol{y}_s) \\ &= \boldsymbol{z}_r - \boldsymbol{z}_s \in \mathcal{L}\end{aligned}$$

□

Lemma 6.2.6. *After the SIEVE loop in the AKS SVP (Step 11),* $|List| \geqslant (2^{8n} - 2\cdot 5^n)\log R_0$ *and* $\forall(\boldsymbol{z}_r, \boldsymbol{x}_r) \in List$, $\|\boldsymbol{z}_r\| \leqslant 8$.

Proof. By Lemma 6.2.2, at each call to the SIEVE procedure, we remove at most $|Centers| \leqslant 5^n$ from $List$. Therefore, after at most $(2\log R_0)$ calls to the procedure (by Lemma 6.2.4), we have that:

$$|List| \geqslant N - 2\log R_0 \cdot 5^n = (2^{8n} - 2\cdot 5^n)\log R_0$$

Also, by Lemma 6.2.5, at (Step 11) we have that $\forall r \in List, \|\boldsymbol{x}_r\| < R = 6$, thus, for all $r \in List$:

$$\|\boldsymbol{z}_r\| = \|\boldsymbol{x}_r - \boldsymbol{y}_r\| \leqslant \|\boldsymbol{x}_r\| + \|\boldsymbol{y}_r\| \leqslant 6 + 2 = 8$$

□

The main issue with this algorithm is that, even though in step 11 we have a list of many short vectors (at least $(2^{8n} - 2 \cdot 5^n) \log R_0$), there might be the problem that all of them are the zero vector. To deal with this, we argue that because of the randomization used by the algorithm, and considering that the Sieve step is oblivious to the decomposition of $\boldsymbol{x}_r$ as $\boldsymbol{z}_r + \boldsymbol{y}_r, \forall r \in List$ (by Lemma 6.2.3), then it must produce some nonzero vectors. Consider the following analysis (by [Reg04a]), which changes the algorithm to only sample the perturbation $\boldsymbol{y}_r$ when it is used:

Let $\boldsymbol{v}$ be the shortest nonzero vector of $\mathcal{L}(\boldsymbol{B})$, which, by Lemma 6.2.1, $2 \leqslant \|\boldsymbol{v}\| < 3$. Let $C_1 = B_n(0, 2) \cap B_n(\boldsymbol{v}, 2)$ and $C_2 = B_n(0, 2) \cap B_n(-\boldsymbol{v}, 2)$, as shown in the image 6.1.

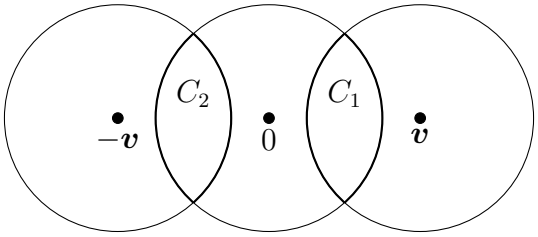


Figure 6.1: Tossing of $\boldsymbol{y}$

Lemma 6.2.7. *Let $C_1 = B_n(0, 2) \cap B_n(\boldsymbol{v}, 2)$ and $C_2 = B_n(0, 2) \cap B_n(-\boldsymbol{v}, 2)$,* (with $\boldsymbol{v}$ the shortest nonzero vector, so $2 \leq \|\boldsymbol{v}\| < 3$ by Lemma 6.2.1). *Then:*

$$\frac{vol(C_1)}{vol(B_n(0,2))} = \frac{vol(C_2)}{vol(B_n(0,2))} \geqslant 2^{-2n}$$

Proof. The equality can be proven by symmetry. For the inequality, consider that the smallest volume of C_1 happens when $\|\boldsymbol{v}\| \approx 3$. In that case, notice C_1 contains a cylinder of height $1/2$ and radius $\sqrt{15/16}$ centered around $\boldsymbol{v}/2$. The cylinder is oriented along the direction of $\boldsymbol{v}$, extending $1/4$ in both directions. The volume of C_1 is bounded by:

$$\text{vol}(C_1) \geqslant \frac{1}{2} \cdot \text{vol}\left(B_{n-1}\left(\sqrt{\frac{15}{16}}\right)\right)$$

We can also bound upper bound $B_n(0, 2)$ by a cylinder of height 4 and radius 2:

$$\text{vol}(B_n(0, 2)) \leqslant 4 \cdot \text{vol}(B_{n-1}(2))$$

Therefore:

$$\frac{\text{vol}(C_1)}{\text{vol}(B_n(0,2))} \geqslant \frac{1}{8} \cdot \frac{\text{vol}\left(B_{n-1}\left(\sqrt{\frac{15}{16}}\right)\right)}{\text{vol}(B_{n-1}(2))} = \frac{1}{8}\left(\frac{15}{16 \cdot 4}\right)^{(n-1)/2} \geqslant \left(\frac{1}{4}\right)^{(n-1)} \geqslant 2^{-2n}$$

□

Define a bijection τ on $B_n(0,2)$ that maps C_1 to C_2, C_2 to C_1 and the rest to itself:

$$\tau(\boldsymbol{y}) = \begin{cases} \boldsymbol{y} + \boldsymbol{v} & \boldsymbol{y} \in C_2 \\ \boldsymbol{y} - \boldsymbol{v} & \boldsymbol{y} \in C_1 \\ \boldsymbol{y} & \text{otherwise} \end{cases}$$

Since $\boldsymbol{v} \in \mathcal{L}$, then, for any $\boldsymbol{y} \in B_n(0,2)$ the sampling of $\boldsymbol{x} = \boldsymbol{y} \bmod \mathcal{P}(B) = \tau(\boldsymbol{y}) \bmod \mathcal{P}(B)$, and, if $\boldsymbol{y}$ is chosen uniformly from $B_n(0,2)$, then $\tau(\boldsymbol{y})$ is also uniform on $B_n(0,2)$.

Remark (analytic use only)**.** *The map τ depends on $\boldsymbol{v}$ and is used solely in the analysis to couple the mass of C_1 and C_2; it is never evaluated by the algorithm. The algorithm samples $\boldsymbol{y}$ uniformly in $B_n(0,2)$ and outputs $\boldsymbol{y} \bmod \mathcal{P}(B)$. Since $\boldsymbol{v} \in \mathcal{L}$, translations by $\pm\boldsymbol{v}$ leave the reduction invariant: $\boldsymbol{y} \bmod \mathcal{P}(B) = (\boldsymbol{y} \pm \boldsymbol{v}) \bmod \mathcal{P}(B)$.*

Modification 1. Now we modify the AKS SVP algorithm so that when calling the SAMPLE procedure, with probability $1/2$ we apply the bijection τ on the sampled perturbation $\boldsymbol{y}$. Let this operation be known as "*tossing* $\boldsymbol{y}$". Since $\boldsymbol{y}$ and $\tau(\boldsymbol{y})$ are identically distributed, the resulting perturbed vector $\boldsymbol{x}$ remains the same and the modified algorithm behaves the same as the original.

Modification 2. The next modification we can apply is, instead of *tossing* each perturbation during the initial SAMPLE procedure, we can delay it until the perturbation $\boldsymbol{y}$ is actually used in the algorithm (i.e. in the SIEVE procedure for all the elements in *Centers* and in the beginning of Step 11 of the AKS SVP for the rest of the elements in *List*). This modification also does not change the algorithm, since each $\boldsymbol{x}$ is independent of the *tossing* of $\boldsymbol{y}$.

Lemma 6.2.8. *In the initial set $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$, there are at least $2^{6n-1} \log R_0$ points such that $\boldsymbol{y}_i \in C_1 \cup C_2$, with high probability.*

Proof. By Lemma 6.2.7, a sampled point $\boldsymbol{y}_i \in C_1 \cup C_2$ with probability greater than $p = 2^{-2n}$. Let Y be the number of $\boldsymbol{y}_i \in C_1 \cup C_2, i = 1, \ldots, N$. Therefore, the expected number of points in $C_1 \cup C_2$ is given by:

$$\mathbb{E}[Y] = pN \geqslant 2^{-2n} \cdot 2^{8n} \log R_0 \geqslant 2^{6n} \log R_0$$

The variance is bounded by:

$$\mathrm{Var}[Y] = pN(1-p) \leqslant pN$$

The probability there are less than $pN/2$ points is at most:

$$\Pr\left[|Y - \mathbb{E}\left[Y\right]| \geqslant \mathbb{E}\left[Y\right]/2\right] \leqslant \mathrm{Var}\left[Y\right]/(\frac{\mathbb{E}\left[Y\right]}{2})^2 \qquad \text{(by Chebishev's Inequality)}$$

$$\Rightarrow \Pr\left[Y \leqslant \mathbb{E}\left[Y\right]/2\right] = \Pr\left[Y \leqslant pN/2\right] \leqslant \frac{4}{pN}$$

□

Now, with the modification made, and considering the state at the beginning of step 11 and before tossing the $\boldsymbol{y}$, there are at least $(2^{6n-1} - 2 \cdot 5^n) \log R_0 > 2^{5n}$ points with perturbation in $C_1 \cup C_2$, and, for each perturbation $\boldsymbol{y}_i$, we know that $\boldsymbol{x}_i - \boldsymbol{y}_i \in \mathcal{L}$ and $\|\boldsymbol{x}_i - \boldsymbol{y}_i\| \leqslant 8$ (By Lemma 6.2.6.

Lemma 6.2.9. *Let $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$ be a lattice such that $2 \leqslant \lambda_1(\mathcal{L}) < 3$, then, the number of possible lattice points inside a ball $B_n(8)$ is at most 2^{4n}.*

Proof. The distance between two lattice points is at least $\lambda_1(\mathcal{L}) \geqslant 2$. If we put a ball of radius 1 around each point in $B_n(8) \cap \mathcal{L}(\boldsymbol{B})$, the balls are disjoint. On the other hand, these balls are contained in $B_n(9)$. Therefore, the maximum number of lattice points inside the $B_n(8)$ can be bounded by:

$$|B_n(8) \cap \mathcal{L}(\boldsymbol{B})| \leqslant \frac{\mathrm{vol}(B_n(9))}{\mathrm{vol}(B_n(1))} = 9^n \leqslant 2^{4n}$$

□

By Lemma 6.2.9 above and considering that in the beginning of step 11 there are at there least 2^{5n} lattice points inside a ball $B_n(8)$ such that their perturbation is in $C_1 \cup C_2$, then there exists a lattice vector $\boldsymbol{w}$ and at least $2^{5n}/2^{4n} = 2^n$ indices i for which $\boldsymbol{z}_i = \boldsymbol{x}_i - \boldsymbol{y}_i = \boldsymbol{w}$ ($\boldsymbol{y}_i \in C_1 \cup C_2$). Then, after tossing the remaining $\boldsymbol{y}_i, \forall i \in \mathit{List}$, $\boldsymbol{x}_i - \boldsymbol{y}_i$ remains $\boldsymbol{w}$ with probability $1/2$ or becomes either $\boldsymbol{w} + \boldsymbol{v}$ or $\boldsymbol{w} - \boldsymbol{v}$. Therefore the probability that all "throws" are the same, for at least 2^n indices, and we end up with all $\boldsymbol{w}$ is less than 2^{-2^n}, and thus, with a very high probability, in the last step we find a pair of indices i and j such that $(\boldsymbol{x}_i - \boldsymbol{y}_i) - (\boldsymbol{x}_j - \boldsymbol{y}_j) = \boldsymbol{v}$, the shortest vector of the lattice.

Chapter 7

Conclusion

This thesis set out to trace and connect the theoretical lines of research linking the very first lattice-based public-key proposals to the standardized ML-KEM suite, now setup for widespread PQC adoption. Along the way, we aimed for two main goals:

1. Expose the common *worst-case to average-case* "DNA" of lattice-based cryptography, showing the reductions that connect the hardness of lattice problems to the security of public-key schemes, running through the Ajtai-Dwork, Regev's LWE and the Ring-LWE and Module-LWE problems.

2. Examine how the SVP solvers function and exert pressure on the security of lattice-based schemes, and shape the parameters of the cryptosystems.

The main findings can be summarized into the following take-aways:

1. **Reductions remain the standard for confidence:** Starting from the groundbreaking work of Ajtai [Ajt96] and following its first implementation to a public-key cryptosystem by Ajtai and Dwork [AD97], the reductions show that *any* break of an average-case instance would lead to a break of the worst-case lattice problem. Regev's LWE [Reg05] reduction further strengthened this connection, showing another problem that also has a worst-case to average-case reduction, and opens up the door to a wide range of public-key schemes. Chapter 4 showed how the same proof skeleton can be used to prove the security of the passage from LWE to Ring-LWE and Module-LWE, with a careful use of ideal and module lattice geometry. Finally, the standardized ML-KEM suite [MLK24] and its original algorithm CRYSTALS-Kyber [SAB+20] ultimately relies on these reductions, showing the transformation and evolution from a theoretical, but impractical, public-key scheme to a practical, production-ready KEM.

2. **Most optimizations incur in a trade-off:** The optimizations implemented in the ML-KEM scheme are necessary to make the scheme practical, but most introduce certain trade-offs that need to be considered and analyzed in terms of security. We

initially discuss the trade-offs from the introduction of structure when moving from LWE to Ring-LWE (and further to Module-LWE), where even though the more efficient communication and computation (through NTT multiplication) is achieved, this additional structure might be potentially exploited by an attacker, compromising security. Other optimizations worth mentioning is fixing the ring and manipulating the rank in the Module-LWE based schemes, which allow for more efficient implementations and fixed NTT algorithms, but in order to increase security, the parameters need to be adjusted, increasing the size of keys and ciphertexts. Finally, we comment on ciphertext compression, which reduces the size of ciphertexts, but introduces extra error terms in the decryption process and need to be taken into account when choosing the parameters to not increase the decryption failure rate, and allow for other attacks to be mounted.

3. **Exact SVP solvers still limit the possible attacks:** The SVP solvers discussed in Chapter 6 are exponential (or super-exponential) time algorithms, and as such, they are not practical attacks for lattices with large dimensions. The use of module lattices, as in the ML-KEM suite, when no other attack path that makes use of the module structure is known, generates lattices that have very large dimensions ($N = kn$), which increases the perceived security of the scheme. However, it is important to note that even though the exact SVP solvers for larger dimensions are not practical, a successful attack would only need to leverage *approximate* SVP solvers for certain approximation factors, which can be much more efficient. Many of the best algorithms to attack lattices, such as BKZ [SE94] and its variations, aim for that approach, running until they find a short enough vector that can be used to break the scheme, however, they still make use of the SVP solvers in smaller dimensions and for smaller approximation factors, which still limits how effective they can be against the Module-LWE based schemes.

4. **Migration paths to PQC are visible, but not fixed:** The standardization of the ML-KEM suite has directed the community towards it for KEMs, but it is important to note that the path to PQC is not fixed and there are still many open questions and research directions such as:

    - developing tighter (ideally classical) reductions for Module-LWE based schemes and more accurate methods to analyze and estimate the security of the schemes;
    - designing hybrid protocols and schemes that layer newer PQC algorithms over existing classical ones in a way that leverages hardness assumptions without the need of very long keys and ciphertexts; and
    - proving stronger worst-case links for structured lattice problems (ideal and module lattices), so that the security of the schemes can be better understood and estimated.

In conclusion, from Ajtai-Dwork's first bulky public-key scheme proposal to the practical ML-KEM suite, lattice cryptography transitioned from a theoretical solution to an industry standard. This thesis charted that journey and highlighted the key reductions

and optimizations that made it possible, providing a framework for understanding the security of lattice-based schemes and reasoning about future evolutions and developments. Advances in lattice attacks, algorithms and quantum hardware will continue to shape the landscape of cryptography, yet, the reduction-based foundation of lattice-based schemes remains a basis for confidence in their security. As long as worst-case SVP and SIVP problems remain hard in their respective lattice settings, cryptographic schemes built on these foundations will remain hard to break, and designers can continue to build and iterate on these constructions without reinventing the whole security case each time.

References

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th Annual ACM Symposium on Theory of Computing*, pages 284–293, El Paso, TX, USA, May 4–6, 1997. ACM Press.

[AD17] Martin R. Albrecht and Amit Deo. Large modulus ring-LWE $\geq$ module-LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 267–296, Hong Kong, China, December 3–7, 2017. Springer, Cham, Switzerland.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011: 38th International Colloquium on Automata, Languages and Programming, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415, Zurich, Switzerland, July 4–8, 2011. Springer Berlin Heidelberg, Germany.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108, Philadephia, PA, USA, May 22–24, 1996. ACM Press.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd Annual ACM Symposium on Theory of Computing*, pages 601–610, Crete, Greece, July 6–8, 2001. ACM Press.

[AR04] Dorit Aharonov and Oded Regev. Lattice problems in NP cap coNP. In *45th Annual Symposium on Foundations of Computer Science*, pages 362–371, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[Ban93] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296:625–635, 1993.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325,

Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery.

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *32nd Annual ACM Symposium on Theory of Computing*, pages 435–440, Portland, OR, USA, May 21–23, 2000. ACM Press.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524, Santa Barbara, CA, USA, August 14–18, 2011. Springer Berlin Heidelberg, Germany.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer Berlin Heidelberg, Germany.

[GMSS99] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and J-P Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Information Processing Letters*, 71(2):55–61, 1999.

[GR02] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002. https://arxiv.org/abs/quant-ph/0208112.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *15th Annual ACM Symposium on Theory of Computing*, pages 193–206, Boston, MA, USA, April 25–27, 1983. ACM Press.

[Kho04] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. In *45th Annual Symposium on Foundations of Computer Science*, pages 126–135, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261:515–534, 1982.

[LPR12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. Cryptology ePrint Archive, Report 2012/230, 2012.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. Cryptology ePrint Archive, Report 2013/293, 2013.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[Mic14] Daniele Micciancio. Lecture notes on the dual lattice, 2014. https://cseweb.ucsd.edu/classes/sp14/cse206A-a/lec3.pdf.

[MLK24] Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM). National Institute of Standards and Technology, NIST FIPS PUB 203, U.S. Department of Commerce, August 2024.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[Pei15] Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939, 2015.

[Reg04a] Oded Regev. Lecture notes on $2^{O(n)}$-time algorithm for SVP, 2004. https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/svpalg.pdf.

[Reg04b] Oded Regev. Lecture notes on the CVP, 2004. https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/cvp.pdf.

[Reg04c] Oded Regev. Lecture notes on the dual lattice, 2004. https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/DualLattice.pdf.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[SAB+20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66:181–199, 1994.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.

[vEB81] P. 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. Department, Univ., 1981.